gnuplot 5.4

Un Programma di Plotting Interattivo

Thomas Williams & Colin Kelley

Versione 5.4 organizzata da: Ethan A Merritt e molti altri

Maggiori contributori (in ordine alfabetico):
Christoph Bersch, Hans-Bernhard Bröker,
John Campbell, Robert Cunningham,
David Denholm, Gershon Elber,
Roger Fearick, Carsten Grammes,
Lucas Hart, Lars Hecking, Péter Juhász,
Thomas Koenig, David Kotz,
Ed Kubaitis, Russell Lang, Timothée Lecomte,
Alexander Lehmann, Jérôme Lodewyck,
Alexander Mai, Bastian Märkisch,
Ethan A Merritt, Petr Mikulík,
Daniel Sebald, Carsten Steger, Shigeharu Takeno,
Tom Tkacik, Jos Van der Woude,
James R. Van Zandt, Alex Woo, Johannes Zellner

Mailing list per commenti: `gnuplot-info@lists.sourceforge.net`
Web site e issue tracker: `http://sourceforge.net/projects/gnuplot`

Questo manuale è stato preparato originariamente da Dick Crawford.
Tradotto in italiano da Alice Lopoeta per BiElle s.r.l. `http://www.bielle.it`

Versione 5.4 (Maggio 2021)

Contents

Part I
Gnuplot

Copyright

AUTORI

Software Originale:
Thomas Williams, Colin Kelley.
Aggiunte a Gnuplot 2.0:
Russell Lang, Dave Kotz, John Campbell.
Aggiunte a Gnuplot 3.0:
Gershon Elber e molti altri.
Aggiunte a Gnuplot 4.0 e 5.0:
Vedere la lista dei collaboratori all'inizio di
questo documento.

Introduzione

Gnuplot è un utility grafica portabile a linea di comando per Linux, OS/2, MS Windows, OSX, VMS, e molte altre piattaforme. Il codice sorgente è protetto da copyright ma liberamente distribuibile (cioè, non si deve pagare). È stato originariamente creato per permettere a scienziati e studenti di visualizzare funzioni matematiche e dati in modo interattivo, ma è cresciuto fino a supportare molti usi non interattivi come lo scripting web. È anche usato come plotting engine da applicazioni di terze parti come Octave. Gnuplot è stato supportato e sviluppato attivamente dal 1986.

Gnuplot supporta molti tipi di grafici, sia in 2D che in 3D. Può disegnare usando linee, punti, caselle, contorni, campi vettoriali, superfici e vari testi associati. Supporta anche vari tipi di grafici specializzati.

Gnuplot supporta molti tipi diversi di output: terminali a schermo interattivo (con input dato da mouse e tasti di scelta rapida), output diretto su plotter a penna o stampanti moderne, e output in molti formati di file (eps, emf, fig, jpeg, LaTeX, pdf, png, postscript, ...). Gnuplot è facilmente estensibile per includere nuove modalità di output. Aggiunte recenti includono terminali interattivi basati su wxWidgets (utilizzabili su più piattaforme), e Qt. I grafici utilizzabili con il mouse incorporati su pagine web possono essere generati usando i driver del terminale svg o canvas HTML5.

Il linguaggio di comando di **gnuplot** è sensibile ai caratteri maiuscoli e minuscoli, cioè i comandi e i nomi delle funzioni scritti in minuscolo non sono uguali a quelli scritti in maiuscolo. Tutti i nomi dei comandi possono essere abbreviati, purchè l'abbreviazione non sia ambigua. Qualsiasi numero di comandi può apparire su una linea, separati da punto e virgola (;). Le stringhe possono essere separate da virgolette singole o doppie, anche se esistono delle sottili differenze. Vedere **sintassi (p. 60)** e **virgolette (p. 61)** per maggiori dettagli. Esempio:

```
set title "My First Plot";  plot 'data';  print "all done!"
```

I comandi possono essere estesi su più linee di input terminando ogni linea, tranne l'ultima, con un backslash (\). Il backslash deve essere _l'ultimo_ carattere su ogni linea. L'effetto è come se il backslash e il newline non fossero presenti. Cioè, nessuno spazio bianco è implicito, e nessun commento è terminato. Pertanto, commentare una linea continua commenta l'intero comando (vedere **commenti (p. 33)**). Ma si noti che se si verifica un errore da qualche parte in un comando multilinea, il parser potrebbe non essere in grado di localizzare precisamente dove si trova l'errore e in quel caso non indicherà necessariamente la linea corretta.

In questo documento, le parentesi graffe ({}) denotano argomenti opzionali e una barra verticale (|) separa le scelte che si escludono a vicenda. Le keyword di **Gnuplot** o gli argomenti **help** sono indicati da virgolette o **grassetto** (dove disponibile). Le parentesi angolari (<>) sono usate per segnare i token sostituibili. In molti casi, un valore predefinito del token sarà preso per gli argomenti opzionali se il token è omesso, ma questi casi non sono sempre indicati con parentesi graffe intorno alle parentesi angolari.

Per aiuto integrato su qualsiasi argomento, digitare **help** seguito dal nome dell'argomento o **help ?** per visualizzare un menu degli argomenti disponibili.

È disponibile un ampio set di grafici dimostrativi sulla pagina web `http://www.gnuplot.info/demo/`

Quando viene eseguito dalla linea di comando, gnuplot viene invocato usando la sintassi

```
gnuplot {OPTIONS} file1 file2 ...
```

dove file1, file2, ecc. sono file di input come nel comando **load**. Nei sistemi basati su X11, si può usare

```
gnuplot {X11OPTIONS} {OPTIONS} file1 file2 ...
```

vedere la propria documentazione X11 e **x11 (p. 311)** in questo documento.

Le opzioni interpretate da gnuplot possono trovarsi in qualsiasi punto della linea. I file vengono eseguiti nell'ordine specificato, così come i comandi forniti dall'opzione -e, per esempio

```
gnuplot   file1.in   -e "reset"   file2.in
```

Il filename speciale `"-"` è usato per forzare la lettura da stdin. **Gnuplot** viene chiuso dopo che l'ultimo file è stato elaborato. Se non viene nominato alcun file di caricamento, **Gnuplot** prende l'input interattivo da stdin. Vedere help **batch/interactive (p. 31)** per maggiori dettagli. Le opzioni specifiche a gnuplot possono essere elencate digitando

```
gnuplot --help
```

Vedere **command-line-options (p. 31)** per ulteriori dettagli.

Nelle sessioni con una finestra di grafico interattiva è possibile premere 'h' in qualsiasi punto del grafico per aiuto sulle funzioni **hotkeys** e **mousing**. La sezione **richiesta assistenza** aiuterà a trovare ulteriori informazioni, aiuto e FAQ.

Richiesta assistenza

La home page ufficiale di gnuplot può essere trovata all'indirizzo http://www.gnuplot.info

Prima di chiedere aiuto, controllare il file FAQ.pdf o il sito web qui sopra per la lista delle FAQ (Frequently Asked Questions).

Un'altra risorsa per dell'aiuto con problemi specifici di plotting (non bug) è

```
https://stackoverflow.com/questions/tagged/gnuplot
```

Le segnalazioni di bug e le richieste di funzionalità dovrebbero essere caricate sui tracker all'indirizzo

```
http://sourceforge.net/projects/gnuplot/support
```

Si prega di controllare le segnalazioni precedenti per controllare se il bug che si vuole segnalare è già stato risolto in una versione più recente.

Quando si segnala un bug o si posta una domanda, si prega di includere tutti i dettagli della versione di gnuplot, del tipo di terminale e del sistema operativo. Un breve script autoconclusivo che dimostri il problema è molto utile.

Le istruzioni per iscriversi alla mailing list di gnuplot possono essere trovate tramite il sito web di sviluppo di gnuplot su SourceForge http://sourceforge.net/projects/gnuplot

Si prega di notare che prima di scrivere ad una qualsiasi delle mailing list di gnuplot è necessario essere già iscritto alla lista. Questo aiuta a ridurre la quantità di spam.

L'indirizzo per scrivere ai membri della lista è:

```
gnuplot-info@lists.sourceforge.net
```

La mailing list per chi è interessato alla versione di sviluppo di gnuplot è:

```
gnuplot-beta@lists.sourceforge.net
```

Nuove funzionalità

Funzionalità introdotte nella versione 5.4

Queste sezioni elencano nuovi comandi, stili di grafico, e altre funzionalità introdotte nella versione 5.4.

Supporto per l'aritmetica intera a 64-bit

- Tutta la valutazione delle espressioni e delle funzioni utilizza l'aritmetica a 64 interi se supportata dalla piattaforma.
- L'overflow degli interi viene rilevato e gestito secondo le preferenze dell'utente. Vedere **overflow (p. 188)**.

Griglie voxel

Gnuplot ora supporta operazioni basate su griglie 3D di dati voxel.

- **set vgrid $gridname size N** crea una griglia NxNxN di voxel.
- **set vxrange** [**vxmin:vxmax**] insieme a **set vyrange** e **set vzrange** definiscono quale regione dello spazio occupa la griglia. Questa può essere o meno identica all'intervallo xyz del grafico.
- **voxel(x,y,z)** può essere usato nelle espressioni per leggere o scrivere un voxel individuale.

- **vfill DATA_SOURCE using x:y:z:radius:(<expression>)** agisce in modo analogo a un comando plot, tranne che invece di plottare, incrementa i voxel vicino a ogni punto nei dati di input.
- **vclear $gridname** resetta una griglia di voxel esistente per contenere tutti i valori zero.
- il contenuto attuale di una o più griglie di voxel può essere indicato dai comandi **splot** per assegnare colori o altre proprietà agli elementi del grafico utilizzando la funzione **voxel** negli specificatori d'uso. Vedere la demo **voxel.dem**.
- le griglie di voxel possono anche essere plottate per nome nei comandi **splot** con stili di grafico **dots**, **points**, o **isosurface**. Vedere demo **vplot.dem**.

Nuovi stili di grafici e opzioni di stile

- lo stile di grafico 3D **with polygons** legge le facce di un poligono da un file di dati. Questo può essere usato per creare una superficie o per costruire un oggetto solido. Vedere **with polygons (p. 82)**.
- **splot $voxelgrid with {dots|points}** segna tutti i voxel il cui valore è superiore a un livello di soglia richiesto.
- **splot $voxelgrid with isosurface** crea una superficie 3D tassellata che racchiude voxel al di sopra di un livello di soglia richiesto. Vedere **isosurface (p. 89)**.
- I valori della griglia di voxel possono essere referenziati negli specificatori **using** per i grafici 3D.
- **set spiderplot** seleziona una nuova modalità di plotting che permette la creazione di grafici spider (noti anche come grafici radar). Questi sono essenzialmente grafici ad assi paralleli dove gli assi sono disposti radialmente piuttosto che verticalmente. Vedere **spiderplot (p. 83)**, **set style spiderplot (p. 213)**, **set paxis (p. 195)**.
- Lo stile di grafico **with circles** può essere usato nei grafici 3D .
- Lo stile di grafico **with boxes** può essere usato nei grafici 3D.
- Lo stile di grafico 2D **with arrows** è identico a **with vectors** eccetto che ogni freccia è specificata usando x:y:length:angle invece che x:y:xdelta:ydelta
- splot (grafico 3D) FOO con colore di riempimento pm3d <colorspec>
- Le superfici pm3d possono avere stile di riempimento individuale e colore di riempimento superiore/inferiore separati
- L'opzione pm3d **noclipcb** fa sì che i quadrangloli con il colore della palette al di fuori di cbrange vengano saltati piuttosto che essere disegnati con il colore agganciato a cbmin o cbmax.
- Tipi di linee di contorno personalizzate. Vedere **set cntrparam (p. 144)**.

Nuovi filtri di pre-elaborazione dei dati

- **smooth zsort** ordina punti 2D in base ai valori in una terza colonna. Vedere **smooth zsort (p. 120)**.

Nuovi comandi e opzioni di comando

- Comandi della griglia di voxel. Vedere **set vgrid (p. 221)**, **set vxrange (p. 223)**, **vclear (p. 246)**, **vfill (p. 247)**, e **voxel (p. 40)**.
- Nuove opzioni per mostrare i piani xy, xz e yz nei grafici 3D. Vedere **set walls (p. 223)**, **set grid vertical (p. 161)**.
- **set table separator {tab|comma|"char"}** può essere usato per creare file csv. Vedere **plot with table (p. 215)**.
- Nuove opzioni **set view projection {xy|xz|yz}** regolano gli angoli di vista, i tic (tacche) degli assi e il posizionamento dell'etichetta per generare una proiezione 2D di un splot 3D. **set view projection xy** equivale a **set view map**.

- **set rgbmax** <**value**> controlla l'interpretazione dei valori RGB in input.
- La dimensione dell'array può essere implicita se è presente un inizializzatore, per esempio **Array A = [1,2,3]**.
- Il ritaglio radiale opzionale dei segmenti di linea in modalità polare. Vedere **set clip (p. 143)**.
- Possono essere aggiunte linee extra per personalizzare la chiave sostituendo **keyentry** al posto del nome di un file o di una funzione nei comandi **plot** e **splot**. Questo crea una linea nella chiave senza generare un grafico corrispondente. Vedere **keyentry (p. 168)**.
- Traslazione specificata dall'utente delle coordinate del mouse. (SPERIMENTALE). Vedere map_projection demo.
- **set datafile columnheaders** fa sì che la prima linea di input sia letta come stringa piuttosto che come valori di dati . Equivale a **set key autotitle columnheader** eccetto che non influisce sulla generazione di key entry. Se questa opzione è attiva il comando **stats** genererà un array di stringhe contenenti le intestazioni di colonna trovate.
- È possibile impostare più stili di caselle di testo. Vedere **set style textbox (p. 214)**.

Nuovi terminali e opzioni di terminali

- Il terminale **pcl5** è stato esteso per supportare stampanti PCL5e/PCL5c e molte funzionalità moderne di gnuplot.
- Il terminale **pstricks** è stato esteso per supportare molte funzionalità moderne di gnuplot, compresi i colori RGB e la trasparenza, poligoni, e caselle.
- Nuovo terminale **pict2e** per utilizzare l'ambiente LaTeX2e pict2e. Sostituisce direttamente i vecchi terminali **latex**, **emtex**, **eepic**, e **tpic**, che non sono più realizzati di default.
- Il terminale **texdraw** è stato esteso per supportare il testo ad angoli arbitrari, lo spessore di linea variabile, linee tratteggiate v5, e caselle e poligoni pieni. Ora può essere usato anche con plain TeX.
- La variante Direct2D precedentemente sperimentale del terminale di **windows** sostituisce le varianti GDI and GDI+. Ora supporta la stampa con D2D e font a colori.
- Il terminale **pm** OS/2 è stato aggiornato per supportare, per esempio, uft8, testo in grassetto e corsivo, e linee tratteggiate. (Da 5.2.7).
- I terminali DOS **dospc** e **svga** sono stati aggiornati e adesso supportano l'input interattivo da tastiera e mouse (solo svga).

Pixmaps come oggetti

- **set pixmap** permette di importare un'immagine in formato standard (png jpeg gif) come una pixmap che può essere posizionata ovunque in un grafico o sulla pagina. A differenza del plotting **with image**, gli oggetti pixmap mantengono le loro proporzioni e dimensioni originali indipendentemente dal ridimensionamento o dalla rotazione degli assi. Vedere **pixmap (p. 196)**.

Formati di tempo settimana-data

La pandemia di Covid-19 del 2020/2021 ha generato un maggiore interesse nel plotting di dati epidemiologici, che sono spesso tabulati utilizzando una convenzione di segnalazione "settimana-data". Questo ha rivelato carenze nel supporto di gnuplot per questa convenzione, compresi gli errori nei formati di tempo %W e %U. Questi formati funzionavano in modo errato prima della versione 5.4.2.

- Il formato dello specificatore di tempo %W è stato adattato allo standard ISO 8601.
- Il formato dello specificatore di tempo %U è stato adattato allo standard CDC/MMWR.

- La nuova funzione **tm_week(time, std)** riporta la settimana dell'anno standard ISO o CDC.
- La nuova funzione **weekdate_iso(year, week, day)** converte la data della settimana standard ISO nella data del calendario.
- La nuova funzione **weekdate_cdc(year, week, day)** converte la data della settimana standard CDC nella data del calendario.

Altre nuove funzionalità

- Modalità di testo avanzato accetta \U+xxxx (xxxx è un esadecimale di 4 o 5 caratteri) come se rappresentasse un punto di codice Unicode che viene convertito nella corrispondente sequenza di byte UTF-8 in output.
- La sequenza di caratteri $# in uno specificatore **using** valuta il numero totale di colonne disponibili nella linea di dati corrente. Per esempio "plot FOO using 0:(column($# - 1))" plotta il penultimo campo di ogni riga.
- L'input di ora/data riconosce il formato %p per gestire l'ambito am/pm
- I titoli dei grafici sono valutati *dopo* il plotting, invece che prima. Questo permette al titolo di fare riferimento a quantità calcolate durante il plotting.
- Funzioni di Bessel modificate incorporate (besi0 besi1 besin)
- Funzioni di Bessel di ordine N di 1° e 2° tipo incorporate (besjn besyn)

Cambiamenti

- **pm3d filled area quadrangles** (quadrangoli ad area riempita pm3d) sono ritagliati sull'attuale zrange. Questo influenza le superfici pm3d e anche le facce di scatole 3D, poligoni, etc.
- Sintassi revisionata per lo stile di grafico 'with parallelaxes'. Vedere **parallel (p. 81)**. Gli stili istogramma, parallelaxis, e spiderplot ora usano una sintassi simile che può iterare sugli elementi del grafico: **plot for [column=1:N] DATA using column**
- Il campionamento generato dallo pseudofile '+' è influenzato da **set trange**.
- Gli offset da **set offsets** sono applicati solo agli assi autoscalati. La documentazione lo ha sempre detto, ma non è stato applicato in modo sistematico.
- I valori immaginari restituiti dallo specificatore d'uso di un grafico 2D vengono trattati come valori indefiniti (NaN) piuttosto che (valori) reali. Questo è sempre stato vero per grafici di funzioni e grafici di dati 3D. Per esempio, i due grafici seguenti sono equivalenti. plot [-1:1] sqrt(x); plot [-1:1] '+' using 1:(sqrt($1)
- Il comando **set fontpath** è deprecato. Il percorso di ricerca dei font da incorporare nell'output del terminale postscript è stato rivisto.

Funzionalità introdotte nella versione 5.2

Nuovi stili di grafici e opzioni di stile

- Stile di grafico 3D **with zerrorfill**. Vedere **zerrorfill (p. 89)**, **fenceplots (p. 89)** e `demo zerror.`
- Grafici beeswarm. Vedere **set jitter (p. 165)**, **beeswarm (p. 63)** e `demo grafico beeswarm.`
- Il simbolo usato per i singoli punti in un grafico può essere controllato dai valori dei dati. (vedere **pointtype variable (p. 82)**)

Nuovi filtri di pre-elaborazione dei dati

- Frequenza normalizzata di occorrenza in un set di dati (vedere **smooth fnormal (p. 119)**)
- Binning automatizzato dei dati (vedere **bins (p. 114)**)

Miglioramenti ed estensioni alla modalità polare

- Coordinate polari possono essere usate nelle definizioni di etichette, frecce e oggetti
- **set [m]ttics** colloca ticmark ed etichette sul perimetro di un grafico polare. Vedere `polar axis and ticlabels demo`
- **set rlabel (p. 204)** colloca un'etichetta sopra l'asse r
- **rrange (p. 204)** invertito (cioè: set rrange [90:0]) permette l'uso di coordinate celesti orizzontali. Vedere `solar path demo`
- **set border polar (p. 140)** disegna una linea continua intorno al perimetro di un grafico polare
- **set theta (p. 216)** controlla la posizione di theta = 0 intorno il perimetro di un grafico polare e il senso (orario o antiorario) di aumento del theta

Sistemi di coordinate non lineari

- A qualsiasi asse di un grafico possono essere assegnate un paio di funzioni, possibilmente non lineari, che descrivano la mappatura in avanti e indietro in un intervallo lineare (vedere **set nonlinear (p. 183)**) `Nonlinear x/y axis demo`
- Il comando **set logscale** è stato reimplementato come un caso speciale di asse non lineare dove le funzioni associate sono log(x) and exp(x).

Nuovi comandi e opzioni di comando

- All'interno della clausola tra parentesi di un interazione, **continue** va immediatamente all'iterazione successiva, **break** esce immediatamente dall'iterazione
- **toggle {<plotno> | "plottitle" | all}"** attiva o disattiva interattivamente la visualizzazione di un elemento del grafico corrente (vedere **toggle (p. 245)**)
- **save fit** sostituisce il comando deprecato **update**
- **set table "outfile.name" append** aggiungerà i successivi grafici tabulati a un file di testo esistente piuttosto che sostituire il suo contenuto
- **set pm3d lighting** descrive un modello di illuminazione con evidenziazione speculare (vedere **lighting (p. 198)**)
- **set minussign** dice a gnuplot di usare un simbolo speciale, nella codifica corrente, per sostituire il carattere ascii '-' nei numeri negativi
- **set micro** dice a gnuplot di usare un simbolo speciale, nella codifica corrente, per sostituire il carattere ascii 'u' per il prefisso di notazione scientifica "micro". I simboli tipografici speciali per micro e per il segno meno sono usati solamente nelle etichette dei tic degli assi e nelle stringhe create esplicitamente con gprintf(). La sequenza di byte usata per rappresentare questi caratteri dipende dalla codifica corrente.

Nuovi dati di tipo "array"

- Questa versione di gnuplot introduce un nuovo tipo di dati **array name[size]**. Un array deve essere dichiarato prima dell'uso. Ogni elemento array A[i] potrebbe essere una stringa, un intero, un numero

reale, o un valore complesso. Un singolo array potrebbe contenere elementi di diverso tipo. L'operatore cardinale |A| restituisce la dimensione dell'array A. Vedere **array (p. 46)**.

Nuovi terminali e opzioni di terminali

- Vedere **sixelgd (p. 298)** per la descrizione di un nuovo terminale che supporta grafici interleaving, con le linee di comando che li hanno generati, se gnuplot viene eseguito dentro un emulatore di terminale compatibile con vt340
- Il terminale **domterm (p. 261)** supporta i grafici interleaving, con le linee di comando che li hanno generati, se gnuplot è eseguito dentro un emulatore di terminale svg-aware
- Il terminale **windows (p. 305)** supporta il salvataggio del grafico (graph) corrente su un file bitmap
- La finestra del grafico (graph) del terminale **windows** può essere agganciata alla finestra di testo wgnuplot
- Nuovo (sperimentale) motore Direct2D/DirectWrite per il terminale **windows**
- Il terminale **wxt** supporta l'esportazione a un file EMF o stampante su Windows
- Il terminale **dumb** supporta i colori ANSI per le linee e l'area di riempimento
- Il terminale **tkcanvas** è stato riscritto per supportare molte più moderne funzionalità di gnuplot, e anche nuove lingue (Da 5.0.3)

Altre nuove funzionalità

- Un ulteriore angolo di rotazione **azimuth** influisce sull'orientamento dei grafici 3D. Questo può essere impostato dalla linea di comando (vedere **set view azimuth (p. 222)**) o trascinando con il tasto destro del mouse.
- gnuplot in esecuzione sotto Windows può interpretare gli script di input Unicode (BMP) convertendoli nella codifica corrente da **set encoding**, incluso UTF-8
- Alle caselle di testo possono essere assegnati un bordo colorato e un colore di sfondo (vedere **set style textbox (p. 214)**)
- Legende per grafici personalizzate (vedere **plot title (p. 129)**, **set key (p. 166)**, **multiple keys (p. 170)**)
- Uno specificatore dell'intervallo di campionamento per plottare con pseudofile '+' può includere un intervallo di campionamento. Per esempio: plot sample [t=0:100:10] '+' using (t):(1):(label[t]) with labels
- Lo pseudo-file '++' genera campioni sugli assi u e v, invece che su x e y. Questo permette il posizionamento di molteplici superfici parametriche in 3D che occupano regioni diverse del piano cartesiano. Vedere **sampling.dem**.
- I nuovi descrittori di formati tH tM tS gestiscono tempi relativi (lunghezze di intervallo). Vedere **time_specifiers (p. 159)**.
- ^R avvia una ricerca nella cronologia per la readline incorporata che è usata anche su Windows, vedere **command-line-editing (p. 33)**.
- Rivisto il supporto alla stampa su Windows usando **set output "PRN"**, vedere **windows printing (p. 307)**.

Funzionalità introdotte nella versione 5.0

- Il pattern punto-trattino di una linea adesso può essere stabilito indipendentemente da altre proprietà della linea. Vedere **dashtype (p. 52)**, **set dashtype (p. 148)**, **set linetype (p. 173)**

- La sequenza di colori predefinita usata per gli elementi successivi di un grafico è più facilmente distinguibile dagli utenti con difetti di visione dei colori. La sequenza dei colori è sotto il controllo dell'utente (vedere **set colorsequence (p. 143)**). Questo meccanismo può essere usato anche per generare grafici monocromatici (vedere **set monochrome (p. 177)**). In versioni precedenti di gnuplot **monochrome** poteva essere selezionato solamente quando si cambiava il terminale in uso via **set terminal**.
- Nuovi stili di grafico **with parallelaxes**, **with table**, e contorni etichettati.
- Nuovo filtro di pre-elaborazione dei dati per spline cubiche monotoniche (vedere **smooth mcsplines (p. 118)**)
- Il markup del testo ora supporta l'impostazione dei font in grassetto e corsivo oltre che pedice (subscript), apice (superscript), dimensione del font e altre proprietà precedentemente disponibili. La modalità di testo avanzato è ora abilitata di default. Vedere **enhanced text (p. 35)**. Gli elementi di testo possono essere racchiusi in una casella (vedere **set style textbox (p. 214)**).
- I terminali interattivi supportano etichette ipertestuali che appaiono solo quando il muose passa sopra il punto di ancoraggio dell'etichetta.
- Nuovi sistemi di coordinate (Degrees, Minutes, Seconds). Vedere **set xtics geographic (p. 232)**.
- Il formato predefinito per le etichette degli assi è "% h" ("$%h$" per termilali LaTeX). Questo formato è come il formato %g C standard, eccetto che il termine esponenziale, se presente, è scritto usando l'apice. Per esempio: 1.2 x 10^5 invece che 1.2E05.
- Gli script di comando potrebbero posizionare dati in-line in un blocco di dati denominato per plotting ripetuti. Vedere **inline data (p. 48)**.
- Supporto per il canale Alpha a 32 bit + colore RGB #AARRGGBB. Vedere **colorspec (p. 50)**.
- Supporto per lo spazio di colore HSV tramite una funzione di traduzione hsv2rgb(H,S,V)
- Gli assi secondari (x2, y2) possono essere agganciati agli assi primari tramite una funzione di mappatura. Nel caso più semplice questo garantisce che gli intervalli degli assi primari e secondari siano identiche. Nel caso generale permette di definire un asse non lineare, qualcosa che precedentemente era possibile solo per il log scaling. Vedere **set link (p. 174)**.
- Ogni funzione in un comando plot può essere preceduta da un intervallo di campionamento. Questo non influenza l'intervallo complessivo del grafico, solo l'intervallo sul quale viene campionata questa funzione. Vedere **plot (p. 107)** and **piecewise.dem**.
- Se la libreria esterna libcerf è disponibile, essa viene usata per fornire routine matematiche complesse cerf, cdawson, erfi, faddeeva, e il profilo VP Voigt (x,sigma,gamma).
- Il comando **import** allega un nome di funzione definito dall'utente a una funzione fornita da un oggetto esterno condiviso (il supporto dipende dal sistema operativo). Un'intestazione di template, un sorgente di esempio e dei make file per creare un oggetto esterno condiviso adeguato sono forniti nella collezione demo.
- Comandi precedenti nella cronologia di una sessione interattiva possono essere eseguiti di nuovo tramite numero. Per esempio, **history !5** eseguirà di nuovo il comando numerato con 5 nella **history** list (cronologia).
- Operatori bit-shift >> e <<.
- L'invocazione della shell di gnuplot può passare i parametri a uno script di gnuplot. gnuplot -c scriptfile.gp ARG1 ARG2 ARG3 ...

Differenze tra le versioni 4 e 5

Alcuni cambiamenti introdotti nella versione 5 possono indurre alcuni script scritti per versioni precedenti di gnuplot a comportarsi in modo diverso.

* Rivista gestione dei dati di input contenenti NaN, numero inconsitente di colonne di dati, o altri contenuti inaspettati. Vedere Note sotto **missing (p. 149)** per esempi e figure.

* Coordinate temporali sono memorizzate internamente come il numero di secondi relativi all'epoca unix standard 1-Gen-1970. Versioni precedenti di gnuplot usavano un'epoca interna diversa (1-Gen-2000). Questo cambiamento risolve le inconsistenze presenti ogni volta che il tempo in secondi era generato esternamente. La convenzione dell'epoca usata da una particolare installazione di gnuplot può essere determinata usando il comando **print strftime("%F",0)**. Il tempo adesso è memorizzato con una precisione al millisecondo.

* La funzione **timecolumn(N,"timeformat")** adesso ha 2 parametri. Poichè il nuovo secondo parametro non è associato a un asse di dati particolare, questo permette di usare la funzione **timecolumn** per leggere dati temporali per motivi diversi dallo specificare la coordinata x o y. Questa funzionalità sostituisce la sequenza di comandi **set xdata time; set timefmt "timeformat"**. Permette di unire su un unico grafico dati temporali provenienti da più file con formati diversi.

* La keyword **reverse** del comando **set [axis]range** influisce solo l'autoscaling. Non inverte nè altera in altro modo il significato di un comando, come ad esempio **set xrange [0:1]**. Se si vuole invertire la direzione dell'asse x, in tal caso, è necessario usare **set xrange [1:0]**.

* Il comando **call** fornisce una serie di variabili ARGC, ARG0, ..., ARG9. ARG0 contiene il nome del file script in esecuzione. Da ARG1 fino a ARG9 sono variabili stringa e quindi possono essere riferite direttamente o espanse come macro, per esempio: @ARG1. I contenuti di ARG0 ... ARG9 possono essere accessibili in alternativa come elementi array ARGV[0] ... ARGV[ARGC]. Una vecchia convenzione di gnuplot di fare riferimento ai parametri di chiamata come token $0 ... $9 è deprecata.

* La larghezza di banda facoltativa per l'opzione di smoothing (livellamento) della densità del kernel è presa da una keyword piuttosto che da una colonna di dati. Vedere **smooth kdensity (p. 119)**.

Sintassi deprecate

Gnuplot versione 4 ha deprecato alcune sintassi usate nelle versioni precedenti ma ha fornito un'opzione di configurazione che permette la retrocompatibilità. Il supporto per la vecchia sintassi è stato rimosso.

Deprecata nella versione 4 e rimossa nella versione 5:

```
set title "Old" 0,-1
set data linespoints
plot 'file' thru f(x)
plot 1 2 4              # linea orizzontale a y=1
update
```

Equivalente attuale:

```
TITLE = "New"
set title TITLE offset char 0, char -1
set style data linespoints
plot 'file' using 1:(f(column(2)))
plot 1 linetype 2 pointtype 4
save fit "filename"
```

Deprecata nella versione 5

```
if (defined(VARNAME)) ...
set style increment user
call 'script' 1.23 ABC
   (in script:  print $0, "$1", "number of args = $#")
set fontpath
set clabel
fit control variables FIT_*
```

Equivalente attuale:

```
if (exists("VARNAME")) ...
```

```
set linetype
call 'script' 1.23 "ABC"
   (in script:  print ARG1, ARG2, "number of args = ", ARGC
set cntrlabel
set fit <option> <value>
```

Deprecata nella versione 5.4

```
# uso di un file contenente `reread` per eseguire l'iterazione
N = 0;  load "file-containing-reread";
file content:
    N = N+1
    plot func(N,x)
    pause -1
    if (N<5) reread
```

Equivalente attuale

```
do for [N=1:5] {
    plot func(N, x)
    pause -1
}
```

Demo e Esempi Online

La distribuzione di **gnuplot** contiene una raccolta di esempi nella directory **demo**. È possibile sfogliare le versioni on-line di questi esempi prodotti dai terminali png, svg e canvas su http://gnuplot.info/demos

I comandi che generano ciascuna demo di un grafico sono mostrati a fianco del grafico, e il corrispondente script di gnuplot può essere scaricato come modello per generare grafici simili.

Operazioni Batch/Interattive

Gnuplot può essere eseguito sia in modalità batch che interattiva, e le due possono anche essere usate insieme su molti sistemi.

Si presume che gli argomenti della linea di comando siano opzioni del programma (vedere **opzioni della linea di comando (p. 31)**) o nomi di file che contengono comandi **gnuplot**. Ogni file o stringa di comando sarà eseguito nell'ordine specificato. Il nome di file speciale "-" indica che i comandi devono essere letti da stdin. **Gnuplot** si chiude dopo che l'ultimo file è stato processato. Se non vengono specificati file di caricamento o stringhe di comando, **gnuplot** accetta un input interattivo da stdin.

Opzioni della linea di comando

Gnuplot accetta le seguenti opzioni sulla linea di comando

```
-V, --version
-h, --help
-p  --persist
-d  --default-settings
-s  --slow
-e  "command1; command2; ..."
-c  scriptfile ARG1 ARG2 ...
```

-p dice al programma di non chiudere le finestre interattive rimanenti del grafico quando il programma esce.

-d dice al programma di non eseguire alcuna inizializzazione privata o di sistema (vedere **inizializzazione (p. 57)**).

-s dice al programma di aspettare l'inizializzazione lenta dei font all'avvio. Altrimenti stampa un errore e continua con metriche dei font sbagliate.

-e "comando" dice a gnuplot di eseguire quel particolare comando prima di continuare.

-c equivale a -e "call scriptfile ARG1 ARG2 ...". Vedere **call (p. 91)**.

Esempi

Per lanciare una sessione interattiva:

```
gnuplot
```

Per lanciare una sessione batch usando due file di comando "input1" e "input2":

```
gnuplot input1 input2
```

Per lanciare una sessione interattiva dopo un file di inizializzazione "header" e seguito da un altro file di comando "trailer":

```
gnuplot header - trailer
```

Per dare comandi a **gnuplot** direttamente nella linea di comando, usando l'opzione "-persist" in modo che il grafico rimanga sullo schermo:

```
gnuplot -persist -e "set title 'Sine curve'; plot sin(x)"
```

Per impostare le variabili definite dall'utente a e s prima di eseguire i comandi da un file:

```
gnuplot -e "a=2; s='file.png'" input.gpl
```

Dimensione del canvas

Questa documentazione usa il termine "canvas" per identificare l'intera area di disegno disponibile per posizionare il grafico e gli elementi associati come etichette, titoli, key, etc. NB: Per informazioni riguardo i terminali canvas HTML5 vedere **set term canvas (p. 255)**.

In versioni precedenti di gnuplot, alcuni tipi di terminale usavano i valori presi da **set size** per controllare anche le dimensioni dell'output del canvas; altri invece no. L'uso di 'set size' per questo scopo è stato deprecato nella versione 4. Ora quasi tutti i terminali si comportano come di seguito:

set term <terminal_type> size <XX>, <YY> controlla la dimensione del file output, o "canvas". Per default, il grafico riempirà questo canvas.

set size <XX>, <YY> scala il grafico stesso rispetto alla dimensione del canvas. Valori di scala inferiori a 1 faranno sì che il grafico non riempia l'intero canvas. Valori di scala superiori a 1 faranno sì che solo una parte del grafico rientri nel canvas. Bisogna tenere presente che impostare valori di scala più grandi di 1 può causare problemi.

Esempio:

```
set size 0.5, 0.5
set term png size 600, 400
set output "figure.png"
plot "data" with lines
```

Questi comandi producono un file di output "figure.png" che è largo 600 pixels e alto 400 pixels. Il grafico riempirà il quadrante in basso a sinistra di questo canvas. Questo è coerente con il modo in cui la modalità multiplot ha sempre funzionato.

Editing della linea di comando

L'editing della linea di comando e la cronologia dei comandi sono supportati utilizzando una libreria readline gnu esterna, una libreria libedit BSD esterna, oppure un equivalente incorporato. Questa scelta è un'opzione di configurazione nel momento in cui gnuplot è sviluppato.

I comandi di editing della versione incorporata sono dati di seguito. Si prega di notare che l'azione del tasto DEL è dipendente dal sistema. Le librerie readline gnu readline gnu e libedit BSD hanno la propria documentazione.

Command-line Editing Commands	
Carattere	Funzione
	Line Editing
^B	si sposta indietro di un solo carattere.
^F	si sposta in avanti di un solo carattere.
^A	si sposta all'inizio della linea.
^E	si sposta alla fine della linea.
^H	cancella il carattere precedente.
DEL	cancella il carattere corrente.
^D	cancella il carattere corrente. manda alla fine del file se la linea è vuota.
^K	cancella dalla posizione attuale fino alla fine della linea.
^L	riscrive la linea nel caso venga cancellata.
^U	cancella tutta la linea.
^W	cancella la parola precedente.
^V	impedisce l'interpretazione del seguente tasto come comando di modifica.
TAB	completa il nome del file.
	Cronologia
^P	si sposta indietro nella cronologia.
^N	si sposta in avanti nella cronologia.
^R	inizia una ricerca a ritroso.

Commenti

Il carattere di commento # potrebbe apparire quasi ovunque in una linea di comando, e **gnuplot** ignorerà il resto della linea. Un # non ha questo effetto se è in una stringa virgolettata. Si noti che se una linea commentata finisce con '\', allora anche la linea seguente viene trattata come parte del commento.

Vedere anche **set datafile commentschars (p. 151)** per specificare un carattere di commento per i file di dati.

Coordinate

I comandi **set arrow**, **set key**, **set label** e **set object** permettono di disegnare qualcosa in una posizione arbitraria sul grafico (graph). Questa posizione è specificata dalla sintassi:

```
{<system>} <x>, {<system>} <y> {,{<system>} <z>}
```

Ogni <system> può essere o **first**, **second**, **polar**, **graph**, **screen**, o **character**.

first colloca la coordinata x, y, o z nel sistema definito dagli assi sinistro e inferiore; **second** la colloca nel sistema definito dagli assi x2,y2 (superiore e destro); **graph** specifica l'area all'interno degli assi — 0,0 è in basso a sinistra e 1,1 è in alto a destra (per splot, 0,0,0 è in basso a sinistra dell'area di plotting; usare z negativa per arrivare alla base — vedere **set xyplane (p. 233)**); **screen** specifica l'area dello schermo (l'intera area — non solo la parte selezionata da **set size**), con 0,0 in basso a sinistra e 1,1 in alto a destra. Le

coordinate **character** sono usate principalmente per gli offset, non per le posizioni assolute. La dimensione verticale e orizzontale del **character** dipende dal font.

polar fa sì che i primi due valori vengano interpretati come angolo theta e raggio r invece che x e y. Ciò potrebbe essere usato, per esempio, per posizionare delle etichette su un grafico 2D in coordinate polari o un grafico 3D in coordinate cilindriche.

Se non viene specificato il sistema di coordinate per x, viene usato **first**. Se non viene specificato il sistema per y, viene adottato quello per x.

In alcuni casi, la coordinata fornita non è una posizione assoluta ma bensì un valore relativo (ad es., la seconda posizione in **set arrow** ... **rto**). Nella maggior parte dei casi, il valore fornito serve come differenza rispetto alla prima posizione. Se la coordinata data appartiene a un asse a scala logaritmica, un valore relativo viene interpretato come moltiplicatore. Per esempio,

```
set logscale x
set arrow 100,5 rto 10,2
```

plotta una freccia dalla posizione 100,5 alla posizione 1000,7 dato che l'asse x è logaritmico mentre l'asse y è lineare.

Se uno (o più) assi sono una serie temporale, la coordinata appropriata dovrebbe essere data come una stringa temporale tra virgolette in base alla stringa di formato **timefmt**. Vedere **set xdata (p. 224)** e **set timefmt (p. 218)**. **Gnuplot** accetterà anche un'espressione intera, che sarà interpretata come secondi relativi al 1 gennaio 1970.

Stringhe di dati

I file di dati possono contenere dati stringa che consistono o in una stringa arbitraria di caratteri stampabili che non contengono spazi bianchi o una stringa arbitraria di caratteri, che potrebbe includere spazi bianchi, delimitata da doppie virgolette. La seguente linea da un file di dati viene interpretata come composta da quattro colonne, con un campo di testo nella colonna 3:

```
1.000 2.000 "Third column is all of this text" 4.00
```

I campi di testo possono essere posizionati sia in un grafico 2-D che 3-D usando i seguenti comandi:

```
plot 'datafile' using 1:2:4 with labels
splot 'datafile' using 1:2:3:4 with labels
```

Una colonna di dati di testo può anche essere usata per etichettare i tic lungo uno o più assi del grafico. L'esempio di seguito plotta una linea attraverso una serie di punti con coordinate (X,Y) prese dalle colonne 3 e 4 del file di dati di input. Tuttavia, invece di generare tic spaziati regolarmente lungo l'asse x etichettato numericamente, gnuplot posizionerà un tic lungo l'asse x in corrispondenza della coordinata X di ogni punto ed etichetterà il tic con il testo preso dalla colonna 1 del file di dati di input.

```
set xtics
plot 'datafile' using 3:4:xticlabels(1) with linespoints
```

Esiste anche un'opzione che interpreta la prima entry in una colonna di dati di input (cioè l'intestazione della colonna) come un campo di testo, e la usa come titolo key per i dati plottati da quella colonna. L'esempio fornito di seguito usa la prima entry della colonna 2 per generare un titolo nella casella key, mentre elabora il resto delle colonne 2 e 4 per disegnare la linea richiesta:

```
plot 'datafile' using 1:(f($2)/$4) with lines title columnhead(2)
```

Un altro esempio:

```
plot for [i=2:6] 'datafile' using i title "Results for ".columnhead(i)
```

Questo uso delle intestazioni di colonna è automatizzato da **set datafile columnheaders** o **set key autotitle columnhead**. Vedere **labels (p. 79)**, **using xticlabels (p. 123)**, **plot title (p. 129)**, **using (p. 121)**, **key autotitle (p. 168)**.

Modalità di testo avanzato

Molti tipi di terminali supportano una modalità di testo avanzato nella quale informazioni sulla formattazione aggiuntive sono incorporate nella stringa di testo. Per esempio, "x^2" scriverà x alla seconda come siamo abituati a vederlo, ovvero con 2 all'apice. Questa modalità è selezionata di default quando si imposta il terminale, ma può essere attivata o disattivata in seguito usando "set termoption [no]enhanced", o segnando stringhe individuali come "set label 'x_2' noenhanced".

Codici di controllo del testo avanzato			
Comando	Esempio	Spiegazione	
^	a^x	a^x	apice
_	a_x	a_x	pedice
@	a@^b_{cd}	a^b_{cd}	casella fantasma (non occupa nessuna larghezza)
&	d&{space}b	d␣␣␣␣␣b	inserisce uno spazio di una larghezza specifica
~	~a{.8-}	$\tilde{a}$	sovrestampa '-' su 'a', sollevato di .8 volte la dimensione attuale del carattere
	{/Times abc}	abc	stampa abc nel font Times nella dimensione attuale
	{/Times*2 abc}	abc	stampa abc nel font Times al doppio della dimensione attuale
	{/Times:Italic abc}	*abc*	stampa abc nel font Times con lo stile corsivo
	{/Arial:Bold=20 abc}	**abc**	stampa abc in grassetto nel font Arial con dimensione 20
\U+	\U+221E	∞	punto Unicode U+221E INFINITY

I caratteri di controllo della marcatura (markup) agiscono sul singolo carattere o sulla clausola tra parentesi seguenti. La clausola tra parentesi potrebbe contenere una stringa di caratteri con nessun'altra marcatura, per esempio 2^{10}, o potrebbe contenere delle marcature aggiuntive che cambiano le proprietà del font. Gli specificatori del font DEVONO essere preceduti da un carattere '/' che segue immediatamente la parentesi '{' di apertura. Se il nome di un font contiene degli spazi deve essere racchiusto tra virgolette singole o doppie.

Esempi: Il primo esempio dimostra l'inserimento di una clausola tra parentesi dentro un'altra per generare una A in grassetto con un pedice i in corsivo, il tutto nel font attuale. Se la clausola introdotta da :Normal fosse omessa, il pedice sarebbe sia in corsivo che in grassetto. Il secondo esempio mostra la stessa marcatura applicata al font "Times New Roman" con dimensione in punti 20.

```
{/:Bold A_{/:Normal{/:Italic i}}}
{/"Times New Roman":Bold=20 A_{/:Normal{/:Italic i}}}
```

La casella fantasma è utile per a@^b_c per allineare gli apici e i pedici ma non funziona bene per applicare l'accento su una lettera. Per quest'ultimo, è molto meglio usare una codifica (per esempio: iso_8859_1 o utf8) che contiene una grande varietà di lettere con accenti o altri segni diatrici. Vedere **set encoding (p. 154)**. Poichè la casella non ha spaziatura, è ragionevole mettere l'abbreviazione dell'apice e del pedice nella casella (cioè, dopo il @).

Uno spazio uguale alla lunghezza di una stringa può essere inserito usando il carattere '&'. Quindi

```
'abc&{def}ghi'
```

produrrebbe

```
'abc   ghi'.
```

Il carattere '~ ' fa sì che il carattere o il testo tra parentesi successivo sia sovrastampato dal carattere o dal testo tra parentesi successivo. Il secondo testo sarà centrato orizzontalmente sul primo. Quindi '~ a/' si tradurrà in una 'a' con uno slash che la attraversa. È possibile anche spostare il secondo testo verticalmente facendolo precedere da un numero, che definirà la frazione dell'attuale dimensione del font di cui il testo sarà alzato o abbassato. In questo caso il numero e il testo devono essere racchiusi tra parentesi perchè è necessario più di un solo carattere. Se il testo sovrastampato inizia con un numero, bisogna mettere uno

spazio tra l'offset verticale e il testo ('~ {abc}{.5 000}'); altrimenti non è necessario alcuno spazio ('~ {abc}{.5 — }'). È possibile cambiare il font di una oppure di entrambe le stringhe ('~ a{.5 /*.2 o}' — una 'a' con una 'o' grande un quinto sopra — e lo spazio tra il numero e lo slash è necessario), ma non lo si può cambiare dopo l'inizio della stringa. Né si può usare qualsiasi altra sintassi speciale all'interno di entrambe le stringhe. È possibile, naturalmente, usare i caratteri di controllo eseguendo l'escape (vedere sotto), come per esempio '~ a{\^}'

Si può eseguire l'escape di caratteri di controllo usando \, ad es., \\, \{, e così via. Vedere **sequenze di escape (p. 36)** di seguito.

Si noti che le stringhe racchiuse tra virgolette doppie sono analizzate diversamente da quelle racchiuse tra virgolette singole. La differenza principale è che i backslash potrebbero aver bisogno di essere raddoppiati quando si trovano in stringhe tra virgolette doppie.

Il file "ps_guide.ps" nella sottocartella /docs/psdoc della distribuzione sorgente di gnuplot contiene altri esempi della sintassi avanzato, così come la demo **enhanced_utf8.dem**

Sequenze di escape

Il carattere backslash \ è usato per eseguire l'escape di codici di carattere a singolo byte o punti di accesso Unicode.

La forma \ooo (dove ooo è un valore ottale a 3 caratteri) può essere usata per indicizzare un codice di carattere noto in una specifica codifica di font. Per esempio il font Adobe Symbol utilizza una codifica personalizzata in cui l'ottale 245 rappresenta il simbolo dell'infinito. Si potrebbe incorporarlo in una stringa di testo avanzato dando il nome del font e il codice del carattere "{/Symbol \245}". Questo è utile soprattutto per il terminale PostScript che non può facilmente gestire la codifica UTF-8.

È possibile specificare un carattere dal suo codice Unicode, come \U+hhhh, dove hhhh è il codice esadecimale a 4 o 5 caratteri. Per esempio il codice per il simbolo dell'infinito è \U+221E. Questo sarà convertito in una sequenza di byte UTF-8 in output, se appropriato. In un ambiente UTF-8 questo meccanismo non è necessario per caratteri speciali stampabili, poiché sono gestiti in una stringa di testo come qualsiasi altro carattere. Tuttavia, è utile per combinare forme o segni diatrici supplementari (per esempio una freccia sopra una lettera per rappresentare un vettore). Vedere **set encoding (p. 154)**, **utf8 (p. 154)**, e `online unicode demo.`

Ambiente

Un certo numero di variabili d'ambiente della shell sono capite da **gnuplot**. Nessuna di queste è necessaria, ma potrebbero essere utili.

GNUTERM, se definita, è usata per impostare il tipo di terminale all'avvio (start-up). A partire dalla versione 5.2 l'intera stringa in GNUTERM viene passata a "set term" in modo che le opzioni di terminale possano essere incluse. Per esempio

```
GNUTERM="postscript eps color size 5in, 3in"
```

Questo può essere sovrascritto dal file di avvio ~ /.gnuplot (o equivalente) (vedere **startup (p. 57)**) e naturalmente dai successivi comandi **set term** espliciti.

GNUHELP può essere definita come il pathname del file HELP (gnuplot.gih).

Su VMS, il nome logico GNUPLOT$HELP dovrebbere essere definito come il nome della libreria help (aiuto) per **gnuplot**. L'help di **gnuplot** può essere inserito in qualsiasi libreria help di sistema VMS.

Su Unix, HOME è usata come nome di una directory per cercare un file .gnuplot se non ne viene trovato nessuno nella directory corrente. Su MS-DOS, Windows e OS/2, viene usata GNUPLOT. Su Windows, viene provata anche la variabile USERPROFILE specifica di NT. Su VMS, viene usata SYS$LOGIN. Digitare **help startup**.

Su Unix, PAGER è utilizzata come un filtro di output per i messaggi help.

Su Unix, SHELL è usata per il comando **shell**. Su MS-DOS e OS/2, COMSPEC è utilizzata per il comando **shell**.

FIT_SCRIPT può essere usata per specificare un comando **gnuplot** da eseguire quando viene interrotto un fit — vedere **fit (p. 95)**. **FIT_LOG** specifica il nome del file predefinito del logfile mantenuto da fit.

GNUPLOT_LIB può essere usato per definire directory di ricerca aggiuntive per i file di dati e di comandi. La variabile può contenere un singolo nome di directory, o una lista di directory divise da un separatore di percorso specifico della piattaforma, ad es. ':' su Unix, o ';' sulle piattaforme DOS/Windows/OS/2. Il contenuto di GNUPLOT_LIB viene aggiunto alla variabile **loadpath**, ma non viene salvato con i comandi **save** e **save set**.

Diversi driver del terminale gnuplot accedono ai font TrueType tramite la libreria gd. Per questi driver il percorso di ricerca dei font è controllato dalla variabile d'ambiente GDFONTPATH. Inoltre, un font predefinito per questi driver può essere impostato tramite la variabile d'ambiente GNUPLOT_DEFAULT_GDFONT.

Il terminale postscript usa il proprio percorso di ricerca dei font. È controllato dalla variabile d'ambiente GNUPLOT_FONTPATH.

GNUPLOT_PS_DIR è usata dal driver postscript per cercare file di prologo esterni. A seconda del processo di compilazione, gnuplot contiene o una copia incorporata di questi file o un percorso predefinito a codifica fissa. È possibile usare questa variabile per far sì che il terminale postscript usi file di prologo personalizzati piuttosto che i file predefiniti. Vedere **postscript prologue (p. 293)**.

Espressioni

In generale, qualsiasi espressione matematica accettata da C, FORTRAN, Pascal, o BASIC è valida. La precedenza di questi operatori è determinata dalle specifiche del linguaggio di programmazione C. Lo spazio bianco (spazi e tab) è ignorato dentro le espressioni.

Si noti che gnuplot usa aritmetica sia "reale" che "intera", come FORTRAN e C. I numeri interi sono inseriti come "1", "-10", etc; i numeri reali come "1.0", "-10.0", "1e1", "3.5e-1", etc. La differenza più importante tra le due forme consiste nella divisione: la divisione di interi tronca: 5/2 = 2; la divisione di reali no: 5.0/2.0 = 2.5. Nelle espressioni miste, gli interi sono "promossi" a reali prima della valutazione: 5/2e0 = 2.5. Il risultato della divisione di un intero negativo per uno positivo può variare tra i compilatori. Si consiglia di fare un test come "print -5/2" per determinare se il proprio sistema arrotonda sempre per difetto (-5/2 produce -3) o arrotonda sempre verso lo zero (-5/2 produce -2).

L'espressione intera "1/0" può essere usata per generare un flag "non definito", che fa sì che un punto venga ignorato. Oppure si può usare la variabile predefinita NaN per ottenere lo stesso risultato. Vedere **using (p. 121)** per un esempio.

Gnuplot può anche eseguire semplici operazioni su stringhe e variabili stringa. Per esempio, l'espressione ("A" . "B" eq "AB") si valuta come vera, illustrando l'operatore di concatenazione delle stringhe e l'operatore di eguaglianza delle stringhe.

Una stringa che contiene un valore numerico viene promossa al corrispondente valore intero o reale se usata in un'espressione numerica. Quindi ("3" + "4" == 7) e (6.78 == "6.78") valutano entrambe a vero. Un intero, ma non un valore reale o complesso, viene promosso a stringa se usato nella concatenazione di stringhe. Un caso tipico è l'utilizzo di interi per costruire nomi di file o altre stringhe; per esempio: ("file" . 4 eq "file4") è vero.

Le sottostringhe possono essere specificate usando un descrittore di intervallo post-fisso [beg:end]. Per esempio, "ABCDEF"[3:4] == "CD" e "ABCDEF"[4:*] == "DEF" La sintassi "string"[beg:end] equivale esattamente a chiamare la funzione incorporata con valore di stringa substr("string",beg,end), eccetto che non si può omettere beg o end dalla chiamata della funzione.

Aritmetica complessa

Le operazioni aritmetiche e la maggior parte delle funzioni incorporate supportano l'uso di argomenti complessi. Le costanti complesse sono espresse come {<real>,<imag>}, dove <real> e <imag> devono essere costanti numeriche. Quindi {0,1} rappresenta 'i'. Le componenti reali e immaginarie di un valore x complesso possono essere ricavate come real(x) e imag(x). Il modulo è dato da abs(x).

Gli stili di grafico 2D e 3D standard di Gnuplot possono plottare solo valori reali; se occorre plottare una funzione di valore complesso f(x) con componenti immaginarie diverse da zero è necessario scegliere tra plottare real(f(x)) o abs(f(x)). Per esempi di rappresentazione di valori complessi usando i colori, vedere demo di funzione trigonometrica complessa (complex_trig.dem)

Costanti

Le costanti intere sono interpretate tramite la routine della libreria C strtoll(). Questo significa che le costanti che iniziano con "0" sono interpretate come ottali, e le costanti che iniziano con "0x" o "0X" sono interpretate come esadecimali.

Le costanti in virgola mobile sono interpretate tramite la routine della libreria C atof().

Le costanti complesse sono espresse come {<real>,<imag>}, dove <real> e <imag> devono essere costanti numeriche. Per esempio, {3,2} rappresenta 3 + 2i; {0,1} rappresenta 'i'. Le parentesi graffe sono esplicitamente richieste in questo caso.

Le costanti stringa consistono in qualsiasi sequenza di caratteri racchiusi tra virgolette singole o virgolette doppie. La distinzione tra virgolette singole e doppie è importante. Vedere **virgolette (p. 61)**.

Esempi:

```
1 -10 0xffaabb        # costanti intere
1.0 -10. 1e1 3.5e-1   # costanti in virgola mobile
{1.2, -3.4}           # costante complessa
"Line 1\nLine 2"      # costante stringa (\n viene espanso a newline)
'123\n456'            # costante stringa (\ e n sono caratteri ordinari)
```

Funzioni

Gli argomenti delle funzioni matematiche in **gnuplot** possono essere interi, reali o complessi a meno che non sia indicato diversamente. Le funzioni che accettano o restituiscono angoli (ad esempio sin(x)) trattano i valori degli angoli come radianti, ma questo può essere cambiato in gradi usando il comando **set angles**.

Libreria funzioni matematiche		
Funzioni	Argomenti	Return
abs(x)	qualsiasi	valore assoluto di x, $\|x\|$; stesso tipo
abs(x)	complesso	lunghezza di x, $\sqrt{\text{real}(x)^2 + \text{imag}(x)^2}$
acos(x)	qualsiasi	$\cos^{-1} x$ (inverso del coseno)
acosh(x)	qualsiasi	$\cosh^{-1} x$ (inverso del coseno iperbolico) in radianti
airy(x)	qualsiasi	funzione di Airy Ai(x)
arg(x)	complesso	la fase di x
asin(x)	qualsiasi	$\sin^{-1} x$ (inverso della funzione del seno)
asinh(x)	qualsiasi	$\sinh^{-1} x$ (inversa della funzione del seno iperbolico) in radianti
atan(x)	qualsiasi	$\tan^{-1} x$ (inverso della tangente)
atan2(y,x)	int o reale	$\tan^{-1}(y/x)$ (inverso della tangente)
atanh(x)	qualsiasi	$\tanh^{-1} x$ (inversa della tangente iperbolica) in radianti
EllipticK(k)	reale k $\in$ (-1:1)	$K(k)$ integrale ellittico completo del primo tipo

Libreria funzioni matematiche		
Funzioni	Argomenti	Return
EllipticE(k)	reale k ∈ [-1:1]	$E(k)$ integrale ellittico completo del secondo tipo
EllipticPi(n,k)	reale n<1, reale k ∈ (-1:1)	$\Pi(n,k)$ integrale ellittico completo del terzo tipo
besj0(x)	int o reale	J_0 funzione di Bessel di x in radianti
besj1(x)	int or reale	J_1 funzione di Bessel di x in radianti
besjn(n,x)	int, reale	J_n funzione di Bessel di x in radianti
besy0(x)	int o reale	Y_0 funzione di Bessel di x in radianti
besy1(x)	int o reale	Y_1 funzione di Bessel di x in radianti
besyn(n,x)	int, reale	Y_n funzione di Bessel di x in radianti
besi0(x)	reale	Funzione di Bessel modificata di ordine 0, x in radianti
besi1(x)	reale	Funzione di Bessel modificata di ordine 1, x in radianti
besin(n,x)	int, reale	Funzione di Bessel modificata di ordine n, x in radianti
ceil(x)	qualsiasi	$\lceil x \rceil$, il più piccolo intero non inferiore a x (parte reale)
cos(x)	qualsiasi	$\cos x$, coseno di x
cosh(x)	qualsiasi	$\cosh x$, coseno iperbolico di x in radianti
erf(x)	qualsiasi	erf(real(x)), funzione degli errori di (x) reale
erfc(x)	qualsiasi	erfc(real(x)), 1.0 - funzione degli errori di (x) reale
exp(x)	qualsiasi	e^x, funzione esponenziale di x
expint(n,x)	int $n \geq 0$, reale $x \geq 0$	$E_n(x) = \int_1^\infty t^{-n} e^{-xt}\, dt$, funzione integrale esponenziale di x
floor(x)	qualsiasi	$\lfloor x \rfloor$, il più grande intero non maggiore di x (parte reale)
gamma(x)	qualsiasi	gamma(real(x)), funzione gamma di un reale (x)
ibeta(p,q,x)	qualsiasi	ibeta(real(p, q, x)), funzione ibeta di un reale (p,q,x)
inverf(x)	qualsiasi	funzione degli errori di un numero reale(x)
igamma(a,x)	qualsiasi	igamma(real(a, x)), funzione gamma di un numero reale(a,x)
imag(x)	complesso	parte immaginaria di x come numero reale
invnorm(x)	qualsiasi	funzione di distribuzione normale inversa di un numero reale(x)
int(x)	reale	parte intera di x, troncata verso lo zero
lambertw(x)	reale	funzione W di Lambert
lgamma(x)	qualsiasi	lgamma(real(x)), funzione lgamma di un numero reale(x)
log(x)	qualsiasi	$\log_e x$, logaritmo naturale (base e) di x
log10(x)	qualsiasi	$\log_{10} x$, logaritmo (base 10) di x
norm(x)	qualsiasi	funzione di distribuzione normale (gaussiana) di un numero reale(x)
rand(x)	int	numero pseudo casuale nell'intervallo aperto (0:1)
real(x)	qualsiasi	parte reale di x
sgn(x)	qualsiasi	1 se $x > 0$, -1 se $x < 0$, 0 se $x = 0$. imag(x) ignorato
sin(x)	qualsiasi	$\sin x$, seno di x
sinh(x)	qualsiasi	$\sinh x$, seno iperbolico di x in radianti
sqrt(x)	qualsiasi	$\sqrt{x}$, radice quadrata di x
tan(x)	qualsiasi	$\tan x$, tangente di x
tanh(x)	qualsiasi	$\tanh x$, tangente iperbolica di x in radians
voigt(x,y)	reale	funzione Voigt/Faddeeva $\frac{y}{\pi} \int \frac{exp(-t^2)}{(x-t)^2+y^2} dt$ Nota: voigt$(x, y) = real$(faddeeva$(x + iy)$)

Funzioni speciali di libcerf (solo se disponibili)		
Funzione	Argomenti	Return
cerf(z)	complesso	funzione degli errori complessa
cdawson(z)	complesso	estensione complessa dell'integrale di Dawson $D(z) = \frac{\sqrt{\pi}}{2} e^{-z^2} erfi(z)$
faddeeva(z)	complesso	funzione degli errore complessa riscalata $w(z) = e^{-z^2}\ erfc(-iz)$
erfi(x)	reale	funzione di errori immaginari $erf(x) = -i * erf(ix)$

Funzioni speciali di libcerf (solo se disponibili)		
Funzione	Argomenti	Return
VP(x,σ,γ)	reale	profilo di Voigt $VP(x,\sigma,\gamma) = \int_{-\infty}^{\infty} G(x';\sigma)L(x-x';\gamma)dx'$

Funzioni di stringa		
Funzione	Argomenti	Return
gprintf("format",x,...)	qualsiasi	risultato della stringa dall'applicazione del parser di formato di gnuplot
sprintf("format",x,...)	vari	risultato della stringa da sprintf in linguaggio C
strlen("string")	stringa	numero di caratteri in una stringa
strstrt("string","key")	stringhe	int indice del primo carattere della sottostringa "key"
substr("string",beg,end)	vari	stringa "string"[beg:end]
strftime("timeformat",t)	qualsiasi	risultato della stringa dall'applicazione del parser temporale di gnuplot
strptime("timeformat",s)	stringa	secondi dall'anno 1970 come dati nella stringa s
system("command")	stringa	stringa contenente il flusso di output del comando shell
trim(" string ")	stringa	stringa senza spazi bianchi iniziali o finali
word("string",n)	stringa, int	restituisce la parola n in "string"
words("string")	stringa	restituisce il numero di parole in "string"

altre funzioni di **gnuplot**		
Funzione	Argomenti	Return
column(x)	int o stringa	colonna x durante la manipolazione del file di dati.
columnhead(x)	int	stringa contenente la prima voce della colonna x nel file di dati.
exists("X")	stringa	restituisce 1 se una variabile chiamata X è definita, altrimenti 0.
hsv2rgb(h,s,v)	h,s,v $\in$ [0:1]	valore di colore RGB a 24bit.
palette(z)	doppio	colore della palette RGB mappato a z.
stringcolumn(x)	int o stringa	contenuto della colonna x come una stringa.
timecolumn(N,"timeformat")	int, stringa	dati temporali della colonna N durante l'input dei dati.
tm_hour(t)	tempo in sec	l'ora (0..23)
tm_mday(t)	tempo in sec	il giorno del mese (1..31)
tm_min(t)	tempo in sec	il minuto (0..59)
tm_mon(t)	tempo in sec	il mese (0..11)
tm_sec(t)	tempo in sec	il secondo (0..59)
tm_wday(t)	tempo in sec	il giorno della settimana (Dom..Sab) come (0..6)
tm_week(t)	tempo in sec	la settimana dell'anno nel sistema ISO8601 "week date" (1..53)
tm_yday(t)	tempo in sec	il giorno dell'anno (0..365)
tm_year(t)	tempo in sec	l'anno
time(x)	qualsiasi	il sistema orario corrente in secondi
valid(x)	int	verifica della validità della colonna column(x) durante la manip. dei file di dati
value("name")	stringa	restituisce il valore corrente della variabile nominata.
voxel(x,y,z)	reale	valore del voxel della griglia attiva che contiene il punto (x,y,z)

Integrali ellittici

La funzione **EllipticK(k)** restituisce l'integrale ellittico completo del primo tipo, cioè l'integrale definito tra 0 e pi/2 della funzione **(1-(k*sin(p))**2)**(-0.5)**. Il dominio di **k** è da -1 a 1 (esclusi).

La funzione **EllipticE(k)** restituisce l'integrale ellittico completo del secondo tipo, cioè l'integrale definito tra 0 e pi/2 della funzione **(1-(k*sin(p))**2)**0.5**. Il dominio di **k** è da -1 a 1 (inclusi).

La funzione **EllipticPi(n,k)** restituisce l'integrale ellittico completo del terzo tipo, cioè l'integrale definito tra 0 e pi/2 della funzione **(1-(k*sin(p))**2)**(-0.5)/(1-n*sin(p)**2)**. Il parametro **n** deve essere minore di 1, mentre **k** deve essere tra -1 e 1 (esclusi). Notare che per definizione EllipticPi(0,k) == EllipticK(k) per tutti i possibili valori di **k**.

Generatore di numeri casuali(random)

La funzione **rand()** produce una sequenza di numeri pseudo-casuali compresi tra 0 e 1 usando un algoritmo di P. L'Ecuyer e S. Cote, "Implementing a random number package with splitting facilities", ACM Transactions on Mathematical Software, 17:98-111 (1991).

```
rand(0)     restituisce un numero pseudo casuale nell'intervallo
            aperto (0:1) generato dal valore corrente di due
            semi 32-bit interni.
rand(-1)    resetta entrambi i semi a un valore standard.
rand(x)     per intero 0 < x < 2^31-1 imposta entrambi i semi
            interni su x.
rand({x,y}) per intero 0 < x,y < 2^31-1 imposta il seme1 su x
            e seme2 a y.
```

Valore

B = value("A") è effettivamente uguale a B = A, dove A è il nome di una variabile definita dall'utente. Questo è utile quando il nome della variabile è esso stesso contenuto in una variabile stringa. Vedere **variabili definite dall'utente (p. 45)**. Consente anche di leggere il nome di una variabile da un file di dati. Se l'argomento è un'espressione numerica, value() restituisce il valore di quell'espressione. Se l'argomento è una stringa che non corrisponde a una variabile attualmente definita, value() restituisce NaN.

Contare ed estrarre parole

word("string",n) restituisce la n-esima parola nella stringa. Per esempio, **word("one two three",2)** restituisce la stringa "two".

words("string") restituisce il numero di parole nella stringa. Per esempio, **words(" a b c d")** restituisce 4.

Le funzioni **word** e **words** forniscono un supporto limitato per le stringhe tra virgolette, possono essere usate sia le virgolette singole che doppie:

```
print words("\"double quotes\" or 'single quotes'")   # 3
```

Una virgoletta iniziale deve essere preceduta da uno spazio bianco oppure essere lei stessa il primo carattere della stringa. Questo significa che gli apostrofi nel mezzo o alla fine delle parole vengono considerati come parte della rispettiva parola:

```
print words("Alexis' phone doesn't work") # 4
```

L'escape di caratteri virgolette non è supportato. Se si vogliono tenere alcune virgolette, la rispettiva sezione deve essere circondata dall'altro tipo di virgolette:

```
s = "Keep \"'single quotes'\" or '\"double quotes\"'"
print word(s, 2) # 'single quotes'
print word(s, 4) # "double quotes"
```

Notare che nell'ultimo esempio le virgolette di escape sono necessarie solamente per la definizione della stringa. **trim(" padded string ")** restituisce la stringa originale senza spazi bianchi all'inizio o alla fine. Questo è utile per i confronti tra stringhe di campi di dati di input che potrebbero contenere spazi bianchi extra. Per esempio:

```
plot FOO using 1:( trim(strcol(3)) eq "A" ? $2 : NaN )
```

Operatori

Gli operatori in **gnuplot** sono gli stessi dei corrispondenti operatori nel linguaggio di programmazione C, eccetto che tutti gli operatori accettano argomenti interi, reali e complessi, se non diversamente specificato. L'operatore ** (esponenziale) è supportato, come in FORTRAN.

Le parentesi possono essere usate per cambiare l'ordine di valutazione.

Unario

La seguente è una lista di tutti gli operatori unari e dei loro usi:

Unary Operators		
Simbolo	Esempio	Spiegazione
`-`	`-a`	meno unario
`+`	`+a`	più unario (nessuna operazione)
`~`	`~a`	* complemento a uno
`!`	`!a`	* negazione logica
`!`	`a!`	* fattoriale
`$`	`$3`	* chiama arg/colonna durante la manipolazione 'using'
`|`	`|A|`	cardinalità dell'array A

(*) Le spiegazioni con asterisco indicano che l'operatore necessita un argomento intero.

La precedenza degli operatori è la stessa di Fortran e C. Come in questi linguaggi, le parentesi possono essere usate per cambiare l'ordine delle operazioni. Quindi -2**2 = -4, ma (-2)**2 = 4.

L'operatore fattoriale restituisce un intero quando N! è sufficentemente piccolo (N <= 20 per 64-bit interi). Restituisce un'approssimazione in virgola mobile per valori maggiori di N.

Questo operatore restituidce il numero di elementi |A| quando viene applicato all'array A. Restituisce il numero di linee di dati |$DATA| quando viene applicato al datablock $DATA.

Binario

La seguente è una lista di tutti gli operatori binari e dei loro usi:

Operatoti binari		
Simbolo	Esempio	Spiegazione
`**`	`a**b`	potenza
`*`	`a*b`	moltiplicazione
`/`	`a/b`	divisione
`%`	`a%b`	* modulo
`+`	`a+b`	addizione
`-`	`a-b`	sottrazione
`==`	`a==b`	uguaglianza
`!=`	`a!=b`	disuguaglianza
`<`	`a<b`	minore di
`<=`	`a<=b`	minore o uguale a
`>`	`a>b`	maggiore di
`>=`	`a>=b`	maggiore o uguale a
`<<`	`0xff<<1`	shift sinistro senza segno
`>>`	`0xff>>1`	shift destro senza segno
`&`	`a&b`	* AND 'bit a bit'
`^`	`a^b`	* OR esclusivo 'bit a bit'
`|`	`a|b`	* OR inclusivo 'bit a bit'
`&&`	`a&&b`	* congiunzione logica AND
`||`	`a||b`	* congiunzione logica OR
`=`	`a = b`	assegnazione
`,`	`(a,b)`	valuazione seriale
`.`	`A.B`	concatenazione di stringhe
`eq`	`A eq B`	uguaglianza di stringhe
`ne`	`A ne B`	disuguaglianza di stringhe

(*) Le spiegazioni con asterisco indicano che l'operatore necessita argomenti interi. Le lettere maiuscole A e B indicano che l'operatore richiede argomenti stringa.

Gli AND (&&) e gli OR (||) logici vanno in corto circuito come in C. Cioè, il secondo operando **&&** non viene valutato se il primo è falso; il secondo operando || non viene valutato se il primo è vero.

La valutazione seriale si verifica solo nelle parentesi ed è garantito che proceda in ordine da sinistra verso destra. Viene restituito il valore della sottoespressione più a destra.

Ternario

Vi è un solo operatore ternario:

Operatore ternario		
Simbolo	Esempio	Spiegazione
`?:`	`a?b:c`	operazione ternaria

L'operatore ternario si comporta come in C. Il primo argomento (a), il quale deve essere un intero, è valutato. Se è vero (diverso da zero), il secondo argomento (b) è valutato e restituito; altrimenti il terzo argomento (c) è valutato e restituito.

L'operatore ternario è molto utile sia nella costruzione di funzioni definite a tratti che nel plottare punti solo quando sono sottisfatte determinate condizioni.

Esempi:

Plotta una funzione che deve essere uguale a sin(x) per 0 <= x < 1, 1/x per 1 <= x < 2, e indefinita altrove:

```
f(x) = 0<=x && x<1 ? sin(x) : 1<=x && x<2 ? 1/x : 1/0
plot f(x)
```

Si noti che **gnuplot** ignora i valori non definiti, quindi il ramo finale della della funzione (1/0) non produrrà alcun punto plottabile. Si noti anche che f(x) sarà plottato come una funzione continua attraverso la discontinuità se viene usato uno stile linea. Per plottarlo in modo discontinuo, è necessario creare funzioni separate per i due pezzi. (Le funzioni parametriche sono utili anche a questo scopo.)

Per i dati in un file, plottare la media dei dati nelle colonne 2 e 3 contro il dato nella colonna 1, ma solo se il dato nella colonna 4 è non negativo:

```
plot 'file' using 1:( $4<0 ? 1/0 : ($2+$3)/2 )
```

Per una spiegazione della sintassi **using**, vedere **plot datafile using (p. 121)**.

Sommatoria

Un'espressione sommatoria ha la forma

```
sum [<var> = <start> : <end>] <expression>
```

<var> è trattato come una variabile intera che assume i valori integrali successivi da <start> fino a <end>. Per ciascuno di essi, il valore attuale di <expression> viene aggiunto a un totale parziale il cui valore finale diventa il valore dell'espressione di sommatoria. Esempi:

```
print sum [i=1:10] i
    55.
# Equivale a plot 'data' using 1:($2+$3+$4+$5+$6+...)
plot 'data' using 1 : (sum [col=2:MAXCOL] column(col))
```

Non è necessario che <expression> contenga la variabile <var>. Anche se <start> e <end> possono essere specificate come variabili o espressioni, il loro valore non può essere cambiato dinamicamente come effetto collaterale dell'esecuzione della sommatoria. Se <end> è minore di <start> allora il valore della sommatoria è zero.

Variabili definite da gnuplot

Gnuplot mantiene un certo numero di variabili di sola lettura che riflettono l'attuale stato interno del programma e il grafico più recente. Queste variabili iniziano con il prefisso "GPVAL_". Alcuni esempi includono GPVAL_TERM, GPVAL_X_MIN, GPVAL_X_MAX, GPVAL_Y_MIN. Digitare **show variables all** per visualizzare la lista completa e le variabili attuali. I valori relativi ai parametri degli assi (intervalli, base log) sono valori usati durante l'ultimo grafico, non quelli attualmente **set** (impostati). Esempio: Calcolare le coordinate dello schermo frazionario del punto [X,Y]

```
GRAPH_X = (X - GPVAL_X_MIN) / (GPVAL_X_MAX - GPVAL_X_MIN)
GRAPH_Y = (Y - GPVAL_Y_MIN) / (GPVAL_Y_MAX - GPVAL_Y_MIN)
SCREEN_X = GPVAL_TERM_XMIN + GRAPH_X * (GPVAL_TERM_XMAX - GPVAL_TERM_XMIN)
SCREEN_Y = GPVAL_TERM_YMIN + GRAPH_Y * (GPVAL_TERM_YMAX - GPVAL_TERM_YMIN)
FRAC_X = SCREEN_X * GPVAL_TERM_SCALE / GPVAL_TERM_XSIZE
FRAC_Y = SCREEN_Y * GPVAL_TERM_SCALE / GPVAL_TERM_YSIZE
```

La variabile di sola lettura GPVAL_ERRNO viene impostata a un valore diverso da zero se un qualsiasi comando di gnuplot viene terminato per via di un errore. Il messaggio di errore più recente viene conservato nella variabile stringa GPVAL_ERRMSG. Sia GPVAL_ERRNO che GPVAL_ERRMSG possono essere azzerate usando il comando **reset errors**.

Terminali interattivi con la funzionalità **mouse** mantengono variabili di sola lettura con il prefisso "MOUSE_". Vedere **variabili mouse (p. 55)** per i dettagli.

Il meccanismo **fit** utilizza diverse variabili con nomi che iniziano con "FIT_". È più sicuro evitare di usare tali nomi. Quando si usa **set fit errorvariables**, l'errore per ogni parametro adattato sarà memorizzato in

una variabile chiamata come il parametro ma con "_err" aggiunto di seguito. Vedere la documentazione su **fit (p. 95)** e **set fit (p. 155)** per maggiori dettagli.

Vedere **variabili definite dall'utente (p. 45)**, **reset errors (p. 135)**, **variabili mouse (p. 55)**, e **fit (p. 95)**.

Variabili e funzioni definite dall'utente

Le nuove variabili e funzioni definite dall'utente da una a dodici variabili possono essere dichiarate e utilizzate ovunque, incluso nel comando **plot** stesso.

Sintassi di funzioni definite dall'utente:

```
<func-name>( <dummy1> {,<dummy2>} ... {,<dummy12>} ) = <expression>
```

dove <expression> è definito in termini di <dummy1> fino a <dummy12>.

Sintassi di variabili definite dall'utente:

```
<variable-name> = <constant-expression>
```

Esempi:

```
w = 2
q = floor(tan(pi/2 - 0.1))
f(x) = sin(w*x)
sinc(x) = sin(pi*x)/(pi*x)
delta(t) = (t == 0)
ramp(t) = (t > 0) ? t : 0
min(a,b) = (a < b) ? a : b
comb(n,k) = n!/(k!*(n-k)!)
len3d(x,y,z) = sqrt(x*x+y*y+z*z)
plot f(x) = sin(x*a), a = 0.2, f(x), a = 0.4, f(x)
file = "mydata.inp"
file(n) = sprintf("run_%d.dat",n)
```

Gli ultimi due esempi illustrano una variabile stringa definita dall'utente e una funzione stringa definita dall'utente.

Notare che le variabili **pi** (3.14159...) e **NaN** (IEEE "Not a Number") sono già definite. È possibile ridefinire queste variabili in qualcosa di diverso se è assolutamente necessario. I valori originali possono essere recuperati impostando:

```
NaN = GPVAL_NaN
pi  = GPVAL_pi
```

Altre variabili potrebbero essere definite in varie operazioni di gnuplot come mousing in terminali interattivi o fitting; vedere **variabili definite da gnuplot (p. 44)** per ulteriori dettagli.

È possibile controllare l'esistenza di una determinata variabile V con l'espressione exists("V"). Per esempio

```
a = 10
if (exists("a")) print "a is defined"
if (!exists("b")) print "b is not defined"
```

I nomi validi sono gli stessi della maggior parte dei linguaggi di programmazione: devono iniziare con una lettera, ma i caratteri seguenti possono essere lettere, cifre o "_".

Ogni definizione di funzione è disponibile come una speciale variabile con valore di stringa con prefisso 'GPFUN_'.

Esempio:

```
set label GPFUN_sinc at graph .05,.95
```

Vedere **show functions (p. 161)**, **funzioni (p. 126)**, **variabili definite da gnuplot (p. 44)**, **macro (p. 59)**, **value (p. 41)**.

Array

Gli array sono implementati come elenchi indicizzati di variabili utente. Gli elementi in un array non sono limitati a un solo tipo di variabile. Gli array devono essere creati esplicitamente prima di essere referenziati. La dimensione di un array non può essere cambiata dopo la sua creazione. Tutti gli elementi sono inizialmente non definiti. Nella maggior parte dei casi un elemento di array può essere usato al posto di una variabile utente con nome.

La cardinalità (numero di elementi) dell'array A è data dall'espressione |A|.

Esempio:

```
array A[6]
A[1] = 1
A[2] = 2.0
A[3] = {3.0, 3.0}
A[4] = "four"
A[6] = A[2]**3
array B[6] = [ 1, 2.0, A[3], "four", , B[2]**3 ]

do for [i=1:6] { print A[i], B[i] }
    1 1
    2.0 2.0
    {3.0, 3.0} {3.0, 3.0}
    four four
    <undefined> <undefined>
    8.0 8.0
```

Nota: Gli array e le variabili condividono lo stesso namespace. Per esempio, l'assegnazione del nome FOO a una variabile stringa distruggerà qualsiasi array precedentemente creato con nome FOO.

Il nome di un array può essere usato in un comando **plot**, **splot**, **fit**, o **stats**. Questo equivale a fornire un file in cui la colonna 1 contiene l'indice dell'array (da 1 alla dimensione), la colonna 2 contiene il valore di real(A[i]) e la colonna 3 contiene il valore di imag(A[i]).

Esempio:

```
array A[200]
do for [i=1:200] { A[i] = sin(i * pi/100.) }
plot A title "sin(x) in centiradians"
```

Quando si plotta la parte immaginaria di valori array complessi, può essere referenziata sia come imag(A[$1]) che come $3. Questi due comandi sono equivalenti

```
plot A using (real(A[$1])) : (imag(A[$1]))
plot A using 2:3
```

Font

Gnuplot non fornisce alcun font proprio. Si basa sulla gestione esterna dei font i cui dettagli purtroppo variano da un tipo di terminale all'altro. Qui viene riportata una breve documentazione dei meccanismi dei font che si applicano a più di un tipo di terminale. Per informazioni sull'uso dei font da parte di altri terminali individuali, vedere la documentazione di quello specifico terminale.

Nononstante sia possibile includere simboli non alfabetici passando temporaneamente a un font speciale, per esempio il font Adobe Symbol, ora il metodo preferito è quello di scegliere la codifica UTF-8 e trattare il simbolo come un qualsiasi altro carattere. In alternativa si può specificare il punto di ingresso unicode per il simbolo desiderato come sequenza di escape nella modalità di testo avanzato.

Vedere **encoding (p. 154)**, **unicode (p. 36)**, **locale (p. 175)**, e **sequenze di escape (p. 36)**.

Cairo (pdfcairo, pngcairo, epscairo, wxt terminals)

Questi terminali trovano e accedono ai font usando il set di strumenti esterni fontconfig. Vedere manuale utente fontconfig.

Di solito è sufficiente richiedere in gnuplot un font con un nome e dimensione generici, lasciando che fontconfig sostituisca un font simile se necessario. Tutti i seguenti probabilmente funzioneranno:

```
set term pdfcairo font "sans,12"
set term pdfcairo font "Times,12"
set term pdfcairo font "Times-New-Roman,12"
```

Gd (png, gif, jpeg, sixel terminals)

La gestione dei font per i terminali png, gif, jpeg, e sixelgd è fatta dalla libreria libgd. Libgd fornisce 5 font di base. Si tratta di **tiny** (5x8 pixels), **small** (6x12 pixels), **medium**, (7x13 Bold), **large** (8x16) or **giant** (9x15 pixels). Questi font non possono essere ridimensionati o ruotati. Usare una di queste keyword al posto della keyword **font**. Ad es.:

```
set term png tiny
```

Sulla maggior parte dei sistemi libgd fornisce accesso anche ai font Adobe Type 1 (*.pfa) e ai font TrueType (*.ttf). È necessario dare il nome del file dei font, non il nome del font al suo interno, sotto forma di "<face> {,<pointsize>}". <face> è o il percorso completo ai file di font o la prima parte del nome di un file in una delle directory elencate nella variabile d'ambiente GDFONTPATH. Cioè, 'set term png font "Face"' cercherà un file di font chiamato o <somedirectory>/Face.ttf o <somedirectory>/Face.pfa. Per esempio, se GDFONTPATH contiene **/usr/local/fonts/ttf:/usr/local/fonts/pfa**, le seguenti coppie di comandi sono equivalenti

```
set term png font "arial"
set term png font "/usr/local/fonts/ttf/arial.ttf"
set term png font "Helvetica"
set term png font "/usr/local/fonts/pfa/Helvetica.pfa"
```

Per richiedere allo stesso tempo una dimensione predefinita del font:

```
set term png font "arial,11"
```

Entrambi i font TrueType e Adobe Type 1 sono completamente ridimensionabili e ruotabili. Se non viene richiesto uno specifico font nel comando "set term", gnuplot controlla la variabile d'ambiente GNUPLOT_DEFAULT_GDFONT per vedere se è presente un font predefinito preferito.

Postscript (anche postscript incapsulato *.eps)

La gestione dei font PostScript è fatta dalla stampante o dal programma di visualizzazione. Gnuplot può creare un PostScript valido o un PostScript (*.eps) incapsulato anche se nessun font è installato sul proprio computer. Gnuplot fa riferimento semplicemente al font per nome nel file di output, e presume che la stampante o il programma di visualizzazione sappiano come trovare o approssimare un font con quel nome.

Tutte le stampanti e i visualizzatori PostScript dovrebbero conoscere il set standard di font Adobe **Times-Roman**, **Helvetica**, **Courier**, e **Symbol**. È probabile che molti altri font siano disponibili, ma il set specifico dipende dal proprio sistema o configurazione della stampante. Gnuplot non sa o non si preoccupa di questo; i file di output *.ps o *.eps che crea si riferiranno semplicemente ai nomi dei font che si richiederanno.

Quindi

```
set term postscript eps font "Times-Roman,12"
```

produrrà un output adatto a tutte le stampanti e i visualizzatori.

D'altra parte

```
set term postscript eps font "Garamond-Premier-Pro-Italic"
```

produrrà un file di output che contiene PostScript valido, ma visto che fa riferimento a un font specializzato, solo alcune stampanti o visualizzatori saranno in grado di mostrare il font specificatamente richiesto. La maggior parte sostituirà un font diverso.

Nonostante ciò, è possibile incorporare un font specifico nel file di output così che tutte le stampanti possano usarlo. Questo richiede che un file adeguato di descrizione del font sia disponibile sul vostro sistema. Notare che alcuni file di font richiedono licenze specifiche se sono incorporati in questo modo. Vedere **postscript fontfile (p. 292)** per una descrizione più dettagliata ed esempi.

Glossario

In tutto questo documento si è cercato di mantenere la coerenza della nomenclatura. Questo non può essere completamente riuscito perché mano a mano che **gnuplot** si è evoluto nel tempo, sono stati adottati alcuni nomi di comandi e keyword che impediscono una tale perfezione. Questa sezione contiene spiegazioni sul modo in cui alcuni termini sono usati.

Una "pagina" (page) o "videata" (screen) o "canvas" è l'intera area indirizzabile da **gnuplot**. Su un desktop è un'intera finestra; su un plotter, è un singolo foglio di carta; in modalità svga è l'intero schermo del monitor.

Una videata potrebbe contenere uno o più "grafici" (plot). Un grafico è definito da un'ascissa e un'ordinata, anche se queste non devono necessariamente apparire su di esso, così come i margini e qualunque testo scritto su di esso.

Un grafico (plot) contiene un "grafico" (graph). Un grafico è definito da un'ascissa e un'ordinata, anche se non è necessario che questi appaiano effettivamente su di esso.

Un grafico (graph) potrebbe contenere una o più "linee" (lines). Una linea è una singola funzione o insieme di dati. "Line" è anche uno stile di plotting. La parola può anche essere usata nel senso di "una linea di testo". Presumibilmente il contesto rimuoverà qualsiasi ambiguità.

Le linee su un grafico (graph) potrebbero avere nomi individuali. Questi potrebbero essere elencati insieme a un esempio dello stile di plotting usato per rappresentarli nella "chiave" (key), talvolta chiamata anche "legenda" (legend).

La parola "titolo" (title) si presenta con molteplici significati in **gnuplot**. In questo documento sarà sempre accompagnata da "plot" (grafico), "linea", o "key" per differenziarla. Un grafico (graph) 2D potrebbe avere fino a quattro **axes** (assi) etichettati. I nomi dei quattro assi sono "x" per l'asse lungo il bordo inferiore del grafico (plot), "y" per l'asse lungo il bordo sinistro, "x2" per il bordo superiore, e "y2" per il bordo destro. Vedere **assi (p. 108)**.

Un grafico (graph) 3D potrebbe avere fino a tre **axes**(assi) etichettati – "x", "y" e "z". Non è possibile dire in quale punto del grafico (graph) cadrà un particolare asse perché si può cambiare la direzione da cui si vede il grafico (graph) con **set view**.

Quando si parla di file di dati, il termine "record" verrà ripreso e usato per indicare una singola linea di testo nel file, cioè i caratteri tra newline o i caratteri end-of-record. Un "punto" (point) è il dato estratto da un singolo record. Un "blocco" (block) di dati è un insieme di record consecutivi delimitati da record vuoti. Una linea, quando viene riferita nel contesto di un file di dati, è un sottoinsieme di un blocco. Notare che il termine "data block" può anche essere usato per fare riferimento a un blocco denominato di dati in linea (vedere **datablocks (p. 48)**).

Inline data and datablocks

Vi sono due meccanismi per incorporare dati in un flusso di comandi gnuplot. Se il filename speciale '-' appare in un comando plot, allora le linee subito dopo il comando plot sono interpretate come dati inline.

Vedere **special-filenames (p. 120)**. I dati forniti in questo modo possono essere usati solo una volta, dal comando plot che segue.

Il secondo meccanismo definisce un blocco di dati denominato come un here-document. I dati denominati sono persistenti e possono essere riferiti da più di un comando plot. Esempio:

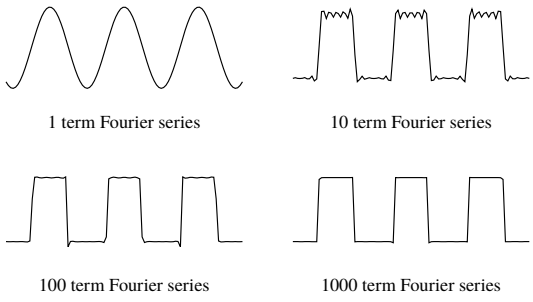
```
$Mydata << EOD
11 22 33 first line of data
44 55 66 second line of data
# comments work just as in a data file
77 88 99
EOD
stats $Mydata using 1:3
plot $Mydata using 1:3 with points, $Mydata using 1:2 with impulses
```

I nomi dei blocchi di dati devono iniziare con un carattere $, che li distingue da altri tipi di variabili persistenti. Il delimitatore della fine dei dati (EOD nell'esempio) può essere una qualsiasi sequenza di caratteri alfanumerici.

Lo storage associato con blocchi di dati denominati può essere rilasciato usando il comando **undefine**. **undefine $*** libera tutti i blocchi di dati.

Iteration

gnuplot supporta l'iterazione dei comandi e i costrutti strutturati a blocchi if/else/while/do. Vedere **if (p. 103)**, **while (p. 247)**, e **do (p. 94)**. La semplice iterazione è possibile all'interno dei comandi **plot** o **set**. Vedere **plot for (p. 128)**. L'iterazione generale che copre più comandi è possibile utilizzando un costrutto a blocchi come mostrato di seguito. Per una nuova funzione correlata, vedere il tipo di espressione **sommatoria (p. 44)**. Ecco un esempio che usa diverse di queste nuove funzioni sintattiche:




```
set multiplot layout 2,2
fourier(k, x) = sin(3./2*k)/k * 2./3*cos(k*x)
do for [power = 0:3] {
    TERMS = 10**power
    set title sprintf("%g term Fourier series",TERMS)
    plot 0.5 + sum [k=1:TERMS] fourier(k,x) notitle
}
unset multiplot
```

L'iterazione è controllata da uno specificatore di iterazione con sintassi

```
for [<var> in "string of N elements"]
```

o

```
for [<var> = <start> : <end> { : <increment> }]
```

Nel primo caso, <var> è una variabile stringa che valuta successivamente a sottostringhe di parole singole da 1 a N della stringa nello specificatore di iterazione. Nel secondo caso, <start>, <end>, e <increment> sono interi o espressioni intere.

Le variabile di gnuplot sono globali, con una sola eccezione. C'è una lista unica, persistente, di variabili attive indicizzate per nome. L'assegnazione a una variabile crea o sostituisce una entry in quella lista. L'unico modo per rimuovere una variabile da quella lista è il comando **undefine**.

L'unica eccezione a questo è la variabile usata in uno specificatore di iterazione. L'ambito della variabile di iterazione è riservato a quell'iterazione. Non si può cambiare permanentemente il valore della variabile di iterazione all'interno della clausola iterata. Se la variabile di iterazione ha un valore prima dell'iterazione, quel valore sarà mantenuto o ripristinato alla fine dell'iterazione. Per esempio, i comandi seguenti stamperanno 1 2 3 4 5 6 7 8 9 10 A.

```
i = "A"
do for [i=1:10] { print i; i=10; }
print i
```

Tipi di linea, colori, e stili

In versioni precedenti di gnuplot, ogni tipo di terminale forniva un set distinto di "linetypes" (tipi di linea) che potevano variare in colore, spessore, pattern punto/trattino (dot/dash), o qualche combinazione di colore e punto/trattino. Non era garantito che questi colori e pattern fossero consistenti tra i diversi tipi di terminale, anche se la maggior parte usava la sequenza di colori rosso/verde/blu/magenta/ciano/giallo. È possibile selezionare questo vecchio comportamento tramite il comando **set colorsequence classic**, ma la versione 5 di gnuplot usa di default una sequenza terminale-indipendente di 8 colori.

È possibile personalizzare ulteriormente la sequenza delle proprietà dei tipi di linea in modo interattivo o in un file di inizializzazione. Vedere **set linetype (p. 173)**. Diversi esempi di file di inizializzazione sono forniti nel pacchetto di distribuzione.

Le proprietà del tipo di linea corrente per un particolare terminale possono essere visualizzate in anteprima lanciando il comando **test** dopo aver impostato il tipo di terminale.

Alle funzioni o file di dati successivi plottati da un singolo comando saranno assegnati tipi di linea successivi nella sequenza predefinita corrente. È possibile sovrascrivere ciò per ogni singola funzione, file di dati o elemento di un grafico dando esplicite proprietà di linea nel comando plot.

Esempi:

```
plot "foo", "bar"                # plotta due file usando i tipi di linea 1, 2
plot sin(x) linetype 4           # usa il colore del tipo di linea 4
```

In generale, i colori possono essere specificati usando i loro nomi, componenti rgb (rosso, verde, blu), componenti hsv (tonalità, saturazione, valore), o una coordinata sulla corrente palette pm3d.

Esempi:

```
plot sin(x) lt rgb "violet"      # uno dei colori con nome di gnuplot
plot sin(x) lt rgb "#FF00FF"     # terna RGB esplicita in esadecimale
plot sin(x) lt palette cb -45    # qualunque colore corrisponda a -45
                                 # nel cbrange corrente della palette
plot sin(x) lt palette frac 0.3  # valore frazionario lungo la palette
```

Vedere **colorspec (p. 50)**, **show colornames (p. 147)**, **hsv (p. 40)**, **set palette (p. 190)**, **cbrange (p. 236)**. Vedere anche **set monochrome (p. 177)**.

I tipi di linea hanno associato un pattern punto/trattino anche se non tutti i tipi di terminale sono in grado di usarlo. La versione 5 di gnuplot permette di specificare il pattern punto/trattino indipendentemente dal colore della linea. Vedere **dashtype (p. 52)**.

Colorspec

Molti comandi permettono di specificare un tipo di linea con un colore esplicito.

Sintassi:

```
... {linecolor | lc} {"colorname" | <colorspec> | <n>}
... {textcolor | tc} {<colorspec> | {linetype | lt} <n>}
... {fillcolor | fc} {<colorspec> | linetype <n> | linestyle <n>}
```

dove <colorspec> ha una delle forme seguenti:

```
rgbcolor "colorname"    # per esempio "blue"
rgbcolor "0xRRGGBB"     # stringa contenente costante esadecimale
rgbcolor "0xAARRGGBB"   # stringa contenente costante esadecimale
rgbcolor "#RRGGBB"      # stringa contenente esadecimale in formato x11
rgbcolor "#AARRGGBB"    # stringa contenente esadecimale in formato x11
rgbcolor <integer val>  # valore intero che rappresenta AARRGGBB
rgbcolor variable       # valore intero viene letto dal file di input
palette frac <val>      # <val> va da 0 a 1
palette cb <value>      # <val> si trova in cbrange
palette z
variable                # indice del colore viene letto dal file di input
bgnd                    # colore di sfondo
black
```

La "<n>" è il numero del tipo di linea il cui colore è usato, vedere **test (p. 245)**.

"colorname" si riferisce a uno dei nomi dei colori incorporati in gnuplot. Per una lista dei nomi disponibili vedere **show colornames (p. 147)**.

Le costanti esadecimali possono essere date tra virgolette come "#RRGGBB" o "0xRRGGBB", dove RRGGBB rappresenta le componenti the rosse, verdi, e blu del colore e devono essere tra 00 e FF. Per esempio, il magenta = rosso full-scale + blu full-scale potrebbe essere rappresento da "0xFF00FF", che è una rappresentazione esadecimale di (255 << 16) + (0 << 8) + (255).

"#AARRGGBB" rappresenta un colore RGB con un valore di canale alpha (trasparenza) nei bit elevati. Un valore alpha di 0 rappresenta un colore totalmente opaco; Cioè, "#00RRGGBB" è uguale a "#RRGGBB". Un valore alpha di 255 (FF) rappresenta trasparenta completa.

La palette di colori è un gradiente lineare di colori che mappa un singolo valore numerico a un colore particolare. Due di queste mappature sono sempre attive. La **palette frac** mappa un valore frazionario tra 0 e 1 sull'intera l'intera gamma della palette di colori. La **palette cb** mappa la gamma dell'asse di colori sulla stessa palette. Vedere **set cbrange (p. 236)**. Vedere anche **set colorbox (p. 146)**. Si può usare una qualsiasi di queste mappature per selezionare un colore dalla palette corrente.

La "palette z" mappa il valore z di ciascun segmento o elemento del grafico nel cbrange mapping della palette. Ciò consente una variazione uniforme del colore lungo una linea o una superficie 3D. Permette anche di colorare i grafici 2D con valori della palette letti da una colonna extra di dati (non tutti gli stili di grafico 2D permettono una colonna aggiuntiva).

Vi sono due specificatori di colore speciali: **bgnd** per il colore dello sfondo e **black** (nero).

Colore dello sfondo

La maggior parte dei terminali permette di impostare un colore dello sfondo specifico per il grafico. Il tipo di linea speciale **bgnd** disegnerà con quel colore, e anche **bgnd** sarà riconosciuto come un colore. Esempi:

```
# Questo cancellerà una sezione del canvas scrivendoci sopra con il
# colore di sfondo
set term wxt background rgb "gray75"
set object 1 rectangle from x0,y0 to x1,y1 fillstyle solid fillcolor bgnd
# Questo disegnerà una linea "invisibile" lungo l'asse x
plot 0 lt bgnd
```

Variabile colore della linea

lc variable dice al programma di usare il valore letto da una delle colonne dei dati di input come un indice di tipo di linea, e di usare il colore appartenente a quel tipo di linea. Questo richiede una colonna aggiuntiva corrispondente nello specificatore **using**. I colori del testo possono essere impostati in modo simile usando **tc variable**.

Esempi:

```
# Usa la terza colonna di dati per assegnare i colori ai singoli punti
plot 'data' using 1:2:3 with points lc variable

# Un singolo file di dati può contenere più set di dati, separati da
# due linee vuote. Ogni set di dati viene assegnato come valore di
# indice (vedere `index`) che può essere recuperato tramite lo
# specificatore `using` `column(-2)`. Vedere `pseudocolumns`. Questo
# esempio usa il valore nella colonna -2 per disegnare ogni set di
# dati con un colore di linea diverso.
plot 'data' using 1:2:(column(-2)) with lines lc variable
```

Variabile rgbcolor

È possibile assegnare un colore diverso a ogni punto di dati, segmento di linea, o etichetta nel proprio grafico. **lc rgbcolor variable** dice al programma di leggere l'informazione del colore RGB per ogni linea nel file di dati. Questo richiede una colonna corrispondente aggiuntiva nello specificatore **using**. La colonna extra viene interpretata come una terna RGB 24 bit packed. Se il valore è fornito direttamente nel file di dati, è più facile darlo come un valore esadecimale (vedere **rgbcolor (p. 50)**). In alternativa, lo specificatore **using** può contenere un'espressione che valuta a un colore RGB a 24 bit, come nell'esempio qui sotto. I colori del testo sono impostati in modo simile usando **tc rgbcolor variable**.

Esempio:

```
# Posiziona punti colorati in 3D alle coordinate x,y,z corrispondenti
# alle loro componenti rosse, verdi e blu
rgb(r,g,b) = 65536 * int(r) + 256 * int(g) + int(b)
splot "data" using 1:2:3:(rgb($1,$2,$3)) with points lc rgb variable
```

Dashtype

Nella versione 5 di gnuplot, il dash pattern (**dashtype**) è una proprietà separata associata a ogni linea, analoga a **linecolor** o **linewidth**. Non è necessario mettere il terminale attuale in una modalità speciale solamente per disegnare delle linee tratteggiate. Cioè il comando **set term** <**termname**> **{solid|dashed}** adesso viene ignorato. Se è richiesta la compatibilità con vecchi script scritti per la versione 4, si possono invece usare le seguenti linee:

```
if (GPVAL_VERSION >= 5.0) set for [i=1:9] linetype i dashtype i
if (GPVAL_VERSION < 5.0) set termoption dashed
```

Tutte le linee hanno la proprietà **dashtype solid** a meno che non si specifichi diversamente. Si può cambiare il default per un tipo di linea particolare usando il comando **set linetype** in modo che influisca su tutti i comandi successivi, o è possibile includere il dashtype desiderato come parte del comado plot o di un altro comando.

Sintassi:

```
dashtype N          # dashtype predefinito invocato da numero
dashtype "pattern"  # stringa contenente una combinazione dei caratteri
                    # punto (.) trattino (-) underscore(_) e spazio.
dashtype (s1,e1,s2,e2,s3,e3,s4,e4)  # dash pattern specificato da 1 a 4
                    # coppie numeriche <solid length>, <emptyspace length>
```

Esempio:

```
# Due funzioni che usano il linetype 1 ma si distinguono per il dashtype
plot f1(x) with lines lt 1 dt solid, f2(x) with lines lt 1 dt 3
```

Alcuni terminali supportano dash pattern definiti dagli utenti in aggiunta a qualsiasi set predefinito di dash pattern che offrono.

Esempi:

```
plot f(x) dt 3            # usa dash pattern 3 specifico del terminale
plot f(x) dt ".. "        # costruisce un dash pattern sul momento
plot f(x) dt (2,5,2,15)   # rappresentazione numerica dello stesso pattern
set dashtype 11 (2,4,4,7) # definisce un nuovo dashtype da chiamare per indice
plot f(x) dt 11           # plotta usando il nuovo dashtype
```

Se si specifica un dash pattern usando una stringa, il programma lo convertirà in una sequenza di coppie <solid>,<empty> (pieno e vuoto). Il punto "." diventa (2,5), il trattino "-" diventa (10,10), l'underscore "_" diventa (20,10), e ogni carattere di spazio " " aggiunge 10 al precedente valore <empty>. Il comando **show dashtype** mostrerà sia la stringa originale che la sequenza numerica convertita.

Stili di linea vs tipi di linea

Un **linestyle** (stile di linea) è un'associazione temporanea delle proprietà linecolor, linewidth (spessore linea), dashtype, e pointtype (tipo di punto). Viene definito usando il comando **set style line**. Una volta definito uno stile di linea, lo si può usare in un comando plot per controllare l'aspetto di uno o più elementi del grafico. In altre parole, è proprio come un tipo di linea tranne che per la sua durata. Mentre i **linetype** sono permanenti (durano finché non li si ridefinisce esplicitamente), i **linestyle** durano fino al prossimo reset dello stato della grafica.

Esempi:

```
# definisce un nuovo linestyle con colore ciano indipendente dal terminale,
# linewidth 3, e point type associato 6 (un cerchio con un punto dentro).
set style line 5 lt rgb "cyan" lw 3 pt 6
plot sin(x) with linespoints ls 5          # linestyle 5 definito dall'utente
```

Layers

Un grafico di gnuplot è costruito disegnando i suoi vari componenti in un ordine fisso. Questo ordine può essere modificato assegnando alcuni componenti a un layer (livello) specifico usando le keyword **behind**, **back**, o **front**. Per esempio, per sostituire il colore dello sfondo dell'area del grafico si potrebbe definire un rettangolo colorato con l'attributo **behind**.

```
set object 1 rectangle from graph 0,0 to graph 1,1 fc rgb "gray" behind
```

L'ordine di disegno è

```
behind
back
il grafico in sé
la legenda del grafico (‘key‘)
front
```

All'interno di ciascun layer gli elementi sono disegnati nell'ordine

```
elementi griglia, assi e bordi
pixmap in ordine numerico
oggetti (rettangoli, cerchi, ellissi, poligoni) in ordine numerico
etichette in ordine numerico
frecce in ordine numerico
```

Nel caso di più grafici su una signola pagina (modalità multiplot) questo ordine si applica separamente a ciascun grafico componente, non al multiplot nel suo insieme.

Mouse input

Molti terminali permettono l'interazione con il grafico attuale usando il mouse. Alcuni supportano anche la definizione di tasti di scelta rapida per attivare funzioni predefinite premendo un singolo tasto mentre il focus del mouse è nella finestra attiva del grafico. È anche possibile combinare l'input del mouse con gli script di comandi **batch**, invocando il comando **pause mouse** e poi usando le variabili del mouse restituite dal click del mouse come parametri per le successive azioni scriptate. Vedere **bind (p. 54)** e **variabili mouse (p. 55)**. Vedere anche il comando **set mouse (p. 178)**.

Bind

Sintassi:

```
bind {allwindows} [<key-sequence>] ["<gnuplot commands>"]
bind <key-sequence> ""
reset bind
```

Il **bind** permette di definire o ridefinire un tasto di scelta rapida, cioè una sequenza di comandi di gnuplot che saranno eseguiti quanto verrà premuto un determinato tasto o una sequenza di tasti mentre la finestra del driver ha il focus di input. Notare che **bind** è disponibile solo se gnuplot è stato compilato con supporto **mouse** ed è usato da tutti i terminali dotati di mouse. Un binding specificato dall'utente sostituisce qualsiasi binding incorporato eccetto che <space> e 'q' non possono normalmente essere sostituiti. Per un'eccezione, vedere **bind space (p. 55)**.

Solo il tasto 1 del mouse può essere associato, e solo per i grafici 2D.

La lista di tutti i tasti di scelta rapida si ottiene digitando **show bind** o **bind** o digitando il tasto di scelta rapida 'h' nella finestra del grafico (graph).

Le associazioni dei tasti sono ripristinate al loro stato predefinito da **reset bind**.

Notare che i binding di più tasti con modificatori devono essere dati tra virgolette.

Normalmente i tasti di scelta rapida sono riconosciuti solo quando la finestra del grafico attualmente attiva ha il focus. **bind allwindows <key> ...** (forma breve: **bind all <key> ...**) fa sì che il collegamento per <key> si applichi a tutte le finestre di grafico di gnuplot, attive o no. In questo caso la variabile di gnuplot MOUSE_KEY_WINDOW è impostata all'ID della finestra di origine e può essere usata dal comando bound.

Esempi:

- imposta binding:

```
bind a "replot"
bind "ctrl-a" "plot x*x"
bind "ctrl-alt-a" 'print "great"'
bind Home "set view 60,30; replot"
bind all Home 'print "This is window ",MOUSE_KEY_WINDOW'
```

- mostra binding:

```
bind "ctrl-a"          # mostra il binding per ctrl-a
bind                   # mostra tutti i binding
show bind              # mostrare tutti i binding
```

- rimuovi binding:

```
bind "ctrl-alt-a" ""   # rimuove il binding per ctrl-alt-a
                         (notare che le build-in non possono essere rimosse)
reset bind             # installa i binding predefiniti (integrati)
```

- associa un tasto per attivare qualcosa:

```
v=0
bind "ctrl-r" "v=v+1;if(v%2)set term x11 noraise; else set term x11 raise"
```

I modificatori (ctrl / alt) non sono sensibili alle maiuscole e alle minuscole, i tasti sì:

```
ctrl-alt-a == CtRl-alT-a
ctrl-alt-a != ctrl-alt-A
```

Elenco dei modificatori (alt == meta):

```
ctrl, alt, shift (validi solo per Button1 Button2 Button3)
```

Elenco dei tasti speciali supportati:

```
"BackSpace", "Tab", "Linefeed", "Clear", "Return", "Pause", "Scroll_Lock",
"Sys_Req", "Escape", "Delete", "Home", "Left", "Up", "Right", "Down",
"PageUp", "PageDown", "End", "Begin",

"KP_Space", "KP_Tab", "KP_Enter", "KP_F1", "KP_F2", "KP_F3", "KP_F4",
"KP_Home", "KP_Left", "KP_Up", "KP_Right", "KP_Down", "KP_PageUp",
"KP_PageDown", "KP_End", "KP_Begin", "KP_Insert", "KP_Delete", "KP_Equal",
"KP_Multiply", "KP_Add", "KP_Separator", "KP_Subtract", "KP_Decimal",
"KP_Divide",

"KP_1" - "KP_9", "F1" - "F12"
```

I seguenti sono eventi della finestra piuttosto che tasti effettivi

```
"Button1" "Button2" "Button3" "Close"
```

Vedere anche aiuto per **mouse (p. 178)**.

Bind space

Se gnuplot è stato costruito con l'opzione di configurazione –enable-raise-console, poi digitando <space> nella finestra del grafico si apre la finestra di comando di gnuplot. Questo tasto di scelta rapida può essere cambiato in ctrl-spazio avviando gnuplot come 'gnuplot -ctrlq', o impostando la XResource 'gnuplot*ctrlq'. Vedere **x11 command-line-options (p. 313)**.

Variabili mouse

Quando **mousing** è attivo, cliccando sulla finestra attiva si imposteranno diverse variabili utente a cui si può accedere dalla linea di comando di gnuplot. Le coordinate del mouse al momento del click sono memorizzate in MOUSE_X MOUSE_Y MOUSE_X2 e MOUSE_Y2. Il tasto del mouse cliccato, e qualsiasi meta tasto attivo in quel momento, vengono memorizzati in MOUSE_BUTTON MOUSE_SHIFT MOUSE_ALT e MOUSE_CTRL. Queste variabili sono impostate come indefinite all'inizio di ogni grafico, e diventano definite solo nel caso di un click del mouse nella finestra attiva del grafico. Per determinare da uno script se il mouse è stato cliccato nella finestra attiva del grafico, è sufficente verificare che una qualsiasi di queste variabili sia definita.

```
plot 'something'
pause mouse
if (exists("MOUSE_BUTTON")) call 'something_else'; \
else print "No mouse click."
```

È anche possibile tracciare la pressione dei tasti nella finestra del grafico usando il codice del mouse.

```
plot 'something'
pause mouse keypress
print "Keystroke ", MOUSE_KEY, " at ", MOUSE_X, " ", MOUSE_Y
```

Quando **pause mouse keypress** è terminato con la pressione di un tasto, allora MOUSE_KEY conterrà il valore del carattere ascii del tasto che è stato premuto. MOUSE_CHAR conterrà il carattere stesso come una variabile stringa. Se il comando pausa viene terminato in modo anomalo (ad es. da ctrl-C o chiudendo esternamente la finestra del grafico) allora MOUSE_KEY sarà uguale a -1.

Notare che dopo uno zoom con il mouse, è possibile leggere i nuovi intervalli come GPVAL_X_MIN, GPVAL_X_MAX, GPVAL_Y_MIN, e GPVAL_Y_MAX, vedere **variabili definite da gnuplot (p. 44)**.

Persist

Molti terminali gnuplot (aqua, pm, qt, x11, windows, wxt, ...) aprono finestre di visualizzazione separate sullo schermo in cui vengono disegnati i grafici. L'opzione **persist** dice a gnuplot di lasciare aperte queste finestre quando il programma principale termina. Non ha effetto su terminali output non interattivi. Per esempio se si emette il comando

```
gnuplot -persist -e 'plot [-5:5] sinh(x)'
```

gnuplot aprirà una finestra di display, vi disegnerà il grafico, e poi terminerà, lasciando la finestra di display, che contiene il grafico, sullo schermo. Si può anche specificare **persist** o **nopersist** quando si imposta un nuovo terminale.

```
set term qt persist size 700,500
```

A seconda del tipo di terminale, alcune operazioni di spostamento del mouse potrebbero ancora essere possibili nella finestra persistente. Tuttavia, operazioni come zoom/unzoom che richiedono di ridisegnare il grafico non sono possibili perchè il programma principale è terminato. Se si vuole lasciare la finestra del grafico aperta e completamente utilizzabile con il mouse dopo aver creato il grafico, per esempio quando si esegue gnuplot da un file script piuttosto che interattivamente, vedere **pause mouse close (p. 107)**.

Plotting

Vi sono quattro comandi **gnuplot** che creano effetivamente un grafico: **plot**, **splot**, **replot**, and **refresh**. Gli altri comandi controllano il layout, stile, e contenuti del grafico che verrà eventualmente realizzato. **plot** genera grafici 2D. **splot** genera grafici 3D (in realtà proiezioni 2D, naturalmente). **replot** riesegue i precedenti comandi **plot** or **splot**. **refresh** è simile a **replot** ma riutilizza qualsiasi dato precedentemente memorizzato piuttosto che rileggere i dati da un file o da un flusso di input.

Ogni volta che viene emesso uno di questi quattro comandi, esso ridisegnerà la schermata o genererà una nuova pagina di output contenente tutti gli assi, etichette, titoli attualmente definiti e tutte le varie funzioni o sorgenti di dati elencate nel comando plot originale. Se invece è necessario mettere diversi grafici completi uno accanto all'altro sulla stessa pagina, ad es. per creare un pannello di sotto-figure o per inserire un piccolo grafico dentro un grafico più grande, bisogna usare il comando **set multiplot** per sopprimere la generazione di una nuova pagina per ogni comando plot.

Molte delle informazioni generali sul plotting possono essere trovate nella discussione di **plot**; informazioni specifiche per il 3D possono essere trovate nella sezione **splot**.

plot opera in coordinate rettangolari o polari – vedere **set polar (p. 202)**. **splot** opera in coordinate cartesiane, ma accetterà coordinate azimutali o cilindriche in input. Vedere **set mapping (p. 175)**. **plot** permette anche di utilizzare ciascuno dei quattro bordi – x (inferiore), x2 (superiore), y (sinistra) and y2 (destra) – come assi indipendenti. L'opzione **axes** permette di scegliere su quale coppia di assi verrà plottata una data funzione o serie di dati. Esiste una serie di comandi **set** che da il controllo completo sulle proporzioni

e sull'etichettatura di ogni asse. Alcuni comandi hanno il nome di un asse incorporato nei loro nomi, come **set xlabel**. Altri comandi hanno uno o più nomi di assi come opzioni, come ad esempio **set logscale xy**. Comandi e opzioni che controllano l'asse z non hanno effetto su i grafici 2D.

splot può plottare superfici e contorni oltre a punti e/o linee. Vedere **set isosamples (p. 164)** per informazioni sulla definizione della griglia di una funzione 3D. Vedere **splot datafile (p. 238)** per informazioni riguardanti la struttura di file necessaria per i dati 3D. Per i contorni vedere **set contour (p. 147)**, **set cntrlabel (p. 144)**, e **set cntrparam (p. 144)**.

In **splot**, il controllo sulle scale e sulle etichette degli assi è lo stesso che con **plot**, eccetto che è presente anche un asse z e l'etichettatura degli assi x2 e y2 è possibile solo per i grafici pseudo-2D creati usando **set view map**.

Plugins

L'insieme delle funzioni disponibili per plottare o per la valutare delle espressioni può essere ampliato tramite un meccanismo di plugin che importa funzioni eseguibili da una libreria condivisa. Per esempio, le versioni di gnuplot fino alla 5.4 non forniscono un'implementazione incorporata della funzione gamma incompleta superiore Q(a,x).
$Q(a,x) = \frac{1}{\Gamma(x)} \int_x^\infty t^{a-1} e^{-t} dt$.

Si potrebbe definire un'approssimazione direttamente in gnuplot come questa:

```
Q(a,x) = 1. - igamma(a,x)
```

Tuttavia questo ha una precisione intrinsecamente limitata quando il valore di Q si avvicina a 1. Se si ha bisogno di un'implementazione più accurata, sarebbe meglio fornirne una tramite un plugin (vedi sotto). Una volta importata, la funzione può essere usata come qualsiasi altra funzione incorporata o definita dall'utente in gnuplot. Vedere **import (p. 105)**.

La distribuzione di gnuplot include il codice sorgente e le istruzioni per creare una libreria di plugin nella directory demo/plugin. È possibile modificare il semplice file di esempio **demo_plugin.c** sostituendo una o più funzioni di esempio con un'implementazione della funzione alla quale si è interessati. Ciò potrebbe includere l'invocazione di funzioni da una libreria matematica esterna.

La directory demo/plugin contiene anche la sorgente per un plugin che implementa Q(a,x). Come notato sopra, questo plugin permette alle versioni precedenti di gnuplot di fornire la stessa funzione **uigamma** dell'attuale versione di sviluppo.

```
import Q(a,x) from "uigamma_plugin"
uigamma(a,x) = ((x<1 || x<a) ? 1.0-igamma(a,x) : Q(a,x))
```

Start-up (inizializzazione)

Quando gnuplot viene eseguito, prima cerca un file di inizializzazione a livello di sistema **gnuplotrc**. La posizione di questo file è determinata quando il programma è costruito ed è riportata da **show loadpath**. Il programma cerca quindi nella directory HOME dell'utente un file chiamato **.gnuplot** su sistemi simili a Unix, o **GNUPLOT.INI** su altri sistemi. (OS/2 lo cercherà nella directory nominata nella variabile d'ambiente **GNUPLOT**; Windows userà **APPDATA**). Nota: Il programma può essere configurato per cercare prima nella directory attuale, ma questo non è consigliato perché è una cattiva pratica di sicurezza.

Costanti stringa, variabili stringa, e funzioni stringa

Oltre alle costanti stringa, la maggior parte dei comandi di gnuplot accetta anche una variabile stringa, un'espressione stringa o una funzione che restituisce una stringa. Per esempio, i quattro metodi seguenti per creare un grafico hanno tutti come risultato lo stesso titolo plot:

```
four = "4"
graph4 = "Title for plot #4"
graph(n) = sprintf("Title for plot #%d",n)

plot 'data.4' title "Title for plot #4"
plot 'data.4' title graph4
plot 'data.4' title "Title for plot #".four
plot 'data.4' title graph(4)
```

Dato che gli interi sono promossi a stringhe quando vengono elaborati dall'operatore di concatenazione di stringhe (carattere '.'), funziona anche il metodo seguente:

```
N = 4
plot 'data.'.N title "Title for plot #".N
```

In generale, gli elementi sulla linea di comando saranno valutati a possibili variabili stringa solo se non sono altrimenti riconoscibili come parte della normale sintassi di gnuplot. Quindi la seguente sequenza di comandi è corretta, anche se probabilemte dovrebbe essere evitata per non creare confusione:

```
plot = "my_datafile.dat"
title = "My Title"
plot plot title title
```

Sottostringhe

Le sottostringhe possono essere specificate aggiungendo uno specificatore di intervallo a qualsiasi stringa, variabile stringa o funzione con valore stringa. Lo specificatore di intervallo ha la forma [begin:end], dove begin è l'indice del primo carattere della sottostringa ed end è l'indice dell'ultimo carattere della sottostringa. Il primo carattere ha come indice 1. I campi begin o end possono essere vuoti o contenere '*', per indicare il vero inizio o la vera fine fine della stringa originale. Ad es. str[:] e str[*:*] descrivono entrambi l'intera stringa str.

Operatori di stringa

Tre operatori binari necessitano operandi di stringa: l'operatore di concatenazione delle stringhe ".", l'operatore di uguaglianza delle stringhe "eq" e l'operatore di disuguaglianza delle stringhe "ne". L'esempio seguente stamperà TRUE.

```
if ("A"."B" eq "AB") print "TRUE"
```

Funzioni stringa

Gnuplot fornisce diverse funzioni incorporate che operano su stringhe. Funzioni di formattazione generale: vedere **gprintf (p. 158)** **sprintf (p. 40)**. Funzioni di formattazione temporali: vedere **strftime (p. 40)** **strptime (p. 40)**. Manipolazione delle stringhe: vedere **substr (p. 40)** **strstrt (p. 40)** **trim (p. 41)** **word (p. 41)** **words (p. 41)**.

Codifica delle stringhe

Le funzioni di manipolazione delle stringhe incorporate in gnuplot sono sensibili alla codifica utf-8 (vedere **set encoding (p. 154)**). Per esempio

```
utf8string = "αβγ"
strlen(utf8string) returns 3 (number of characters, not number of bytes)
utf8string[2:2] evaluates to "β"
strstrt(utf8string,"β") evaluates to 2
```

Sostituzione e linea di comando delle macro

Quando una linea di comando per gnuplot viene letta per la prima volta, cioè prima che venga interpretata o eseguita, vengono eseguite due forme di sostituzione lessicale. Queste sono attivate dalla presenza di testo tra virgolette (carattere ascii 96) o preceduto da @ (carattere ascii 64).

Sostituzione dei comandi di sistema tra backquote

La sostituzione a linea di comando è specificata da un comando di sistema racchiuso tra backquote. Questo comando viene generato e l'output che produce sostituisce il testo tra backquote sulla linea di comando. Lo stato di uscita del comando di sistema viene restituito nelle variabili GPVAL_SYSTEM_ERRNO e GPVAL_SYSTEM_ERRMSG.

CHANGE (differisce dalle versioni da 4 a 5.2): Carriage-return interno ('\r') e i caratteri newline ('\n') non sono tolti dallo stream di input durante la sostituzione. Questo cambiamento porta la sostituzione di backquote in linea con la funzione system().

La sostituzione della linea di comando può essere usata in qualsiasi punto della linea di comando **gnuplot** tranne che all'interno di stringhe delimitate da virgolette singole .

Esempio:

Questo eseguirà il programma **leastsq** e sostituirà **leastsq** (inclusi i backquote) sulla linea di comando con il suo output:

```
f(x) = `leastsq`
```

oppure, in VMS

```
f(x) = `run leastsq`
```

Questi genereranno etichette con l'orario corrente e l'ID utente:

```
set label "generated on `date +%Y-%m-%d` by `whoami`" at 1,1
set timestamp "generated on %Y-%m-%d by `whoami`"
```

Sostituzione di variabili stringa come macro

Il carattere @ è usato per attivare la sostituzione del valore attuale di una variabile stringa nella linea di comando. Il testo nella variabile stringa può contenere qualsiasi numero di elementi lessicali. Questo permette alle variabili di stringa di essere usate come macro della linea di comando. Solo le costanti stringa possono essere espanse usando questo meccanismo, non le espressioni con valore di stringa. Per esempio:

```
style1 = "lines lt 4 lw 2"
style2 = "points lt 3 pt 5 ps 2"
range1 = "using 1:3"
range2 = "using 1:5"
plot "foo" @range1 with @style1, "bar" @range2 with @style2
```

La linea che contiene simboli @ è espansa in input, in modo che nel momento in cui viene eseguita, l'effetto è identico a quello di aver digitato per intero

```
plot "foo" using 1:3 with lines lt 4 lw 2, \
     "bar" using 1:5 with points lt 3 pt 5 ps 2
```

La funzione exists() può essere utile in relazione alla valutazione macro. L'esempio seguente verifica che C può essere tranquillamente espanso come nome di una variabile definita dall'utente:

```
C = "pi"
if (exists(C)) print C," = ", @C
```

L'espansione delle macro non avviene all'interno di virgolette singole o doppie. Tuttavia l'espansione delle macro avviene all'interno dei backquote.

L'espansione delle macro è gestita come la prima cosa che l'interprete fa guardando una nuova linea di comandi e viene fatta solo una volta. Pertanto, codici come il seguente verranno eseguiti correttamente:

```
A = "c=1"
@A
```

ma questa linea non lo sarà, poiché la macro è definita sulla stessa linea e non sarà ampliata in tempo

```
A = "c=1"; @A   # non si espande a c=1
```

L'espansione delle macro all'interno di un'iterazione tra parentesi si verifica prima che il loop venga eseguito; cioè, @A si comporterà sempre come il valore originale di A anche se A stesso viene riassegnato all'interno del loop.

Per l'esecuzione di comandi completi può essere utile anche il comando **evaluate**.

String variables, macros, and command line substitution

L'interazione di variabili stringa, backquote e sostituzione delle macro è un po' complicata. I backquote non fermano la sostituzione delle macro, quindi

```
filename = "mydata.inp"
lines = ‘ wc --lines @filename | sed "s/ .*//" ‘
```

fa sì che il numero di linee in mydata.inp sia memorizzato nelle linee variabili intere. E le virgolette doppie non fermano la sostituzione dei backquote, quindi

```
mycomputer = "‘uname -n‘"
```

fa sì che la stringa restituita dal comando di sistema **uname -n** sia memorizzata nella variabile stringa mycomputer.

Tuttavia, la sostituzione delle macro non avviene all'interno delle virgolette doppie, quindi non si può definire un comando di sistema come una macro e poi usare la sostituzione sia delle macro che dei backquote allo stesso tempo.

```
machine_id = "uname -n"
mycomputer = "‘@machine_id‘"  # non funziona!!
```

Questo non funziona perché le virgolette doppie impediscono che @machine_id sia interpretato come una macro. Per memorizzare un comando di sistema come una macro ed eseguirlo in seguito è necessario invece includere i backquote come parte della macro stessa. Ciò è possibile definendo la macro come mostrato di seguito. Notare che il formato sprintf annida tutti e tre i tipi di virgolette.

```
machine_id = sprintf('"‘uname -n‘"')
mycomputer = @machine_id
```

Sintassi

Le opzioni e gli eventuali parametri di accompagnamento sono separati da spazi, mentre le liste e le coordinate sono separate da virgole. Gli intervalli sono separati da due punti e racchiusi tra parentesi [], testo e nomi di file sono racchiusi tra virgolette, e un po' di cose varie sono racchiuse tra parentesi.

Le virgole sono usate per separare le coordinate nei comandi **set**: **arrow**, **key**, e **label**; la lista delle variabili da adattare (la lista dopo la keyword **via** del comando **fit**); liste di contorni discreti o i parametri del loop che

li specificano sul comando **set cntrparam**; gli argomenti dei comandi **set**: **dgrid3d**, **dummy**, **isosamples**, **offsets**, **origin**, **samples**, **size**, **time**, e **view**; le liste di tic o i parametri del loop che le specificano; gli offset per le etichette dei titoli e degli assi; funzioni parametriche da usare per calcolare le coordinate x, y e z sui comandi **plot**, **replot** e **splot**; e i set completi di keyword che specificano singoli grafici (insiemi di dati o funzioni) sui comandi **plot**, **replot** e **splot**.

Le parentesi sono usate per delimitare set di tic espliciti (al contrario dei parametri del loop) e per indicare computazioni nel filtro **using** dei comandi **fit**, **plot**, **replot** e **splot**.

(Le parentesi e le virgole sono anche usate come al solito nella notazione delle funzioni.)

Le parentesi quadre sono usate per delimitare gli intervalli dati nei comandi **set**, **plot** o **splot**.

I due punti sono usati per separare gli estremi nelle specifiche **range** (se sono date nei comandi **set**, **plot** o **splot**) e per separare le entry nel filtro **using** dei comandi **plot**, **replot**, **splot** e **fit**.

I punto e virgola sono usati per separare i comandi dati su una singola linea di comando.

Le parentesi graffe sono usate nella sintassi per la modalità di testo avanzato e per delimitare i blocchi nelle dichiarazioni if/then/else. Si usano anche per indicare i numeri complessi: {3,2} = 3 + 2i.

Virgolette

Gnuplot usa tre tipi di virgolette per delimitare stringhe di testo, le virgolette doppie (ascii 34), le virgolette singole (ascii 39), e i backquote (ascii 96).

I nomi dei file possono essere inseriti con virgolette singole o doppie. In questo manuale, gli esempi di comandi in genere riportano per chiarezza i nomi dei file tra virgolette singole e altri token di stringa tra virgolette doppie.

Le costanti stringa e le stringhe di testo usate per etichette, titoli o altri elementi del grafico possono essere racchiuse tra virgolette singole o doppie. Ulteriori elaborazioni del testo tra virgolette dipendono dalla scelta del tipo di virgolette.

L'elaborazione del backslash dei caratteri speciali come \n (newline) e \345 (codice carattere ottale) viene eseguita solo per strighe racchiuse tra virgolette doppie. Nelle stringhe tra virgolette singole, i backslash sono semplicemente caratteri normali. Per mettere una virgoletta singola (ascii 39) in una stringa tra virgolette singole, essa deve essere raddoppiata. Quindi le stringhe "d\" s' b\\" e 'd" s'' b\' sono del tutto equivalenti.

L'allineamento del testo è uguale per ogni linea di una stringa multilinea. Quindi la stringa allineata al centro

```
"This is the first line of text.\nThis is the second line."
```

produrrà

```
This is the first line of text.
   This is the second line.
```

ma

```
'This is the first line of text.\nThis is the second line.'
```

produrrà

```
This is the first line of text.\nThis is the second line.
```

L'elaborazione del testo avanzato viene eseguita sia per il testo tra virgolette doppie che testo tra virgolette singole, ma solo dai terminali che supportano questa modalità. Vedere **enhanced text (p. 35)**.

I backquote sono usati per racchiudere i comandi di sistema per la sostituzione nella linea di comando. Vedere **substitution (p. 59)**.

Dati ora/data

gnuplot supporta l'uso di informazioni sull'ora e/o data come dati di input. Questa funzionalità viene attivata dai comandi **set xdata time**, **set ydata time**, etc.

Internamente tutti gli orari e le date sono convertiti in numero di secondi dall'anno 1970. Il comando **set timefmt** definisce il formato predefinito per tutti gli input: file di dati, intervalli, tic (tacche), posizioni dell'etichetta – qualsiasi cosa che accetti un un valore di dati temporali lo riceve per default in questo formato. Solo un formato predefinito può essere in vigore in un dato momento. Quindi, se entrambi i dati x e y in un file sono ora/data, per default sono interpretati nello stesso formato. Tuttavia questo default può essere sostituito quando si legge un particolare file o colonna di input usando la funzione **timecolumn** nel corrispondente specificatore **using**.

La conversione verso e da i secondi presuppone il Tempo Universale (che è lo stesso del tempo standard di Greenwich). Non c'è nessuna disposizione per cambiare il fuso orario o per l'ora legale. Se tutti i propri dati fanno riferimento allo stesso fuso orario (e sono tutti diurni o standard) non occorre preoccuparsi di queste cose. Ma se il tempo assoluto è cruciale per la propria applicazione, è necessario convertire in UT da soli.

Comandi come **show xrange** reinterpreteranno il numero intero secondo **timefmt**. Se si cambia **timefmt**, e poi si usa il comando **show** per mostrare di nuovo la quantità, essa sarà visualizzata nel nuovo **timefmt**. Per questo motivo, se si resetta il flag del tipo di dati per quell'asse (per esempio **set xdata**), la quantità sarà mostrata nella sua forma numerica.

I comandi **set format** o **set tics format** definiscono il formato che sarà usato per le etichette dei tic, se l'input per l'asse specificato è ora/data o meno.

Se l'informazione ora/data deve essere plottata da un file, l'opzione **using** *deve* essere usata sul comando **plot** o **splot**. Questi comandi usano semplicemente lo spazio bianco per separare le colonne, ma lo spazio bianco può essere incorporato all'interno della stringa ora/data. Se si usa tab come separatore, potrebbe essere necessario un po' di tentativi ed errori per scoprire come il proprio sistema li gestisce.

La funzione **time** può essere usata per ottenere l'ora corrente del sistema. Questo valore può essere convertito in una stringa di date con la funzione **strftime**, o può essere usato in concomitanza con **timecolumn** per generare grafici ora/data relativi. Il tipo di argomento determina cosa viene restituito. Se l'argomento è un numero intero, **time** restituisce il tempo attuale come un intero, in secondi dal 1 gennaio 1970. Se l'argomento è un numero reale (o complesso), il risultato sarà anch'esso reale. La precisione della parte frazionale (inferiore a 1 secondo) dipende dal proprio sistema operativo. Se l'argomento è una stringa, si presume che sia una format string, ed è passata a **strftime** per fornire una stringa ora/data formattata.

L'esempio seguente dimostra il plotting di ora/data.

Supponiamo che il file "dati" contenga record come

```
03/21/95 10:00  6.02e23
```

Questo file può essere plottato da

```
set xdata time
set timefmt "%m/%d/%y"
set xrange ["03/21/95":"03/22/95"]
set format x "%m/%d"
set timefmt "%m/%d/%y %H:%M"
plot "data" using 1:3
```

il quale produrà etichette xtic che assomigliano a "03/21".

Gnuplot traccia il tempo con precisione al millisecondo. I formati di tempo sono stati modificati per adattarsi a questo. Esempio: stampa l'ora corrente con precisione in msec

```
print strftime("%H:%M:%.3S %d-%b-%Y",time(0.0))
18:15:04.253 16-Apr-2011
```

Vedere **time_specifiers (p. 159)**.

Part II
Stili di plotting

Sono disponibili molti stili di plotting in gnuplot. Sono elencati in ordine alfabetico qui sotto. I comandi **set style data** e **set style function** cambiano lo stile di plotting predefinito per i successivi comandi **plot** e **splot**.

Si può anche specificare espressamente lo stile del grafico come parte del comando del comando **plot** o **splot**. Se si desidera combinare stili di grafici diversi all'interno di un solo grafico, si deve specificare lo stile di grafico per ogni componente.

Esempio:

```
plot 'data' with boxes, sin(x) with lines
```

Ogni stile di grafico ha il proprio set di entry di dati previsto in un file di dati. Per esempio, per default lo stile **lines** si aspetta o una singola colonna di valori y (con ordine x implicito) o una coppia di colonne con x nella prima e y nella seconda. Per maggiori informazioni su come mettere a punto il modo in cui le colonne in un file vengono interpretate come dati del grafico, vedere **using (p. 121)**.

Arrows (frecce)

Lo stile 2D **arrows** disegna una freccia con lunghezza e angolo di orientamento specificati in ogni punto (x,y). Colonne di input aggiuntive potranno essere usate per fornire informazioni variabili (per-datapoint) sul colore o sullo stile della freccia. Lo stile **arrows** è identico allo stile 2D **with vectors** eccetto che la punta della freccia è posizionata usando la lunghezza + angolo piuttosto che delta_x + delta_y. Vedere **with vectors (p. 84)**.

```
4 columns:  x  y  length  angle
```

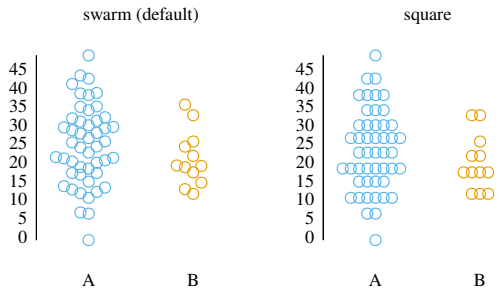
Le keyword **with arrows** possono essere seguite da proprietà di stile freccia in linea, un riferimento a uno stile freccia predefinito, o **arrowstyle variable** per caricare l'indice dello stile freccia desiderato per ogni freccia da una colonna separata.

length > 0 (lunghezza) è interpretata nelle coordinate dell'asse x. -1 $<$ **length** < 0 è interpretata in coordinate del grafico (graph) orizzontali; cioè |length| è una frazione della larghezza totale del grafico. Il programma aggiusterà le differenze di ridimensionamento di x e y o il formato del grafico, in modo che la lunghezza visiva sia indipendente dall'angolo di orientamento.

angle (angolo) è sempre specificato in gradi.

Bee swarm plots (grafici a sciame d'api)

I grafici "bee swarm" risultano dall'applicazione del jitter per separare i punti sovrapposti. Un uso tipico è quello di confrontare la distribuzione dei valori y esibiti da due o più categorie di punti, dove la categoria determina la coordinata x. Vedere il comando **set jitter (p. 165)** per come controllare i criteri di sovrapposizione e il modello di spostamento usato per il jittering. I grafici in figura sono stati creati con lo stesso comando plot ma con diverse impostazioni di jitter



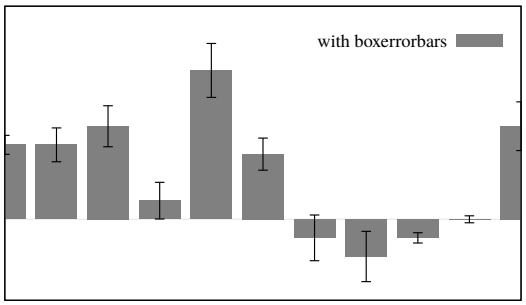

```
set jitter
plot $data using 1:2:1 with points lc variable
```

Boxerrorbars

Lo stile **boxerrorbars** è rilevante solo per il plotting di dati 2D. È una combinazione degli stili **boxes** (caselle) e **yerrorbars**. Ha bisogno di 3, 4, o 5 colonne di dati. Un'ulteriore (quarta, quinta o sesta) colonna di input può essere usata per fornire informazioni variabili (per-datapoint) sul colore (vedere **linecolor (p. 50)** e **rgbcolor variable (p. 52)**). La barra di errore sarà disegnata nello stesso colore del bordo della casella.

```
3 columns:  x  y  ydelta
4 columns:  x  y  ydelta xdelta        # boxwidth != -2
4 columns:  x  y  ylow  yhigh          # boxwidth == -2
5 columns:  x  y  ylow  yhigh  xdelta
```

La boxwidth (larghezza della casella) proverrà dalla quarta colonna se gli errori y sono dati come "ydelta" e la larghezza della casella non è stata precedentemente impostata a -2.0 (**set boxwidth -2.0**), o proverrà dalla quinta colonna se gli errori y sono nella forma di "ylow yhigh". Il caso speciale **boxwidth = -2.0** è per dati a quattro colonne con errori y nella forma forma "ylow yhigh". In questo caso la larghezza della casella sarà calcolata in modo tale che ogni casella tocchi i box adiacenti. La larghezza sarà anche calcolata nei casi in cui vengono usate 3 colonne di dati.



L'altezza del box è determinata dall'errore y nello stesso modo dello stile **yerrorbars** — o da y-ydelta a y+ydelta o da ylow a yhigh, a seconda di quante colonne di dati sono fornite.

Boxes (Diagramma a barre)

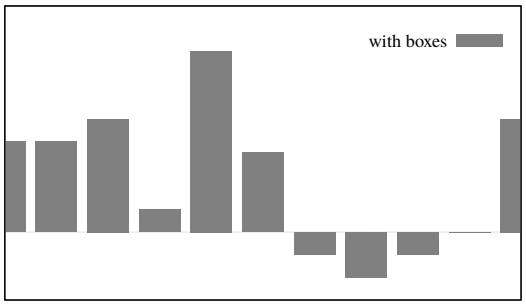
Nei grafici 2D lo stile **boxes** disegna un rettangolo centrato sulla coordinata x data che si estende dall'asse x, cioè da y=0, non dal bordo del grafico (graph), fino alla coordinata y data. La larghezza del box può essere fornita in una colonna di input aggiuntiva o controllata da **set boxwidth**. Altrimenti ogni box si estenderà fino a toccare i box adiacenti.

Nei grafici 3D lo stile **boxes** disegna un box centrato sulla coordinata [x,y] data che si estende dal piano in z=0 alla coordinata z data. La larghezza del box sull'asse x può essere fornita in una colonna di input separata o controllata da **set boxwidth**. La profondità sull'asse y del box è controllata da **set boxdepth**. I box non si espandono automaticamente per toccarsi l'un l'altro come nei grafici 2D.

2D boxes

plot with boxes usa 2 o 3 colonne di dati di base. Colonne di input aggiuntive possono essere usate per fornire informazioni come la linea variabile o il colore di riempimento. Vedere **rgbcolor variable (p. 52)**.

```
2 columns:  x  y
3 columns:  x  y  x_width
```



La larghezza del box si ottiene in uno dei tre modi seguenti. Se i dati di input hanno una terza colonna, questa sarà usata per impostare la larghezza del box. Altrimenti, se è stata impostata una larghezza usando il comando **set boxwidth**, verrà usata quella fornita. Se nessuno di questi è disponibile, la larghezza di ogni box sarà calcolata in modo che tocchi i box adiacenti.

Gli interni dei box sono disegnati usando lo stile di riempimento corrente. In alternativa uno stile di riempimento può essere specificato nel comando plot. Vedere **set style fill (p. 209)**. Se non viene dato alcun colore di riempimento nel comando plot, viene usato il colore della linea attuale.

Esempi:

Per plottare un file di dati con box solidi riempiti e con un piccolo spazio verticale che li separa (bargraph):

```
set boxwidth 0.9 relative
set style fill solid 1.0
plot 'file.dat' with boxes
```

Per plottare una curva di seno e una curva di coseno in stile pattern-filled box:

```
set style fill pattern
plot sin(x) with boxes, cos(x) with boxes
```

Il grafico sin userà il pattern 0; il grafico cos userà il pattern 1. Qualsiasi grafico aggiuntivo scorrerebbe attraverso i pattern supportati dal driver del terminale.

Per specificare esplicitamente gli stili di riempimento e i colori di riempimento per ogni set di dati:

```
plot 'file1' with boxes fs solid 0.25 fc 'cyan', \
     'file2' with boxes fs solid 0.50 fc 'blue', \
     'file3' with boxes fs solid 0.75 fc 'magenta', \
     'file4' with boxes fill pattern 1, \
     'file5' with boxes fill empty
```

3D boxes

splot with boxes necessita almeno 3 colonne di dati di input. Ulteriori colonne di input possono essere usate per fornire informazioni come la larghezza del box o il colore di riempimento.

```
3 columns:  x  y  z
4 columns:  x  y  z  [x_width or color]
5 columns:  x  y  z  x_width  color
```

L'ultima colonna viene usata come colore solo se il comando splot specifica una modalità di variabile colore. Esempi

```
splot 'blue_boxes.dat' using 1:2:3 fc "blue"
splot 'rgb_boxes.dat' using 1:2:3:4 fc rgb variable
splot 'category_boxes.dat' using 1:2:3:4:5 lc variable
```

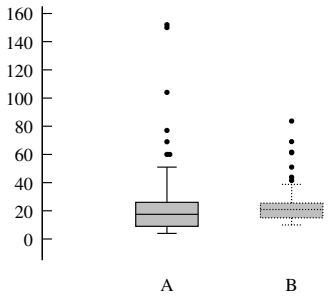
Nel primo esempio tutti i box sono blu e la loro larghezza è stata precedentemente impostata da **set boxwidth**. Nel secondo esempio la larghezza del box è sempre presa da **set boxwidth** perchè la quarta colonna è interpretata come un colore RGB a 24 bit. Il terzo comando di esempio legge la larghezza del box dalla colonna 4 e interpreta il valore nella colonna 5 come un tipo di linea intero da cui viene derivato il colore.

Per default i box non hanno spessore; consistono in un singolo rettangolo parallelo al piano xz alla coordinata y specificata. Si può cambiare questo in un vero box con quattro lati e una parte superiore impostando un'estensione diversa da zero su y. Vedere **set boxdepth (p. 142)**.

I box 3D sono elaborati come quadrangoli pm3d invece che superfici. Per questo motivo l'ordine di disegno front/back (fronte/dietro) non è influenzato da **set hidden3d**. Allo stesso modo se si vuole che ogni faccia del box abbia un bordo si deve usare **set pm3d border** piuttosto che **set style fill border**. Vedere **set pm3d (p. 197)**. Per migliori risultati usare una combinazione di **set pm3d depthorder base** e **set pm3d lighting**.

Boxplot (Diagramma a scatola e baffi)

I boxplot sono un modo comune per rappresentare una distribuzione statistica di valori. I confini del quartile sono determinati in modo tale che 1/4 dei punti abbia un valore uguale o inferiore al confine del primo quartile, 1/2 dei punti abbia un valore uguale o inferiore al valore del secondo quartile (mediana), etc. Un box è disegnato intorno alla ragione tra il primo e il terzo quartile, con una linea orizzontale al valore medio. Dei whiskers (baffi) si estendono dal box fino ai limiti specificati dall'utente. I punti che si trovano al di fuori di questi limiti vengono disegnati individualmente.



Esempi

```
# Posiziona un boxplot alla coordinata x 1.0 che rappresenta i valori y
# nella colonna 5
plot 'data' using (1.0):5

# Stesso grafico, ma sopprime gli outlier e forza la larghezza del boxplot a 0.3
set style boxplot nooutliers
plot 'data' using (1.0):5:(0.3)
```

Per default viene prodotto solo un boxplot che rappresenta tutti i valori y dalla seconda colonna della specifica d'uso. Tuttavia, un'ulteriore (quarta) colonna può essere aggiunta alla specifica. Se presenti, i valori di quella colonna saranno interpretati come i livelli discreti di una variabile fattore. Verranno disegnati tanti boxplot quanti sono i livelli nella variabile fattore. La separazione tra questi boxplot è per default 1.0, ma può essere cambiata da **set style boxplot separation**. Per default, il valore della variabile fattore è indicato come un'etichetta di un tic sotto (o sopra) ogni boxplot.

Esempio

```
# Supponiamo che la colonna 2 di 'data' contenga o "control" o "treatement"
# L'esempio seguente produce due boxplot, uno per ogni livello del fattore
plot 'data' using (1.0):5:(0):2
```

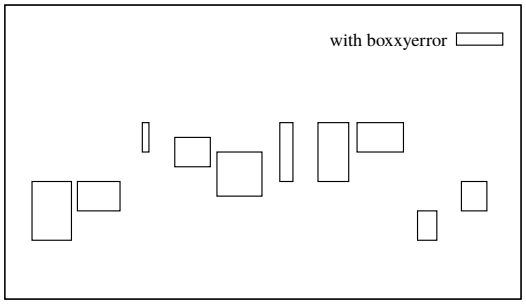
La larghezza predefinita del box può essere impostata da **set boxwidth <width>** o può essere specificata come una terza colonna opzionale nella clausola **using** del comando plot. La prima e la terza colonna (coordinata x e larghezza) sono normalmente fornite come costanti invece che come colonne di dati.

Per default i baffi si estendono dalle estremità del box fino al punto più distante il cui valore y si trova entro 1.5 volte l'intervallo interquartile. Per default gli outlier (valori anomali) sono disegnati come cerchi (tipo di punto 7). La larghezza delle barre alla fine dei baffi può essere controllata usando **set bars** o **set errorbars**.

Queste proprietà predefinite possono essere cambiate usando il comando **set style boxplot**. Vedere **set style boxplot (p. 207)**, **bars (p. 155)**, **boxwidth (p. 142)**, **fillstyle (p. 209)**, **candlesticks (p. 67)**.

Boxxyerror

Lo stile di grafico **boxxyerror** è rilevante solo per plottare dati 2D. È simile allo stile **xyerrorbars** fatta eccezione che disegna aree rettangolari invece che croci. Utilizza 4 o 6 colonne di base di dati di input. Ulteriori colonne di input possono essere usate per fornire informazioni come variabile linea o colore di riempimento (vedere **rgbcolor variable (p. 52)**).




```
4 columns:  x  y  xdelta  ydelta
6 columns:  x  y  xlow  xhigh  ylow  yhigh
```

La larghezza e l'altezza del box sono determinate dagli errori di x e y nello stesso modo dello stile **xyerrorbars** — o da xlow a xhigh e da ylow a yhigh, o da x-xdelta a x+xdelta e da y-ydelta a y+ydelta, a seconda di quante colonne di dati sono fornite.

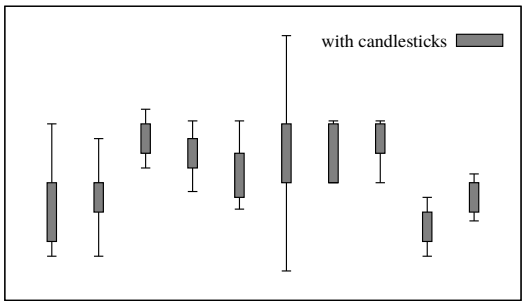
La forma a 6 colonne del comando fornisce un modo conveniente per plottare rettangoli con limiti x e y arbitrari.

Un'ulteriore (quinta o settima) colonna di input può essere usata per fornire informazioni variabili (per-datapoint) sul colore (vedere **linecolor (p. 50)** e **rgbcolor variable (p. 52)**).

L'interno dei box è disegnato secondo stile di riempimento corrente. Vedere **set style fill (p. 209)** e **boxes (p. 64)** per i dettagli. In alternativa un nuovo stile di riempimento può essere specificato nel comando plot.

Candlesticks

Lo stile **candlesticks** può essere usato per il plotting 2D di dati finanziari o per generare grafici box-and-whisker di dati statistici. Il simbolo è un box rettangolare, centrato orizzontalmente alla coordinata x e limitato verticalmente dai prezzi di apertura e di chiusura. Un segmento di linea verticale alla coordinata x si estende verso l'alto dalla parte superiore del rettangolo al prezzo massimo e un altro segmento verso il basso. La linea verticale rimane invariata se il prezzo più basso e il prezzo più alto vengono scambiati.



Sono necessarie cinque colonne di dati base:

```
financial data:   date  open  low  high  close
whisker plot:     x  box_min  whisker_min  whisker_high  box_high
```

La larghezza del rettangolo può essere controllata dal comando **set boxwidth**. Per retrocompatibilità con versioni precedenti di gnuplot, quando il parametro della larghezza del box non è stato impostato, allora la larghezza del rettangolo candlestick è presa da **set errorbars <width>**.

In alternativa, una larghezza esplicita per ogni raggruppamento box-and-whiskers può essere specificata in una sesta colonna opzionale di dati. La larghezza deve essere data nelle stesse unità della coordinata x.

Un'ulteriore (sesta, o settima se la sesta colonna è usata per dati sulla larghezza) colonna di input può essere utilizzata per fornire informazioni variabili (per-datapoint) sul colore (vedere **linecolor (p. 50)** e **rgbcolor variable (p. 52)**).

Per default i segmenti di linea verticale non hanno crossbars (barre trasversali) in alto o in basso. Se si desiderano barre trasversali, che sono usate tipicamente per grafici box-and-whisker, allora bisogna aggiungere la keyword **whiskerbars** al comando plot. Per default queste whiskerbar si estendono per tutta la

larghezza orizzontale del candlestick, ma questo si può modificare specificando una frazione della larghezza totale.

La convenzione abituale per i dati finanziari è che il rettangolo è vuoto se (open < close) e riempito (solid fill) se (close < open). Questo è il comportamento che si ottiene se lo stile di riempimento corrente è impostato su "empty" (vuoto). Vedere **fillstyle (p. 209)**. Se si imposta lo stile di riempimento su solido o pattern, allora questo sarà usato per tutti i box indipendentemente dai valori di chiusura e apertura. Vedere anche **set errorbars (p. 155)** e **financebars (p. 71)**. Vedere anche le demo di `candlestick`

e `finance`

.

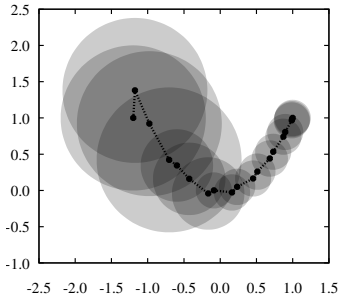
Nota: Per aggiungere altri simboli, come il valore medio, su un grafico box-and-whisker sono necessari ulteriori comandi plot come in questo esempio:

```
# Colonne di dati:X Min 1°Quartile Mediana 3°Quartile Max
set errorbars 4.0
set style fill empty
plot 'stat.dat' using 1:3:2:6:5 with candlesticks title 'Quartiles', \
     ''         using 1:4:4:4:4 with candlesticks lt -1 notitle

# Grafico con barre trasversali sui baffi, le barre trasversali sono il
# 50% della larghezza totale
plot 'stat.dat' using 1:3:2:6:5 with candlesticks whiskerbars 0.5
```

Vedere **set boxwidth (p. 142)**, **set errorbars (p. 155)**, **set style fill (p. 209)**, e **boxplot (p. 66)**.

Circles (Cerchi)

Lo stile **circles** plotta un cerchio con un raggio specifico in ogni data point. Il raggio è sempre interpretato nelle unità dell'asse orizzontale (x or x2) del grafico. La scala su y e le proporzioni del grafico vengono entrambe ignorate. Se il raggio non è dato in una colonna separata per ciascun punto, il raggio viene preso da **set style circle**. In questo caso il raggio potrebbe usare le coordinate del grafico (graph) o dello schermo.



Sono possibili molte combinazioni di proprietà per punto e di proprietà impostate in precedenza. Per i grafici 2D esse includono

```
using x:y
using x:y:radius
using x:y:color
using x:y:radius:color
using x:y:radius:arc_begin:arc_end
using x:y:radius:arc_begin:arc_end:color
```

Un cerchio completo verrà disegnato per default. È possibile invece plottare segmenti di archi specificando un angolo iniziale e uno finale (in gradi) nelle colonne 4 e 5.

Un colore per cerchio può essere fornito nell'ultima colonna dello specificatore d'uso. In questo caso il comando plot deve includere un corrispondente termine variabile colore come **lc variable** o **fillcolor rgb variable**.

Per i grafici 3D lo specificatore d'uso deve contenere

```
splot DATA using x:y:z:radius:color
```

dove la colonna della variabile colore è opzioni. Vedere **set style circle (p. 212)** e **set style fill (p. 209)**.

Esempi:

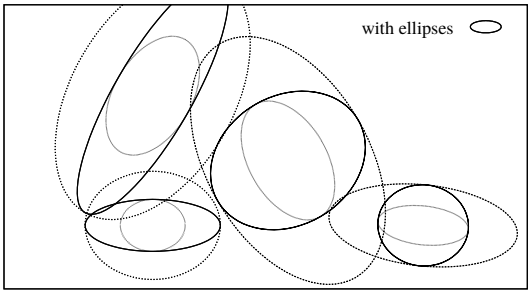
```
# disegna cerchi la cui area è proporzionale al valore nella colonna 3
set style fill transparent solid 0.2 noborder
plot 'data' using 1:2:(sqrt($3)) with circles, \
     'data' using 1:2 with linespoints

# disegna Pac-men invece di cerchi
plot 'data' using 1:2:(10):(40):(320) with circles

# disegna un grafico a torta con dati inline
set xrange [-15:15]
set style fill transparent solid 0.9 noborder
plot '-' using 1:2:3:4:5:6 with circles lc var
0    0    5    0    30    1
0    0    5   30    70    2
0    0    5   70   120    3
0    0    5  120   230    4
0    0    5  230   360    5
e
```

Il risultato è simile a utilizzare un grafico **points** con punti di variabile dimensione e stile di punto 7, eccetto che i cerchi scaleranno con l'intervallo dell'asse x. Vedere anche **set object circle (p. 186)** e **fillstyle (p. 209)**.

Ellipses (Ellissi)

Lo stile **ellipses** plotta un'ellisse in ciascun data point. Questo stile è rilevante solo per il plotting 2D. Ogni ellisse è descritta in termini del suo centro, i diametri maggiore e minore, e l'angolo tra il suo diametro maggiore e l'asse x.

```
2 columns: x y
3 columns: x y major_diam
4 columns: x y major_diam minor_diam
5 columns: x y major_diam minor_diam angle
```



Se solo due colonne di input sono presenti, esse sono prese come coordinate dei centri, e le ellissi saranno disegnate con l'estensione predefinita (vedere **set style ellipse (p. 213)**). L'orientamento dell'ellisse, che è definito come l'angolo tra il diametro maggiore e l'asse x del grafico, è preso dallo stile ellisse predefinito (vedere **set style ellipse (p. 213)**). Se vengono fornite tre colonne di input, la terza viene utilizzata per entrambi i diametri. L'angolo di orientamento è predefinito a zero. Se sono presenti quattro colonne, esse sono interpretate come x, y, diametro maggiore, diametro minore. Notare che questi sono diametri, non raggi. Una quinta colonna opzionale può specificare l'angolo di orientamento in gradi. Le ellissi saranno anche disegnate con la loro estensione predefinita se uno dei diametri forniti nella forma a 3-4-5 colonne è negativo.

In tutti i casi di cui sopra, i dati della variabile colore opzionali possono essere forniti in un'ulteriore ultima (terza, quarta, quinta o sesta) colonna. Vedere **colorspec (p. 50)**.

Per default, il diametro maggiore è interpretato nelle unità dell'asse orizzontale (x or x2) del grafico, mentre il diametro minore è interpretato in quelle dell'asse verticale (y or y2). Se le scale degli assi x e y non sono uguali la proporzione del diametro maggiore/minore non sarà più corretta dopo una rotazione. Comunque, questo può essere cambiato con la keyword **units**.

Vi sono tre alternative: se **units xy** è incluso nella specifica del grafico, gli assi verranno scalati come descritto sopra. **units xx** assicura che entrambi i diametri siano interpretati nelle unità dell'asse x, mentre **units yy** signifca che entrambi i diametri sono interpretati nelle unità dell'asse y. In questi ultimi due casi le ellissi avranno le proporzioni corrette, anche se il grafico viene ridimensionato. Se **units** è omesso dal comando plot, sarà usata l'impostazione di **set style ellipse**.

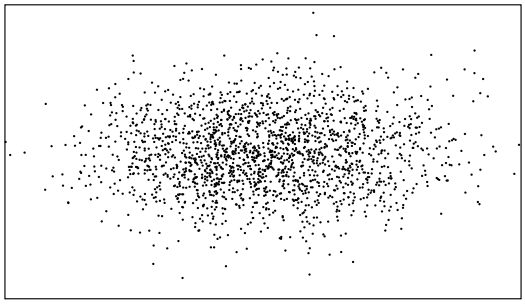
Esempio (disegna ellissi, scorrendo attraverso i tipi di linea disponibili):

```
plot 'data' using 1:2:3:4:(0):0 with ellipses
```

Vedere anche **set object ellipse (p. 186)**, **set style ellipse (p. 213)** e **fillstyle (p. 209)**.

Dots (Punti)

Lo stile **dots** plotta un piccolo puntino ad ogni punto; questo è utile per i grafici di dispersione con molti punti. In 2D sono richieste 1 o 2 colonne di dati di input. Sono necessarie tre colonne in 3D.

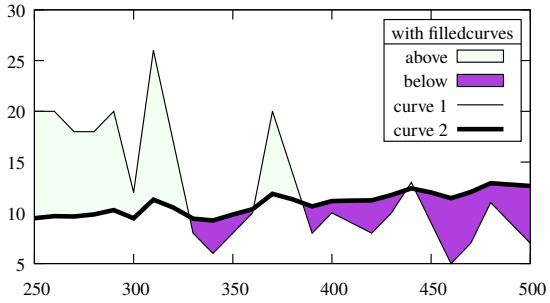


Per alcuni terminali (post, pdf) la dimensione del puntino può essere controllata cambiando lo spessore della linea.

```
1 column    y          # x è il numero di riga
2 columns:  x  y
3 columns:  x  y  z    # solo 3D (splot)
```

Filledcurves

Lo stile **filledcurves** viene usato solo per il plotting 2D. Ha tre varianti. Le prime due varianti richiedono o una singola funzione o due colonne (x,y) di dati input, e possono essere ulteriormente modificate dalle opzioni elencate di seguito.



Sintassi:

```
plot ... with filledcurves [option]
```

dove l'opzione può essere una delle seguenti

```
[closed | {above | below}
{x1 | x2 | y | r}[=<a>] | xy=<x>,<y>]
```

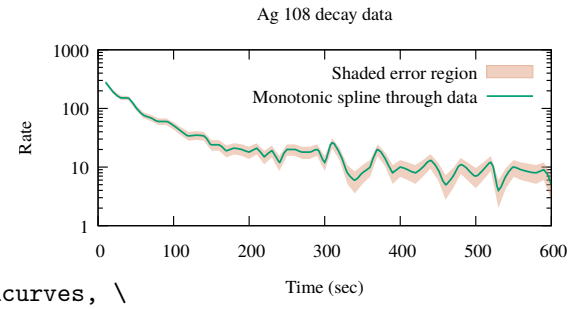
La prima variante, **closed**, tratta la curva stessa come un poligono chiuso. Questo è il default se ci sono due colonne di dati di input.

La seconda variante consiste nel riempire l'area tra la curva e un dato asse, una linea orizzontale o verticale, o un punto.

```
filledcurves closed    ... solo curva chiusa riempita,
filledcurves x1        ... asse x1,
```

```
filledcurves x2       ... asse x2, ecc per assi y1 e y2,
filledcurves y=42     ... linea a y=42, cioè parallela all'asse x,
filledcurves xy=10,20 ... punto 10,20 degli assi x1,y1 (forma ad arco).
filledcurves above r=1.5  l'area di un grafico polare fuori dal raggio 1.5
```

La terza variante riempie l'area tra due curve campionate allo stesso set di coordinate x. Essa richiede tre colonne di dati di input (x, y1, y2). Questo è il default se ci sono tre o più colonne di dati di input. Se si ha un valore y nella colonna 2 e un valore di errore associato nella colonna 3 l'area di incertezza può essere rappresentata dall'ombreggiatura. Vedere anche il simile stile grafico 3D **zerrorfill (p. 89)**.




```
3 columns:  x  y  yerror

plot $DAT using 1:($2-$3):($2+$3) with filledcurves, \
     $DAT using 1:2 smooth mcs with lines
```

Le opzioni **above** (sopra) e **below** (sotto) si applicano sia ai comandi della forma

```
... filledcurves above {x1|x2|y|r}=<val>
```

che ai comandi della forma

```
... using 1:2:3 with filledcurves below
```

In entrambi i casi l'opzione limita l'area riempita a un lato della linea o curva di delimitazione.

Note: Non tutti i tipi i terminale supportano questa modalità di plotting.

```
The x= and y= keywords are ignored for 3 columns data plots
```

Zoommare su una curva riempita, disegnata da un file di dati può produrre aree vuote o errate perché gnuplot sta ritagliando punti e linee, e non aree.

Se i valori <x>, <y>, o <a> sono al di fuori del confine del disegno, essi vengono spostati sul confine del grafico (graph). Quindi l'area di riempimento effettiva nel caso dell'opzione xy=<x>,<y> dipenderà da xrange e yrange.

Fill properties

Plottare con **with filledcurves** può essere ulteriormente personalizzato dando uno stile di riempimento (solido/trasparente/pattern) o un colore di riempimento. Se non viene fornito alcun stile di riempimento (**fs**) nel comando plot, allora viene usato lo stile di riempimento predefinito corrente. Se non viene fornito nessun colore di riempimento (**fc**) nel comando plot, allora viene seguita la solita sequenza di colori dei tipi di linea.

La proprietà {{no}border} dello stile di riempimento è onorata dalla modalità filledcurves **closed**, il default. È ignorata da tutte le altre modalità filledcurves. Esempio:

```
plot 'data' with filledcurves fc "cyan" fs solid 0.5 border lc "blue"
```

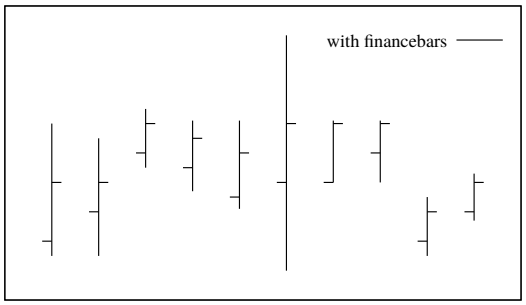
Financebars

Lo stile **financebars** è rilevante solo per plottare dati 2D di dati finanziari. Richiede 1 coordinata x (solitamente una data) e 4 valori y (prezzi).

```
5 columns:   date  open  low  high  close
```

Un'ulteriore (sesta) colonna di input può essere usata per fornire informazioni variabili (per record) sul colore (vedere **linecolor (p. 50)** e **rgbcolor variable (p. 52)**).

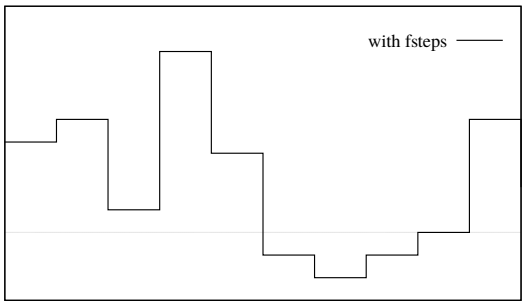
Il simbolo è un segmento di linea verticale, posizionato orizzontalmente alla coordinata x e limitato verticalmente dai prezzi alti e bassi. Un tic orizzontale sulla sinistra segna il prezzo di apertura e una sulla destra segna il prezzo di chiusura. La lunghezza di questi tic può essere cambiata da **set errorbars**. Il simbolo rimane invariato se i prezzi alti e bassi vengono scambiati. Vedere **set errorbars (p. 155)** e **candlesticks (p. 67)**, e anche la demo di finanza .



Fsteps

Lo stile **fsteps** è rilevante solo per il plotting 2D. Collega punti consecutivi con due segmenti di linea: il primo da (x1,y1) a (x1,y2) e il secondo da (x1,y2) a (x2,y2). I requisiti della colonna di input sono gli stessi degli stili di grafico **lines** e **points**. La differenza tra **fsteps** e **steps** è che **fsteps** traccia prima il cambiamento in y e poi il cambiamento in x. **steps** traccia prima il cambiamento in x e poi il cambiamento in y.

Vedere anche la demo steps .

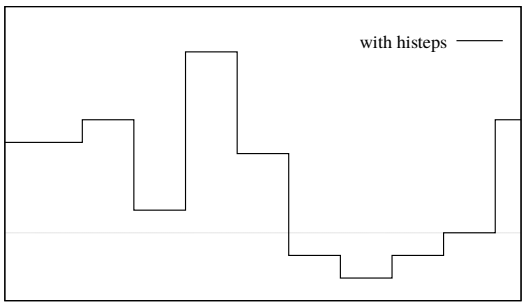


Fillsteps

Lo stile **fillsteps** è uguale a **steps**, eccetto che l'area tra la curva e y=0 è riempita con lo stile di riempimento corrente. Vedere **steps (p. 84)**.

Histeps

Lo stile **histeps** è rilevante solo per il plotting 2D. È destinato a plottare istogrammi. Si presume che i valori Y siano centrati sui valori x; il punto a x1 è rappresentato come una linea orizzontale da ((x0+x1)/2,y1) a ((x1+x2)/2,y1). Le linee che rappresentano i punti finali sono estese in modo tale che lo step (gradino) sia centrato su x. I punti adiacenti sono collegati da una linea verticale alla loro x media, cioè da ((x1+x2)/2,y1) a ((x1+x2)/2,y2). I requisiti della colonna di input sono gli stessi degli stili di grafico **lines** e **points**.



Se **autoscale** è in vigore, seleziona xrange dai dati piuttosto che dagli step, quindi i punti finali appariranno larghi solo la metà degli altri. Vedere anche la steps demo .

Histograms (Istrogrammi)

Lo stile **histograms** è rilevante solo per il plotting 2D. Produce un grafico a barre da una sequenza di colonne di dati parallele. Ogni elemento del comando **plot** deve specificare una singola sorgente di dati di input (ad es. una colonna del file di input), possibilmente con valori di tic associati o titoli key. Attualmente sono supportati quattro stili di layout dell'istogramma.

```
set style histogram clustered {gap <gapsize>}
set style histogram errorbars {gap <gapsize>} {<linewidth>}
set style histogram rowstacked
set style histogram columnstacked
set style histogram {title font "name,size" tc <colorspec>}
```

Lo stile predefinito corrisponde a **set style histogram clustered gap 2**. In questo stile, ogni serie di valori di dati paralleli è raccolta in un gruppo di box riunito alla coordinata dell'asse x corrispondente alla loro posizione sequenziale (riga #) nelle colonne dei file di dati selezionate. Quindi se <n> colonne di dati sono selezionate, il primo gruppo è centrato su x=1, e contiene <n> box le cui altezze sono prese dalla prima

entry nelle corrispondenti colonne di dati <n>. Questo è seguito da uno spazio vuoto e poi da un secondo gruppo di box centrati su x=2 corrispondente alla seconda entry nelle rispettive colonne di dati, e così via. La larghezza dello spazio predefinita di 2 indica che lo spazio vuoto tra i gruppi è equivalente alla larghezza di 2 box. A tutti i box derivati da una qualsiasi colonna viene assegnato lo stesso colore di riempimento e/o pattern (vedere **set style fill (p. 209)**).

Ogni raggruppamento di box è derivato da una singola riga del file di dati input. È comune in tali file di input che il primo elemento di ogni riga sia un'etichetta. Le etichette di questa colonna possono essere posizionate lungo l'asse x sotto il gruppo di box appropriato con l'opzione **xticlabels** a **using**.

Lo stile **errorbars** è molto simile allo stile **clustered**, eccetto che richiede colonne aggiuntive di input per ogni entry. La prima colonna contiene l'altezza (valore y) di quel box, esattamente come per lo stile **clustered**.

```
2 columns:        y yerr          bar extends from y-yerr to y+err
3 columns:        y ymin ymax     bar extends from ymin to ymax
```

L'aspetto delle barre di errore è controllato dal valore attuale di **set errorbars** e dalla specifica opzionale <linewidth>.

Sono supportati due stili di istogrammi impilati, scelti dal comando **set style histogram {rowstacked|columnstacked}**. In questi stili i valori di dati dalle colonne selezionate sono raggruppati in pile di box. I valori positivi vengono impilati verso l'alto da y=0; i valori negativi vengono impilati verso il basso. Valori positivi e negativi misti produrranno sia una pila verso l'alto che una pila verso il basso. La modalità di impilamento predefinita è **rowstacked**.

Lo stile **rowstacked** colloca un box appoggiato sull'asse x per ogni valore di dati nella prima colonna selezionata; il primo valore di dati risulta in un box a x=1, il secondo a x=2, e così via. I box corrispondenti alla seconda e successive colonne di dati sono stratificati sopra questi, risultante in una pila di box a x=1 che rappresenta il primo valore di dati da ogni colonna, una pila di box a x=2 che rappresenta il secondo secondo valore di dati da ogni colonna, e così via. A tutti i box derivati da una qualsiasi colonna viene dato lo lo stesso colore di riempimento e/o pattern. (vedere **set style fill (p. 209)**).

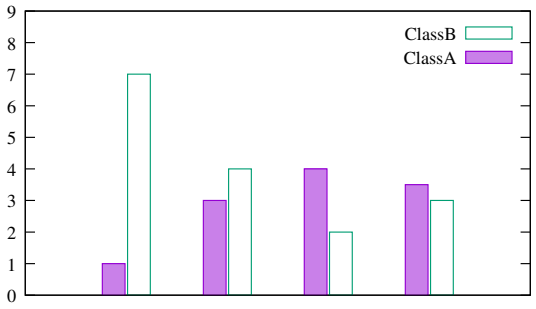
Lo stile **columnstacked** è simile, eccetto che ogni pila di box è costruita da una singola colonna di dati. Ogni valore di dati dalla prima colonna specificata produce un box nella pila a x=1, ogni valore di dati dalla seconda colonna specificata di dati produce un box nella pila a x=2, e così via. In questo stile il colore di ogni box è preso dal numero della riga, invece che dal numero della colonna, del campo di dati corrispondente.

Le larghezze dei box possono essere modificate usando il comando **set boxwidth**. Gli stili di riempimento dei box possono essere impostati usando il comando **set style fill**.

Gli istogrammi usano sempre l'asse x1, ma possono usare sia y1 che y2. Se un grafico contiene sia istogrammi che altri stili di grafici, gli elementi del grafico che non è istogramma possono utilizzare l'asse x1 o x2.

Esempi:

Supponiamo che il file di input contenga valori di dati nelle colonne 2, 4, 6, ... e stime di errore nelle colonne 3, 5, 7, ... Questo esempio plotta i valori nelle colonne 2 e 4 come un istogramma di box raggruppati (lo stile default). Poiché si utilizza l'iterazione nel comando plot, qualsiasi numero di colonne di dati può essere gestito in un singolo comando. Vedere **plot for (p. 128)**.



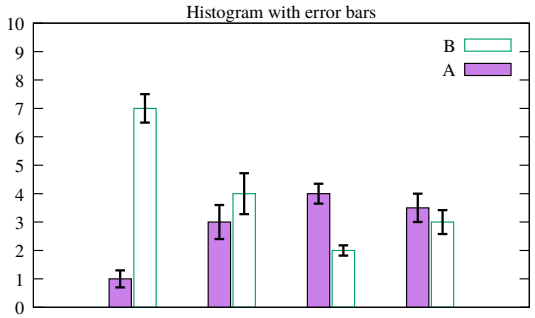

```
set boxwidth 0.9 relative
set style data histograms
set style histogram cluster
set style fill solid 1.0 border lt -1
plot for [COL=2:4:2] 'file.dat' using COL
```

Questo produrrà un grafico con gruppi di due box (barre verticali) centrati a ciascun valore integrale sull'asse x. Se la prima colonna del file di input contiene etichette, esse possono essere posizionate lungo l'asse x usando

il comando di variante

```
plot for [COL=2:4:2] 'file.dat' using COL:xticlabels(1)
```

Se il file contiene entrambe le informazioni di magnitudine e intervallo per ciascun valore, allora le barre di errore possono essere aggiunte al grafico. I comandi seguenti aggiungeranno barre di errore che si estendono da (y-<error>) a (y+<error>), coperte da estremità della barra orizzontale disegnate con la stessa larghezza del box stesso. Le barre di errore e le estremità delle barre sono disegnate con spessore di linea 2, usando il tipo di linea del bordo dallo stile di riempimento corrente.

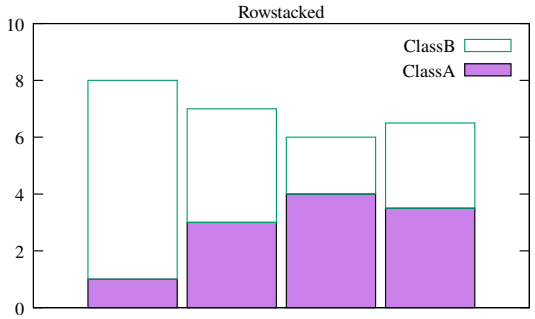



```
set errorbars fullwidth
set style fill solid 1 border lt -1
set style histogram errorbars gap 2 lw 2
plot for [COL=2:4:2] 'file.dat' using COL:COL+1
```

Questo mostra come plottare gli stessi dati come un istogramma a righe impilate. Giusto per essere diverso, questo esempio elenca le colonne separate esplicitamente invece che usando un'iterazione.

```
set style histogram rowstacked
plot 'file.dat' using 2, '' using 4:xtic(1)
```

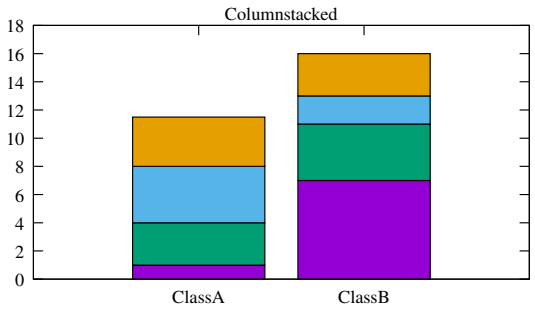
Questo produrrà un grafico in cui ciascuna barra verticale corrisponderà a una riga di dati. Ogni barra verticale continene una pila di due segmenti, corrispondenti in altezza ai valori trovati nelle colonne 2 e 4 dei file dati.



In fine, i comandi

```
set style histogram columnstacked
plot 'file.dat' using 2, '' using 4
```

produrranno due pile verticali, una per ciascuna colonna di dati. La pila a x=1 conterrà un box per ogni entry nella colonna 2 del file dati. La pila a x=2 conterrà un box per ogni entry parallela nella colonna 4 del file di dati.



Poiché questo scambia l'interpretazione abituale di gnuplot di righe e colonne di input, anche la specificazione dei titoli key e delle etichette dei tic dell'asse x devono essere modificate di conseguenza. Vedere i commenti che seguono.

```
set style histogram columnstacked
plot '' u 5:key(1)            # usa la prima colonna per generare titoli key
plot '' u 5 title columnhead  # usa la prima riga per generare etichette xtic
```

Notare che i due esempi appena dati presentano esattamente gli stessi valori di dati, ma in formati diversi.

Newhistogram

Sintassi:

```
newhistogram {"<title>" {font "name,size"} {tc <colorspec>}}
             {lt <linetype>} {fs <fillstyle>} {at <x-coord>}
```

Più di un set di istogrammi può apparire in un singolo grafico. In questo caso si può forzare uno spazio tra di loro e un'etichetta separata per ogni set, usando il comando **newhistogram**. Per esempio

```
set style histogram  cluster
plot newhistogram "Set A", 'a' using 1, '' using 2, '' using 3, \
     newhistogram "Set B", 'b' using 1, '' using 2, '' using 3
```

Le etichette "Set A" e "Set B" appariranno sotto i rispettivi set di istogrammi, sotto l'etichetta generale dell'asse x.

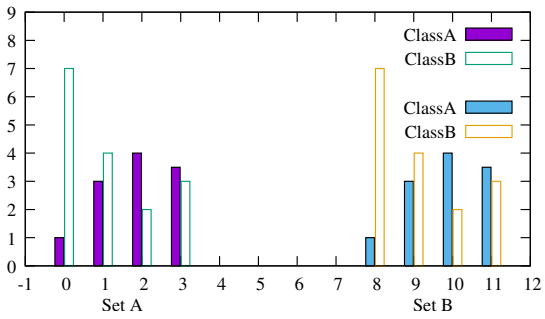
Il comando newhistogram può anche essere usato per forzare la colorazione dell'istogramma a iniziare con un colore specifico (linetype). Per impostazione predefinita i colori continueranno a incrementare successivamente anche attraverso i confini dell'istogramma. Ecco un esempio che usa la stessa colorazione per più istogrammi

```
plot newhistogram "Set A" lt 4, 'a' using 1, '' using 2, '' using 3, \
     newhistogram "Set B" lt 4, 'b' using 1, '' using 2, '' using 3
```

In modo simile è possibile forzare il prossimo istogramma ad iniziare con uno stile di riempimento specifico. Se lo stile di riempimento è impostato su **pattern**, allora il pattern usato per il riempimento sarà incrementato automaticamente.

L'opzione **at <x-coord>** imposta la posizione della coordinata x dell'istogramma seguente a <x-coord>. Per esempio

```
set style histogram cluster
set style data histogram
set style fill solid 1.0 border -1
set xtic 1 offset character 0,0.3
plot newhistogram "Set A", \
     'file.dat' u 1 t 1, '' u 2 t 2, \
     newhistogram "Set B" at 8, \
     'file.dat' u 2 t 2, '' u 2 t 2
```



posizionerà il secondo istogramma per iniziare a x=8.

Iterazione automatizzata su più colonne

Se si vuole creare un istogramma da molte colonne di dati in un singolo file, è molto comodo usare la funzione di iterazione del grafico. Vedere **plot for (p. 128)**. Per esempio, per creare istogrammi impilati dei dati nelle colonne da 3 a 8

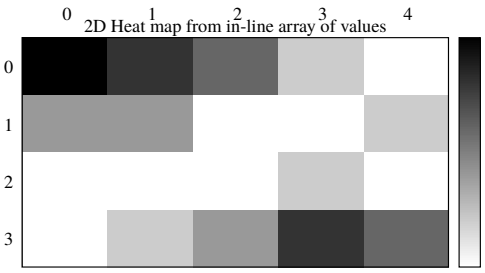
```
set style histogram columnstacked
plot for [i=3:8] "datafile" using i title columnhead
```

Image (Immagine)

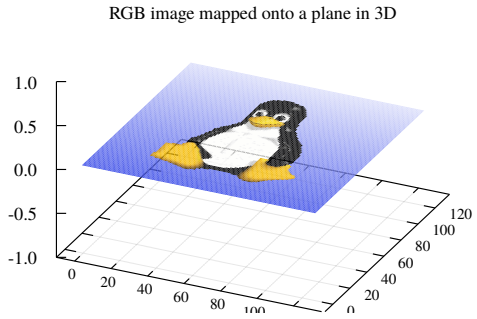
Gli stili di grafici **image**, **rgbimage**, e **rgbalpha** proiettano tutti una griglia uniformemente campionata di valori di dati su un piano in 2D o 3D. I dati di input possono essere una vera immagine bitmap, magari convertita da un formato standard come PNG, o un semplice array di valori numerici.

Questa figura illustra la generazione di una mappa di calore da un array di valori scalari. La palette corrente è usata per mappare ogni valore sul colore assegnato al pixel corrispondente.

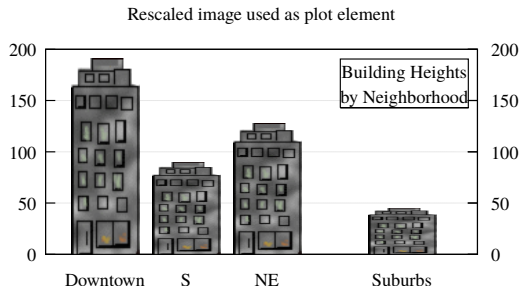
```
plot '-' matrix with image
5 4 3 1 0
2 2 0 0 1
0 0 0 1 0
0 1 2 4 3
e
e
```



Ogni pixel (data point) dell'immagine 2D in input diventerà un rettangolo o un parallelepipedo nel grafico. Le coordinate di ogni data point determineranno il centro del parallelepipedo. Cioè, un insieme di dati M x N formerà un'immagine con M x N pixel. Questo è diverso dallo stile di plotting pm3d, dove un insieme di dati M x N forma una superficie di (M-1) x (N-1) elementi. Le direzioni di scansione per una griglia di dati di immagine binaria può essere ulteriormente controllata da addizionali keyword. Vedere **binary keywords flipx (p. 111)**, **keywords center (p. 112)**, e **keywords rotate (p. 112)**.



I dati dell'immagine possono essere scalati per riempire un particolare rettangolo all'interno di un grafico 2D specificando l'estensione x e y di ogni pixel. Vedere **binary keywords dx (p. 111)** e **dy (p. 111)**. Per generare la figura a destra, la stessa immagine di input è stata posizionata più volte, ognuna con una specifica dx, dy e origine. L'immagine PNG di input di un edificio è di 50x128 pixel. L'edificio alto è stato disegnato mappandolo usando **dx=0.5 dy=1,5**. L'edificio basso ha usato una mappatura **dx=0,5 dy=0,35**.



Lo stile **image** gestisce i pixel di input che contengono un valore di scala di grigi o di palette di colori. Quindi i grafici 2D (comando **plot**) richiedono 3 colonne di dati (x,y,valore), mentre i grafici 3D (comando **splot**) richiedono 4 colonne di dati (x,y,z,valore).

Lo stile **rgbimage** gestisce i pixel in input che sono descritti da tre valori separati per i componenti rosso, verde e blu. Perciò i dati 5D (x,y,r,g,b) sono necessari per **plot** e dati 6D (x,y,z,r,g,b) per **splot**. I singoli componenti rosso, verde e blu si presume che si trovino nell'intervallo [0:255]. Questo corrisponde alla convenzione usata nei file PNG e JPEG (vedere **binary filetype (p. 110)**). Tuttavia alcuni file di dati usano una convenzione alternativa in cui i componenti RGB sono valori in virgola mobile nell'intervallo [0:1]. Per poter usare lo stile **rgbimage** con questo tipo di dati, bisogna prima usare il comando **set rgbmax 1.0**.

Lo stile **rgbalpha** gestisce i pixel in input che contengono informazioni sul canale alpha (trasparenza) oltre ai componenti rosso, verde e blu. Perciò i dati 6D (x,y,r,g,b,a) sono necessari per **plot** e dati 7D (x,y,z,r,g,b,a) per **splot**. I componenti r, g, b, e alpha si presume che si trovino nell'intervallo [0:255]. Per plottare i dati per i quali le componenti RGBA sono valori in virgola mobile nell'intervallo [0:1], bisogna prima usare il comando **set rgbmax 1.0**

Se viene fornita solo una singola colonna di dati per le componenti di colore dei grafici rgbimage o rgbalpha, essa viene interpretata come contenente dati ARGB imballati a 32 bit usando la convenzione che alpha=0 significa opaco e alpha=255 significa completamente trasparente. Si noti che questo è al contrario rispetto alla convenzione alpha se alpha è fornito in una colonna separata, ma corrisponde alla convenzione di impacchettamento ARGB per i singoli comandi per impostare il colore. Vedere **colorspec (p. 50)**.

Transparency

Lo stile di plotting **rgbalpha** presume che ogni pixel dei dati di input contenga un valore alpha nell'intervallo [0:255]. Un pixel con alpha = 0 è semplicemente trasparente e non altera il contenuto sottostante del grafico. Un pixel con alpha = 255 è semplicemente opaco. Tutti i tipi di terminale possono gestire questi due casi estremi. Un pixel con 0 < alpha < 255 è parzialmente trasparente. I tipi di terminale che non supportano la trasparenza parziale arrotonderanno questo valore a 0 o 255.

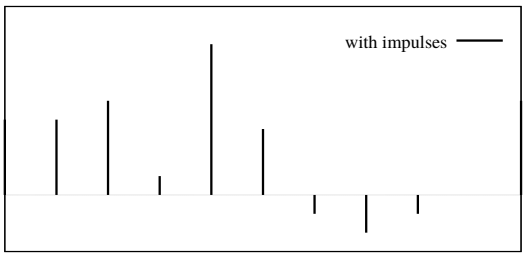
Image pixels

Alcuni terminali usano ottimizzazioni specifiche del dispositivo o della libreria per renderizzare i dati delle immagini all'interno di un'area rettangolare 2D. Ciò a volte produce un output indesiderato, ad es. cattivo ritaglio o scalatura, bordi mancanti. La keyword **pixels** dice a gnuplot di usare un codice generico che renderizza l'immagine pixel per pixel. Questa modalità di rendering è più lenta e può risultare in file di output molto più grandi, ma dovrebbe produrre una rappresentazione coerente su tutti i terminali. Esempio:

```
plot 'data' with image pixels
```

Impulses (Impulsi)

Lo stile **impulses** visualizza una linea verticale da y=0 al valore y di ogni punto (2D) o da z=0 al valore z di ogni punto (3D). Si noti che i valori y o z possono essere negativi. I dati delle colonne aggiuntive possono essere usati per controllare il colore di ogni impulso. Per usare efficacemente questo stile nei grafici 3D, è utile scegliere linee spesse (linewidth > 1). Questo approssima un grafico a barre 3D.




```
1 column:   y
2 columns:  x  y     # linea da [x,0] a [x,y]  (2D)
3 columns:  x  y  z  # linea da [x,y,0] a [x,y,z] (3D)
```

Labels (Etichette)

Lo stile **labels** legge le coordinate e il testo da un file di dati e posiziona la stringa di testo nella corrispondente posizione 2D o 3D. Sono necessarie 3 o 4 colonne di input di dati di base. Colonne di input aggiuntive possono essere usate per fornire proprietà che variano punto per punto come l'angolo di rotazione del testo (keyword **rotate variable**) o il colore (vedere **textcolor variable (p. 52)**).


```
3 columns:  x  y  string    # versione 2D
4 columns:  x  y  z  string # versione 3D
```

Il font, colore, angolo di rotazione e altre proprietà del testo stampato possono essere specificate come opzioni di comando aggiuntive (vedere **set label (p. 170)**). L'esempio sotto genera un grafico 2D con etichette di testo costruite a partire dalla città il cui nome è preso dalla colonna 1 del file di input, e le quali coordinate geografiche sono nelle colonne 4 e 5. La dimensione del font è calcolata dal valore nella colonna 3, in questo caso la popolazione.

```
CityName(String,Size) = sprintf("{/=%d %s}", Scale(Size), String)
plot 'cities.dat' using 5:4:(CityName(stringcolumn(1),$3)) with labels
```

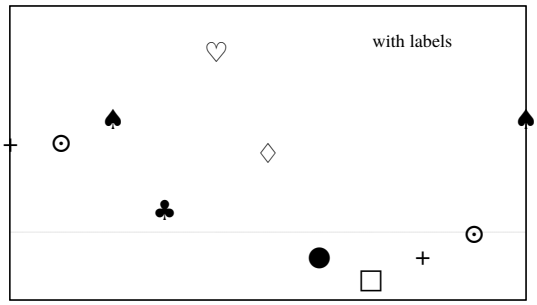
Se non si volesse regolare la dimensione del font a una dimensione diversa per ogni nome di città, il comando sarebbe molto più semplice:

```
plot 'cities.dat' using 5:4:1 with labels font "Times,8"
```

Se le etichette sono marcate come **hypertext** (ipertesto) allora il testo appare solo se il mouse passa sopra il punto di ancoraggio corrispondente. Vedere **hypertext (p. 173)**. In questo caso bisogna abilitare l'attributo **point** dell'etichetta in modo che ci sia un punto che funga da ancora ipertestuale:

```
plot 'cities.dat' using 5:4:1 with labels hypertext point pt 7
```

Lo stile **labels** può anche essere usato al posto dello stile **points** quando l'insieme dei simboli di punti predefiniti non è adatto o non è sufficientemente flessibile. Per esempio, qui viene definito un insieme di simboli a carattere singolo scelti, e assegnati uno di essi ad ogni punto di un grafico in base al valore nella colonna dati 3:



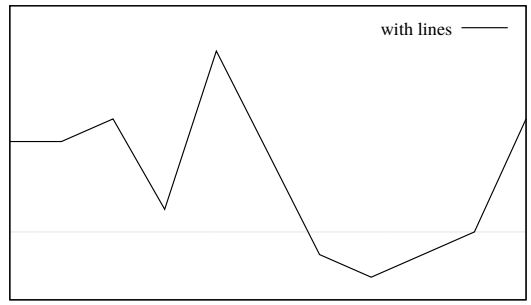

```
set encoding utf8
symbol(z) = "•□+⊙♠♣♡♢"[int(z):int(z)]
splot 'file' using 1:2:(symbol($3)) with labels
```

Questo esempio mostra l'uso di etichette con angolo di rotazione variabile nella colonna 4 e colore del testo ("tc") nella colonna 5. Notare che il colore della variabile è sempre preso dall'ultima colonna nello specificatore **using**.

```
plot $Data using 1:2:3:4:5 with labels tc variable rotate variable
```

Lines (Linee)

Lo stile **lines** connette punti adiacenti con segmenti di linea retta. Può essere usato sia in grafici 2D che 3D. La forma base richiede 1, 2, o 3 colonne di dati di input. Ulteriori colonne di input possono essere utilizzate per fornire informazioni come la variabile colore di linea (vedere **rgbcolor variable (p. 52)**).



forma 2D (no "using" spec)

```
1 column:   y       # x implicito dal numero
                    # di riga
2 columns:  x  y
```

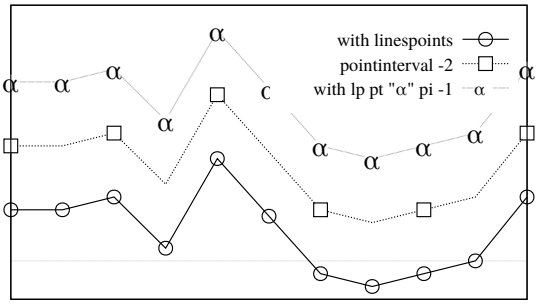
forma 3D (no "using" spec)

```
1 column:   z       # x implicito dalla riga, y dall'indice
3 columns:  x  y  z
```

Vedere anche **linetype (p. 173)**, **linewidth (p. 210)**, e **linestyle (p. 210)**.

Linespoints

Lo stile **linespoints** (forma abbreviata **lp**) collega punti adiacenti con segmenti di linea retta e poi torna a disegnare un piccolo simbolo in ogni punto. I punti sono disegnati con la dimensione predefinita determinata da **set pointsize**, a meno che una dimensione specifica del punto sia data nel comando plot o una dimensione del punto variabile sia fornita in una colonna aggiuntiva di dati di input. Colonne di input aggiuntive possono anche essere usate per fornire informazioni come la variabile colore delle linee. Vedere **linee (p. 80)** e **punti (p. 82)**.



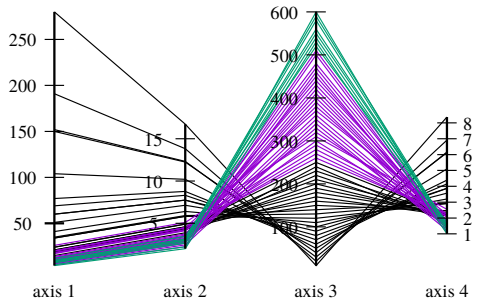
Due keyword controllano se ogni punto nel grafico è contrassegnato o meno da un simbolo, **pointinterval** (forma breve **pi**) e **pointnumber** (forma breve **pn**).

pi N or **pi -N** dice a gnuplot di mettere un simbolo solo ogni N punti. Un valore negativo per N cancellerà la porzione di segmento di linea che passa sotto il simbolo. La dimensione della porzione cancellata è controllata da **set pointintervalbox**.

pn N or **pn -N** dice a gnuplot di etichettare solo N dei data point, uniformemente distanziati sul set dei dati. Come con **pi**, un valore negativo per N cancellerà la porzione di segmento di linea che passa sotto il simbolo.

Parallelaxes

I grafici ad assi paralleli possono evidenziare la correlazione in una serie di dati multidimensionali. Le singole colonne di dati di input sono associate ciascuna a un asse verticale scalato separatamente. Se tutte le colonne sono disegnate da un singolo file, allora ogni linea sul grafico rappresenta i valori di una singola riga di dati in quel file. È comune usare qualche categorizzazione discreta per assegnare i colori delle linee, permettendo l'esplorazione visiva della correlazione tra questa categorizzazione e le dimensioni degli assi.



Sintassi:

```
set style data parallelaxes
plot $DATA using col1{:varcol1} {at <xpos>} {<line properties}, \
     $DATA using col2, ...
```

CAMBIAMENTO: La versione 5.4 di gnuplot introduce un cambiamento nella sintassi per lo stile del grafico parallelaxes. La sintassi modificata permette un numero illimitato di assi paralleli.

```
gnuplot 5.2:   plot $DATA using 1:2:3:4:5 with parallelaxes
gnuplot 5.4:   plot for [col=1:5] $DATA using col with parallelaxes
```

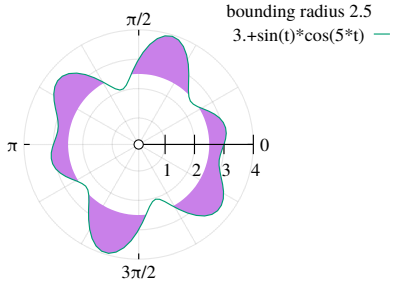
La nuova sintassi permette anche il posizionamento esplicito degli assi verticali paralleli lungo l'asse x, come nell'esempio qui sotto. Se non viene fornita nessuna coordinata x esplicita, allora l'asse N sarà posto a x=N.

```
array xpos[5] = [1, 5, 6, 7, 11, 12]
plot for [col=1:5] $DATA using col with parallelaxes at xpos[col]
```

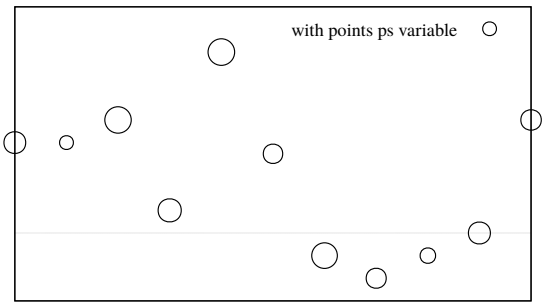
Per default gnuplot determina automaticamente l'intervallo e la scala dei singoli assi dai dati di input, ma i soliti comandi **set axis range** possono essere usati per personalizzarli. Vedere **set paxis (p. 195)**.

Polar plots (grafici polari)

I grafici polari sono generati cambiando il sistema di coordinate corrente in polare prima di emettere un comando di plot. L'opzione **set polar** dice a gnuplot di interpretare le coordinate 2D in input come <angle>,<radius> piuttosto che <x>,<y>. Molti, ma non tutti, stili di grafico 2D funzionano in modalità polare. La figura mostra una combinazione di stili di grafico **lines** e **filledcurves**. Vedere **set polar (p. 202)**, **set rrange (p. 204)**, **set size square (p. 205)**, **set theta (p. 216)**, **set ttics (p. 220)**.



Points (Punti)



Lo stile **points** mostra un piccolo simbolo in ogni punto. Il comando **set pointsize** può essere usato per cambiare la dimensione predefinita di tutti i punti. Il tipo di punto diventa per default come quello del tipo di linea. Vedere **linetype (p. 173)**. Se non si trova nessuna specifica **using** nel comando plot, le colonne dei dati di input sono interpretate implicitamente nell'ordine

```
x y pointsize pointtype color
```

Tutte le colonne oltre le prime due (x e y) sono opzionali; corrispondono a proprietà aggiuntive del grafico **pointsize variable**, **pointtype variable**, etc.

I primi 8 tipi di punti sono condivisi da tutti i terminali. I singoli terminali possono fornire un numero molto più grande di tipi di punti diversi. Usare il comando **test** per mostrare quali sono forniti dalle impostazioni correnti del terminale.

In alternativa, qualsiasi singolo carattere stampabile può essere dato al posto di un tipo di punto numerico, come nell'esempio seguente. È possibile usare qualsiasi carattere unicode come tipo di punto (presuppone il supporto utf8). Vedere **sequenze di escape (p. 36)**. Le stringhe più lunghe possono essere plottate usando lo stile di grafico **labels** piuttosto che **points**.

```
plot f(x) with points pt "#"
plot d(x) with points pt "\U+2299"
```

Quando si usano le keyword **pointtype**, **pointsize**, o **linecolor** in un comando plot la keyword aggiuntiva **variable** può essere data al posto di un numero. In questo caso le proprietà corrispondenti di ciascun punto sono assegnate da colonne aggiuntive di dati di input. La variabile pointsize è sempre presa dalla prima colonna aggiuntiva fornita in una specifica **using**. La variabile colore è sempre presa dall'ultima colonna aggiuntiva. Vedere **colorspec (p. 50)**. Se tutte e tre le proprietà sono specificate per ogni punto, l'ordine delle colonne dei dati di input è così

```
plot DATA using x:y:pointsize:pointtype:color \
     with points lc variable pt variable ps variable
```

Nota: per informazioni sulle variabili di programma definite dall'utente, vedere **variabili (p. 45)**.

Polygons (Poligoni)

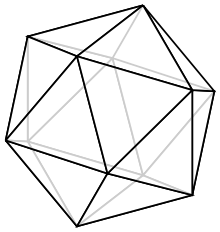
```
splot DATA {using x:y:z} with polygons
      {fillstyle <fillstyle spec>}
      {fillcolor <colorspec>}
```

splot with polygons usa pm3d per renderizzare singoli triangoli, quadrangoli, e poligoni più grandi in 3D. Queste possono essere sfaccettature di una superficie 3D o forme isolate. Il codice presuppone che i vertici si trovino in un piano. I vertici che definiscono poligoni individuali vengono letti da record successivi del file input. Una linea bianca separa un poligono dal successivo.

Lo stile e il colore di riempimento possono essere specificati nel comando splot, altrimenti viene usato lo stile di riempimento globale da **set style fill**. A causa di limitazioni nel codice pm3d, un singolo stile linea di bordo da **set pm3d border** viene applicato a tutti i poligoni. Questa restrizione potrebbe essere rimossa in una versione successiva di gnuplot.

L'ordine e l'illuminazione di pm3d sono applicati alle facce. Probabilmente è sempre preferibile usare **set pm3d depthsort**.

```
set xyplane at 0
set view equal xyz
unset border
unset tics
set pm3d depth
set pm3d border lc "black" lw 1.5
splot 'icosahedron.dat' with polygons \
      fs transparent solid 0.8 fc bgnd
```



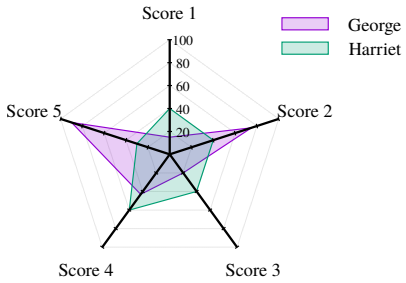
Spiderplot

Gli spider plot sono essenzialmente grafici ad assi paralleli in cui gli assi sono disposti radialmente piuttosto che verticalmente. Questi grafici sono chiamati a volte **rader charts**. In gnuplot questo richiede di lavorare all'interno di un sistema di coordinate stabilito dal comando **set spiderplot**, analogo a **set polar**, eccetto che la coordinata angolare è determinata implicitamente dal numero dell'asse parallelo. L'aspetto, l'etichettatura e il posizionamento dei tic degli assi è controllato da **set paxis**. Ulteriori scelte di stile sono controllate usando **set style spiderplot**, **set grid**, e i singoli componenti del comando plot.

Poiché ogni spider plot corrisponde a una riga di dati piuttosto che a una colonna, non avrebbe senso generare titoli di key entry nel modo normale. Invece, se una componente del grafico contiene un titolo, il testo viene usato per etichettare l'asse corrispondente. Questo sovrascrive qualsiasi precedente **set paxis n label "Foo"**. Per mettere un titolo nella key, si può usare un comando separato **keyentry** o estrarre il testo da una colonna del file input con lo specificatore d'uso **key(column)**. Vedere **keyentry (p. 168)**, **using key (p. 123)**.

In questa figura uno spiderplot con 5 assi è usato per confrontare più entità che sono caratterizzate ciascuna da cinque punteggi. Ogni linea (riga) in $DATA genera un nuovo poligono sul grafico.

```
set spiderplot
set style spiderplot fs transparent solid 0.2 border
set for [p=1:5] paxis p range [0:100]
set for [p=2:5] paxis p tics format ""
set             paxis 1 tics font ",9"
set for [p=1:5] paxis p label sprintf("Score %d",p)
set grid spiderplot
plot for [i=1:5] $DATA using i:key(1)
```



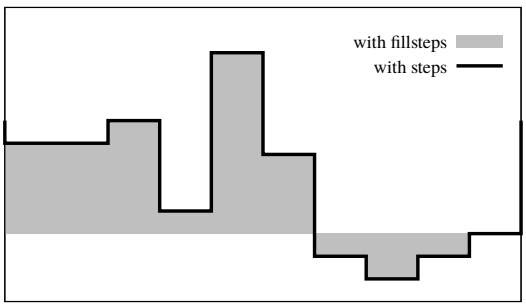
Newspiderplot

Normalmente gli elementi sequenziali di un comando plot **with spiderplot** corrispondono ciascuno ad un vertice di un singolo poligono. Per descrivere più poligoni nello stesso comando plot, essi devono essere separati da **newspiderplot**. Esempio:

```
# Un poligono con 10 vertici
plot for [i=1:5] 'A' using i, for [j=1:5] 'B' using j
# Due poligoni con 5 vertici
plot for [i=1:5] 'A' using i, newspiderplot, for [j=1:5] 'B' using j
```

Steps

Lo stile **steps** è rilevante solo per il plotting 2D. Collega punti consecutivi punti con due segmenti di linea: il primo da (x1,y1) a (x2,y1) e il secondo da (x2,y1) a (x2,y2). Le colonne di input richieste sono le stesse degli stili di grafico **lines** and **points**. La differenza tra **fsteps** and **steps** è che **fsteps** traccia prima il cambiamento in y e poi il cambiamento in x. **steps** traccia prima il cambiamento in x e poi il cambiamento in y. Per riempire l'area tra la curva e la linea di base a y=0, usare **fillsteps**. Vedere anche la demo steps .



Rgbalpha

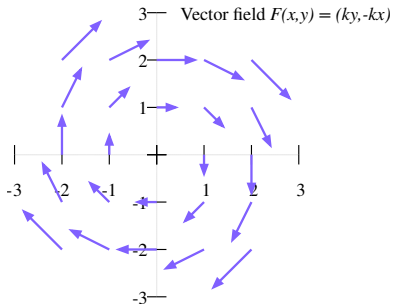
Vedere **image (p. 76)**.

Rgbimage

Vedere **image (p. 76)**.

Vectors

Lo stile 2D **vectors** disegna un vettore da (x,y) a (x+xdelta,y+ydelta). Lo stile 3D **vectors** è simile, ma richiede sei colonne di dati base. In entrambi i casi, una colonna di input aggiuntiva (quinta in 2D, settima in 3D) può essere usata per fornire informazioni di variabile colore (per-datapoint). (vedere **linecolor (p. 50)** e **rgbcolor variable (p. 52)**). Una piccola punta di freccia è disegnata alla fine di ogni vettore.




```
4 columns:  x  y  xdelta  ydelta
6 columns:  x  y  z  xdelta  ydelta  zdelta
```

Le keyword "with vectors" possono essere seguite da specifiche di uno stile di freccia in linea, un riferimento a uno stile di freccia predefinito o una richiesta di lettura dell'indice dello stile di freccia desiderato per ciascun vettore da una colonna separata. Nota: Se si sceglie "arrowstyle variable", verranno compilate tutte le proprietà della freccia nel momento in cui il vettore corrispondente viene disegnato; non si può mischiare questa keyword con altri qualificatori di stile linea o freccia nel comando plot.

```
plot ... with vectors filled heads
plot ... with vectors arrowstyle 3
plot ... using 1:2:3:4:5 with vectors arrowstyle variable
```

Esempio:

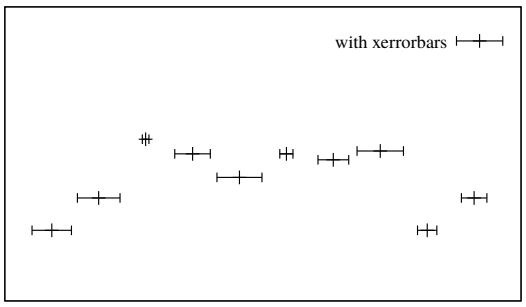
```
plot 'file.dat' using 1:2:3:4 with vectors head filled lt 2
splot 'file.dat' using 1:2:3:(1):(1):(1) with vectors filled head lw 2
```

splot con vettori è supportato solo per **set mapping cartesian**. **set clip one** e **set clip two** influenzano i vettori disegnati in 2D. Vedere **set clip (p. 143)** e **arrowstyle (p. 206)**.

Vedere anche lo stile di grafico 2D **with arrows (p. 63)** che è identico a **with vectors (p. 84)**, tranne che ogni freccia è specificata usando x:y:lunghezza:angolo.

Xerrorbars

Lo stile **xerrorbars** è rilevante solo per i grafici di dati 2D. **xerrorbars** è come **points**, eccetto che viene disegnata anche una barra orizzontale di errore. In ogni punto (x,y), una linea è disegnata da (xlow,y) a (xhigh,y) o da (x-xdelta,y) a (x+xdelta,y), a seconda del numero di colonne di dati fornite. L'aspetto del segno del tic (tacca) alle estremità della barra è controllato da **set errorbars**. Lo stile di base richiede 3 o 4 colonne:

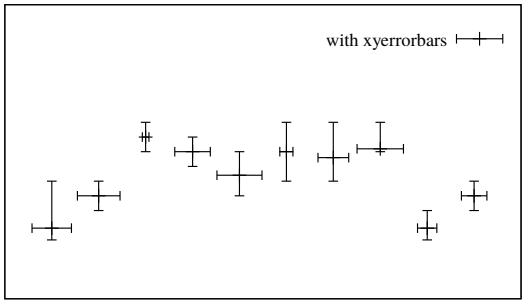



```
3 columns:  x  y  xdelta
4 columns:  x  y  xlow  xhigh
```

Un'ulteriore colonna di input (quarta o quinta) può essere usata per fornire informazioni come il colore della variabile punto.

Xyerrorbars

Lo stile **xyerrorbars** è rilevante solo per i grafici di dati 2D. **xyerrorbars** è come **points**, eccetto che vengono disegnate anche una barra orizzontale e una verticale di errore. In ogni punto (x,y), vengono disegnate delle linee da (x,y-ydelta) a (x,y+ydelta) e da (x-xdelta,y) a (x+xdelta,y), o da (x,ylow) a (x,yhigh) e da (xlow,y) a (xhigh,y), a seconda del numero di colonne di dati fornite. L'aspetto del segno del tic alle estremità della barra è controllato da **set errorbars**. Sono necessarie 4 o 6 colonne di input.




```
4 columns:  x  y  xdelta  ydelta
6 columns:  x  y  xlow  xhigh  ylow  yhigh
```

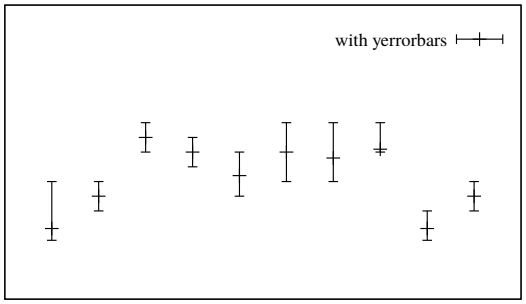
Se i dati sono forniti in una forma mista non supportata, il filtro **using** sul comando **plot** dovrebbe essere usato per impostare la forma appropriata. Per esempio, se i dati sono della forma (x,y,xdelta,ylow,yhigh), allora si può usare

```
plot 'data' using 1:2:($1-$3):($1+$3):4:5 with xyerrorbars
```

Un'ulteriore colonna di input (quinta o settima) può essere usata per fornire informazioni come il colore della variabile punto.

Yerrorbars

Lo stile **yerrorbars** (o **errorbars**) è rilevante solo per i grafici di dati 2D. **yerrorbars** è come **points**, eccetto che viene disegnata anche una barra verticale di errore. In ogni punto (x,y), una linea è disegnata da (x,y-ydelta) a (x,y+ydelta) o da (x,ylow) a (x,yhigh), a seconda del numero di colonne di dati fornite. L'aspetto del segno del tic alle estremità della barra è controllato da **set errorbars**.



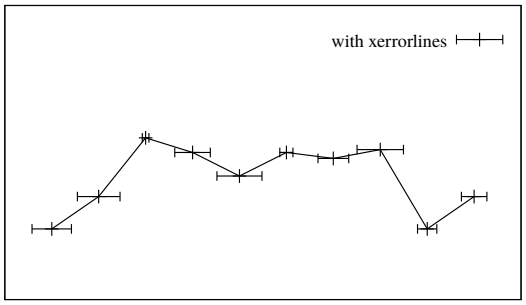

```
2 columns:  [implicit x] y ydelta
3 columns:  x  y  ydelta
4 columns:  x  y  ylow  yhigh
```

Un'ulteriore colonna di input (quarta o quinta) può essere usata per fornire informazioni come il colore della variabile punto.

Vedere anche la errorbar demo .

Xerrorlines

Lo stile **xerrorlines** è rilevante solo per i grafici di dati 2D. **xerrorlines** è come **linespoints**, eccetto che viene disegnata anche una linea verticale di errore. In ogni punto (x,y), una linea è disegnata da (xlow,y) a (xhigh,y) o da (x-xdelta,y) a (x+xdelta,y), a seconda del numero di colonne di dati fornite. L'aspetto del segno del tic alle estremità della barra è controllato da **set errorbars**. Lo stile di base richiede 3 o 4 colonne:

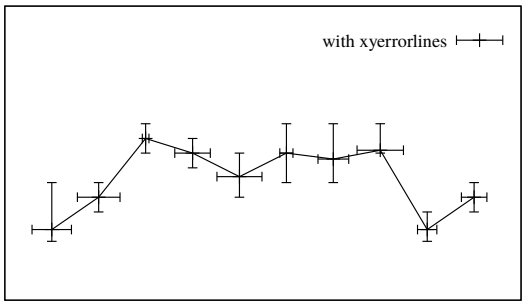



```
3 columns:  x  y  xdelta
4 columns:  x  y  xlow  xhigh
```

Un'ulteriore colonna di input (quarta o quinta) può essere usata per fornire informazioni come il colore della variabile punto.

Xyerrorlines

Lo stile **xyerrorlines** è rilevante solo per i grafici di dati 2D. **xyerrorlines** è come **linespoints**, eccetto che vengono disegnate anche una barra orizzontale e una verticale di errore. In ogni punto (x,y), vengono disegnate delle linee da (x,y-ydelta) a (x,y+ydelta) e da (x-xdelta,y) a (x+xdelta,y), o da (x,ylow) a (x,yhigh) e da (xlow,y) a (xhigh,y), a seconda del numero di colonne di dati fornite. L'aspetto del segno del tic alle estremità della barra è controllato da **set errorbars**. Sono necessarie 4 o 6 colonne di input.




```
4 columns:  x  y  xdelta  ydelta
6 columns:  x  y  xlow  xhigh  ylow  yhigh
```

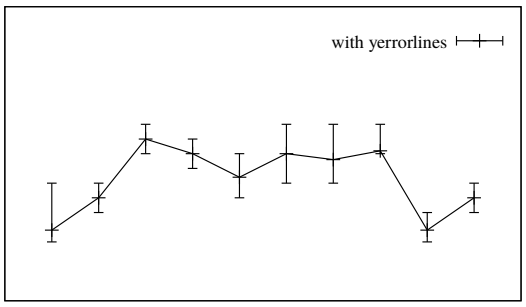
Se i dati sono forniti in una forma mista non supportata, il filtro **using** sul comando **plot** dovrebbe essere usato per impostare la forma appropriata. Per esempio, se i dati sono della forma (x,y,xdelta,ylow,yhigh), allora si può usare

```
plot 'data' using 1:2:($1-$3):($1+$3):4:5 with xyerrorlines
```

Un'ulteriore colonna di input (quinta o settima) può essere usata per fornire informazioni come il colore della variabile punto.

Yerrorlines

Lo stile **yerrorlines** (or **errorlines**) è rilevante solo per i grafici di dati 2D. **yerrorlines** è come **linespoints**, eccetto che viene disegnata anche una linea verticale di errore. In ogni punto (x,y), una linea è disegnata da (x,y-ydelta) a (x,y+ydelta) o da (x,ylow) a (x,yhigh), a seconda del numero di colonne di dati fornite. L'aspetto del segno del tic alle estremità della barra è controllato da **set errorbars**. Sono necessarie 3 o 4 colonne di input.




```
3 columns:  x  y  ydelta
4 columns:  x  y  ylow  yhigh
```

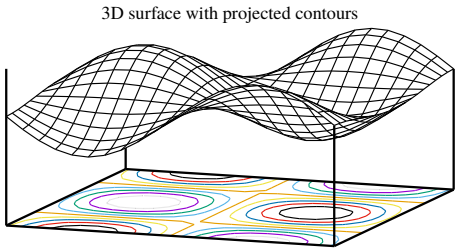
Un'ulteriore colonna di input (quarta o quinta) può essere usata per fornire informazioni come il colore della variabile punto.

Vedere anche la demo errorbar .

Grafici 3D

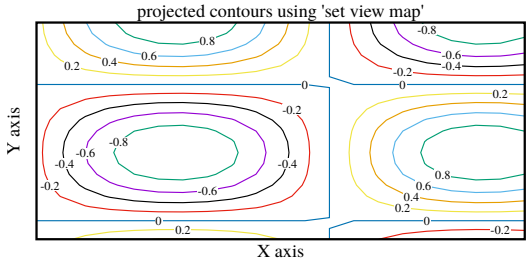
I grafici 3D sono generati usando il comando **splot** piuttosto che **plot**. Molti degli stili di grafico 2D (points, images, impulse, labels, vectors) possono anche essere usati in 3D fornendo una colonna extra di dati che contiene la coordinata z. Alcuni tipi di grafici (pm3d coloring, surfaces, contours) devono essere generati usando il comando **splot** anche se si vuole solo una proiezione 2D.

Surface plots (Grafici a superficie)



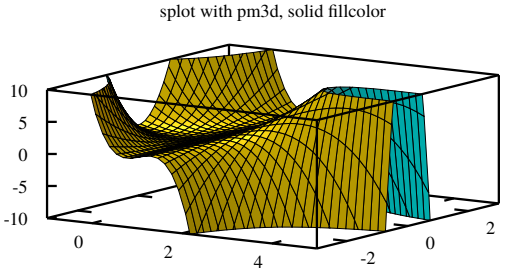
Gli stili **splot with lines** e **splot with surface** entrambi generano una superficie costituita da una griglia di linee. Le superfici solide possono essere generate usando lo stile **splot with pm3d**. Di solito la superficie viene visualizzata ad un certo angolo di visione conveniente tale da rappresentare chiaramente una superficie 3D. Vedere **set view (p. 221)**. In questo caso gli assi X, Y e Z sono tutti visibili nel grafico. L'illusione del 3D è migliorata scegliendo la rimozione delle linee nascoste. Vedere **hidden3d (p. 162)**. Il comando **splot** può anche calcolare e disegnare linee di contorno corrispondenti a valori Z costanti. Queste linee di contorno possono essere disegnate sulla superficie stessa, o proiettate sul piano XY. Vedere **set contour (p. 147)**.

2D projection (set view map)



Un importante caso speciale del comando **splot** è quello di mappare la coordinata Z su una superficie 2D proiettando il grafico lungo l'asse Z sul piano xy. Vedere **set view map (p. 221)**. Questa modalità è utile per i contour plots e le mappe di calore. Questa figura mostra i contorni plottati una volta con lo stile di grafico **lines** e una volta con stile **labels**.

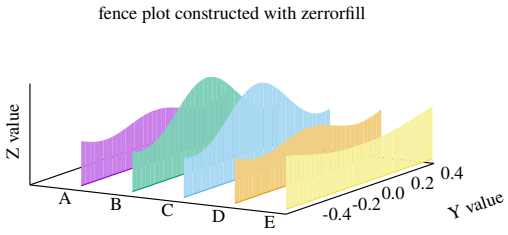
PM3D plots



Le superfici 3D possono anche essere disegnate usando quadrangoli pm3d solidi piuttosto che linee. In questo caso non c'è rimozione della superficie nascosta, ma se le sfaccettature del componente sono disegnate al contrario, si ottiene un effetto simile. Vedere **set pm3d depthorder (p. 199)**. Mentre le superfici pm3d sono colorate di default usando una palette di colori uniformi (vedere**set palette (p. 190)**), è anche possibile specificare una superficie a tinta unita o specificare tinte unite distinte per la superficie superiore e inferiore come nella figura mostrata qui. Vedere **pm3d fillcolor (p. 201)**. A differenza del line-trimming in modalità hidden3d, le superfici pm3d possono essere uniformemente clippate allo zrange corrente. Vedere **set pm3d clipping (p. 200)**.

Fence plots

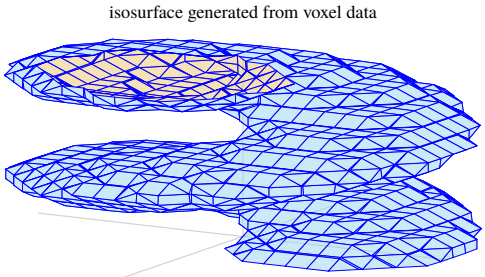
I fence plot combinano diversi grafici 2D allineando le loro coordinate Y e separandoli l'uno dall'altro con uno spostamento lungo X. Riempire l'area tra un valore di base e la serie di valori Z di ogni grafico migliora l'impatto visivo dell'allineamento su Y e del confronto su Z. Ci sono diversi modi in cui tali grafici possono essere creati in gnuplot. Il più semplice è usare la variante a 5 colonne dello stile **zerrorfill**. Supponiamo che ci siano curve separate z = Fi(y) indicizzate da i. Un fence plot è generato da **splot con zerrorfill** usando colonne di input




```
i y z_base z_base Fi(y)
```

Isosurface (Isosuperficie)

Questo stile di grafico 3D richiede una griglia voxel popolata (vedere **set vgrid (p. 221)**, **vfill (p. 247)**). L'interpolazione lineare dei valori della griglia voxel è usata per stimare le coordinate della griglia frazionata corrispondenti all'isolivello richiesto. Questi punti sono poi utilizzati per generare una superficie tassellata. Le sfaccettature che compongono la superficie sono renderizzate come poligoni pm3d, quindi la colorazione della superficie, la trasparenza e i bordi sono controllati da **set pm3d**. In generale la superficie è più facile da interpretare visivamente se alle sfaccettature viene dato un bordo sottile che è più scuro del colore di riempimento. Per default la tassellatura usa un misto di quadrangoli e triangoli. Per usare solo triangoli, vedere **set isosurface (p. 165)**. Esempio:




```
set style fill solid 0.3
set pm3d depthorder border lc "blue" lw 0.2
splot $helix with isosurface level 10 fc "cyan"
```

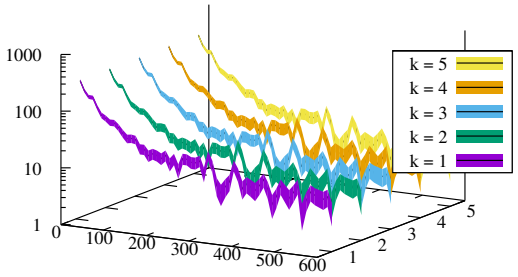
Zerrorfill

Sintassi:

```
splot DATA using 1:2:3:4[:5] with zerrorfill {fc|fillcolor <colorspec>}
          {lt|linetype <n>} {<line properties>}
```

Lo stile di grafico **zerrorfill** è simile a una variante dello stile di grafico 2D **filledcurves**. Riempie l'area tra due funzioni o linee di dati che sono campionate negli stessi punti x e y. Richiede 4 o 5 colonne di input:

```
4 columns:  x  y  z  zdelta
5 columns:  x  y  z  zlow  zhigh
```



L'area tra zlow e zhigh viene riempita e poi viene disegnata una linea attraverso i valori z. Per default sia la linea che l'area di riempimento usano lo stesso colore, ma questo si può cambiare nel comando splot. Le proprietà dell'area di riempimento sono anche influenzate dallo stile di riempimento globale; vedere **set style fill (p. 209)**.

Se ci sono più curve nel comando splot, ogni nuova curva può occludere tutte le curve precedenti. Per ottenere un corretto ordinamento della profondità in modo che le curve possano essere solo occluse dalle curve più vicine allo spettatore, usare **set pm3d depthorder base**. Sfortunatamente questo fa sì che tutte le aree riempite siano disegnate dopo tutte le linee di valori z corrispondenti. Per vedere sia le linee che le aree di riempimento ordinate in profondità, probabilmente si dovrà rendere le aree di riempimento parzialmente trasparenti o usare un riempimento a pattern piuttosto che un riempimento solido.

L'area di riempimento nei primi due esempi qui sotto è la stessa.

```
splot 'data' using 1:2:3:4 with zerrorfill fillcolor "grey" lt black
splot 'data' using 1:2:3:($3-$4):($3+$4) with zerrorfill
splot '+' using 1:(const):(func1($1)):(func2($1)) with zerrorfill
splot for [k=1:5] datafile[k] with zerrorfill lt black fc lt (k+1)
```

Questo stile di grafico può anche essere usato per creare fence plot. Vedere **fenceplots (p. 89)**.

Part III
Comandi

Questa sezione elenca i comandi accettabili da **gnuplot** in ordine alfabetico. Le versioni stampate di questo documento contengono tutti i comandi; il testo disponibile interattivamente potrebbe non essere completo. In effetti, su alcuni sistemi potrebbero non esserci comandi elencati sotto questa voce.

Si noti che nella maggior parte dei casi le abbreviazioni non ambigue per i nomi dei comandi e le loro opzioni sono consentite, cioè, **"p f(x) w li"** invece di **"plot f(x) with lines"**.

Nelle descrizioni della sintassi, le parentesi graffe ({}) denotano argomenti opzionali e una barra verticale (|) separa le scelte reciprocamente esclusive.

Break

Il comando **break** ha senso solo all'interno della clausola di iterazione tra parentesi di una dichiarazione **do** o **while**. Fa sì che le dichiarazioni rimanenti all'interno della clausola siano saltate e l'iterazione sia terminata. L'esecuzione riprende alla dichiarazione che segue la parentesi di chiusura. Vedere anche **continue (p. 93)**.

Cd

Il comando **cd** cambia la directory di lavoro.

Sintassi:

```
cd '<directory-name>'
```

Il nome della directory deve essere racchiuso tra virgolette.

Esempi:

```
cd 'subdir'
cd ".."
```

Si raccomanda agli utenti di Windows di usare le virgolette singole, perché il backslash [\] ha un significato speciale all'interno delle virgolette doppie e deve essere evitato. Per esempio,

```
cd "c:\newdata"
```

non funziona, ma

```
cd 'c:\newdata'
cd "c:\\newdata"
```

funziona come previsto.

Call

Il comando **call** è identico al comando **load** con un'eccezione: il nome del file da caricare può essere seguito da un massimo di nove parametri.

```
call "inputfile" <param-1> <param-2> <param-3> ... <param-9>
```

Le versioni precedenti di gnuplot eseguivano la sostituzione (simile a quella delle macro) dei token speciali $0, $1, ... $9 con il contenuto letterale di questi parametri. Questo meccanismo è ora deprecato (vedere **call old-style (p. 93)**).

Gnuplot fornisce ora un insieme di variabili stringa ARG0, ARG1, ..., ARG9 e una variabile intera ARGC. Quando viene eseguito un comando **call** ARG0 è impostato al nome del file di input, ARGC è impostato al numero di parametri presenti, e ARG1 fino a ARG9 sono caricati dai parametri che lo seguono sulla linea di comando. Qualsiasi contenuto esistente delle variabili ARG viene salvato e ripristinato attraverso un comando **call**.

Poiché i parametri ARG1 ... ARG9 sono memorizzati in normali variabili stringa, essi possono essere dereferenziati dall'espansione delle macro (analogamente alla vecchia sintassi deprecata). Tuttavia in molti casi è più naturale usarli come si farebbe con qualsiasi altra variabile.

Parallelamente ai parametri stringa ARG1 ... ARG9, i parametri passati sono memorizzati in un array ARGV[9]. Vedere **argv (p. 92)**.

Argv[ ]

Quando uno script di gnuplot viene inserito tramite il comando **call** qualsiasi parametro passato dal chiamante è disponibile attraverso due meccanismi. Ciascun parametro è memorizzato come una stringa nelle variabili ARG1, ARG2, ... ARG9. Ogni parametro è anche memorizzato come un elemento dell'array ARGV[9]. I valori numerici sono memorizzati come variabili complesse. Tutti gli altri valori sono memorizzati come stringhe. Quindi dopo una call

```
call 'routine_1.gp'  1 pi "title"
```

I tre argomenti sono disponibili all'interno di routine_1.gp come segue

```
ARG1 = "1"          ARGV[1] = 1.0
ARG2 = "3.14159"    ARGV[2] = 3.14159265358979...
ARG3 = "title"      ARGV[3] = "title"
```

In questo esempio ARGV[1] e ARGV[2] hanno la precisione di una variabile in virgola mobile. ARG2 ha perso precisione nell'essere memorizzato come una stringa usando il formato "%g".

Esempio

```
Call site
    MYFILE = "script1.gp"
    FUNC = "sin(x)"
    call MYFILE FUNC 1.23 "This is a plot title"
Upon entry to the called script
    ARG0 holds "script1.gp"
    ARG1 holds the string "sin(x)"
    ARG2 holds the string "1.23"
    ARG3 holds the string "This is a plot title"
    ARGC is 3
The script itself can now execute
    plot @ARG1 with lines title ARG3
    print ARG2 * 4.56, @ARG2 * 4.56
    print "This plot produced by script ", ARG0
```

Si noti che, poiché ARG1 è una stringa, deve essere dereferenziata come una macro, ma ARG2 può essere dereferenziato sia come una macro (che produce una costante numerica) o una variabile (che produce lo stesso valore numerico dopo l'autopromozione della stringa "1.23" a un reale).

Lo stesso risultato potrebbe essere ottenuto direttamente da uno script di shell invocando gnuplot con l'opzione della linea di comando **-c**:

```
gnuplot -persist -c "script1.gp" "sin(x)" 1.23 "This is a plot title"
```

Old-style

Questo descrive il meccanismo call deprecato usato dalle vecchie versioni di gnuplot.

```
call "<input-file>" <param-0> <param-1> ... <param-9>
```

Il nome del file di input deve essere racchiuso tra virgolette. Mentre ogni linea viene letta dal file di input, essa viene scansionata per le seguenti sequenze di caratteri speciali: $0 $1 $2 $3 $4 $5 $6 $7 $8 $9 $#. Se trovata, la sequenza **$**+cifra è sostituita dal parametro corrispondente dalla linea di comando **call**. I caratteri di virgolette non vengono copiati e la sostituzione delle variabili stringa non viene eseguita. La sequenza di caratteri **$#** è sostituita dal numero di parametri passati. **$** seguito da qualsiasi altro carattere è trattato come una sequenza di escape; utilizzare **$$** per ottenere un singolo **$$**.

Esempio:

Se il file 'calltest.gp' contiene la linea:

```
print "argc=$# p0=$0 p1=$1 p2=$2 p3=$3 p4=$4 p5=$5 p6=$6 p7=x$7x"
```

inserire il comando:

```
call 'calltest.gp' "abcd" 1.2 + "'quoted'" -- "$2"
```

visualizzerà:

```
argc=7 p0=abcd p1=1.2 p2=+ p3='quoted' p4=- p5=- p6=$2 p7=xx
```

NOTE: Questo uso del carattere **$** è in conflitto sia con la sintassi di gnuplot per le colonne di file di dati che con l'uso di **$** per indicare le variabili in una shell simile a unix. La sequenza speciale **$#** è stata male interpretata come delimitatore di commento nelle versioni di gnuplot dalla 4.5 alla 4.6.3. I caratteri di virgolette sono ignorati durante la sostituzione, quindi le costanti di stringa sono facilmente corrotte.

Clear

Il comando **clear** cancella lo schermo corrente o il dispositivo di output come specificato da **set terminal** e **set output**. Questo di solito genera un formfeed su dispositivi di copia cartacea.

Per alcuni terminali, **clear** cancella solo la porzione della superficie di plotting definita da **set size**, quindi per questi può essere usato insieme a **set multiplot** per creare un inserto.

Esempio:

```
set multiplot
plot sin(x)
set origin 0.5,0.5
set size 0.4,0.4
clear
plot cos(x)
unset multiplot
```

Si prega di vedere **set multiplot (p. 180)**, **set size (p. 205)**, e **set origin (p. 188)** per i dettagli.

Continue

Il comando **continue** ha valore solo all'interno della clausola di iterazione tra parentesi di una dichiarazione **do** o **while**. Fa sì che le restanti dichiarazioni all'interno della clausola tra parentesi vengano saltate. L'esecuzione riprende all'inizio della prossima iterazione (se ne rimane qualcuna nella condizione del loop). Vedere anche **break (p. 91)**.

Do

Sintassi:

```
do for <iteration-spec> {
     <commands>
     <commands>
}
```

Esegue una sequenza di comandi più volte. I comandi devono essere racchiusi tra parentesi graffe, e il "{" di apertura deve essere sulla stessa linea della keyword **do**. Questo comando non può essere usato con dichiarazioni old-style if/else (senza parentesi). Vedere **if (p. 103)**. Per esempi di specificatori di iterazione, vedere **iteration (p. 49)**. Esempio:

```
set multiplot layout 2,2
do for [name in "A B C D"] {
    filename = name . ".dat"
    set title sprintf("Condition %s",name)
    plot filename title name
}
unset multiplot
```

Vedere anche **while (p. 247)**, **continue (p. 93)**, **break (p. 91)**.

Evaluate

Il comando **evaluate** esegue i comandi dati come argomenti stringa. I caratteri newline non sono ammessi all'interno della stringa.

Sintassi:

```
eval <string expression>
```

Questo è particolarmente utile per una ripetizione di comandi simili.

Esempio:

```
set_label(x, y, text) \
  = sprintf("set label '%s' at %f, %f point pt 5", text, x, y)
eval set_label(1., 1., 'one/one')
eval set_label(2., 1., 'two/one')
eval set_label(1., 2., 'one/two')
```

Si prega di vedere **sostituzione macro (p. 59)** per un altro modo di eseguire comandi da una stringa.

Exit

```
exit
exit message "error message text"
exit status <integer error code>
```

I comandi **exit** e **quit**, così come il carattere END-OF-FILE (solitamente Ctrl-D) terminano l'input dal flusso di input corrente: sessione di terminale, pipe, o input di file (pipe). Se i flussi di input sono annidati (script **load** ereditati), allora la lettura continuerà nel flusso del genitore. Quando il flusso di livello superiore è chiuso, il programma stesso uscirà.

Il comando **exit gnuplot** causerà immediatamente e incondizionatamente l'uscita di gnuplot anche se il flusso di input è annidato in più punti. In questo caso qualsiasi file output aperto potrebbe non essere completato in modo corretto. Esempio di utilizzo:

```
bind "ctrl-x" "unset output; exit gnuplot"
```

Il comando **exit error "error message"** simula un errore del programma. In modalità interattiva stampa il messaggio di errore e ritorna alla linea di comando, interrompendo tutti i cicli o le chiamate annidati. In modalità non interattiva il programma uscirà.

Quando gnuplot esce alla shell di controllo, il valore di ritorno di solito non è informativo. Questa variante del comando permette di restituire un valore specifico.

```
exit status <value>
```

Vedere aiuto per **batch/interactive (p. 31)** per ulteriori dettagli.

Fit

Il comando **fit** adatta un'espressione con valori reali fornita dall'utente a un set di data point, utilizzando l'algoritmo di Marquardt-Levenberg dei minimi quadrati non lineari. Ci possono essere fino a 12 variabili indipendenti, vi è sempre una variabile dipendente, e qualsiasi numero di parametri può essere adattato. Opzionalmente, le stime di errore possono essere inserite per la ponderazione dei data point.

L'uso di base di **fit** è spiegato in modo migliore da un semplice esempio:

```
f(x) = a + b*x + c*x**2
fit f(x) 'measured.dat' using 1:2 via a,b,c
plot 'measured.dat' u 1:2, f(x)
```

Sintassi:

```
fit {<ranges>} <expression>
    '<datafile>' {datafile-modifiers}
    {{unitweights} | {y|xy|z}error | errors <var1>{,<var2>,...}}
    via '<parameter file>' | <var1>{,<var2>,...}
```

Gli intervalli possono essere specificati per filtrare i dati usati nel fitting. I data point fuori dall'intervallo sono ignorati. La sintassi è

```
[{dummy_variable=}{<min>}{:<max>}],
```

analoga a **plot**; vedere **plot ranges (p. 126)**.

<expression> può essere una qualsiasi espressione **gnuplot** valida, anche se la più comune è una funzione precedentemente definita dall'utente della forma f(x) o f(x,y). Deve essere a valore reale. I nomi delle variabili indipendenti sono impostati dal comando **set dummy**, o nella parte <ranges> del comando (vedi sotto); per default, le prime due sono chiamate x e y. Inoltre, l'espressione dovrebbe dipendere da una o più variabili il cui valore deve essere determinato dalla procedura di fitting.

<datafile> è trattato come nel comando **plot**. Tutti i modificatori **plot datafile** (**using**, **every**,...) eccetto **smooth** sono applicabili a **fit**. Vedere **plot datafile (p. 112)**.

Il contenuti del file dati possono essere interpretati in modo flessibile fornendo un qualificatore **using** come nei i comandi plot. Per esempio per generare la variabile indipendente x come somma delle colonne 2 e 3, mentre si prende z dalla colonna 6 e richiedendo pesi uguali:

```
fit ... using ($2+$3):6
```

In assenza di una specifica **using**, il fit presuppone implicitamente che ci sia solo una singola variabile indipendente. Se il file stesso, o la specifica using, contiene solo una colonna di dati, il numero della linea viene preso come la variabile indipendente. Se viene data una specifica **using**, ci possono essere fino a 12 variabili indipendenti (e di più se appositamente configurato in fase di compilazione).

L'opzione **unitweights** , che è l'impostazione predefinita, fa sì che tutti i data point siano ponderati equamente. Ciò può essere cambiato usando la keyword **errors** per leggere le stime di errore di una o più variabili dal file di dati. Queste stime di errore sono interpretate come le deviazioni standard del valore della variabile corrispondente e utilizzate per computare un peso per il dato come 1/s**2.

Nel caso di stime di errore delle variabili indipendenti, questi pesi sono ulteriormente moltiplicati per le derivate della funzione di fitting secondo il "metodo della varianza efficace". (Jay Orear, Am. J. Phys., Vol. 50, 1982).

La keyword **errors** deve essere seguita da un elenco separato da virgole di uno o più nomi di variabili per le quali devono essere inseriti gli errori; la variabile dipendente z deve sempre essere tra loro, mentre le variabili indipendenti sono opzionali. Per ogni variabile in questa lista, un'ulteriore colonna sarà letta dal file, contenente la stima di errore di quella variabile. Di nuovo, un'interpretazione flessibile è possibile fornendo il qualificatore **using**. Si noti che il numero di variabili indipendenti è quindi implicitamente dato dal numero totale di colonne nel qualificatore **using**, meno 1 (per la variabile dipendente), meno il numero di variabili nel qualificatore **errors**.

Per esempio, se si hanno 2 variabili indipendenti, e gli errori per la prima variabile indipendente e la variabile dipendente, si usa il qualificatore **errors x,z** e un qualificatore **using** con 5 colonne, che sono interpretate come x:y:z:sx:sz (dove x e y sono le variabili indipendenti, z è la variabile dipendente e sx e sz sono le deviazioni standard di x e z).

Sono disponibili alcune abbreviazioni per il qualificatore **errors**: **yerrors** (per adattamenti (fits) con 1 colonna di variabile indipendente), e **zerrors** (per il caso generale) sono tutti equivalenti a **errors z**, indicando che è presente una singola colonna extra con gli errori della variabile dipendente.

xyerrors, per il caso di 1 variabile indipendente, indica che ci sono due colonne extra, con gli errori sia della variabile indipendente che della variabile dipendente. In questo caso gli errori su x e y sono trattati con il metodo della varianza efficace di Orear.

Si noti che **yerror** e **xyerror** sono simili sia nella forma che nell'interpretazione agli stili di grafici 2D **yerrorlines** e **xyerrorlines**.

Con il comando **set fit v4** la sintassi del comando fit è compatibile con la versione 4 di **gnuplot**. In questo caso devono essere presenti due qualificatori **using** in più (z e s) rispetto alle variabili indipendenti, a meno che non vi sia una sola variabile. **gnuplot** utilizza quindi i seguenti formati, a seconda del numero di colonne dato nella specificazione **using**:

```
z                       # 1 variabile indipendente (numero linea)
x:z                     # 1 variabile indipendente (1° colonna)
x:z:s                   # 1 variabile indipendente (3 colonne totali)
x:y:z:s                 # 2 variabili indipendenti (4 colonne totali)
x1:x2:x3:z:s            # 3 variabili indipendenti (5 colonne totali)
x1:x2:x3:...:xN:z:s     # N variabili indipendenti (N+2 colonne totali)
```

Si prega di fare attenzione al fatto che questo significa che è necessario fornire z-errors s in un fit con due o più variabili indipendenti. Se si vogliono i pesi unitari è necessario fornirli esplicitamente usando ad es. il formato x:y:z:(1).

I nomi delle variabili dummy (fittizie) possono essere cambiati quando si specifica un intervallo come indicato sopra. Il primo intervallo corrisponde alla prima spec. **using**, e così via. Un intervallo può essere fornito anche per z (la variabile dipendente), nel qual caso i data point per cui f(x,...) è fuori dall'intervallo z non contribuiranno a minimizzare il residuo.

Multipli dataset possono essere adattati (fit) simultaneamente con funzioni di una variabile indipendente, rendendo y una 'pseudo-variabile', ad es., il numero dataline, e adattandola come due variabili indipendenti. Vedere **fit multi-branch (p. 101)**.

Il qualificatore **via** specifica quali parametri devono essere ottimizzati, o direttamente, o facendo riferimento ad un file di parametri.

Esempi:

```
f(x) = a*x**2 + b*x + c
g(x,y) = a*x**2 + b*y**2 + c*x*y
set fit limit 1e-6
fit f(x) 'measured.dat' via 'start.par'
fit f(x) 'measured.dat' using 3:($7-5) via 'start.par'
fit f(x) './data/trash.dat' using 1:2:3 yerror via a, b, c
fit g(x,y) 'surface.dat' using 1:2:3 via a, b, c
fit a0 + a1*x/(1 + a2*x/(1 + a3*x)) 'measured.dat' via a0,a1,a2,a3
fit a*x + b*y 'surface.dat' using 1:2:3 via a,b
fit [*:*][yaks=*:*] a*x+b*yaks 'surface.dat' u 1:2:3 via a,b

fit [][][t=*:*] a*x + b*y + c*t 'foo.dat' using 1:2:3:4 via a,b,c

set dummy x1, x2, x3, x4, x5
h(x1,x2,x3,x4,s5) = a*x1 + b*x2 + c*x3 + d*x4 + e*x5
fit h(x1,x2,x3,x4,x5) 'foo.dat' using 1:2:3:4:5:6 via a,b,c,d,e
```

Dopo ogni step dell'iterazione, informazioni dettagliate sullo stato attuale dell'adattamento vengono scritte sul display. La stessa informazione sugli stati iniziale e finale è scritta in un file di log, "fit.log". Questo file è sempre allegato, in modo da non perdere qualsiasi cronologia fit precedente; esso dovrebbe essere cancellato o rinominato come desiderato. Usando il comando **set fit logfile**, il nome del file log può essere cambiato.

Se attivato usando **set fit errorvariables**, l'errore di ogni parametro adattato sarà memorizzato in una variabile chiamata come il parametro, ma con "_err" aggiunto dopo. Così gli errori possono essere usati come input per ulteriori computazioni.

Se **set fit prescale** viene attivato, i parametri di adattamento sono prescalati dai loro valori iniziali. Questo aiuta la routine Marquardt-Levenberg a convergere più rapidamente e in modo affidabile nei casi in cui i parametri differiscono di diversi ordini di grandezza.

L'adattamento può essere interrotto premendo Ctrl-C (Ctrl-Break in wgnuplot). Dopo il completamento dell'iterazione corrente, si ha l'opzione di (1) interrompere l'adattamento e accettare i valori attuali dei parametri, (2) continuare l'adattamento, (3) eseguire un comando **gnuplot** come specificato da **set fit script** o dalla variabile d'ambiente **FIT_SCRIPT**. Il default è **replot**, quindi se precedentemente si ha plottato sia i dati che la funzione di adattamento in un grafico (graph), è possibile visualizzare lo stato attuale dell'adattamento.

Una volta che **fit** ha finito, il comando **save fit** può essere usato per memorizzare i valori finali in un file per un successivo utilizzo come un file di parametri. Vedere **save fit (p. 135)** per i dettagli.

Parametri regolabili

Ci sono due modi in cui **via** può specificare i parametri da regolare, o direttamente sulla linea di comando o indirettamente, facendo riferimento ad un file di parametri. I due modi usano mezzi diversi per fissare i valori iniziali.

I parametri regolabili possono essere specificati da una lista separata da virgole di nomi di variabili dopo la keyword **via**. Qualsiasi variabile che non è già definita viene creata con un valore iniziale di 1.0. Tuttavia, è più probabile che l'adattamento converga rapidamente se le variabili sono state precedentemente dichiarate con valori iniziali più appropriati.

In un file di parametri, ogni parametro da variare e un valore iniziale corrispondente sono specificati, uno per riga, nella forma

```
varname = value
```

I commenti, contrassegnati da '#', e le linee vuote sono ammessi. La forma speciale

```
varname = value       # FIXED
```

significa che la variabile è trattata come un 'fixed parameter' (parametro fisso), inizializzata dal file del parametro ma non regolata da **fit**. Per chiarezza, può essere utile designare le variabili come parametri fissi in modo che i loro valori siano riportati da **fit**. La keyword **# FIXED** deve apparire esattamente in questa forma.

Breve introduzione

fit è usato per trovare un insieme di parametri che 'meglio' adatta i propri dati alla propria funzione definita dall'utente. L'adattamento è giudicato sulla base della somma delle differenze quadratiche o 'residui' (SSR) tra i data point di input e i valori della funzione, valutati negli stessi punti. Questa quantità è spesso chiamata 'chisquare' (cioè, la lettera greca chi, alla potenza di 2). L'algoritmo cerca di minimizzare gli SSR, o più precisamente, WSSR, poiché i residui sono 'ponderati' (weighted) dagli errori dei dati di input (o 1.0) prima di venire elevati al quadrato; vedere **fit error_estimates (p. 99)** per i dettagli.

Ecco perché si chiama 'least-squares fitting'. Guardiamo un esempio per vedere cosa si intende per 'non lineare', ma prima è meglio ripassare alcuni termini. Qui è pratico usare z come variabile dipendente per funzioni definite dall'utente o di una variabile indipendente, z=f(x), o di due variabili indipendenti, z=f(x,y). Un parametro è una variabile definita dall'utente che **fit** regolerà, cioè una quantità sconosciuta nella dichiarazione della funzione. La linearità/non linearità si riferisce alla relazione della variabile dipendente, z, ai parametri che **fit** sta regolando, non di z alle variabili indipendenti, x e/o y. (Per essere tecnici, le derivate seconde {e superiori} della funzione di adattamento rispetto ai parametri sono zero per un problema lineare ai minimi quadrati).

Per i minimi quadrati lineari (LLS), la funzione definita dall'utente sarà una somma di funzioni semplici, senza alcun parametro, ciascuna moltiplicata per un parametro. NLLS gestisce funzioni più complicate in cui i parametri possono essere utilizzati in un gran numero di modi. Un esempio che illustra la differenza tra minimi quadrati lineari e non lineari è la serie di Fourier. Un componente può essere scritto come

```
z=a*sin(c*x) + b*cos(c*x).
```

Se a e b sono i parametri sconosciuti e c è costante, allora la stima dei parametri è un problema ai minimi quadrati lineari. Tuttavia, se c è un parametro sconosciuto, il problema non è lineare.

Nel caso lineare, i valori dei parametri possono essere determinati da un'algebra lineare relativamente semplice, in un passo diretto. Tuttavia LLS è un caso speciale che viene risolto insieme a problemi NLLS più generali dalla procedura iterativa utilizzata da **gnuplot**. **fit** cerca di trovare il minimo facendo una ricerca. Ogni passo (iterazione) calcola WSSR con un nuovo set di valori dei parametri. L'algoritmo Marquardt-Levenberg seleziona i valori dei parametri per l'iterazione successiva. Il processo continua fino a quando un criterio prestabilito è soddisfatto, o (1) il fit è "convergente" (il cambiamento relativo in WSSR è inferiore a un certo limite, vedere **set fit limit (p. 155)**), o (2) raggiunge un limite di conteggio di iterazione preimpostato (vedere **set fit maxiter (p. 155)**). Il fit può anche essere interrotto e successivamente arrestato dalla tastiera (vedere **fit (p. 95)**). La variabile utente FIT_CONVERGED contiene 1 se il precedente comando fit è terminato a causa di convergenza; contiene 0 se il precedente fit è terminato per qualsiasi altro motivo. FIT_NITER contiene il numero di iterazioni che sono state fatte durante l'ultimo fit.

Spesso la funzione da adattare sarà basata su un modello (o teoria) che tenta di descrivere o prevedere il comportamento dei dati. Quindi **fit** può essere usato per trovare valori per i parametri liberi del modello, per determinare quanto bene i dati si adattano al modello, e per stimare un intervallo di errore per ogni parametro. Vedere **fit error_estimates (p. 99)**.

In alternativa, nel curve-fitting, le funzioni sono selezionate indipendentemente da un modello (in base all'esperienza su quale è probabile descrivere l'andamento dei dati con la risoluzione desiderata e un numero minimo di parametri*funzione). La soluzione **fit** fornisce quindi una rappresentazione analitica della curva.

Tuttavia, se tutto quello che si desidera è una curva uniforme attraverso i propri data point, l'opzione **smooth** di **plot** potrebbe essere quello che si sta cercando invece di **fit**.

Stime di errore

In **fit**, il termine "error" è usato in due contesti diversi, stime di errore dei dati e stime di errore dei parametri.

Le stime di errori dei dati sono utilizzate per calcolare il peso relativo di ogni data point quando si determina la somma ponderata dei residui al quadrato, WSSR o chisquare. Possono influenzare le stime dei parametri, poiché determinano quanta influenza la deviazione di ogni data point dalla funzione adattata ha sui valori finali. Alcune delle informazioni output di **fit**, incluse le stime di errore dei parametri, sono più significative se sono state fornite stime accurate di errore dei dati.

La **statistical overview** (panoramica statistica) descrive alcuni dei **fit** di output e fornisce un po' di background per le 'practical guidelines' (indicazioni pratiche).

Panoramica statistica

La teoria dei minimi quadrati non lineari (NLLS) è generalmente descritta in termini di una distribuzione normale degli errori, cioè si assume che i dati di input siano un campione da una popolazione con una data media e una distribuzione gaussiana (normale) intorno alla media con una data deviazione standard. Per un campione sufficentemente grande e conoscendo la deviazione standard della popolazione, si può usare la statistica della distribuzione chisquare per descrivere una "goodness of fit" (bontà di adattamento) guardando la variabile spesso chiamata "chisquare". Qui, è sufficiente dire che un chisquare ridotto (chisquare/gradi di libertà, dove i gradi di libertà sono il numero di data point meno il numero di parametri che stanno venendo adattati) di 1.0 è un'indicazione che la somma ponderata di deviazioni al quadrato tra la funzione adattata e i data point è uguale a quella prevista per un campione casuale preso da una popolazione caratterizzata dalla funzione con il valore attuale dei parametri e le deviazioni standard date.

Se la deviazione standard della popolazione non è costante, come nella statistica di conteggio in cui varianza = conteggi, allora ogni punto dovrebbe essere ponderato individualmente quando si confronta la somma osservata delle deviazioni e la somma prevista di deviazioni.

Alla conclusione **fit** riporta 'stdfit', la deviazione standard del fit, che è il rms dei residui, e la varianza dei residui, chiamata anche 'chisquare ridotto' quando i data point sono ponderati. Il numero di gradi di libertà (il numero di data point meno il numero di parametri adattati) è usato in queste stime perché i parametri usati nel calcolare i residui dei data point sono stati ottenuti dagli stessi dati. Se i data point hanno dei pesi, **gnuplot** calcola il cosiddetto valore p, cioè uno meno la funzione di distribuzione cumulativa della distribuzione chisquare per il numero di gradi di libertà e il risultante chisquare, vedere **practical_guidelines (p. 100)**. Questi valori sono esportati alle variabili

```
FIT_NDF = Number of degrees of freedom
FIT_WSSR = Weighted sum-of-squares residual
FIT_STDFIT = sqrt(WSSR/NDF)
FIT_P = p-value
```

Per stimare i livelli di confidenza per i parametri, si può usare il minimo chisquare ottenuto dalle statistiche di fit e chisquare per determinare il valore di chisquare corrispondente al livello di confidenza desiderato, ma sono necessari molti più calcoli per determinare le combinazioni di parametri che producono tali valori.

Piuttosto che determinare gli intervalli di confidenza, **fit** riporta le stime di errore dei parametri che sono facilmente ottenibili dalla matrice di varianza-covarianza dopo l'iterazione finale. Per convenzione, queste stime sono chiamate "errori standard" o "errori standard asintotici", poiché sono calcolati nello stesso modo degli errori standard (deviazione standard di ogni parametro) di un problema ai minimi quadrati lineari, anche se le condizioni statistiche per designare la quantità calcolata come deviazione standard non sono generalmente valide per il problema NLLS. Gli errori standard asintotici sono generalmente troppo ottimistici e non dovrebbero essere usati per determinare i livelli di confidenza ma sono utili per scopi qualitativi.

La soluzione finale produce anche una matrice di correlazione che indica la correlazione dei parametri nella regione della soluzione. Gli elementi diagonali principali, autocorrelazione, sono sempre 1; se tutti i parametri fossero indipendenti, gli elementi fuori diagonale sarebbero quasi 0. Due variabili che si compensano completamente a vicenda avrebbero un elemento fuori diagonale di grandezza unitaria, con un segno che dipende

dal fatto che la relazione sia proporzionale o inversamente proporzionale. Più piccole sono le grandezze degli elementi fuori diagonale, più le stime della deviazione standard di ogni parametro saranno vicine all'errore standard asintotico.

Indicazioni pratiche

Se si ha una base per assegnare i pesi a ciascun data point, questo permette di fare uso di conoscenze aggiuntive sulle proprie misurazioni, ad esempio tenere conto che alcuni punti possono essere più affidabili di altri. Questo può influenzare i valori finali dei parametri.

La ponderazione dei dati fornisce una base per interpretare il **fit** di output aggiuntivo dopo l'ultima iterazione. Anche se si pesa ogni punto in modo uguale, stimare una deviazione standard media piuttosto che usare un peso di 1 rende il WSSR una variabile priva di dimensione, come il chisquare lo è per definizione.

Ogni iterazione di fit mostrerà informazioni che possono essere utilizzate per valutare il progresso dell'adattamento. (Un '*' indica che non ha trovato un WSSR più piccolo e sta provando di nuovo.) La 'somma dei quadrati dei residui', chiamata anche 'chisquare', è il WSSR tra i dati e la propria funzione adattata; **fit** l'ha minimizzata. In questa fase, con dati ponderati, ci si aspetta che chisquare si avvicini al numero di gradi di libertà (data point meno parametri). Il WSSR può essere usato per calcolare il chisquare ridotto (WSSR/ndf) o stdfit, la deviazione standard del fit, sqrt(WSSR/ndf). Entrambi sono riportati per il WSSR finale.

Se i dati non sono pesati, stdfit è il valore efficace rms della deviazione dei dati dalla funzione adattata, in unità utente.

Se sono stati forniti validi errori di dati, il numero di data point è abbastanza grande, e il modello è corretto, il chisquare ridotto dovrebbe essere circa l'unità. (Per i dettagli, cercare la 'distribuzione chi-quadrato' nel proprio riferimento statistico preferito.) Se è così, ci sono ulteriori test, oltre lo scopo di questa panoramica, per determinare quanto bene il modello si adatti ai dati.

Un chisquare ridotto molto più grande di 1.0 può essere dovuto a stime errate dell'errore dei dati, errori di dati non distribuiti normalmente, errori di misura sistematici, 'outlier', o una funzione di modello errata. Un grafico dei residui, ad es., **plot 'datafile' using 1:($2-f($1))**, può aiutare a mostrare qualsiasi tendenza sistematica. Plottare sia i data point che la funzione può aiutare a suggerire un altro modello.

Allo stesso modo, un chisquare ridotto inferiore a 1.0 indica che WSSR è inferiore a quello previsto per un campione casuale dalla funzione con errori normalmente distribuiti. Le stime di errore dei dati possono essere troppo grandi, le ipotesi statistiche potrebbero non essere giustificate, o la funzione di modello potrebbe essere troppo generale, adattando le fluttuazioni in un particolare campione oltre alle tendenze sottostanti. In quest'ultimo caso, una funzione più semplice può essere più appropriata.

Il valore p del fit è uno meno la funzione di distribuzione cumulativa della distribuzione chisquare per il numero di gradi di libertà e il chisquare risultante. Questo può servire come misura della bontà dell'adattamento. L'intervallo del valore p è compreso tra zero e uno. Un valore p molto piccolo o grande indica che il modello non descrive bene i dati e i suoi errori. Come descritto sopra, questo potrebbe indicare un problema con i dati, i suoi errori o il modello, o una combinazione di essi. Un valore p piccolo potrebbe indicare che gli errori sono stati sottostimati e gli errori dei parametri finali dovrebbero quindi essere scalati. Vedere anche **set fit errorscaling (p. 155)**.

Bisogna abituarsi sia al **fit** che al tipo di problemi a cui lo si applica prima di poter mettere in relazione gli errori standard con alcune stime più pratiche delle incertezze dei parametri o valutare la significatività della matrice di correlazione.

Si noti che **fit**, in comune con la maggior parte delle implementazioni NLLS, minimizza la somma pesata delle distanze quadrate (y-f(x))**2. Non fornisce alcun mezzo per rendere conto degli "errori" nei valori di x, solo in y. Inoltre, qualsiasi "outlier" (data point al di fuori della distribuzione normale del modello) avrà un effetto esagerato sulla soluzione.

Control

Ci sono un certo numero di variabili d'ambiente che possono essere definite per influenzare **fit** prima di avviare **gnuplot**, vedere **fit control environment (p. 101)**. In fase di esecuzione le regolazioni dell'operazione del comando **fit** possono essere controllate da **set fit**. Vedere **fit control variables (p. 101)**.

Variabili di controllo

DEPRECATO nella versione 5. Queste variabili dell'utente influenzavano il comportamento di fit.

```
FIT_LIMIT - use ‘set fit limit <epsilon>‘
FIT_MAXITER - use ‘set fit maxiter <number_of_cycles>‘
FIT_START_LAMBDA - use ‘set fit start-lambda <value>‘
FIT_LAMBDA_FACTOR - use ‘set fit lambda-factor <value>‘
FIT_SKIP - use the datafile ‘every‘ modifier
FIT_INDEX - See ‘fit multi-branch‘
```

Variabili d'ambiente

Le variabili d'ambiente devono essere definite prima dell'esecuzione di **gnuplot**; come farlo dipende dal proprio sistema operativo.

```
FIT_LOG
```

cambia il nome (e/o il percorso) del file in cui verrà scritto il fit log dall'impostazione predefinita di "fit.log" nella directory di lavoro. Il valore predefinito può essere sovrascritto usando il comando **set fit logfile**.

```
FIT_SCRIPT
```

specifica un comando che può essere eseguito dopo un interrupt dell'utente. L'impostazione predefinita è **replot**, ma un comando **plot** o **load** può essere utile per visualizzare un grafico personalizzato per mettere in evidenza il progresso del fit. Questa impostazione può essere cambiata anche usando **set fit script**.

Multi-branch

In adattamenti multi-branch, più insiemi di dati possono essere adattati simultaneamente, con funzioni di una variabile indipendente che hanno parametri comuni, minimizzando il WSSR totale. La funzione e i parametri (branch/ramo) per ogni set di dati sono selezionati usando una 'pseudo-variabile', ad es., o il numero di dataline (un indice di 'colonna' di -1) o l'indice del file di dati (-2), come la seconda variabile indipendente.

Esempio: Dati due decadimenti esponenziali della forma, z=f(x), ognuno dei quali descrive un diverso set di dati ma con un tempo di decadimento comune, stimare i valori dei parametri. Se il file di dati ha il formato x:z:s, allora

```
f(x,y) = (y==0) ? a*exp(-x/tau) : b*exp(-x/tau)
fit f(x,y) 'datafile' using  1:-2:2:3  via a, b, tau
```

Per un esempio più complicato, vedere il file "hexa.fnc" usato dalla demo "fit.dem".

Potrebbe essere necessaria una ponderazione appropriata poiché i pesi unitari possono far predominare un ramo, se c'è una differenza nella scala della variabile dipendente. Adattare ciascun ramo separamente, usando la soluzione multi-branch come valori iniziali, potrebbe dare un'indicazione sull'effetto relativo di ogni ramo sulla soluzione congiunta.

Valori iniziali

L'adattamento non lineare non è garantito per convergere verso l'ottimo globale (la soluzione con la più piccola somma dei residui quadratici, SSR), e può bloccarsi ad un minimo locale. La routine non ha modo di determinarlo, spetta a voi giudicare se questo è successo.

fit può, e spesso si "perde" se viene avviato lontano da una soluzione, dove SSR è grande e cambia lentamente man mano che i parametri vengono variati, o può raggiungere una regione numericamente instabile (ad esempio, un numero troppo grande che causa un overflow in virgola mobile) che risulta in un messaggio di "undefined value" (valore non definito) o **gnuplot** si ferma.

Per migliorare le possibilità di trovare l'ottimo globale, si dovrebbero impostare i valori di partenza almeno approssimativamente nelle vicinanze della soluzione, ad esempio, entro un ordine di grandezza, se possibile. Più i propri valori di partenza sono vicini alla soluzione, meno possibilità ci sono di fermarsi ad un falso minimo. Un modo per trovare i valori di partenza è quello di plottare i dati e la funzione di adattamento sullo stesso grafico (graph) e cambiare i valori dei parametri e fare **replot** fino a raggiungere una ragionevole somiglianza. Lo stesso grafico è anche utile per controllare se il fit ha trovato un falso minimo.

Naturalmente trovare un buon fit non prova che non ci sia un fit "migliore" (o in senso statistico, caratterizzato da un miglior criterio di goodness of fit, o in senso fisico, con una soluzione più coerente con il modello). A seconda del problema, può essere desiderabile "adattare" **fit** con varie serie di valori di partenza, che coprono un intervallo ragionevole per ogni parametro.

Tips

Ecco alcuni consigli da tenere a mente per ottenere il massimo dal **fit**. Non sono molto organizzati, quindi sarà necessario leggerli più volte fino a quando non si saranno compresi pienamente.

Le due forme dell'argomento **via** di **fit** servono a due scopi ampiamente distinti. La forma **via "file"** è meglio usata per operazioni batch (possibilmente non presidiate) dove si forniscono i valori dei parametri di partenza in un file.

La forma **via var1, var2, ...** è usata meglio in maniera interattiva, dove il meccanismo di command history può essere usato per modificare la lista dei parametri da adattare o per fornire nuovi valori startup per il prossimo tentativo. Questo è particolarmente utile per problemi difficili, dove un adattamento diretto a tutti i parametri in una sola volta non funzionerà senza buoni valori di partenza. Per trovarlo, si può iterare più volte, adattando solo alcuni dei parametri, finché i valori sono abbastanza vicini all'obiettivo che l'adattamento finale a tutti i parametri contemporaneamente funzionerà.

Bisogna assicurarsi che non ci sia dipendenza reciproca tra i parametri della funzione che si sta adattando. Per esempio, non si deve cercare di adattare a*exp(x+b), perché a*exp(x+b)=a*exp(b)*exp(x). Adattare invece o a*exp(x) o exp(x+b).

Una questione tecnica: Maggiore è il rapporto tra i valori dei parametri assoluti più grande e più piccolo, più lenta sarà la convergenza dell'adattamento. Se il rapporto è vicino o superiore all'inverso della precisione in virgola mobile della macchina, potrebbe impiegare un'eternità per convergere, o rifiutarsi di convergere del tutto. Per evitare questo si dovrà adattare la propria funzione, ad es., sostituire 'parameter' con '1e9*parameter' nella definizione della funzione e dividere il valore iniziale per 1e9, o usare **set fit prescale** che lo esegue internamente in base ai valori di partenza dei parametri.

Se si può scrivere la propria funzione come una combinazione lineare di funzioni semplici ponderate dai parametri da adattare, è consigliabile farlo. Questo è molto d'aiuto, perché il problema non è più non lineare e dovrebbe convergere con solo un piccolo numero di iterazioni, forse solo una.

Alcune prescrizioni per l'analisi dei dati, date nei corsi di sperimentazione pratica, possono far prima adattare alcune funzioni ai propri dati, forse in un processo a più fasi di contabilizzazione di diversi aspetti della teoria sottostante uno per uno, e poi estrarre le informazioni che si volevano veramente dai parametri di adattamento di quelle funzioni. Con **fit**, questo può spesso essere fatto in un solo passo scrivendo la funzione di modello direttamente nei termini dei parametri desiderati. Anche la trasformazione dei dati può essere spesso evitata,

sebbene a volte al costo di un problema di adattamento più difficile. Se pensate che questo contraddica il paragrafo precedente sulla semplificazione della funzione di adattamento, avete ragione.

Un messaggio "singular matrix" indica che questa implementazione dell'algoritmo Marquardt-Levenberg non può calcolare i valori dei parametri per la prossima iterazione. Bisogna provare diversi valori di partenza, scrivendo la funzione in un'altra forma, o una funzione più semplice.

Infine, una bella citazione dal manuale di un altro pacchetto di fitting (fudgit), che riassume un po' tutti questi problemi: "Il fitting non lineare è un'arte!"

Help

Il comando **help** visualizza l'aiuto integrato. Per informazioni specifiche su un particolare argomento usare la sintassi:

```
help {<topic>}
```

Se <topic> non è specificato, viene stampato un breve messaggio su **gnuplot**. Dopo aver fornito l'aiuto per l'argomento richiesto, viene mostrato un menu di sotto-argomenti; l'aiuto per un sottoargomento può essere richiesto digitando il suo nome, estendendo la richiesta di aiuto. Dopo che quel sottoargomento è stato stampato, la richiesta può essere estesa di nuovo o si può tornare indietro di un livello all'argomento precedente. Alla fine, ritornerà la linea di comando **gnuplot**.

Se un punto interrogativo (?) è dato come topic, viene visualizzato sullo schermo l'elenco degli argomenti attualmente disponibile.

History

Il comando **history** stampa o salva i comandi precedenti nella cronologia, o riesegue una entry precedente nell'elenco. Per modificare il comportamento di questo comando, vedere **set history (p. 164)**.

Le linee di input con **history** come primo comando non sono memorizzate nella command history.

Esempi:

```
history               # mostra la cronologia completa
history 5             # mostra le ultime 5 entry della cronologia
history quiet 5       # mostra le ultime 5 entry senza numeri di inserimento
history "hist.gp"     # scrivi la cronologia completa nel file hist.gp
history "hist.gp" append # apponi la cronologia completa nel file hist.gp
history 10 "hist.gp"  # scrivi gli ultimi 10 comandi nel file hist.gp
history 10 "|head -5 >>diary.gp" # scrivi 5 comandi history usando pipe
history ?load         # mostra tutte le entry della cronologia che iniziano con "load"
history ?"set c"      # come sopra, diverse parole racchiuse tra virgolette
hist !"set xr"        # come sopra, diverse parole racchiuse tra virgolette
hist !55              # riesegui il comando alla entry 55 della cronologia
```

If

Nuova sintassi:

```
if (<condition>) { <commands>;
       <commands>
       <commands>
} else {
```

```
        <commands>
}
```

Vecchia sintassi:

```
if (<condition>) <command-line> [; else if (<condition>) ...; else ...]
```

Questa versione di gnuplot supporta le istruzioni if/else strutturate a blocchi. Se la keyword **if** o **else** è immediatamente seguita da un "{" di apertura, allora l'esecuzione condizionale si applica a tutte le istruzioni, possibilmente su multiple linee di input, fino a quando un "}" corrispondente termina il blocco. I comandi if possono essere annidati.

La vecchia sintassi if/else a linea singola è ancora supportata, ma non può essere mescolata con la nuova sintassi strutturata a blocchi. Vedere **if-old (p. 104)**.

If-old

Fino alla versione 4.4 di gnuplot, l'ambito dei comandi if/else era limitato a una singola linea di input. Ora una clausola multilinea può essere racchiusa tra parentesi graffe. La vecchia sintassi è ancora rispettata ma non può essere usata all'interno di una clausola tra parentesi.

Se nessun "{" di apertura segue la parola chiave **if**, il comando o i comandi in <command-line> saranno eseguiti se la condizione <condition> è vera (diversa da zero) o saltati se <condition> è falsa (zero). Entrambi i casi consumeranno i comandi sulla linea di input fino alla fine della linea o un'occorrenza di **else**. Si noti che l'uso di **;** per permettere comandi multipli sulla stessa linea *non* terminerà i comandi condizionati.

Esempi:

```
pi=3
if (pi!=acos(-1)) print "?Fixing pi!"; pi=acos(-1); print pi
```

mostrerà:

```
?Fixing pi!
3.14159265358979
```

ma

```
if (1==2) print "Never see this"; print "Or this either"
```

non mostrerà nulla.

else (altrimenti):

```
v=0
v=v+1; if (v%2) print "2" ; else if (v%3) print "3"; else print "fred"
```

(ripetere ripetutamente l'ultima riga!)

For

I comandi **plot**, **splot**, **set** e **unset** possono eventualmente contenere un'iterazione per clausola. Questo ha l'effetto di eseguire il comando di base più volte, ogni volta rivalutando qualsiasi espressione che faccia uso della variabile di controllo dell'iterazione. L'iterazione di sequenze di comandi arbitrari può essere richiesta usando il comando **do**. Attualmente sono supportate due forme di clausola di iterazione:

```
for [intvar = start:end{:increment}]
for [stringvar in "A B C D"]
```

Esempi:

```
plot for [filename in "A.dat B.dat C.dat"] filename using 1:2 with lines
plot for [basename in "A B C"] basename.".dat" using 1:2 with lines
set for [i = 1:10] style line i lc rgb "blue"
unset for [tag = 100:200] label tag
```

L'iterazione annidata è supportata:

```
set for [i=1:9] for [j=1:9] label i*10+j sprintf("%d",i*10+j) at i,j
```

Vedere la documentazione aggiuntiva per **iteration (p. 49)**, **do (p. 94)**.

Import

Il comando **import** associa un nome di funzione definito dall'utente ad una funzione esportata da un oggetto esterno condiviso. Questo costituisce un meccanismo di plugin che estende l'insieme delle funzioni disponibili in gnuplot. Vedere **plugins (p. 57)**.

Sintassi:

```
import func(x[,y,z,...]) from "sharedobj[:symbol]"
```

Esempi:

```
# rendi la funzione myfun, esportata da "mylib.so" o "mylib.dll"
# disponibile per il plotting o il calcolo numerico in gnuplot
import myfun(x) from "mylib"
import myfun(x) from "mylib:myfun"    # same as above

# rendi la funzione theirfun, definita in "theirlib.so" o "theirlib.dll"
# disponibile sotto un nome diverso
import myfun(x,y,z) from "theirlib:theirfun"
```

Il programma estende il nome dato per l'oggetto condiviso con ".so" o ".dll" a seconda del sistema operativo, e lo cerca prima come nome di percorso completo e poi come percorso relativo alla directory corrente. Il sistema operativo stesso può anche cercare qualsiasi directory in LD_LIBRARY_PATH o DYLD_LIBRARY_PATH.

Load

Il comando **load** esegue ogni linea del file di input specificato come se fosse stata digitata interattivamente. I file creati dal comando **save** possono successivamente essere caricati con **load**. Qualsiasi file di testo contenente comandi validi può essere creato e poi eseguito dal comando **load**. I file che vengono caricati con **load** possono essi stessi contenere comandi **load** o **call**. Vedere **commenti (p. 33)** per informazioni su commenti nei comandi. Per **load** con argomenti, vedere **call (p. 91)**.

Sintassi:

```
load "<input-file>"
```

Il nome del file di input deve essere racchiuso tra virgolette.

Il filename speciale "-" può essere usato per caricare con **load** i comandi dall'input standard. Questo permette ad un file di comando **gnuplot** di accettare alcuni comandi dall'input standard. Vedere l'aiuto per **batch/interactive (p. 31)** per maggiori dettagli.

Su alcuni sistemi che supportano una funzione popen (Unix), il load file può essere letto da una pipe facendo iniziare il nome del file con un '<'.

Esempi:

```
load 'work.gnu'
load "func.dat"
load "< loadfile_generator.sh"
```

Il comando **load** viene eseguito implicitamente su qualsiasi nome di file dato come argomento a **gnuplot**. Questi vengono caricati nell'ordine specificato, e poi **gnuplot** esce.

Lower

Vedere **raise (p. 134)**.

Pause

Il comando **pause** visualizza qualsiasi testo associato al comando e poi aspetta una quantità di tempo specificata o finché non viene premuto il ritorno a capo. **pause** è particolarmente utile in combinazione con i file **load**.

Sintassi:

```
pause <time> {"<string>"}
pause mouse {<endcondition>}{, <endcondition>} {"<string>"}
pause mouse close
```

<time> può essere qualsiasi costante o espressione in virgola mobile. **pause -1** aspetterà fino al ritorno a capo, zero (0) non farà alcuna pausa, e un numero positivo aspetterà il numero di secondi specificato. **pause 0** è sinonimo di **print**.

Se il terminale corrente supporta **mousing**, allora **pause mouse** terminerà su un clic del mouse o su ctrl-C. Per tutti gli altri terminali, o se l'uso del mouse non è attivo, **pause mouse** equivale a **pause -1**.

Se una o più condizioni finali sono date dopo **pause mouse**, allora una qualsiasi delle condizioni terminerà la pausa. Le possibili condizioni finali sono **keypress**, **button1**, **button2**, **button3**, **close**, e **any**. Se la pausa termina con la pressione di un tasto, allora il valore ascii del tasto premuto viene restituito in MOUSE_KEY. Il carattere stesso viene restituito come stringa di un carattere in MOUSE_CHAR. I tasti di scelta rapida (comando bind) sono disabilitati se la pressione del tasto è una delle condizioni finali. Lo zoom è disattivato se button3 è una delle condizioni finali.

In tutti i casi le coordinate del mouse sono restituite in variabili MOUSE_X, MOUSE_Y, MOUSE_X2, MOUSE_Y2. Vedere **variabili mouse (p. 55)**.

Nota: Poiché **pause** comunica con il sistema operativo piuttosto che con la grafica, può comportarsi diversamente con diversi device driver (a seconda di come testo e grafica sono mescolati).

Esempio:

```
pause -1    # Aspetta fino a quando non viene premuto un ritorno a capo
pause 3     # Aspetta tre secondi
pause -1  "Hit return to continue"
pause 10  "Isn't this pretty?  It's a cubic spline."
pause mouse "Click any mouse button on selected data point"
pause mouse keypress "Type a letter from A-F in the active window"
pause mouse button1,keypress
pause mouse any "Any key or button will terminate"
```

La variante "pause mouse key" riprenderà dopo qualsiasi pressione di un tasto nella finestra di grafico attiva. Se si desidera aspettare la pressione di un tasto particolare, è possibile usare un loop come:

```
print "I will resume after you hit the Tab key in the plot window"
```

```
plot <something>
pause mouse key
while (MOUSE_KEY != 9) {
    pause mouse key
}
```

Pause mouse close

Il comando **pause mouse close** è un esempio specifico di pausa per aspettare un evento esterno. In questo caso il programma attende un evento di chiusura "close" dalla finestra del grafico. Esattamente come generare un tale evento varia con il proprio ambiente desktop e con la configurazione, ma di solito è possibile chiudere la finestra del grafico cliccando su qualche widget sul bordo della finestra o digitando una sequenza di tasti di scelta rapida come <alt><F4> o <ctrl>q. Se non si è sicuri che un widget o un tasto di scelta rapida adatto sia disponibile all'utente, si può anche definire una sequenza di tasti di scelta rapida utilizzando il meccanismo proprio di gnuplot. Vedere **bind (p. 54)**.

La sequenza di comandi che segue può essere utile quando si esegue gnuplot da uno script piuttosto che dalla linea di comando.

```
plot <...whatever...>
bind all "alt-End" "exit gnuplot"
pause mouse close
```

Plot

plot è il comando principale per disegnare grafici con **gnuplot**. Questo comando offre diverse rappresentazioni grafiche per funzioni e dati. **plot** è usato per disegnare funzioni e dati 2D. **splot** disegna proiezioni 2D di superfici e dati 3D.

Sintassi:

```
plot {<ranges>} <plot-element> {, <plot-element>, <plot-element>}
```

Ogni elemento del grafico consiste in una definizione, una funzione o un sorgente dati insieme a proprietà o modificatori opzionali:

```
plot-element:
     {<iteration>}
     <definition> | {sampling-range} <function> | <data source>
                  | keyentry
     {axes <axes>} {<title-spec>}
     {with <style>}
```

La rappresentazione grafica di ogni elemento del grafico è determinata dalla keyword **with**, ad es. **with lines** o **with boxplot**. Vedere **plotting styles (p. 63)**.

I dati da plottare sono o generati da una funzione (due funzioni se in modalità parametrica), o letti da un file di dati o letti da un blocco di dati con nome che è definito in precedenza. Più file di dati, blocchi di dati e/o funzioni possono essere plottati in un singolo comando plot separati da virgole. Vedere **data (p. 112)**, **inline data (p. 48)**, **functions (p. 126)**.

Un elemento di un grafico che contiene la definizione di una funzione o di una variabile non crea alcun output visibile, vedere il terzo esempio.

Esempi:

```
plot sin(x)
plot sin(x), cos(x)
```

```
plot f(x) = sin(x*a), a = .2, f(x), a = .4, f(x)
plot "datafile.1" with lines, "datafile.2" with points
plot [t=1:10] [-pi:pi*2] tan(t), \
     "data.1" using (tan($2)):($3/$4) smooth csplines \
              axes x1y2 notitle with lines 5
plot for [datafile in "spinach.dat broccoli.dat"] datafile
```

Vedere anche **show plot (p. 197)**.

Axes

Ci sono quattro possibili set di assi disponibili; la keyword <axes> è usata per selezionare gli assi per i quali una particolare linea dovrebbe essere scalata. **x1y1** si riferisce agli assi in basso e a sinistra; **x2y2** a quelli in alto e a destra; **x1y2** a quelli in basso e a destra; e **x2y1** a quelli in alto e a sinistra. Gli intervalli specificati nel comando **plot** si applicano solo alla prima serie di assi (in basso a sinistra).

Binary

BINARY DATA FILES:

È necessario fornire la keyword **binary** dopo il nome del file. Dettagli adeguati sul formato del file devono essere forniti sulla linea di comando o estratti dal file stesso per un **filetype** binario supportato. In particolare, ci sono due strutture per i file binari, il formato binary matrix (matrice binaria) e il formato binary general (generale binario).

Il formato **binary matrix** contiene un array bidimensionale di valori float IEEE a 32 bit più una colonna e una riga aggiuntive di valori di coordinate. Nello specificatore **using** di un comando plot, la colonna 1 si riferisce alla coordinata della riga della matrice, la colonna 2 si riferisce alla coordinata della colonna della matrice, e la colonna 3 si riferisce al valore memorizzato nell'array a quelle coordinate.

Il formato **binary general** contiene un numero arbitrario di colonne per le quali devono essere specificate informazioni alla linea di comando. Per esempio, **array**, **record**, **format** e **using** possono indicare la dimensione (size), il formato e la dimensione (dimension) dei dati. Vi sono una varietà di comandi utili per saltare le intestazioni dei file e cambiare l'endianess. Ci sono una serie di comandi per posizionare e tradurre i dati, siccome spesso le coordinate non fanno parte del file quando il campionamento uniforme è inerente ai dati. A differenza della lettura da un file di testo o di matrice binaria, il binary general non tratta le colonne generate come 1, 2 o 3 nella lista **using**. Invece la colonna 1 si riferisce alla colonna 1 del file, o come specificato nella lista **format**.

Ci sono impostazioni predefinite globali per le varie opzioni binarie che possono essere impostate usando la stessa sintassi delle opzioni quando sono usate come parte del comando **(s)plot <filename> binary ...**. Questa sintassi è **set datafile binary ...**. La regola generale è che i parametri comuni specificati dalla linea di comando sovrascrivono i parametri estratti dal file che sovrascrivono i parametri predefiniti.

Binary matrix è il formato binario predefinito quando non sono date keyword specifiche per **binary general**, cioè, **array**, **record**, **format**, **filetype**.

I dati binari generali possono essere inseriti nella linea di comando tramite il file name speciale '-'. Tuttavia, questo è inteso per l'uso attraverso un pipe dove i programmi possono scambiare dati binari, non per le tastiere. Non c'è un carattere "end of record" per i dati binari. Gnuplot continua a leggere da una pipe finché ha letto il numero di punti dichiarati nel qualificatore **array**. Vedere **binary matrix (p. 238)** o **binary general (p. 109)** per ulteriori dettagli.

La keyword **index** non è supportata, poiché il formato del file permette solo una superficie per file. I filtri **every** e **using** sono supportati. **using** opera come se i dati fossero letti nella forma di tripletta di cui sopra. `Binary File Splot Demo.`

General

La keyword **binary** che appare da sola indica un file di dati binari che contiene sia informazioni sulle coordinate che descrivono una griglia non uniforme sia il valore di ogni punto della griglia (vedere **binary matrix (p. 238)**). I dati binari in qualsiasi altro formato richiedono ulteriori keyword per descrivere la disposizione dei dati. Sfortunatamente la sintassi di queste keyword aggiuntive richieste è contorta. Tuttavia la modalità general binary è particolarmente utile per programmi applicativi che inviano grandi quantità di dati a gnuplot.

Sintassi:

```
plot '<file_name>' {binary <binary list>} ...
splot '<file_name>' {binary <binary list>} ...
```

Il formato general binary viene attivato dalle keyword in <binary list> che riguardano le informazioni sulla struttura dei file, cioè, **array**, **record**, **format** o **filetype**. Altrimenti, si assume il formato matrix binary non uniforme. (Vedere **binary matrix (p. 238)** per maggiori dettagli).

Gnuplot sa come leggere alcuni tipi di file binari standard che sono completamente autodescrittivi, ad es. le immagini PNG. Digitare **show datafile binary** nella linea di comando per una lista. Oltre a questi, si può pensare ai file di dati binari concettualmente come ai dati di testo. Ogni punto ha colonne di informazioni che sono selezionate tramite la specifica **using**. Se non viene specificata alcuna stringa **format**, gnuplot leggerà in numero di valori binari pari alla colonna più grande data nella <**using list**>. Per esempio, **using 1:3** avrà come risultato la lettura di tre colonne, di cui la seconda sarà ignorata. Alcuni tipi di grafico hanno una specifica d'uso predefinita associata. Per esempio, **with image** ha un valore predefinito di **using 1**, mentre **with rgbimage** ha un valore predefinito di **using 1:2:3**.

Array

Descrive le dimensioni dell'array di campionamento associate al file binario. Le coordinate saranno generate da gnuplot. Un numero deve essere specificato per ogni dimensione dell'array. Per esempio, **array=(10,20)** significa che la struttura di campionamento sottostante è bidimensionale con 10 punti lungo la prima dimensione (x) e 20 punti lungo la seconda dimensione (y). Un numero negativo indica che i dati devono essere letti fino alla fine del file. Se c'è solo una dimensione, le parentesi possono essere omesse. Un due punti può essere usato per separare le dimensioni per record multipli. Per esempio, **array=25:35** indica che ci sono due record monodimensionali nel file.

Record

Questa keyword ha la stessa funzione di **array** e ha la stessa sintassi. Tuttavia, **record** fa sì che gnuplot non generi informazioni sulle coordinate. Questo è per il caso in cui tali informazioni possono essere incluse in una delle colonne del file di dati binari.

Skip

Questa keyword permette di saltare sezioni di un file binario. Per esempio, se il file contiene un'intestazione di 1024 byte prima dell'inizio della regione dei dati, probabilmente si vorrebbe usare

```
plot '<file_name>' binary skip=1024 ...
```

Se ci sono più record nel file, si può specificare un offset iniziale per ciascuno. Per esempio, per saltare 512 byte prima del primo record e 256 byte prima del secondo e del terzo record

```
plot '<file_name> binary record=356:356:356 skip=512:256:256 ...
```

Format

Il formato binario predefinito è un float. Per una maggiore flessibilità, il formato può includere dettagli sulle dimensioni delle variabili. Per esempio, **format="%uchar%int%float"** associa un carattere senza segno alla prima colonna d'uso, un int alla seconda colonna e un float alla terza colonna. Se il numero di specifiche della dimensione è inferiore al numero della colonna più grande, la dimensione è implicitamente presa per essere simile all'ultima dimensione variabile data.

Inoltre, in modo simile alla specifica **using**, il formato può includere colonne scartate tramite il carattere ***** e avere una ripetizione implicita tramite un campo di ripetizione numerica. Per esempio, **format="%*2int%3float"** fa sì che gnuplot scarti due int prima di leggere tre float. Per elencare le dimensioni delle variabili, digitare **show datafile binary datasizes**. Ci sono un gruppo di nomi che sono dipendenti dalla macchina insieme alle loro dimensioni in byte per la particolare compilazione. Esiste anche un gruppo di nomi che tentano di essere indipendenti dalla macchina.

Endian

Spesso l'endianess dei dati binari nel file non concorda con l'endianess utilizzato dalla piattaforma su cui gnuplot è in esecuzione. Diverse parole possono indicare a gnuplot come disporre i byte. Ad esempio "endian=little" significa trattare il file binario come avente un significato in byte dal minore al maggiore. Le opzioni sono

```
          little:  dal meno significativo al più significativo
             big:  dal più significativo al meno significativo
         default:  assumere che l'endianess del file sia lo stesso
                   del compilatore
    swap (swab):  Intercambia il significato.  (Se le cose
                   non sembrano giuste, prova questo.)
```

Gnuplot può supportare l'endian "middle" ("pdp") se è compilato con quell'opzione.

Filetype

Per alcuni formati di file binari standard gnuplot può estrarre tutte le informazioni necessarie dal file in questione. Per esempio, "format=edf" leggerà file di formato ESRF Header File. Per una lista dei formati di file attualmente supportati, digitare **show datafile binary filetypes**.

C'è un tipo speciale di file chiamato **auto** per il quale gnuplot controllerà se l'estensione del file binario è un'estensione quasi standard per un formato supportato.

Le keyword della linea di comando possono essere usate per sovrascrivere le impostazioni estratte dal file. Le impostazioni del file sovrascrivono qualsiasi impostazione predefinita. Vedere **set datafile binary (p. 151)**.

Avs **avs** è uno dei tipi di file binari riconosciuti automaticamente per le immagini. AVS è un formato estremamente semplice, adatto soprattutto per lo streaming tra applicazioni. Consiste di 2 lunghezze (xwidth, ywidth) seguite da un flusso di pixel, ciascuno con quattro byte di informazioni alpha/rosso/verde/blu.

Edf **edf** è uno dei tipi di file binari riconosciuti automaticamente per le immagini. EDF sta per ESRF Data Format, e supporta entrambi i formati edf e ehf (quest'ultimo significa ESRF Header Format). Maggiori informazioni sulle specifiche possono essere trovate su

```
http://www.edfplus.info/specs
```

Png Se gnuplot è stato configurato per utilizzare la libreria libgd per l'output png/gif/jpeg, allora può anche essere usato per leggere questi stessi tipi di immagini come file binari. È possibile utilizzare un comando esplicito

```
plot 'file.png' binary filetype=png
```

Oppure il tipo di file sarà riconosciuto automaticamente dall'estensione se è stato precedentemente richiesto

```
set datafile binary filetype=auto
```

Keywords

Le keyword seguenti si applicano solo quando si generano coordinate da file di dati binari. Cioè, il controllo che mappa i singoli elementi di un array binario, matrice o immagine a specifiche posizioni x/y/z.

Scan Una grande confusione può sorgere riguardo alla relazione tra come gnuplot analizza un file binario e le dimensioni viste sul grafico. Per diminuire la confusione, pensate concettualmente a gnuplot che analizza *sempre* il file binario punto/linea/piano o veloce/medio/lento. Poi questa keyword viene usata per dire a gnuplot come mappare questa convenzione di scansione alla convenzione cartesiana mostrata nei grafici, cioè, x/y/z. Il qualificatore per scan è un codice di due o tre lettere che rappresenta dove il punto è assegnato (prima lettera), la linea è assegnata (seconda lettera), e il piano è assegnato (terza lettera). Per esempio, **scan=yx** significa che l'incremento più veloce, punto per punto, dovrebbe essere mappato lungo la dimensione cartesiana y e l'incremento medio, linea per linea, dovrebbe essere mappato lungo la dimensione x.

Quando la modalità di plotting è **plot**, il codice qualificatore può includere le due lettere x e y. Per **splot**, può includere le tre lettere x, y e z.

Non c'è nulla che limiti la mappatura intrinseca da punto/linea/piano da applicare solo alle coordinate cartesiane. Per questo motivo ci sono sinonimi di coordinate cilindriche per i codici qualificatori dove t (theta), r e z sono analoghe a x, y e z delle coordinate cartesiane.

Transpose Notazione abbreviata per **scan=yx** o **scan=yxz**. Cioè influenza l'assegnazione dei pixel alle linee di analisi durante l'input. Per trasporre invece un'immagine quando viene visualizzata provare

```
plot 'imagefile' binary filetype=auto flipx rotate=90deg with rgbimage
```

Dx, dy, dz Quando gnuplot genera le coordinate, usa la spaziatura descritta da queste keyword. Per esempio **dx=10 dy=20** significherebbe spaziare campioni lungo la dimensione x di 10 e spaziare campioni lungo la dimensione y di 20. **dy** non può apparire se non appare **dx**. Allo stesso modo, **dz** non può apparire se **dy** non appare. Se le dimensioni sottostanti sono maggiori delle keyword specificate, la spaziatura della dimensione superiore indicata viene estesa alle altre dimensioni. Per esempio, se un'immagine viene letta da un file e viene dato solo **dx=3.5**, gnuplot usa un delta x e un delta y di 3.5.

Anche le seguenti keyword si applicano solo quando si generano le coordinate. Tuttavia possono essere utilizzate anche con i file di matrice binaria.

Flipx, flipy, flipz A volte le direzioni di scansione in un file di dati binari non sono coerenti con quella assunta da gnuplot. Queste keyword possono capovolgere la direzione di scansione lungo le dimensioni x, y, z.

Origin= Quando gnuplot genera le coordinate basate su trasposizione e capovolgimento (flip), gnuplot cerca sempre di posizionare il punto in basso a sinistra nell'array all'origine, cioè, i dati si trovano nel primo quadrante di un sistema cartesiano dopo la trasposizione e il capovolgimento.

Per posizionare l'array da qualche altra parte nel grafico (graph), la keyword **origin** indirizza gnuplot a posizionare il punto inferiore sinistro dell'array in un punto specificato da una tupla. La tupla dovrebbe essere un doppia per **plot** e un terna per **splot**. Per esempio, **origin=(100,100):(100,200)** è per due record nel file ed è destinato a plottare in due dimensioni. Un secondo esempio, **origin=(0,0,3.5)**, è per plottare in tre dimensioni.

Center Simile a **origin**, questa keyword posizionerà l'array in modo tale che il suo centro si trovi nel punto dato dalla tupla. Per esempio, **center=(0,0)**. Il centro non si applica quando la dimensione dell'array è **Inf**.

Rotate I comandi transpose e flip forniscono una certa flessibilità nel generare e orientare le coordinate. Tuttavia, per i gradi di libertà completi, è possibile applicare un vettore di rotazione descritto da un angolo di rotazione in due dimensioni.

La keyword **rotate** si applica al piano bidimensionale, sia che si tratti di **plot** o **splot**. La rotazione è fatta rispetto all'angolo positivo del piano cartesiano.

L'angolo può essere espresso in radianti, radianti come multiplo di pi, o gradi. Per esempio, **rotate=1.5708**, **rotate=0.5pi** e **rotate=90deg** sono equivalenti.

Se **origin** è specificato, la rotazione è fatta intorno al punto di campionamento in basso a sinistra prima della traslazione. Altrimenti, la rotazione viene fatta intorno all'array **center**.

Perpendicular Per **splot**, il concetto di vettore di rotazione è implementato da una terna che rappresenta il vettore da orientare normalmente al piano bidimensionale x-y. Naturalmente, il valore predefinito è (0,0,1). Quindi specificando sia rotate che perpendicular, insieme possono orientare i dati in una miriade di modi nel trispazio.

La rotazione bidimensionale viene fatta per prima, seguita dalla rotazione tridimensionale. Cioè, se R' è la matrice rotazionale 2 x 2 descritta da un angolo e P è la matrice 3 x 3 che proietta (0,0,1) in (xp,yp,zp), si lasci che R venga costruita da R' nella sottomatrice superiore sinistra, 1 nell'elemento 3,3 e zeri altrove. Allora la formula della matrice per la traslazione dei dati è v' = P R v, dove v è il vettore 3 x 1 dei dati estratti dal file di dati. Nei casi in cui i dati del file non sono di per sé tridimensionali, si usano regole logiche per collocare i dati nel trispazio. (Ad es., di solito impostando il valore della dimensione z a zero e mettendo i dati 2D nel piano x-y.)

Data

I dati discreti contenuti in un file possono essere visualizzati specificando il nome del file di dati (racchiuso tra virgolette singole o doppie) sulla linea di comando **plot**.

Sintassi:

```
plot '<file_name>' {binary <binary list>}
                   {{nonuniform} matrix}
                   {index <index list> | index "<name>"}
                   {every <every list>}
                   {skip <number-of-lines>}
                   {using <using list>}
                   {smooth <option>}
                   {bins <options>}
                   {volatile} {noautoscale}
```

I modificatori **binary**, **index**, **every**, **skip**, **using**, **bins**, e **smooth** vengono discussi separatamente. In breve

- **skip N** dice al programma di ignorare N linee all'inizio del file di input

- **binary** indica che il file contiene dati binari piuttosto che testo
- **index** seleziona quali serie di dati in un file con più serie di dati devono essere plottati
- **every** specifica quali punti all'interno di una singola serie di dati devono essere plottati
- **using** specifica quali colonne nel file devono essere usate e in quale ordine
- **smooth** esegue un semplice filtraggio, interpolazione o curve-fitting dei dati prima di plottare
- **bins** ordina i singoli punti di input in intervalli di uguali dimensioni lungo x e plotta un singolo valore accumulato per intervallo
- **volatile** indica che il contenuto del file potrebbe non essere disponibile per essere riletto in seguito e quindi dovrebbe essere conservato internamente per essere riutilizzato.

splot ha una sintassi simile ma non supporta **smooth** o **bins**.

La keyword **noautoscale** significa che i punti che compongono questo grafico saranno ignorati quando si determinano automaticamente i limiti dell'intervallo degli assi.

TEXT DATA FILES:

Ogni linea non vuota in un file di dati descrive un data point, eccetto che i record che iniziano con # (e anche con **!** su VMS) saranno trattati come commenti e ignorati.

A seconda dello stile di grafico e delle opzioni selezionate, da uno a otto valori vengono letti da ogni linea e associati a un singolo data point. Vedere **using (p. 121)**.

I record individuali su una singola linea di dati devono essere separati da spazio bianco (uno o più spazi vuoti o tabs) uno speciale carattere separatore di campo è specificato dal comando **set datafile**. Un singolo campo può esso stesso contenere caratteri di spazio bianco se l'intero campo è racchiuso tra una coppia di virgolete doppie, o se è presente un separatore di campo diverso dallo spazio bianco. Gli spazi bianchi all'interno di una coppia di virgolette doppie sono ignorati quando si contano le colonne, quindi la seguente linea di datafile ha tre colonne:

```
1.0 "second column" 3.0
```

I dati possono essere scritti in formato esponenziale con l'esponente preceduto dalla lettera e o E. Gli specificatori esponenziali fortran d, D, q, e Q possono anche essere usati se il comando **set datafile fortran** è attivo.

I record vuoti in un file di dati sono significativi. Singoli record vuoti designano discontinuità in un **plot**; nessuna linea unirà punti separati da un record vuoto (se sono plottati con un line style). Due record vuoti in una riga indicano un'interruzione tra insiemi di dati separati. Vedere **index (p. 116)**.

Se l'autoscaling è stato abilitato (**set autoscale**), gli assi sono automaticamente estesi per includere tutti i data point, con un numero intero di segni dei tic se i tic vengono disegnati. Questo ha due conseguenze: i) Per **splot**, l'angolo della superficie può non coincidere con l'angolo della base. In questo caso, nessuna linea verticale viene tracciata. ii) Quando si plottano dati con lo stesso intervallo di x su un grafico (graph) a due assi, le coordinate x potrebbero non coincidere se i x2tics non stanno venendo disegnati. Questo perché l'asse x è stato esteso automaticamente a un numero numero di tic, ma l'asse x2 no. L'esempio seguente dimostra il problema:

```
reset; plot '-', '-' axes x2y1
1 1
19 19
e
1 1
19 19
e
```

Per evitarlo, si può usare il modificatore **noextend** dei comandi **set autoscale** o **set [axis]range**. Questo disattiva l'estensione dell'intervallo degli assi per includere il prossimo segno di tic.

Le coordinate delle etichette e il testo possono anche essere letti da un data file (vedere **labels (p. 79)**).

Bins

Sintassi:

```
plot 'DATA' using <XCOL> {:<YCOL>} bins{=<NBINS>}
     {binrange [<LOW>:<HIGH>]} {binwidth=<width>}
     {binvalue={sum|avg}
```

L'opzione **bins** del comando **plot** assegna prima i dati originali a dei bin di uguale larghezza su x e poi plotta un singolo valore per bin. Il numero predefinito di bin è controllato da **set samples**, ma questo può essere cambiato dando un numero esplicito di bin nel comando.

Se non viene dato un binrange, l'intervallo viene preso dagli estremi dei valori x trovati in 'DATA'.

Dato l'intervallo e il numero di bin, la larghezza del bin (bin width) è calcolata automaticamente e i punti vengono assegnati ai bin da 0 a NBINS-1

```
BINWIDTH = (HIGH - LOW) / (NBINS-1)
xmin = LOW - BINWIDTH/2
xmax = HIGH + BINWIDTH/2
il primo bin contiene punti con (xmin <= x < xmin + BINWIDTH)
l'ulyimo bin contine punti con (xmax-BINWIDTH <= x < xman)
ogni punto è assegnato a bin i = floor(NBINS * (x-xmin)/(xmax-xmin))
```

In alternativa è possibile fornire una larghezza dei bin fissa, nel qual caso nbins è calcolato come il numero più piccolo di bin che coprirà l'intervallo.

In output i bins sono plottati o tabulati per punto medio. Ad esempio, se il programma calcola la larghezza dei bin come mostrato sopra, la coordinata x in output per il primo bin è x=LOW (non x=xmin).

Se viene data una sola colonna nella clausola using, allora ogni data point contribuisce con un conteggio di 1 all'accumulo dei conteggi totali nel bin per quel valore di coordinata x. Se viene data una seconda colonna, allora il valore in quella colonna viene aggiunto all'accumulo per il bin. Così i due seguenti comandi plot sono equivalenti:

```
plot 'DATA" using N bins=20
set samples 20
plot 'DATA' using (column(N)):(1)
```

Il valore y plottato per ciascun bin è la somma dei valori y su tutti i punti in quel bin. Questo corrisponde a **binvalue=sum**. SPERIMENTALE: **binvalue=avg** invece plotta il valore medio y per quel bin.

Per gli stili di plotting correlati vedere **smooth frequency (p. 119)** e **smooth kdensity (p. 119)**.

Columnheaders

Le linee extra all'inizio di un file di dati possono essere esplicitamente ignorate usando la keyword **skip** nel comando plot. Può essere presente una singola linea aggiuntiva contenente le intestazioni della colonna di testo. Viene saltata automaticamente se il comando plot si riferisce esplicitamente alle intestazioni di colonna, ad esempio usandole per i titoli. Altrimenti potrebbe essere necessario saltarlo esplicitamente o aggiungendo uno allo skip count o impostando l'attributo **set datafile columnheaders**. Vedere **skip (p. 117)**, **columnhead (p. 40)**, **autotitle columnheader (p. 168)**, **set datafile (p. 148)**.

Csv files

Sintassi:

```
set datafile separator {whitespace | tab | comma | "chars"}
```

"csv" è l'abbreviazione di "comma-separated values" (valori separati da virgola). Il termine "file csv" è genericamente applicato ai file in cui i campi di dati sono delimitati da un carattere specifico, non necessariamente una virgola. Per leggere dati da un file csv è necessario dire a gnuplot qual è il carattere di

delimitazione del campo. Per esempio per leggere da un file usando il punto e virgola come delimitatore di campo:

```
set datafile separator ";"
```

Vedere **set datafile separator (p. 150)**. Questo si applica solo ai file usati per l'input. Per creare un file csv in output, bisogna usare la corrispondente opzione **separator** in **set table**.

Every

La keyword **every** permette di plottare un campionamento periodico di un insieme di dati.

Per i file ordinari un "point" single record (linea); un "block" di dati è un insieme di record consecutivi con linee vuote prima e dopo il blocco.

Per i dati della matrice un "block" e un "point" corrispondono a "riga" e "colonna". Vedere **matrix every (p. 240)**.

Sintassi:

```
plot 'file' every {<point_incr>}
                  {:{<block_incr>}
                    {:{<start_point>}
                      {:{<start_block>}
                        {:{<end_point>}
                          {:<end_block>}}}}}
```

I data point da plottare sono selezionati secondo un loop da <**start_point**> a <**end_point**> con incremento <**point_incr**> e i blocchi secondo un loop da <**start_block**> a <**end_block**> con incremento <**block_incr**>.

Il primo dato di ogni blocco è numerato '0', così come il primo blocco del file.

Si noti che i record che contengono informazioni non plottabili sono contate.

Uno qualsiasi dei numeri può essere omesso; gli incrementi sono predefiniti all'unità, i valori iniziali al primo punto o blocco, e i valori finali all'ultimo punto o blocco. Non è permesso ':' alla fine dell'opzione **every**. Se **every** non è specificato, tutti i punti in tutte le linee vengono plottati.

Esempi:

```
every :::3::3    # seleziona solo il quarto blocco ('0' è il primo)
every :::::9     # seleziona i primi 10 blocchi
every 2:2        # seleziona ogni altro punto in ogni altro blocco
every ::5::15    # seleziona i punti da 5 a 15 in ogni blocco
```

Vedere demo di plot semplice (simple.dem)

, Demo di splot non parametrici

, e Demo di splot parametrici

.

Esempio datafile

Questo esempio plotta i dati nel file "population.dat" e una curva teorica:

```
pop(x) = 103*exp((1965-x)/10)
set xrange [1960:1990]
plot 'population.dat', pop(x)
```

Il file "population.dat" potrebbe contenere:

```
# Popolazione di gnu in Antartide dal 1965
   1965   103
   1970   55
   1975   34
   1980   24
   1985   10
```

Esempi binari:

```
# Seleziona due valori float (secondo implicito) con un valore float
# scartato tra di loro per una lunghezza indefinita di dati 1D.
plot '<file_name>' binary format="%float%*float" using 1:2 with lines

# Il header del file di dati contiene tutti i dettagli necessari per
# creare le coordinate da un file EDF.
plot '<file_name>' binary filetype=edf with image
plot '<file_name>.edf' binary filetype=auto with image

# Seleziona tre caratteri senza segno per i componenti di un'immagine
# RGB grezza e capovolge la dimensione y in modo che l'orientamento
# tipico dell'immagine (inizia dall'angolo in alto a sinistra) si
# si traduca nel piano cartesiano. Viene data la spaziatura dei pixel
# e nel file sono presenti due immagini. Una di esse è tradotta tramite
# origine.
plot '<file_name>' binary array=(512,1024):(1024,512) format='%uchar' \
     dx=2:1 dy=1:2 origin=(0,0):(1024,1024) flipy u 1:2:3 w rgbimage

# Quattro record separati in cui le coordinate fanno parte del file di dati.
# Il file è stato creato con un endianess diverso dal sistema su cui
# gnuplot è in esecuzione.
splot '<file_name>' binary record=30:30:29:26 endian=swap u 1:2:3

# Stesso file di input, ma questa volta saltiamo il 1° e il 3° record
splot '<file_name>' binary record=30:26 skip=360:348 endian=swap u 1:2:3
```

Vedere anche **binary matrix (p. 238)**.

Index

La keyword **index** permette di selezionare specifici set di dati in un file multi-data-set per il plotting.

Sintassi:

```
plot 'file' index { <m>{:<n>{:<p>}} | "<name>" }
```

I set di dati sono separati da coppie di record vuoti. **index** <**m**> seleziona solo set <m>; **index** <**m**>**:**<**n**> seleziona i set nell'intervallo da <m> a <n>; e **index** <**m**>**:**<**n**>**:**<**p**> seleziona gli indici <m>, <m>+<p>, <m>+2<p>, etc., ma si ferma a <n>. In seguito all'indicizzazione C, l'indice 0 è assegnato al primo set di dati nel file. Specificando un indice troppo grande si ottiene un messaggio di errore. Se <p> è specificato ma <n> è lasciato vuoto, allora ogni <p>-esimo set di dati viene letto fino alla fine del file. Se **index** non è specificato, l'intero file viene plottato come una singola serie di dati.

Esempio:

```
plot 'file' index 4:5
```

Per ogni punto nel file, il valore dell'indice dell'insieme di dati in cui appare è disponibile tramite la pseudo-colonna **column(-2)**. Questo porta ad un modo alternativo di distinguere i singoli set di dati all'interno di un file, come mostrato di seguito. Questo è più scomodo del comando **index**, se tutto quello che si sta facendo è selezionare un insieme di dati da plottare, ma è molto utile se si vogliono assegnare diverse proprietà a ciascun insieme di dati. Vedere **pseudocolumns (p. 123)**, **lc variable (p. 52)**.

Esempio:

```
plot 'file' using 1:(column(-2)==4 ? $2 : NaN)        # molto scomodo
plot 'file' using 1:2:(column(-2)) linecolor variable # molto utile!
```

index '<name>' seleziona l'insieme di dati con nome '<name>'. I nomi sono assegnati agli insiemi di dati nelle linee di commento. Il carattere di commento e lo spazio bianco iniziale vengono rimossi dalla linea di commento. Se la linea risultante inizia con <name>, il seguente set di dati ora si chiama <name> e può essere selezionato.

Esempio:

```
plot 'file' index 'Population'
```

Si noti che ogni commento che inizia con <name> darà il nome al successivo insieme di dati. Per evitare problemi può essere utile scegliere uno schema di denominazione come '== Population ==' o '[Population]'.

Skip

La keyword **skip** dice al programma di saltare le linee all'inizio di un file di dati di testo (cioè non binario). Le linee che vengono saltate non sono contate nel conteggio delle linee usate per elaborare la keyword **every**. Si noti che **skip N** salta le linee solo all'inizio del file, mentre **every ::N** salta le linee all'inizio di ogni blocco di dati nel file. Vedere anche **binary skip (p. 109)** per un'opzione simile che si applica ai file di dati binari.

Smooth

gnuplot include alcune routine di uso generale per filtrare, interpolare e raggruppare i dati come in input; questi sono raggruppati sotto l'opzione **smooth**. Un'elaborazione dei dati più sofisticata può essere eseguita preprocessando i dati esternamente o utilizzando **fit** con un modello appropriato.

Sintassi:

```
smooth {unique | frequency | fnormal | cumulative | cnormal | bins
               | kdensity {bandwidth} {period}
               | csplines | acsplines | mcsplines | bezier | sbezier
               | unwrap | zsort}
```

unique, **frequency**, **fnormal**, **cumulative** e **cnormal** ordinano i dati su x e poi plottano qualche aspetto della distribuzione dei valori di x.

Le opzioni spline e Bezier determinano i coefficienti che descrivono una curva continua tra gli endpoint dei dati. Questa curva viene poi plottata nello stesso modo di una funzione, cioè trovando il suo valore a intervalli uniformi lungo l'ascissa (vedere **set samples (p. 204)**) e collegando questi punti con segmenti di linea retta. Se la serie di dati è interrotta da linee vuote o da valori indefiniti, viene adattata una curva continua separata per ogni sottoinsieme ininterrotto dei dati. Segmenti adiacenti adattati separatamente possono essere separati da uno spazio o da una discontinuità.

unwrap manipola i dati per evitare salti più grandi di pi greco aggiungendo o sottraendo multipli di 2*pi.

zsort utilizza una terza colonna di input per ordinare i punti prima di plottare.

Se **autoscale** è in funzione, gli intervalli degli assi saranno computati per la curva finale piuttosto che per i dati originali.

Se **autoscale** non è in funzione, e viene generata una curva spline, il campionamento dell'adattamento della spline è fatto attraverso l'intersezione dell'intervallo x coperto dai dati di input e l'intervallo di ascissa fisso definito da **set xrange**.

Se sono disponibili troppi pochi punti per applicare l'operazione smoothing richiesta viene prodotto un messaggio di errore.

Le opzioni **smooth** non hanno effetto sui grafici delle funzioni.

Acsplines L'opzione **smooth acsplines** approssima i dati con una spline di smoothing naturale. Dopo aver reso i dati monotonici in x (vedere **smooth unique (p. 118)**), una curva viene costruita in modo frammentario da segmenti di polinomi cubici i cui coefficienti vengono trovati adattandoli ai singoli punti dei dati ponderati per il valore, se presente, dato nella terza colonna della specifica di utilizzo. L'impostazione predefinita è equivalente a

```
plot 'data-file' using 1:2:(1.0) smooth acsplines
```

Qualitativamente, la grandezza assoluta dei pesi determina il numero di segmenti utilizzati per costruire la curva. Se i pesi sono grandi, l'effetto effetto di ogni dato è grande e la curva si avvicina a quella prodotta collegando punti consecutivi con spline cubiche naturali. Se i pesi sono piccoli, la curva è composta da meno segmenti e quindi è più uniforme; il caso limite è il singolo segmento prodotto da un adattamento lineare dei minimi quadrati ponderati a tutti i dati. Il peso di smoothing può essere espresso in termini di errori come un peso statistico per un punto diviso per un "smoothing factor" (fattore di uniformità) per la curva in modo che gli errori (standard) nel file possano essere usati come pesi di smoothing.

Esempio:

```
sw(x,S)=1/(x*x*S)
plot 'data_file' using 1:2:(sw($3,100)) smooth acsplines
```

Bezier L'opzione **smooth bezier** approssima i dati con una curva di Bezier di grado n (il numero di data point) che collega i punti finali.

Bins **smooth bins** è uguale a **bins**. Vedere **bins (p. 114)**. Per gli stili di plotting correlati vedere **smooth frequency (p. 119)** e **smooth kdensity (p. 119)**.

Csplines L'opzione **smooth csplines** collega punti consecutivi con spline cubiche naturali dopo aver reso i dati monotonici (vedere **smooth unique (p. 118)**).

Mcsplines L'opzione **smooth mcsplines** collega punti consecutivi tramite spline cubiche vincolate in modo tale che la funzione regolata conservi la monotonicità e la convessità dei data point originali. Questo riduce l'effetto degli outlier. FN Fritsch & RE Carlson (1980) "Monotone Piecewise Cubic Interpolation", SIAM Journal on Numerical Analysis 17: 238–246.

Sbezier L'opzione **smooth sbezier** prima rende i dati monotonici (**unique**) e poi applica l'algoritmo **bezier**.

Unique L'opzione **smooth unique** rende i dati monotonici in x; i punti con lo stesso valore di x sono sostituiti da un unico punto che ha il valore medio di y. I punti risultanti sono poi collegati da segmenti di linea retta.

Unwrap L'opzione **smooth unwrap** modifica i dati di input in modo che due punti successivi non differiscano di più di pi greco; un punto il cui valore y è al di fuori di questo intervallo sarà incrementato o decrementato di multipli di 2pi fino a che non rientri entro pi del punto precedente. Questa operazione è utile per rendere le misure di fase avvolte continue nel tempo.

Frequency L'opzione **smooth frequency** rende i dati monotoni in x; i punti con lo stesso valore di x sono sostituiti da un unico punto avente i valori di y sommati. Per plottare un istogramma del numero di valori di dati in bin di uguali dimensioni, impostare il valore y a 1.0 in modo che la somma sia un conteggio delle occorrenze in quel bin. Questo viene fatto implicitamente se viene fornita una sola colonna. Esempio:

```
binwidth = <something>  # imposta larghezza di x valori in ogni bin
bin(val) = binwidth * floor(val/binwidth)
plot "datafile" using (bin(column(1))):(1.0) smooth frequency
plot "datafile" using (bin(column(1))) smooth frequency  # stesso risultato
```

Vedere anche smooth.dem

Fnormal L'opzione **smooth fnormal** funziona proprio come l'opzione **frequency**, ma produce un istogramma normalizzato. Rende i dati monotonici in x e normalizza i valori y in modo che sommino tutti a 1. I punti con lo stesso valore x sono sostituiti da un singolo punto contenente i valori y sommati. Per plottare un istogramma del numero di valori di dati in intervalli di uguale dimensione, impostare il valore y a 1.0 in modo che la somma sia un conteggio delle occorrenze in quel bin. Questo viene fatto implicitamente se viene fornita una sola colonna. Vedere anche smooth.dem

Cumulative L'opzione **smooth cumulative** rende i dati monotonici in x; i punti con lo stesso valore di x sono sostituiti da un singolo punto contenente la somma cumulativa dei valori y di tutti i data point con valori x inferiori (cioè a sinistra del data point corrente). Questo può essere usato per ottenere una funzione di distribuzione cumulativa dai dati. Vedere anche smooth.dem

Cnormal L'opzione **smooth cnormal** rende i dati monotoni in x e normalizza i valori y sull'intervallo [0:1]. I punti con lo stesso valore di x sono sostituiti da un unico punto contenente la somma cumulativa dei valori y di tutti i data point con valori x inferiori (cioè a sinistra del data point corrente) diviso per la somma totale di tutti i valori y. Questo può essere usato per ottenere una funzione di distribuzione cumulativa normalizzata dai dati (utile quando si confrontano insiemi di campioni con un diverso numero di membri). Vedere anche smooth.dem

Kdensity L'opzione **smooth kdensity** genera e plotta una stima della densità del kernel utilizzando kernel gaussiani per la distribuzione da cui è stato estratto un insieme di valori. I valori sono presi dalla prima colonna di dati, i pesi opzionali sono presi dalla seconda colonna. Una gaussiana è posta nella posizione di ogni punto punto e la somma di tutte queste gaussiane viene plottata come funzione. Per ottenere un istogramma normalizzato, ogni peso dovrebbe essere 1/numero di punti.

Larghezza di banda: Per impostazione predefinita gnuplot calcola e utilizza la larghezza di banda che sarebbe ottimale per valori di dati normalmente distribuiti.

```
default_bandwidth = sigma * (4/3N) ** (0.2)
```

Questo di solito sarà una larghezza di banda molto conservativa, cioè ampia. In alternativa, è possibile fornire una larghezza di banda esplicita.

```
plot $DATA smooth kdensity bandwidth <value> with boxes
```

La larghezza di banda utilizzata nel grafico precedente è memorizzata in GPVAL_KDENSITY_BANDWIDTH.

Periodo: Per i dati periodici le singole componenti gaussiane dovrebbero essere trattate come se si ripetessero a intervalli di un periodo. Un esempio sono i dati misurati come funzione di angolo, dove il periodo è 2pi. Un altro esempio sono i dati indicizzati per giorno dell'anno e misurati su più anni, dove il periodo è 365. In questi casi il periodo dovrebbe essere fornito nel comando plot:

```
plot $ANGULAR_DAT smooth kdensity period 2*pi with lines
```

Zsort Sintassi

```
plot FOO using x:y:z:color smooth zsort with points lc palette
```

L'uso previsto è quello di filtrare la presentazione di grafici 2D di dispersione con un enorme numero di punti in modo che la distribuzione dei punti ad alto punteggio rimanga evidente. Ordinare i punti in base a z garantisce che i punti con un alto valore z non saranno oscurati da punti con valori z inferiori. Limitato allo stile di grafico "with points".

Special-filenames

Ci sono alcuni filename che hanno un significato speciale: '', '-', '+' e '++'.

Il filename vuoto '' dice a gnuplot di riutilizzare il file di input precedente nello stesso comando plot. Quindi per plottare due colonne dallo stesso file di input:

```
plot 'filename' using 1:2, '' using 1:3
```

Il filename può anche essere riutilizzato nei successivi comandi plot, tuttavia **save** registrerà solo il nome in un commento.

I filename speciali '+' e'++' sono un meccanismo per consentire l'intera gamma di specificatori 'using' e stili di grafico con funzioni in linea. Normalmente un grafico di funzione può avere solo un singolo valore y (o z) associato a ciascun punto campionato. Il pseudo-file '+' tratta i punti campionati come colonna 1 e permette di specificare ulteriori valori di colonna tramite una specifica **using**, proprio come per un vero file di input. Il numero di campioni è controllato tramite **set samples**. Per default i campioni sono generati nell'intervallo dato da **set trange**, o se trange non è stato impostato, allora su tutto l'intervallo dato da **set xrange**.

Nota: L'uso di trange è un cambiamento rispetto alle precedenti versioni di gnuplot. Esso permette all'intervallo di campionamento di differire dall'intervallo dell'asse x.

```
plot '+' using ($1):(sin($1)):(sin($1)**2) with filledcurves
```

Un intervallo di campionamento indipendente può essere dato subito prima di '+'. Come nei normali grafici di funzione, può essere assegnato un nome alla variabile indipendente. Se viene dato per il primo elemento del grafico, lo specificatore dell'intervallo di campionamento deve essere preceduto dalla keyword **sample** (vedere anche **plot sampling (p. 127)**).

```
plot sample [beta=0:2*pi] '+' using (sin(beta)):(cos(beta)) with lines
```

Inoltre, lo specificatore di intervallo '+' supporta il conferimento di un incremento di campionamento.

```
plot $MYDATA, [t=-3:25:1] '+' using (t):(f(t))
```

Il pseudo-file '++' restituisce 2 colonne di dati che formano una griglia regolare di coordinate [u,v] con il numero di punti lungo u controllato da **set samples** e il numero di punti lungo v controllato da **set isosamples**. È necessario impostare urange e vrange prima di plottare '++'. Tuttavia gli intervalli x e y possono essere autoscalati o possono essere esplicitamente impostati su valori diversi da urange e vrange. L'uso di u e v per campionare '++' è un CAMBIAMENTO introdotto nella versione 5.2 Esempi:

```
splot '++' using 1:2:(sin($1)*sin($2)) with pm3d
plot '++' using 1:2:(sin($1)*sin($2)) with image
```

Il filename speciale **'-'** specifica che i dati sono in linea; cioè, seguono il comando. Solo i dati seguono il comando; le opzioni **plot** come filtri, titoli e stili di linea rimangono sulla linea di comando **plot**. Questo è simile a << in uno script di shell unix e a $DECK in VMS DCL. I dati sono inseriti come se fossero letti da un file, un data point per record. La lettera "e" all'inizio della prima colonna termina l'inserimento dei dati.

'-' è destinato a situazioni in cui è utile avere dati e comandi insieme, per esempio quando entrambi sono inviati a **gnuplot** da un'altra applicazione. Alcuni demo, per esempio, potrebbero usare questa caratteristica.

Mentre opzioni **plot** come **index** e **every** sono riconosciute, il loro uso costringe a inserire dati che non saranno utilizzati. Per tutti i casi, tranne i più semplici, è probabilmente più facile definire prima un datablock e poi leggere da esso piuttosto che da **'-'**. Vedere **datablocks (p. 48)**.

Se si usa **'-'** con **replot**, potrebbe essere necessario inserire i dati più di una volta. Vedere **replot (p. 134)**, **refresh (p. 134)**. Anche in questo caso sarebbe meglio usare un datablock.

Un filename vuoto ('') specifica che il filename precedente dovrebbe essere riutilizzato. Questo può essere utile con cose come

```
plot 'a/very/long/filename' using 1:2, '' using 1:3, '' using 1:4
```

(Se si usano sia **'-'** che **''** sullo stesso comando **plot**, è necessario avere due serie di dati in linea, come nell'esempio sopra.)

Piped-data

Sui sistemi con una funzione popen, il file di dati può essere convogliato attraverso un comando shell facendo iniziare il nome del file con un '<'. Per esempio,

```
pop(x) = 103*exp(-x/10)
plot "< awk '{print $1-1965, $2}' population.dat", pop(x)
```

plotterebbe le stesse informazioni come il primo esempio di popolazione ma con anni dal 1965 come asse x. Se si desidera eseguire questo esempio, è necessario cancellare tutti i commenti dal file di dati riportato sopra o sostituire il comando seguente per la prima parte del comando precedente (la parte fino alla virgola):

```
plot "< awk '$0 !~ /^#/ {print $1-1965, $2}' population.dat"
```

Sebbene questo approccio è il più flessibile, è possibile ottenere un semplice filtraggio con la keyword **using**.

Sui sistemi con una funzione fdopen(), i dati possono essere letti da un descrittore di file arbitrario collegato a un file o a una pipe. Per leggere dal descrittore di file **n** utilizzare **'<&n'**. Questo permette di inserire facilmente diversi datafile in una pipe con una singola chiamata da una shell POSIX:

```
$ gnuplot -p -e "plot '<&3', '<&4'" 3<data-3 4<data-4
$ ./gnuplot 5< <(myprogram -with -options)
gnuplot> plot '<&5'
```

Using

Il modificatore di file di dati più comune è **using**. Dice al programma quali colonne di dati nel file di input devono essere plottate.

Sintassi:

```
plot 'file' using <entry> {:<entry> {:<entry> ...}} {'format'}
```

Se viene specificato un formato, esso viene usato per leggere in ogni record di file di dati usando la funzione 'scanf' della libreria C. Altrimenti il record viene interpretato come costituito da colonne (campi) di dati separate da spazi bianchi (spazi e/o tab), ma si consiglia di vedere **datafile separator (p. 150)**.

Ogni <entry> può essere un semplice numero di colonna che seleziona il valore da un campo del file di input, da una stringa che corrisponde a un'etichetta di colonna nella prima linea di un set di dati, da un'espressione racchiusa tra parentesi o da una funzione speciale non racchiusa tra parentesi come xticlabels(2).

Se l'entry è un'espressione tra parentesi, allora la funzione column(N) può essere usata per indicare il valore nella colonna N. Cioè, column(1) si riferisce al primo elemento letto, column(2) al secondo, e così via. I simboli speciali $1, $2, ... sono abbreviazioni per column(1), column(2) ...

Il simbolo speciale $# valuta il numero totale di colonne nella linea di input corrente, così column($#) o stringcolumn($#) restituisce sempre il contenuto della colonna finale anche se il numero di colonne è sconosciuto o se diverse linee del file contengono diversi numeri di colonne.

La funzione **valid(N)** verifica se la colonna N contiene un numero valido. Se ogni colonna di dati nel file di input contiene un'etichetta nella prima riga piuttosto che un valore di dati, questa etichetta può essere usata per identificare la colonna in input e/o nella legenda del grafico. La funzione column() può essere usata per selezionare una colonna di input tramite etichetta piuttosto che per il numero di colonna. Per esempio,se il file di dati contiene

```
Height    Weight    Age
val1      val1      val1
...       ...       ...
```

allora i comandi plot seguenti sono tutti equivalenti

```
plot 'datafile' using 3:1, '' using 3:2
plot 'datafile' using (column("Age")):(column(1)), \
             '' using (column("Age")):(column(2))
plot 'datafile' using "Age":"Height", '' using "Age":"Weight"
```

La stringa completa deve corrispondere. Il confronto è sensibile alle maiuscole e alle minuscole. Per utilizzare le etichette delle colonne nella legenda del grafico, usare **set key autotitle columnhead**.

Oltre alle effettive colonne 1...N nel file di dati di input, gnuplot presenta i dati da diverse "pseudo-colonne" che contengono informazioni contabili. Ad esempio $0 o column(0) restituisce il numero d'ordine di questo record di dati all'interno di un dataset. Si prega di vedere **pseudocolumns (p. 123)**.

Una <entry> vuota tornerà automaticamente alla sua posizione nella lista delle entry. Per esempio, **using ::4** è interpretato come **using 1:2:4**.

Se la lista **using** ha una sola entry, quella <entry> sarà usata per y e il numero del data point (pseudo-colonna $0) sarà usato per x; per esempio, **"plot 'file' using 1"** è identico a **"plot 'file' using 0:1"**. Se la lista **using** ha due entry, queste saranno usate per x e y. Vedere **set style (p. 206)** e **fit (p. 95)** per i dettagli sugli stili di plotting che fanno uso di dati da colonne aggiuntive di input.

'scanf' accetta diverse specifiche numeriche, ma **gnuplot** richiede che tutti gli input siano variabili in virgola mobile a doppia precisione, quindi "%lf" è essenzialmente l'unica specifica ammissibile. Una stringa di formato data dall'utente deve contenere almeno uno di questi specificatori di input, e non più di sette. 'scanf' si aspetta di vedere uno spazio bianco - uno spazio vuoto, un tab ("t"), newline ("n"), o formfeed ("f")–tra i numeri; qualsiasi altra cosa nel flusso di input deve essere esplicitamente saltata.

Si noti che l'uso di "\t", "\n", o "\f" richiede l'uso di virgolette doppie piuttosto che virgolette singole.

Using_examples Questo crea un grafico della somma del 2° e 3° dato contro il primo: La stringa di formato specifica colonne separate da virgole piuttosto che da spazi. Lo stesso risultato potrebbe essere ottenuto specificando **set datafile separator comma**.

```
plot 'file' using 1:($2+$3) '%lf,%lf,%lf'
```

In questo esempio i dati sono letti dal file "MyData" usando un formato più complicato:

```
plot 'MyData' using "%*lf%lf%*20[^\n]%lf"
```

Il significato di questo formato è:

```
%*lf        ignora un numero
%lf         leggi un numero a doppia precisione (x di default)
%*20[^\n]   ignora 20 caratteri non newline
%lf         leggi un numero a doppia precisione (y di default)
```

Un trucco è usare l'operatore ternario **?:** per filtrare i dati:

```
plot 'file' using 1:($3>10 ? $2 : 1/0)
```

che plotta il dato nella colonna due contro quello della colonna uno, a condizione che il dato nella colonna tre sia superiore a dieci. **1/0** è indefinito; **gnuplot** ignora i punti non definiti, quindi i punti non adatti vengono soppressi. Oppure si può usare la variabile predefinita NaN per ottenere lo stesso risultato.

Infatti, è possibile utilizzare un'espressione costante per il numero di colonna, a condizione che non inizi con una parentesi aperta; si possono usare costrutti come **using 0+(complicated expression)**. Il punto cruciale è che l'espressione viene valutata una volta se non inizia con una parentesi sinistra, o una volta per ogni data point letto se lo fa.

Se si stanno usando dati di serie temporali, il tempo può estendersi su più colonne. La colonna iniziale dovrebbe essere specificata. Si noti che gli spazi all'interno del periodo devono essere inclusi quando si calcolano le colonne iniziali per altri dati. Ad es., se il primo elemento su una linea è un tempo con uno spazio incorporato, il valore y dovrebbe essere specificato come colonna tre.

Va notato che (a) **plot 'file'**, (b) **plot 'file' using 1:2**, e (c) **plot 'file' using ($1):($2)** possono essere leggermente diversi. Il comportamento specifico è stato cambiato nella versione 5. Vedere **missing (p. 149)**.

Spesso è possibile plottare un file con molte linee di informazioni inutili in testa semplicemente specificando

```
plot 'file' using 1:2
```

Tuttavia, se si vuole lasciare del testo nei file di dati, è più sicuro mettere il carattere di commento (#) nella prima colonna delle linee di testo.

Pseudocolumns Le espressioni nella clausola "using" di una dichiarazione del plot possono riferirsi a valori contabili aggiuntivi oltre ai valori effettivi dei dati contenuti nel file di input. Questi sono contenuti in "pseudocolonne".

```
column(0)   L'ordine sequenziale di ogni punto all'interno di un set di dati.
            Il contatore parte da 0, aumenta ad ogni linea non vuota,
            linea non di commento, e viene azzerato da due record
            sequenziali vuoti. È disponibile la forma abbreviata $0.
column(-1)  Questo contatore parte da 0, incrementa su una singola linea
           vuota e viene azzerato da due linee vuote sequenziali.
            Questo corrisponde alla linea di dati nell'array o nei dati
            della griglia. Può anche essere usato per distinguere segmenti
            di linea separati o poligoni all'interno di un set di dati.
column(-2)  Inizia da 0 e incrementa su due linee vuote sequenziali.
            Questo è il numero di indice del set di dati attuale
            all'interno di un file che contiene più set di dati.
            Vedere 'index'.
column($#)  Il simbolo speciale $# valuta il numero totale di
            colonne disponibili, quindi column($#) si riferisce
            all'ultimo (più a destra) campo nella linea di input
            corrente. column($# - 1) si riferisce alla penultima
            colonna, ecc.
```

Key Il layout di alcuni stili di grafico (istogrammi a colonne impilate, spider plot) è tale che non avrebbe senso generare titoli plot da un'intestazione di colonna di dati. Inoltre non avrebbe senso generare le etichette dei tic degli assi dal contenuto di una colonna di dati (ad es., **using 2:3:xticlabels(1)**). Questi stili di grafici invece usano la forma **using 2:3:key(1)** per generare titoli plot per la key dal contenuto testuale di una colonna di dati, di solito una prima colonna di intestazioni di riga. Vedere l'esempio per **spiderplot (p. 83)**.

Xticlabels Le etichette dei tic degli assi possono essere generate tramite una funzione stringa, di solito prendendo una colonna di dati come argomento. La forma più semplice usa la colonna di dati stessa come

una stringa. Cioè, xticlabels(N) è l'abbreviazione di xticlabels(stringcolumn(N)). Questo esempio usa il contenuto della colonna 3 come etichette dei tic dell'asse x.

```
plot 'datafile' using <xcol>:<ycol>:xticlabels(3) with <plotstyle>
```

Le etichette dei tic degli assi possono essere generate per qualsiasi asse del grafico: x x2 y y2 z. Gli specificatori **ticlabels(<labelcol>)** devono venire dopo tutti gli specificatori delle coordinate dei dati nella parte **using** del comando. Per ogni data point che ha un insieme valido di coordinate X,Y[,Z], il valore della stringa dato a xticlabels() viene aggiunto alla lista delle etichette xtic alla stessa coordinata X del punto a cui appartiene. **xticlabels()** può essere abbreviato in **xtic()** e così via.

Esempio:

```
splot "data" using 2:4:6:xtic(1):ytic(3):ztic(6)
```

In questo esempio le etichette dei tic degli assi x e y sono prese da colonne diverse rispetto ai valori delle coordinate x e y. I tic dell'asse z, invece, sono generate dalla coordinata z del punto corrispondente.

Esempio:

```
plot "data" using 1:2:xtic( $3 > 10. ? "A" : "B" )
```

Questo esempio mostra l'uso di una funzione con valore di stringa per generare le etichette dei tic dell'asse x. Ogni punto nel file di dati genera un segno di tic su x etichettato o "A" o "B" a seconda del valore nella colonna 3.

X2ticlabels Vedere **plot using xticlabels (p. 123)**.

Yticlabels Vedere **plot using xticlabels (p. 123)**.

Y2ticlabels Vedere **plot using xticlabels (p. 123)**.

Zticlabels Vedere **plot using xticlabels (p. 123)**.

Cbticlabels SPERIMENTALE (i dettagli possono cambiare in una versione futura) grafici 2D: le etichette della barra dei colori sono posizionate alla coordinata della palette usata dal grafico per la colorazione delle variabili "lc palette z". grafici 3D: le etichette della barra dei colori sono posizionate sulla coordinata z del punto. Si noti che nel caso di una mappa di calore 3D con colore variabile che non corrisponde a z, questa non è probabilmente l'etichetta corretta. Vedere anche **plot using xticlabels (p. 123)**.

Volatile

La keyword **volatile** in un comando plot, indica che i dati precedentemente letti dal flusso di input o dal file potrebbero non essere disponibili per la rilettura. Questo dice al programma di usare i comandi **refresh** piuttosto che **replot** quando possibile. Vedere **refresh (p. 134)**.

Errorbars

Le barre di errore sono supportate per i grafici di file di dati 2D leggendo da una a quattro colonne aggiuntive (o entry **using**); questi valori aggiuntivi sono utilizzati in modi diversi dai vari stili di barre di errore.

Nella situazione predefinita, **gnuplot** si aspetta di vedere tre, quattro o sei numeri su ogni linea del file di dati — o

```
(x, y, ydelta),
(x, y, ylow, yhigh),
(x, y, xdelta),
(x, y, xlow, xhigh),
(x, y, xdelta, ydelta), or
(x, y, xlow, xhigh, ylow, yhigh).
```

La coordinata x deve essere specificata. L'ordine dei numeri deve essere esattamente come indicato sopra, anche se il qualificatore **using** può manipolare l'ordine e fornire valori per le colonne mancanti. Per esempio,

```
plot 'file' with errorbars
plot 'file' using 1:2:(sqrt($1)) with xerrorbars
plot 'file' using 1:2:($1-$3):($1+$3):4:5 with xyerrorbars
```

L'ultimo esempio è per un file contenente una combinazione non supportata di errori relativi x e assoluti y. La entry **using** genera la x assoluta min e max dall'errore relativo.

La barra di errore y è una linea verticale plottata da (x, ylow) a (x, yhigh). Se ydelta è specificato invece di ylow e yhigh, sono derivati ylow = y - ydelta e yhigh = y + ydelta. Se ci sono solo due numeri sul record, yhigh e ylow sono entrambi impostati su y. La barra di errore x è una linea orizzontale computata allo stesso modo. Per ottenere linee plottate tra i data point, bisogna plottare (**plot**) il file di dati due volte, una volta con le barre di errore e una volta con le linee (ma è bene ricordare di usare l'opzione **notitle** su una di esse per evitare di avere due entry nella key). In alternativa, usare il comando errorlines (vedere **errorlines (p. 125)**).

I segni di tic alle estremità della barra sono controllati da **set errorbars**.

Se l'autoscaling è attivo, gli intervalli saranno regolati per includere le barre di errore.

Vedere anche la demo errorbars .

Vedere **plot using (p. 121)**, **plot with (p. 131)**, e **set style (p. 206)** per maggiori informazioni.

Errorlines

Le linee con barre di errore sono supportate per i grafici di file di dati 2D, leggendo da una a quattro colonne aggiuntive (o entry **using**); questi valori aggiuntivi sono usati in modi diversi dai vari stili di linee di errore.

Nella situazione predefinita, **gnuplot** si aspetta di vedere tre, quattro o sei numeri su ogni linea del file di dati — o

```
(x, y, ydelta),
(x, y, ylow, yhigh),
(x, y, xdelta),
(x, y, xlow, xhigh),
(x, y, xdelta, ydelta), or
(x, y, xlow, xhigh, ylow, yhigh).
```

La coordinata x deve essere specificata. L'ordine dei numeri deve essere indicato esattamente come sopra, anche se il qualificatore **using** può manipolare l'ordine e fornire valori per le colonne mancanti. Per esempio,

```
plot 'file' with errorlines
plot 'file' using 1:2:(sqrt($1)) with xerrorlines
plot 'file' using 1:2:($1-$3):($1+$3):4:5 with xyerrorlines
```

L'ultimo esempio è per un file contenente una combinazione non supportata di errori relativi x e assoluti y. L'entry **using** genera x min e max assoluti dall'errore relativo.

La barra di errore y è una linea verticale plottata da (x, ylow) a (x, yhigh). Se ydelta è specificato invece di ylow e yhigh, sono derivati ylow = y - ydelta e yhigh = y + ydelta. Se ci sono solo due numeri sul record,

yhigh e ylow sono entrambi impostati su y. La barra di errore x è una linea orizzontale computata allo stesso modo.

I segni di tic alle estremità della barra sono controllati da **set errorbars**.

Se l'autoscaling è attivo, gli intervalli saranno regolati per includere le barre di errore.

Vedere **plot using (p. 121)**, **plot with (p. 131)**, e **set style (p. 206)** per maggiori informazioni.

Funzioni

Le funzioni incorporate o definite dall'utente possono essere visualizzate dai comandi **plot** e **splot** in aggiunta, o al posto, dei dati letti da un file. La funzione richiesta viene valutata campionando a intervalli regolari che coprono l'intervallo di assi indipendenti (independent axis range[s]). Vedere **set samples (p. 204)** e **set isosamples (p. 164)**. Esempio:

```
approx(ang) = ang - ang**3 / (3*2)
plot sin(x) title "sin(x)", approx(x) title "approximation"
```

Per impostare uno stile di grafico predefinito per le funzioni, vedere **set style function (p. 210)**. Per informazioni sulle funzioni incorporate, vedere **espressioni funzioni (p. 38)**. Per informazioni su come definire le proprie funzioni, vedere **user-defined (p. 45)**.

Parametric

Quando si è in modalità parametrica (**set parametric**), le espressioni matematiche devono essere fornite in coppia per **plot** e in terna per **splot**.

Esempi:

```
plot sin(t),t**2
splot cos(u)*cos(v),cos(u)*sin(v),sin(u)
```

I file di dati sono plottati come prima, eccetto che ogni funzione parametrica precedente deve essere completamente specificata prima che un file di dati sia dato come grafico. In altre parole, la funzione parametrica x (**sin(t)** sopra) e la funzione parametrica y (**t**2** sopra) non devono essere interrotte da alcun modificatore o funzione dati; così facendo si genera un errore di sintassi che indica che la funzione parametrica non è completamente specificata.

Altri modificatori, come **with** o **title**, possono essere specificati solo dopo che la funzione parametrica è stata completata:

```
plot sin(t),t**2 title 'Parametric example' with linespoints
```

Vedere anche `Parametric Mode Demos.`

Ranges

Questa sezione descrive solo gli intervalli di assi opzionali che possono apparire come i primi elementi di un comando **plot**. Se presenti, questi intervalli sovrascrivono qualsiasi limite di intervallo stabilito da una precedente istruzione **set range**. Per gli intervalli opzionali in altri parti di un comando **plot** che limitano il campionamento di un singolo componente del grafico, vedere **sampling (p. 127)**.

Sintassi:

```
[{<dummy-var>=}{{<min>}:{<max>}}]
[{{<min>}:{<max>}}]
```

La prima forma si applica alla variabile indipendente (**xrange** o **trange**, se in modalità parametrica). La seconda forma si applica alle variabili dipendenti. <dummy-var> stabilisce, su richiesta, un nuovo nome per la variabile indipendente. (Il nome predefinito può essere cambiato con **set dummy**).

In modalità non-parametrica, gli intervalli devono essere dati nell'ordine

```
plot [<xrange>][<yrange>][<x2range>][<y2range>] ...
```

In modalità parametrica, gli intervalli devono essere dati nell'ordine

```
plot [<trange>][<xrange>][<yrange>][<x2range>][<y2range>] ...
```

Il seguente comando **plot** mostra l'impostazione di **trange** a [-pi:pi], **xrange** a [-1.3:1.3] e **yrange** a [-1:1] per la durata del grafico (graph):

```
plot [-pi:pi] [-1.3:1.3] [-1:1] sin(t),t**2
```

***** può essere usato per permettere l'autoscaling di min o max. Usare un intervallo vuoto [] come placeholder se necessario.

Gli intervalli specificati sulla linea di comando **plot** o **splot** hanno effetto solo su quel grafico (graph); usa i comandi **set xrange**, **set yrange**, ecc. per cambiare gli intervalli predefiniti per i grafici (graph) futuri.

L'uso di specificatori di intervallo sul momento in un comando plot potrebbe non produrre il risultato previsto per gli assi collegati (vedere **set link (p. 174)**).

Per i dati di tempo è necessario fornire l'intervallo tra virgolette, usando lo stesso formato usato per leggere il tempo dal file di dati. Vedere **set timefmt (p. 218)**.

Esempi:

Questo usa gli intervalli attuali:

```
plot cos(x)
```

Questo imposta solo l'intervallo x:

```
plot [-10:30] sin(pi*x)/(pi*x)
```

Questo è lo stesso, ma usa t come variabile dummy:

```
plot [t = -10 :30]  sin(pi*t)/(pi*t)
```

Questo imposta entrambi gli intervalli x e y:

```
plot [-pi:pi] [-3:3]  tan(x), 1/x
```

Questo imposta solo l'intervallo y:

```
plot [ ] [-2:sin(5)*-8] sin(x)**besj0(x)
```

Questo imposta solo xmax e ymin:

```
plot [:200] [-pi:]  $mydata using 1:2
```

Questo imposta l'intervallo x per una serie temporale:

```
set timefmt "%d/%m/%y %H:%M"
plot ["1/6/93 12:00":"5/6/93 12:00"] 'timedata.dat'
```

Sampling

1D sampling (x or t axis)

Per default, le funzioni computate o i dati generati per lo pseudo-file "+" sono campionati sull'intero intervallo del grafico, come impostato da un precedente comando **set xrange**, da un esplicito specificatore di intervallo globale all'inizio del comando plot o del comando splot, o autoscalando l'intervallo x (xrange) per coprire i dati visti in tutti gli elementi di questo grafico. Tuttavia, ai singoli componenti del grafico può essere assegnato un intervallo di campionamento più ristretto.

Esempi:

Questo stabilisce un intervallo totale su x che va da 0 a 1000 e poi plotta dati da un file e due funzioni che coprono ciascuna una porzione dell'intervallo totale:

```
plot [0:1000] 'datafile', [0:200] func1(x), [200:500] func2(x)
```

Questo è simile, tranne per il fatto che l'intervallo totale è stabilito dai contenuti del file di dati. In questo caso le funzioni campionate possono essere o non essere interamente contenute nel grafico:

```
set autoscale x
plot 'datafile', [0:200] func1(x), [200:500] func2(x)
```

Questo comando è ambiguo. L'intervallo iniziale sarà interpretato come applicato all'intero grafico, non solo al campionamento della prima funzione come era probabilmente l'obiettivo:

```
plot [0:10] f(x), [10:20] g(x), [20:30] h(x)
```

Questo comando rimuove l'ambiguità dell'esempio precedente inserendo la keyword **sample** in modo che l'intervallo non sia applicato all'intero grafico:

```
plot sample [0:10] f(x), [10:20] g(x), [20:30] h(x)
```

Questo esempio mostra un modo di tracciare un'elica in un grafico 3D

```
splot [-2:2][-2:2] sample [h=1:10] '+' using (cos(h)):(sin(h)):(h)
```

2D sampling (assi u e v)

Le funzioni computate o i dati generati per lo pseudo-file '++' usano campioni generati lungo gli assi u e v. Questo è un CAMBIAMENTO rispetto alle versioni precedenti alla 5.2 che campionavano lungo gli assi x e y. Vedere **special-filenames ++ (p. 120)**. Il campionamento 2D può essere usato sia nei comandi **plot** che **splot**.

Esempio di campionamento 2D in un comando 2D **plot**. Questi comandi hanno generato il grafico mostrato per lo stile di grafico **with vectors**. Vedere **vectors (p. 84)**.

```
set urange [ -2.0 : 2.0 ]
set vrange [ -2.0 : 2.0 ]
plot '++' using ($1):($2):($2*0.4):(-$1*0.4) with vectors
```

Esempio di campionamento 2D in un comando 3D **splot**. Questi comandi sono simili a quelli usati in **sampling.dem**. Si noti che le due superfici sono campionate su intervalli u e v più piccoli degli intervalli x e y completi del grafico risultante.

```
set title "3D sampling range distinct from plot x/y range"
set xrange [1:100]
set yrange [1:100]
splot sample [u=30:70][v=0:50] '++' using 1:2:(u*v) lt 3, \
      [u=40:80][v=30:60] '++' using (u):(v):(u*sqrt(v)) lt 4
```

Gli specificatori di intervallo per il campionamento su u e v possono includere un intervallo di campionamento esplicito per controllare il numero e la spaziatura dei campioni:

```
splot sample [u=30:70:1][v=0:50:5] '++' using 1:2:(func($1,$2))
```

Per loop in comandi plot

Se molti file o funzioni simili devono essere plottati insieme, può essere utile farlo iterando su un comando plot condiviso.

Sintassi:

```
plot for [<variable> = <start> : <end> {:<increment>}]
plot for [<variable> in "string of words"]
```

L'ambito di un'iterazione termina alla virgola successiva o alla fine del comando, quello che viene prima. Un'eccezione a questo è data dal fatto che le definizioni sono raggruppate con l'elemento seguente del grafico anche se c'è una virgola in mezzo. Si noti che l'iterazione non funziona per i grafici in modalità parametrica.

Esempio:

```
plot for [j=1:3] sin(j*x)
```

Esempio:

```
plot for [dataset in "apples bananas"] dataset."dat" title dataset
```

In questo esempio l'iterazione è usata sia per generare un nome di file che un titolo corrispondente.

Esempio:

```
file(n) = sprintf("dataset_%d.dat",n)
splot for [i=1:10] file(i) title sprintf("dataset %d",i)
```

Questo esempio definisce una funzione con valore di stringa che genera nomi di file, e plotta dieci di questi file insieme. La variabile di iterazione ('i' in questo esempio) è trattata come un intero e può essere usata più di una volta.

Esempio:

```
set key left
plot for [n=1:4] x**n sprintf("%d",n)
```

Questo esempio plotta una famiglia di funzioni.

Esempio:

```
list = "apple banana cabbage daikon eggplant"
item(n) = word(list,n)
plot for [i=1:words(list)] item[i].".dat" title item(i)
list = "new stuff"
replot
```

Questo esempio passa attraverso una lista e plotta una volta per ogni elemento. Poiché gli elementi vengono recuperati dinamicamente, è possibile cambiare la lista e poi rifare (replot) il grafico.

Esempio:

```
list = "apple banana cabbage daikon eggplant"
plot for [i in list] i.".dat" title i
list = "new stuff"
replot
```

Questo esempio fa esattamente la stessa cosa dell'esempio precedente, ma usa la forma iteratore di stringa del comando piuttosto che un iteratore intero.

Se un'iterazione deve continuare finché tutti i dati disponibili sono esauriti, utilizzare il simbolo * invece di un intero <end>. Ciò può essere usato per elaborare tutte le colonne in una linea, tutti gli insiemi di dati (separati da 2 linee vuote) in un file, o tutti i file che corrispondono a un template.

Esempi:

```
plot for [i=2:*] 'datafile' using 1:i with histogram
splot for [i=0:*] 'datafile' index i using 1:2:3 with lines
plot for [i=1:*] file=sprintf("File_%03d.dat",i) file using 2 title file
```

Title

Per default, ogni grafico è elencato nella key in base al nome della funzione o del file corrispondente. È possibile invece dare un titolo esplicito al grafico usando l'opzione **title**.

Sintassi:

```
title <text> | notitle [<ignored text>]
title columnheader | title columnheader(N)
      {at {beginning|end}} {{no}enhanced}
```

dove <text> è una stringa tra virgolette o un'espressione che valuta una stringa. Le virgolette non saranno mostrate nella key. Nota: A partire dalla versione 5.4 di gnuplot, se <text> è un'espressione o una funzione esso viene valutato dopo che la funzione o il flusso di dati corrispondente viene plottato. Questo permette al titolo di fare riferimento a quantità calcolate o inserite durante il plotting, cosa che non era possibile nelle precedenti versioni di gnuplot.

Esiste anche un'opzione che interpreta la prima voce in una colonna di dati di input (cioè l'intestazione della colonna) come un campo di testo, e lo userà come titolo della key. Vedere **stringhe di dati (p. 34)**. Questo può essere reso predefinito specificando **set key autotitle columnhead**.

Il titolo della linea e il campione possono essere omessi dalla key usando la keyword **notitle**. Un titolo nullo (**title ''**) è equivalente a **notitle**. Se si vuole solo il campione, bisogna usare uno o più spazi vuoti (**title ' '**). Se **notitle** è seguito da una stringa, questa stringa viene ignorata.

Se è impostato **key autotitles** (che è il default) e non sono specificati né **title** né **notitle**, il titolo della linea è il nome della funzione o il nome del file come come appare nel comando **plot**. Se è un file name, qualsiasi modificatore di file di dati specificato sarà incluso nel titolo predefinito.

Il layout della key stessa (posizione, allineamento del titolo, ecc.) può essere controllato usando **set key (p. 166)**.

La keyword **at** permette di posizionare il titolo del grafico da qualche parte al di fuori della casella key generata automaticamente. Il titolo può essere posizionato immediatamente prima o dopo la linea nel grafico (graph) stesso usando **at {beginning|end}**. Questa opzione può essere utile quando si plotta con **with lines**, ma ha poco senso per la maggior parte degli altri stili.

Per mettere il titolo del grafico in una posizione arbitraria della pagina, è necessario usare la forma **at <x-position>,<y-position>**. Per impostazione predefinita la posizione è interpretata in coordinate dello schermo; ad es., **at 0.5, 0.5** è sempre il centro dello schermo indipendentemente dalle scale degli assi o dai bordi del grafico. Il formato dei titoli posizionati in questo modo è ancora influenzato dalle opzioni key. Vedere **set key (p. 166)**.

Esempi:

Questo plotta y=x con il titolo 'x':

```
plot x
```

Questo plotta x al quadrato con titolo "x^2" e file "data.1" con titolo "measured data":

```
plot x**2 title "x^2", 'data.1' t "measured data"
```

Plotta multiple colonne di dati, ognuna delle quali contiene il proprio titolo sulla prima linea del file. Posiziona i titoli dopo le linee corrispondenti piuttosto che in una key separata:

```
unset key
set offset 0, graph 0.1
plot for [i=1:4] 'data' using i with lines title columnhead at end
```

```
Crea una singola area key per due grafici separati:
     set key Left reverse
     set multiplot layout 2,2
     plot sin(x) with points pt 6 title "Left plot is sin(x)" at 0.5, 0.30
     plot cos(x) with points pt 7 title "Right plot is cos(x)" at 0.5, 0.27
     unset multiplot
```

With

Le funzioni e i dati possono essere visualizzati in uno tra molti stili. La keyword **with** fornisce i mezzi di selezione.

Sintassi:

```
with <style> { {linestyle | ls <line_style>}
               | {{linetype  | lt <line_type>}
                  {linewidth | lw <line_width>}
                  {linecolor | lc <colorspec>}
                  {pointtype | pt <point_type>}
                  {pointsize | ps <point_size>}
                  {arrowstyle | as <arrowstyle_index>}
                  {fill | fs <fillstyle>} {fillcolor | fc <colorspec>}
                  {nohidden3d} {nocontours} {nosurface}
                  {palette}}
             }
```

dove <style> è uno di

```
lines        dots       steps     vectors      yerrorlines
points       impulses   fsteps    xerrorbar    xyerrorbars
linespoints  labels     histeps   xerrorlines  xyerrorlines
financebars  surface    arrows    yerrorbar    parallelaxes
```

o

```
boxes         boxplot        ellipses       histograms  rgbalpha
boxerrorbars  candlesticks   filledcurves   image       rgbimage
boxxyerror    circles        fillsteps      pm3d        polygons
isosurface    zerrorfill
```

o

```
table
```

Il primo gruppo di stili ha proprietà di linea, punto e testo associate. Il secondo gruppo di stili ha anche proprietà di riempimento. Vedere **fillstyle (p. 209)**. Alcuni stili hanno ulteriori sotto-stili. Vedere **plotting styles (p. 63)** per i dettagli di ciascuno. Lo stile **table** produce un output tabulare piuttosto che un grafico. Vedere **set table (p. 214)**.

Uno stile predefinito può essere scelto da **set style function** e **set style data**.

Per default, ogni funzione e file di dati userà un diverso tipo di linea e tipo di punto, fino al numero massimo di tipi disponibili. Tutti i driver supportano almeno sei tipi di punto diversi, e li riutilizzano, in ordine, se ne sono necessari altri. Per vedere l'insieme completo dei tipi di linea e di punto disponibili per il terminale corrente, digitare **test (p. 245)**.

Se si desidera scegliere il tipo di linea o di punto per un singolo grafico, <line_type> e <point_type> possono essere specificati. Queste sono costanti (o espressioni) intere positive che specificano il tipo di linea e il tipo di punto da usare per il grafico. Usare **test** per visualizzare i tipi disponibili per il proprio terminale.

Si può anche scalare lo spessore della linea e la dimensione del punto per un grafico usando <line_width> e <point_size>, che sono specificati rispetto ai valori predefiniti per ogni terminale. La dimensione del punto può anche essere alterata globalmente — vedere **set pointsize (p. 202)** per i dettagli. Ma si noti che sia <point_size> come impostato qui che come impostato da **set pointsize** moltiplicano la dimensione predefinita del punto — i loro effetti non sono cumulativi. Cioè, **set pointsize 2; plot x w p ps 3** userà punti grandi tre volte la dimensione predefinita, non sei.

È anche possibile specificare la variabile **pointsize variable** sia come parte di uno stile di linea o per un grafico individuale. In questo caso è necessaria una colonna extra di input, cioè 3 colonne per un grafico 2D

e 4 colonne per un grafico (splot) 3D. La dimensione di ogni singolo punto è determinata moltiplicando la dimensione dei punti globale per il valore letto dal file di dati.

Se si sono definite combinazioni specifiche di tipo/spessore di linea e tipo/dimensione del punto con **set style line**, uno di queste può essere selezionata impostando <line_style> all'indice dello stile desiderato.

Sia i grafici 2D che 3D (comandi **plot** e **splot**) possono usare i colori da una palette uniforme impostata in precedenza con il comando **set palette**. Il valore di colore corrisponde al valore z del punto stesso o una coordinata di colore separata fornita in una colonna aggiuntiva opzionale **using**. I valori di colore possono essere trattati sia come una frazione dell'intervallo della palette (**palette frac**) o come un valore di coordinate mappato sull'intervallo del colorbox (**palette** o **palette z**). Vedere **colorspec (p. 50)**, **set palette (p. 190)**, **linetype (p. 173)**.

La keyword **nohidden3d** si applica solo ai grafici creati con il comando **splot**. Normalmente l'opzione globale **set hidden3d** si applica a tutti i grafici (plot) del grafico (graph). È possibile associare l'opzione **nohidden3d** a qualsiasi grafico individuale che si vuole escludere dall'elaborazione hidden3d. I singoli elementi diversi dalle superfici (cioè linee, punti, etichette, ...) di un grafico marcato **nohidden3d** verranno tutti disegnati, anche se normalmente sarebbero oscurati da altri elementi del grafico.

In modo simile, la keyword **nocontours** disattiverà il contouring per un singolo grafico anche se la proprietà globale **set contour** è attiva.

Allo stesso modo, la keyword **nosurface** disattiverà la superficie 3D per un singolo grafico anche se la proprietà globale **set surface** è attiva.

Le keywords possono essere abbreviate come indicato.

Notare che le opzioni **linewidth**, **pointsize** e **palette** non sono supportate da tutti i terminali.

Esempi:

Questo plotta sin(x) con impulsi:

```
plot sin(x) with impulses
```

Questo plotta x con punti, x**2 con l'impostazione predefinita:

```
plot x w points, x**2
```

Questo plotta tan(x) con lo stile di funzione predefinito, il file "data.1" con le linee:

```
plot [ ] [-2:5] tan(x), 'data.1' with l
```

Questo plotta "leastsq.dat" con impulsi:

```
plot 'leastsq.dat' w i
```

Queso plotta il file di dati "population" con box:

```
plot 'population' with boxes
```

Questo plotta "exper.dat" con barre di errore e linee che collegano i punti (le barre di errore richiedono tre o quattro colonne):

```
plot 'exper.dat' w lines, 'exper.dat' notitle w errorbars
```

Un altro modo per plottare "exper.dat" con linee di errore (le barre di errore richiedono tre o quattro colonne):

```
plot 'exper.dat' w errorlines
```

Questo plotta sin(x) e cos(x) con linee-punti, usando lo stesso tipo di linea ma diversi tipi di punti:

```
plot sin(x) with linesp lt 1 pt 3, cos(x) with linesp lt 1 pt 4
```

Questo plotta il "data" file con punti di tipo 3 e di dimensioni doppie rispetto al solito:

```
plot 'data' with points pointtype 3 pointsize 2
```

Questo plotta il "data" file con variabile pointsize letta dalla colonna 4:

```
plot 'data' using 1:2:4 with points pt 5 pointsize variable
```

Questo plotta due serie di dati con linee che differiscono solo per il peso:

```
plot 'd1' t "good" w l lt 2 lw 3, 'd2' t "bad" w l lt 2 lw 1
```

Questo plotta la curva riempita di x*x e una striscia di colore:

```
plot x*x with filledcurve closed, 40 with filledcurve y=10
```

Questo plotta x*x e una casella di colore (color box):

```
plot x*x, (x>=-5 && x<=5 ? 40 : 1/0) with filledcurve y=10 lt 8
```

Questo plotta una superficie con linee di colore:

```
splot x*x-y*y with line palette
```

Questo plotta due superfici di colore a diverse altitudini:

```
splot x*x-y*y with pm3d, x*x+y*y with pm3d at t
```

Print

Il comando **print** stampa il valore di <expression> (espressione) sullo schermo. È sinonimo di **pause 0**. <expression> può essere qualsiasi cosa che **gnuplot** può valutare che produca un numero, o può essere una stringa.

Sintassi:

```
print <expression> {, <expression>, ...}
```

Vedere **espressioni (p. 37)**. Il file output può essere impostato con **set print**. Vedere anche **printerr (p. 133)**.

Printerr

printerr è uguale a print tranne che l'output viene sempre inviato a stderr anche se un precedente comando **set print** rimane in vigore.

Pwd

Il comando **pwd** stampa il nome della directory di lavoro sullo schermo.

Notare che se si vuole memorizzare la directory corrente in una variabile stringa o usarla in espressioni stringa, allora si può usare la variabile GPVAL_PWD, vedere **show variables all (p. 221)**.

Quit

I comandi **exit** e **quit** e il carattere END-OF-FILE chiuderanno **gnuplot**. Ognuno di questi comandi svuoterà il dispositivo di output (come fa il comando **clear**) prima di uscire.

Raise

Sintassi:

```
raise {plot_window_id}
lower {plot_window_id}
```

I comandi **raise** e **lower** funzionano solo per alcuni tipi di terminale e possono dipendere anche dal proprio window manager e dalle impostazioni delle preferenze di visualizzazione. Un esempio di utilizzo è mostrato qui

```
set term wxt 123     # crea la prima finestra del grafico
plot $FOO
lower                # abbassa l'unica finestra di grafico che esiste finora
set term wxt 456     # crea una seconda finestra di grafico può occludere
                     # la prima
plot $BAZ
raise 123            # alza la prima finestra di grafico
```

Questi comandi sono noti per essere inaffidabili.

Refresh

Il comando **refresh** è simile a **replot**, con due differenze importanti. **refresh** riformatta e ridisegna il grafico attuale usando i dati già letti. Questo significa che si può usare **refresh** per i grafici con dati in linea (pseudo-dispositivo '-') e per grafici da file di dati il cui contenuto è volatile. Non si può usare il comando **refresh** per aggiungere nuovi dati a un grafico esistente.

Le operazioni con mouse, in particolare lo zoom e l'unzoom, useranno **refresh** piuttosto che **replot**, se appropriato. Esempio:

```
plot 'datafile' volatile with lines, '-' with labels
100 200 "Special point"
e
# Qui vanno varie operazioni di mousing
set title "Zoomed in view"
set term post
set output 'zoom.ps'
refresh
```

Replot

Il comando **replot** senza argomenti ripete l'ultimo comando **plot** o **splot**. Questo può essere utile per visualizzare un grafico con diverse opzioni **set**, o quando si genera lo stesso grafico per diversi dispositivi.

Gli argomenti specificati dopo un comando **replot** saranno aggiunti all'ultimo comando **plot** o **splot** (con un separatore implicito ',') prima di essere ripetuto. **replot** accetta gli stessi argomenti dei comandi **plot** e **splot** eccetto che gli intervalli non possono essere specificati. Quindi si può usare **replot** per plottare una funzione sul secondo asse se il comando precedente era **plot** ma non se era **splot**.

N.B. — l'uso di

```
plot '-' ; ... ; replot
```

non è consigliato, perché richiederà di digitare di nuovo i dati. Nella maggior parte dei casi si può invece usare il comando **refresh**, che ridisegnerà il grafico usando i dati precedentemente letti.

Si noti che nella modalità multiplot, **replot** può riprodurre solo il più recente dei componenti del grafico, non l'intera serie.

Vedere anche **editing della linea di comando (p. 33)** per modi di modificare l'ultimo comando **plot (p. 107)** (**splot (p. 237)**).

Vedere anche **show plot (p. 197)** per mostrare l'intero comando di plotting corrente, e la possibilità di copiarlo nella **history (p. 103)** (cronologia).

Reread

[DEPRECATO nella versione 5.4]

Questo comando è deprecato in favore dell'iterazione esplicita. Vedere **iterate (p. 49)**. Il comando **reread** fa sì che il file di comando **gnuplot** corrente, come specificato da un comando **load**, sia resettato al suo punto di partenza prima che altri comandi vengano letti da esso. Questo essenzialmente implementa un loop infinito dei comandi dall'inizio del file di comando al comando **reread**. Il comando **reread** non ha effetto quando si legge in modo interattivo (da stdin).

Reset

```
reset {bind | errors | session}
```

Il comando **reset** fa sì che tutte le opzioni relative al grafico (graph) che possono essere impostate con il comando **set**, tornino ai loro valori predefiniti. Questo comando può essere usato per ripristinare le impostazioni predefinite dopo l'esecuzione di un file di comando caricato (loaded), o per tornare ad uno stato definito dopo che molte impostazioni sono state cambiate.

I seguenti *non* sono influenzati dal **reset**:

```
‘set term‘ ‘set output‘ ‘set loadpath‘ ‘set linetype‘ ‘set fit‘
‘set encoding‘ ‘set decimalsign‘ ‘set locale‘ ‘set psdir‘
‘set overflow‘ ‘set multiplot‘
```

Si noti che **reset** non riporta necessariamente le impostazioni allo stato in cui si trovavano all'ingresso del programma, perché i valori predefiniti possono essere stati alterati da comandi nei file di inizializzazione gnuplotrc o $HOME/.gnuplot. Tuttavia, questi comandi possono essere rieseguiti usando la variante del comando **reset session**.

reset session cancella tutte le variabili e funzioni definite dall'utente, ripristina le impostazioni predefinite, e poi riesegue il file di inizializzazione di sistema gnuplotrc e qualsiasi file privato delle preferenze $HOME/.gnuplot. Vedere **initialization (p. 57)**.

reset errors cancella solo le variabili di stato di errore GPVAL_ERRNO e GPVAL_ERRMSG.

reset bind ripristina tutte le associazioni dei tasti di scelta rapida al loro stato predefinito.

Save

Sintassi:

```
save  {functions | variables | terminal | set | fit} ’<filename>’
```

Se non viene specificata alcuna opzione, **gnuplot** salva le funzioni, le variabili, le opzioni **set** e l'ultimo comando **plot** (**splot**).

I file salvati con **save** sono scritti in formato testo e possono essere letti dal comando **load**. Per **save** con l'opzione **set** o senza alcuna opzione, la scelta del **terminal** e il filename **output** sono scritti come un

commento, per ottenere un file di output che funzioni in altre installazioni di gnuplot, senza modifiche e senza il rischio di sovrascrivere involontariamente i file.

save terminal scriverà solo lo stato del **terminal**, senza il marcatore di commento davanti ad esso. Questo è utile principalmente per cambiare l'impostazione del **terminal** per un breve periodo, e tornare al terminale precedentemente impostato, in seguito, caricando lo stato del **terminal** salvato. Si noti che per una singola sessione di gnuplot si può usare piuttosto l'altro metodo per salvare e ripristinare il terminale corrente con i comandi **set term push** e **set term pop**, vedere **set term (p. 215)**.

save fit salva solo le variabili usate nel comando **fit** più recente. Il file salvato può essere usato come file di parametri per inizializzare futuri comandi fit usando la keyword **via**.

Il filename deve essere racchiuso tra virgolette.

Il filename speciale `"-"` può essere usato per salvare con **save** i comandi sullo standard output. Sui sistemi che supportano una funzione popen (Unix), l'output di save può essere convogliato (piped) attraverso un programma esterno facendo iniziare il filename con un '|'. Questo fornisce un'interfaccia coerente alle impostazioni interne di **gnuplot** a programmi che comunicano con **gnuplot** attraverso una pipe. Si prega di vedere **batch/interactive (p. 31)** per maggiori dettagli.

Esempi:

```
save 'work.gnu'
save functions 'func.dat'
save var 'var.dat'
save set 'options.dat'
save term 'myterm.gnu'
save '-'
save '|grep title >t.gp'
```

Set-show

Il comando **set** può essere usato per impostare *molte* opzioni. Nessuna schermata è disegnata, tuttavia, finché non viene dato un comando **plot**, **splot** o **replot**.

Il comando **show** mostra le loro impostazioni; **show all** mostra tutte le impostazioni.

Le opzioni cambiate usando **set** possono essere riportate allo stato predefinito dando il corrispondente comando **unset**. Vedere anche il comando **reset (p. 135)**, che riporta tutti i parametri impostabili ai valori predefiniti.

I comandi **set** e **unset** possono eventualmente contenere una clausola di iterazione. Vedere **plot for (p. 128)**.

Angles

Per default, **gnuplot** presuppone che la variabile indipendente nei grafici (graph) polari sia in unità di radianti. Se **set angles degrees** è specificato prima di **set polar**, allora l'intervallo predefinito è [0:360] e la variabile indipendente ha unità di gradi. Questo è particolarmente utile per i grafici di file di dati. L'impostazione dell'angolo si applica anche alla mappatura 3D come impostata tramite il comando **set mapping**.

Sintassi:

```
set angles {degrees | radians}
show angles
```

L'angolo specificato in **set grid polar** viene anche letto e visualizzato nelle unità specificate da **set angles**.

set angles influenza anche gli argomenti delle funzioni definite dalla macchina sin(x), cos(x) e tan(x), e gli output di asin(x), acos(x), atan(x) atan2(x), e arg(x). Non ha effetto sugli argomenti delle funzioni

iperboliche o delle funzioni di Bessel. Tuttavia, gli argomenti di output delle funzioni iperboliche inverse di argomenti complessi sono influenzati; se queste funzioni vengono utilizzate, **set angles radiants** deve essere in funzione per mantenere la coerenza tra gli argomenti di input e di output.

```
x={1.0,0.1}
set angles radians
y=sinh(x)
print y         #stampa {1.16933, 0.154051}
print asinh(y)  #stampa {1.0, 0.1}
```

ma

```
set angles degrees
y=sinh(x)
print y         #stampa {1.16933, 0.154051}
print asinh(y)  #stampa {57.29578, 5.729578}
```

Vedere anche `poldat.dem: polar plot using set angles demo.`

Arrow

Frecce arbitrarie possono essere posizionate su un grafico usando il comando **set arrow**.

Sintassi:

```
set arrow {<tag>} from <position> to <position>
set arrow {<tag>} from <position> rto <position>
set arrow {<tag>} from <position> length <coord> angle <ang>
set arrow <tag> arrowstyle | as <arrow_style>
set arrow <tag> {nohead | head | backhead | heads}
                {size <headlength>,<headangle>{,<backangle>}} {fixed}
                {filled | empty | nofilled | noborder}
                {front | back}
                {linestyle | ls <line_style>}
                {linetype | lt <line_type>}
                {linewidth | lw <line_width>}
                {linecolor | lc <colorspec>}
                {dashtype | dt <dashtype>}

unset arrow {<tag>}
show arrow {<tag>}
```

<tag> è un intero che identifica la freccia. Se non viene dato alcun tag, viene assegnato automaticamente il valore di tag più basso non utilizzato. Il tag può essere usato per cancellare o cambiare una specifica freccia. Per cambiare qualsiasi attributo di una freccia esistente, bisogna usare il comando **set arrow** con il tag appropriato e specificare le parti della freccia da cambiare.

La posizione del primo end point della freccia è sempre specificata da "from". L'altro end point può essere specificato usando uno di tre diversi meccanismi. Le <position> (posizioni) sono specificate da x,y o x,y,z, e possono essere precedute da **first**, **second**, **graph**, **screen**, o **character** per selezionare il sistema di coordinate. Le coordinate non specificate hanno come valore predefinito 0. Vedere **coordinate (p. 33)** per i dettagli. Uno specificatore di sistema di coordinate non viene riportato dalla descrizione del primo endpoint al secondo.

1) "to <position>" specifica le coordinate assolute dell'altra estremità.

2) "rto <position>" specifica un offset alla posizione "from". Per gli assi lineari, coordinate **graph** e **screen**, la distanza tra il punto iniziale e punto finale corrisponde alla coordinata relativa data. Per gli assi logaritmici

la coordinata relativa data corrisponde al fattore della coordinata tra il punto iniziale e quello finale. Così, un valore relativo negativo o zero non sono ammessi per gli assi logaritmici.

3) "length <coordinate> angle <angle>" specifica l'orientamento della freccia nel piano del grafico (graph). Anche in questo caso uno qualsiasi dei sistemi di coordinate può essere usato per specificare la lunghezza. L'angolo è sempre in gradi.

Altre caratteristiche della freccia possono essere specificate come uno stile di freccia predefinito o fornendole nel comando **set arrow**. Per una dettagliata spiegazione delle caratteristiche della freccia, vedere **arrowstyle (p. 206)**.

Esempi:

Per impostare una freccia che punta dall'origine a (1,2) con stile linea definito dall'utente 5, usare:

```
set arrow to 1,2 ls 5
```

Per impostare una freccia dal basso a sinistra dell'area di plotting a (-5,5,3), ed etichettare (tag) la freccia numero 3, usare:

```
set arrow 3 from graph 0,0 to -5,5,3
```

Per cambiare la freccia precedente e farla finire a 1,1,1, senza la punta della freccia e raddoppiare il suo spessore, usare:

```
set arrow 3 to 1,1,1 nohead lw 2
```

Per tracciare una linea verticale dal basso verso l'alto del grafico a x=3, usare:

```
set arrow from 3, graph 0 to 3, graph 1 nohead
```

Per disegnare una freccia verticale con estremità a forma di T, usare:

```
set arrow 3 from 0,-5 to 0,5 heads size screen 0.1,90
```

Per disegnare una freccia relativamente al punto di partenza, dove le distanze relative sono date in coordinate del grafico (graph), usare:

```
set arrow from 0,-5 rto graph 0.1,0.1
```

Per disegnare una freccia con punto finale relativo nell'asse x logaritmico, usare:

```
set logscale x
set arrow from 100,-5 rto 10,10
```

Questo disegna una freccia da 100,-5 a 1000,5. Per l'asse x logaritmico, la coordinata relativa 10 significa "factor 10" (fattore) mentre per l'asse y lineare, la coordinata relativa 10 significa "difference 10" (differenza).

Per cancellare la freccia numero 2, usare:

```
unset arrow 2
```

Per cancellare tutte le frecce, usare:

```
unset arrow
```

Per mostrare tutte le frecce (in ordine di tag), usare:

```
show arrow
```

`arrows demos.`

Autoscale

Autoscaling può essere impostato individualmente sull'asse x, y o z o globalmente su tutti gli assi. L'impostazione predefinita è autoscalare tutti gli assi. Se si desidera autoscalare in base a un sottoinsieme dei grafici nella figura, è possibile contrassegnare gli altri con il flag **noautoscale**. Vedere **datafile (p. 112)**.

Sintassi:

```
set autoscale {<axes>{|min|max|fixmin|fixmax|fix} | fix | keepfix}
set autoscale noextend
unset autoscale {<axes>}
show autoscale
```

dove <axes> è **x**, **y**, **z**, **cb**, **x2**, **y2**, **xy**, o **paxis {n}**. Una keyword con **min** o **max** aggiunto (questo non può essere fatto con **xy**) dice a **gnuplot** di autoscalare solo il minimo o il massimo di quell'asse. Se non viene data alcuna keyword, vengono autoscalati tutti gli assi.

Quando si autoscala, l'intervallo degli assi viene computato automaticamente e l'asse dipendente (y per un **plot** e z per **splot**) viene scalato per includere l'intervallo della funzione o dei dati plottati.

Se l'autoscaling dell'asse dipendente (y o z) non è impostato, viene utilizzato l'intervallo y o z corrente.

L'autoscaling delle variabili indipendenti (x per **plot** e x,y per **splot**) è una richiesta per impostare il dominio in modo che corrisponda a qualsiasi file di dati che si sta plottando. Se non ci sono file di dati, l'autoscaling di una variabile indipendente non ha alcun effetto. In altre parole, in assenza di un file di dati, le funzioni da sole non influenzano l'intervallo x (o l'intervallo y se si plotta z = f(x,y)).

Si prega di vedere **set xrange (p. 226)** per ulteriori informazioni sugli intervalli.

Il comportamento dell'autoscaling rimane coerente nella modalità parametrica, (vedere **set parametric (p. 195)**). Tuttavia, ci sono più variabili dipendenti e quindi più controllo sulle scale degli assi x, y e z. In modalità parametrica, la variabile indipendente o dummy è t per grafici **plot** e u, v per grafici **splot**. **autoscale** in modalità parametrica, quindi, controlla tutti gli intervalli (t, u, v, x, y e z) e permette a x, y e z di essere completamente autoscalati.

Quando i tic vengono mostrati sui secondi assi ma non è stato specificato alcun grafico per questi assi, x2range e y2range sono ereditati da xrange e yrange. Questo viene fatto *prima* di applicare offset o di estendere automaticamente gli intervalli ad un intero numero di tic, il che può provocare risultati inaspettati. Per evitare questo si può collegare esplicitamente l'intervallo dell'asse secondario all'intervallo dell'asse primario. Vedere **set link (p. 174)**.

Noextend

```
set autoscale noextend
```

Per default l'autoscaling imposta i limiti dell'intervallo dell'asse alla più vicina posizione dell'etichetta tic che include tutti i dati del grafico. Le keyword **fixmin**, **fixmax**, **fix** o **noextend** dicono a gnuplot di disabilitare l'estensione dell'intervallo dell' asse alla prossima posizione del segno tic. In questo caso il limite dell'intervallo dell'asse corrisponde esattamente alla coordinata del data point più estremo. **set autoscale noextend** è un sinonimo di **set autoscale fix**. L'estensione dell'intervallo per un singolo asse può essere disabilitato aggiungendo la keyword **noextend** al corrispondente comando di intervallo, ad esempio

```
set yrange [0:*] noextend
```

set autoscale keepfix autoscala tutti gli assi lasciando le impostazioni invariate.

Esempi

Esempi:

Questo imposta l'autoscaling dell'asse y (gli altri assi non sono interessati):

```
set autoscale y
```

Questo imposta l'autoscaling solo per il minimo dell'asse y (il massimo dell'asse y e gli altri assi non sono interessati):

```
set autoscale ymin
```

Questo disabilita l'estensione dei tic dell'asse x2 al successivo segno tic, mantenendo così l'intervallo esatto come si trova nei dati e nelle funzioni plottati:

```
set autoscale x2fixmin
set autoscale x2fixmax
```

Questo imposta l'autoscaling degli assi x e y:

```
set autoscale xy
```

Questo imposta l'autoscaling degli assi x, y, z, x2 e y2:

```
set autoscale
```

Questo disabilita l'autoscaling degli assi x, y, z, x2 e y2:

```
unset autoscale
```

Questo disabilita l'autoscaling solo dell'asse z:

```
unset autoscale z
```

Modalità polare

Quando si è in modalità polare (**set polar**), l'intervallo xrange e l'intervallo yrange possono essere lasciati in modalità autoscale. Se **set rrange** è usato per limitare l'estensione dell'asse polare, allora xrange e yrange si regoleranno per corrispondere automaticamente a questo. Tuttavia, i comandi espliciti di xrange e yrange possono essere usati in seguito per fare ulteriori aggiustamenti. Vedere **set rrange (p. 204)**.

Vedere anche `polar demos`.

Bind

show bind mostra lo stato attuale di tutte le associazioni di tasti di scelta rapida. Vedere **bind (p. 54)**.

Bmargin

Il comando **set bmargin** imposta la dimensione del margine inferiore. Vedere **set margin (p. 176)** per i dettagli.

Border

I comandi **set border** e **unset border** controllano la visualizzazione dei bordi del grafico (graph) per i comandi **plot** e **splot**. Notare che i bordi non coincidono necessariamente con gli assi; con **plot** lo fanno spesso, ma con **splot** di solito no.

Sintassi:

```
set border {<integer>}
           {front | back | behind}
           {linestyle | ls <line_style>}
           {linetype | lt <line_type>} {linewidth | lw <line_width>}
           {linecolor | lc <colorspec>} {dashtype | dt <dashtype>}
           {polar}
unset border
show border
```

Con uno **splot** visualizzato in un orientamento arbitrario, come **set view 56,103**, i quattro angoli del piano x-y possono essere indicati come "front", "back", "left" e "right". Naturalmente esiste un insieme simile di quattro angoli per la superficie superiore. Così il bordo che collega, ad esempio, gli angoli posteriore e destro del piano x-y è il bordo "inferiore destro posteriore", e il bordo che collega gli angoli frontali superiore e

inferiore è il "verticale frontale". (Questa nomenclatura è definita unicamente per permettere al lettore di capire la tabella che segue).

I bordi sono codificati in un intero a 12 bit: i quattro bit bassi controllano il bordo per **plot** e i lati della base per **splot**; i successivi quattro bit controllano le verticali in **splot**; i quattro bit alti controllano i margini sulla cima di un **splot**. Le impostazioni dei bordi sono quindi la somma delle appropriate voci della seguente tabella:

Graph Border Encoding		
Bit	plot	splot
1	inferiore	inferiore sinistro frontale
2	sinistro	inferiore sinistro posteriore
4	superiore	inferiore destro frontale
8	destro	inferiore destro posteriore
16	nessun effetto	sinistro verticale
32	nessun effetto	posteriore verticale
64	nessun effetto	destro verticale
128	nessun effetto	frontale verticale
256	nessun effetto	superiore sinistro posteriore
512	nessun effetto	superiore destro posteriore
1024	nessun effetto	superiore sinistro frontale
2048	nessun effetto	superiore destro frontale
4096	polare	nessun effetto

L'impostazione predefinita è 31, che è tutti e quattro i lati per **plot**, e la base e l'asse z per **splot**.

Nei grafici 2D il bordo è normalmente disegnato sopra a tutti gli elementi del grafico (**front**). Se si desidera che il bordo sia disegnato dietro gli elementi del grafico, bisogna usare **set border back**.

Nei grafici hidden3d le linee che compongono il bordo sono normalmente soggette alla stessa elaborazione hidden3d degli elementi del grafico. **set border behind** sovrascrive questa impostazione predefinita.

Usando gli specificatori opzionali <linestyle>, <linetype>, <linewidth>, <linecolor>, and <dashtype>, il modo in cui le linee del bordo sono disegnate può essere influenzato (limitato da ciò che il driver del terminale corrente supporta). Oltre al bordo stesso, questo stile di linea è usato per i tic, indipendentemente dal fatto che siano plottati sul bordo o sugli assi (vedere **set xtics (p. 228)**).

Per **plot**, i tic possono essere disegnati sui margini diversi dall'inferiore e dal sinistro abilitando il secondo asse – vedere **set xtics (p. 228)** for details.

Se uno **splot** disegna solo sulla base, come nel caso di **"unset surface; set contour base"**, allora le verticali e la parte superiore non vengono disegnate anche se sono specificate.

Le opzioni **set grid** 'back', 'front' e 'layerdefault' controllano anche l'ordine in cui le linee di confine sono disegnate rispetto l'output dei dati plottati.

La keyword **polar** permette un bordo circolare per i grafici polari.

Esempi:

Disegna i bordi predefiniti:

```
set border
```

Disegna solo i bordi sinistro e inferiore (**plot**) o entrambi i bordi frontale e posteriore inferiori a sinistra (**splot**):

```
set border 3
```

Disegna un riquadro (box) completo intorno a un **splot**:

```
set border 4095
```

Disegna un riquadro senza parte superiore attorno a un grafico **splot**, omettendo la parte frontale verticale:

```
set border 127+256+512 # or set border 1023-128
```

Disegna solo i bordi superiore e destro per un grafico **plot** e li etichetta come assi:

```
unset xtics; unset ytics; set x2tics; set y2tics; set border 12
```

Boxwidth

Il comando **set boxwidth** è usato per impostare la larghezza predefinita dei box negli stili **boxes**, **boxerrorbars**, **boxplot**, **candlesticks** e **histograms**.

Sintassi:

```
set boxwidth {<width>} {absolute|relative}
show boxwidth
```

Per default, i box adiacenti sono estesi in larghezza finché non si toccano. Una diversa larghezza predefinita può essere specificata usando il comando **set boxwidth**. Le larghezze **relative** sono interpretate come una frazione di questa larghezza predefinita.

Un valore esplicito per la larghezza del box (boxwidth) è interpretato come un numero di unità lungo l'asse x corrente (assoluto **absolute**) a meno che non venga dato il modificatore **relative**. Se l'asse x è una scala logaritmica (vedere **set log (p. 175)**), allora il valore della larghezza del box è veramente "assoluto" solo a x=1; questa larghezza fisica è mantenuta ovunque lungo l'asse (cioè i box non diventano più stretti all'aumentare del valore di x). Se l'intervallo attraversato da un asse x in scala logaritmica è lontano da x=1, può essere necessaria qualche sperimentazione per trovare un valore utile della larghezza del box.

L'impostazione predefinita è sostituita da informazioni esplicite sulla larghezza prese da una colonna di dati extra nello stile **boxes** o **boxerrorbars**. In un set di dati a quattro colonne, la quarta colonna sarà interpretata come la larghezza del box a meno che la larghezza non sia impostata a -2.0, nel qual caso la larghezza sarà calcolata automaticamente. Vedere **style boxes (p. 64)** e **style boxerrorbars (p. 64)** per ulteriori dettagli.

Per impostare la larghezza del box su automatico usare il comando

```
set boxwidth
```

o, per dati a quattro colonne,

```
set boxwidth -2
```

Lo stesso effetto può essere ottenuto con la keyword **using** in **plot**.:

```
plot 'file' using 1:2:3:4:(-2)
```

Per impostare la larghezza del box a metà della dimensione automatica usare

```
set boxwidth 0.5 relative
```

Per impostare la larghezza del box a un valore assoluto di 2, usare

```
set boxwidth 2 absolute
```

Boxdepth

Il comando **set boxdepth** influisce solo sui grafici 3D creati da **splot with boxes**. Imposta l'estensione di ogni box lungo l'asse y, cioè il suo spessore.

Color

Gnuplot supporta due set alternativi di tipi di linea. Il set predefinito utilizza un colore diverso per ogni tipo di linea, sebbene permetta anche di disegnare linee punteggiate (dotted) o tratteggiate (dashed) in quel colore. Il set alternativo monocromatico usa solo il pattern di punto/linea o lo spessore delle linee per distinguere i tipi di linea. Il comando **set color** seleziona i tipi di linea a colori. Vedere **set monochrome (p. 177)**, **set linetype (p. 173)**, e **set colorsequence (p. 143)**.

Colorsequence

Sintassi:

```
set colorsequence {default|classic|podo}
```

set colorsequence default seleziona una sequenza ripetuta indipendente dal terminale di otto colori. Vedere **set linetype (p. 173)**, **colors (p. 50)**.

set colorsequence classic permette ad ogni tipo di terminale diverso di fornire la propria sequenza di colori di linea. Il numero fornito varia da 4 a più di 100, ma la maggior parte inizia con rosso/verde/blu/magenta/ciano/giallo. Questo era il comportamento predefinito prima della versione 5.

set colorsequence podo seleziona otto colori tratti da un set raccomandato da Wong (2011) [Nature Methods 8:441] come facilmente distinguibili dai daltonici con protanopia o deuteranopia.

In ogni caso è possibile personalizzare ulteriormente la lunghezza della sequenza e i colori utilizzati. Vedere **set linetype (p. 173)**, **colors (p. 50)**.

Clabel

Questo comando è deprecato. Usare invece **set cntrlabel**. **unset clabel** è sostituito da **set cntrlabel onecolor**. **set clabel "format"** è sostituito da **set cntrlabel format "format"**.

Clip

Sintassi:

```
set clip {points|one|two|radial}
unset clip {points|one|two|radial}
show clip
```

Stato di default:

```
unset clip points
set clip one
unset clip two
unset clip radial
```

I data point il cui centro si trova all'interno dei confini del grafico, vengono disegnati normalmente anche se la dimensione finita del simbolo punto lo fa estendere oltre una linea di confine. **set clip points** fa sì che tali punti siano tagliati (cioè non disegnati) anche se il centro del punto si trova all'interno dei confini di un grafico 2D. I data point il cui centro si trova al di fuori dei confini del grafico non vengono mai disegnati.

unset clip fa sì che un segmento di linea in un grafico non venga disegnato se una delle estremità di quel segmento si trova al di fuori dei confini del grafico (cioè xrange e yrange).

set clip one fa sì che **gnuplot** disegni la porzione all'interno dell'intervallo dei segmenti di linea con un punto finale nell'intervallo e un punto finale fuori dall'intervallo. **set clip two** fa sì che **gnuplot** disegni la porzione all'interno dell'intervallo dei segmenti di linea con entrambi i punti finali fuori dall'intervallo. I segmenti di linea che si trovano completamente al di fuori dei confini del grafico non vengono mai disegnati.

set clip radial influisce solo sul plotting in modalità polare. Taglia le linee rispetto al limite radiale stabilito da **set rrange [0:MAX]**. Questo criterio viene applicato insieme a **set clip {one|two}**. Cioè la porzione di una linea tra due punti con R > RMAX che passa attraverso il cerchio R = RMAX viene disegnata solo se sono impostati sia **clip two** che **clip radial**.

Note:

* **set clip** influenza solo i punti e le linee prodotte dagli stili di grafico **lines**, **linespoints**, **points**, **arrows**, e **vectors**.

* Il ritaglio di quadrangoli colorati disegnati per superfici pm3d e altri oggetti solidi è controllato da **set pm3d clipping**. L'impostazione predefinita è il ritaglio uniforme contro lo zrange corrente.

* Il ritaglio dell'oggetto è controllato dalla proprietà **clip** o **noclip** dell'oggetto individuale.

* Nella versione corrente di gnuplot, "plot with vectors" in modalità polare non verifica o ritaglia rispetto al raggio massimo.

Cntrlabel

Sintassi:

```
set cntrlabel {format "format"} {font "font"}
set cntrlabel {start <int>} {interval <int>}
set contrlabel onecolor
```

set cntrlabel controlla l'etichettatura dei contorni, o nella key (default) o sul grafico stesso nel caso di **splot ... with labels**. In quest'ultimo caso le etichette sono posizionate lungo ogni linea di contorno secondo la proprietà **pointinterval** o **pointnumber** del descrittore di etichette. Per impostazione predefinita un'etichetta è posta sul 5° segmento di linea che compone la linea di contorno ed è ripetuta ogni 20° segmento. Queste impostazioni predefinite sono equivalenti a

```
set cntrlabel start 5 interval 20
```

Possono essere cambiate sia tramite il comando **set cntrlabel** che specificando l'intervallo nel comando **splot** stesso

```
set contours; splot $FOO with labels point pointinterval -1
```

Impostare l'intervallo su un valore negativo significa che l'etichetta appare solo una volta per ogni linea di contorno. Tuttavia se **set samples** o **set isosamples** è grande allora possono essere create molte linee di contorno, ognuna con una singola etichetta.

Un'etichetta di contorno è posta nella key del grafico per ogni tipo di linea utilizzato. Per impostazione predefinita ad ogni livello di contorno viene dato il proprio tipo di linea, quindi appare un'etichetta separata per ciascuno. Il comando **set cntrlabel onecolor** fa sì che tutti i contorni siano disegnati utilizzando lo stesso tipo di linea, in modo da far apparire una sola etichetta nella key del grafico. Questo comando sostituisce un vecchio comando **unset clabel**.

Cntrparam

set cntrparam controlla la generazione dei contorni e la loro scorrevolezza per un grafico a contorni. **show contour** mostra le impostazioni correnti di **cntrparam** e di **contour**.

Sintassi:

```
set cntrparam { { linear
                | cubicspline
                | bspline
                | points <n>
                | order <n>
                | levels { <n>
                           | auto {<n>}
                           | discrete <z1> {,<z2>{,<z3>...}}
                           | incremental <start>, <incr> {,<end>}
                         }
                  {{un}sorted}
                  {firstlinetype N}
                }
              }
show contour
```

Questo comando ha due funzioni. In primo luogo, imposta i valori di z per i quali i contorni devono essere determinati. Il numero di livelli di contorno <n> dovrebbe essere un'espressione costante integrale. <z1>, <z2> ... sono espressioni di valore reale. In secondo luogo, controlla l'aspetto delle singole linee di contorno.

Le keyword che controllano la scorrevolezza delle linee di contorno:

linear, **cubicspline**, **bspline** — Controlla il tipo di approssimazione o interpolazione. Se è **linear**, allora segmenti di linea retta collegano punti di uguale grandezza z. Se **cubicspline**, allora i contorni lineari a tratti sono interpolati tra gli stessi punti z uguali per formare dei contorni un po' più regolari, ma che possono ondulare. Se **bspline**, viene disegnata una curva più liscia garantita, che approssima solo la posizione dei punti di z uguale.

points — In definitiva tutti i disegni sono fatti con tratti lineari. Questo numero controlla il numero di segmenti di linea usati per approssimare la curva **bspline** o **cubicspline**. Numero di segmenti (tratti) cubicspline o bspline = **points** * numero di segmenti lineari.

order — Ordine dell'approssimazione bspline da utilizzare. Più grande è questo ordine, più regolare è il contorno risultante. (Naturalmente, le curve bspline di ordine superiore si allontaneranno di più dai dati lineari a tratti originali). Questa opzione è rilevante solo per la modalità **bspline**. I valori consentiti sono interi nell'intervallo da 2 (lineare) a 10.

Keyword che controllano la selezione dei livelli di contorno:

levels auto — Questo è il default. <n> specifica un numero nominale di livelli; il numero effettivo sarà regolato per dare etichette semplici. Se la superficie è delimitata da zmin e zmax, i contorni saranno generati a multipli interi di dz tra zmin e zmax, dove dz è 1, 2, o 5 volte una potenza di dieci (come il salto tra due segni tic).

levels discrete — I contorni saranno generati a z = <z1>, <z2> ... come specificato; il numero di livelli discreti stabilisce il numero di livelli di contorno. In modalità **discrete** (discreta), qualsiasi **set cntrparam levels <n>** viene ignorato.

levels incremental — I contorni sono generati a valori di z a partire da <start> e aumentando di <increment>, fino a raggiungere il numero di contorni. <end> è usato per determinare il numero di livelli di contorno, che sarà cambiato da qualsiasi seguente **set cntrparam levels <n>**. Se l'asse z è logaritmico, <increment> sarà interpretato come un fattore moltiplicativo, come per **set ztics**, e <end> non dovrebbe essere usato.

Keyword che controllano l'assegnazione del tipo di linea ai contorni:

Per default i contorni sono generati nell'ordine inverso specificato (**unsorted**). Quindi **set cntrparam levels increment 0, 10, 100** creerà 11 livelli di contorni che iniziano con 100 e finiscono con 0. Aggiungendo la keyword **sorted** i contorni vengono riordinati per valore numerico crescente, che in questo caso significherebbe che il primo contorno viene disegnato a 0.

Per default i contorni sono disegnati usando tipi di linea successivi a partire dal tipo di linea seguente a quello utilizzato per la superficie corrispondente. Così **splot x*y lt 5** userebbe lt 6 per il primo contorno generato. Se la modalità **hidden3d** è attiva, allora ogni superficie usa due tipi di linea. In questo caso usare le impostazioni predefinite farebbe sì che il primo contorno usi lo stesso tipo di linea della superficie nascosta, il che è indesiderabile. Questo può essere evitato in uno dei seguenti due modi. (1) Usare **set hidden3d offset N** per cambiare il tipo di linea usato per la superficie nascosta. Una buona scelta può essere **offset -1** poiché eviterà tutti i tipi di linee di contorno. (2) Usare l'opzione **set cntrparam firstlinetype N** per specificare un blocco di tipi di linea usati per le linee di contorno indipendentemente da quello usato per la superficie. Questo è particolarmente utile se si vuole personalizzare l'insieme dei tipi di linee di contorno. N <= 0 ripristina il default.

Se il comando **set cntrparam** è dato senza alcun argomento specificato tutte le opzioni vengono riportate al valore predefinito:

```
set cntrparam order 4 points 5
set cntrparam levels auto 5 unsorted
set cntrparam firstlinetype 0
```

Esempi

Esempi:

```
set cntrparam bspline
set cntrparam points 7
set cntrparam order 10
```

Per selezionare automaticamente i livelli, 5 se i criteri di incremento del livello sono soddisfatti:

```
set cntrparam levels auto 5
```

Per specificare livelli discreti a .1, .37 e .9:

```
set cntrparam levels discrete .1,1/exp(1),.9
```

Per specificare i livelli da 0 a 4 con incremento 1:

```
set cntrparam levels incremental  0,1,4
```

Per impostare il numero di livelli a 10 (cambiando una fine incrementale o eventualmente il numero di livelli automatici):

```
set cntrparam levels 10
```

Per impostare l'inizio e l'incremento mantenendo il numero di livelli:

```
set cntrparam levels incremental 100,50
```

Per definire e utilizzare un blocco personalizzato di tipi di linee di contorno

```
set linetype 100 lc "red" dt '....'
do for [L=101:199] {
    if (L%10 == 0) {
        set linetype L lc "black" dt solid lw 2
    } else {
        set linetype L lc "gray" dt solid lw 1
    }
}
set cntrparam firstlinetype 100
set cntrparam sorted levels incremental 0, 1, 100
```

Vedere anche **set contour (p. 147)** per il controllo su dove vengono disegnati i contorni, e **set cntrlabel (p. 144)** per il controllo del formato delle etichette dei contorni e dei tipi di linea.

Vedere anche `contours demo (contours.dem)`

e `contours with user defined levels demo (discrete.dem).`

Color box

Lo schema dei colori, cioè il gradiente del colore uniforme con i valori min_z e max_z della **palette** di **pm3d**, è disegnato in un color box, a meno che **unset colorbox**.

```
set colorbox
set colorbox {
           { vertical | horizontal } {{no}invert}
           { default | user }
           { origin x, y }
           { size x, y }
           { front | back }
           { noborder | bdefault | border [line style] }
         }
show colorbox
unset colorbox
```

La posizione del color box può essere **default** o **user**. Se quest'ultimo è specificato i valori dati con i sottocomandi **origin** e **size** sono usati. Il box può essere disegnato dopo (**front**) o prima (**back**) del grafico (graph) o della superficie.

L'orientamento del gradiente di colore può essere cambiato dalle opzioni **vertical** e **horizontal**.

origin x, y e **size x, y** sono usati solo in combinazione con l'opzione **user**. I valori x e y sono interpretati per default come coordinate dello schermo, e questa è l'unica opzione legale per i grafici 3D. I grafici 2D, incluso lo splot con **set view map**, permettono di specificare qualsiasi sistema di coordinate. Provare per esempio:

```
set colorbox horiz user origin .1,.02 size .8,.04
```

che disegnerà un gradiente orizzontale da qualche parte nella parte inferiore del grafico (graph).

border attiva il bordo (questo è il default). **noborder** disattiva il bordo. Se viene dato un argomento intero positivo dopo **border**, esso è usato come tag stile linea che viene usato per disegnare il bordo, ad es:

```
set style line 2604 linetype -1 linewidth .4
set colorbox border 2604
```

userà lo stile di linea **2604**, una linea sottile con il colore predefinito del bordo (-1) per disegnare il bordo. **bdefault** (che è il default) userà lo stile di linea predefinito per disegnare il bordo della color box.

L'asse della color box è chiamato **cb** ed è controllato per mezzo dei soliti comandi degli assi, cioè **set/unset/show** con **cbrange**, **[m]cbtics**, **format cb**, **grid [m]cb**, **cblabel**, e forse anche **cbdata**, **[no]cbdtics**, **[no]cbmtics**.

set colorbox senza alcun parametro cambia la posizione al default. **unset colorbox** reimposta i parametri predefiniti per il colorbox e disattiva il colorbox.

Vedere anche aiuto per **set pm3d (p. 197)**, **set palette (p. 190)**, **x11 pm3d (p. 316)**, e **set style line (p. 210)**.

Colornames

Gnuplot conosce un numero limitato di nomi di colori. È possibile utilizzarli per definire la gamma di colori di una palette pm3d, o per assegnare un colore indipendente dal terminale ad un particolare tipo di linea o stile di linea. Per vedere la lista dei nomi di colori conosciuti, usare il comando **show colornames (p. 147)**. Esempio:

```
set style line 1 linecolor "sea-green"
```

Contour

set contour abilita il disegno dei contorni per le superfici. Questa opzione è disponibile solo per **splot**. Richiede griglie di dati, vedere **grid_data (p. 241)** per maggiori dettagli. Se i contorni sono richiesti da dati non a griglia, **set dgrid3d** può essere usato per creare una griglia appropriata.

Sintassi:

```
set contour {base | surface | both}
unset contour
show contour
```

Le tre opzioni specificano dove disegnare i contorni: **base** disegna i contorni sulla base della griglia dove sono posizionati gli x/ytic, **surface** disegna i contorni sulle contorni sulle superfici stesse, e **both** disegna i contorni sia sulla base che la superficie. Se non viene fornita alcuna opzione, il default è **base**.

Vedere anche **set cntrparam (p. 144)** per i parametri che influenzano il disegno dei contorni, e **set cntrlabel (p. 144)** per il controllo dell'etichettatura dei contorni.

La superficie può essere disattivata (vedere **unset surface (p. 214)**), dando un grafico (graph) di solo contorno. Sebbene sia possibile usare **set size** per ingrandire il grafico fino a riempire lo schermo, si può ottenere un maggiore controllo sul formato di output scrivendo le informazioni sui contorni in un datablock, e rileggendole come un grafico di un file di dati 2D:

```
unset surface
set contour
set cntrparam ...
set table $datablock
splot ...
unset table
# info contorno ora in $datablock
set term <whatever>
plot $datablock
```

Per disegnare i contorni, i dati devono essere organizzati come "grid data" (griglie di dati). In tale file sono elencati tutti i punti per una singola y-isolinea, poi tutti i punti per la y-isolinea successiva, e così via. Una singola linea vuota (una linea che non contenga caratteri diversi da spazi vuoti e da un ritorno a capo e/o un avanzamento di riga) separa una y-isolinea dalla successiva.

Mentre **set contour** è in vigore, **splot with** <**style**> posizionerà gli gli elementi di stile (punti, linee, impulsi, etichette, ecc.) lungo le linee di contorno. **with pm3d** produrrà una superficie pm3d e anche le linee di contorno. Se si vogliono mescolare altri elementi del grafico, ad esempio etichette lette da un file, con i contorni generati mentre **set contour** è attivo, si deve aggiungere la keyword **nocontours** dopo quella clausola nel comando splot.

Vedere anche **splot datafile (p. 238)**.

Vedere anche `contours demo (contours.dem)`

e `contours with user defined levels demo (discrete.dem).`

Dashtype

Il comando **set dashtype** permette di definire un dash (trattino) pattern che può essere richiamato dal suo indice. Questa è puramente una comodità, poiché ovunque venisse accettato il dashtype per il suo indice numerico verrebbe accettato anche un dash pattern esplicito. Esempio:

```
set dashtype 5 (2,4,2,6)   # definisci o ridefinisci dashtype numero 5
plot f1(x) dt 5            # plotta usando il nuovo dashtype
plot f1(x) dt (2,4,2,6)    # esattamente lo stesso grafico di cui sopra
set linetype 5 dt 5        # usa sempre questo dash pattern con linetype 5
set dashtype 66 "..-"      # definisci un nuovo dashtype usando una stringa
```

Vedere anche **dashtype (p. 52)**.

Data style

Questa forma del comando è deprecata. Vedere **set style data (p. 208)**.

Datafile

Le opzioni di comando **set datafile** controllano l'interpretazione dei campi letti da file di dati di input dai comandi **plot**, **splot** e **fit**. Diverse opzioni sono attualmente implementate.

Set datafile columnheaders

Il comando **set datafile columnheaders** garantisce che la prima riga di input sarà interpretata come intestazioni di colonna piuttosto che come valori di dati. Influisce su tutte le fonti di dati in input per i comandi plot, splot, fit e stats. Se questa impostazione è disabilitata da **unset datafile columnheaders**, lo stesso effetto viene attivato su una base per file se c'è un'esplicita funzione columnheader() in uno specificatore d'uso o in un titolo di grafico associato a quel file. Vedere anche **set key autotitle (p. 168)** e **columnheader (p. 129)**.

Set datafile fortran

Il comando **set datafile fortran** abilita un controllo speciale per i valori nel file di input espressi come costanti Fortran D o Q. Questo controllo extra rallenta il processo di input e dovrebbe essere selezionato solo se si hanno effettivamente file di dati contenenti costanti Fortran D o Q. L'opzione può essere disabilitata di nuovo usando **unset datafile fortran**.

Set datafile nofpe_trap

Il comando **set datafile nofpe_trap** dice a gnuplot di non reinizializzare un gestore di eccezioni in virgola mobile prima di ogni valutazione di espressione usata durante la lettura dei dati da un file di input. Questo può velocizzare significativamente l'input di dati da file molto grandi, con il rischio di terminare il programma se viene generata un'eccezione in virgola mobile.

Set datafile missing

Sintassi:

```
set datafile missing "<string>"
set datafile missing NaN
show datafile missing
unset datafile
```

Il comando **set datafile missing** dice a **gnuplot** che è presente una stringa speciale usata nei file di dati di input per indicare una entry di dati mancante. Non esiste un carattere predefinito per **missing**. Gnuplot fa una distinzione tra dati mancanti e dati non validi (ad es. "NaN", 1/0.). Per esempio i dati non validi creano uno spazio vuoto in una linea disegnata attraverso punti di dati sequenziali; i dati mancanti no.

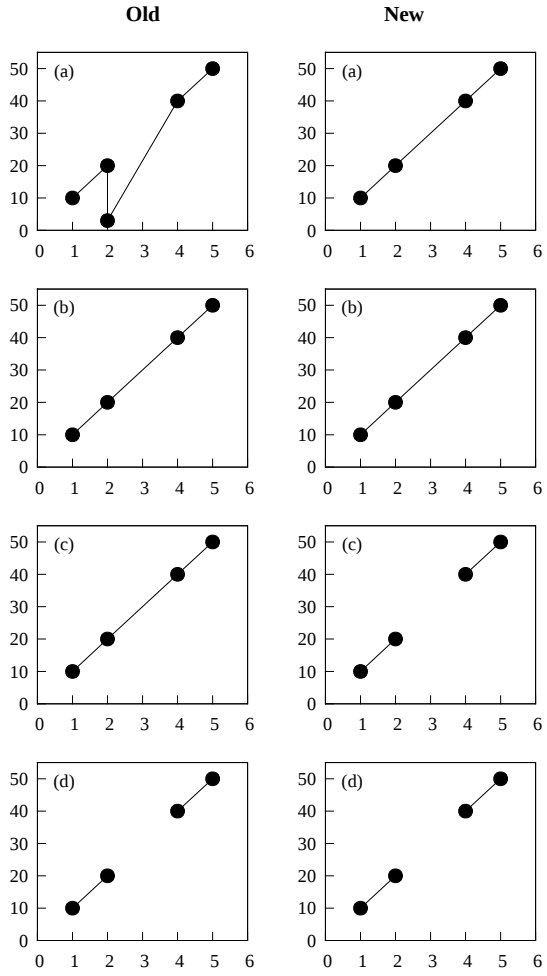
I caratteri non numerici trovati in un campo numerico saranno solitamente interpretati come non validi piuttosto che come un data point mancante, a meno che non coincidano con la stringa mancante (**missing**).

Al contrario, **set datafile missing NaN** fa sì che tutti i dati o le espressioni che valutano un non-numero (NaN) siano trattati come dati mancanti.

L'esempio qui sotto mostra le differenze tra le versioni di gnuplot 4 e 5.

Esempio:

```
set style data linespoints
plot '-' title "(a)"
   1 10
   2 20
   3 ?
   4 40
   5 50
   e
set datafile missing "?"
plot '-' title "(b)"
   1 10
   2 20
   3 ?
   4 40
   5 50
   e
plot '-' using 1:2 title "(c)"
   1 10
   2 20
   3 NaN
   4 40
   5 50
   e
plot '-' using 1:($2) title "(d)"
   1 10
   2 20
   3 NaN
   4 40
   5 50
   e
```



Il grafico (a) è diverso in gnuplot 4 e gnuplot 5 perché la terza linea contiene solo un numero valido. La versione 4 è passata a una convenzione di un singolo dato su una linea per cui il numero di linea è "x" e il dato è "y", posizionando erroneamente il punto a(2,3).

Sia la vecchia che la nuova versione di gnuplot gestiscono gli stessi dati correttamente se il carattere '?' è designato come un marcatore per i dati mancanti (b).

Le vecchie versioni di gnuplot gestivano NaN in modo diverso a seconda della forma della clausola **using**, come mostrato nei grafici (c) e (d). Gnuplot ora gestisce NaN allo stesso modo stesso sia che la colonna di input sia stata specificata come N o ($N). Vedere anche `demo imageNaN`.

Allo stesso modo gnuplot 5.4 si accorgerà del flag di valore mancante nella colonna N sia che il comando plot specifichi **using N** o **using ($N)** o **using (func($N))**. Tuttavia, se il valore "mancante" viene incontrato durante la valutazione di qualche espressione più complicata, per esempio **using (column(strcol(1))**, potrebbe valutare come NaN ed essere trattato come un dato non valido piuttosto che come un data point mancante. Se si vuole comunque trattarli come dati mancanti, usare il comando **set datafile missing NaN**.

Set datafile separator

Il comando **set datafile separator** dice a **gnuplot** che i campi di dati nei successivi file di input sono separati da un carattere specifico piuttosto che da spazi bianchi. L'uso più comune è quello di leggere file csv (comma-separated value, valore separato da virgole) scritti da fogli di calcolo o programmi di database. Per default i campi di dati sono separati da spazi bianchi.

Sintassi:

```
set datafile separator {whitespace | tab | comma | "<chars>"}
```

Esempi:

```
# Il file di input contiene campi separati da tab
set datafile separator "\t"

# Il file di input contiene campi con valori separati da virgola
set datafile separator comma

# Il file di input contiene campi separati da * o |
set datafile separator "*|"
```

Set datafile commentschars

Il comando **set datafile commentschars** specifica quali caratteri possono essere usati in un file di dati per iniziare le linee di commento. Se il primo carattere non vuoto su una linea è uno di questi caratteri allora il resto della linea di dati viene ignorato. Il valore predefinito della stringa è "#!" su VMS e altrimenti "#".

Sintassi:

```
set datafile commentschars {"<string>"}
show datafile commentschars
unset commentschars
```

Quindi, la seguente linea in un file di dati è completamente ignorata

```
# 1 2 3 4
```

ma la seguente

```
1 # 3 4
```

sarà interpretata come spazzatura nella 2a colonna seguita da dati validi nella nella 3a e 4a colonna.

Esempio:

```
set datafile commentschars "#!%"
```

Set datafile binary

Il comando **set datafile binary** è usato per impostare i valori predefiniti quando si leggono dei file di dati binari. La sintassi corrisponde esattamente a quella usata per i comandi **plot** e **splot**. Vedere **binary matrix (p. 238)** e **binary general (p. 109)** per i dettagli sulle keyword che possono essere presenti in <binary list>.

Sintassi:

```
set datafile binary <binary list>
show datafile binary
show datafile
unset datafile
```

Esempi:

```
set datafile binary filetype=auto
set datafile binary array=(512,512) format="%uchar"

show datafile binary   # elenca le impostazioni attuali
```

Decimalsign

Il comando **set decimalsign** seleziona un segno decimale per i numeri stampati nelle etichette dei tic o nelle stringhe **set label**.

Sintassi:

```
set decimalsign {<value> | locale {"<locale>"}}
unset decimalsign
show decimalsign
```

L'argomento <value> è una stringa da usare al posto del solito punto decimale. Le scelte tipiche includono il punto, '.', e la virgola, ',', ma anche altre possono essere utili. Se si omette l'argomento <value>, il separatore decimale non viene modificato dal solito default, che è un punto. Disabilitare segni decimali (decimalsign) ha lo stesso effetto di omettere <value>.

Esempio:

La corretta impostazione (typesetting) nella maggior parte dei paesi europei richiede:

```
set decimalsign ','
```

Notare: Se si imposta una stringa esplicita, questa ha effetto solo sui numeri che vengono stampati usando la routine di formattazione gprintf() di gnuplot, inclusi i tic degli assi. Non influisce sul formato previsto per i dati di input, e non influisce sui i numeri stampati con la routine di formattazione sprintf(). Per cambiare il comportamento della formattazione sia in input che in output, usare invece la forma

```
set decimalsign locale
```

Questo ordina al programma di usare entrambi i formati di input e output in accordo con l'impostazione corrente delle variabili d'ambiente LC_ALL, LC_NUMERIC o LANG.

```
set decimalsign locale "foo"
```

Questo ordina al programma di formattare tutti gli input e gli output in accordo con locale "foo", che deve essere installato. Se il locale "foo" non viene trovato, allore viene stampato un messaggio di errore e l'impostazione del segno decimale rimane invariata. Sui sistemi linux è possibile ottenere una lista dei locali installati sulla propria macchina digitando "locale -a". Una tipica stringa locale linux è della forma "sl_SI.UTF-8". Una tipica stringa locale per Windows è della forma "Slovenian_Slovenia.1250" o "slovenian". Notare che l'interpretazione delle impostazioni locali è fatta dalla libreria C in fase di esecuzione. Le vecchie librerie C possono offrire solo un supporto parziale per impostazioni locali come il carattere separatore di raggruppamento delle migliaia.

```
set decimalsign locale; set decimalsign "."
```

Questo imposta tutti gli input e gli output per usare qualsiasi segno decimale che sia corretto per il locale corrente, ma lo sovrascrive con un '.' esplicito nei numeri formattati usando la funzione interna gprintf() di gnuplot.

Dgrid3d

Il comando **set dgrid3d** abilita, e può impostare parametri per, la mappatura dei dati da non a grilgia ad a griglia. Vedere **splot grid_data (p. 241)** per maggiori dettagli sulla struttura dei dati a griglia.

Sintassi:

```
set dgrid3d {<rows>} {,{<cols>}}
            { splines |
              qnorm {<norm>} |
              (gauss | cauchy | exp | box | hann)
                {kdensity} {<dx>} {,<dy>} }
unset dgrid3d
show dgrid3d
```

Per default **dgrid3d** è disabilitato. Quando è abilitato, i dati 3D letti da un file sono sempre trattati come un insieme di dati sparsi. Una griglia con dimensioni derivate da un riquadro di delimitazione dei dati sparsi e la dimensione specificata dai parametri row/col_size viene creata per il plotting e il contouring. La griglia è equamente spaziata in x (righe) e in y (colonne); i valori z sono computati come medie ponderate o interpolazioni spline dei valori z dei punti sparsi. In altre parole, viene creata una griglia regolarmente spaziata e viene valutata un'approssimazione regolare ai dati grezzi per tutti i punti della griglia. Questa approssimazione viene plottata al posto dei dati grezzi.

Il numero di colonne corrisponde al numero di righe, che per default è 10.

Sono disponibili diversi algoritmi per calcolare l'approssimazione dai dati grezzi. Alcuni di questi algoritmi possono prendere parametri aggiuntivi. Queste interpolazioni sono tali che più il data point è vicino a un punto della griglia, più effetto ha su quel punto della griglia.

L'algoritmo **splines** calcola un'interpolazione basata su "thin plate splines" (splines a piastra sottile). Non prende parametri aggiuntivi.

L'algoritmo **qnorm** calcola una media ponderata dei dati di input in ogni punto della griglia. Ogni data point è ponderato dall'inverso della sua distanza dal punto della griglia elevato ad una certa potenza. La potenza è specificata come un parametro intero opzionale che ha come valore predefinito 1. Questo algoritmo è quello predefinito.

Infine, sono disponibili diversi smoothing kernels per calcolare medie ponderate: z = Sum_i w(d_i) * z_i / Sum_i w(d_i), dove z_i è il valore dell'i-esimo data point e d_i è la distanza tra il corrente punto della griglia e la posizione dell'i-esimo data point. Tutti i kernel assegnano pesi maggiori ai data point che sono vicini al punto corrente della griglia e pesi più bassi ai data point più lontani.

Sono disponibili i seguenti kernel:

```
gauss :     w(d) = exp(-d*d)
cauchy :    w(d) = 1/(1 + d*d)
exp :       w(d) = exp(-d)
box :       w(d) = 1                    if d<1
                 = 0                    otherwise
hann :      w(d) = 0.5*(1+cos(pi*d))    if d<1
            w(d) = 0                    otherwise
```

Quando si usa uno di questi cinque smoothing kernel, si possono specificare fino a due parametri numerici aggiuntivi: dx e dy. Questi sono usati per ridimensionare le differenze di coordinate quando si calcola la distanza: d_i = sqrt(((x-x_i)/dx)**2 + ((y-y_i)/dy)**2), dove x,y sono le coordinate del corrente punto della griglia e x_i,y_i sono le coordinate dell'i-esimo data point. Il valore di dy è per default il valore di dx, che ha come valore predefinito 1. I parametri dx e dy permettono di controllare il raggio su cui i data point contribuiscono a un punto della griglia NELLE UNITÀ DEI DATI STESSI.

La keyword opzionale **kdensity**, che deve venire dopo il nome del kernel, ma prima dei parametri di scala (opzionali), modifica l'algoritmo in modo che i valori calcolati per i punti della griglia non siano divisi per la somma dei pesi (z = Sum_i w(d_i) * z_i). Se tutte le z_i sono costanti, questo effettivamente plotta una stima di densità del kernel bivariata: una funzione kernel (una delle cinque definite sopra) è posta in ogni data point, la somma di questi kernel viene valutata in ogni punto della griglia e questa superficie regolare (smooth) viene plottata al posto dei dati originali. Questo è simile in principio a quello che l'opzione **smooth kdensity** fa ai set di dati 1D. (Vedere kdensity2d.dem per dimostrazione d'uso)

Una sintassi leggermente diversa è anche supportata per ragioni di retrocompatibilità. Se nessun algoritmo di interpolazione è stato esplicitamente selezionato, si assume l'algoritmo **qnorm**. Si possono specificare fino a tre parametri separati da virgole, che sono interpretati rispettivamente come il numero di righe, il numero di colonne e il valore della norma.

L'opzione **dgrid3d** è uno schema semplice che sostituisce i dati sparsi con medie ponderate su una griglia regolare. Esistono approcci più sofisticati a questo problema e dovrebbero essere usati per preprocessare i dati al di fuori di **gnuplot** se questa semplice soluzione è ritenuta inadeguata.

Vedere anche `dgrid3d.dem: dgrid3d demo.`

e scatter.dem: dgrid3d demo.

Dummy

Il comando **set dummy** cambia i nomi predefiniti delle variabili dummy (fittizie).

Sintassi:

```
set dummy {<dummy-var>} {,<dummy-var>}
show dummy
```

Per default, **gnuplot** presuppone che la variabile indipendente, o "dummy", per il comando **plot** sia "t" se in modalità parametrica o polare, o altrimenti "x". Allo stesso modo le variabili indipendenti per il comando **splot** sono "u" e "v" in modalità parametrica (**splot** non può essere usato in modalità polare), o altrimenti "x" e "y".

Può essere più vantaggioso chiamare una variabile dummy con un nome più fisicamente significativo o convenzionale. Per esempio, quando si plottano le funzioni temporali:

```
set dummy t
plot sin(t), cos(t)
```

Esempi:

```
set dummy u,v
set dummy ,s
```

Il secondo esempio imposta la seconda variabile a s. Per riportare i nomi delle variabili dummy ai loro valori predefiniti, usare

```
unset dummy
```

Encoding

Il comando **set encoding** seleziona una codifica di caratteri.

Sintassi:

```
set encoding {<value>}
set encoding locale
show encoding
```

I valori validi sono

```
default     - dice a un terminale di usare la sua codifica predefinita
iso_8859_1  - la codifica più comune in Europa occidentale prima di UTF-8.
              Nota nel mondo PostScript come 'ISO-Latin1'.
iso_8859_15 - una variante di iso_8859_1 che include il simbolo Euro
iso_8859_2  - usato in Europa centrale e orientale
iso_8859_9  - usato in Turchia (noto anche come Latin5)
koi8r       - popolare codifica cirillica Unix
koi8u       - codifica cirillica Unix ucraina
cp437       - codepage per MS-DOS
cp850       - codepage per OS/2, Europa occidentale
cp852       - codepage per OS/2, Europa centrale e orientale
cp950       - Versione MS di Big5 (solo terminale emf)
cp1250      - codepage per MS Windows, Europa centrale e orientale
cp1251      - codepage per 8-bit russo, serbo, bulgaro, macedone
cp1252      - codepage per MS Windows, Europa occidentale
```

```
cp1254      - codepage per MS Windows, turco (superset di Latin5)
sjis        - codifica giapponese shift-JIS
utf8        - rappresentazione a lunghezza variabile (multibyte)
              del punto di entry Unicode per ogni carattere
```

Il comando **set encoding locale** è diverso dalle altre opzioni. Tenta di determinare il locale corrente dall'ambiente di esecuzione. Sulla maggior parte dei sistemi ciò è controllato dalle variabili d'ambiente LC_ALL, LC_CTYPE o LANG. Questo meccanismo è necessario, per esempio, per passare codifiche di caratteri multibyte come UTF-8 o EUC_JP ai terminali wxt e pdf. Questo comando non influisce sulla rappresentazione specifica del locale di date o numeri. Vedere anche **set locale (p. 175)** e **set decimalsign (p. 152)**.

Generalmente è necessario impostare la codifica prima di impostare il tipo di terminale, in quanto può influenzare la scelta dei font appropriati.

Errorbars

Il comando **set errorbars** controlla i tic alle estremità delle barre di errore, e anche alla fine alla fine dei baffi appartenenti a un boxplot (grafico a scatola e baffi).

Sintassi:

```
set errorbars {small | large | fullwidth | <size>} {front | back}
              {line-properties}
unset errorbars
show errorbars
```

small è sinonimo per 0.0 (no crossbar), e **large** per 1.0. Il default è 1.0 se non viene data una dimensione.

La keyword **fullwidth** è rilevante solo per i boxplot e per gli istogrammi con barre di errore. Imposta la larghezza delle estremità della barra di errore per essere uguale alla larghezza del box associato. Non cambia la larghezza del box stesso.

Le keyword **front** e **back** sono rilevanti solo per le barre di errore attaccate a rettangoli riempiti (scatole, candlesticks, istogrammi).

Le barre di errore sono disegnate per default usando le stesse proprietà di linea del bordo del box associato. È possibile cambiare ciò fornendo un insieme separato di proprietà di linea separate per le barre di errore.

```
set errorbars linecolor black linewidth 0.5 dashtype '.'
```

Fit

Il comando **set fit** controlla le opzioni per il comando **fit**.

Sintassi:

```
set fit {nolog | logfile {"<filename>"|default}}
        {{no}quiet|results|brief|verbose}
        {{no}errorvariables}
        {{no}covariancevariables}
        {{no}errorscaling}
        {{no}prescale}
        {maxiter <value>|default}
        {limit <epsilon>|default}
        {limit_abs <epsilon_abs>}
        {start-lambda <value>|default}
        {lambda-factor <value>|default}
        {script {"<command>"|default}}
```

```
        {v4 | v5}
unset fit
show fit
```

L'opzione **logfile** definisce dove il comando **fit** scrive il suo output. L'argomento <filename> deve essere racchiuso tra virgolette singole o doppie. Se non viene dato alcun filename o viene usato **unset fit**, il file log viene resettato al suo valore predefinito "fit.log" o al valore della variabile d'ambiente **FIT_LOG**. Se il nome del logfile dato termina con una / o \, viene interpretato come un nome di directory, e il filename effettivo sarà "fit.log" in quella directory.

Per default le informazioni scritte nel logfile sono anche riportate nella sessione del terminale. **set fit quiet** disattiva l'eco, mentre **results** stampa solo i risultati finali. **brief** dà un riepilogo di una linea per ogni iterazione del fit in aggiunta. **verbose** produce rapporti dettagliati sulle iterazioni come nella versione 4.

Se l'opzione **errorvariables** è attiva, l'errore di ogni parametro adattato (fitted) computato da **fit**, sarà copiato in una variabile definita dall'utente il cui nome è formato aggiungendo "_err" al nome del parametro stesso. Questo è utile soprattutto per mettere il parametro e il suo errore su un grafico dei dati e della funzione adattata, per riferimento, come in:

```
set fit errorvariables
fit f(x) 'datafile' using 1:2 via a, b
print "error of a is:", a_err
set label 1 sprintf("a=%6.2f +/- %6.2f", a, a_err)
plot 'datafile' using 1:2, f(x)
```

Se viene specificata l'opzione **errorscaling**, che è il default, gli errori dei parametri calcolati vengono scalati con il quadrato chi ridotto. Questo equivale a fornire errori di dati pari alla deviazione standard calcolata dell'adattamento (FIT_STDFIT) risultante in un quadrato chi ridotto di uno. Con l'opzione **noerrorscaling** gli errori stimati sono le deviazioni standard non scalate dei parametri di adattamento. Se non vengono specificati pesi per i dati, gli errori dei parametri sono sempre scalati.

Se l'opzione **prescale** è attiva, i parametri sono prescalati dai loro valori iniziali prima di essere passati alla routine di Marquardt-Levenberg. Questo aiuta notevolmente se ci sono parametri che differiscono di molti ordini di grandezza. I parametri di adattamento con un valore iniziale di esattamente zero non sono mai prescalati.

Il numero massimo di iterazioni può essere limitato con l'opzione **maxiter**. Un valore di 0 o **default** significa che non ci sono limiti.

L'opzione **limit** può essere usata per cambiare il limite epsilon di default (1e-5) per rilevare la convergenza. Quando la somma dei residui quadrati cambia di un fattore inferiore a questo numero (epsilon), l'adattamento è considerato "convergente" ('converged'). L'opzione **limit_abs** impone un ulteriore limite assoluto nel cambiamento della somma dei residui quadrati ed è predefinito a zero.

Se si ha bisogno di un controllo ancora maggiore sull'algoritmo, e si conosce bene l'algoritmo di Marquardt-Levenberg, le seguenti opzioni possono essere usate per influenzarlo. Il valore iniziale di **lambda** è normalmente calcolato automaticamente dalla ML-matrix, ma se si desidera, si può fornire il proprio valore usando l'opzione **start_lambda**. Impostandola su **default** si riabiliterà la selezione automatica. L'opzione **lambda_factor** imposta il fattore di cui **lambda** viene aumentato o diminuito ogni volta che la funzione target del chi-quadrato è aumentata o diminuita significativamente. Impostandola su **default** riabilita il fattore predefinito di 10.0.

L'opzione **script** può essere usata per specificare un comando **gnuplot** da eseguire quando un adattamento (fit) viene interrotto–vedere **fit (p. 95)**. Questa impostazione ha la precedenza sul default di **replot** e sulla variabile d'ambiente **FIT_SCRIPT**.

Se l'opzione **covariancevariables** è attiva, le covarianze tra i parametri finali saranno salvate in variabili definite dall'utente. Il nome della variabile per una certa combinazione di parametri si forma anteponendo "FIT_COV_" al nome del primo parametro e unendo i nomi dei due parametri con "_". Per esempio, dati i parametri "a" e "b", la variabile di covarianza è chiamata "FIT_COV_a_b".

Nella versione 5 la sintassi del comando fit è cambiata e ora è predefinita a unitweights se non viene data la keyword 'error'. L'opzione **v4** ripristina il comportamento predefinito della versione 4 di gnuplot, vedere anche **fit (p. 95)**.

Fontpath

Sintassi:

```
set fontpath "/directory/where/my/fonts/live"
set term postscript fontfile <filename>
```

[DEPRECATO nella versione 5.4]

La directory **fontpath** è rilevante solo per incorporare i font nell'output postscript prodotto dal terminale postscript. Non ha effetto su altri terminali di gnuplot. Se non si incorporano font non si ha bisogno di questo comando, e anche se si incorporano font, è necessario solo per i font che non possono essere trovati tramite gli altri percorsi sottostanti.

Versioni precedenti di gnuplot cercavano di emulare un gestore di font localizzando più alberi di directory contenenti font. Questo è stato sostituito da una ricerca nei seguenti punti: (1) un percorso assoluto dato nel comando **set term postscript fontfile** (2) la directory attuale (3) una qualsiasi delle directory specificate da **set loadpath**. (4) la directory specificata da **set fontpath**. (5) la directory fornita nella variabile d'ambiente GNUPLOT_FONTPATH

Nota: Il percorso di ricerca dei font specificati da filename per i terminali libgd (png gif jpeg sixel) è controllato dalla variabile d'ambiente GDFONTPATH.

Format

Il formato delle etichette dei segni di tic può essere impostato con il comando **set format** o con i comandi **set tics format** o i comandi individuali **set {axis}tics format**.

Sintassi:

```
set format {<axes>} {"<format-string>"} {numeric|timedate|geographic}
show format
```

dove <axes> è o **x**, **y**, **xy**, **x2**, **y2**, **z**, **cb** o niente (che applica il formato a tutti gli assi). I seguenti due comandi sono equivalenti:

```
set format y "%.2f"
set ytics format "%.2f"
```

La lunghezza della stringa è limitata a 100 caratteri. Il formato predefinito è "% h", "$%h$" per i terminali LaTeX. Altri formati come "%.2f" o "%3.0em" sono spesso preferibili. "set format" senza stringa seguente ripristinerà il default.

Se viene data la stringa vuota "", i tic non avranno etichette, anche se il segno di tic sarà comunque plottato. Per eliminare i segni di tic, usare **unset xtics** o **set tics scale 0**.

La newline (\n) e il markup del testo avanzato sono accettati nella stringa di formato. In questo caso bisogna usare le virgolette doppie piuttosto che quelle singole. Vedere anche **sintassi (p. 60)**. I caratteri non preceduti da "%" sono stampati parola per parola. Quindi è possibile includere spazi ed etichette nella propria stringa di formato, come "%g m", che metterà " m" dopo ogni numero. Se si vuole il "%" stesso, lo si raddoppia: "%g %%".

Vedere anche **set xtics (p. 228)** per maggiori informazioni sulle etichette dei tic, e **set decimalsign (p. 152)** per come usare separatori decimali non predefiniti nei numeri stampati in questo modo. Vedere anche la `demo electron (electron.dem)`.

Gprintf

La funzione stringa gprintf("format",x) usa gli specificatori di formato di gnuplot, come fanno i comandi gnuplot **set format**, **set timestamp** e altri. Questi specificatori di formato non sono gli stessi usati dalla routine standard del linguaggio C sprintf(). gprintf() accetta solo una singola variabile da formattare. Gnuplot fornisce anche una routine sprintf("format",x1,x2,...) se si preferisce. Per un elenco delle opzioni di formato di gnuplot, vedere **specificatori di formato (p. 158)**.

Specificatori di formato

I formati accettabili (se non in modalità ora/data) sono:

Specifiche del formato numerico dell'etichetta del segno di tic	
Formato	Spiegazione
`%f`	notazione in virgola mobile
`%e` or `%E`	notazione esponenziale; una "e" o "E" prima della potenza
`%g` or `%G`	il più breve di `%e` (o `%E`) e `%f`
`%h` or `%H`	come `%g with "x10^{%S}" or "*10^{%S}" instead of "e%S"`
`%x` or `%X`	esadecimale
`%o` or `%O`	ottale
`%t`	mantissa in base 10
`%l`	mantissa alla base dell'attuale scala logaritmica
`%s`	mantissa alla base dell'attuale scala logaritmica; potenza scientifica
`%T`	power in base 10
`%L`	power alla base dell'attuale scala logaritmica
`%S`	potenza scientifica
`%c`	sostituzione del carattere per la potenza scientifica
`%b`	mantissa della notazione ISO/IEC 80000 (ki, Mi, Gi, Ti, Pi, Ei, Zi, Yi)
`%B`	prefisso della notazione ISO/IEC 80000 (ki, Mi, Gi, Ti, Pi, Ei, Zi, Yi)
`%P`	multiplo di pi greco

Una potenza "scientifica" ('scientific') è una tale per cui l'esponente è un multiplo di tre. La sostituzione dei caratteri delle potenze scientifiche (**"%c"**) è stata implementata per le potenze nell'intervallo da -18 a +18. Per i numeri al di fuori di questo intervallo il formato ritorna esponenziale.

Altri modificatori accettabili (che vengono dopo il "%" ma prima dello specificatore di formato) sono "-", che allinea a sinistra il numero; "+", forza tutti i numeri ad essere esplicitamente segnati; " " (uno spazio), che fa sì che i numeri positivi abbiano uno spazio davanti, mentre i numeri negativi hanno "-"; "#", mette un punto decimale dopo le virgole che hanno solo zeri dopo il punto decimale; un intero positivo, che definisce la larghezza del campo; "0" (la cifra, non la lettera) subito prima della larghezza del campo, indica che devono essere usati gli zeri iniziali invece degli spazi vuoti iniziali; e un punto decimale seguito da un intero non negativo, che definisce la precisione (il numero minimo di cifre di un intero, o il numero di cifre dopo il punto decimale di una virgola mobile).

Alcuni sistemi potrebbero non supportare tutti questi modificatori, ma potrebbero anche supportarne altri; in caso di dubbio, controllare la documentazione appropriata e poi sperimentare.

Esempi:

```
set format y "%t"; set ytics (5,10)          # "5.0" e "1.0"
set format y "%s"; set ytics (500,1000)      # "500" e "1.0"
set format y "%+-12.3f"; set ytics(12345)    # "+12345.000  "
set format y "%.2t*10^%+03T"; set ytic(12345)# "1.23*10^+04"
set format y "%s*10^{%S}"; set ytic(12345)   # "12.345*10^{3}"
set format y "%s %cg"; set ytic(12345)       # "12.345 kg"
set format y "%.0P pi"; set ytic(6.283185)   # "2 pi"
set format y "%.0f%%"; set ytic(50)          # "50%"
```

```
set log y 2; set format y '%l'; set ytics (1,2,3)
#displays "1.0", "1.0" and "1.5" (since 3 is 1.5 * 2^1)
```

Ci sono alcuni casi problematici che sorgono quando numeri come 9.999 vengono stampati con un formato che richiede sia l'arrotondamento che una potenza.

Se il tipo di dati per l'asse è ora/data, la stringa di formato deve contenere codici validi per la funzione 'strftime' (fuori da **gnuplot**, digitare "man strftime"). Vedere **set timefmt (p. 218)** per una lista dei codici di formato di input consentiti.

Specificatori ora/data

Ci sono due gruppi di specificatori di formato orario: ora/data e tempo relativo. Questi possono essere usati per generare etichette dei tic dell'asse o per codificare il tempo in una stringa. Vedere **set xtics time (p. 231)**, **strftime (p. 40)**, **strptime (p. 40)**.

I formati ora/data sono

Specificatori di data	
Formato	Spiegazione
%a	nome abbreviato del giorno della settimana
%A	nome completo del giorno della settimana
%b o %h	nome abbreviato del mese
%B	nome completo del mese
%d	giorno del mese, 01–31
%D	abbreviazione per "%m/%d/%y" (solo output)
%F	abbreviazione per "%Y-%m-%d" (solo output)
%k	ora, 0–23 (una o due cifre)
%H	ora, 00–23 (sempre due cifre)
%l	ora, 1–12 (una o due cifre)
%I	ora, 01–12 (sempre due cifre)
%j	giorno dell'anno, 001–366
%m	mese, 01–12
%M	minuto, 00–60
%p	"am" o "pm"
%r	abbreviazione per "%I:%M:%S %p" (solo output)
%R	abbreviazione per %H:%M" (solo output)
%S	secondo, intero 00–60 in output, (doppio) in input
%s	numero di secondi dall'inizio dell'anno 1970
%T	abbreviazione per "%H:%M:%S" (solo output)
%U	settimana dell'anno (CDC/MMWR "epi week") (ignorato in input)
%w	giorno della settimana, 0–6 (Domenica = 0)
%W	settimana dell'anno (data settimana ISO 8601) (ignorato in input)
%y	anno, 0-99 in range 1969-2068
%Y	anno, 4 cifre
%z	fuso orario, [+-]hh:mm
%Z	nome fuso orario, stringa ignorata

Per ulteriori informazioni sul formato %W (settimana dell'anno ISO) vedere **tm_week (p. 160)**. Il formato %U (settimana epidemiologica CDC/MMWR) è simile a %W tranne per il fatto che usa le settimane che iniziano di domenica piuttosto che di lunedì. Avvertimento: Entrambi i formati %W e %U erano inaffidabili nelle versioni di gnuplot precedenti alla 5.4.2. Vedere unit test "week_date.dem".

I formati di tempo relativo esprimono la lunghezza di un intervallo di tempo su entrambi i lati di un punto di tempo zero. I formati di tempo relativo sono

Specificatori di orario	
Formato	Spiegazione
%tD	+/- giorni relativi al tempo=0
%tH	+/- ore relative al tempo=0 (non si interrompe a 24)
%tM	+/- minuti relativi al tempo=0
%tS	+/- secondi associati al precedente campo tH o tM

I formati numerici possono essere preceduti da uno "0" ("zero") per riempire il campo con zeri iniziali, e preceduti da una cifra positiva per definire la larghezza minima del campo. I formati %S e %t accettano anche uno specificatore di precisione in modo che possano essere scritti ore/minuti/secondi frazionati.

Esempi Esempi di formato della data:

Supponiamo che il valore di x in secondi corrisponda a un tempo leggermente precedente la mezzanotte del 25 dicembre 1976. Il testo stampato per un'etichetta tic in questa posizione sarebbe

```
set format x                # predefinito a "12/25/76 \n 23:11"
set format x "%A, %d %b %Y" # "Saturday, 25 Dec 1976"
set format x "%r %D"        # "11:11:11 pm 12/25/76"
```

Esempi di formato dell'ora:

Gli specificatori del formato della data codificano un tempo in secondi come un'ora dell'orologio in un giorno particolare. Quindi le ore vanno solo da 0-23, i minuti da 0-59, e i valori negativi corrispondono a date precedenti all'epoca (1-Gen-1970). Per riportare un valore di tempo in secondi come un certo numero di ore/minuti/secondi relativi a un tempo 0, bisogna usare i formati dell'ora %tH %tM %tS. Per riportare un valore di -3672.50 secondi

```
set format x                # formato data predefinito "12/31/69 \n 22:58"
set format x "%tH:%tM:%tS"  # "-01:01:12"
set format x "%.2tH hours"  # "-1.02 hours"
set format x "%tM:%.2tS"    # "-61:12.50"
```

Tm_week

La funzione **tm_week(t, standard)** interpreta il suo primo argomento t come un tempo in secondi dal 1 gennaio 1970. Nonostante il nome di questa funzione, essa non riporta un campo della struttura POSIX tm.

Se standard = 0, essa restituisce il numero della settimana nel sistema ISO 8601 "week date". Questo corrisponde al formato di tempo %W di gnuplot. Se standard = 1, essa restituisce il numero della settimana epidemiologica del CDC ("epi week"). Questo corrisponde al formato di tempo %U di gnuplot. Per le corrispondenti funzioni inverse che convertono le date della settimana in tempo del calendario vedere **weekdate_iso (p. 160)**, **weekdate_cdc (p. 161)**.

In sintesi, la settimana ISO 1 dell'anno YYYY inizia con il lunedì più vicino al 1 gennaio YYYY. Questo può collocarlo nell'anno civile precedente. Per esempio Mar 30 Dic 2008 ha la data della settimana ISO 2009-W01-2 (2° giorno della settimana 1 del 2009). All'inizio di gennaio possono esserci fino a un massimo di tre giorni prima del lunedì della settimana ISO 1; questi giorni sono assegnati all'ultima settimana dell'anno civile precedente. Ad esempio, venerdì 1 gennaio 2021 ha la data della settimana ISO 2020-W53-5.

La settimana epidemiologica dell'US Center for Disease Control (CDC) è una convenzione simile per la data della settimana che differisce dallo standard ISO definendo una settimana a partire dalla domenica, piuttosto che dal lunedì.

Weekdate_iso

Sintassi:

```
time = weekdate_iso( year, week [, day] )
```

Questa funzione converte dai componenti anno, settimana, giorno di una data nel formato ISO 8601 "week date" alla data del calendario come tempo in secondi dalla data epocale 1 gennaio 1970. Si noti che l'anno nominale nel sistema di data settimanale non è necessariamente lo stesso dell'anno civile. La settimana è un intero da 1 a 53. Il parametro del giorno è opzionale. Se è omesso o uguale a 0, il tempo restituito è l'inizio della settimana. Altrimenti il giorno è un intero da 1 (lunedì) a 7 (domenica). Vedere **tm_week (p. 160)** per ulteriori informazioni su una funzione inversa che converte dalla data del calendario al numero della settimana nella convenzione standard ISO.

Esempio:

```
# Plotta i dati di un file con la colonna 1 contenente le settimane ISO
#     Settimana     casi  decessi
#     2020-05        432     1
calendar_date(w) = weekdate_iso( int(w[1:4]), int(w[6:7]) )
set xtics time format "%b\n%Y"
plot FILE using (calendar_date(strcol(1))) : 2   title columnhead
```

Weekdate_cdc

Sintassi:

```
time = weekdate_cdc( year, week [, day] )
```

Questa funzione converte dai componenti anno, settimana, giorno di una data nel formato CDC/MMWR "epi week" alla data del calendario come tempo in secondi dalla data epocale 1 gennaio 1970. La convenzione della data settimanale CDC differisce dalla data settimanale ISO in quanto è definita in termini di ogni settimana che va da giorno 1 = domenica al giorno 7 = sabato. Se il terzo parametro è 0 o è omesso, il tempo restituito è l'inizio della settimana. Vedere **tm_week (p. 160)** e **weekdate_iso (p. 160)**.

Function style

Questa forma del comando è deprecata. Usare **set style function**.

Functions

Il comando **show functions** elenca tutte le funzioni definite dall'utente e le loro definizioni.

Sintassi:

```
show functions
```

Per informazioni sulla definizione e l'uso delle funzioni in **gnuplot**, si consiglia di vedere **espressioni (p. 37)**. Vedere anche `spline come funzioni definite dall'utente (spline.dem)`

e `uso di funzioni e variabili complesse per airfoils (airfoil.dem).`

Grid

Il comando **set grid** permette di disegnare linee di griglia sul grafico.

Sintassi:

```
set grid {{no}{m}xtics} {{no}{m}ytics} {{no}{m}ztics}
         {{no}{m}x2tics} {{no}{m}y2tics} {{no}{m}rtics}
         {{no}{m}cbtics}
         {polar {<angle>}}
```

```
          {layerdefault | front | back}
          {{no}vertical}
          {<line-properties-major> {, <line-properties-minor>}}
unset grid
show grid
```

La griglia può essere abilitata e disabilitata per i segni di tic maggiori e/o minori su qualsiasi asse, e il tipo di linea e lo spessore della linea possono essere specificati per le linee maggiori e minori della griglia, anche tramite un stile di linea predefinito, nella misura in cui il driver del terminale attivo lo supporta (vedere **set style line (p. 210)**).

Una griglia polare può essere disegnata per i grafici 2D. Questa è l'azione predefinita di **set grid** se il programma è già in modalità polare, ma può essere abilitata esplicitamente da **set grid polar** <**angle**> **rtics** indipendentemente dal fatto che il programma sia o meno in modalità polare. I cerchi sono disegnati per intersecare i tic maggiori e/o minori lungo l'asse r, e le linee radiali sono disegnate con una distanza di <angle>. I segni di tic intorno al perimetro sono controllati da **set ttics**, ma questi non producono linee di griglia radiali.

I tic pertinenti devono essere abilitati prima che **set grid** possa disegnarli; **gnuplot** ignorerà tranquillamente le istruzioni per disegnare linee di griglia su tic inesistenti, ma appariranno se i tic vengono successivamente abilitati.

Se non viene specificato alcun tipo di linea per le linee di griglia minori, viene usato lo stesso tipo di linea delle maggiori. L'angolo polare predefinito è di 30 gradi.

Se è dato **front**, la griglia è disegnata sopra i dati riportati sul grafico (graph). Se è dato **back**, la griglia è disegnata sotto i dati riportati sul grafico (graph). Usare **front** eviterà che la griglia sia oscurata da dati densi. La configurazione predefinita, **layerdefault**, equivale a **back** per i grafici 2D. Nei grafici 3D l'impostazione predefinita è dividere la griglia e il riquadro (box) del grafico (graph) in due layer: uno dietro, l'altro davanti ai dati plottati e alle funzioni. Poiché la modalità **hidden3d** esegue il proprio ordinamento, ignora tutte le opzioni di ordine di disegno della griglia e passa le linee della griglia attraverso il meccanismo di rimozione delle linee nascoste. Queste opzioni in realtà influenzano non solo la griglia, ma anche le linee emesse (output) da **set border** e i vari segni di tic (vedere **set xtics (p. 228)**).

Nei grafici 3D le linee della griglia nelle posizioni tic degli assi x e y sono disegnate per default solo sul piano di base parallelo a z=0. La keyword **vertical** attiva il disegno delle linee della griglia anche nei piani xz e yz, che vanno da zmin a zmax.

Le linee della griglia Z sono disegnate sul fondo del grafico. Questo ha un aspetto migliore se un box parziale è disegnato intorno al grafico — vedere **set border (p. 140)**.

Hidden3d

Il comando **set hidden3d** consente la rimozione delle linee nascoste per il plotting di superficie (vedere **splot (p. 237)**). Alcune caratteristiche opzionali dell'algoritmo sottostante possono anche essere controllate usando questo comando.

Sintassi:

```
set hidden3d {defaults} |
             { {front|back}
               {{offset <offset>} | {nooffset}}
               {trianglepattern <bitpattern>}
               {{undefined <level>} | {noundefined}}
               {{no}altdiagonal}
               {{no}bentover} }
unset hidden3d
show hidden3d
```

In contrasto alla solita visualizzazione in gnuplot, la rimozione delle linee nascoste in realtà tratta la funzione data o le griglie di dati come superfici reali che non possono essere viste attraverso, quindi gli elementi del grafico dietro la superficie saranno nascosti da essa. Per far sì che questo funzioni, la superficie deve avere una 'grid structure' (struttura a griglia) (vedere **splot datafile (p. 238)**), e deve essere disegnata **with lines** (con linee) o **with linespoints** (con punti e linee).

Quando **hidden3d** è impostato, sia la porzione nascosta della superficie ed eventualmente i suoi contorni disegnati sulla base (vedere **set contour (p. 147)**) che la griglia saranno nascoste. Se viene plottata più di una superficie, ognuna di esse avrà le sue parti nascoste rimosse rispetto a se stessa e alle altre superfici. I contorni disegnati sulla superficie (**set contour surface**) non funzionano.

hidden3d influisce anche sugli stili di plotting 3D **points**, **labels**, **vectors**, e **impulses** anche se nessuna superficie è presente nel grafico (graph). Le porzioni non oscurate di ogni vettore sono disegnate come segmenti di linea (senza punte di freccia). I singoli grafici (plot) all'interno del grafico (graph) possono essere esplicitamente esclusi da questa elaborazione aggiungendo l'opzione extra **nohidden3d** allo specificatore **with**.

Hidden3d non ha effetto sulle superfici solide disegnate usando la modalità pm3d. Per ottenere un effetto simile solo per le superfici pm3d, bisogna invece usare **set pm3d depthorder**. Per mescolare le superfici pm3d con la normale elaborazione **hidden3d**, usare l'opzione **set hidden3d front** per obbligare tutti gli elementi inclusi nell'elaborazione hidden3d a essere disegnati dopo qualsiasi altro elemento del grafico, inclusa la superficie pm3d.

Le funzioni sono valutate alle intersezioni delle isolinee. L'algoritmo interpola in modo lineare tra i punti della funzione o i data point quando determina i segmenti di linea visibili. Questo significa che l'aspetto di una funzione può essere diverso quando è plottata con **hidden3d** rispetto a quando è plottata con **nohidden3d** perché in quest'ultimo caso le funzioni sono valutate ad ogni campione. Si prega di vedere **set samples (p. 204)** e **set isosamples (p. 164)** per una discussione sulla differenza.

L'algoritmo usato per rimuovere le parti nascoste delle superifici ha alcune funzionalità aggiuntive controllabili da questo comando. Specificando **defaults** verranno ripristinate tutte alle loro impostazioni predefinite, come spiegato di seguito. Se non viene dato **defaults**, saranno influenzate solo le opzioni espressamente specificate: tutte le altre manterranno i loro valori precedenti, in modo da poter attivare/disattivare la rimozione delle linee nascoste tramite **set {no}hidden3d**, senza modificare il set delle opzioni scelte.

La prima opzione, **offset**, influenza il tipo di linea usato per le linee sul lato posteriore ('back'). Normalmente, sono disegnate in un tipo di linea con un numero di indice più alto di uno di quello usato per la parte frontale, per rendere i due lati della superficie distinguibili. È possibile specificare un diverso offset di tipo di linea da aggiungere invece del default 1, con **offset** <**offset**>. L'opzione **nooffset** sta per **offset 0**, facendo sì che i due lati della superficie usino lo stesso tipo di linea.

Segue l'opzione **trianglepattern** <**bitpattern**>. <bitpattern> deve essere un numero tra 0 e 7, interpretato come uno schema di bit. Ogni bit determina la visibilità di un bordo dei triangoli in cui è suddivisa ogni superficie. Bit 0 è per i bordi 'orizzontali' della griglia, Bit 1 per quelli 'verticali' e Bit 2 per le diagonali che dividono ogni cella della griglia originale in due triangoli. Il pattern predefinito è 3, che rende visibili tutte le linee orizzontali e verticali, ma non le diagonali. Si consiglia di scegliere 7 per vedere anche quelle diagonali.

L'opzione **undefined** <**level**> permette di decidere cosa deve fare l'algoritmo con i data point che sono indefiniti (dati mancanti, o valori di funzione indefiniti), o che superano gli intervalli x, y o z dati. Tali punti possono essere plottati comunque, o eliminati dal set di dati di input. Tutti gli elementi della superficie che toccano un punto che è stato tolto saranno tolti anch'essi, creando così un buco nella superficie. Se <level> = 3, equivalente all'opzione **noundefined**, non verrà eliminato nessun punto. Questo può produrre problemi di ogni tipo altrove, quindi si dovrebbe evitare. <level> = 2 eliminerà i punti non definiti, ma manterrà quelli fuori dall'intervallo. <level> = 1, il default, eliminerà anche i punti fuori dall'intervallo.

Specificando **noaltdiagonal**, si può sovrascrivere la gestione predefinita di un caso speciale che può verificarsi se **undefined** è attivo (cioè <level> non è 3). Ogni cella della superficie di input strutturata a griglia sarà divisa in due triangoli lungo una delle sue diagonali. Normalmente, tutte queste diagonali hanno lo stesso orientamento rispetto alla griglia. Se esattamente uno dei quattro angoli della cella è escluso dal gestore

undefined, e questo è sulla solita diagonale, entrambi i triangoli saranno esclusi. Tuttavia se l'impostazione predefinita di **altdiagonal** è attiva, sarà invece scelta l'altra diagonale per questa cella, minimizzando la dimensione del buco nella superficie.

L'opzione **bentover** controlla cosa succede in un altro caso speciale, questa questa volta in concomitanza con **trianglepattern**. Per le superfici piuttosto increspate, può succedere che i due triangoli in cui è divisa una cella di superficie siano visti da lati opposti (cioè il quadrangolo originale è 'bent over' (piegato su se stesso)), come illustrato nella seguente arte ASCII:

```
                                                    C----B
quadrangolo originale:  A--B      quadrangolo visualizzato:  |\   |
  ("set view 0,0")      | /|    ("set view 75,75" perhaps)   | \  |
                        |/ |                                 |  \ |
                        C--D                                 |   \|
                                                             A    D
```

Se i bordi diagonali delle celle di superficie non vengono generalmente resi visibili dal bit 2 del <bitpattern>, il bordo CB sopra non verrà disegnato affatto, rendendo la visualizzazione risultante difficile da capire. Pertanto, in questo caso, l'opzione predefinita di **bentover** lo renderà visibile. Se non lo si desidera, è possibile scegliere invece **nobentover**. Vedere anche `hidden line removal demo (hidden.dem)`

e `complex hidden line demo (singulr.dem).`

Historysize

(Deprecato). **set historysize N** equivale a **set history size N**. **unset historysize** equivale a **set history size -1**.

History

Sintassi:

```
set history {size <N>} {quiet|numbers} {full|trim} {default}
```

Quando si esce da gnuplot, il valore della dimensione della cronologia limita il numero di linee salvate nel file della cronologia. **set history size -1** permette di scrivere un numero di linee illimitato nel file della cronologia.

Per default il comando **history** stampa un numero di linea davanti a ogni comando. **history quiet** cancella il numero solo per questo comando. **set history quiet** cancella i numeri per tutti i futuri comandi **history**.

L'opzione **trim** riduce il numero di righe duplicate nell'elenco della cronologia rimuovendo le precedenti istanze del comando corrente.

Impostazioni default: **set history size 500 numbers trim**.

Isosamples

La densità delle isolinee (griglia) per plottare delle funzioni come superfici può essere cambiata con il comando **set isosamples**.

Sintassi:

```
set isosamples <iso_1> {,<iso_2>}
show isosamples
```

Ogni grafico della superficie della funzione avrà linee iso_u <iso_1> e linee iso_v <iso_2>. Se si specifica solo <iso_1>, <iso_2> sarà impostato allo stesso valore di <iso_1>. Per default, il campionamento è impostato

su 10 isolinee per asse u o v. Un tasso di campionamento più alto produrrà grafici più accurati, ma richiederà più tempo. Questi parametri non hanno effetto sul plotting dei file di dati.

Una isolinea è una curva parametrizzata da uno dei parametri della superficie mentre l'altro parametro della superficie è fisso. Le isolinee forniscono un modo semplice per visualizzare una superficie. Fissando il parametro u della superficie s(u,v), si producono le linee iso-u della forma c(v) = s(u0,v), e fissando il parametro v, si producono le linee iso-v della forma c(u) = s(u,v0).

Quando viene fatto un grafico della superficie di una funzione senza la rimozione delle linee nascoste, **set samples** controlla il numero di punti campionati lungo ogni isolinea; vedere **set samples (p. 204)** e **set hidden3d (p. 162)**. L'algoritmo del contorno presuppone che un campionamento della funzione si verifichi ad ogni intersezione delle isolinee, quindi un cambiamento nei **samples** (campioni) e negli **isosamples** può essere desiderabile quando si cambia la risoluzione di una superficie/contorno di funzione.

Isosurface

Sintassi:

```
set isosurface {mixed|triangles}
set isosurface {no}insidecolor <n>
```

Le superfici plottate dal comando **splot $voxelgrid with isosurface** sono costruite di default da un misto di quadrangoli e triangoli. L'uso di quadrangoli crea un'impressione visiva meno complicata. Questo è il default. Questo comando fornisce un'opzione per tessellare solo con triangoli.

Per default l'interno di un'isosuperficie è disegnato in un colore diverso. Il metodo per scegliere quel colore è lo stesso delle superfici hidden3d, dove un offset <n> viene aggiunto al tipo di linea di base. Per disegnare sia la superficie interna che la superficie esterna con lo stesso colore, usare **set isosurface noinsidecolor**.

Jitter

Sintassi:

```
set jitter {overlap <yposition>} {spread <factor>} {wrap <limit>}
           {swarm|square|vertical}
```

Esempi:

```
set jitter                    # jitter points within 1 character width
set jitter overlap 1.5        # jitter points within 1.5 character width
set jitter over 1.5 spread 0.5  # uguale ma con metà dello spostamento su x
```

Quando una o entrambe le coordinate di un set di dati sono limitate a valori discreti, allora molti punti possono trovarsi esattamente uno sopra l'altro. Il jittering introduce un offset alle coordinate di questi punti sovrapposti che li distribuisce in un cluster (ragruppamento). Il valore di soglia per trattare i punti come se fossero sovrapposti può essere specificato nelle larghezze dei caratteri o in una qualsiasi delle solite opzioni di coordinate. Vedere **coordinate (p. 33)**. Il jitter influenza gli stili di grafico 2D **with points** e **with impulses**. Influenza anche il plotting 3D di griglie voxel.

L'operazione di jittering di default sposta i punti solo lungo x. Questo produce un pattern distintivo a volte chiamato "bee swarm plot" (grafico a sciame d'api). La keyword opzionale **square** regola la coordinata y dei punti spostati oltre alla loro coordinata x, in modo che i punti si trovino in layer (strati) diversi separati da almeno la distanza **overlap**.

Per il jitter (solo) lungo y piuttosto che lungo x, usare la keyword **vertical**.

Lo spostamento massimo (in unità di carattere) può essere limitato usando la keyword **wrap**.

Si noti che sia il criterio di sovrapposizione che la grandezza del jitter sono predefiniti a un'unità di carattere. Quindi l'aspetto del grafico cambierà con la dimensione del font del terminale, dimensione del canvas o

il fattore di zoom. Per evitare questo è possibile specificare il criterio di sovrapposizione nel sistema di coordinate dell'asse y (la prima (**first**) keyword) e regolare la dimensione del punto e il moltiplicatore di diffusione come appropriato. Vedere **coordinate (p. 33)**, **pointsize (p. 202)**.

Avvertimento: jitter è incompatibile con "pointsize variable".

set jitter è anche utile nei grafici 3D di dati voxel. Poiché le griglie voxel sono reticoli regolari di punti uniformemente distanziati, molti punti di vista fanno sì che i punti si sovrappongano e/o generino pattern Moiré. Questi artefatti possono essere rimossi spostando il simbolo disegnato in ogni punto della griglia di una quantità casuale.

Chiave

Il comando **set key** abilita una chiave (key) (o legenda) contenente un titolo e un campione (linea, punto, casella(box)) per ogni grafico (plot) nel grafico (graph). La chiave può essere disattivata richiedendo **set key off** o **unset key**. Le singole entry della chiave possono essere disattivate usando la keyword **notitle** nel comando plot corrispondente. Il testo dei titoli è controllato dall'opzione **set key autotitle** o dalla keyword **title** dei singoli comandi **plot** e **splot**. Vedere **plot title (p. 129)** per maggiori informazioni.

Sintassi:

```
set key {on|off} {default}
      {{inside | outside | fixed} | {lmargin | rmargin | tmargin | bmargin}
        | {at <position>}}
      {left | right | center} {top | bottom | center}
      {vertical | horizontal} {Left | Right}
      {{no}enhanced}
      {{no}opaque {fc <colorspec>}}
      {{no}reverse} {{no}invert}
      {samplen <sample_length>} {spacing <line_spacing>}
      {width <width_increment>} {height <height_increment>}
      {{no}autotitle {columnheader}}
      {title {"<text>"} {{no}enhanced} {center | left | right}}
      {font "<face>,<size>"} {textcolor <colorspec>}
      {{no}box {linestyle <style> | linetype <type> | linewidth <width>}}
      {maxcols {<max no. of columns> | auto}}
      {maxrows {<max no. of rows> | auto}}
unset key
show key
```

Gli elementi all'interno della chiave sono impilati secondo la modalità **vertical** o **horizontal**. Nel caso di **vertical**, la chiave occupa il minor numero possibile di colonne. Ovvero, gli elementi sono allineati in una colonna fino a quando non si esaurisce lo spazio verticale, a quel punto si inizia una nuova colonna. Lo spazio verticale può essere limitato usando 'maxrows'. Nel caso di **horizontal**, la chiave occupa il minor numero possibile di righe. Lo spazio orizzontale può essere limitato usando 'maxcols'.

Per default la chiave è posizionata nell'angolo interno superiore destro del grafico (graph). Le keyword **left**, **right**, **top**, **bottom**, **center**, **inside**, **outside**, **lmargin**, **rmargin**, **tmargin**, **bmargin** (, **above**, **over**, **below** e **under**) possono essere usate per posizionare automaticamente la chiave in altre posizioni del grafico (graph). Anche un **at** <**position**> può essere dato per indicare precisamente dove il grafico dovrebbe essere posizionato. In questo caso, le keyword **left**, **right**, **top**, **bottom** e **center** servono uno scopo analogo per l'allineamento. Per maggiori informazioni, vedere **key placement (p. 169)**.

L'allineamento dei titoli del grafico all'interno della chiave è controllato da **Left** o **Right** (default). Il testo e il campione possono essere invertiti (**reverse**) e una casella può essere disegnata intorno alla chiave (**box {...}**) in un **linetype** e una **linewidth** specificati, o un **linestyle** definito dall'utente.

Il testo nella chiave è impostato di default in modalità **enhanced**, questo può essere cambiato con l'opzione **{no}enhanced**, anche indipendentemente per il solo titolo della chiave e per ogni singolo grafico.

Per default la chiave è costruita un grafico alla volta. Cioè, il simbolo e il titolo della chiave sono disegnati nello stesso momento del grafico corrispondente. Questo significa che i grafici più recenti possono a volte porre degli elementi sopra la chiave. **set key opaque** fa sì che la chiave venga generata dopo tutti i grafici. In questo caso l'area della chiave è riempita con il colore di sfondo o con il colore di riempimento richiesto, e in seguito vengono scritti i simboli e i titoli della chiave. L'impostazione predefinita può essere ripristinata da **set key noopaque**.

Per default la prima etichetta del grafico è in cima alla chiave e le etichette successive sono inserite sotto di essa. L'opzione **invert** fa sì che la prima etichetta sia posta in fondo alla chiave, con le etichette successive inserite sopra di essa. Questa opzione è utile per forzare l'ordine verticale delle etichette nella chiave per corrispondere all'ordine dei tipi di casella in un istogramma impilato.

<height_increment> è un numero di altezze di carattere da aggiungere o sottrarre all'altezza della casella della chiave. Questo è utile soprattutto quando si sta ponendo un riquadro intorno alla chiave e si desiderano bordi più grandi intorno alle entry della chiave.

Un titolo generale può essere messo sulla chiave (**title "<text>"**) — vedere anche **sintassi (p. 60)** per la distinzione tra testo fra virgolette singole o doppie. L'allineamento del titolo è posto per default al centro e può essere cambiato con le keyword **right** o **left**.

I valori predefiniti per **set key** sono **on**, **right**, **top**, **vertical**, **Right**, **noreverse**, **noinvert**, **samplen 4**, **spacing 1**, **notitle**, e **nobox**. Il <linetype> predefinito è lo stesso usato per i bordi del grafico. Inserendo **set key default** si riporta la chiave alla sua configurazione predefinita.

Ogni grafico è rappresentato nella chiave da una singola linea contenente una linea o simbolo o forma che rappresenta lo stile di grafico e un titolo corrispondente. Usando la keyword **notitle** nel comando plot si sopprime la generazione della linea. I grafici di contorno hanno generato entry aggiuntive nella chiave, una per ogni contorno (vedere **cntrlabel (p. 144)**). È possibile aggiungere ulteriori linee alla chiave inserendo un comando plot dummy che usa la keyword **keyentry** piuttosto che un filename o una funzione. Vedere **keyentry (p. 168)**.

Quando si usa il gruppo di terminali TeX/LaTeX o terminali in cui le informazioni di formattazione sono incorporate nella stringa, **gnuplot** può solo stimare la larghezza della stringa per il posizionamento delle chiavi. Se la chiave deve essere posizionata a sinistra, può essere pratico usare la combinazione **set key left Left reverse**.

Chiave 3D

Il posizionamento della chiave per i grafici 3D (**splot**) usa di default l'opzione **fixed**. Nota: questo è un cambiamento rispetto alla versione 5.0 e precedenti di gnuplot. Il posizionamento **fixed** è molto simile al posizionamento **inside**, con un'importante differenza. I confini di un grafico 3D cambiano quando il punto di vista viene ruotato o scalato. Se la chiave è posizionata "all'interno" (**inside**) di questi confini, allora anche la chiave si sposta quando la visuale viene cambiata. Il posizionamento "fisso" (**fixed**) ignora le modifiche dei punti di vista o ridimensionamento; cioè la chiave rimane fissa in una posizione sul canvas mentre il grafico viene ruotato.

Per i grafici 2D l'opzione **fixed** equivale esattamente a **inside**.

Se **splot** viene usato per disegnare i contorni, per default viene generata una key entry separata per ogni livello di contorno con un tipo di linea distinto. Per modificare questo vedere **set cntrlabel (p. 144)**.

Esempi chiave

Questo colloca la chiave nella posizione predefinita:

```
set key default
```

Questo disabilita la chiave:

```
unset key
```

Questo pone una chiave alle coordinate 2,3.5,2 nel sistema di coordinate predefinito (primo):

```
set key at 2,3.5,2
```

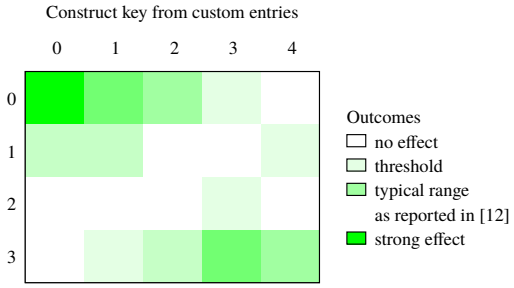
Questo pone la chiave sotto il grafico (graph):

```
set key below
```

Questo posiziona la chiave nell'angolo in basso a sinistra, allinea a sinistra il testo, gli dà un titolo e disegna un riquadro intorno ad esso nel linetype 3:

```
set key left bottom Left title 'Legend' box 3
```

Extra key entries



Normalmente ogni grafico autogenera una singola linea di entry nella chiave. Se si ha bisogno di più controllo su ciò che appare nella chiave si può usare la keyword **keyentry** nel comando **plot** o **splot** per inserire linee extra. Invece di fornire un filename o una funzione da plottare, si può usare **keyentry** come segnaposto seguito da informazioni sullo stile del grafico (usate per generare un simbolo chiave) e un titolo. Si applicano tutte le solite opzioni per il font del titolo, il colore del testo, le coordinate **at** e marcatura avanzata del testo. Esempio:

```
plot $HEATMAP matrix with image notitle, \
     keyentry with boxes fc palette cb 0 title "no effect", \
     keyentry with boxes fc palette cb 1 title "threshold", \
     keyentry with boxes fc palette cb 3 title "typical range", \
     keyentry with labels nopoint title "as reported in [12]", \
     keyentry with boxes fc palette cb 5 title "strong effect"
```

Key autotitle

set key autotitle fa sì che ogni grafico sia identificato nella chiave dal nome del file di dati o della funzione usata nel comando plot. Questo è il default. **set key noautotitle** disabilita la generazione automatica dei titoli del grafico. Il comando **set key autotitle columnheader** fa sì che la prima entry di ogni colonna dei dati di input sia interpretata come una stringa di testo e usata come titolo per il grafico corrispondente. Se la quantità che si sta plottando è una funzione di dati presi da più colonne, gnuplot può essere confuso su quale sia la colonna da cui deve prendere il titolo. In questo caso è necessario specificare esplicitamente la colonna nel comando plot, ad esempio

```
plot "datafile" using (($2+$3)/$4) title columnhead(3) with lines
```

Nota: L'effetto di **set key autotitle columnheader**, il trattamento della prima linea in un file di dati come intestazioni di colonna piuttosto che dati si applica anche se la chiave è disabilitata da **unset key**. Si applica anche ai comandi **stats** e **fit**, sebbene non generino alcuna chiave. Se si vuole che la prima linea di dati sia trattata come intestazione di colonna ma *non* utilizzarla per i titoli dei grafici, usare **set datafile columnheaders**.

In ogni caso, una keyword esplicita **title** o **notitle** nel comando plot sovrascriverà il valore predefinito di **set key autotitle**.

Posizionamento della chiave

Questa sezione descrive il posizionamento della chiave primaria autogenerata. Per costruire una chiave secondaria o posizionare altrove i titoli del grafico, vedere **multiple keys (p. 170)**.

Per capire il posizionamento, il concetto migliore è pensare a una regione, cioè, dentro/fuori, o uno dei margini. Insieme alla regione, le parole chiave **left/center/right** (l/c/r) e **top/center/bottom** (t/c/b) controllano dove, all'interno della particolare regione, dovrebbe essere posizionata la chiave.

Quando sono nella modalità **inside**, le keyword **left** (l), **right** (r), **top** (t), **bottom** (b), e **center** (c) spingono la chiave verso il confine del grafico come illustrato:

```
t/l   t/c   t/r

c/l    c    c/r

b/l   b/c   b/r
```

Quando si è in modalità **outside**, il posizionamento automatico è simile a quello dell'illustrazione, ma rispetto alla vista, piuttosto che al confine del grafico (graph). Ovvero, un bordo viene spostato verso l'interno per fare spazio alla chiave al di fuori dell'area di plotting, anche se questo può interferire con altre etichette e può causare un errore su alcuni dispositivi. Il particolare bordo del grafico che viene spostato dipende dalla posizione descritta sopra e dalla direzione di impilamento. Per opzioni centrate in una delle dimensioni, non c'è ambiguità su quale bordo bisogna spostare. Per gli angoli, quando la direzione dello stack è **vertical**, il bordo sinistro o destro viene spostato verso l'interno in modo appropriato. Quando la direzione della pila è **horizontal**, il bordo superiore o inferiore viene spostato verso l'interno in modo appropriato.

La sintassi dei margini permette il posizionamento automatico della chiave a prescindere dalla direzione della pila. Quando uno dei margini **lmargin** (lm), **rmargin** (rm), **tmargin** (tm), e **bmargin** (bm) è abbinato ad una singola keyword di direzione non contrastante, le seguenti posizioni illustrate possono contenere la chiave:

```
      l/tm  c/tm  r/tm

t/lm                  t/rm

c/lm                  c/rm

b/lm                  b/rm

      l/bm  c/bm  r/bm
```

Le keyword **above** e **over** sono sinonimi di **tmargin**. Per compatibilità di versione, **above** o **over** senza una keyword aggiuntiva l/c/r o keyword di direzione della pila usa **center** e **horizontal**. Le keyword **below** e **under** sono sinonimo di **bmargin**. Per compatibilità, **below** o **under** senza una keyword aggiuntiva l/c/r o keyword di direzione della pila usa **center** e **horizontal**. Un ulteriore problema di compatibilità di compatibilità consiste nel fatto che **outside** che appare senza una keyword aggiuntiva t/b/c o di direzione della pila usa **top**, **right** e **vertical** (cioè, come t/rm sopra).

La posizione <position> può essere un semplice x,y,z come nelle versioni precedenti, ma questi possono essere preceduti da una delle cinque keyword (**first**, **second**, **graph**, **screen**, **character**) che seleziona il sistema di coordinate in cui viene specificata la posizione della prima linea di campionamento. Vedere **coordinates (p. 33)** per ulteriori dettagli. L'effetto di **left**, **right**, **top**, **bottom**, e **center** quando è dato <position> è di allineare la chiave come se fosse un testo posizionato usando il comando dell'etichetta (label) cioè, **left** significa allineare a sinistra con la chiave a destra di <position>, ecc.

Key samples

Per default, ogni grafico (plot) sul grafico (graph) genera una entry corrispondente nella chiave. Questa entry contiene un titolo del grafico e un campione linea/punto/box dello stesso colore e proprietà di riempimento utilizzati nel grafico stesso. Le proprietà del font e del colore del testo controllano l'aspetto dei singoli titoli del grafico che appaiono nella chiave. Impostare il colore del testo su "variable" fa sì che il testo per ogni key entry sia dello stesso colore della linea o del colore di riempimento di quel grafico. Questa era l'impostazione predefinita in alcune versioni precedenti di gnuplot.

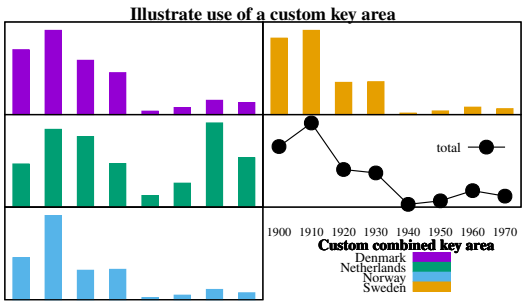
La lunghezza del campione della linea può essere controllata da **samplen**. La lunghezza del campione è computata come la somma della lunghezza del tic e di <sample_length> volte la larghezza del carattere. **samplen** influenza anche le posizioni dei campioni di punti nella chiave, poiché questi sono disegnati nel punto medio del campione della linea, anche se la linea di campionamento stessa non è disegnata.

Le linee delle key entry sono a spaziatura singola in base alla dimensione attuale del font. Questo può essere regolato da **set key spacing** <**line-spacing**>.

Il <width_increment> è un numero di larghezze di carattere da aggiungere o sottrarre alla lunghezza della stringa. Questo è utile solo quando si sta mettendo un riquadro intorno alla chiave e si stanno usando caratteri di controllo nel testo. **gnuplot** conta semplicemente il numero di caratteri nella stringa quando computa la larghezza del riquadro; questo permette di correggerla.

Multiple keys

È possibile costruire una legenda/chiave manualmente piuttosto che far apparire tutti i titoli dei grafici nella chiave autogenerata. Questo permette, per esempio, di creare una singola legenda per i pannelli dei componenti in un multiplot. Di seguito un esempio:




```
set multiplot layout 3,2 columnsfirst
set style data boxes
plot $D using 0:6  lt 1 title at 0.75, 0.20
plot $D using 0:12 lt 2 title at 0.75, 0.17
plot $D using 0:13 lt 3 title at 0.75, 0.14
plot $D using 0:14 lt 4 title at 0.75, 0.11
set label 1 at screen 0.75, screen 0.22 "Custom combined key area"
plot $D using 0:($6+$12+$13+$14) with linespoints title "total"
unset multiplot
```

Etichetta

Etichette arbitrarie possono essere collocate sul grafico usando il comando **set label**.

Sintassi:

```
set label {<tag>} {"<label text>"} {at <position>}
          {left | center | right}
          {norotate | rotate {by <degrees>}}
          {font "<name>{,<size>}"}
          {noenhanced}
          {front | back}
          {textcolor <colorspec>}
          {point <pointstyle> | nopoint}
          {offset <offset>}
          {nobox} {boxed {bs <boxstyle>}}
          {hypertext}
```

```
unset label {<tag>}
show label
```

La <position> (posizione) è specificata da x,y o x,y,z, e può essere preceduta da **first**, **second**, **polar**, **graph**, **screen**, o **character** per indicare il sistema di coordinate. Vedere **coordinate (p. 33)** per i dettagli.

Il tag è un intero che viene usato per identificare l'etichetta. Se nessun <tag> è dato, il valore del tag più basso non utilizzato viene assegnato automaticamente. Il tag può essere usato per cancellare o modificare un'etichetta specifica. Per cambiare qualsiasi attributo di un'etichetta esistente, è necessario usare il comando **set label** con il tag appropriato e specificare le parti dell'etichetta da cambiare.

Il <label text> può essere una costante stringa, una variabile stringa o un'espressione con valore di stringa. Vedere **strings (p. 57)**, **sprintf (p. 40)**, e **gprintf (p. 158)**.

Per default, il testo è posizionato a filo a sinistra rispetto al punto x,y,z. Per regolare il modo in cui l'etichetta è posizionata rispetto al punto x,y,z, bisogna aggiungere il parametro di giustificazione, che può essere **left**, **right** o **center**, indicando che il punto deve essere a sinistra, a destra o al centro del testo. Le etichette al di fuori dei confini plottati sono permesse ma possono interferire con le etichette degli assi o con altro testo.

Alcuni terminali supportano l'inserimento dell'etichetta in un riquadro. Vedere **set style textbox (p. 214)**. Non tutti i terminali possono gestire i riquadri per il testo ruotato.

Se è dato **rotate**, l'etichetta verrà scritta verticalmente. Se viene dato **rotate by** <**degrees**> (ruotare di <gradi>), la linea di base del testo sarà impostata all'angolo specificato. Alcuni terminali non supportano la rotazione del testo.

Il font e la sua dimensione possono essere scelti esplicitamente da **font "**<**name**>**{,**<**size**>**}"** se il terminale supporta le impostazioni del font. Altrimenti sarà usato il font predefinito del terminale.

Normalmente l'interpretazione delle stringhe in modalità di testo avanzato, se abilitata per il terminale corrente, viene applicata a tutte le stringhe di testo compreso il testo dell'etichetta. La proprietà **noenhanced** può essere usata per esentare una specifica etichetta dall'elaborazione della modalità di testo avanzato. Questo può essere utile, per esempio, se l'etichetta contiene sottolineature. Vedere **enhanced text (p. 35)**.

Se è dato **front**, l'etichetta è scritta sopra i dati riportati sul grafico. Se è dato **back** (il default), l'etichetta è scritta sotto i dati riportati sul grafico. Usando **front** si evita che l'etichetta venga oscurata da dati densi.

textcolor <**colorspec**> cambia il colore del testo dell'etichetta. <colorspec> può essere un tipo di linea, un colore rgb, o una mappatura della palette. Vedere aiuto per **colorspec (p. 50)** e **palette (p. 40)**. **textcolor** può essere abbreviato **tc**.

```
'tc default' resets the text color to its default state.
'tc lt <n>' sets the text color to that of line type <n>.
'tc ls <n>' sets the text color to that of line style <n>.
'tc palette z' selects a palette color corresponding to the label z position.
'tc palette cb <val>' selects a color corresponding to <val> on the colorbar.
'tc palette fraction <val>', with 0<=val<=1, selects a color corresponding to
    the mapping [0:1] to grays/colors of the 'palette'.
'tc rgb "#RRGGBB"' or 'tc rgb "0xRRGGBB"' sets an arbitrary 24-bit RGB color.
'tc rgb 0xRRGGBB'  As above; a hexadecimal constant does not require quotes.
```

Se è dato un <pointstyle>, usando le keyword **lt**, **pt** e **ps**, vedere **style (p. 131)**, un punto con il dato stile e colore del tipo di linea dato viene plottato nella posizione dell'etichetta e il testo dell'etichetta è leggermente spostato. Questa opzione è usata in modo predefinito per mettere le etichette nei terminali **mouse** avanzati. Usare **nopoint** per disattivare il disegno di un punto vicino all'etichetta (questo è il default).

Lo spostamento predefinito è 1,1 in unità **pointsize** se viene dato un <pointstyle>, 0,0 se non è dato alcun <pointstyle>. Lo spostamento può essere controllato dall'opzionale **offset** <**offset**> dove <offset> è specificato da x,y o x,y,z, e può essere preceduto da **first**, **second**, **graph**, **screen**, o **character** per selezionare il sistema di coordinate. Vedere **coordinate (p. 33)** per i dettagli.

Se uno (o più) assi è una serie temporale, la coordinata appropriata dovrebbe essere data come una stringa temporale tra virgolette secondo la stringa di formato **timefmt**. Vedere **set xdata (p. 224)** e **set timefmt**

(p. 218).

Le opzioni disponibili per **set label** sono disponibili anche per lo stile di grafico **labels**. Vedere **labels (p. 79)**. In questo caso le proprietà **textcolor**, **rotate**, e **pointsize** possono essere seguite dalla keyword **variable** piuttosto che da un valore fisso. In questo caso la proprietà corrispondente delle singole etichette è determinata da colonne aggiuntive nello specificatore **using**.

Esempi

Esempi:

Per impostare un'etichetta a (1,2) su "y=x", usare:

```
set label "y=x" at 1,2
```

Per impostare un Sigma di dimensione 24, dal set di font Symbol, al centro del grafico (graph), usare:

```
set label "S" at graph 0.5,0.5 center font "Symbol,24"
```

Per impostare un'etichetta "y=x^2" con la destra del testo a (2,3,4), e contrassegnare (tag) l'etichetta come numero 3, usare:

```
set label 3 "y=x^2" at 2,3,4 right
```

Per cambiare l'etichetta precedente in allineamento centrale, usare:

```
set label 3 center
```

Per cancellare l'etichetta numero 2, usare:

```
unset label 2
```

Per cancellare tutte le etichette, usare:

```
unset label
```

Per mostrare tutte le etichette (in ordine di tag), usare:

```
show label
```

Per impostare un'etichetta su un grafico (graph) con una serie temporale sull'asse x, usare, per esempio:

```
set timefmt "%d/%m/%y,%H:%M"
set label "Harvest" at "25/8/93",1
```

Per visualizzare un parametro appena adattato (fitted) sul grafico con i dati e la funzione adattata, fare questo dopo il **fit**, ma prima del **plot**:

```
set label sprintf("a = %3.5g",par_a) at 30,15
bfit = gprintf("b = %s*10^%S",par_b)
set label bfit at 30,20
```

Per visualizzare la definizione di una funzione insieme ai suoi parametri adattati, usare:

```
f(x)=a+b*x
fit f(x) 'datafile' via a,b
set label GPFUN_f at graph .05,.95
set label sprintf("a = %g", a) at graph .05,.90
set label sprintf("b = %g", b) at graph .05,.85
```

Per impostare un'etichetta un po' spostata da un piccolo punto:

```
set label 'origin' at 0,0 point lt 1 pt 2 ps 3 offset 1,-1
```

Per impostare un'etichetta il cui colore corrisponde al valore z (in questo caso 5,5) di qualche punto su uno splot 3D colorato usando pm3d:

```
set label 'text' at 0,0,5.5 tc palette z
```

Hypertext (Ipertesto)

Alcuni terminali (wxt, qt, svg, canvas, win) permettono di attaccare l'ipertesto a punti specifici sul grafico (graph) o in altre parti del canvas. Quando il mouse passa sopra il punto di ancoraggio, viene visualizzata una casella pop-up contenente il testo. I terminali che non supportano l'ipertesto non visualizzeranno nulla. È necessario abilitare l'attributo **point** dell'etichetta affinché l'ipertesto sia ancorato. Esempi:

```
set label at 0,0 "Plot origin" hypertext point pt 1
plot 'data' using 1:2:0 with labels hypertext point pt 7 \
     title 'mouse over point to see its order in data set'
```

Per i terminali wxt e qt, cliccando con il tasto sinistro del mouse su un ancoraggio ipertestuale dopo che il il testo è apparso copierà l'ipertesto negli appunti.

SPERIMENTALE (i dettagli dell'implementazione possono cambiare) Un testo della forma "image{<xsize>,<ysize>}:<filename>{n<caption text>}" attiverà la visualizzazione del file immagine in un riquadro pop-up. La dimensione opzionale sovrascrive la dimensione predefinita della casella 300x200. I tipi di file immagine riconosciuti possono variare a seconda del tipo di terminale, ma *.png dovrebbe sempre funzionare. Qualsiasi linea di testo aggiuntiva che segue il nome del file immagine sono visualizzate come al solito per l'ipertesto. Esempio:

```
set label 7 "image:../figures/Fig7_inset.png\nFigure 7 caption..."
set label 7 at 10,100 hypertext point pt 7
```

Linetype (Tipo di linea)

Il comando **set linetype** permette di ridefinire i tipi di linea di base usati per i grafici. Le opzioni di comando sono identiche a quelle di "set style line". A differenza degli stili si linea, le ridefinizioni tramite **set linetype** sono persistenti; esse non sono influenzate da **reset**.

Per esempio, qualunque sia inizialmente l'aspetto dei tipi di linea uno e due, se li si ridefinisce in questo modo:

```
set linetype 1 lw 2 lc rgb "blue" pointtype 6
set linetype 2 lw 2 lc rgb "forest-green" pointtype 8
```

ovunque si utilizzi lt ora ci sarà una spessa linea blu. Questo include usi come la definizione di un tipo di linea temporaneo derivato dal tipo di linea base 1. Allo stesso modo lt 2 ora produrrà una spessa linea verde.

Questo meccanismo può essere usato per definire un insieme di preferenze personali per la sequenza di linee usate in gnuplot. Il modo consigliato per farlo è quello di aggiungere al file di inizializzazione di run-time ~ /.gnuplot una sequenza di comandi come

```
set linetype 1 lc rgb "dark-violet" lw 2 pt 1
set linetype 2 lc rgb "sea-green"   lw 2 pt 7
set linetype 3 lc rgb "cyan"        lw 2 pt 6 pi -1
set linetype 4 lc rgb "dark-red"    lw 2 pt 5 pi -1
set linetype 5 lc rgb "blue"        lw 2 pt 8
set linetype 6 lc rgb "dark-orange" lw 2 pt 3
set linetype 7 lc rgb "black"       lw 2 pt 11
set linetype 8 lc rgb "goldenrod"   lw 2
set linetype cycle 8
```

Ogni volta che si esegue gnuplot i tipi di linea saranno inizializzati a questi valori. Si possono inizializzare quanti tipi di linea che si desiderano. Se non si ridefinisce, ad es., linetype 3, allora esso continuerà ad avere le proprietà di default (in questo caso blu, pt 3, lw 1, ecc).

File di script simili possono essere usati per definire scelte di colore basate su temi, o set di colori ottimizzati per un particolare tipo di grafico o dispositivo di output.

Il comando **set linetype cycle 8** dice a gnuplot di riutilizzare queste definizioni per il colore e lo spessore delle linee dei tipi di liena con numeri più alti. Ovvero, i tipi di linea 9-16, 17-24, e così via useranno questa stessa sequenza di colori e spessori. Le proprietà del punto (pointtype, pointsize, pointinterval) non sono influenzate da questo comando. **unset linetype cycle** disabilita questa funzionalità. Se le proprietà della linea di un tipo di linea di numero più alto sono esplicitamente definite, questo ha la precedenza sulle proprietà riciclate di un tipo di linea con numero più basso.

Link

Sintassi:

```
set link {x2 | y2} {via <expression1> inverse <expression2>}
unset link
```

Il comando **set link** stabilisce una mappatura tra gli assi x e x2, o gli assi y e y2. <expression1> mappa le coordinate dell'asse primario sull'asse secondario. <expression2> mappa le coordinate dell'asse secondario sull'asse primario.

Esempi:

```
set link x2
```

Questa è la forma più semplice del comando. Costringe l'asse x2 ad avere esattamente lo stesso intervallo, scala e direzione dell'asse x. I comandi **set xrange**, **set x2range**, **set auto x**, ecc influenzeranno entrambi gli assi x e x2.

```
set link x2 via x**2 inverse sqrt(x)
plot "sqrt_data" using 1:2 axes x2y1, "linear_data" using 1:2 axes x1y1
```

Questo comando stabilisce la mappatura in avanti e inversa tra gli assi x e x2. La mappatura in avanti è usata per generare etichette dei tic x2 e coordinate del mouse x2. La mappatura inversa è usata per plottare le coordinate date nel sistema di coordinate x2. Si noti che la mappatura data è valida solo per x non negativo. Quando si mappano sull'asse y2, sia <expression1> che <expression2> devono usare y come variabile dummy.

Lmargin

Il comando **set lmargin** imposta la dimensione del margine sinistro. Vedere **set margin (p. 176)** per i dettagli.

Loadpath

L'impostazione **loadpath** definisce posizioni aggiuntive per file di dati e di comando ricercati dai comandi **call**, **load**, **plot** e **splot**. Se un file non può essere trovato nella directory corrente, si prova a cercare nelle directory in **loadpath**.

Sintassi:

```
set loadpath {"pathlist1" {"pathlist2"...}}
show loadpath
```

I nomi dei percorsi possono essere inseriti come nomi di directory singole o come elenco di nomi di percorsi divisi da un separatore di percorso specifico della piattaforma, ad es. due punti (':') su Unix, punto e virgola (';') sulle piattaforme DOS/Windows/OS/2. I comandi **show loadpath**, **save** e **save set** sostituiscono il separatore specifico della piattaforma con un carattere di spazio (' ').

Se la variabile d'ambiente GNUPLOT_LIB è impostata, il suo contenuto viene aggiunto a **loadpath**. Tuttavia, **show loadpath** stampa il contenuto di **set loadpath** e GNUPLOT_LIB separatamente. Inoltre, i comandi **save** e **save set** ignorano il contenuto di GNUPLOT_LIB.

Locale

L'impostazione **locale** determina il linguaggio con cui **{x,y,z}{d,m}tics** scriverà i giorni e i mesi.

Sintassi:

```
set locale {"<locale>"}
```

<locale> può essere qualsiasi designazione di linguaggio accettabile per la propria installazione. Consultare la documentazione del proprio sistema per vedere le opzioni disponibili. Il comando **set locale ""** cercherà di determinare il locale dalle variabili d'ambiente LC_TIME, LC_ALL, o LANG.

Per cambiare il locale del punto decimale, vedere **set decimalsign (p. 152)**. Per cambiare la codifica dei caratteri nel locale corrente, vedere **set encoding (p. 154)**.

Logscale (Scala logaritmica)

Sintassi:

```
set logscale <axes> {<base>}
unset logscale <axes>
show logscale
```

dove <axes> può essere una qualsiasi combinazione di **x**, **x2**, **y**, **y2**, **z**, **cb**, e **r** in qualsiasi ordine. <base> è la base del log scaling (il default è base 10). Se non viene specificato alcun asse, il comando ha effetto su tutti gli assi eccetto **r**. Il comando **unset logscale** disattiva il log scaling per tutti gli assi. Si noti che i segni di tic generati per gli assi logscalati non sono distanziati uniformemente. Vedere **set xtics (p. 228)**.

Esempi:

Per abilitare il log scaling in entrambi gli assi x e z:

```
set logscale xz
```

Per abilitare il log scaling in base 2 dell'asse y:

```
set logscale y 2
```

Per abilitare gli assi log z e colore per un grafico pm3d:

```
set logscale zcb
```

Per disabilitare il log scaling dell'asse z:

```
unset logscale z
```

Macro

In questa versione di gnuplot la sostituzione delle macro è sempre abilitata. I token nella linea di comando della forma @<stringavariablename> saranno sostituiti dalla stringa di testo contenuta in <stringavariablename>. Vedere **sostituzione (p. 59)**.

Mapping

Se i dati vengono forniti a **splot** in coordinate sferiche o cilindriche, il comando **set mapping** dovrebbe essere usato per istruire **gnuplot** su come interpretarli.

Sintassi:

```
set mapping {cartesian | spherical | cylindrical}
```

Viene usato per default un sistema di coordinate cartesiane.

Per un sistema di coordinate sferiche, i dati occupano due o tre colonne (o entry **using**). Le prime due sono interpretate come gli angoli azimutale e polare: theta e phi (o "longitudine" e "latitudine"), nelle unità specificate da **set angles**. Il raggio r è preso dalla terza colonna se è presente, o è fissato all'unità se non c'è una terza colonna. La mappatura è:

```
x = r * cos(theta) * cos(phi)
y = r * sin(theta) * cos(phi)
z = r * sin(phi)
```

Si noti che questo è un sistema sferico "geografico", piuttosto che uno "polare" (cioè, il phi è misurato dall'equatore, piuttosto che dal polo).

Per un sistema di coordinate cilindriche, i dati occupano di nuovo due o tre colonne. Le prime due sono interpretate come theta (nelle unità specificate da **set angles**) e z. Il raggio è preso dalla terza colonna o fissato all'unità, come nel caso sferico. La mappatura è:

```
x = r * cos(theta)
y = r * sin(theta)
z = z
```

Gli effetti del **mapping** possono essere duplicati con il filtro **using** del comando **splot**, ma il **mapping** può essere più conveniente se devono essere elaborati molti file di dati. Tuttavia, anche se si usa il **mapping**, **using** può essere ancora necessario se i dati nel file non sono nell'ordine richiesto.

mapping non ha alcun effetto su **plot**. `world.dem: mapping demos.`

Margin

Il **margin** è la distanza tra il bordo del grafico e il confine esterno del canvas. La dimensione del margine viene scelta automaticamente, ma può essere sovrascritta dai comandi **set margin**. **show margin** mostra le impostazioni in vigore. Per modificare la distanza tra l'interno del bordo del grafico e i dati nel grafico stesso, vedere **set offsets** **(p. 187)**.

Sintassi:

```
set lmargin {{at screen} <margin>}
set rmargin {{at screen} <margin>}
set tmargin {{at screen} <margin>}
set bmargin {{at screen} <margin>}
set margins <left>, <right>, <bottom>, <top>
show margin
```

Le unità predefinite di <margin> sono le altezze o le larghezze dei caratteri, ove opportuno. Un valore positivo definisce la dimensione assoluta del margine. Un valore negativo (o nessun valore) fa sì che **gnuplot** torni al valore computato. Per i grafici 3D, solo il margine sinistro può essere impostato usando le unità di caratteri.

La keyword **at screen** indica che il margine è specificato come una frazione dell'intera area di disegno. Questo può essere usato per allineare con precisione gli angoli di singoli grafici (graph) 2D e 3D in un multiplot. Questo posizionamento ignora i valori attuali di **set origin** e **set size**, ed è inteso come un metodo alternativo per posizionare i grafici (graph) all'interno di un multiplot.

Normalmente i margini di un grafico sono calcolati automaticamente in base ai tic, alle etichette dei tic, alle etichette degli assi, al titolo del grafico, alla marca temporale e alla dimensione della chiave se si trova al di fuori dei margini. Se, tuttavia, i tic sono attaccati agli assi (per esempio, **set xtics axis**), né i tic stessi né le loro etichette saranno inclusi nel calcolo dei margini o nel calcolo delle posizioni di altro testo da scrivere nel margine. Questo può portare a etichette dei tic che sovrascrivono altro testo se l'asse è molto vicino al bordo.

Micro

Per impostazione predefinita, lo specificatore di formato "%c" per la notazione scientifica usato per generare etichette dei tic degli assi, utilizza una u minuscola come prefisso per indicare "micro" (10^-6). Il comando **set micro** dice a gnuplot di usare un carattere tipografico diverso (unicode U+00B5). La sequenza di byte usata per rappresentare questo carattere dipende dalla codifica corrente. Vedere **specificatori di formato (p. 158)**, **encoding (p. 154)**.

Questo comando è SPERIMENTALE. I dettagli di implementazione possono cambiare.

Minussign (Segno meno)

Gnuplot utilizza la routine sprintf() della libreria del linguaggio C per la maggior parte degli input formattati. Tuttavia ha anche la propria routine di formattazione **gprintf()** che è usata per generare le etichette tic degli assi. La routine della libreria C usa sempre un carattere trattino (hyphen) (ascii \055) per indicare un numero negativo, come in -7. Molte persone preferiscono per questo scopo un diverso carattere tipografico del segno meno (unicode U+2212), come in −7. Il comando

```
set minussign
```

fa sì che gprintf() usi questo carattere del segno meno, invece di un trattino, nell'output numerico. In un locale utf-8 questa è la sequenza multibyte corrispondente all'unicode U+2212. In un locale Window codepage 1252 questo è il carattere 8-bit ALT+150 ("en dash" lineetta). Il comando **set minussign** influenzerà le etichette dei tic degli assi e qualsiasi etichetta creata invocando esplicitamente gprintf. Non ha effetto su altre stringhe che contengono un trattino. Vedere **gprintf (p. 158)**.

Notare che questo comando è ignorato quando si sta utilizzando uno qualsiasi dei terminali LaTeX poiché LaTeX ha un proprio meccanismo per gestire i segni meno. Inoltre non è necessario quando si usa il terminale postscript perché il prologo postscript prodotto da gnuplot rimappa il codice ascii del trattino \055 in un diverso glifo chiamato **minus**.

Questo comando è SPERIMENTALE. I dettagli di implementazione possono cambiare.

Esempio (presuppone il locale utf8):

```
set minus
A = -5
print "A = ",A                 # la stringa stampata conterrà un trattino
print gprintf("A = %g",A)      # la stringa stampata conterrà il carattere U+2212
set label "V = -5"             # l'etichetta conterrà un trattino
set label sprintf("V = %g",-5) # l'etichetta conterrà un trattino
set label gprintf("V = %g",-5) # l'etichetta conterrà il carattere U+2212
```

Monochrome

Sintassi:

```
set monochrome {linetype N <linetype properties>}
```

Il comando **set monochrome** seleziona un set alternativo di tipi di linea che differiscono per il pattern di punti/linee o per lo spessore della linea piuttosto che per il colore. Questo comando sostituisce l'opzione monocromatica offerta da alcuni tipi di terminale nelle precedenti versioni di gnuplot. Per compatibilità con le versioni precedenti questi tipi di terminale ora invocano implicitamente "set monochrome" se la loro opzione "mono" è presente. Ad esempio,

```
set terminal pdf mono
```

è equivalente a

```
set terminal pdf
set mono
```

Selezionare la modalità monocromatica non impedisce di disegnare esplicitamente linee usando colori RGB o della palette, ma si consiglia di vedere anche **set palette gray (p. 193)**. Per impostazione predefinita sono definiti sei tipi di linea monocromatici. È possibile cambiare le loro proprietà o aggiungere ulteriori tipi di linea monocromatici usando la forma completa del comando. Le modifiche fatte ai tipi di linea monocromatici non hanno effetto sui tipi di linea a colori e viceversa. Per ripristinare il solito insieme di tipi di linea a colori, usare o **unset monochrome** o **set color**.

Mouse

Il comando **set mouse** abilita le azioni del mouse per il terminale interattivo corrente. Solitamente è abilitato di default in modalità interattiva, ma disabilitato di default se i comandi vengono letti da un file.

Ci sono due modalità di utilizzo del mouse. La modalità 2D funziona per i comandi **plot** e per le mappe **splot** (cioè **set view** con rotazione z 0, 90, 180, 270 o 360 gradi, incluso **set view map**). In questa modalità la posizione del mouse viene tracciata e si può fare una panoramica o uno zoom usando i pulsanti del mouse o i tasti freccia. Alcuni terminali supportano l'attivazione/disattivazione di grafici individuali cliccando sul titolo key corrispondente o su un widget separato.

Per grafici (graph) 3D **splot**, la visualizzazione e il ridimensionamento del grafico (graph) possono essere cambiati con i pulsanti del mouse 1 e 2, rispettivamente. Un movimento verticale del pulsante 2 con il tasto shift tenuto premuto cambia il **xyplane** (piano xy). Se oltre a questi tasti viene tenuto premuto anche il modificatore <ctrl>, gli assi delle coordinate vengono visualizzati ma i dati sono soppressi. Questo è utile per grandi data set. Il tasto 3 del mouse controlla l'azimut dell'asse z (vedere **set view azimuth (p. 222)**).

Il mouse non è disponibile all'interno della modalità multiplot. Quando il multiplot viene completato usando **unset multiplot**, allora il mouse sarà riattivato ma agirà solo sul grafico più recente all'interno del multiplot (come fa replot).

Sintassi:

```
set mouse {doubleclick <ms>} {nodoubleclick}
          {{no}zoomcoordinates}
          {zoomfactors <xmultiplier>, <ymultiplier>}
          {noruler | ruler {at x,y}}
          {polardistance{deg|tan} | nopolardistance}
          {format <string>}
          {mouseformat <int> | <string> | function <f(x,y)>}
          {{no}labels {"labeloptions"}}
          {{no}zoomjump} {{no}verbose}
unset mouse
```

Le opzioni **noruler** e **ruler** disattivano e attivano il righello, quest'ultima imposta su richiesta l'origine alle coordinate date. Mentre il righello è attivo, la distanza in unità utente dall'origine del righello al mouse viene visualizzata costantemente. Per default, il righello viene attivato/disattivato con il tasto 'r'.

L'opzione **polardistance** determina se la distanza tra il cursore del mouse e il righello viene mostrata anche in coordinate polari (distanza e angolo in gradi o tangente (pendenza)). Questo corrisponde all'associazione predefinita con il tasto '5'.

È necessario scegliere l'opzione **labels** per definire le etichette persistenti di gnuplot usando il pulsante 2. Il valore predefinito è **nolabels**, che fa sì che il pulsante 2 disegni solo un'etichetta temporanea nella posizione del mouse. Le etichette sono disegnate con l'impostazione vigente di **mouseformat**. La stringa **labeloptions** viene passata al comando **set label**. L'impostazione predefinita è "point pointstyle 1" che plotterà un piccolo più nella posizione dell'etichetta. Le etichette temporanee scompariranno al prossimo **replot** o alla prossima operazione di zoom del mouse. Le etichette persistenti possono essere rimosse tenendo

premuto il tasto Ctrl mentre si clicca il pulsante 2 sul punto dell'etichetta. La soglia per quanto si deve essere vicini all'etichetta è determinata anch'essa dal **pointsize**.

Se l'opzione **verbose** è attivata, i comandi di comunicazione sono mostrati durante l'esecuzione. Questa opzione può anche essere attivata o disattivata premendo **6** nella finestra del driver. **verbose** è disattivata di default.

Per un breve riassunto delle associazioni (binding) di mouse e tasti premere 'h' nella finestra del driver. Questo visualizzerà anche le associazioni definite dall'utente o gli **hotkeys** (tasti di scelta rapida) che possono essere definiti usando il comando **bind**, vedere aiuto per **bind (p. 54)**. Si noti che gli **hotkeys** definiti dall'utente possono sovrascrivere le associazioni predefinite. Vedere aiuto anche per **bind (p. 54)** e **label (p. 170)**.

Doubleclick (Doppio click)

La risoluzione del doppio clic è data in millisecondi ed è usata per il pulsante 1, che copia la posizione attuale del mouse nella **clipboard** (appunti) su alcuni terminali. Il valore predefinito è 300 ms. Impostando il valore a 0 ms si attiva la copia su un singolo clic.

Format

Il comando **set mouse format** specifica una stringa di formato per sprintf() che determina il modo in cui le coordinate del cursore del mouse [x,y] sono stampate nella finestra del grafico e negli appunti. Il default è `"% #g"`.

Questa impostazione è sostituita da `"set mouse mouseformat"`.

Mouseformat

Sintassi:

```
set mouse mouseformat i
set mouse mouseformat "custom format"
set mouse mouseformat function string_valued_function(x, y)
```

Questo comando controlla il formato usato per riferire la posizione attuale del mouse. Un argomento intero seleziona una delle opzioni di formato nella tabella sottostante. Un argomento stringa è usato come formato per sprintf() nell'opzione 7 e dovrebbe contenere due specificatori float, uno per x e uno per y.

L'uso di una funzione personalizzata che restituisce una stringa è SPERIMENTALE. Permette la lettura di sistemi di coordinate in cui la mappatura inversa dalle coordinate dello schermo alle coordinate del grafico richiede la considerazione congiunta di x e y. Vedere per esempio la demo map_projection.

Esempio:

```
'set mouse mouseformat "mouse x,y = %5.2g, %10.3f"'.
```

Usare **set mouse mouseformat** `""` per spegnere di nuovo questa stringa.

Sono disponibili i seguenti formati:

```
0   default (come 1)
1   coordinate degli assi                          1.23, 2.45
2   coordinate del grafico (graph) (da 0 a 1)    /0.00, 1.00/
3   x = timefmt    y = asse                      [(come imposato da 'set timefmt'), 2.45]
4   x = data       y = asse                      [31. 12. 1999, 2.45]
5   x = ora        y = asse                      [23:59, 2.45]
6   x = data ora   y = asse                      [31. 12. 1999 23:59, 2.45]
7   formato da 'set mouse mouseformat <format-string>'
8   formato da 'set mouse mouseformat function <func>'
```

Scrolling

Il ridimensionamento degli assi X e Y nei grafici (graph) 2D e 3D può essere regolato usando la rotella del mouse. <wheel-up> scorre verso l'alto (aumenta sia YMIN che YMAX del dieci per cento dellintervallo Y, e aumenta sia Y2MIN che Y2MAX allo stesso modo), e <wheel down> scorre verso il basso. <shift-wheel-up> scorre verso sinistra (diminuisce sia XMIN e XMAX, e sia X2MIN che X2MAX), e <shift-wheel-down> scorre verso destra. <control-wheel-up> aumenta lo zoom verso il centro del grafico, e <control-wheel-down> diminuisce lo zoom. <shift-control-wheel-up> aumenta lo zoom solo lungo gli assi X e X2, e <shift-control-wheel-down> diminuisce lo zoom solo lungo gli assi X e X2.

X11 mouse

Se sono state aperte più finestre di grafici X11 usando l'opzione terminale **set term x11 <n>**, allora solo la finestra del grafico corrente supporta l'intera gamma di comandi del mouse e tasti di scelta rapida. Le altre finestre, tuttavia, continueranno a mostrare le coordinate del mouse in basso a sinistra.

Zoom

Lo zoom viene solitamente effettuato tenendo premuto il tasto sinistro del mouse e trascinando il mouse per delineare una regione di zoom. Alcune piattaforme possono richiedere l'uso di un diverso pulsante del mouse. Il grafico originale può essere ripristinato digitando il tasto di scelta rapida 'u' nella finestra del grafico. I tasti di scelta rapida 'p' e 'n' vanno avanti e indietro attraverso una cronologia di operazioni di zoom.

L'opzione **zoomcoordinates** determina se le coordinate della casella dello zoom sono disegnate sui bordi durante lo zoom. Questo è attivo per default.

Se l'opzione **zoomjump** è attiva, il puntatore del mouse sarà automaticamente spostato di una piccola distanza dopo aver selezionato una regione di zoom con il tasto 3. Questo può essere utile per evitare una regione di zoom minuscola (o addirittura vuota). **zoomjump** è disattivato di default.

Mttics

I segni di tic secondari intorno al perimetro di un grafico polare sono controllati da **set mttics**. Vedere **set mxtics (p. 182)**.

Multiplot

Il comando **set multiplot** pone **gnuplot** in modalità multiplot, in cui diversi grafici sono posti uno accanto all'altro sulla stessa pagina o finestra dello schermo.

Sintassi:

```
set multiplot
    { title <page title> {font <fontspec>} {enhanced|noenhanced} }
    { layout <rows>,<cols>
      {rowsfirst|columnsfirst} {downwards|upwards}
      {scale <xscale>{,<yscale>}} {offset <xoff>{,<yoff>}}
      {margins <left>,<right>,<bottom>,<top>}
      {spacing <xspacing>{,<yspacing>}}
    }
set multiplot {next|previous}
unset multiplot
```

Per alcuni terminali, nessun grafico viene visualizzato fino a quando non viene dato il comando **unset multiplot**, il quale fa sì che l'intera pagina sia disegnata e poi riporta gnuplot alla sua normale modalità a singolo grafico. Per altri terminali, ogni comando **plot** distinto produce una visualizzazione aggiornata.

Il comando **clear** è usato per cancellare l'area rettangolare della pagina che sarà usata per il prossimo grafico. Questo è necessario tipicamente per inserire un piccolo grafico all'interno di un grafico più grande.

Tutte le etichette o le frecce che sono state definite saranno disegnate per ogni grafico secondo la dimensione e l'origine attuali (a meno che le loro coordinate non siano definite nel sistema **screen**). Quasi tutto il resto che può essere impostato (**set**) viene applicato anche ad ogni grafico. Se si desidera che qualcosa appaia una sola volta nella pagina, per esempio una singola marca temporale, sarà necessario mettere una coppia **set time**/**unset time** attorno a uno dei comandi **plot**, **splot** o **replot** all'interno del blocco **set multiplot**/**unset multiplot**.

Il titolo multiplot è separato dai titoli dei singoli grafici, se presenti. Lo spazio riservato ad esso è nella parte superiore della pagina, e si estende per tutta la larghezza del canvas.

I comandi **set origin** e **set size** devono essere usati per posizionare correttamente ogni grafico se non viene specificato alcun layout o se si desidera una regolazione precisa. Vedere **set origin (p. 188)** e **set size (p. 205)** per i dettagli sul loro uso.

Esempio:

```
set multiplot
set size 0.4,0.4
set origin 0.1,0.1
plot sin(x)
set size 0.2,0.2
set origin 0.5,0.5
plot cos(x)
unset multiplot
```

Questo mostra un grafico di cos(x) impilato sopra un grafico di sin(x).

set size e **set origin** si riferiscono all'intera area di plotting usata per ogni grafico. Vedere anche **set term size (p. 32)**. Se si desidera che gli assi siano allineati, è possibile garantire che i margini siano della stessa dimensione con i comandi **set margin**. Vedere **set margin (p. 176)** per il loro uso. Si noti che le impostazioni dei margini sono assolute, in unità di carattere, quindi l'aspetto del grafico (graph) nello spazio rimanente dipenderà dalle dimensioni dello schermo del dispositivo di visualizzazione, ad es., forse molto diverso su uno schermo video e su una stampante.

Con l'opzione **layout** si possono generare semplici multiplot senza dover dare i comandi **set size** e **set origin** prima di ogni grafico: Questi sono generati automaticamente, ma possono essere sovrascritti in qualsiasi momento. Con **layout** il display sarà diviso da una griglia con righe <rows> e colonne <cols>. Questa griglia viene riempita prima con le righe o prima con le colonne, a seconda se l'opzione corrispondente è data nel comando multiplot. La pila di grafici può crescere **downwards** (verso il basso) o **upwards** (verso l'alto). L'impostazione predefinita è **rowsfirst** (prima le righe) e **downwards**. I comandi **set multiplot next** e **set multiplot previous** sono rilevanti solo nel contesto dell'utilizzo dell'opzione di layout. **next** salta la posizione successiva nella griglia, lasciando uno spazio vuoto. **prev** ritorna alla posizione della griglia immediatamente precedente all'ultima posizione plottata.

Ogni grafico può essere ridimensionato con **scale** e spostato con **offset**; se i valori y per la scala o l'offset sono omessi, verrà utilizzato il valore di x. **unset multiplot** disattiva il layout automatico e ripristina i valori di **set size** e **set origin** come erano prima di **set layout multiplot**.

Esempio:

```
set size 1,1
set origin 0,0
set multiplot layout 3,2 columnsfirst scale 1.1,0.9
[ up to 6 plot commands here ]
unset multiplot
```

L'esempio precedente produrrà 6 grafici in 2 colonne riempite dall'alto verso il basso, da sinistra a destra. Ogni grafico avrà una dimensione orizzontale di 1.1/2 e una verticale di 0.9/3.

Un'altra possibilità è quella di impostare margini uniformi per tutti i grafici nel layout con le opzioni **layout margins** e **spacing**, che devono essere usate insieme. Con **margins** si impostano i margini esterni dell'intera griglia multiplot.

spacing fornisce la dimensione dello spazio tra due subplot adiacenti, e può anche essere data in unità di **character** o **screen**. Se viene dato un singolo valore, esso viene utilizzato per entrambe le direzioni x e y, altrimenti possono essere selezionati due valori diversi.

Se un valore non ha unità, viene usata quella dell'impostazione del margine precedente.

Esempio:

```
set multiplot layout 2,2 margins 0.1, 0.9, 0.1, 0.9 spacing 0.0
```

In questo caso i due subplot più a sinistra avranno i confini di sinistra alla coordinata dello schermo 0,1, i due subplot più a destra avranno i confini di destra alla coordinata dello schermo 0,9, e così via. Poiché la distanza tra i subplot è data come 0, i loro confini interni si sovrappongono.

Esempio:

```
set multiplot layout 2,2 margins char 5,1,1,2 spacing screen 0, char 2
```

Questo produce un layout in cui il confine di entrambi i subplot di sinistra è a 5 larghezze di carattere dal bordo sinistro del canvas, e il confine destro dei subplot di destra è a 1 larghezza di carattere dal bordo del canvas. Il margine inferiore complessivo è di un'altezza di carattere e il margine superiore complessivo è di 2 altezze di carattere. Non c'è uno spazio orizzontale tra le due colonne di subplot. Lo spazio verticale tra i subplot è uguale a 2 altezze di carattere.

Esempio:

```
set multiplot layout 2,2 columnsfirst margins 0.1,0.9,0.1,0.9 spacing 0.1
set ylabel 'ylabel'
plot sin(x)
set xlabel 'xlabel'
plot cos(x)
unset ylabel
unset xlabel
plot sin(2*x)
set xlabel 'xlabel'
plot cos(2*x)
unset multiplot
```

Vedere anche `demo multiplot (multiplt.dem)`

Mx2tics

I tic secondari lungo l'asse x2 (in alto) sono controllati da **set mx2tics**. Vedere **set mxtics (p. 182)**.

Mxtics

I tic secondari lungo l'asse x sono controllati da **set mxtics**. Possono essere disattivati con **unset mxtics**. Comandi simili controllano tic secondari lungo gli altri assi.

Sintassi:

```
set mxtics {<freq> | default}
unset mxtics
show mxtics
```

La stessa sintassi si applica a **mytics**, **mztics**, **mx2tics**, **my2tics**, **mrtics**, **mttics** e **mcbtics**.

<freq> è il numero di sottointervalli (NON il numero di tic secondari) tra i tic principali (l'impostazione predefinita per un asse lineare è di due o cinque a seconda dei tic principali, quindi ci sono uno o quattro tic secondari tra i tic principali). Selezionando **default** si riporta il numero di tic secondari al suo valore predefinito.

Se l'asse è logaritmico, il numero di sottointervalli sarà impostato di default su un numero ragionevole (basato sulla lunghezza di un decennio). Questo sarà sovrascritto se viene data <freq>. Tuttavia i soliti tic secondari (2, 3, ..., 8, 9 tra 1 e 10, per esempio) si ottengono impostando <freq> a 10, anche se ci sono solo nove sottointervalli.

Per impostare tic secondari in posizioni arbitrarie, bisogna usare la forma di **set {x|x2|y|y2|z}tics** con <label> vuoto e <level> impostato su 1.

I comandi **set m{x|x2|y|y2|z}tics** funzionano solo quando ci sono tic principali uniformemente distanziati. Se tutti i tic principali sono stati posti esplicitamente da **set {x|x2|y|y2|z}tics**, allora i comandi dei tic secondari sono ignorati. I tic principali impliciti e i tic secondari espliciti possono essere combinati usando **set {x|x2|y|y2|z}tics** e **set {x|x2|y|y2|z}tics add**.

Esempi:

```
set xtics 0, 5, 10
set xtics add (7.5)
set mxtics 5
```

Tic principali a 0,5,7.5,10, tic secondari a 1,2,3,4,6,7,8,9

```
set logscale y
set ytics format ""
set ytics 1e-6, 10, 1
set ytics add ("1" 1, ".1" 0.1, ".01" 0.01, "10^-3" 0.001, \
               "10^-4" 0.0001)
set mytics 10
```

Tic principali con formattazione speciale, tic secondari in posizioni logaritmiche

Per default, i tic secondari sono disattivati per gli assi lineari e attivati per gli assi logaritmici. I tic secondari ereditano le impostazioni per **axis|border** e **{no}mirror** specificate per i tic principali. Vedere **set xtics (p. 228)** per informazioni su di essi.

My2tics

I segni di tic secondari lungo l'asse y2 (a destra) sono controllati da **set my2tics**. Si prega di vedere **set mxtics (p. 182)**.

Mytics

I segni di tic secondari lungo l'asse y sono controllati da **set mytics**. Vedere **set mxtics (p. 182)**.

Mztics

I segni di tic secondari lungo l'asse z sono controllati da **set mztics**. Vedere **set mxtics (p. 182)**.

Nonlinear

Sintassi:

```
set nonlinear <axis> via f(axis) inverse g(axis)
unset nonlinear <axis>
```

Questo comando è simile al comando **set link** eccetto che solo uno dei due assi collegati è visibile. L'asse nascosto rimane lineare. Le coordinate lungo l'asse visibile sono mappate applicando g(x) alle coordinate dell'asse nascosto. f(x) mappa le coordinate dell'asse visibile sull'asse lineare nascosto. È necessario fornire sia l'espressione diretta che quella inversa.

Per illustrare come funziona, si consideri il caso di un asse x2 su scala logaritmica.

```
set x2ange [1:1000]
set nonlinear x2 via log10(x) inverse 10**x
```

Questo ottiene lo stesso effetto di **set log x2**. L'asse nascosto in questo caso ha intervallo [0:3], ottenuto calcolando [log10(xmin):log10(xmax)].

Le funzioni di trasformazione f() e g() devono essere definite utilizzando una variabile dummy appropriata all'asse non lineare:

```
axis: x x2   dummy variable x
axis: y y2   dummy variable y
axis: z cb   dummy variable z
axis: r      dummy variable r
```

Esempio:

```
set xrange [-3:3]
set nonlinear x via norm(x) inverse invnorm(x)
```

Questo esempio stabilisce un asse x con scala di probabilità ("probit"), tale che plottare la normale funzione cumulativa Phi(x) produce un grafico a linea retta contro un asse y lineare.

Esempio:

```
logit(p) = log(p/(1-p))
logistic(a) = 1. / (1. + exp(-a))
set xrange [.001 : .999]
set nonlinear y via logit(y) inverse logistic(y)
plot logit(x)
```

Questo esempio stabilisce un asse y con scala logit tale che plottare logit(x) su un asse x lineare produce un grafico a linea retta.

Esempio:

```
f(x) = (x <= 100) ? x : (x < 500) ? NaN : x-390
g(x) = (x <= 100) ? x : x+390
set xrange [0:1000] noextend
set nonlinear x via f(x) inverse g(x)
set xtics add (100,500)
plot sample [x=1:100] x, [x=500:1000] x
```

Questo esempio crea un "asse rotto". Le coordinate X 0-100 sono a sinistra, le coordinate X 500-1000 sono a destra, c'è un piccolo spazio (10 unità) tra di loro. Questo funziona come previsto finché non vengono plottati data point con ($100 < x < 500$).

Object

Il comando **set object** definisce un singolo oggetto che apparirà in grafici successivi. Si possono definire quanti oggetti si desidera. Attualmente i tipi di oggetto supportati sono **rectangle**, **circle**, **ellipse**, e **polygon**. I rettangoli ereditano un set predefinito di proprietà di stile (riempimento, colore, bordo) da quelle impostate dal comando **set style rectangle**, ma ad ogni oggetto possono anche essere date proprietà di stile individuali. Cerchi, ellissi e poligoni ereditano lo stile di riempimento da **set style fill**. Gli oggetti

da disegnare nei grafici 2D possono essere definiti in qualsiasi combinazione di coordinate di assi, di grafici, polari o dello schermo.

Le specificazioni degli oggetti nei grafici (plot) 3D non possono usare le coordinate del grafico (graph). I rettangoli e le ellissi nei grafici 3D sono limitati alle coordinate dello schermo.

Sintassi:

```
set object <index>
    <object-type> <object-properties>
    {front|back|behind|depthorder}
    {clip|noclip}
    {fc|fillcolor <colorspec>} {fs <fillstyle>}
    {default} {lw|linewidth <width>} {dt|dashtype <dashtype>}
unset object <index>
```

<object-type> è o **rectangle**, **ellipse**, **circle**, o **polygon**. Ciascun tipo di oggetto ha il suo insieme di proprietà caratteristiche.

Le opzioni **front**, **back**, **behind** controllano se l'oggetto è disegnato prima o dopo il grafico stesso. Vedere **layers (p. 53)**. Impostando **front** l'oggetto verrà disegnato davanti a tutti gli elementi del grafico, ma dietro a qualsiasi etichetta che è anch'essa marcata **front**. Impostando **back** l'oggetto verrà posizionato dietro tutte le curve e le etichette del grafico. Impostando **behind** si posizionerà l'oggetto dietro a tutto, compresi gli assi e i rettangoli **back**, quindi

```
set object rectangle from screen 0,0 to screen 1,1 behind
```

può essere usato per fornire uno sfondo colorato per l'intero grafico (graph) o pagina.

Per default, gli oggetti sono ritagliati sul confine del grafico (graph) a meno che uno o più vertici siano dati in coordinate dello schermo. Impostando **noclip** si disabilita il ritaglio al confine del grafico (graph), ma continuerà a ritagliare contro la dimensione dello schermo.

Il colore di riempimento dell'oggetto è preso da <colorspec>. **fillcolor** può essere abbreviato **fc**. Lo stile di riempimento è preso da <fillstyle>. Vedere **colorspec (p. 50)** e **fillstyle (p. 209)**. Se viene data la keyword **default**, queste proprietà sono ereditate dalle impostazioni predefinite nel momento in cui viene disegnato un grafico. Vedere **set style rectangle (p. 212)**.

Rectangle (Rettangolo)

Sintassi:

```
set object <index> rectangle
    {from <position> {to|rto} <position> |
     center <position> size <w>,<h> |
     at <position> size <w>,<h>}
```

La posizione del rettangolo può essere specificata dando la posizione di due angoli diagonali (in basso a sinistra e in alto a destra) o dando la posizione del centro seguita dalla larghezza e dall'altezza. In entrambi i casi le posizioni possono essere date in coordinate degli assi, del grafico (graph) o dello schermo. Vedere **coordinate (p. 33)**. Le opzioni **at** e **center** sono sinonimi.

Esempi:

```
# Forza l'intera area racchiusa dagli assi ad avere il colore di sfondo ciano
set object 1 rect from graph 0, graph 0 to graph 1, graph 1 back
set object 1 rect fc rgb "cyan" fillstyle solid 1.0

# Posiziona un quadrato rosso con coordinate in basso a sinistra a 0,0
# e in alto a destra a 2,3
set object 2 rect from 0,0 to 2,3 fc lt 1
```

```
# Posiziona un rettangolo vuoto (senza riempimento) con un bordo blu
set object 3 rect from 0,0 to 2,3 fs empty border rgb "blue"

# Riporta il riempimento e il colore allo stile predefinito ma lascia
# i vertici invariati
set object 2 rect default
```

Gli angoli dei rettangoli specificati in coordinate dello schermo possono estendersi oltre il bordo del il grafico (graph) corrente. Altrimenti il rettangolo viene ritagliato per adattarsi al grafico (graph).

Ellipse (Ellisse)

Sintassi:

```
set object <index> ellipse {at|center} <position> size <w>,<h>
    {angle <orientation>} {units xy|xx|yy}
    {<other-object-properties>}
```

La posizione dell'ellisse è specificata dando il centro seguito dalla larghezza e l'altezza (in realtà gli assi maggiore e minore). Le keyword **at** e **center** sono sinonimi. La posizione centrale può essere data in coordinate degli assi, grafico (graph) o dello schermo. Vedere **coordinate (p. 33)**. Le lunghezze dell'asse maggiore e minore devono essere date in coordinate degli assi. L'orientamento dell'ellisse è specificato dall'angolo tra l'asse orizzontale e il diametro maggiore dell'ellisse. Se non viene dato alcun angolo, al suo posto sarà usato l'orientamento predefinito dell'ellisse (vedere **set style ellipse (p. 213)**). La keyword **units** controlla il ridimensionamento degli assi dell'ellisse. **units xy** significa che l'asse maggiore è interpretato in termini di unità lungo l'asse x, mentre l'asse asse minore in quello dell'asse y. **units xx** significa che entrambi gli assi delle ellissi sono scalati nelle unità dell'asse x, mentre **units yy** significa che entrambi gli assi sono in unità dell'asse y. Il valore predefinito è **xy** o qualsiasi altra cosa su cui è stato impostato **set style ellipse units**.

NB: Se le scale degli assi x e y non sono uguali, (ad esempio **units xy** è in vigore) allora il rapporto asse maggiore/minore non sarà più corretto dopo rotazione.

Si noti che **set object ellipse size <2r>,<2r>** non produce in generale lo stesso risultato di **set object circle <r>**. Il raggio del cerchio è sempre interpretato in termini di unità lungo l'asse x, e produrrà sempre un cerchio anche se le scale degli assi x e y sono diverse e anche se il rapporto d'aspetto del grafico non è 1. Se **units** è impostato su **xy**, allora 'set object ellipse' interpreta il primo <2r> in termini di unità dell'asse x e il secondo <2r> in termini di unità dell'asse y. Questo produrrà un cerchio solo se le scale degli assi x e y sono identiche e il rapporto d'aspetto del grafico è 1. D'altra parte, se **units** è impostato a **xx** o **yy**, allora i diametri specificati nel comando 'set object' saranno interpretati nelle stesse unità, così l'ellisse avrà il corretto rapporto d'aspetto, e manterrà il suo rapporto d'aspetto anche se il grafico viene ridimensionato.

Circle (Cerchio)

Sintassi:

```
set object <index> circle {at|center} <position> size <radius>
    {arc [<begin>:<end>]} {no{wedge}}
    {<other-object-properties>}
```

La posizione del cerchio è specificata dando la posizione del centro seguita dal raggio. Le keywords **at** e **center** sono sinonimi. Nei grafici 2D la posizione e il raggio possono essere dati in qualsiasi sistema di coordinate. Vedere **coordinate (p. 33)**. I cerchi nei grafici 3D non possono usare le coordinate del grafico (graph). In tutti i casi il raggio è calcolato rispetto alla scala orizzontale dell'asse, del grafico (graph) o del canvas. Qualsiasi disparità tra la scala orizzontale e quella verticale verrà corretta in modo che il risultato sia sempre un cerchio. Se si vuole disegnare un cerchio nelle coordinate del grafico (in modo che appaia come un'ellisse se la scala orizzontale e verticale sono diverse), usare invece **set object ellipse**.

Per default viene disegnato un cerchio completo. Il qualificatore opzionale **arc** specifica un angolo iniziale e un angolo finale, in gradi, per un arco di cerchio. L'arco è sempre disegnato in senso antiorario.

Vedere anche **set style circle (p. 212)**, **set object ellipse (p. 186)**.

Polygon (Poligono)

Sintassi:

```
set object <index> polygon
    from <position> to <position> ... {to <position>}
```

o

```
    from <position> rto <position> ... {rto <position>}
```

La posizione del poligono può essere specificata fornendo la posizione di una sequenza di vertici. Questi possono essere dati in qualsiasi sistema di coordinate. Se si usano coordinate relative (rto) allora il tipo di coordinate deve corrispondere a quello del vertice precedente. Vedere **coordinate (p. 33)**.

Esempio:

```
set object 1 polygon from 0,0 to 1,1 to 2,0
set object 1 fc rgb "cyan" fillstyle solid 1.0 border lt -1
```

Depthorder L'opzione **set object N depthorder** si applica solo agli oggetti poligonali 3D. Invece di assegnare l'oggetto al layer front/back/behind esso viene incluso nella lista dei quadrangoli pm3d ordinati e renderizzati in ordine di profondità da **set pm3d depthorder**. Come per le superfici pm3d, la colorazione su due lati può essere generata specificando il colore di riempimento dell'oggetto come uno stile di linea. In questo caso l'ordine dei primi tre vertici del poligono determina il "lato".

Se si imposta questa proprietà per un oggetto che non è un poligono 3D, probabilmente non sarà disegnato.

Offsets

L'autoscaling imposta gli intervalli degli assi x e y in modo che corrispondano alle coordinate dei dati che vengono plottati. Gli offset forniscono un meccanismo per espandere questi intervalli per lasciare spazio vuoto tra i dati e i bordi del grafico. L'autoscaling in seguito estende ulteriormente ogni intervallo per raggiungere il successivo tic dell'asse, a meno che questo non sia stato soppresso da **set autoscale noextend** o **set xrange noextend**. Vedere **noextend (p. 139)**. Gli offset influenzano solo il ridimensionamento per gli assi x1 e y1.

Sintassi:

```
set offsets <left>, <right>, <top>, <bottom>
unset offsets
show offsets
```

Ogni offset può essere una costante o un'espressione. Ognuno di essi ha come valore predefinito 0. Per default, gli offset sinistro e destro sono dati in unità del primo asse x, gli offset nella parte superiore e nella parte inferiore in unità del primo asse y. In alternativa, è possibile specificare gli offset come frazione della dimensione totale del grafico (graph) usando la keyword "graph". Solo gli offset "graph" sono possibili per gli assi non lineari.

Un offset positivo espande l'intervallo degli assi nella direzione specificata, ad esempio un offset inferiore positivo rende ymin più negativo. Gli offset negativi interagiscono male con l'autoscaling e il ritaglio (clipping).

Esempio:

```
set autoscale noextend
set offsets graph 0.05, 0, 2, 2
plot sin(x)
```

Questo grafico (graph) di sin(x) avrà un intervallo y [-3:3] perché la funzione sarà autoscalata a [-1:1] e gli offset verticali aggiungeranno 2 ad ogni estremità dell'intervallo. L'intervallo x sarà [-11:10] perché il default è [-10:10] ed è stato espanso a sinistra di 0.05 dell'intervallo totale.

Origin

Il comando **set origin** è usato per specificare l'origine di una superficie di plotting (cioè il grafico (graph) e i suoi margini) sullo schermo. Le coordinate sono date nel sistema di coordinate dello schermo (**screen**) (vedere **coordinate (p. 33)** per informazioni su questo sistema).

Sintassi:

```
set origin <x-origin>,<y-origin>
```

Output

Per default, le schermate vengono visualizzate sull'output standard. Il comando **set output** reindirizza la visualizzazione al file o al dispositivo specificato.

Sintassi:

```
set output {"<filename>"}
show output
```

Il filename deve essere racchiuso tra virgolette. Se il filename viene omesso, qualsiasi file di output aperto da una precedente invocazione di **set output** verrà chiuso e il nuovo output verrà inviato a STDOUT. (Se si dà il comando **set output "STDOUT"**, il proprio output potrebbe essere inviato a un file chiamato "STDOUT"! ["Potrebbe", non "sarà", perchè alcuni terminali, come **x11** o **wxt**, ignorano **set output**.])

Quando si usano sia **set terminal** che **set output**, è più sicuro dare prima **set terminal**, perché alcuni terminali impostano un flag che è necessario in alcuni sistemi operativi. Questo sarebbe il caso, per esempio, se il sistema operativo avesse bisogno di un comando di apertura separato per i file binari.

Sulle piattaforme che supportano le pipe, può essere utile convogliare l'output del terminale. Per esempio,

```
set output "|lpr -Plaser filename"
set term png; set output "|display png:-"
```

Sulle macchine MSDOS, **set output "PRN"** dirigerà l'output alla stampante predefinita. Su VMS, l'output può essere inviato direttamente a qualsiasi dispositivo di spooling.

Overflow

Sintassi:

```
set overflow {float | NaN | undefined}
unset overflow
```

Questa versione di gnuplot supporta l'aritmetica intera a 64 bit. Questo significa che per i valori da 2^53 a 2^63 (all'incirca 10^16 a 10^19) la valutazione dei numeri interi conserva una maggiore precisione rispetto alla valutazione con l'aritmetica in virgola mobile IEEE 754. Tuttavia, a differenza della rappresentazione in virgola mobile IEEE che sacrifica la precisione per coprire un intervallo totale di approssimativamente [-10^307 : 10^307], le operazioni intere che risultano in valori fuori dell'intervallo [-2^63 : 2^63] eccedono (overflow). Il comando **set overflow** permette di controllare cosa succede in caso di overflow. Vedere le opzioni qui sotto.

set overflow è uguale a **set overflow float**. Fa sì che il risultato venga restituito come un numero reale piuttosto che come un intero. Questo è il default.

Il comando **unset overflow** fa sì che l'overflow aritmetico intero venga ignorato. Non viene mostrato alcun errore. Questo può essere desiderabile se la propria piattaforma permette solo l'aritmetica dei numeri interi a 32 bit e si vuole approssimare il comportamento delle versioni di gnuplot precedenti alla 5.4.

Il comando **reset** non influisce sullo stato della gestione dell'overflow.

Le versioni precedenti di gnuplot erano limitate all'aritmetica a 32 bit e ignoravano l'overflow dei numeri interi. Si noti, tuttavia, che alcuni operatori incorporati non utilizzavano del tutto l'aritmetica dei numeri interi, anche quando venivano dati argomenti interi. Questo includeva l'operatore di potenza N**M e l'operatore di sommatoria (vedere **sommatoria (p. 44)**). Queste operazioni ora restituiscono un valore intero quando vengono dati argomenti interi, rendendole potenzialmente suscettibili a overflow e quindi influenzate dallo stato di **set overflow**.

Float

Se un'espressione aritmetica di numeri interi supera (overflows) l'intervallo limite, [-2^63 : 2^63] per gli interi a 64 bit, il risultato viene restituito come valore in virgola mobile. Ciò non viene trattato come un errore. Esempio:

```
gnuplot> set overflow float
gnuplot> A = 2**62 - 1;  print A, A+A, A+A+A
4611686018427387903 9223372036854775806 1.38350580552822e+19
```

NaN

Se un'espressione aritmetica di numeri interi supera (overflows) l'intervallo limite, [-2^63 : 2^63] per gli interi a 64 bit, il risultato viene restituito come NaN (non è un numero). Ciò non viene trattato come un errore. Esempio:

```
gnuplot> set overflow NaN
gnuplot> print 10**18, 10**19
1000000000000000000 NaN
```

Undefined (Indefinito)

Se un'espressione aritmetica di numeri interi supera l'intervallo limite, [-2^63 : 2^63] per gli interi a 64 bit, il risultato è indefinito. Ciò viene trattato come un errore. Esempio:

```
gnuplot> set overflow undefined
gnuplot> A = 10**19
             ^
         undefined value
```

Affected operations

Lo stato di **set overflow** interessa gli operatori aritmetici

```
+ - * / **
```

e l'operazione di sommatoria incorporata **sum**.

Tutte queste operazioni restituiranno un risultato intero se tutti gli argomenti sono interi, a condizione che non si verifichi un overflow durante la valutazione.

Lo stato **set overflow** non influisce sulle operazioni logiche o di bit

```
<< >>  | ^ &
```

Se si verifica un overflow in qualsiasi momento durante la valutazione di una sommatoria **set overflow float** farà sì che il risultato venga restituito come un numero reale anche se la somma finale è all'interno dell'intervallo di rappresentazione dei numeri interi.

Palette

La palette è un insieme di colori, di solito ordinati in forma di uno o più gradienti a gradini, usati per le superfici **pm3d** e altri elementi del grafico (graph) colorati in base al valore z. I colori nella palette corrente sono automaticamente mappati dai valori z delle coordinate del grafico o da una colonna di dati extra di valori di grigio. Si può accedere esplicitamente ai colori della palette in una specificazione di colore (vedere **colorspec (p. 50)**)

- come un **gray value** (valore grigio) conosciuto anche come **palette fraction** (frazione di palette) nell'intervallo [0:1]
- come un **z value** corrispondente alla coordinata z di un elemento del grafico
- come un **cb value** nell'intervallo [cbmin:cbmax] (vedere **set cbrange (p. 236)**)

La palette corrente è mostrata di default in una **colorbox** separata disegnata accanto ai grafici che usano lo stile di grafico **pm3d**. La colorbox può essere selezionata o disabilitata manualmente. Vedere **set colorbox (p. 146)**.

Sintassi:

```
set palette
set palette {
            { gray | color }
            { gamma <gamma> }
            {   rgbformulae <r>,<g>,<b>
              | defined { ( <gray1> <color1> {, <grayN> <colorN>}... ) }
              | file '<filename>' {datafile-modifiers}
              | functions <R>,<G>,<B>
            }
            { cubehelix {start <val>} {cycles <val>} {saturation <val>} }
            { model { RGB | HSV | CMY } }
            { positive | negative }
            { nops_allcF | ps_allcF }
            { maxcolors <maxcolors> }
          }
show palette
show palette palette <n> {{float | int}}
show palette gradient
show palette fit2rgbformulae
show palette rgbformulae
show colornames
```

set palette (cioè senza opzioni) imposta i valori predefiniti. Altrimenti, le opzioni possono essere date in qualsiasi ordine. **show palette** mostra le proprietà correnti della palette.

show palette gradient mostra il gradiente che definisce la palette (all'occorrenza). **show palette rgbformulae** stampa le formule di trasformazione grigio fisso –> colore disponibili. **show colornames** stampa i nomi dei colori conosciuti.

show palette palette <n> stampa sullo schermo o sul file dato da **set print** una tabella di terzine RGB calcolate per le impostazioni correnti della palette e una palette con <n> colori discreti. L'ampia tabella predefinita può essere limitata a 3 colonne di valori float r,g,b [0..1] o valori interi [0..255] con le opzioni float o int, rispettivamente. In questo modo, l'attuale palette di colori di gnuplot può essere caricata in altre applicazioni di rappresentazione (imaging), per esempio Octave. In alternativa, il comando **test palette** plotterà i profili R,G,B per la palette corrente e lascerà i valori dei profili in un datablock $PALETTE.

Le seguenti opzioni determinano le proprietà di colorazione.

La figura che usa questa palette può essere **gray** o **color**. Ad esempio, nelle superfici di colore **pm3d** il grigio di ogni piccolo punto è ottenuto mappando la coordinata z media dei 4 angoli dei quadrangoli di

superficie nell'intervallo [min_z,max_z] fornendo una gamma di grigi [0:1]. Questo valore può essere usato direttamente come grigio per le mappe dei grigi. La mappa dei colori richiede una trasformazione grigio (gray) –> (R,G,B), cioè una mappatura [0:1] –> ([0:1],[0:1],[0:1]).

Fondamentalmente si possono usare due diversi tipi di mappature: Formule analitiche per convertire il grigio in colore, o tabelle di mappatura discrete che vengono interpolate. **palette rgbformulae** e **palette functions** utilizzano formule analitiche mentre **palette defined** e **palette file** utilizzano tabelle interpolate. **palette rgbformulae** riduce la dimensione dell'output di postscript al minimo.

Il comando **show palette fit2rgbformulae** trova la migliore corrispondenza di **set palette rgbformulae** per l'attuale **set palette**. Naturalmente, ha senso usarlo per le palette non-rgbformulae. Questo comando può risultare utile principalmente per programmi esterni che usano la stessa definizione rgbformulae di palette di gnuplot, come zimg (http://zimg.sourceforge.net

).

set palette gray passa a una palette solo di grigi. **set palette rgbformulae**, **set palette defined**, **set palette file** e **set palette functions** passano a una mappatura del colore. **set palette color** è un modo semplice per tornare dalla palette dei grigi all'ultima mappatura dei colori.

La correzione automatica della gamma tramite **set palette gamma <gamma>** può essere fatta per mappe dei grigi (**set palette gray**) e per gli schemi della palette di colori **cubehelix**. Gamma = 1 produce una rampa lineare di intensità. Vedere **test palette (p. 245)**.

Molti terminali supportano solo un discreto numero di colori (ad esempio 256 colori in gif). Dopo che i colori predefiniti del tipo di linea di gnuplot sono assegnati, il resto dei colori disponibili sono riservati di default per pm3d. Così un multiplot che usa molteplici palette potrebbe non riuscire perché la prima palette ha utilizzato tutte le posizioni di colore disponibili. È possibile mitigare questa limitazione utilizzando **set palette maxcolors <N>** con un valore ragionevolmente piccolo di N. Questa opzione fa sì che N colori distinti siano selezionati da una palette continua campionata a intervalli equidistanti. Se si desidera una spaziatura ineguale di N colori distinti, usare **set palette defined** invece di una singola palette continua.

Lo spazio dei colori RGB potrebbe non essere lo spazio dei colori più utile in cui lavorare. Per questa ragione si può cambiare il **model** (modello) dello spazio dei colori in uno di **RGB**, **HSV**, **CMY**. L'uso dei nomi dei colori per le tabelle **set palette defined** e uno spazio dei colori diverso da RGB risulterà in colori strani. Tutte le spiegazioni sono state scritte per lo spazio dei colori RGB, quindi si prega di notare che **R** può essere **H**, o **C**, a seconda dello spazio dei colori effettivo (**G** e **B** di conseguenza).

Tutti i valori per tutti gli spazi dei colori sono limitati a [0,1].

RGB sta per Red, Green, Blue (rosso, verde, blu); CMY sta per Cyan, Magenta, Yellow (ciano, magenta, giallo); HSV sta per Hue, Saturation, Value (tinta, saturazione, valore). Per ulteriori informazioni sui modelli di colore vedere: http://en.wikipedia.org/wiki/Color_space

Nota: Le versioni precedenti di gnuplot accettavano anche i modelli di spazio dei colori YIQ e XYZ, ma l'implementazione non era mai completa o corretta.

Rgbformulae

Per le **rgbformulae** (formule rgb) devono essere scelte tre funzioni di mappatura adeguate. Questo viene fatto tramite **rgbformulae <r>,<g>,<b>**. Le funzioni di mappatura disponibili sono elencate da **show palette rgbformulae**. L'impostazione predefinita è **7,5,15**, altri esempi sono **3,11,6**, **21,23,3** o **3,23,21**. I numeri negativi, come **3,-11,-6**, significano colore invertito (cioè 1-gray passato nella formula, vedere anche le opzioni **positive (p. 192)** e **negative (p. 192)** qui sotto).

Alcuni schemi interessanti nello spazio dei colori RGB

```
7,5,15   ... pm3d tradizionale (nero-blu-rosso-giallo)
3,11,6   ... verde-rosso-viola
23,28,3  ... ocean (verde-blu-bianco); provare anche tutte le altre permutazioni
21,22,23 ... hot (nero-rosso-giallo-bianco)
30,31,32 ... colore stampabile su grigio (nero-blu-viola-giallo-bianco)
```

```
33,13,10 ... arcobaleno (blu-verde-giallo-rosso)
34,35,36 ... AFM hot (nero-rosso-giallo-bianco)
```

Una palette di colori completa nello spazio dei colori HSV

```
3,2,2    ... rosso-giallo-verde-ciano-blu-magenta-rosso
```

Si noti che anche se chiamate **rgbformulae** le formule potrebbero effettivamente determinare <H>,<S>,<V> o <X>,<Y>,<Z> o ... componenti di colore come al solito.

Usare **positive** e **negative** per invertire i colori della figura.

Si noti che è possibile trovare un insieme delle migliori formule rgbformulae corrispondenti per qualsiasi altro schema di colori con il comando

```
show palette fit2rgbformulae
```

Defined

La mappatura gray-to-rgb (da grigio a rgb) può essere impostata manualmente con l'uso di **palette defined**: Un gradiente di colore è definito e usato per dare i valori rgb. Tale gradiente è una mappatura lineare a tratti da valori di grigio in [0,1] allo spazio RGB [0,1]x[0,1]x[0,1]. È necessario specificare i valori di grigio e i corrispondenti valori RGB tra i quali verrà effettuata l'interpolazione lineare.

Sintassi:

```
set palette  defined { ( <gray1> <color1> {, <grayN> <colorN>}... ) }
```

<grayX> sono valori di grigio che sono mappati a [0,1] e <colorX> sono i colori rgb corrispondenti. Il colore può essere specificato in tre modi diversi:

```
<color> :=  { <r> <g> <b> | '<color-name>' | '#rrggbb' }
```

O da tre numeri (ciascuno in [0,1]) per rosso, verde e blu, separati da spazi bianchi, o dal nome del colore tra virgolette o dagli specificatori di colore in stile X anch'essi tra virgolette. È possibile mischiare liberamente i tre tipi in una definizione di gradiente, ma il colore chiamato "red" sarà qualcosa di strano se RGB non è selezionato come spazio dei colori. Usare **show colornames** per una lista di nomi di colori conosciuti.

Si prega di notare che anche se scritto come <r>, questo potrebbe essere in realtà il componente <H> nello spazio dei colori HSV a seconda del modello di colore selezionato.

I valori <gray> devono formare una sequenza ascendente di numeri reali; la sequenza sarà automaticamente riscalata a [0,1].

set palette defined (senza una definizione di gradiente tra parentesi) passa allo spazio dei colori RGB e utilizza un gradiente di colore a spettro completo preimpostato. Usare **show palette gradient** per visualizzare il gradiente.

Esempi:

Per produrre una palette di grigi (inutile ma istruttivo) usare:

```
set palette model RGB
set palette defined ( 0 "black", 1 "white" )
```

Per produrre una palette blu giallo rosso usare (tutti equivalenti):

```
set palette defined ( 0 "blue", 1 "yellow", 2 "red" )
set palette defined ( 0 0 0 1, 1 1 1 0, 2 1 0 0 )
set palette defined ( 0 "#0000ff", 1 "#ffff00", 2 "#ff0000" )
```

Per produrre una palette simile all'arcobaleno usare:

```
set palette defined ( 0 "blue", 3 "green", 6 "yellow", 10 "red" )
```

Spettro di colori completo nello spazio dei colori HSV:

```
set palette model HSV
set palette defined ( 0 0 1 1, 1 1 1 1 )
set palette defined ( 0 0 1 0, 1 0 1 1, 6 0.8333 1 1, 7 0.8333 0 1)
```

Approssimare la palette di default usata da MATLAB:

```
set pal defined (1 '#00008f', 8 '#0000ff', 24 '#00ffff', \
                 40 '#ffff00', 56 '#ff0000', 64 '#800000')
```

Per produrre una palette con pochi colori equidistanti:

```
set palette model RGB maxcolors 4
set palette defined ( 0 "yellow", 1 "red" )
```

Palette 'semaforo' (salti di colore non graduali a grigio = 1/3 e 2/3).

```
set palette model RGB
set palette defined (0 "dark-green", 1 "green", \
                     1 "yellow",     2 "dark-yellow", \
                     2 "red",        3 "dark-red" )
```

Functions

Usare **set palette functions** <**Rexpr**>, <**Gexpr**>, <**Bexpr**> per definire tre formule per la mappatura R(gray), G(gray) e B(gray). Le tre formule possono dipendere dalla variabile **gray** che prenderà valori in [0,1] e dovrebbe anche produrre valori in [0,1]. Si noti che <Rexpr> potrebbe essere una formula per il valore H se è stato scelto lo spazio dei colori HSV (lo stesso per tutte le altre formule e spazi dei colori).

Esempi:

Per produrre una palette di colori completa usare:

```
set palette model HSV functions gray, 1, 1
```

Una bella palette dal nero all'oro:

```
set palette model RGB functions 1.1*gray**0.25, gray**0.75, 0
```

Una palette in bianco e nero con correzione di gamma

```
gamma = 2.2
color(gray) = gray**(1./gamma)
set palette model RGB functions color(gray), color(gray), color(gray)
```

Gray

set palette gray passa a un'ombreggiatura della palette in scala di grigi da 0.0 = nero a 1.0 = bianco. **set palette color** è un modo semplice per passare dalla palette dei grigi all'ultima mappatura dei colori.

Cubehelix

L'opzione "cubehelix" definisce una famiglia di palette in cui il colore (tonalità) varia lungo la ruota dei colori standard mentre allo stesso tempo l'intensità netta aumenta monotonicamente mentre il valore di grigio va da 0 a 1.

```
D A Green (2011) http://arxiv.org/abs/1108.5083
```

start definisce il punto di partenza lungo la ruota dei colori in radianti. **cycles** definisce quanti cicli della ruota dei colori coprono la gamma della palette. Valori più grandi di **saturation** producono un colore più saturo; saturazione > 1 può portare al clipping delle singole componenti RGB e potrebbe fare sì che l'intensità diventi non monotonica. La palette è anche influenzata da **set palette gamma**. I valori predefiniti sono

```
set palette cubehelix start 0.5 cycles -1.5 saturation 1
set palette gamma 1.5
```

File

set palette file è fondamentalmente un **set palette defined (<gradient>)** dove <gradient> viene letto da un file di dati. Devono essere selezionate o 4 colonne (gray, R, G, B) o solo tre colonne (R,G,B) tramite il modificatore **using** del file di dati. Nel caso di tre colonne, il numero della linea sarà usato come grigio. L'intervallo dei grigi viene automaticamente riscalato a [0,1]. Il file viene letto come un normale file di dati, quindi tutti i modificatori di file di dati possono essere usati. Si prega di notare che **R** potrebbe essere ad esempio **H** se è selezionato lo spazio dei colori HSV.

Come al solito <filename> può essere **'-'**, il che significa che i dati seguono il comando in linea e sono terminati da una singola **e** su una linea a sé stante.

Usare **show palette gradient** per visualizzare il gradiente.

Esempi:

Leggere in una palette di terzine RGB, ciascuna nell'intervallo [0,255]:

```
set palette file 'some-palette' using ($1/255):($2/255):($3/255)
```

Palette arcobaleno (blu-verde-giallo-rosso) equidistante:

```
set palette model RGB file "-"
0 0 1
0 1 0
1 1 0
1 0 0
e
```

Sono supportati anche i file binari di palette, vedere **binary general (p. 109)**. Esempio: mettere 64 terzine di doppi R,G,B nel file palette.bin e caricarlo con

```
set palette file "palette.bin" binary record=64 using 1:2:3
```

Correzione di gamma

Per le mappature di grigio la correzione di gamma può essere attivata da **set palette gamma <gamma>**. <gamma> è predefinito a 1.5 che è sufficientemente adatto per la maggior parte dei terminali.

La correzione di gamma viene applicata alla famiglia della palette di colori cubehelix, ma non ad altri schemi di colorazione di palette. Tuttavia, si può facilmente implementare la correzione di gamma per funzioni di colore esplicite.

Esempio:

```
set palette model RGB
set palette functions gray**0.64, gray**0.67, gray**0.70
```

Per usare la correzione di gamma con gradienti interpolati, specificare i valori di grigio intermedi con colori appropriati. Invece di

```
set palette defined ( 0 0 0 0, 1 1 1 1 )
```

usare, ad es.,

```
set palette defined ( 0 0 0 0, 0.5 .73 .73 .73, 1 1 1 1 )
```

o anche molti più punti intermedi fino a quando l'interpolazione lineare si adatta all'interpolazione "corretta dalla gamma" abbastanza bene.

Postscript

Per ridurre le dimensioni dei file postscript, nel file vengono scritti il valore di grigio e non tutti e tre i valori r,g,b calcolati. Perciò le formule analitiche sono codificate direttamente nel linguaggio postscript come intestazione appena prima del disegno pm3d, vedere le definizioni /g e /cF. Di solito, ha senso scrivere qui le definizioni delle sole 3 formule utilizzate. Ma per multiplot o per qualsiasi altra ragione si potrebbe voler modificare manualmente le trasformazioni direttamente nel file postscript. Questa è l'opzione predefinita **nops_allcF**. L'utilizzo dell'opzione **ps_allcF** scrive le definizioni postscript di tutte le formule. Questo può essere interessante se si desidera modificare il file postscript per avere diverse palette per diverse superfici in un grafico (graph). Si può ottenere questa funzionalità tramite **multiplot** con **origin** e **size** fissi.

Se si sta scrivendo una superficie pm3d in un file postscript, può essere possibile ridurre la dimensione del file fino al 50% con lo script awk **pm3dCompress.awk** allegato. Se i dati si trovano su una griglia rettangolare, una compressione ancora maggiore può essere possibile utilizzando lo script **pm3dConvertToImage.awk**. Utilizzo:

```
awk -f pm3dCompress.awk thefile.ps >smallerfile.ps
awk -f pm3dConvertToImage.awk thefile.ps >smallerfile.ps
```

Parametric

Il comando **set parametric** cambia il significato di **plot** (**splot**) da funzioni normali a funzioni parametriche. Il comando **unset parametric** ripristina lo stile di plotting al normale plotting di espressioni a valore singolo.

Sintassi:

```
set parametric
unset parametric
show parametric
```

Per il plotting 2D, una funzione parametrica è determinata da una coppia di funzioni parametriche che operano su un parametro. Un esempio di funzione parametrica 2D sarebbe **plot sin(t),cos(t)**, che disegna un cerchio (se il rapporto d'aspetto è impostato correttamente–vedere **set size** **(p. 205)**). **gnuplot** mostrerà un messaggio di errore se entrambe le funzioni non sono fornite per un **plot** parametrico.

Per il plotting 3D, la superficie è descritta come x=f(u,v), y=g(u,v), z=h(u,v). Quindi è necessaria una terna di funzioni. Un esempio di funzione parametrica 3D sarebbe **cos(u)*cos(v),cos(u)*sin(v),sin(u)**, che disegna una sfera. **gnuplot** mostrerà un messaggio di errore se tutte e tre le funzioni non sono fornite per un **splot** parametrico.

L'insieme totale dei grafici possibili è un superset dei semplici grafici in stile f(x), poiché le due funzioni possono descrivere i valori x e y da computare separatamente. Infatti, i grafici del tipo t,f(t) sono equivalenti a quelli prodotti con f(x) perché i valori di x sono computati usando la funzione identità. Allo stesso modo, i grafici 3D del tipo u,v,f(u,v) sono equivalenti a f(x,y).

Si noti che l'ordine in cui le funzioni parametriche sono specificate è xfunction, yfunction (e zfunction) e che ognuna opera sul dominio parametrico comune.

Inoltre, la funzione **set parametric** implica una nuova gamma di valori. Mentre il normale stile di plotting f(x) e f(x,y) presuppongono un xrange e un yrange (e zrange), la modalità parametrica specifica inoltre un trange, un urange e un vrange. Questi intervalli possono essere impostati direttamente con **set trange**, **set urange**, e **set vrange**, o specificando l'intervallo nei comandi **plot** o **splot**. Attualmente l'intervallo predefinito per queste variabili parametriche è [-5:5]. Ci si aspetta che gli intervalli siano impostati su qualcosa di più significativo.

Paxis

Sintassi:

```
set paxis <axisno> {range <range-options> | tics <tic-options>}
set paxis <axisno> label <label-options> { offset <radial-offset> }
show paxis <axisno> {range | tics}
```

Il comando **set paxis** è equivalente ai comandi **set xrange** e **set xtics**, salvo che agisce su uno degli assi p1, p2, ... usati nei grafici ad assi paralleli e negli spiderplot. Vedere **parallelaxes (p. 81)**, **set xrange (p. 226)**, e **set xtics (p. 228)**. Le normali opzioni per i comandi range e tic sono accettate sebbene non tutte le opzioni abbiano senso per i grafici ad assi paralleli.

set paxis <**axisno**> **label** <**label-options**> è rilevante per gli spiderplot ma altrimenti ignorato. Gli assi di un grafico ad assi paralleli possono essere etichettati usando l'opzione **title** del comando plot, che genera un'etichetta xtic. Si noti che questo potrebbe richiedere anche **set xtics**.

Le proprietà del tipo di linea dell'asse sono controllate usando **set style parallelaxis**.

Pixmap

Sintassi:

```
set pixmap <index> "filename" at <position>
           {width <w> | height <h> | size <w>,<h>}
           {front|back|behind} {center}
show pixmaps
unset pixmaps
unset pixmap <index>
```

Il comando **set pixmap** è simile a **set object** nel senso che definisce un oggetto che apparirà nei successivi grafici. L'array rettangolare di valori rosso/verde/blu/alpha che compongono la pixmap sono letti da un file png, jpeg, o gif. La posizione e l'estensione occupata dalla pixmap nell'output di gnuplot possono essere specificate in qualsiasi sistema di coordinate (vedere **coordinate (p. 33)**). Le coordinate date da **at** <**position**> si riferiscono all'angolo inferiore sinistro della pixmap, a meno che non sia presente la keyword **center**.

Se x-extent (estensione) della pixmap renderizzata è impostata usando **width** <**x-extent**> il rapporto d'aspetto dell'immagine originale viene mantenuto e né il rapporto d'aspetto né l'orientamento della pixmap cambia con il ridimensionamento o la rotazione degli assi. Allo stesso modo se y-extent è impostato usando **height** <**y-extent**>. Se sia x-extent che y-extent sono dati usando **size** <**x-extent**> <**y-extent**>, questo sovrascrive il formato originale. Se non viene impostata alcuna dimensione, viene utilizzata la dimensione originale in pixel (la dimensione effettiva dipende quindi dal terminale).

Le pixmap non sono ritagliate sul bordo del grafico. Come eccezione al comportamento generale di oggetti e layer, una pixmap assegnata al layer **behind** viene renderizzata solo per il primo grafico in un multiplot. Questo permette a tutti i pannelli in un multiplot di condividere una singola pixmap di sfondo.

Esempi:

```
# Usa un gradiente come sfondo per tutti i plotting
# Sia x che y saranno ridimensionati per riempire l'intero canvas
set pixmap 1 "gradient.png"
set pixmap 1 at screen 0, 0 size screen 1, 1 behind

# Mette un logo in basso a destra di ogni pagina plottata
set pixmap 2 "logo.jpg"
set pixmap 2 at screen 0.95, 0 width screen 0.05 behind

# Posiziona una piccola immagine ad una certa coordinata 3D
# Si muoverà come se fosse attaccata alla superficie da plottare,
# ma sarà sempre rivolta in avanti e rimarrà in posizione verticale
set pixmap 3 "image.png" at my_x, my_y, f(my_x,my_y) width screen .05
splot f(x,y)
```

Plot

Il comando **show plot** mostra il comando plotting corrente come risulta dall'ultimo **plot** e/o **splot** ed eventuali comandi **replot** successivi.

Inoltre, il comando **show plot add2history** aggiunge questo comando plot corrente nella **history** (cronologia). È utile se è stato usato **replot** per aggiungere più curve al grafico corrente e adesso si vuole modificare l'intero comando.

Pm3d

pm3d è uno stile **splot** per disegnare dati 3d e 4d mappati con palette come mappe a colori/grigio e superfici. Permette di plottare dati a griglia o non a griglia, senza pre-elaborazione. Le opzioni di stile pm3d influenzano anche i poligoni a riempimento solido usati per costruire altri elementi del grafico 3D.

Sintassi (le opzioni possono essere date in qualsiasi ordine):

```
set pm3d {
           { at <position> }
           { interpolate <steps/points in scan, between scans> }
           { scansautomatic | scansforward | scansbackward
                            | depthorder {base} }
           { flush { begin | center | end } }
           { ftriangles | noftriangles }
           { clip {z} | clip1in | clip4in }
           { {no}clipcb }
           { corners2color
             { mean|geomean|harmean|rms|median|min|max|c1|c2|c3|c4 }
           }
           { {no}lighting
             {primary <fraction>} {specular <fraction>} {spec2 <fraction>}
           }
           { border {<linestyle-options>}}
           { implicit | explicit }
           { map }
         }
show pm3d
unset pm3d
```

Si noti che i grafici pm3d sono plottati in modo sequenziale nell'ordine dato nel comando splot. Quindi i primi grafici possono essere oscurati da quelli successivi. Per evitare ciò è possibile utilizzare l'opzione di scansione **depthorder**.

Le superfici pm3d possono essere proiettate sul **top** (parte superiore) o sul **bottom** (parte inferiore) della finestra di visualizzazione. Vedere **pm3d position (p. 199)**. Il comando seguente disegna tre superfici a colori a diverse altitudini:

```
set border 4095
set pm3d at s
splot 10*x with pm3d at b, x*x-y*y, x*x+y*y with pm3d at t
```

Vedere anche aiuto per **set palette (p. 190)**, **set cbrange (p. 236)**, **set colorbox (p. 146)**, e il file demo **demo/pm3d.dem**.

Implicit

Una superficie a colori pm3d viene disegnata se il comando splot specifica **with pm3d**, se lo **style** (stile) dei dati o della funzione sono impostati globalmente su pm3d, o se la modalità pm3d è **set pm3d implicit**.

Negli ultimi due casi, la superficie pm3d è disegnata in aggiunta alla mesh prodotta dallo stile specificato nel comando plot. Ad es.

```
splot 'fred.dat' with lines, 'lola.dat' with lines
```

disegnerebbe sia una mesh di linee che una superficie pm3d per ogni set di dati. Se l'opzione **explicit** è attiva (o **implicit** è disattiva) solo i grafici specificati dall'attributo **with pm3d** sono plottati con una superficie pm3d, ad esempio:

```
splot 'fred.dat' with lines, 'lola.dat' with pm3d
```

plotterebbe 'fred.dat' (solo) con linee e 'lola.dat' con una superficie pm3d.

All'avvio di gnuplot, la modalità è **explicit**. Per ragioni di cronologia e di compatibilità, i comandi **set pm3d;** (cioè nessuna opzione) e **set pm3d at X ...** (cioè **at** è la prima opzione) cambiano la modalità in **implicit**. Il comando **set pm3d;** imposta le altre opzioni al loro stato predefinito.

Se si imposta lo stile predefinito dei dati o delle funzioni su **pm3d**, ad es:

```
set style data pm3d
```

allora le opzioni **implicit** e **explicit** non hanno alcun effetto.

Algorithm

Descriviamo prima come viene disegnata una mappa/superficie. I dati di input provengono da una funzione valutata o da un **splot data file**. Ogni superficie consiste in una sequenza di scansioni separate (isolinee). L'algoritmo pm3d riempie la regione tra due punti vicini in una scansione con altri due punti nella scansione successiva con un grigio (o colore) secondo i valori z (o secondo una colonna aggiuntiva 'color', vedere aiuto per **using (p. 121)**) di questi 4 angoli; per default i valori dei 4 angoli sono mediati, ma questo può essere cambiato dall'opzione **corners2color**. Per ottenere una superficie ragionevole, le scansioni vicine non dovrebbero incrociarsi e il numero di punti nelle scansioni vicine non dovrebbe variare troppo; naturalmente, il grafico migliore è fatto con scansioni che hanno lo stesso numero di punti. Non ci sono altri requisiti (ad esempio, i dati non devono essere messi in griglia). Un altro vantaggio è il fatto che l'algoritmo pm3d non disegna nulla al di fuori della regione di input (misurata o calcolata).

La colorazione della superficie funziona con i seguenti dati di input:

1. splot di una funzione o di un file di dati con una o tre colonne di dati: La scala di grigi/colore si ottiene mappando la coordinata z media (o **corners2color**) dei quattro angoli del quadrangolo sopra specificato nell'intervallo [min_color_z,max_color_z] di **zrange** o **cbrange** fornendo un valore di grigio nell'intervallo [0:1]. Questo valore può essere usato direttamente come grigio per le mappe di grigi. Il valore di grigio normalizzato può essere ulteriormente mappato in un colore — vedere **set palette (p. 190)** per la descrizione completa.

2. splot di file di dati con due o quattro colonne di dati: Il valore di grigi/colore si ottiene usando la coordinata dell'ultima colonna invece del valore z, permettendo così al colore e alla coordinata z di essere reciprocamente indipendenti. Questo può essere utilizzato per il disegno di dati 4d.

Ulteriori note:

1. Il termine 'scan' (scansione) a cui si fa riferimento sopra è più usato tra i fisici rispetto al termine 'iso_curve' a cui si fa riferimento nella documentazione e nelle sorgenti di gnuplot. Si misurano le mappe registrate una scansione dopo l'altra, proprio per questo.

2. La scala 'gray' (di grigi) o 'color' è una mappatura lineare di una variabile continua su una palette di colori che varia gradualmente. La mappatura è mostrata in un rettangolo accanto al grafico principale. Questa documentazione si riferisce a questo come un "colorbox", e si riferisce alla variabile di indicizzazione come situata sull'asse del colorbox. Vedere **set colorbox (p. 146)**, **set cbrange (p. 236)**.

Lighting

Per default i colori assegnati agli oggetti pm3d non dipendono dall'orientamento o dall'angolo di visualizzazione. Questo stato corrisponde a **set pm3d nolighting**. Il comando **set pm3d lighting** seleziona un

modello di illuminazione semplice che consiste in una singola fonte di illuminazione fissa che contribuisce al 50% dell'illuminazione complessiva. L'intensità di questa luce rispetto all'illuminazione ambientale può essere regolata da **set pm3d lighting primary <fraction>**. L'inclusione dell'evidenziazione speculare può essere regolata impostando un contributo frazionario:

```
set pm3d lighting primary 0.50 specular 0.0   # nessun highlight
set pm3d lighting primary 0.50 specular 0.6   # highlight forti
```

Le superfici pm3d a tinta unita tendono a sembrare molto piatte senza riflessi speculari. Dato che le luci da una singola fonte colpiscono solo un lato della superficie, una seconda fonte di luce potrebbe servire per aggiungere riflessi speculari dalla direzione opposta. Questo è controllato da "spec2<contribution>". SPERIMENTALE (i dettagli possono cambiare in una versione futura): Il secondo riflettore è una fonte di luce rossa pura che per default non contribuisce a nulla (spec2 0.0). Vedere anche hidden_compare.dem (confronto tra il trattamento hidden3d e pm3d di superfici a tinta unita)

Position

La superficie a colori può essere disegnata alla base o in cima (allora è una mappa planare grigio/colore) o alle coordinate z dei punti di superficie (superficie grigio/colore). Questo è definito dall'opzione **at** con una stringa di fino a 6 combinazioni di **b**, **t** e **s**. Per esempio, **at b** plotta solo in basso, **at st** plotta prima la superficie e poi la mappa superiore, mentre **at bstbst** non sarà mai usato seriamente.

I quadrangoli colorati sono plottati uno dopo l'altro. Quando si plottano le superfici (**at s**), i quadrangoli successivi si sovrappongono (overdraw) a quelli precedenti. (Gnuplot non è uno strumento di realtà virtuale per calcolare le intersezioni di mesh di poligoni riempiti.) Si può provare a passare tra **scansforward** e **scansbackward** per forzare la prima scansione dei dati ad essere plottata per prima o per ultima. L'impostazione predefinita è **scansautomatic**, con la quale gnuplot fa un'ipotesi sull'ordine delle scansioni. D'altra parte, l'opzione **depthorder** riordina completamente i quadrangoli. Il rendering viene eseguito dopo un ordinamento di profondità, che permette di visualizzare anche superfici complicate; vedere **pm3d depthorder (p. 199)** per maggiori dettagli.

Scanorder

```
set pm3d {scansautomatic | scansforward | scansbackward | depthorder}
```

Per default i quadrangoli che compongono una superficie solida pm3d sono renderizzati nell'ordine ordine in cui si incontrano lungo i punti della griglia della superficie. Questo ordine può essere controllato dalle opzioni **scansautomatic|scansforward|scansbackward**. Queste opzioni di scansione non sono generalmente compatibili con la rimozione delle superfici nascoste.

Se due scansioni successive non hanno lo stesso numero di punti, allora è necessario decidere se iniziare a prendere i punti per i quadrangoli dall'inizio di entrambe le scansioni (**flush begin**), dalle loro estremità (**flush end**) o centrarle (**flush center**). Si noti che **flush (center|end)** sono incompatibili con **scansautomatic**: se si specifica **flush center** o **flush end** e **scansautomatic** è impostato, si passa automaticamente a **scansforward**.

Se due scansioni successive non hanno lo stesso numero di punti, l'opzione **ftriangles** specifica se i triangoli di colore sono disegnati in corrispondenza delle code della scansione, dove non ci sono abbastanza punti in entrambe le scansioni. Ciò può essere usato per disegnare un confine di mappa uniforme.

Gnuplot non esegue una vera rimozione delle superfici nascoste per le superfici solide, ma spesso è sufficiente renderizzare i quadrangoli componenti secondo un ordine che va dal più lontano al più vicino. Questa modalità può essere selezionata utilizzando l'opzione

```
set pm3d depthorder
```

Si noti che l'opzione globale **set hidden3d** non ha effetto sulle superfici pm3d.

L'opzione **depthorder**, da sola, tende a produrre cattivi risultati quando viene applicata ai rettangoli lunghi e sottili generati da **splot with boxes**. È preferibile aggiungere la keyword **base**, che esegue l'ordinamento di

profondità usando l'intersezione del box con il piano a z=0. Questo tipo di grafico è ulteriormente migliorato aggiungendo un modello di illuminazione. Esempio:

```
set pm3d depthorder base
set pm3d lighting
set boxdepth 0.4
splot $DATA using 1:2:3 with boxes
```

Clipping

Sintassi:

```
set pm3d {clip {z} | clip1in | clip4in}
set pm3d {no}clipcb
```

I quadrangoli componenti di una superficie pm3d o di un altro oggetto 3D sono per default ritagliati in modo uniforme rispetto allo zrange corrente. Questo è un cambiamento rispetto alle versioni precedenti di gnuplot.

In alternativa, le superfici possono essere ritagliate renderizzando interi quadrangoli ma solo quelli con tutti e 4 gli angoli nel raggio d'azione angolo nel raggio d'azione di x, y e z (**set pm3d clip1in**). Le opzioni **clip**, **clip1in**, e **clip4in** si escludono a vicenda.

Separatamente dal ritaglio basato sulle coordinate spaziali x, y e z, i quadrangoli possono essere renderizzati o meno in base ai valori estremi di colori della palette. **clipcb**: (default) valori di colore della palette < cbmin sono ritagliati per essere uguali a cbmin; valori di colore della palette > cbmax sono ritagliati per essere uguali a cbmax. **noclipcb**: i quadrangoli con valore di colore al di fuori di cbrange non vengono disegnati affatto.

Color_assignment

La colorazione pm3d di default assegna un colore individuale ad ogni quadrangolo della griglia della superficie. Per schemi di colorazione alternativi che assegnano un colore uniforme a tutta la superficie, vedere **pm3d fillcolor (p. 201)**.

Un singolo valore grigio/colore (cioè non un gradiente) è assegnato ad ogni quadrangolo. Questo valore è calcolato dalle coordinate z dei quattro angoli del quadrangolo secondo **corners2color** <**option**>. Il valore viene poi usato per selezionare un colore dalla palette corrente. Vedere **set palette (p. 190)**. Non è possibile cambiare palette all'interno di un singolo comando **splot**.

Se viene fornita una quarta colonna di dati, la colorazione dei singoli quadrangoli funziona come sopra, eccetto che il valore di colore è distinto dal valore z. Come opzione di colorazione separata, la quarta colonna di dati può fornire invece un colore RGB. Vedere **rgbcolor variable (p. 52)**. In questo caso il comando di plotting deve essere

```
splot ... using 1:2:3:4 with pm3d lc rgb variable
```

Notare che gli intervalli dei valori z e dei valori di colore per le superfici sono regolabili indipendentemente da **set zrange**, **set cbrange**, **set log z**, **set log cb**, ecc.

Corners2color

Il colore di ogni quadrangolo in una superficie pm3d è assegnato in base ai valori di colore dei suoi quattro vertici confinanti. Le opzioni 'mean' (default), 'geomean', 'harmean, 'rms', e 'median' producono vari tipi di attenuazione del colore della superficie, mentre le opzioni 'min' e 'max' scelgono rispettivamente il valore minimo o massimo. Questo potrebbe non essere desiderabile per immagini pixel o per mappe con picchi nitidi e intensi, nel qual caso le opzioni 'c1', 'c2', 'c3' o 'c4' possono invece essere usate per assegnare il colore del quadrangolo in base alla coordinata z di un solo angolo. Alcuni tentativi potrebbero essere necessari per determinare quale angolo corrisponde a 'c1', poiché l'orientamento dipende dalla direzione del disegno.

Poiché l'algoritmo pm3d non estende la superficie colorata al di fuori dell'intervallo dei data point di input, le opzioni di colorazione 'c<j>' risulteranno in pixel lungo due bordi della griglia che non contribuiscono al colore di nessun quadrangolo. Per esempio, applicare l'algoritmo pm3d alla griglia 4x4 di data point nello script **demo/pm3d.dem** (si prega di controllare) produce solo (4-1)x(4-1)=9 rettangoli colorati.

Border

L'opzione **set pm3d border {line-properties}** disegna linee di delimitazione intorno ad ogni quadrangolo mentre viene renderizzato. Normalmente questo viene usato insieme all'opzione **depthorder** per approssimare la rimozione delle linee nascoste. Si noti che l'opzione globale **set hidden3d** non ha effetto sui grafici pm3d. Le proprietà predefinite della linea (colore, spessore) seguono eventualmente la keyword **border**. Questi valori predefiniti possono essere sovrascritti successivamente in un comando splot.

Esempio di utilizzo raccomandato:

```
set pm3d at s depthorder border lw 0.2 lt black
unset hidden3d
unset surf
splot x*x+y*y linecolor rgb "blue"   # altrimenti sarebbe nero
```

Fillcolor

```
splot FOO with pm3d fillcolor <colorspec>
```

Lo stile di grafico **with pm3d** accetta un colore di riempimento opzionale nel comando splot. Questa specifica è applicata all'intera superficie pm3d. Vedere **colorspec (p. 50)**. La maggior parte delle specifiche dei colori di riempimento risulteranno in una singola tinta unita, che è difficile da interpretare visivamente a meno che non sia presente anche un modello di illuminazione per distinguere i componenti della superficie in base all'orientamento. Vedere **pm3d lighting (p. 198)**.

Esistono alcuni casi speciali. **with pm3d fillcolor palette** produrrebbe lo stesso risultato della colorazione predefinita basata su palette di pm3d, e quindi non è un'opzione utile. **with pm3d fillcolor linestyle N** è più interessante. Questa variante assegna colori diversi alla parte superiore e inferiore della superficie pm3d, simile allo schema di colori usato dalla modalità **hidden3d** di gnuplot. Lo stile linea N è usato per la superficie superiore; stile linea N+1 per la superficie inferiore. Si noti che "top" (superiore) e "bottom" (inferiore) dipendono dall'ordine di scansione, in questo modo i colori sono invertiti per **pm3d scansbackward** rispetto a **pm3d scansforward**. Questa opzione di colorazione funziona meglio con **pm3d depthorder**, che sfortunatamente non permette di controllare l'ordine di scansione, quindi si potrebbe dover scambiare i colori definiti per gli stili linea N e N+1.

Interpolate

L'opzione **interpolate m,n** interpola tra i punti della griglia per generare una mesh più fine. Per i file di dati, questo uniforma la superficie del colore e migliora il contrasto dei picchi sulla superficie. Quando si lavora con le funzioni, l'interpolazione ha poco senso. Di solito avrebbe più senso aumentare **samples** e **isosamples**.

Per m e n positivi, ogni quadrangolo o triangolo viene interpolato m volte e n volte nella rispettiva direzione. Per m e n negativi, la frequenza di interpolazione viene scelta in modo che ci siano almeno |m| e |n| punti disegnati; si può considerare questa come una speciale funzione di gridding.

Nota: **interpolate 0,0**, sceglierà automaticamente un numero ottimale di punti di superficie interpolati.

Nota: Attualmente l'interpolazione del colore è sempre lineare, anche se corners2color è impostato su uno schema non lineare come la media geometrica.

Deprecated_options

L'opzione deprecata **set pm3d map** era equivalente a **set pm3d at b; set view map; set style data pm3d; set style func pm3d;**

L'opzione deprecata **set pm3d hidden3d N** era equivalente a **set pm3d border ls N**.

Pointintervalbox

Le proprietà **pointinterval** e **pointnumber** di un tipo di linea sono usati solo nello stile di grafico **linespoints**. Un valore negativo di pointinterval o pointnumber, ad esempio -N, significa che prima che l'insieme selezionato di simboli punto venga disegnato, viene cancellato un box (in realtà cerchio) dietro ogni simbolo di punto riempiendolo con il colore di sfondo. Il comando **set pointintervalbox** controlla il raggio di questa regione cancellata. È un moltiplicatore per il raggio predefinito, che è uguale alla dimensione del punto.

Pointsize (Dimensione del punto)

Il comando **set pointsize** ridimensiona la dimensione dei punti usati nei grafici.

Sintassi:

```
set pointsize <multiplier>
show pointsize
```

Il default è un moltiplicatore di 1.0. Pointsize più grandi possono essere utili per rendere i punti più visibili nelle grafiche bitmap.

La pointsize di un singolo grafico può essere cambiata con il comando **plot**. Vedere **plot with (p. 131)** per i dettagli.

Si prega di notare che l'impostazione pointsize non è supportata da tutti i tipi di terminale.

Polar

Il comando **set polar** cambia il significato del grafico da coordinate rettangolari a coordinate polari.

Sintassi:

```
set polar
unset polar
show polar
```

Nelle coordinate polari, la variabile dummy (t) rappresenta un angolo theta. L'intervallo predefinito di t è [0:2*pi], o [0:360] se sono state selezionate le unità di grado (vedere **set angles (p. 136)**).

Il comando **unset polar** cambia il significato del grafico riportandolo al sistema di coordinate rettangolari predefinito.

Il comando **set polar** non è supportato per **splot**s. Vedere il comando **set mapping (p. 175)** per una funzionalità simile per **splot (p. 237)**s.

Mentre in coordinate polari il significato di un'espressione in t è veramente r = f(t), dove t è un angolo di rotazione. Il trange controlla il dominio (l'angolo) della funzione. Gli intervalli r, x e y controllano l'estensione del grafico (graph) nelle direzioni x e y. Ciascuno di questi intervalli, così come rrange, può essere autoscalato o impostato esplicitamente. Per i dettagli, vedere **set rrange (p. 204)** e **set xrange (p. 226)**.

Esempio:

```
set polar
plot t*sin(t)
```

```
set trange [-2*pi:2*pi]
set rrange [0:3]
plot t*sin(t)
```

Il primo **plot** utilizza il dominio angolare polare predefinito da 0 a 2*pi. Il raggio e la dimensione del grafico (graph) sono scalati automaticamente. Il secondo **plot** espande il dominio e limita la dimensione del grafico (graph) all'area entro 3 unità dall'origine. Questo ha l'effetto di limitare x e y a [-3:3].

Per default i grafici polari sono orientati in modo che theta=0 sia all'estrema destra, con theta che aumenta in senso antiorario. È possibile cambiare sia l'origine che il senso esplicitamente. Vedere **set theta (p. 216)**.

Potrebbe essere utile impostare **set size square** per far sì che **gnuplot** provi a rendere il rapporto d'aspetto uguale all'unità, in modo che i cerchi sembrino circolari. I segni di tic intorno al perimetro possono essere specificati usando **set ttics**. Vedere anche `demo polar (polar.dem)`

e `polar data plot (poldat.dem)`.

Print

Il comando **set print** reindirizza l'output del comando **print** a un file.

Sintassi:

```
set print
set print "-"
set print "<filename>" [append]
set print "|<shell_command>"
set print $datablock [append]
```

set print senza parametri ripristina l'output a <STDERR>. Il <filename> "-" significa <STDOUT>. Il flag **append** fa sì che il file sia aperto in modalità append . Un <filename> che inizia con "|" è aperto come una pipe verso il comando <shell_command> su piattaforme che supportano il piping.

La destinazione dei comandi **print** può anche essere un data block con nome. I nomi dei data block iniziano con '$', vedere anche **inline data (p. 48)**. Quando si stampa una stringa in un data block, i caratteri newline incorporati vengono espansi per generare più entry di data block. Questo è un CAMBIAMENTO.

Psdir

Il comando **set psdir** <**directory**> controlla il percorso di ricerca usato dal terminale postscript per trovare i file prologue.ps e i file di codifica dei caratteri. Si può usare questo meccanismo per passare tra diversi insiemi di file prolog personalizzati a livello locale. L'ordine di ricerca è

```
1) La directory specificata da ‘set psdir‘, se presente
2) La directory specificata dalla variabile d’ambiente GNUPLOT_PS_DIR
3) Un header incorporato o uno dalla directory di sistema predefinita
4) Directory impostate da ‘set loadpath‘
```

Raxis

I comandi **set raxis** e **unset raxis** X e B permettono di scegliere se l'asse polare è disegnato separatamente o meno dalle linee della griglia e dall'asse x. Se il minimo del rrange corrente è diverso da zero (e non autoscalato), allora viene disegnato un cerchio bianco al centro del grafico polare per indicare che le linee e gli assi del grafico non raggiungono lo 0. La linea dell'asse è disegnata usando lo stesso tipo di linea del bordo del grafico. Vedere **polar (p. 202)**, **rrange (p. 204)**, **rtics (p. 204)**, **rlabel (p. 204)**, **set grid (p. 161)**.

Rgbmax

Sintassi:

```
set rgbmax {1.0 | 255}
unset rgbmax
```

Le componenti di colore rosso/verde/blu di un grafico rgbimage sono per default interpretate come interi nell'intervallo [0:255]. **set rgbmax 1.0** dice al programma che i valori dei dati usati per generare le componenti di colore di un grafico con **rgbimage** o **rgbalpha** sono valori in virgola mobile nell'intervallo [0:1]. **unset rgbmax** ritorna all'intervallo intero predefinito [0:255].

Rlabel

Questo comando mette un'etichetta sopra l'asse r. L'etichetta verrà disegnata indipendentemente dal fatto che il grafico sia in modalità polare o meno. Vedere **set xlabel (p. 225)** per ulteriori keyword.

Rmargin

Il comando **set rmargin** imposta la dimensione del margine destro. Si prega di vedere **set margin (p. 176)** per i dettagli.

Rrange

Il comando **set rrange** imposta l'intervallo della coordinata radiale per un grafico (graph) in modalità polare. Questo ha l'effetto di impostare anche xrange e yrange. I risultanti xrange e yrange sono entrambi [-(rmax-rmin) : +(rmax-rmin)]. Tuttavia, se in seguito si modifica l'intervallo x o y, per esempio facendo uno zoom, ciò non cambia rrange, quindi i data point continuano a essere ritagliati rispetto a rrange. A differenza degli altri assi, l'autoscaling dell'asse raxis risulta sempre in rmin = 0. Il flag di autoscaling **reverse** è ignorato. Nota: Impostare un valore negativo per rmin può produrre risultati inaspettati.

Rtics

Il comando **set rtics** pone dei tic lungo l'asse polare. I tic e le etichette sono disegnati a destra dell'origine. La keyword **mirror** fa sì che siano disegnati anche a sinistra dell'origine. Vedere **polar (p. 202)**, **set xtics (p. 228)**, e **set mxtics (p. 182)** per la discussione delle keyword.

Samples

La frequenza di campionamento predefinita delle funzioni, o per interpolare i dati, può essere cambiata con il comando **set samples**. Per cambiare l'intervallo di campionamento per un particolare grafico, vedere **plot sampling (p. 127)**.

Sintassi:

```
set samples <samples_1> {,<samples_2>}
show samples
```

Per default, il campionamento è impostato su 100 punti. Una frequenza di campionamento più elevata produrrà grafici più precisi, ma richiederà più tempo. Questo parametro non ha effetto sul plotting di file di dati a meno che non venga utilizzata una delle opzioni di interpolazione/approssimazione. Vedere **plot smooth (p. 117)** re dati 2D e **set cntrparam (p. 144)** e **set dgrid3d (p. 152)** re dati 3D.

Quando si sta realizzando un grafico 2D, solo il valore di <samples_1> è rilevante.

Quando un grafico a superficie viene realizzato senza la rimozione delle linee nascoste, il valore dei campioni specifica il numero di campioni che devono essere valutati per le isolinee. Ogni linea iso-v avrà <sample_1> campioni e ogni linea iso-u avrà <sample_2> campioni. Se si specifica solo <samples_1>, <samples_2> sarà impostato allo stesso valore di <samples_1>. Vedere anche **set isosamples (p. 164)**.

Size

Sintassi:

```
set size {{no}square | ratio <r> | noratio} {<xscale>,<yscale>}
show size
```

I valori <xscale> e <yscale> sono i fattori di scala per la dimensione del grafico, che comprende il grafico (graph), le etichette e i margini.

Nota importante:

```
Nelle versioni precedenti di gnuplot, alcuni tipi di terminale usavano
i valori di ‘set size‘ per controllare anche la dimensione del canvas;
altri non lo facevano. Quasi tutti i terminali ora seguono la seguente
convenzione:
```

set term <terminal_type> size <XX>, <YY> controlla la dimensione del file di output, o **canvas**. Si prega di consultare la documentazione dei singoli terminali per i valori consentiti dei parametri di dimensione. Per default, il grafico riempirà questo canvas.

set size <XX>, <YY> scala il grafico stesso rispetto alla dimensione del canvas. Valori di scala inferiori a 1 faranno sì che il grafico non riempia l'intero canvas. Valori di scala maggiori di 1 faranno sì che solo una porzione del grafico si adatti al canvas. Si prega di notare che impostare valori di scala maggiori di 1 può causare problemi su alcuni tipi di terminale.

ratio fa sì che **gnuplot** cerchi di creare un grafico (graph) con un rapporto d'aspetto di <r> (il rapporto tra la lunghezza dell'asse y e quella dell'asse x) entro la porzione del grafico specificata da <xscale> e <yscale>.

Il significato di un valore negativo per <r> è diverso. Se <r>=-1, gnuplot cerca di impostare le scale in modo che l'unità abbia la stessa lunghezza su entrambi gli assi x e y. Questo è l'equivalente 2D del comando 3D **set view equal xy**. Se <r>=-2, l'unità su y ha il doppio della lunghezza dell'unità su x, e così via.

Il successo di **gnuplot** nel produrre il rapporto d'aspetto richiesto dipende dal terminale selezionato. L'area del grafico (graph) sarà il rettangolo più grande di rapporto d'aspetto <r> che si adatterà alla porzione specificata dell'output (lasciando margini adeguati, naturalmente).

set size square è un sinonimo per **set size ratio 1**.

Sia **noratio** che **nosquare** riportano il grafico (graph) al rapporto d'aspetto predefinito del terminale, ma non riportano <xscale> o <yscale> ai loro valori predefiniti (1.0).

ratio e **square** non hanno effetto sui grafici 3D, ma influenzano le proiezioni 3D create usando **set view map**. Vedere anche **set view equal (p. 222)**, che costringe gli assi x e y di un 3D sulla stessa scala.

Esempi:

Per impostare la dimensione in modo che il grafico riempia il canvas disponibile:

```
set size 1,1
```

Per rendere il grafico (graph) dimezzato e quadrato usare:

```
set size square 0.5,0.5
```

Per rendere il grafico (graph) due volte più alto che largo usare:

```
set size ratio 2
```

Spiderplot

Il comando **set spiderplot** cambia l'interpretazione delle coordinate in un sistema polare in cui ogni data point è mappato a una posizione lungo un asse radiale. paxis 1 è sempre verticale; gli assi da 2 a N procedono in senso orario con spaziatura uniforme. Il comando deve essere emesso prima di plottare. Ha effetti aggiuntivi che equivalgono a

```
set style data spiderplot
unset border
unset tics
set key noautotitle
set size ratio 1.0
```

Usare **reset** per ripristinarli dopo il plotting.

Style

Gli stili di plotting predefiniti sono scelti con i comandi **set style data** e **set style function**. Vedere **plot with (p. 131)** vedere X per informazioni su come sovrascrivere lo stile di plotting predefinito per funzioni individuali e set di dati. Vedere **plotting styles (p. 63)** o **plot with (p. 131)** per un elenco completo degli stili.

Sintassi:

```
set style function <style>
set style data <style>
show style function
show style data
```

Si possono anche impostare stili predefiniti per specifici elementi di plotting.

Sintassi:

```
set style arrow <n> <arrowstyle>
set style boxplot <boxplot style options>
set style circle radius <size> {clip|noclip}
set style ellipse size <size> units {xy|xx|yy} {clip|noclip}
set style fill <fillstyle>
set style histogram <histogram style options>
set style line <n> <linestyle>
set style rectangle <object options> <linestyle> <fillstyle>
set style textbox {<n>} {opaque|transparent} {{no}border} {fillcolor}
```

Set style arrow

Ogni terminale ha un set predefinito di tipi di frecce e punti, che può essere visto usando il comando **test**. **set style arrow** definisce un set di tipi e spessoi di freccia e tipi e dimensioni di punto in modo da potervi fare riferimento in seguito tramite un indice invece di ripetere tutte le informazioni ad ogni invocazione.

Sintassi:

```
set style arrow <index> default
set style arrow <index> {nohead | head | backhead | heads}
                        {size <length>,<angle>{,<backangle>} {fixed}}
                        {filled | empty | nofilled | noborder}
                        {front | back}
                        { {linestyle | ls <line_style>}
                          | {linetype | lt <line_type>}
```

```
                                {linewidth | lw <line_width>}
                                {linecolor | lc <colorspec>}
                                {dashtype | dt <dashtype>} }
    unset style arrow
    show style arrow
```

<index> è un intero che identifica lo stile di freccia.

Se viene dato **default**, tutti i parametri di stile di freccia sono impostati ai loro valori predefiniti.

Se l'indice <index> dello stile linea esiste già, solo i parametri forniti vengono cambiati mentre tutti gli altri vengono conservati. In caso contrario, tutti i valori non definiti sono impostati sui valori predefiniti.

Specificando **nohead** si ottengono frecce disegnate senza punta — ovvero un segmento di linea. Questo consente un altro modo per disegnare un segmento di linea sul grafico. Per default, le frecce hanno una sola punta. Specificando **heads** vengono disegnate punte della freccia su entrambe le estremità della linea.

La dimensione della punta può essere modificata usando **size** <**length**>,<**angle**> o **size** <**length**>,<**angle**>,<**backangle**>, dove <**length**> definisce la lunghezza di ogni ramo della punta della freccia e <**angle**> l'angolo (in gradi) che formano con la freccia. <**Length**> è in unità dell'asse x; questo può essere cambiato da **first**, **second**, **graph**, **screen**, o **character** prima di <length>; vedere **coordinate** **(p. 33)** per i dettagli.

Per default la dimensione della punta della freccia è ridotta per le frecce molto corte. Questo può essere disabilitato usando la keyword **fixed** dopo il comando **size**.

<**backangle**> è l'angolo (in gradi) che i rami posteriori formano con la freccia (nella stessa direzione di <**angle**>). Viene ignorato se lo stile è **nofilled**.

Specificando **filled** si generano punte di freccia piene, con una linea di contorno intorno alla punta della freccia. Specificando **noborder** si generano punte di freccia piene senza bordo. In questo caso l'estremità della punta della freccia si trova esattamente sul punto finale del vettore e la punta della freccia è leggermente più piccolanel complesso. Le frecce tratteggiate dovrebbero usare sempre **noborder**, poiché un bordo tratteggiato è sgradevole. Non tutti i terminali supportano punte delle frecce piene.

Lo stile di linea può essere selezionato da una lista di stili di linea definita dall'utente (vedere **set style line** **(p. 210)**) o può essere definito qui fornendo valori per <**line_type**> (un indice dall'elenco predefinito degli stili) e/o <**line_width**> (che è un moltiplicatore per lo spessore predefinito).

Si noti, tuttavia, che se è stato selezionato uno stile linea definito dall'utente, le sue proprietà (tipo e spessore) non possono essere alterate semplicemente emettendo un altro comando **set style arrow** con l'indice appropriato e **lt** o **lw**.

Se è dato **front**, le frecce sono scritte sopra i dati rappresentati sul grafico (graph). Se è dato **back** (il default), la freccia è scritta sotto i dati rappresentati sul grafico (graph). L'uso di **front** impedirà che una freccia sia oscurata da dati densi.

Esempi:

Per disegnare una freccia senza punta e di spessore doppio, usare:

```
    set style arrow 1 nohead lw 2
    set arrow arrowstyle 1
```

Vedere anche **set arrow** **(p. 137)** per altri esempi.

Boxplot (Grafico a scatola e baffi)

Il comando **set style boxplot** permette di cambiare il layout dei grafici creati utilizzando lo stile di grafico **boxplot**.

Sintassi:

```
    set style boxplot {range <r> | fraction <f>}
```

```
{{no}outliers} {pointtype <p>}
{candlesticks | financebars}
{medianlinewidth <width>}
{separation <x>}
{labels off | auto | x | x2}
{sorted | unsorted}
```

La scatola nel boxplot compre sempre l'intervallo di valori dal primo quartile al terzo quartile dei data point. Il limite dei baffi che si estendono dalla scatola può essere controllato in due modi diversi. Per default i baffi si estendono da ogni estremità della scatola per un intervallo pari a 1.5 volte l'intervallo interquartile (cioè l'altezza verticale della scatola vera e propria). Ogni baffo è troncato all'indietro verso la mediana in modo che termini a un valore y appartenente a qualche punto del set di dati. Poiché potrebbe non esserci alcun punto il cui valore sia esattamente 1.5 volte la distanza interquartile, il baffo potrebbe essere più corto del suo intervallo nominale. Questa impostazione predefinita corrisponde a

```
set style boxplot range 1.5
```

In alternativa, è possibile specificare la frazione del numero totale di punti che i baffi dovrebbero coprire. In questo caso l'intervallo è esteso simmetricamente dal valore mediano fino a comprendere la frazione richiesta del data set. Anche in questo caso ogni baffo è costretto a fermarsi in un punto del data set. Per coprire il 95% dei punti nel set

```
set style boxplot fraction 0.95
```

Tutti i punti che si trovano al di fuori della portata dei baffi sono considerati outlier. Per impostazione predefinita questi sono disegnati come cerchi individuali (pointtype 7). L'opzione **nooutliers** lo disabilita. Se gli outlier non vengono disegnati, essi non contribuiscono all'autoscaling.

Per default i boxplot sono disegnati in uno stile simile al candlesticks, ma è disponibile l'opzione di usare invece uno stile simile alle barre finanziarie.

Una barra trasversale che indica la mediana è disegnata usando lo stesso tipo di linea del contorno della scatola. Se si desidera una linea più spessa per la mediana

```
set style boxplot medianlinewidth 2.0
```

Se non si desidera alcuna linea mediana, impostare questo a 0.

Se la specifica di utilizzo di un boxplot contiene una quarta colonna, i valori in quella colonna saranno interpretati come i livelli discreti di una variabile fattore. In questo caso si possono disegnare più boxplot, tanti quanti sono i livelli della variabile fattore. Questi boxplot saranno disegnati uno accanto all'altro, e la distanza tra loro è, per default, 1.0 (in unità dell'asse x). Questa distanza può essere modificata dall'opzione **separation**.

L'opzione **labels** governa come e dove questi boxplot (ognuno dei quali rappresenta una parte del dataset) sono etichettati. Per default il valore del fattore è inserito come etichetta tic sull'asse orizzontale – x o x2, a seconda di quale viene utilizzata per il grafico stesso. Questa impostazione corrisponde all'opzione **labels auto**. Le etichette possono essere forzate a utilizzare uno degli assi x o x2 – opzioni **labels x** e **labels x2**, rispettivamente –, oppure possono essere disattivate del tutto con l'opzione **labels off**.

Per default i boxplot corrispondenti ai diversi livelli della variabile fattore non vengono ordinati; verranno disegnati nello stesso ordine in cui i livelli sono trovati nel file di dati. Questo comportamento corrisponde all'opzione **unsorted**. Se l'opzione **sorted** è attiva, i livelli vengono prima ordinati alfabeticamente, e i boxplot sono disegnati in questo ordine.

Le opzioni **separation**, **labels**, **sorted** e **unsorted** hanno effetto solo se a una quarta colonna viene data la specifica del grafico.

Vedere **boxplot (p. 66)**, **candlesticks (p. 67)**, **financebars (p. 71)**.

Set style data

Il comando **set style data** cambia lo stile di plotting predefinito per i grafici dei dati.

Sintassi:

```
set style data <plotting-style>
show style data
```

Vedere **plotting styles (p. 63)** per le scelte. **show style data** mostra lo stile di plotting dei dati predefinito.

Set style fill

Il comando **set style fill** è usato per impostare lo stile predefinito degli elementi del grafico nei grafici con box, istogrammi, candlesticks e filledcurves. Questo default può essere sostituito da stili di riempimento allegati ai grafici individuali.

Si noti che c'è uno stile di riempimento predefinito separato per i rettangoli creati da **set obj**. Vedere **set style rectangle (p. 212)**.

Sintassi:

```
set style fill {empty
                | {transparent} solid {<density>}
                | {transparent} pattern {<n>}}
               {border {lt} {lc <colorspec>} | noborder}
```

L'opzione **empty** fa sì che le aree piene non vengano riempite. Questo è il default.

L'opzione **solid** provoca il riempimento con una tinta unita, se il terminale lo supporta. Il parametro <density> specifica l'intensità del colore di riempimento. A <density> di 0.0, la casella è vuota, a <density> di 1.0, l'area interna è dello stesso colore del tipo di linea corrente. Alcuni tipi di terminale possono variare la densità in modo continuo; altri implementano solo alcuni livelli di riempimento parziale. Se non è fornito alcun parametro <density>, il valore predefinito è 1.

L'opzione **pattern** fa sì che il riempimento venga fatto con un pattern di riempimento fornito dal driver del terminale. Il tipo e il numero di pattern di riempimento disponibili dipendono dal driver del terminale. Se vengono plottati più dataset che usano box pieni, il pattern scorre attraverso tutti i tipi di pattern disponibili, iniziando dal pattern <n>, così come il tipo di linea scorre per i grafici con multiple linee.

Il colore di riempimento (**fillcolor <colorspec>**) è diverso dallo stile di riempimento. Cioè gli elementi del grafico elementi o gli oggetti possono condividere uno stile di riempimento pur mantenendo colori separati. Nella maggior parte dei casi in cui viene accettato uno stile di riempimento si può anche specificare un colore di riempimento. Il colore di riempimento (fillcolor) può essere abbreviato **fc**. Altrimenti il colore di riempimento è preso dal tipo di linea corrente. Esempio:

```
plot FOO with boxes fillstyle solid 1.0 fillcolor "cyan"
```

Set style fill border La semplice keyword **border** fa sì che l'oggetto riempito sia circondato da una linea continua del tipo di linea e del colore attuali. È possibile cambiare il colore di questa linea aggiungendo un tipo di linea o un colore della linea. **noborder** specifica che non viene disegnata alcuna linea di contorno. Esempi:

```
# Riempimento a mezza intensità, bordo a piena intensità nello stesso colore
set style fill solid 0.5 border
# Riempimento semitrasparente, bordo nero pieno (linetype -1)
set style fill transparent solid 0.5 border -1
# Riempimento pattern nel colore corrente, bordo usando il colore del linetype 5
plot ... with boxes fillstyle pattern 2 border lt 5
# Riempire l'area in ciano, bordo in blu
plot ... with boxes fillcolor "cyan" fs solid border linecolor "blue"
```

Nota: La proprietà del bordo di uno stile di riempimento ha effetto solo sui grafici disegnati con **with filledcurves** nella modalità predefinita (curva chiusa).

Set style fill transparent Alcuni terminali supportano l'attributo **transparent** per le aree riempite. Nel caso di aree a tinta unita trasparente, il parametro **density** è interpretato come un valore alpha; cioè, density 0 è completamente trasparente, density 1 è completamente opaca. Nel caso del riempimento a pattern trasparente, lo sfondo del pattern è completamente trasparente o completamente opaco.

Si noti che possono esistere ulteriori limitazioni per la creazione o la visualizzazione di grafici (graph) contenenti aree di riempimento trasparenti. Per esempio, il terminale png può usare il riempimento trasparente solo se l'opzione "truecolor" è impostata. Alcuni visualizzatori di pdf potrebbero non visualizzare correttamente le aree di riempimento anche se sono descritte correttemente nel file pdf. Ghostscript/gv non visualizza correttamente le aree di riempimento a pattern anche se le stampanti PostScript generalmente non hanno problemi.

Set style function

Il comando **set style function** cambia lo stile di plotting predefinito per i grafici di funzione (ad esempio linee, punti, filledcurves). Vedere **plotting styles (p. 63)**.

Sintassi:

```
set style function <plotting-style>
show style function
```

Set style increment

Nota: Questo comando è stato deprecato. Si prega invece di usare il più recente comando **set linetype**, il quale ridefinisce i tipi di linea stessi anziché cercare uno stile di linea temporaneo adatto da sostituire. Vedere **set linetype (p. 173)**

Sintassi:

```
set style increment {default|userstyles}
show style increment
```

Per default, grafici (plots) successivi all'interno dello stesso grafico (graph) utilizzeranno tipi di linea successivi dal set predefinito per il tipo di terminale corrente. Tuttavia, scegliere **set style increment user** permette di passare attraverso gli stili di linea definiti dall'utente piuttosto che attraverso i tipi di linea predefiniti.

Esempio:

```
set style line 1 lw 2 lc rgb "gold"
set style line 2 lw 2 lc rgb "purple"
set style line 4 lw 1 lc rgb "sea-green"
set style increment user

plot f1(x), f2(x), f3(x), f4(x)
```

dovrebbe plottare le funzioni f1, f2, f4 nei propri 3 stili di linea appena definiti. Se uno stile di linea definito dall'utente non viene trovato, allora viene usato al suo posto il corrispondente tipo di linea predefinito. Ad es., nell'esempio precedente, f3(x) sarà plottato usando il tipo di linea 3 predefinito.

Set style line

Ogni terminale ha un set predefinito di tipi di linee e di punti, che può essere visto usando il comando **test**. **set style line** definisce un set di tipi e spessori di linea, e tipi e dimensioni di punti, in modo da potervi fare riferimento in seguito tramite un indice invece di ripetere tutte le informazioni ad ogni invocazione.

Sintassi:

```
set style line <index> default
set style line <index> {{linetype  | lt} <line_type> | <colorspec>}
                       {{linecolor | lc} <colorspec>}
                       {{linewidth | lw} <line_width>}
                       {{pointtype | pt} <point_type>}
                       {{pointsize | ps} <point_size>}
                       {{pointinterval | pi} <interval>}
                       {{pointnumber | pn} <max_symbols>}
                       {{dashtype | dt} <dashtype>}
                       {palette}
unset style line
show style line
```

default imposta tutti i parametri di uno stile linea a quelli del tipo di linea con lo stesso indice.

Se l'indice <index> dello stile di linea esiste già, solo i parametri forniti vengono cambiati mentre tutti gli altri vengono conservati. In caso contrario, tutti i valori non definiti sono impostati sui valori predefiniti.

Gli stili di linea creati da questo meccanismo non sostituiscono gli stili predefiniti di tipo di linea; possono essere usati entrambi. Gli stili di linea sono temporanei. Essi vengono persi ogni volta che si esegue un comando **reset**. Per ridefinire il tipo di linea stesso, si prega di vedere **set linetype (p. 173)**.

I tipi di linea e di punto sono predefiniti al valore dell'indice. Il simbolo esatto che viene disegnato per quel valore di indice può variare da un tipo di terminale all'altro.

Lo spessore della linea e la dimensione del punto sono moltiplicatori per lo spessore e la dimensione di default del terminale corrente (ma si noti che <point_size> qui non è influenzato dal moltiplicatore dato dal comando **set pointsize**).

Il **pointinterval** controlla la spaziatura tra i punti in un grafico disegnato con stile **linespoints**. Il valore predefinito è 0 (viene disegnato ogni punto). Per esempio, **set style line N pi 3** definisce uno stile di linea che utilizza il tipo di punto N, dimensione del punto e spessore della linea uguali ai valori predefiniti per il terminale, e disegnerà ogni terzo punto nei grafici che utilizzano **with linespoints**. Un valore negativo per l'intervallo è trattato allo stesso modo di un valore positivo, eccetto che alcuni terminali cercheranno di interrompere la linea dove passa attraverso il simbolo del punto.

La proprietà **pointnumber** è simile a **pointinterval** eccetto che, piuttosto che plottare ogni N punti, limita il numero totale di punti a N.

Non tutti i terminali supportano le funzionalità **linewidth** e **pointsize**; se non è supportata, l'opzione sarà ignorata.

I colori indipendenti dal terminale possono essere assegnati usando **linecolor <colorspec>** o **linetype <colorspec>**, abbreviati **lc** o **lt**. Questo richiede di dare una terna di colori RGB, un nome di un colore della palette conosciuto, un indice frazionario nella palette corrente, o un valore costante dalla mappatura corrente della palette su cbrange. Vedere **colors (p. 50)**, **colorspec (p. 50)**, **set palette (p. 190)**, **colornames (p. 147)**, **cbrange (p. 236)**.

set style line <n> linetype <lt> imposterà sia un pattern punto/trattino che un colore dipendenti dal terminale. I comandi **set style line <n> linecolor <colorspec>** o **set style line <n> linetype <colorspec>** imposteranno un nuovo colore di linea lasciando invariato il pattern punto/trattino esistente.

In modalità 3d (comando **splot**), la keyword speciale **palette** è consentita come abbreviazione di "linetype palette z". Il valore del colore corrisponde al valore z (elevazione) dello splot, e varia uniformemente lungo una linea o superficie.

Esempi: Supponiamo che le linee predefinite per gli indici 1, 2, e 3 sono rispettivamente rosso, verde, e blu, e le forme di punto predefinite per gli stessi indici sono rispettivamente un quadrato, una croce e un triangolo. Quindi

```
set style line 1 lt 2 lw 2 pt 3 ps 0.5
```

definisce un nuovo stile linea che è verde e due volte lo spessore predefinito e un nuovo stile di punto che è

un triangolo dimezzato. I comandi

```
set style function lines
plot f(x) lt 3, g(x) ls 1
```

creeranno un grafico di f(x) usando la linea blu predefinita e un grafico di g(x) usando la larga linea verde definita dall'utente. In modo simile i comandi

```
set style function linespoints
plot p(x) lt 1 pt 3, q(x) ls 1
```

creeranno un grafico di p(x) usando i triangoli predefiniti collegati da una linea rossa e q(x) usando piccoli triangoli collegati da una linea verde.

```
splot sin(sqrt(x*x+y*y))/sqrt(x*x+y*y) w l pal
```

crea un grafico a superficie usando colori uniformi secondo la **palette**. Si noti, che questo funziona solo su alcuni terminali. Vedere anche **set palette (p. 190)**, **set pm3d (p. 197)**.

```
set style line 10 linetype 1 linecolor rgb "cyan"
```

assegnerà lo stile linea 10 come una linea ciano piena su qualsiasi terminale che supporta i colori rgb.

Set style circle

Sintassi:

```
set style circle {radius {graph|screen} <R>}
                 {{no}wedge}
                 {clip|noclip}
```

Questo comando imposta il raggio predefinito utilizzato nello stile di grafico "with circles". Esso si applica ai grafici di dati con solo 2 colonne di dati (x,y) e ai grafici di funzioni. L'impostazione predefinita è "set style circle radius graph 0.02". **Nowedge** disattiva il disegno dei due raggi che collegano le estremità di un arco al centro. L'impostazione predefinita è **wedge**. Questo parametro non ha effetto sui cerchi completi. **Clip** ritaglia il cerchio ai confini del grafico, **noclip** lo disabilita. L'impostazione predefinita è **clip**.

Set style rectangle

I rettangoli definiti con il comando **set object** possono avere stili individuali. Tuttavia, se all'oggetto non è assegnato uno stile privato, allora eredita un default che viene preso dal comando **set style rectangle**.

Sintassi:

```
set style rectangle {front|back} {lw|linewidth <lw>}
                    {fillcolor <colorspec>} {fs <fillstyle>}
```

Vedere **colorspec (p. 50)** e **fillstyle (p. 209)**. **fillcolor** può essere abbreviato come **fc**.

Esempi:

```
set style rectangle back fc rgb "white" fs solid 1.0 border lt -1
set style rectangle fc linsestyle 3 fs pattern 2 noborder
```

I valori predefiniti corrispondono a un riempimento solido con il colore di sfondo e un bordo nero.

Set style ellipse

Sintassi:

```
set style ellipse {units xx|xy|yy}
                  {size {graph|screen} <a>, {{graph|screen} <b>}}
                  {angle <angle>}
                  {clip|noclip}
```

Questo comando determina se i diametri delle ellissi vengono interpretati nelle stesse unità o meno. Il default è **xy**, il che significa che il diametro maggiore (primo asse) delle ellissi sarà interpretato nelle stesse unità dell'asse x (o x2), mentre il diametro minore (secondo asse) nelle unità dell'asse y (o y2). In questa modalità il rapporto degli assi dell'ellisse dipende dalle scale degli assi del grafico e dal rapporto d'aspetto del grafico. Quando è impostato su **xx** o **yy**, entrambi gli assi di tutte le ellissi saranno interpretati nelle stesse unità. Questo significa che il rapporto tra gli assi delle ellissi plottate sarà corretto anche dopo la rotazione, ma la loro estensione verticale o orizzontale non sarà corretta.

Questa è un'impostazione globale che influenza tutte le ellissi, sia quelle definite come oggetti che quelle generate con il comando **plot**, tuttavia, il valore di **units** (unità) può anche essere ridefinito in base a grafico e a oggetto.

È anche possibile impostare una dimensione predefinita per le ellissi con la keyword **size**. Questa dimensione predefinita si applica ai grafici di dati con solo 2 colonne di dati (x,y) e ai grafici di funzioni. I due valori sono interpretati come i diametri maggiore e minore (al contrario di assi semi-maggiore e semi-minore) dell'ellisse.

Il default è "set style ellipse size graph 0.05,0.03".

Infine, ma non meno importante, è possibile impostare l'orientamento predefinito con la keyword **angle**. L'orientamento, che è definito come l'angolo tra l'asse maggiore l'asse maggiore dell'ellisse e l'asse x del grafico, deve essere dato in gradi.

Clip ritaglia l'ellisse ai confini del grafico, **noclip** lo disabilita. L'impostazione predefinita è **clip**.

Per definire oggetti ellisse, vedere **set object ellipse (p. 186)**; per lo stile di grafico 2D, vedere **ellipses (p. 69)**.

Set style parallelaxis

Sintassi:

```
set style parallelaxis {front|back} {line-properties}
```

Determina il tipo di linea e il layer per disegnare gli assi verticali nei grafici **with parallelaxes**. Vedere **with parallelaxes (p. 81)**, **set paxis (p. 195)**.

Set style spiderplot

Sintassi:

```
set style spiderplot
          {fillstyle <fillstyle-properties>}
          {<line-properties> | <point-properties>}
```

Questo comando controlla l'aspetto predefinito degli spiderplot. Le proprietà di riempimento, linea e punto possono essere modificate nel primo componente del comando plot. L'aspetto generale del grafico è anche influenzato da altre impostazioni come **set grid spiderplot**. Vedere anche **set paxis (p. 195)**, **spiderplot (p. 83)**. Esempio:

```
# Lo spiderplot predefinito sarà un poligono con un bordo spesso ma
# nessun riempimento
set style spiderplot fillstyle empty border lw 3
# Questo metterà inoltre un cerchio aperto (pt 6) su ogni asse
plot for [i=1:6] DATA pointtype 6 pointsize 3
```

Set style textbox

Sintassi:

```
set style textbox {<boxstyle-index>}
          {opaque|transparent} {fillcolor <color>}
          {{no}border {<bordercolor>}}{linewidth <lw>}
          {margins <xmargin>,<ymargin>}
```

Questo comando controlla l'aspetto delle etichette con l'attributo 'boxed'. I tipi di terminale che non supportano il testo "boxed" (in una casella) ignoreranno questo stile. Nota: L'implementazione per alcuni terminali (svg, latex) è incompleta. La maggior parte dei terminali non può posizionare correttamente una casella intorno al testo ruotato.

Si possono definire tre stili di casella di testo (textbox) numerati. Se non viene dato alcun indice di boxstyle <bs>, viene cambiato lo stile predefinito (non numerato). Esempio:

```
# lo stile predefinito ha solo un bordo nero
set style textbox transparent border lc "black"
# lo stile 2 (bs 2) ha uno sfondo azzurro senza bordo
set style textbox 2 opaque fc "light-cyan" noborder
set label 1 "I'm in a box" boxed
set label 2 "I'm blue" boxed bs 2
```

Surface

Il comando **set surface** è rilevante solo per i grafici 3D (**splot**).

Sintassi:

```
set surface {implicit|explicit}
unset surface
show surface
```

unset surface farà sì che **splot** non disegni punti o linee corrispondenti a qualsiasi punto della funzione o del file di dati. Questo è utile soprattutto per disegnare solo le linee di contorno piuttosto che la superficie da cui sono state derivate. I contorni possono ancora essere disegnati sulla superficie, in base all'opzione **set contour**. Per disattivare la superficie per una singola funzione o file di dati lasciandone attive altre, usare la keyword **nosurface** nel comando **splot**. La combinazione **unset surface; set contour base** è utile per visualizzare contorni sulla base della griglia. Vedere anche **set contour (p. 147)**.

Se un data set 3D è riconoscibile come una mesh (griglia) allora per default il programma tratta implicitamente lo stile di grafico **with lines** come se richiedesse una superficie a griglia. Vedere **grid_data (p. 241)**. Il comando **set surface explicit** sopprime questa espansione, plottando solo le singole linee descritte da blocchi di dati separati nel file di input. Una superficie con griglia può ancora essere plottata richiedendo esplicitamente splot **with surface**.

Table

Quando la modalità **table** è abilitata, i comandi **plot** e **splot** stampano una una tabella di testo multicolonna di valori

```
X Y {Z} <flag>
```

piuttosto che creare un vero e proprio grafico sul terminale corrente. Il carattere flag è "i" se il punto è nell'intervallo attivo, "o" se è fuori intervallo, o "u" se è indefinito. Il formato dei dati è determinato dal formato dei segni di tic dell'asse (vedere **set format (p. 157)**), e le colonne sono separate da spazi singoli. Questo può essere utile se si desidera generare i contorni e poi salvarli per un ulteriore utilizzo. Lo

stesso metodo può essere usato per salvare i dati interpolati (vedere **set samples (p. 204)** e **set dgrid3d (p. 152)**).

Sintassi:

```
set table {"outfile" | $datablock} {append}
          {separator {whitespace|tab|comma|"<char>"}}
plot <whatever>
unset table
```

L'output tabulare successivo viene scritto su "outfile", se specificato, altrimenti viene scritto su stdout o su un altro valore corrente di **set output**. Se **outfile** esiste, verrà sostituito a meno che non venga data la keyword **append**. In alternativa, l'output tabulare può essere reindirizzato a un datablock denominato. I nomi dei datablock iniziano con '$', vedere anche **inline data (p. 48)**. È necessario scrivere esplicitamente **unset table** per tornare al normale plotting sul terminale corrente.

Il carattere **separator** può essere usato per produrre file csv (valori separati da virgola). Questa modalità ha effetto solo sullo stile di grafico **with table**. Vedere **plot with table (p. 215)**.

Plot with table

Questa discussione si applica solo allo stile di grafico speciale **with table**.

Per evitare qualsiasi elaborazione dipendente da uno stile dei dati di input che vengono tabulati (smoothing, errorbar expansion, secondary range checking, ecc), o per aumentare il numero di colonne che possono essere tabulate, usare la keyword "table" invece di un normale stile di grafico. In questo caso l'output non contiene un'extra, ultima, colonna di flag **i**, **o**, **u** indicata inrange/outrange/undefined (nell'intervallo/fuori intervallo/indefinita). La destinazione dell'output deve essere prima specificata con **set table <where>**. Per esempio

```
set table $DATABLOCK1
plot <file> using 1:2:3:4:($5+$6):(func($7)):8:9:10 with table
```

Poiché in questo caso non c'è un vero e proprio stile di grafico, le colonne non corrispondono ad assi specifici. Pertanto xrange, yrange, ecc. sono ignorati.

Se un termine **using** valuta a una stringa, la stringa viene tabulata. I dati numerici sono sempre scritti con il formato %g. Se si desidera un altro formato, usare sprintf o gprintf per creare una stringa formattata.

```
plot <file> using ("File 1"):1:2:3 with table
plot <file> using (sprintf("%4.2f",$1)) : (sprintf("%4.2f",$3)) with table
```

Per creare un file csv usare

```
set table "tab.csv" separator comma
plot <foo> using 1:2:3:4 with table
```

[SPERIMENTALE] Per selezionare solo un sottoinsieme di data point per la tabulazione si può fornire una condizione di filtro di input (**if <expression>**) alla fine del comando. Si noti che il filtro di input può fare riferimento a colonne di dati che non sono parte dell'output. Questa funzionalità può cambiare sostanzialmente prima di apparire in una versione rilasciata di gnuplot.

```
plot <file> using 1:2:($4+$5) with table if (strcol(3) eq "Red")
plot <file> using 1:2:($4+$5) with table if (10. < $1 && $1 < 100.)
plot <file> using 1:2:($4+$5) with table if (filter($6,$7) != 0)
```

Terminal

gnuplot supporta molti dispositivi grafici diversi. È necessario usare **set terminal** per dire a **gnuplot** che tipo di output generare. Usare **set output** per reindirizzare l'output verso un file o un dispositivo.

Sintassi:

```
set terminal {<terminal-type> | push | pop}
show terminal
```

Se <terminal-type> è omesso, **gnuplot** elencherà i tipi di terminale disponibili. <terminal-type> può essere abbreviato.

Se vengono usati insieme **set terminal** e **set output**, è più sicuro dare prima **set terminal**, perché alcuni terminali impostano un flag che è necessario in alcuni sistemi operativi.

Alcuni terminali hanno molte opzioni aggiuntive. Le opzioni usate da una precedente invocazione **set term** <**term**> <**options**> di un dato <**term**> vengono ricordate, quindi un successivo **set term** <**term**> non le resetta. Questo aiuta nella stampa, per esempio, quando si passa tra diversi terminali — le opzioni precedenti non devono essere ripetute.

Il comando **set term push** ricorda il terminale in uso, comprese le sue impostazioni, mentre **set term pop** lo ripristina. Questo è equivalente a **save term** e **load term**, ma senza accedere al filesystem. Perciò possono essere utilizzati per ottenere il ripristino indipendente dalla piattaforma del terminale dopo la stampa, per esempio. Dopo l'avvio di gnuplot, il terminale predefinito o quello dal file **startup** viene inserito automaticamente. Pertanto gli script portatili possono contare sul fatto che **set term pop** ripristina il terminale predefinito su una data piattaforma a meno che un altro terminale sia stato inserito esplicitamente.

Per maggiori informazioni, vedere **lista completa di terminali (p. 248)**.

Termoption

Il comando **set termoption** permette di cambiare il comportamento del terminale in uso senza richiedere un nuovo comando **set terminal**. Solo una opzione può essere cambiata per ogni comando, e solo un piccolo numero di opzioni possono essere cambiate in questo modo. Attualmente le uniche opzioni accettate sono

```
set termoption {no}enhanced
set termoption font "<fontname>{,<fontsize>}"
set termoption fontscale <scale>
set termoption {linewidth <lw>}{lw <lw>}
```

Theta

I grafici di coordinate polari sono orientati di default in modo tale che theta = 0 sia sul lato destro del grafico, con theta che aumenta man mano che si procede in senso antiorario in modo che theta = 90 gradi sia in alto. **set theta** permette di cambiare l'origine e la direzione della coordinata angolare polare theta.

```
set theta {right|top|left|bottom}
set theta {clockwise|cw|counterclockwise|ccw}
```

unset theta ripristina lo stato predefinito "set theta right ccw".

Tics

Il comando **set tics** controlla i segni di tic e le etichette su tutti gli assi contemporaneamente.

I tic possono essere disattivati con il comando **unset tics**, e possono essere attivati (lo stato predefinito) con **set tics**. Il controllo preciso dei tic sui singoli assi è possibile usando i comandi alternativi **set xtics**, **set ztics**, ecc.

Sintassi:

```
set tics {axis | border} {{no}mirror}
         {in | out} {front | back}
         {{no}rotate {by <ang>}} {offset <offset> | nooffset}
```

```
          {left | right | center | autojustify}
          {format "formatstring"} {font "name{,<size>}"} {{no}enhanced}
          { textcolor <colorspec> }
set tics scale {default | <major> {,<minor>}}
unset tics
show tics
```

Le opzioni possono essere applicate a un singolo asse (x, y, z, x2, y2, cb), per esempio

```
set xtics rotate by -90
unset cbtics
```

Tutti i segni di tic sono disegnati usando le stesse proprietà della linea del bordo del grafico (vedere **set border (p. 140)**).

Set tics **back** o **front** si applica a tutti gli assi contemporaneamente, ma solo per i grafici 2D (non splot). Controlla se i tic sono posizionati dietro o davanti agli elementi del grafico, nel caso in cui ci sia una sovrapposizione.

axis o **border** dice a gnuplot **gnuplot** dice a x di mettere i tic (sia i tic stessi e le etichette che li accompagnano) lungo l'asse o il bordo, rispettivamente. Se l'asse è molto vicino al bordo, l'opzione **axis** sposterà le etichette dei tic all'esterno del bordo nel caso in cui il bordo sia stampato (vedere **set border (p. 140)**). In questo caso le impostazioni dei margini rilevanti saranno di solito ridimensionate male dall'algoritmo di layout automatico.

mirror dice a **gnuplot** di mettere i tic non etichettati nelle stesse posizioni sul bordo opposto. **nomirror** non lo fa.

in e **out** cambiano i segni di tic in modo che siano disegnati verso l'interno o verso l'esterno.

set tics scale controlla la dimensione dei segni di tic. Il primo valore <major> controlla i tic principali generati automaticamente o specificati dall'utente (livello 0). Il secondo valore controlla i tic secondari generati automaticamente o specificati dall'utente (livello 1). <major> è di default 1.0, <minor> è di default <major>/2. Valori aggiuntivi controllano la dimensione dei tic specificati dall'utente con livello 2, 3, ... Le dimensioni predefinite dei tic sono ripristinate da **set tics scale default**.

rotate chiede a **gnuplot** di ruotare il testo di 90 gradi, che sarà fatto se il driver del terminale in uso supporta la rotazione del testo. **norotate** annulla questa operazione. **rotate by** <**ang**> chiede la rotazione di <ang> gradi, supportata da alcuni tipi di terminale.

I valori predefiniti sono **border mirror norotate** per i tic sugli assi x e y, e **border nomirror norotate** per i tic sugli assi x2 e y2. Per l'asse z, il default è **nomirror**.

L'<offset> è specificato da x,y o da x,y,z, e può essere preceduto da **first**, **second**, **graph**, **screen**, o **character** per selezionare il sistema di coordinate. <offset> è l'offset dei testi dei tic dalle loro posizioni predefinite, mentre il sistema di coordinate predefinito è **character**. Vedere **coordinate (p. 33)** per i dettagli. **nooffset** disattiva l'offset.

Per impostazione predefinita, le etichette dei tic sono allineate automaticamente a seconda dell'asse e angolo di rotazione per produrre risultati esteticamente piacevoli. Se questo non è desiderato, l'allineamento può essere sovrascritto con una esplicita keyword **left**, **right** o **center**. **autojustify** ripristina il comportamento predefinito.

set tics senza opzioni ripristina i segni di tic speculari e rivolti verso l'interno per gli assi primari. Tutte le altre impostazioni vengono mantenute.

Vedere anche **set xtics (p. 228)** per un miglior controllo dei segni di tic principali (etichettati) e **set mxtics** per il controllo dei segni di tic secondari. Questi comandi forniscono il controllo di ogni asse in modo indipendente.

Ticslevel

Deprecato. Vedere **set xyplane (p. 233)**.

Ticscale

Il comando **set ticscale** è deprecato, usare invece **set tics scale**.

Timestamp

Il comando **set timestamp** inserisce l'ora e la data attuali nel margine del grafico.

Sintassi:

```
set timestamp {"<format>"} {top|bottom} {{no}rotate}
              {offset <xoff>{,<yoff>}} {font "<fontspec>"}
              {textcolor <colorspec>}
unset timestamp
show timestamp
```

La stringa di formato è usata per scrivere la data e l'ora. Il suo valore predefinito è quello che usa asctime(): "%a %b %d %H:%M:%S %Y" (giorno della settimana, nome del mese, giorno del mese, ore, minuti, secondi, anno a quattro cifre). Con **top** o **bottom** si può posizionare la marca temporale (timestamp) lungo il margine superiore sinistro o inferiore sinistro (predefinito: inferiore). **rotate** scrive la marca temporale verticalmente. Le costanti <xoff> e <yoff> sono offset che permettono di regolare la posizione con più precisione. <font> è usato per specificare il font con cui deve essere scritta l'ora.

L'abbreviazione **time** può essere usata al posto di **timestamp**.

Esempio:

```
set timestamp "%d/%m/%y %H:%M" offset 80,-2 font "Helvetica"
```

Vedere **set timefmt (p. 218)** per maggiori informazioni sulle stringhe di formato orario.

Timefmt

Questo comando imposta il formato predefinito utilizzato per inserire i dati temporali. Vedere **set xdata time (p. 224)**, **timecolumn (p. 40)**.

Sintassi:

```
set timefmt "<format string>"
show timefmt
```

I formati validi per entrambi **timefmt** e **timecolumn** sono:

Time Series timedata Format Specifiers	
Formato	Spiegazione
%d	giorno del mese, 1–31
%m	mese dell'anno, 1–12
%y	anno, 0–99
%Y	anno, 4-digit
%j	giorno dell'anno, 1–365
%H	ora, 0–24
%M	minuto, 0–60
%s	secondi dall'epoca di Unix (1970-01-01 00:00 UTC)
%S	secondo, intero 0–60 in output, (doppio) in input
%b	abbreviazione di tre caratteri del nome del mese
%B	nome del mese
%p	due caratteri corrispondono a uno dei seguenti: am AM pm PM

Qualsiasi carattere è permesso nella stringa, ma deve corrispondere esattamente. \t (tab) è riconosciuto. Gli ottali con barra rovesciata (\nnn) vengono convertiti in char. Se non c'è alcun carattere di separazione tra gli elementi ora/data, allora %d, %m, %y, %H, %M e %S leggono due cifre ciascuno. Se un punto decimale segue immediatamente il campo letto da %S, il decimale e tutte le cifre seguenti sono interpretate come un secondo frazionario. %Y legge quattro cifre. %j legge tre cifre. %b richiede tre caratteri e %B ne richiede quanti ne ha bisogno.

Gli spazi sono trattati in modo leggermente diverso. Uno spazio nella stringa rappresenta zero o più caratteri di spazio bianco nel file. Cioè, "%H %M" può essere usato per leggere "1220" e "12 20" così come "12 20".

Ogni set di caratteri non vuoti nei timedata vale come una colonna nella specifica **using n:n**. Così **11:11 25/12/76 21.0** consiste di tre colonne. Per evitare confusione, **gnuplot** richiede che si fornisca una completa specifica **using** se il proprio file contiene timedata.

Se il formato della data include il giorno o il mese in parole, la stringa di formato deve escludere questo testo. Ma può comunque essere stampato con gli specificatori "%a", "%A", "%b", o "%B". **gnuplot** determinerà il mese e il giorno della settimana corretti dai valori numerici. Vedere **set format (p. 157)** per maggiori dettagli su queste e altre opzioni per la stampa dei dati temporali.

Quando si leggono anni a due cifre con %y, i valori 69-99 si riferiscono al XX secolo, mentre i valori 00-68 si riferiscono al XXI secolo. NB: Ciò è conforme alla specifica UNIX98, ma le convenzioni variano notevolmente e i valori dell'anno a due cifre sono intrinsecamente ambigui.

Se il formato %p restituisce "am" o "AM", l'ora 12 sarà interpretata come ora 0. Se il formato %p restituisce "pm" o "PM", le ore < 12 saranno aumentate di 12.

Vedere anche **set xdata (p. 224)** and **ora/data (p. 62)** per maggiori informazioni.

Esempio:

```
set timefmt "%d/%m/%Y\t%H:%M"
```

dice a **gnuplot** di leggere data e ora separate da tab. (Ma si prega di guardare attentamente i propri dati — quello che è iniziato come un tab potrebbe essere stato convertito in spazi da qualche parte lungo la linea; la stringa di formato deve corrispondere a ciò che è effettivamente nel file.) Vedere anche `time data demo`.

Title

Il comando **set title** produce un titolo di grafico che è centrato nella parte superiore del grafico. **set title** è un caso speciale di **set label**.

Sintassi:

```
set title {"<title-text>"} {offset <offset>} {font "<font>{,<size>}"}
          {{textcolor | tc} {<colorspec> | default}} {{no}enhanced}
show title
```

Se <offset> è specificato da x,y o da x,y,z, il titolo viene spostato dall'offset dato. Può essere preceduto da **first**, **second**, **graph**, **screen**, o **character** per selezionare il sistema di coordinate. Vedere **coordinate (p. 33)** per i dettagli. Per default, viene usato il sistema di coordinate **character**. Per esempio, **"set title offset 0,-1"** cambierà solo l'offset y del titolo, spostando il titolo in basso di circa l'altezza di un carattere. La dimensione di un carattere dipende sia dal font che dal terminale.

<font> è usato per specificare il font con cui deve essere scritto il titolo; le unità del <size> (dimensione) del font dipendono dal terminale utilizzato.

textcolor <**colorspec**> cambia il colore del testo. <colorspec> può essere un tipo di linea, un colore rgb o una mappatura della palette. Vedere aiuto per **colorspec (p. 50)** e **palette (p. 40)**.

noenhanced richiede che il titolo non venga processato dal parser in modalità di testo avanzato, anche se la modalità di testo avanzato è attualmente attiva.

set title senza parametri cancella il titolo.

Vedere **sintassi (p. 60)** per i dettagli sull'elaborazione delle sequenze di backslash e la distinzione tra virgolette singole e doppie.

Tmargin

Il comando **set tmargin** imposta la dimensione del margine superiore. Vedere **set margin (p. 176)** per i dettagli.

Trange

Sintassi: set trange [tmin:tmax] L'intervallo della variabile parametrica t è utile in tre ambiti. 1) Nei comandi **plot** in modalità parametrica limita l'intervallo di
`campionamento per entrambe le funzioni generatrici. Vedere ‘set parametric‘, ‘set samples‘.`

2) Nei comandi **plot** in modalità polare limita o definisce l'intervallo
`del parametro angolare theta. Vedere ‘polar‘.`

3) Nei comandi **plot** o **splot** che utilizzano dati campionati monodimensionali
`tramite lo pseudofile "+". Vedere ‘sampling 1D‘, ‘special-filenames‘.`

Ttics

Il comando **set ttics** pone dei tic intorno al perimetro di un grafico polare. Questo è il bordo se **set border polar** è abilitato, altrimenti è il cerchio più esterno della griglia polare disegnata in corrispondenza del segno di tic più a destra lungo l'asse r. Vedere **set grid (p. 161)**, **set rtics (p. 204)**. La posizione angolare è sempre etichettata in gradi. L'intero perimetro può essere etichettato indipendentemente dall'impostazione attuale di trange. L'intervallo desiderato delle etichette dei tic deve essere dato come mostrato sotto. Possono essere impostate ulteriori proprietà dei segni di tic. Vedere **xtics (p. 228)**.

```
set ttics -180, 30, 180
set ttics add ("Theta = 0" 0)
set ttics font ":Italic" rotate
```

Urange

Sintassi: set urange [umin:umax] L'intervallo delle variabili parametriche u e v è utile in due ambiti. 1) **splot** in modalità parametrica. Vedere **set parametric (p. 195)**, **set isosamples (p. 164)**. 2) generare dati campionati bidimensionali sia per **plot** che per **splot** usando lo pseudofile "++". Vedere **sampling 2D (p. 128)**.

Variables

Il comando **show variables** elenca il valore attuale delle variabili definite dall'utente e variabili interne. Gnuplot definisce internamente le variabili i cui nomi iniziano con GPVAL_, MOUSE_, FIT_, and TERM_.

Sintassi:

```
show variables      # mostra le variabili che non iniziano con GPVAL_
show variables all  # mostra tutte le variabili, comprese quelle che
                    # iniziano con GPVAL_
show variables NAME # mostra solo le variabili che iniziano con NAME
```

Version

Il comando **show version** elenca la versione di gnuplot in esecuzione, la sua ultima data di modifica, i detentori del copyright e gli indirizzi email per le FAQ, la mailing list di gnuplot-info, e la segnalazione di bug–in breve, le informazioni elencate sullo schermo quando il programma viene invocato interattivamente.

Sintassi:

```
show version {long}
```

Quando viene data l'opzione **long**, elenca anche il sistema operativo, le opzioni di compilazione usate quando **gnuplot** è stato installato, la posizione del file help e (di nuovo) gli indirizzi email utili.

Vgrid

Sintassi:

```
set vgrid $gridname {size N}
unset vgrid $gridname
show vgrid
```

Se la griglia nominata esiste già, marcarla come attiva (usarla per le successive **vfill** e **voxel**). Se viene data una nuova dimensione, sostituire il contenuto esistente con una griglia N x N x N riempita a zero. Se non esiste già una griglia con questo nome, allocare una griglia N x N x N (default N=100), azzerare il contenuto, e marcarla come attiva. Si noti che i nomi delle griglie devono iniziare con '$'.

show vgrid elenca tutte le griglie voxel attualmente definite. Esempio output:

```
$vgrid1: (active)
         size 100 X 100 X 100
         vxrange [-4:4]  vyrange[-4:4]  vzrange[-4:4]
         non-zero voxel values:  min 0.061237 max 94.5604
         number of zero voxels:  992070   (99.21%)
```

unset vgrid $gridname rilascia tutte le strutture di dati associate a quella griglia voxel. Le strutture di dati vengono rilasciate anche da **reset session**. La funzione **voxel(x,y,z)** restituisce il valore del punto attivo della griglia più vicino a quella coordinata. Vedere anche **splot voxel-grids (p. 242)**.

View

Il comando **set view** imposta l'angolo di visualizzazione di grafici **splot**. Controlla come le coordinate 3D del grafico sono mappate nello spazio dello schermo 2D. Esso fornisce il controllo sulla rotazione e sul ridimensionamento dei dati plottati, ma supporta solo le proiezioni ortografiche. Supporta sia la proiezione 3D che proiezione ortogonale 2D in una mappa simile a un grafico 2D.

Sintassi:

```
set view <rot_x>{,{<rot_z>}{,{<scale>}{,<scale_z>}}}
set view map {scale <scale>}
set view projection {xy|xz|yz}
set view {no}equal {xy|xyz}
set view azimuth <angle>
show view
```

dove <rot_x> e <rot_z> controllano gli angoli di rotazione (in gradi) in un sistema di coordinate 3D virtuale allineato con lo schermo in modo che inizialmente (cioè, prima che le rotazioni siano eseguite) l'asse orizzontale dello schermo è x, l'asse verticale dello schermo è y, e l'asse perpendicolare allo schermo è z. La prima rotazione applicata è <rot_x> intorno all'asse x. La seconda rotazione applicata è <rot_z> intorno al nuovo asse z.

Il comando **set view map** è usato per rappresentare il disegno come una mappa. È utile per i grafici **contour** o le mappe di calore 2D che usano la modalità pm3d piuttosto che **with image**. In quest'ultimo caso, bisogna fare attenzione ad usare correttamente **zrange** e **cbrange**, rispettivamente, per il filtraggio dei data point di input e il ridimensionamento della gamma di colori.

<rot_x> è limitato all'intervallo [0:180] con un valore predefinito di 60 gradi, mentre <rot_z> è limitato all'intervallo [0:360] con un valore predefinito di 30 gradi. <scale> controlla il ridimensionamento dell'intero grafico **splot**, mentre <scale_z> ridimensiona solo l'asse z. Entrambe le scale sono predefinite a 1.0.

Esempi:

```
set view 60, 30, 1, 1
set view ,,0.5
```

Il primo imposta tutti e quattro i valori predefiniti. Il secondo cambia solo la scala, a 0.5.

Azimuth

```
set view azimuth <angle-in-degrees>
```

L'impostazione dell'azimut influenza l'orientamento dell'asse z in un grafico (graph) 3D (splot). All'azimut = 0 predefinito, l'asse z del grafico giace nel piano ortogonale all'orizzontale dello schermo. Cioè la proiezione dell'asse z giace lungo la verticale dello schermo. L'azimut diverso da zero ruota il grafico intorno alla linea di vista attraverso l'origine in modo che una proiezione dell'asse z non sia più verticale. Quando azimut = 90 l'asse z è orizzontale invece che verticale.

Equal_axes

Il comando **set view equal xy** forza la lunghezza unitaria degli assi x e y a essere sulla stessa scala e sceglie tale scala in modo che il grafico si adatti alla pagina. Il comando **set view equal xyz** imposta inoltre la scala dell'asse z in modo che corrisponda agli assi x e y; tuttavia non c'è garanzia che l'intervallo attuale dell'asse z rientri nei confini del grafico. Per default tutti e tre gli assi sono ridimensionati in modo indipendente per riempire l'area disponibile.

Vedere anche **set xyplane (p. 233)**.

Projection

Sintassi:

```
set view projection {xy|xz|yz}
```

Ruota gli angoli di vista di un grafico 3D in modo che uno dei piani primari xy, xz, o yz si trovi nel piano del grafico. Le etichette degli assi e il posizionamento dei tic sono regolati di conseguenza; i tic e le etichette sul terzo asse sono disabilitati. Il grafico viene scalato in modo da corrispondere approssimativamente alla dimensione che 'plot' genererebbe per gli stessi intervalli degli assi. **set view projection xy** è equivalente a **set view map**.

Vrange

Sintassi: set vrange [vmin:vmax] L'intervallo delle variabili parametriche u e v è utile in due ambiti. 1) **splot** in modalità parametrica. Vedere **set parametric (p. 195)**, **set isosamples (p. 164)**. 2) generare dati campionati bidimensionali sia per **plot** che per **splot** usando lo pseudofile "++". Vedere **sampling 2D (p. 128)**.

Vxrange

Sintassi: set vxrange [vxmin:vxmax]

Stabilisce l'intervallo di coordinate x spaziato dalla griglia voxel attiva. Esistono comandi analoghi **set vyrange** e **set vzrange** per le altre due dimensioni della griglia voxel. Se nessun intervallo esplicito è stato impostato prima di il primo comando **vclear**, **vfill**, o **voxel(x,y,z) =** , vmin e vmax saranno copiati dai valori correnti di **xrange**.

Vyrange

Vedere **set vxrange (p. 223)**

Vzrange

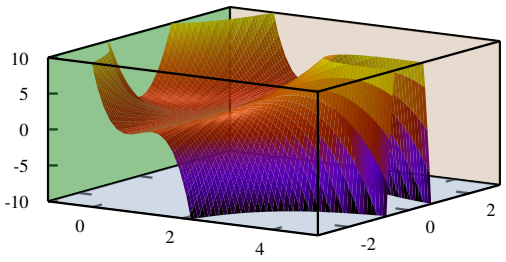
Vedere **set vxrange (p. 223)**

Walls

Sintassi:

```
set walls
set wall {x0|y0|z0|x1|y1} {<fillstyle>} {fc <fillcolor>}
```

Le superfici 3D disegnate da **splot (p. 237)** giacciono all'interno di un cubo unitario normalizzato indipendentemente dagli intervalli degli assi x, y e z. I muri di delimitazione di questo cubo sono descritti dai piani (graph coord x == 0), (graph coord x == 1), ecc. Il comando **set walls** renderizza i muri x0 y0 e z0 come superfici solide. Per default queste superfici sono semitrasparenti (fillstyle transparent solid 0.5). È possibile personalizzare quali muri vengono disegnati e anche il loro colore e lo stile di riempimento individuali. Se si sceglie di abilitare i muri, potrebbe essere utile usare **set xyplane 0**. Esempio:




```
set wall z0 fillstyle solid 1.0 fillcolor "gray"
```

X2data

Il comando **set x2data** imposta i dati sull'asse x2 (superiore) in serie temporali (date/ore). Vedere **set xdata (p. 224)**.

X2dtics

Il comando **set x2dtics** cambia i tic sull'asse x2 (superiore) nei giorni della settimana. Vedere **set xdtics (p. 225)** per i dettagli.

X2label

Il comando **set x2label** imposta l'etichetta per l'asse x2 (superiore). Vedere **set xlabel (p. 225)**.

X2mtics

Il comando **set x2mtics** cambia i tic sull'asse x2 (superiore) nei mesi dell'anno. Vedere **set xdtics (p. 225)** per i dettagli.

X2range

Il comando **set x2range** imposta l'intervallo orizzontale che sarà visualizzato sull'asse x2 (superiore). Vedere **set xrange (p. 226)** per il set completo delle opzioni di comando. Vedere anche **set link (p. 174)**.

X2tics

Il comando **set x2tics** controlla i tic principali (etichettati) sull'asse x2 (superiore). Vedere **set xtics (p. 228)** per i dettagli.

X2zeroaxis

Il comando **set x2zeroaxis** disegna una linea all'origine dell'asse x2 (superiore) (y2 = 0). Per i dettagli, vedere **set zeroaxis (p. 235)**.

Xdata

Questo comando controlla l'interpretazione dei dati sull'asse x. Un comando analogo agisce su ciascuno degli altri assi.

Sintassi:

```
set xdata time
show xdata
```

La stessa sintassi si applica a **ydata**, **zdata**, **x2data**, **y2data** e **cbdata**.

L'opzione **time** segnala che i dati rappresentano un orario/data in secondi. La versione corrente di gnuplot memorizza il tempo con una precisione al millisecondo.

Se non viene specificata alcuna opzione, l'interpretazione dei dati torna alla normalità.

Time

set xdata time indica che la coordinata x rappresenta una data o un orario con precisione al millisecondo. Esiste un comando analogo **set ydata time**.

Ci sono meccanismi di formato separati per l'interpretazione dei dati temporali in input e in output. I dati in input vengono letti da un file o utilizzando il comando globale **timefmt** o usando la funzione timecolumn()

come parte del comando plot. Questi meccanismi di input si applicano anche all'uso di valori di tempo per impostare un intervallo di assi. Vedere **set timefmt (p. 218)**, **timecolumn (p. 40)**.

Esempio:

```
set xdata time
set timefmt "%d-%b-%Y"
set xrange ["01-Jan-2013" : "31-Dec-2014"]
plot DATA using 1:2
```

o

```
plot DATA using (timecolumn(1,"%d-%b-%Y")):2
```

Per l'output, cioè le etichette dei tic lungo quell'asse o le coordinate emesse dal mouse, la funzione 'strftime' (digitare "man strftime" su unix per cercarla) è usata per convertire il tempo interno in secondi in una rappresentazione stringa di una data. **gnuplot** cerca di trovare un formato adeguato per questo. È possibile personalizzare il formato usando **set format x** or **set xtics format**. Vedere **specificatori ora (p. 159)** per un set speciale di specificatori di formato di tempo. Vedere anche **ora/data (p. 62)** per maggiori informazioni.

Xdtics

Il comando **set xdtics** converte i segni di tic dell'asse x in giorni della settimana dove 0=Dom e 6=Sab. Gli overflow sono convertiti da modulo 7 in date. **set noxdtics** riporta le etichette ai loro valori predefiniti. Comandi simili fanno lo stesso per gli altri assi.

Sintassi:

```
set xdtics
unset xdtics
show xdtics
```

La stessa sintassi si applica a **ydtics**, **zdtics**, **x2dtics**, **y2dtics** e **cbdtics**.

Vedere anche il comando **set format (p. 157)**.

Xlabel

Il comando **set xlabel** imposta l'etichetta dell'asse x. Comandi simili impostano le etichette sugli altri assi.

Sintassi:

```
set xlabel {"<label>"} {offset <offset>} {font "<font>{,<size>}"}
           {textcolor <colorspec>} {{no}enhanced}
           {rotate by <degrees> | rotate parallel | norotate}
show xlabel
```

La stessa sintassi si applica a **x2label**, **ylabel**, **y2label**, **zlabel** e **cblabel**.

Se <offset> è specificato da x,y o x,y,z, l'etichetta viene spostata dell'offset specificato. Può essere preceduto da **first**, **second**, **graph**, **screen**, o **character** per selezionare il sistema di coordinate. Vedere **coordinate (p. 33)** per i dettagli. Per default, viene usato il sistema di coordinate **character**. Per esempio, **"set xlabel offset -1,0"** cambierà solo l'offset x del titolo, spostando l'etichetta di circa un carattere a sinistra. La dimensione di un carattere dipende sia dal font che dal terminale.

<font> è usato per specificare il font in cui è scritta l'etichetta; le unità del <size> (dimensione) del font dipendono dal terminale utilizzato.

noenhanced richiede che il testo dell'etichetta non sia processato dal parser in modalità di testo avanzato anche se la modalità di testo avanzato è attualmente attiva.

Per cancellare un'etichetta, è necessario non mettere opzioni sulla linea di comando, ad es., **"set y2label"**.

Le posizioni predefinite delle etichette degli assi sono le seguenti:

xlabel: L'etichetta dell'asse x è centrata sotto la parte inferiore del grafico.

ylabel: L'etichetta dell'asse y è centrata a sinistra del grafico, con orientamento predefinito orizzontale o verticale a seconda del tipo di terminale. Il programma potrebbe non riservare abbastanza spazio a sinistra del grafico per contenere un testo lungo e non ruotato dell'etichetta y. È possibile regolarlo con **set lmargin**.

zlabel: L'etichetta dell'asse z è centrata lungo l'asse z e posta nello spazio sopra il livello della griglia.

cblabel: L'etichetta dell'asse del color box è centrata lungo il box e posta sotto o a destra secondo il gradiente orizzontale o verticale del color box.

y2label: L'etichetta dell'asse y2 è posta a destra dell'asse y2. La posizione è dipendente dal terminale allo stesso modo dell'etichetta dell'asse y.

x2label: L'etichetta dell'asse x2 è posta sopra il grafico ma sotto il titolo. È anche possibile creare un'etichetta per l'asse x2 utilizzando caratteri newline per creare un titolo di grafico multilinea, ad es.,

```
set title "This is the title\n\nThis is the x2label"
```

Si noti che devono essere usate le virgolette doppie. Lo stesso font sarà usato per entrambe le linee, naturalmente.

L'orientamento (angolo di rotazione) delle etichette degli assi x, x2, y e y2 nei grafici 2D può essere cambiato specificando **rotate by** <**degrees**>. L'orientamento delle etichette degli assi x e y nei grafici 3D è di default orizzontale, ma può essere cambiato in modo che sia parallelo all'asse specificando **rotate parallel**.

Se non si è soddisfatti della posizione predefinita di un'etichetta di un asse, si può utilizzare invece **set label** –questo comando offre molto più controllo su dove viene posizionato il testo.

Vedere **sintassi (p. 60)** per ulteriori informazioni sull'elaborazione dei backslash e la differenza tra stringhe tra virgolette singole e doppie.

Xmtics

Il comando **set xmtics** converte i segni di tic dell'asse x in mesi dell'anno dove 1=Gen e 12=Dic. Gli overflow sono convertiti da modulo 12 in mesi. I tic vengono riportati alle loro etichette predefinite da **unset xmtics**. Comandi simili eseguono gli stessi compiti per gli altri assi.

Sintassi:

```
set xmtics
unset xmtics
show xmtics
```

La stessa sintassi si applica a **x2mtics**, **ymtics**, **y2mtics**, **zmtics** e **cbmtics**.

Vedere anche il comando **set format (p. 157)**.

Xrange

Il comando **set xrange** imposta l'intervallo orizzontale che sarà visualizzato. Esiste un comando simile per ciascuno degli altri assi, così come per il raggio polare r e le variabili parametriche t, u e v.

Sintassi:

```
set xrange [{{<min>}:{<max>}}] {{no}reverse} {{no}writeback} {{no}extend}
           | restore
show xrange
```

in cui i termini <min> e <max> sono costanti, espressioni o un asterisco per impostare l'autoscaling. Se i dati sono orario/data, è necessario fornire l'intervallo come una stringa tra virgolette secondo il formato **set**

timefmt. Se <min> o <max> è omesso, il valore corrente non sarà cambiato. Vedere sotto per la sintassi completa dell'autoscaling. Vedere anche **noextend (p. 139)**.

La stessa sintassi si applica a **yrange**, **zrange**, **x2range**, **y2range**, **cbrange**, **rrange**, **trange**, **urange** e **vrange**.

Vedere **set link (p. 174)** per le opzioni che collegano gli intervalli di x e x2, o y e y2.

L'opzione **reverse** inverte la direzione di un asse autoscalato. Ad esempio, se i valori dei dati vanno da 10 a 100, sarà autoscalato all'equivalente di set xrange [100:10]. Il flag **reverse** non ha effetto se l'asse non è autoscalato. NB: Questo è un cambiamento introdotto nella versione 4.7.

Autoscaling: Se <min> (lo stesso vale anche per <max>) è un asterisco `"*"`, l'autoscaling è attivato. L'intervallo in cui viene eseguito l'autoscaling può essere limitato da un limite (bound) inferiore <lb> o un limite superiore <ub> o entrambi. La sintassi è

```
{ <lb> < } * { < <ub> }
```

Per esempio,

```
0 < * < 200
```

imposta <lb> = 0 e <ub> = 200. Con una tale impostazione <min> sarebbe autoscalato, ma il suo valore finale sarà compreso tra 0 e 200 (entrambi inclusi nonostante il segno '<'). Se non viene specificato alcun limite inferiore o superiore, anche il segno '<' è omesso. Se <ub> è inferiore a <lb> i vincoli saranno disattivati e avrà luogo l'autoscaling completo. Questa funzione è utile per plottare dati misurati con autoscaling ma fornendo un limite sull'intervallo, per eliminare i valori anomali (outlier), o per garantire un intervallo minimo che sarà visualizzato anche se i dati non avrebbero bisogno di un intervallo così grande.

L'opzione **writeback** essenzialmente salva l'intervallo trovato da **autoscale** nei buffer che verrebbero riempiti da **set xrange**. Questo è utile se si desidera plottare insieme diverse funzioni ma avere l'intervallo determinato solo da alcune di esse. L'operazione **writeback** viene eseguita durante l'esecuzione di **plot**, quindi deve essere specificata prima di quel comando. Per ripristinare, l'ultimo intervallo orizzontale salvato usa **set xrange restore**. Per esempio,

```
set xrange [-10:10]
set yrange [] writeback
plot sin(x)
set yrange restore
replot x/2
```

risulta un intervallo y di [-1:1] come trovato solo dall'intervallo di sin(x); l'intervallo [-5:5] di x/2 viene ignorato. Eseguire **show yrange** dopo ogni comando nell'esempio precedente dovrebbe aiutare a capire cosa sta succedendo.

In 2D, **xrange** e **yrange** determinano l'estensione degli assi, **trange** determina l'intervallo della variabile parametrica in modalità parametrica o l'intervallo dell'angolo in modalità polare. Allo stesso modo nel 3D parametrico, **xrange**, **yrange** e **zrange** regolano gli assi e **urange** e **vrange** regolano le variabili parametriche.

In modalità polare, **rrange** determina l'intervallo radiale plottato. <rmin> agisce come una costante additiva al raggio, mentre <rmax> agisce come una clip al raggio - non verrà plottato alcun punto con raggio maggiore di <rmax>. **xrange** e **yrange** sono interessati–gli intervalli possono essere impostati come se il grafico (graph) fosse di r(t)-rmin, con rmin aggiunto a tutte le etichette.

Qualsiasi intervallo può essere autoscalato parzialmente o totalmente, sebbene possa non avere senso autoscalare una variabile parametrica a meno che non sia plottata con dei dati.

Gli intervalli possono anche essere specificati sulla linea di comando **plot**. Un intervallo dato sulla linea plot sarà usato per quel singolo comando **plot**; un intervallo dato da un comando **set** sarà usato per tutti i grafici successivi che non specificano i propri intervalli. Lo stesso vale per **splot**.

Examples

Esempi:

Per impostare l'intervallo xrange come predefinito:

```
set xrange [-10:10]
```

Per impostare l'intervallo y in modo che aumenti verso il basso:

```
set yrange [10:-10]
```

Per cambiare zmax a 10 senza alterare zmin (che può ancora essere autoscalato):

```
set zrange [:10]
```

Per autoscalare xmin lasciando xmax invariato:

```
set xrange [*:]
```

Per autoscalare xmin ma mantenendo xmin positivo:

```
set xrange [0<*:]
```

Per autoscalare x ma mantenere un intervallo minimo da 10 a 50 (il valore effettivo potrebbe essere più grande):

```
set xrange [*<10:50<*]
```

Autoscaling ma limite xrange massimo da -1000 a 1000, cioè autoscaling entro [-1000:1000]

```
set xrange [-1000<*:*<1000]
```

Assicurarsi che xmin sia da qualche parte tra -200 e 100:

```
set xrange [-200<*<100:]
```

Extend

set xrange noextend è uguale a **set autoscale x noextend**. Vedere **noextend (p. 139)**.

Xtics

Il controllo preciso dei tic principali (etichettati) sull'asse x è possibile con il comando **set xtics**. I tic possono essere disattivati con il comando **unset xtics** e possono essere attivati (lo stato predefinito) con **set xtics**. Comandi simili controllano i tic principali sugli assi y, z, x2 e y2.

Sintassi:

```
set xtics {axis | border} {{no}mirror}
          {in | out} {scale {default | <major> {,<minor>}}}
          {{no}rotate {by <ang>}} {offset <offset> | nooffset}
          {left | right | center | autojustify}
          {add}
          {  autofreq
           | <incr>
           | <start>, <incr> {,<end>}
           | ({"<label>"} <pos> {<level>} {,{"<label>"}...) }
          {format "formatstring"} {font "name{,<size>}"} {{no}enhanced}
          { numeric | timedate | geographic }
          {{no}logscale}
          { rangelimited }
          { textcolor <colorspec> }
unset xtics
show xtics
```

La stessa sintassi si applica a **ytics**, **ztics**, **x2tics**, **y2tics** e **cbtics**.

axis o **border** dice a **gnuplot** di mettere i tic (sia i tic stessi che le etichette abbinate) lungo l'asse o il bordo, rispettivamente. Se l'asse è molto vicino al bordo, l'opzione **axis** sposterà le etichette dei tic all'esterno del bordo. In questo caso, le impostazioni dei margini rilevanti di solito vengono dimensionate male dall'algoritmo di layout automatico.

mirror dice a **gnuplot** di mettere tic non etichettati nelle stesse posizioni sul bordo opposto. **nomirror** non lo fa.

in e **out** cambiano i segni di tic da disegnare verso l'interno o verso l'esterno.

Con **scale**, può essere regolata la dimensione dei segni di tic. Se <minor> non è specificato, è 0.5*<major>. La dimensione predefinita 1.0 per i tic principali e 0.5 per i tic secondari è richiesta da **scale default**.

rotate chiede a **gnuplot** di ruotare il testo di 90 gradi, operazione che verrà eseguita se il driver del terminale in uso supporta la rotazione del testo. **norotate** annulla questa operazione. **rotate by <ang>** chiede la rotazione di <ang> gradi, supportata da alcuni tipi di terminale.

I default sono **border mirror norotate** per i tic sugli assi x e y, e **border nomirror norotate** per i tic sugli assi x2 e y2. Per l'asse z, l'opzione **{axis | border}** non è disponibile e il default è **nomirror**. Se si desidera specchiare i tic dell'asse z, potrebbe essere necessario creare un po' più di spazio per essi con **set border**.

L'<offset> è specificato da x,y o da x,y,z, e può essere preceduto da **first**, **second**, **graph**, **screen**, o **character** per selezionare il sistema di coordinate. <offset> è l'offset dei testi dei tic dalle loro posizioni predefinite, mentre il sistema di coordinate predefinito è **character**. Vedere **coordinate (p. 33)** per i dettagli. **nooffset** disattiva l'offset.

Esempio:

Spostare gli xtics più vicino al grafico.

```
set xtics offset 0,graph 0.05
```

Per default, le etichette dei tic sono allineate automaticamente a seconda dell'asse e angolo di rotazione per produrre risultati esteticamente piacevoli. Se questo non è ciò che si vuole, l'allineamento può essere sovrascritto con un'esplicita keyword **left**, **right** o **center**. **autojustify** ripristina il comportamento predefinito.

set xtics senza opzioni ripristina il bordo o l'asse predefinito se vengono visualizzati xtics; altrimenti non ha effetto. Qualsiasi frequenza o posizione {ed etichette} di tic precedentemente specificate vengono mantenute.

Le posizioni dei tic sono calcolate automaticamente di default o se viene data l'opzione **autofreq**.

Una serie di posizioni di tic può essere specificata dando o un intervallo di tic da solo, o un punto di inizio, intervallo e un punto finale (vedere **xtics series (p. 229)**).

Posizioni di tic individuali possono essere specificate individualmente fornendo una lista esplicita di posizioni, in cui ciascuna posizione può avere un'etichetta di testo associata. Vedere **xtics list (p. 230)**.

In qualsiasi modo essi vengano specificati, i tic verranno plottati solo quando si trovano nell'intervallo.

Il formato (o l'omissione) delle etichette dei tic è controllato da **set format**, a meno che il testo esplicito di un'etichetta sia incluso nella forma **set xtics ("<label>")**.

I tic secondari (non etichettati) possono essere aggiunti automaticamente tramite il comando **set mxtics**, o in posizioni esplicite con la forma **set xtics ("" <pos> 1, ...)**.

L'aspetto dei tic (stile linea, spessore linea, ecc.) è determinato dalla linea di confine (vedere **set border (p. 140)**), anche se i tic sono disegnati sugli assi.

Xtics series

Sintassi:

```
set xtics <incr>
```

```
set xtics <start>, <incr>, <end>
```

La forma implicita <start>, <incr>, <end> specifica che una serie di tic verrà plottata sull'asse tra i valori <start> e <end>, con un incremento di <incr>. Se non viene dato <end>, si presume che sia infinito. L'incremento può essere negativo. Se non viene dato né <start> né <end>, si presume che <start> sia infinito negativo, si presume che <end> sia infinito positivo, e i tic verranno disegnati ai multipli integrali di <incr>. Se l'asse è logaritmico, l'incremento sarà usato come fattore moltiplicativo.

Se si specifica un <start> o un <incr> negativo dopo un valore numerico (ad esempio, **rotate by** <**angle**> o **offset** <**offset**>), il parser fallisce perché sottrae <start> o <incr> da quel valore. Come soluzione alternativa, specificare **0-**<**start**> o **0-**<**incr**> in quel caso.

Esempio:

```
set xtics border offset 0,0.5 -5,1,5
```

Fallisce con 'invalid expression' (espressione non valida) all'ultima virgola.

```
set xtics border offset 0,0.5 0-5,1,5
```

o

```
set xtics offset 0,0.5 border -5,1,5
```

Imposta i tic sul bordo, il testo dei tic con un offset di 0,0.5 caratteri, e imposta l'inizio, l'incremento e la fine a -5, 1 e 5, come richiesto.

Le opzioni **set grid** 'front', 'back' e 'layerdefault' influenzano anche l'ordine di disegno dei xtics.

Esempi:

Fai dei tic a 0, 0.5, 1, 1.5, ..., 9.5, 10.

```
set xtics 0,.5,10
```

Fai dei tic a ..., -10, -5, 0, 5, 10, ...

```
set xtics 5
```

Fai dei tic a 1, 100, 1e4, 1e6, 1e8.

```
set logscale x; set xtics 1,100,1e8
```

Xtics list

Sintassi:

```
set xtics {add} ("label1" <pos1> <level1>, "label2" <pos2> <level2>, ...)
```

La forma esplicita ("label" <pos> <level>, ...) permette posizioni dei tic arbitrarie o etichette dei tic non numeriche. In questa forma, i tic non hanno bisogno di essere elencati in ordine numerico. Ogni tic ha una posizione, eventualmente con un'etichetta.

L'etichetta è una stringa racchiusa tra virgolette o un'espressione con valore di stringa. Può contenere informazioni di formattazione per convertire la posizione nella sua etichetta, come "%3f clients", o può essere la stringa vuota "". Vedere **set format (p. 157)** per maggiori informazioni. Se non viene data alcuna stringa, viene usata l'etichetta predefinita (numerica).

Un segno di tic esplicito ha un terzo parametro, il livello. L'impostazione predefinita è il livello 0, un tic principale. Il livello 1 genera un tic secondario. Le etichette non vengono mai stampate per i tic secondari. I tic principali e secondari possono essere auto-generati dal programma o specificati esplicitamente dall'utente. I tic con livello 2 e superiore devono essere specificati esplicitamente dall'utente e hanno la priorità sui tic auto-generati. La dimensione dei segni di tic ad ogni livello è controllata dal comando **set tics scale**.

Esempi:

```
set xtics ("low" 0, "medium" 50, "high" 100)
set xtics (1,2,4,8,16,32,64,128,256,512,1024)
set ytics ("bottom" 0, "" 10, "top" 20)
set ytics ("bottom" 0, "" 10 1, "top" 20)
```

Nel secondo esempio, tutti i tic sono etichettati. Nel terzo, solo i tic finali sono etichettati. Nel quarto, il tic non etichettato è un tic secondario.

Normalmente, se vengono dati dei tic espliciti, essi vengono usati al posto dei tic auto-generati . Al contrario, se si specifica **set xtics auto** o simili, verrà cancellato qualsiasi tic esplicito precedentemente specificato. Si possono mischiare tic espliciti e tic auto-generati usando la keyword **add**, che deve apparire prima dello stile di tic che si vuole aggiungere.

Esempio:

```
set xtics 0,.5,10
set xtics add ("Pi" 3.14159)
```

Questo genererà automaticamente segni di tic ogni 0.5 lungo x, ma aggiungerà anche un esplicito segno di tic etichettato a pi greco.

Xtics timedata

Gli orari e le date sono memorizzati internamente come un numero di secondi.

Input: I valori di ora e data non numerici sono convertiti in secondi in input usando lo specificatore di formato in **timefmt**. Anche le posizioni degli assi e i limiti di intervallo possono essere dati come date o orari tra virgolette interpretati usando **timefmt**. Se viene usata la forma <start>, <incr>, <end>, <incr> deve essere in secondi. L'uso di **timefmt** per interpretare i dati di input, l'intervallo e le posizioni dei tic è attivato da **set xdata time**.

Output: Le etichette dei tic degli assi sono generate usando un formato separato specificato o da **set format** o da **set xtics format**. Per default, sono previsti i soliti specificatori di formato numerico (**set xtics numeric**). Altre opzioni sono le coordinate geografiche (**set xtics geographic**), oppure orari o date (**set xtics time**).

Nota: Per la compatibilità con le versioni precedenti di gnuplot, il comando **set xdata time** eseguirà implicitamente anche **set xtics time**, e **set xdata** o **unset xdata** resetterà implicitamente a **set xtics numeric**. Tuttavia è possibile cambiare ciò con una successiva chiamata a **set xtics**.

Esempi:

```
set xdata time           # controlla l'interpretazione dei dati di input
set timefmt "%d/%m"      # formato usato per leggere i dati di input
set xtics timedate       # controlla l'interpretazione del formato di output
set xtics format "%b %d" # formato usato per le etichette dei tic
set xrange ["01/12":"06/12"]
set xtics "01/12", 172800, "05/12"


set xdata time
set timefmt "%d/%m"
set xtics format "%b %d" time
set xrange ["01/12":"06/12"]
set xtics ("01/12", "" "03/12", "05/12")
```

Entrambi produrranno i tic "Dic 1", "Dic 3" e "Dic 5", ma nel secondo esempio il tic a "Dic 3" non sarà etichettato.

Geographic

set xtics geographic indica che i valori dell'asse x devono essere interpretati come coordinate geografiche misurate in gradi. Usare **set xtics format** o **set format x** per specificare l'aspetto delle etichette dei tic dell'asse. Gli specificatori di formato per i dati geografici sono i seguenti:

```
%D                    = gradi interi
%<width.precision>d   = gradi in virgola mobile
%M                    = minuti interi
%<width.precision>m   = minuti in virgola mobile
%S                    = secondi interi
%<width.precision>s   = secondi in virgola mobile
%E                    = etichetta con E/W invece di +/-
%N                    = etichetta con N/S invece di +/-
```

Per esempio, il comando **set format x "%Ddeg %5.2mmin %E"** farà sì che la coordinata x -1.51 sia etichettata come **" 1deg 30.60min W"**.

Se i xtics sono lasciati nello stato predefinito (**set xtics numeric**) la coordinata sarà riportata come un numero decimale di gradi, e si presumerà che **format** contenga i normali specificatori di formato numerico piuttosto che l'insieme speciale di cui sopra.

Per produrre gradi/minuti/secondi in un contesto diverso dai tic degli assi, come ad esempio mettere etichette su una mappa, si possono usare gli specificatori di formato del tempo relativo %tH %tM %tS per strptime. Vedere **specificatori ora (p. 159)**, **strptime (p. 40)**.

Xtics logscale

Se l'attributo **logscale** è impostato per una serie tic lungo un asse a scala logaritmica, l'intervallo tic dei viene interpretato come un fattore moltiplicativo piuttosto che una costante. Per esempio:

```
# genera una serie di tic a y=20 y=200 y=2000 y=20000
set log y
set ytics 20, 10, 50000 logscale
```

Si noti che nessun tic è posto a y=50000 perché non è nella serie 2*10^x. Se la proprietà logscale è disabilitata, l'incremento di tic sarà trattato come una costante additiva anche per un asse a scala logaritmica. Per esempio:

```
# genera una serie di tic a y=20 y=40 y=60 ... y=200
set log y
set yrange [20:200]
set ytics 20 nologscale
```

L'attributo **logscale** è impostato automaticamente dal comando **set log**, quindi normalmente non si ha bisogno di questa keyword a meno che non si voglia forzare un intervallo di tic costante come nel secondo esempio qui sopra.

Xtics rangelimited

Questa opzione limita sia le etichette dei tic dell'asse auto-generate che il corrispondente bordo del grafico all'intervallo di valori effettivamente presenti nei dati che sono stati plottati. Si noti che questo è indipendente dai limiti dell'intervallo attuali per il grafico. Per esempio, supponiamo che i dati in "file.dat" siano tutti nell'intervallo $2 < y < 4$. Allora i seguenti comandi creeranno un grafico in cui il bordo sinistro (asse y) è disegnato solo per questa porzione dell'intervallo y totale, e solo i tic dell'asse in questa regione vengono generati. Cioè, il grafico sarà scalato per l'intero intervallo su y, ma ci sarà uno spazio tra 0 e 2 sul bordo sinistro e un altro spazio tra 4 e 10. Questo stile è talvolta indicato come un grafico (graph) **range-frame**.

```
set border 3
set yrange [0:10]
set ytics nomirror rangelimited
plot "file.dat"
```

Xyplane

Il comando **set xyplane** regola la posizione in cui viene disegnato il piano xy in un grafico 3D. Il sinonimo "set ticslevel" è accettato per la retrocompatibilità.

Sintassi:

```
set xyplane at <zvalue>
set xyplane relative <frac>
set ticslevel <frac>        # equivalente a impostare il piano xy relativo
show xyplane
```

La forma **set xyplane relative** <**frac**> pone il piano xy al di sotto dell'intervallo in Z, dove la distanza dal piano xy a Zmin è data come una frazione dell'intervallo totale in z. Il valore predefinito è 0.5. Sono ammessi valori negativi, ma le etichette dei tic sui tre assi possono sovrapporsi.

La forma alternativa **set xyplane at** <**zvalue**> fissa la posizione del piano xy a un valore Z specifico, indipendentemente dall'intervallo z attuale. Così per forzare gli assi x, y e z ad incontrarsi in un'origine comune si dovrebbe specificare **set xyplane at 0**.

Vedere anche **set view (p. 221)**, e **set zeroaxis (p. 235)**.

Xzeroaxis

Il comando **set xzeroaxis** disegna una linea a y = 0. Per i dettagli, vedere **set zeroaxis (p. 235)**.

Y2data

Il comando **set y2data** imposta i dati dell'asse y2 (a destra) su serie storiche (date/orari). Vedere **set xdata (p. 224)**.

Y2dtics

Il comando **set y2dtics** cambia i tic sull'asse y2 (a destra) in giorni della settimana. Vedere **set xdtics (p. 225)** per i dettagli.

Y2label

Il comando **set y2label** imposta l'etichetta per l'asse y2 (a destra). Vedere **set xlabel (p. 225)**.

Y2mtics

Il comando **set y2mtics** cambia i tic sull'asse y2 (a destra) in mesi dell'anno. Vedere **set xmtics (p. 226)** per i dettagli.

Y2range

Il comando **set y2range** imposta l'intervallo verticale che sarà visualizzato sull'asse y2 (a destra). Vedere **set xrange (p. 226)** per l'insieme completo delle opzioni di comando. Vedere anche **set link (p. 174)**.

Y2tics

Il comando **set y2tics** controlla i tic principali (etichettati) sull'asse y2 (a destra). Vedere **set xtics (p. 228)** per i dettagli.

Y2zeroaxis

Il comando **set y2zeroaxis** disegna una linea all'origine dell'asse (x2 = 0) y2 (a destra). Per i dettagli, vedere **set zeroaxis (p. 235)**.

Ydata

Il comando **set ydata** imposta i dati dell'asse y su serie storiche (date/orari). Vedere **set xdata (p. 224)**.

Ydtics

Il comando **set ydtics** cambia i tic sull'asse y in giorni della settimana. Vedere **set xdtics (p. 225)** per i dettagli.

Ylabel

Questo comando imposta l'etichetta per l'asse y. Vedere **set xlabel (p. 225)**.

Ymtics

Il comando **set ymtics** cambia i tic sull'asse y in mesi dell'anno. Vedere **set xmtics (p. 226)** per i dettagli.

Yrange

Il comando **set yrange** imposta l'intervallo verticale che sarà visualizzato sull'asse y. Vedere **set xrange (p. 226)** per i dettagli.

Ytics

Il comando **set ytics** controlla i tic principali (etichettati) sull'asse y. Vedere **set xtics (p. 228)** per i dettagli.

Yzeroaxis

Il comando **set yzeroaxis** disegna una linea a x = 0. Per i dettagli, vedere **set zeroaxis (p. 235)**.

Zdata

Il comando **set zdata** imposta i dati dell'asse z su serie storiche (date/orari). Vedere **set xdata (p. 224)**.

Zdtics

Il comando **set zdtics** cambia i tic sull'asse z in giorni della settimana. Vedere **set xdtics (p. 225)** per i dettagli.

Zzeroaxis

Il comando **set zzeroaxis** disegna una linea attraverso (x=0,y=0). Questo non ha effetto sui grafici 2D, inclusi splot (grafici 3D) con **set view map**. Per i dettagli, vedere **set zeroaxis (p. 235)** e **set xyplane (p. 233)**.

Cbdata

Imposta i dati dell'asse del color box su serie storiche (date/orari). Vedere **set xdata (p. 224)**.

Cbdtics

Il comando **set cbdtics** cambia i tic sull'asse del color box in giorni della settimana. Vedere **set xdtics (p. 225)** per i dettagli.

Zero

Il valore **zero** è la soglia predefinita per i valori che si avvicinano a 0.0.

Sintassi:

```
set zero <expression>
show zero
```

gnuplot non plotterà un punto se la sua parte immaginaria è più grande in magnitudine della soglia **zero**. Questa soglia è anche usata in varie altre parti di **gnuplot** come una (rozza) soglia di errore numerico. Il valore **zero** predefinito è 1e-8. I valori **zero** più grandi di 1e-3 (il reciproco del numero di pixel in un tipico display bitmap) dovrebbero probabilmente essere evitati, ma non è irragionevole impostare **zero** a 0.0.

Zeroaxis

L'asse x può essere disegnato da **set xzeroaxis** e rimosso da **unset xzeroaxis**. Comandi simili si comportano in modo simile per gli assi y, x2, y2 e z. **set zeroaxis ...** (senza prefisso) agisce congiuntamente sugli assi x, y e z.

Sintassi:

```
set {x|x2|y|y2|z}zeroaxis { {linestyle | ls <line_style>}
                          | {linetype | lt <line_type>}
                            {linewidth | lw <line_width>}
                            {linecolor | lc <colorspec>}
                            {dashtype | dt <dashtype>} }
unset {x|x2|y|y2|z}zeroaxis
show {x|y|z}zeroaxis
```

Per default, queste opzioni sono disattivate. L'asse zero selezionato è disegnato con una linea di tipo <line_type>, spessore <line_width>, colore <colorspec>, e tipo di trattino <dashtype> (se supportato dal driver del terminale attualmente in uso), o uno stile definito dall'utente <line_style>. (vedere **set style line (p. 210)**).

Se non viene specificato alcun tipo di linea, qualsiasi asse zero selezionato sarà disegnato usando il tipo di linea dell'asse (linetype 0).

Esempi:

Per avere semplicemente l'asse y=0 disegnato in modo visibile:

```
set xzeroaxis
```

Se invece si desidera una linea spessa in un colore o pattern diverso:

```
set xzeroaxis linetype 3 linewidth 2.5
```

Zlabel

Questo comando imposta l'etichetta per l'asse z. Vedere **set xlabel (p. 225)**.

Zmtics

Il comando **set zmtics** cambia i tic sull'asse z in mesi dell'anno. Vedere **set xmtics (p. 226)** per i dettagli.

Zrange

Il comando **set zrange** imposta l'intervallo che sarà visualizzato sull'asse z. Il zrange è usato solo da **splot** ed è ignorato da **plot**. Vedere **set xrange (p. 226)** per i dettagli.

Ztics

Il comando **set ztics** controlla i tic principali (etichettati) sull'asse z. Vedere **set xtics (p. 228)** per i dettagli.

Cblabel

Questo comando imposta l'etichetta per l'asse del color box. Vedere **set xlabel (p. 225)**.

Cbmtics

Il comando **set cbmtics** cambia i tic sull'asse del color box in mesi dell'anno. Vedere **set xmtics (p. 226)** per i dettagli.

Cbrange

Il comando **set cbrange** imposta l'intervallo di valori che vengono colorati usando la **palette** corrente con gli stili **with pm3d**, **with image** e **with palette**. I valori al di fuori della gamma di colori usano il colore dell'estremo più vicino.

Se cb-axis (asse del color box) è autoscalato in **splot**, allora l'intervallo del colorbox è preso da **zrange**. Punti disegnati in **splot ... pm3d**|**palette** possono essere filtrati usando diversi **zrange** e **cbrange**.

Vedere **set xrange (p. 226)** per i dettagli sulla sintassi **set cbrange (p. 236)**. Vedere anche **set palette (p. 190)** e **set colorbox (p. 146)**.

Cbtics

Il comando **set cbtics** controlla i tic principali (etichettati) sull'asse del color box. Vedere **set xtics (p. 228)** per i dettagli.

Shell

Il comando **shell** genera una shell interattiva. Per tornare a **gnuplot**, digitare **logout** se si usa VMS, **exit** o il carattere END-OF-FILE se si usa Unix, o **exit** se si usa MS-DOS o OS/2.

Il comando **shell** ignora qualsiasi altra cosa sulla linea di comando di gnuplot. Se invece si desidera passare una stringa di comando a una shell per l'esecuzione immediata utilizzare la funzione **system** o la scorciatoia **!**. Vedere **system (p. 244)**.

Esempi:

```
shell
system "print previous_plot.ps"
! print previous_plot.ps
current_time = system("date")
```

Splot

splot è il comando per disegnare grafici 3D (in realtà proiezioni su una superficie 2D, ma questo lo sapevate). È l'equivalente 3D del comando **plot**. **splot** fornisce solo un singolo asse x,y e z; non esiste alcun equivalente per gli assi secondari x2 e y2 forniti da **plot**.

Vedere il comando **plot (p. 107)** per molte opzioni disponibili sia nei grafici 2D che 3D.

Sintassi:

```
splot {<ranges>}
      {<iteration>}
      <function> | {{<file name> | <datablock name>} {datafile-modifiers}}
                 | <voxelgridname>
                 | keyentry
      {<title-spec>} {with <style>}
      {, {definitions{,}} <function> ...}
```

Il comando **splot** opera su dati generati da una funzione, letti da un file di dati, o memorizzati precedentemente in un data block con nome. I nomi dei file di dati di solito sono forniti come una stringa fra virgolette. La funzione può essere un'espressione matematica o una terna di espressioni matematiche in modalità parametrica.

Una nuova funzionalità nella versione 5.4 consiste nel fatto che **splot** può operare su dati voxel. Vedere **voxel-grids (p. 242)**, **set vgrid (p. 221)**, **vxrange (p. 223)**. Attualmente le griglie voxel possono essere plottate usando gli stili **with dots**, **with points**, o **with isosurface**. I valori della griglia voxel possono anche essere referenziati negli specificatori **using** di altri stili di grafico, per esempio per assegnare i colori.

Per default, **splot** disegna il piano xy completamente al di sotto dei dati plottati. L'offset tra il ztic più basso e il piano xy può essere modificato da **set xyplane**. L'orientamento di una proiezione **splot** è controllato da **set view**. Vedere **set view (p. 221)** e **set xyplane (p. 233)** per ulteriori informazioni.

La sintassi per impostare gli intervalli nel comando **splot** è la stessa del comando **plot**. In modalità non parametrica, gli intervalli devono essere dati nell'ordine

```
splot [<xrange>][<yrange>][<zrange>] ...
```

In modalità parametrica, l'ordine è

```
splot [<urange>][<vrange>][<xrange>][<yrange>][<zrange>] ...
```

L'opzione **title** è la stessa di **plot**. Anche il funzionamento di **with** è lo stesso di **plot** eccetto che non sono disponibili tutti gli stili di plotting 2D.

Le opzioni **datafile** hanno più differenze.

Come alternativa alle superfici disegnate usando la modalità parametrica o di funzione, lo pseudo-file '++' può essere usato per generare campioni su una griglia nel piano xy.

Vedere anche **show plot (p. 197)**, **set view map (p. 221)**, e **sampling (p. 127)**.

Data-file

Splot, come **plot**, può visualizzare da un file.

Sintassi:

```
splot '<file_name>' {binary <binary list>}
                    {{nonuniform} matrix}
                    {index <index list>}
                    {every <every list>}
                    {using <using list>}
```

I filename speciali "" e "-" sono permessi, come in **plot**. Vedere **special-filenames (p. 120)**.

In breve, **binary** e **matrix** indicano che i dati sono in una forma speciale, **index** seleziona quali set di dati in un file con più set di dati devono essere plottati, **every** specifica quali linee di dati (sottoinsiemi) all'interno di un singolo set di dati devono essere plottate, e **using** determina come le colonne all'interno di un singolo record devono essere interpretate.

Le opzioni **index** e **every** si comportano allo stesso modo che con **plot**; **using** si comporta allo stesso modo, eccetto che la lista **using** deve fornire tre voci invece di due.

L'opzione di **plot smooth** non è disponibile per **splot**, ma **cntrparam** e **dgrid3d** forniscono limitate capacità di smoothing.

L'organizzazione del file di dati è essenzialmente la stessa di **plot**, salvo che ogni punto è una tripla (x,y,z). Se viene fornito un solo valore, esso sarà usato per z, il numero del blocco sarà usato per y, e l'indice del data point nel blocco sarà usato per x. Se vengono forniti due o quattro valori, **gnuplot** usa l'ultimo valore per calcolare il colore in grafici pm3d. Tre valori sono interpretati come una tripla (x,y,z). Valori aggiuntivi sono generalmente usati come errori, che possono essere usati da **fit**.

Singoli record vuoti separano blocchi di dati in un file di dati **splot**; **splot** tratta i blocchi come l'equivalente della funzione y-isolinee. Nessuna linea unirà punti separati da un record vuoto. Se tutti i blocchi contengono lo stesso numero di punti, **gnuplot** disegnerà isoline incrociate (cross-isolines) tra i punti nei blocchi, connettendo i punti corrispondenti. Questo è chiamato "grid data", ed è necessario per disegnare una superficie, per impostare il contorno (**set contour**) e la rimozione delle linee nascoste (**set hidden3d**). Vedere anche **splot grid_data (p. 241)**.

Non è più necessario specificare modalità **parametric** per i grafici **splot** a tre colonne.

Matrix

Gnuplot può interpretare l'input di dati delle matrici in due modi diversi.

Il primo modo presuppone una griglia uniforme di coordinate x e y e assegna ogni valore nella matrice di input a un elemento M[i,j] di questa griglia uniforme. Le coordinate x assegnate sono gli interi [0:NCOLS-1]. Le coordinate y assegnate sono gli interi [0:NROWS-1]. Questo è il default per l'input di dati testuali, ma non per l'input binario. Vedere **matrix uniform (p. 238)** per degli esempi e ulteriori keyword.

La seconda interpretazione presuppone una griglia non uniforme con coordinate x e y esplicite. La prima riga di dati di input contiene le coordinate y; la prima colonna di dati di input contiene le coordinate x. Per i dati di input binari il primo elemento della prima riga deve contenere il numero di colonne. Questo è il default per l'input **binary matrix**, ma richiede un'ulteriore keyword **nonuniform** per i dati di input testuali. Vedere **matrix nonuniform (p. 239)** per degli esempi.

Uniform Comandi di esempio per plottare dati di matrici uniformi:

```
splot 'file' matrix using 1:2:3          # text input
splot 'file' binary general using 1:2:3  # binary input
```

In una matrice a griglia uniforme, i valori z vengono letti una riga alla volta, cioè,

```
z11 z12 z13 z14 ...
z21 z22 z23 z24 ...
z31 z32 z33 z34 ...
```

e così via.

Per l'input di testo, se la prima riga contiene etichette di colonne piuttosto che dati, usare la keyword aggiuntiva **columnheaders**. Allo stesso modo, se il primo campo in ciascuna riga contiene un'etichetta piuttosto che dati, usare la keyword aggiuntiva **rowheaders**. Ecco un esempio che usa entrambi:

```
$DATA << EOD
xxx A   B   C   D
aa  z11 z12 z13 z14
bb  z21 z22 z23 z24
cc  z31 z32 z33 z34
EOD
plot $DATA matrix columnheaders rowheaders with image
```

Per l'input di testo, una linea vuota o una linea di commento termina la matrice e inizia un nuovo data block. È possibile selezionare tra i data block in un file tramite l'opzione **index** al comando **splot**, come di consueto. L'opzione columnheaders, se presente, viene applicata solo al primo data block.

Nonuniform La prima riga dei dati di input contiene le coordinate y. La prima colonna dei dati di input contiene le coordinate x. Per i dati di input binari, il primo campo della prima riga deve contenere il numero di colonne. (Questo numero è ignorato per l'input di testo).

Comandi di esempio per plottare dati di matrici non uniformi:

```
splot 'file' nonuniform matrix using 1:2:3  # text input
splot 'file' binary matrix using 1:2:3      # binary input
```

Quindi l'organizzazione dei dati per l'input di matrici non uniformi è

```
<N+1>  <x0>   <x1>   <x2>  ...  <xN>
 <y0> <z0,0> <z0,1> <z0,2> ... <z0,N>
 <y1> <z1,0> <z1,1> <z1,2> ... <z1,N>
  :      :      :      :   ...    :
```

che viene poi convertito in triplette:

```
<x0> <y0> <z0,0>
<x0> <y1> <z0,1>
<x0> <y2> <z0,2>
 :    :     :
<x0> <yN> <z0,N>

<x1> <y0> <z1,0>
<x1> <y1> <z1,1>
 :    :     :
```

Queste triplette sono poi convertite in iso-curve di **gnuplot** e poi **gnuplot** procede nel solito modo per eseguire il resto del plotting.

Every La keyword **every** ha un significato speciale quando viene usata con i dati delle matrici. Piuttosto che essere applicata a blocchi di singoli punti, si applica a valori di righe e di colonne. Si noti che le righe e le colonne della matrice sono indicizzate a partire da 0, quindi la riga con indice N è la (N+1)-esima riga. Sintassi:

```
plot 'file' every {<column_incr>}
                  {:{<row_incr>}
                    {:{<start_column>}
                      {:{<start_row>}
                        {:{<end_column>}
                          {:<end_row>}}}}}
```

Esempi:

```
plot 'file' matrix every :::N::N   # plotta tutti i valori nella riga con indice N
plot 'file' matrix every ::3::7    # plotta le colonne da 3 a 7 per tutte le righe
plot 'file' matrix every ::3:0:7:4 # sottomatrice delimitata da [3,0] e [7,4]
```

Esempi In **binary.c** viene fornita una raccolta di routine (in C) di manipolazione di matrici e vettori. La routine per scrivere dati binari è

```
int fwrite_matrix(file,m,nrl,nrl,ncl,nch,row_title,column_title)
```

Un esempio di utilizzo di queste routine è fornito nel file **bf_test.c**, che genera file binari per il file demo **demo/binary.dem**.

Utilizzo in **plot**:

```
plot ‘a.dat‘ matrix
plot ‘a.dat‘ matrix using 1:3
plot 'a.gpbin' {matrix} binary using 1:3
```

plotterà le righe della matrice, mentre using 2:3 plotterà le colonne della matrice, e using 1:2 le coordinate dei punti (piuttosto inutile). Applicando l'opzione **every** è possibile specificare righe e colonne esplicite.

Esempio – ridimensiona gli assi di una matrice in un file di testo:

```
splot ‘a.dat‘ matrix using (1+$1):(1+$2*10):3
```

Esempio – plotta la terza riga di una matrice in un file di testo:

```
plot 'a.dat' matrix using 1:3 every 1:999:1:2
```

(le righe sono enumerate a partire da 0, quindi 2 invece di 3).

Gnuplot può leggere file binari di matrice utilizzando l'opzione **binary** che appare senza qualifiche di keyword uniche per general binary, cioè **array**, **record**, **format**, o **filetype**. Altre keyword general binary per la traduzione dovrebbero applicarsi anche a matrix binary. (Vedere **binary general (p. 109)** per maggiori dettagli.)

Example datafile

Un semplice esempio di plotting di un file di dati 3D è

```
splot 'datafile.dat'
```

in cui il file "datafile.dat" potrebbe contenere:

```
# The valley of the Gnu.
   0 0 10
   0 1 10
   0 2 10
```

```
1 0 10
1 1 5
1 2 10

2 0 10
2 1 1
2 2 10

3 0 10
3 1 0
3 2 10
```

Si noti che "datafile.dat" definisce una griglia 4 per 3 (4 righe di 3 punti ciascuna). Le righe (blocchi) sono separate da record vuoti.

Si noti anche che il valore x è mantenuto costante all'interno di ogni linea di dati. Se invece si mantiene y costante, e si plotta con la rimozione delle linee nascoste abilitata, si troverà che la superficie è disegnata 'inside-out' (al rovescio).

In realtà, per i dati della griglia non è necessario mantenere i valori x costanti all'interno di un blocco, né è necessario mantenere la stessa sequenza di valori y. **gnuplot** richiede solo che il numero di punti sia lo stesso per ogni blocco. Tuttavia, poiché la mesh della superficie, da cui sono derivati i contorni, collega i punti corrispondenti in modo sequenziale, l'effetto di una griglia irregolare su un grafico a superficie è imprevedibile e dovrebbe essere esaminato caso per caso.

Grid data

Le routine 3D sono progettate per punti in un formato griglia, con un campione, datapoint, ad ogni intersezione della mesh; i datapoint possono provenire sia dalla valutazione di una funzione, vedere **set isosamples (p. 164)**, sia dalla lettura di un datafile, vedere **splot datafile (p. 238)**. Il termine "isoline" è applicato alle linee di mesh sia per le funzioni che per i dati. Si noti che la mesh non deve necessariamente essere rettangolare in x e y, poiché può essere parametrizzata in u e v, vedere **set isosamples (p. 164)**.

Tuttavia, **gnuplot** non necessita quel formato. Nel caso di funzioni, i 'samples' (campioni) non devono necessariamente essere uguali a 'isosamples', cioè, non è indispensabile che ogni campione di x-isolina intersechi una y-isolina. Nel caso di file di dati, se sono presenti un numero uguale di data point sparsi in ogni blocco, allora le "isolines" collegheranno i punti in un blocco, e le "cross-isolines" collegheranno i punti corrispondenti in ogni blocco per generare una "surface". In entrambi i casi, le modalità contour e hidden3d possono dare grafici diversi da quelli che si otterrebbero se i punti fossero nel formato previsto. I dati sparsi possono essere convertiti in un formato di griglia {diverso} con **set dgrid3d**.

Il codice del contorno verifica l'intensità z lungo una linea tra un punto su una y-isolinea e il punto corrispondente nella y-isolinea successiva. Così un contorno **splot** di una superficie con campioni sulle x-isolinee che non coincidono con un'intersezione dell'y-isolinea ignorerà tali campioni. Provare:

```
set xrange [-pi/2:pi/2]; set yrange [-pi/2:pi/2]
set style function lp
set contour
set isosamples 10,10; set samples 10,10;
splot cos(x)*cos(y)
set samples 4,10; replot
set samples 10,4; replot
```

Splot surfaces

splot può visualizzare una superficie come un insieme di punti, o collegando questi punti. Come con **plot**, i punti possono essere letti da un file di dati o risultare dalla valutazione di una funzione a intervalli specificati,

vedere **set isosamples (p. 164)**. La superficie può essere approssimata collegando i punti con segmenti di linea retta, vedere **set surface (p. 214)**, nel qual caso la superficie può essere resa opaca con **set hidden3d (p. 162)**. L'orientamento da cui viene vista la superficie 3d può essere cambiato con **set view**.

Inoltre, per i punti in un formato griglia, **splot** può interpolare punti che hanno un'ampiezza comune (vedere **set contour (p. 147)**) e può quindi collegare questi nuovi punti per visualizzare le linee di contorno, direttamente con segmenti di linea retta o linee smussate (vedere **set cntrparam (p. 144)**). Le funzioni sono già valutate in un formato griglia, determinato da **set isosamples** e **set samples**, mentre i dati di file devono essere o in un formato griglia, come descritto in **data-file**, o essere usati per generare una griglia (vedere **set dgrid3d (p. 152)**).

Le linee di contorno possono essere visualizzate sia sulla superficie che proiettate sulla base. Le proiezioni di base delle linee di contorno possono essere scritte in un file, e poi lette con **plot**, per trarre vantaggio dalle capacità di formattazione aggiuntive di **plot**.

Voxel-grid

Sintassi:

```
splot $voxelgridname with {dots|points} {above <threshold>} ...
splot $voxelgridname with isosurface {level <threshold>} ...
```

I dati dei voxel possono essere plottati con puntini (dots) o punti (points) che contrassegnano i singoli voxel il cui valore è superiore al valore di soglia specificato (soglia predefinita = 0). Le proprietà di colore/pointtype/linewidth possono essere aggiunte come al solito.

In molti angoli di visuale i punti della griglia voxel si occluderanno a vicenda o creeranno effetti Moiré sul display. Questi effetti possono essere evitati introducendo il jitter in modo che il puntino o punto visualizzato sia spostato in maniera casuale dalla coordinate reali della griglia voxel. Vedere **set jitter (p. 165)**.

Le griglie voxel dense possono essere sottocampionate usando la proprietà **pointinterval** (abbreviata **pi**) per ridurre il numero di punti disegnati.

```
splot $vgrid with points pointtype 6 pointinterval 2
```

with isosurface creerà una superficie tassellata in 3D che racchiude tutti i voxel con valore superiore alla soglia richiesta. Il posizionamento della superficie è regolato mediante interpolazione lineare per passare attraverso il valore di soglia stesso.

Vedere **set vgrid (p. 221)**, **vfill (p. 247)**. Vedere le demo **vplot.dem**, **isosurface.dem**.

Stats (Statistical Summary)

Sintassi:

```
stats {<ranges>} 'filename' {matrix | using N{:M}} {name 'prefix'} {{no}output}
```

Questo comando prepara un riassunto statistico dei dati in una o due colonne di un file. Lo specificatore di utilizzo viene interpretato nello stesso modo dei comandi plot. Vedere **plot (p. 107)** per dettagli sulle direttive **index (p. 116)**, **every (p. 115)**, **using (p. 121)**. I data point vengono filtrati sia da xrange che da yrange prima dell'analisi. Vedere **set xrange (p. 226)**. Il riepilogo viene stampato sullo schermo di default. L'output può essere reindirizzato a un file mediante l'uso precedente del comando **set print**, o soppresso del tutto usando l'opzione **nooutput**.

Oltre all'output stampato, il programma memorizza le statistiche individuali in tre set di variabili. Il primo set di variabili riporta come sono disposti i dati nel file. L'array di intestazioni di colonna viene generato solo se l'opzione **set datafile columnheaders** è attiva.

STATS_records	N	numero totale N di record di dati in-range
STATS_outofrange		numero di record filtrati dai limiti di intervallo
STATS_invalid		numero di record non validi/incompleti/mancanti
STATS_blank		numero di linee vuote nel file
STATS_blocks		numero di blocchi di dati indicizzabili nel file
STATS_columns		numero di colonne di dati nella prima riga di dati
STATS_column_header		array di stringhe che contengono le intestazioni di colonna trovate

Il secondo set riporta le proprietà dei dati in-range di una singola colonna. Questa colonna è trattata come y. Se l'asse y è autoscalato, allora non vengono applicati limiti di intervallo. Altrimenti vengono considerati solo i valori nell'intervallo [ymin:ymax].

Se due colonne sono analizzate congiuntamente da un singolo comando **stats**, il suffisso "_x" o "_y" viene aggiunto ad ogni nome di variabile. Cioè STATS_min_x è il valore minimo trovato nella prima colonna, mentre STATS_min_y è il valore minimo trovato nella seconda colonna. In questo caso i punti vengono filtrati testando sia xrange che yrange.

STATS_min		$\min(y)$	valore minimo di data point in-range
STATS_max		$\max(y)$	valore massimo di data point in-range
STATS_index_min		$i \mid y_i = \min(y)$	indice i per cui data[i] == STATS_min
STATS_index_max		$i \mid y_i = \max(y)$	indice i per cui data[i] == STATS_max
STATS_mean	$\bar{y} =$	$\frac{1}{N}\sum y$	valore medio dei data point in-range
STATS_stddev	$\sigma_y =$	$\sqrt{\frac{1}{N}\sum (y-\bar{y})^2}$	deviazione standard della popolazione dei dati in-range
STATS_ssd	$s_y =$	$\sqrt{\frac{1}{N-1}\sum (y-\bar{y})^2}$	deviazione standard del campione dei dati in-range
STATS_lo_quartile			valore del limite del quartile inferiore (1°)
STATS_median			valore medio
STATS_up_quartile			valore del limite del quartile superiore (3°)
STATS_sum		$\sum y$	somma
STATS_sumsq		$\sum y^2$	somma dei quadrati
STATS_skewness		$\frac{1}{N\sigma^3}\sum (y-\bar{y})^3$	asimmetria dei data point in-range
STATS_kurtosis		$\frac{1}{N\sigma^4}\sum (y-\bar{y})^4$	curtosi dei data point in-range
STATS_adev		$\frac{1}{N}\sum \lvert y-\bar{y}\rvert$	deviazione mediana assoluta dei data point in-range
STATS_mean_err		$\sigma_y/\sqrt{N}$	errore standard del valore medio
STATS_stddev_err		$\sigma_y/\sqrt{2N}$	errore standard della deviazione standard
STATS_skewness_err		$\sqrt{6/N}$	errore standard dell'asimmetria
STATS_kurtosis_err		$\sqrt{24/N}$	errore standard della curtosi

La terza serie di variabili è rilevante solo per l'analisi di due colonne di dati.

STATS_correlation	coefficiente di correlazione campionario tra i valori x e y
STATS_slope	A corrispondente a un adattamento lineare y = Ax + B
STATS_slope_err	incertezza di A
STATS_intercept	B corrispondente a un adattamento lineare y = Ax + B
STATS_intercept_err	incertezza di B
STATS_sumxy	somma di x*y
STATS_pos_min_y	coordinata x di un punto con valore y minimo
STATS_pos_max_y	coordinata x di un punto con valore y massimo

La keyword **matrix** indica che l'input consiste di una matrice (vedere **matrix (p. 238)**); le normali statistiche sono generate considerando tutti gli elementi della matrice. Le dimensioni della matrice sono salvate nelle variabili STATS_size_x e STATS_size_y.

STATS_size_x	numero di colonne della matrice
STATS_size_y	numero di righe della matrice

L'indice riportato in STATS_index_xxx corrisponde al valore della pseudo-colonna 0 ($0) nei comandi plot. Cioè il primo punto ha indice 0, l'ultimo punto ha indice N-1.

I valori dei dati vengono ordinati per trovare i limiti della mediana e del quartile. Se il numero totale di punti N è dispari, il valore della mediana viene preso come il valore del data point (N+1)/2. Se N è pari, la mediana è riportata come il valore medio dei punti N/2 e (N+2)/2. Per i confini del quartile viene utilizzato un trattamento equivalente.

Per un esempio di utilizzo del comando **stats** per annotare un grafico successivo, vedere `stats.dem`.

Il comando **stats** in questa versione di gnuplot può gestire dati in scala logaritmica, ma non il contenuto dei campi ora/data (**set xdata time** o **set ydata time**). Questa restrizione potrebbe essere allentata in una versione futura.

Name

Può essere utile tracciare le statistiche da più di un file o colonna di dati in parallelo. L'opzione **name** fa sì che il prefisso "STATS" predefinito sia sostituito da una stringa specificata dall'utente. Per esempio, il valore medio dei dati della colonna 2 di due file diversi potrebbe essere confrontato da

```
stats "file1.dat" using 2 name "A"
stats "file2.dat" using 2 name "B"
if (A_mean < B_mean) {...}
```

Invece di fornire una costante stringa come nome, la keyword **columnheader** o la funzione **columnheader(N)** può essere usata per generare il nome da qualsiasi stringa si trovi in quella colonna nella prima riga del file di dati:

```
do for [COL=5:8] { stats 'datafile' using COL name columnheader }
```

System

Sintassi:

```
system "command string"
! command string
output = system("command string")
show variable GPVAL_SYSTEM
```

system "command" esegue "command" in un sottoprocesso invocando la shell predefinita del sistema operativo. Se chiamato come funzione, **system("command")** restituisce il flusso di caratteri dallo stdout del sottoprocesso come una stringa. Un newline finale viene rimosso dalla stringa risultante, se presente. Vedere anche **backquotes (p. 59)**.

Lo stato di uscita del sottoprocesso è riportato nelle variabili GPVAL_SYSTEM_ERRNO e GPVAL_SYSTEM_ERRMSG. Si noti che se la stringa di comando invoca più di un programma, il sottoprocesso può restituire "Success" anche se uno dei programmi ha prodotto un errore. Ad es., file = system("ls -1 *.plt | tail -1") restituirà "Success" anche se non ci sono file *.plt perché **tail** ha successo anche se **ls** non riesce.

Il comando di sistema può essere usato per importare funzioni esterne in gnuplot come mostrato sotto, tuttavia questo forzerà la creazione di un sottoprocesso diverso ogni volta che la funzione viene invocata. Per le funzioni che saranno invocate molte volte sarebbe meglio importare una subroutine direttamente richiamabile da una libreria condivisa. Vedere **import (p. 105)** e **plugin.dem**.

```
f(x) = real(system(sprintf("somecommand %f", x)))
```

Test

Questo comando verifica o presenta graficamente le capacità del terminale e della palette.

Sintassi:

```
test {terminal | palette}
```

test o **test terminal** crea una visualizzazione degli stili di linea e di punto e altre cose utili supportate dal **terminal**e attualmente in uso.

test palette plotta i profili di R(z),G(z),B(z), dove 0<=z<=1. Queste sono le componenti RGB della **palette** di colori corrente. Plotta anche l'intensità netta apparente come calcolata utilizzando i coefficienti NTSC per mappare RGB su una scala di grigi. I valori dei profili sono anche caricati in un datablock chiamato $PALETTE.

Toggle

Sintassi:

```
toggle {<plotno> | "plottitle" | all}
```

Questo comando ha lo stesso effetto di un clic con il tasto sinistro del mouse sulla key entry per un grafico attualmente visualizzato da un terminale interattivo (qt, wxt, x11). Se il grafico è visibile, viene disattivato; se è attualmente nascosto, viene attivato. **toggle all** agisce su tutti i grafici attivi, equivalente al tasto di scelta rapida `"i"`. **toggle "title"** richiede una corrispondenza esatta con il titolo del grafico. **toggle "ti*"** agisce sul primo grafico il cui titolo corrisponde ai caratteri prima del '*' finale. Se il terminale corrente non è interattivo, il comando toggle non ha effetto.

Undefine

Cancella una o più variabili utente definite in precedenza. Questo è utile per resettare lo stato di uno script contenente un test di inizializzazione.

Il nome di una variabile può contenere il carattere jolly ***** come ultimo carattere. Se viene trovato il carattere jolly, tutte le variabili con nomi che iniziano con il prefisso che precede il carattere jolly verranno rimosse. Questo è utile per rimuovere diverse variabili che condividono un prefisso comune. Notare che il carattere jolly è consentito solo alla fine del nome della variabile! Specificare il carattere jolly come unico argomento di **undefine** non ha alcun effetto.

Esempio:

```
undefine foo foo1 foo2
if (!exists("foo")) load "initialize.gp"

bar = 1; bar1 = 2; bar2 = 3
undefine bar*                # rimuove tutte e tre le variabili
```

Unset

Le opzioni impostate usando il comando **set** possono essere riportate al loro stato predefinito dal corrispondente comando **unset**. Il comando **unset** può contenere una clausola di iterazione opzionale. Vedere **plot for (p. 128)**.

Esempi:

```
set xtics mirror rotate by -45 0,10,100
...
unset xtics

# Disattiva le etichette numerate da 100 a 200
unset for [i=100:200] label i
```

Linetype

Sintassi:

```
unset linetype N
```

Rimuove tutte le caratteristiche precedentemente associate ad un singolo tipo di linea. L'utilizzo successivo di questo tipo di linea userà qualsiasi caratteristica e colore che è nativo del tipo di terminale corrente (cioè le proprietà predefinite dei tipi di linea disponibili nelle versioni di gnuplot precedenti alla 4.6).

Monochrome

Cambia il set attivo dei tipi di linea da monocromatico a colore. Equivalente a **set color**.

Output

Poiché alcuni tipi di terminale permettono di scrivere più grafici in un singolo file di output, il file di output non viene chiuso automaticamente dopo il plotting. Per poter stampare o usare il file in modo sicuro, esso dovrebbe prima essere chiuso esplicitamente usando **unset output** o usando **set output** per chiudere il file precedente e poi aprirne uno nuovo.

Terminal

Il terminale predefinito che è attivo al momento dell'ingresso del programma dipende dalla piattaforma di sistema, dalle opzioni di compilazione di gnuplot e dalla variabile d'ambiente GNUTERM. Qualunque sia questo default, gnuplot lo salva nella variabile interna GNUTERM. Il comando **unset terminal** ripristina il tipo di terminale iniziale. È equivalente a **set terminal GNUTERM**. Tuttavia, se la stringa in GNUTERM contiene opzioni del terminale oltre al semplice nome del terminale, si potrebbe voler usare invece **set terminal @GNUTERM**.

Update

Nota: Questo comando è DEPRECATO. Usare **save fit** al suo posto.

Vclear

Sintassi:

```
vclear {$gridname}
```

Azzera il valore di tutti i voxel in una griglia esistente. Se non viene dato il nome di una griglia, cancella la griglia attualmente attiva.

Vfill

Sintassi:

```
vfill FILE using x:y:z:radius:(<expression>)
```

Il comando **vfill** agisce in modo analogo a un comando **plot**, eccetto che invece di creare un grafico, modifica i voxel nella griglia voxel attualmente attiva. Per ogni punto letto dal file di input, il voxel contenente quel punto e anche tutti gli altri voxel all'interno di una sfera di un dato raggio centrata su (x,y,z), sono incrementati come segue:

- la variabile utente VoxelDistance è impostata sulla distanza da (x,y,z) alle coordinate della griglia di quel voxel (vx,vy,vz).
- Viene valutata l'espressione fornita nel 5° specificatore **using**. Questa espressione può utilizzare il nuovo valore di VoxelDistance.
- voxel(vx,vy,vz) += risultato della valutazione di <expression>

Esempio:

```
vfill "file.dat" using 1:2:3:(3.0):(1.0)
```

Questo comando aggiunge 1 al valore di ogni voxel entro una sfera di raggio 3.0 intorno ad ogni punto di file.dat.

Esempio:

```
vfill "file.dat" using 1:2:3:4:(VoxelDistance < 1 ? 1 : 1/VoxelDistance)
```

Questo comando modifica tutti i voxel in una sfera il cui raggio è determinato per ogni punto dal contenuto della colonna 4. L'incremento aggiunto a un voxel diminuisce con la sua distanza dal data point.

Si noti che **vfill** incrementa sempre i valori esistenti nella griglia voxel corrente. Per azzerarli, usare **vclear**.

While

Sintassi:

```
while (<expr>) {
    <commands>
}
```

Esegue un blocco di comandi ripetutamente fintanto che <expr> valuta a un valore diverso da zero. Questo comando non può essere mischiato con dichiarazioni vecchio stile (senza parentesi) if/else. Vedere anche **do** **(p. 94)**, **continue** **(p. 93)**, **break** **(p. 91)**.

Part IV
Tipi di terminale

Lista completa di terminali

Gnuplot supporta un gran numero di formati di output. Questi vengono selezionati scegliendo un tipo di terminale appropriato, eventualmente con opzioni di modifica aggiuntive. Vedere **set terminal (p. 215)**.

Questo documento potrebbe descrivere tipi di terminale che non sono a vostra disposizione perché non sono stati configurati o installati sul vostro sistema. Per vedere un elenco di terminali disponibili su una particolare installazione di gnuplot, digitare 'set terminal' senza modificatori.

I terminali segnati come **legacy** non sono incorporati di default nelle recenti versioni di gnuplot e potrebbero non funzionare.

Aifm

NOTA: Legacy terminal, originariamente scritto per Adobe Illustrator 3.0+. Poiché Adobe Illustrator capisce direttamente i comandi PostScript di livello 1, si dovrebbe usare invece **set terminal post level1**.

Sintassi:

```
set terminal aifm {color|monochrome} {"<fontname>"} {<fontsize>}
```

Aqua

Questo terminale ricorre ad AquaTerm.app per la visualizzazione su Mac OS X.

Sintassi:

```
set terminal aqua {<n>} {title "<wintitle>"} {size <x> <y>}
                  {font "<fontname>{,<fontsize>}"}
                  {linewidth <lw>}"}
                  {{no}enhanced} {solid|dashed} {dl <dashlength>}}
```

dove <n> è il numero della finestra in cui disegnare (il default è 0), <wintitle> è il nome mostrato nella barra del titolo (default "Figure <n>"), <x> <y> è la dimensione del grafico (il default è 846x594 pt = 11.75x8.25 in).

Usare <fontname> per specificare il font (il default è "Times-Roman"), e <fontsize> per specificare la dimensione del font (il default è 14.0 pt).

Il terminale aqua supporta la modalità di testo avanzato (vedere **enhanced (p. 35)**), eccetto per la sovrastampa. Il supporto dei font è limitato ai font disponibili sul sistema. La codifica dei caratteri può essere selezionata da **set encoding** e attualmente supporta iso_latin_1, iso_latin_2, cp1250, e UTF8 (default).

Le linee possono essere disegnate sia continue che tratteggiate, (il default è continua) e la spaziatura dei trattini può essere modificata da <dashlength> che è un moltiplicatore > 0.

Be

Il tipo di terminale **be** è presente se gnuplot è costruito per il sistema operativo **beos** e per l'uso con server X. Viene selezionato all'avvio del programma se è impostata la variabile d'ambiente **DISPLAY**, se la variabile d'ambiente **TERM** è impostata su **xterm**, o se viene usata l'opzione di linea di comando **-display**.

Sintassi:

```
set terminal be {reset} {<n>}
```

Sono supportate più finestre di grafici: **set terminal be <n>** dirige l'output alla finestra di grafico numero n. Se n>0, il numero del terminale sarà aggiunto al titolo della finestra e l'icona ora sarà etichettata **gplt <n>**. La finestra attiva può essere identificata tramite un cambiamento del cursore (dal default a un mirino.)

Le finestre dei grafici rimangono aperte anche quando il driver **gnuplot** viene cambiato su un dispositivo diverso. Una finestra di grafico può essere chiusa premendo la lettera q mentre quella finestra ha il focus dell'input, o scegliendo **close** dal menu di un window manager. Tutte le finestre dei grafici possono essere chiuse specificando **reset**, che termina effettivamente il sottoprocesso che mantiene le finestre (a meno che non sia specificato **-persist**).

Le finestre dei grafici saranno chiuse automaticamente alla fine della sessione a meno che non sia stata data l'opzione **-persist**.

La dimensione e il rapporto d'aspetto di un grafico possono essere cambiati ridimensionando la finestra di **gnuplot**.

Lo spessore delle linee e le dimensioni dei punti possono essere cambiati dall'interno di **gnuplot** con **set linestyle**.

Per il tipo di terminale **be**, **gnuplot** accetta (quando inizializzate) le opzioni e le risorse standard di X Toolkit come geometria, font, e il nome dagli argomenti della linea di comando o da un file di configurazione. Vedere la pagina man di X(1) (o il suo equivalente) per una descrizione di tali opzioni.

Sono disponibili diverse altre opzioni di **gnuplot** per il terminale **be**. Queste possono essere specificate sia come opzioni della linea di comando quando è invocato **gnuplot** o come risorse nel file di configurazione "Xdefaults". Sono impostate al momento dell'inizializzazione e non possono essere modificate durante una sessione di **gnuplot**.

Command-line_options

Oltre alle opzioni di X Toolkit, le seguenti opzioni possono essere specificate sulla linea di comando quando si avvia **gnuplot** o come risorse nel tuo file ".Xdefaults":

'-mono'	forza il rendering monocromatico sui display a colori.
'-gray'	richiede il rendering in scala di grigi su display in scala di grigi o a colori. (I display in scala di grigi ricevono, per default, un rendering monocromatico.)
'-clear'	richiede che la finestra venga momentaneamente cancellata prima che venga visualizzato un nuovo grafico.
'-raise'	alza la finestra del grafico dopo ogni grafico.
'-noraise'	non alza la finestra del grafico dopo ogni grafico.
'-persist'	le finestre dei grafici sopravvivono dopo l'uscita del programma gnuplot principale.

Le opzioni sono mostrate qui sopra nella loro sintassi della linea di comando. Quando sono inserite come risorse in ".Xdefaults", richiedono una sintassi diversa.

Esempio:

```
gnuplot*gray: on
```

gnuplot fornisce anche un'opzione della linea di comando (**-pointsize <v>**) e una risorsa, **gnuplot*pointsize: <v>**, per controllare la dimensione dei punti plottati con lo stile di plotting **points**. Il valore **v** è un numero reale (maggiore di 0 e minore o uguale a dieci) usato come fattore di scala per le dimensioni dei punti. Per esempio, **-pointsize 2** usa punti grandi il doppio della dimensione predefinita, e **-pointsize 0.5** usa punti grandi la metà della dimensione normale.

Monochrome_options

Per i display monocromatici, **gnuplot** non rispetta i colori di primo piano (foreground) o di sfondo (background). Il default è nero su bianco. **-rv** o **gnuplot*reverseVideo: on** richiede bianco su nero.

Color_resources

Per i display a colori, **gnuplot** rispetta le seguenti risorse (mostrate qui con i loro valori predefiniti) o le risorse in scala di grigi. I valori possono essere nomi di colori come elencati nel file BE rgb.txt sul proprio sistema, specifiche esadecimali di colori RGB (vedere documentazione BE), o il nome di un colore seguito da una virgola e da un valore **intensity** (intensità) da 0 a 1. Per esempio, **blue, 0.5** significa un blu a mezza intensità.

```
gnuplot*background: white
gnuplot*textColor: black
gnuplot*borderColor: black
gnuplot*axisColor: black
gnuplot*line1Color: red
gnuplot*line2Color: green
gnuplot*line3Color: blue
gnuplot*line4Color: magenta
gnuplot*line5Color: cyan
gnuplot*line6Color: sienna
gnuplot*line7Color: orange
gnuplot*line8Color: coral
```

La sintassi della linea di comando per questi è, ad esempio,

Esempio:

```
gnuplot -background coral
```

Grayscale_resources

Quando viene selezionato **-gray**, **gnuplot** rispetta le risorse seguenti per i display in scala di grigi o a colori (mostrati qui con i loro valori predefiniti). Notare che lo sfondo predefinito è nero.

```
gnuplot*background: black
gnuplot*textGray: white
gnuplot*borderGray: gray50
gnuplot*axisGray: gray50
gnuplot*line1Gray: gray100
gnuplot*line2Gray: gray60
gnuplot*line3Gray: gray80
gnuplot*line4Gray: gray40
gnuplot*line5Gray: gray90
gnuplot*line6Gray: gray50
gnuplot*line7Gray: gray70
gnuplot*line8Gray: gray30
```

Line_resources

gnuplot rispetta le seguenti risorse per impostare lo spessore (in pixel) delle linee del grafico (mostrate qui con i loro valori predefiniti). 0 o 1 significa uno spessore di linea minimo di 1 pixel. Un valore di 2 o 3 può migliorare l'aspetto di alcuni grafici.

```
gnuplot*borderWidth: 2
gnuplot*axisWidth: 0
gnuplot*line1Width: 0
gnuplot*line2Width: 0
gnuplot*line3Width: 0
gnuplot*line4Width: 0
gnuplot*line5Width: 0
gnuplot*line6Width: 0
gnuplot*line7Width: 0
gnuplot*line8Width: 0
```

gnuplot rispetta le seguenti risorse per impostare lo stile del trattino (dash) usato per le linee di plotting. 0 significa una linea continua. Un numero a due cifre **jk** (**j** e **k** sono >= 1 e <= 9) significa una linea tratteggiata con un pattern ripetuto di **j** pixel on seguito da **k** pixel off. Per esempio, '16' è una linea "punteggiata" con un pixel on seguito da sei pixel off. Pattern on/off più elaborati possono essere specificati con un valore a quattro cifre. Per esempio, '4441' è quattro on, quattro off, quattro on, uno off. I valori predefiniti mostrati qui sotto sono per i display monocromatici o rendering monocromatici su display a colori o su scala di grigi. Per i display a colori, il valore predefinito per ciascuno è 0 (linea continua) tranne che per **axisDashes** che ha come valore predefinito una linea punteggiata '16'.

```
gnuplot*borderDashes: 0
gnuplot*axisDashes: 16
gnuplot*line1Dashes: 0
gnuplot*line2Dashes: 42
gnuplot*line3Dashes: 13
gnuplot*line4Dashes: 44
gnuplot*line5Dashes: 15
gnuplot*line6Dashes: 4441
gnuplot*line7Dashes: 42
gnuplot*line8Dashes: 13
```

Caca

[SPERIMENTALE] Il terminale **caca** è una modalità di output realizzata principalmente per divertimento la quale utilizza **libcaca** per plottare usando caratteri ascii. A differenza del terminale **dumb** essa include supporto per colore, riempimento, immagini, testo ruotato, poligoni pieni, e interazione del mouse.

Sintassi:

```
set terminal caca {{driver | format} {default | <driver> | list}}
                  {color | monochrome}
                  {{no}inverted}
                  {enhanced | noenhanced}
                  {background <rgb color>}
                  {title "<plot window title>"}
                  {size <width>,<height>}
                  {charset ascii|blocks|unicode}
```

L'opzione **driver** seleziona il diplay driver **libcaca** o il **format** di esportazione. Usare **default** significa lasciare che **libcaca** scelga il display driver predefinito della piattaforma. Il driver predefinito può essere cambiato impostando la variabile d'ambiente CACA_DRIVER prima di avviare **gnuplot**. Usare **set term caca driver list** per stampare una lista delle modalità di output supportate.

Le opzioni **color** e **monochrome** selezionano l'output colorato o mono. Notare che questo cambia anche i simboli della linea. Usare l'opzione **inverted** se si preferisce uno sfondo nero invece del bianco predefinito. Questo cambia anche il colore predefinito dei tipi di linea da nero a bianco.

Il supporto del testo avanzato può essere attivato usando l'opzione **enhanced**, vedere **enhanced text (p. 35)**.

Il titolo della finestra di output può essere cambiato con l'opzione **title**, se supportato dal driver **libcaca**.

L'opzione **size** seleziona la dimensione del canvas in caratteri. Il default è 80 per 25. Se supportata dal backend, la dimensione del canvas sarà automaticamente adattata alla dimensione della finestra/terminale attuale. La dimensione predefinita della finestra "x11" e "gl" può essere controllata tramite la variabile d'ambiente CACA_GEOMETRY. La geometria della finestra del driver "win32" può essere controllata e modificata in modo permanente tramite il menu dell'applicazione.

L'opzione **charset** seleziona il set di caratteri usati per linee, punti, il riempimento di poligoni e scatole e la retinatura (dithering) di immagini. Si noti che alcune combinazioni backend/terminale/font potrebbero non supportare alcuni caratteri dei set **blocks** o **unicode**. Su Windows si raccomanda di usare un font non-raster come "Lucida Console" o "Consolas".

Il terminale caca supporta l'interazione con il mouse. Bisogna tenere presente che alcuni backend di **libcaca** (per esempio slang, ncurses) aggiornano la posizione del mouse solo su clic del mouse. I tasti modificatori (ctrl, alt, shift) non sono supportati da **libcaca** e quindi non sono disponibili.

L'**encoding** (codifica) predefinita del terminale **caca** è utf8. Supporta anche l'**encoding** cp437.

Il numero di colori supportati dai backend **libcaca** può variare. La maggior parte dei backend supporta solo 16 colori di primo piano e 16 colori di sfondo, mentre il backend "x11" supporta il truecolor, per esempio.

A seconda del terminale e del backend **libcaca**, potrebbero essere supportati solo 8 diversi colori di sfondo. I colori brillanti (con il bit più significativo del set di colori di sfondo) sono quindi interpretati come indicatori per il testo lampeggiante. In questo caso si può provare ad usare **background rgb "gray"**.

Vedere anche il sito web di libcaca http://caca.zoy.org/wiki/libcaca

e le variabili d'ambiente di libcaca http://caca.zoy.org/doxygen/libcaca/libcaca-env.html

Limitazioni e bug di caca

Il terminale **caca** ha bug e limitazioni noti:

Il supporto Unicode dipende dal driver e dal terminale. Il backend "x11" supporta unicode dalla versione 0.99.beta17 di libcaca. A causa di un bug in **libcaca** <0.99.beta20, il driver "slang" non supporta unicode. Si noti che **libcaca** <0.99.beta19 contiene un bug che provoca un loop infinito se si forniscono sequenze illegali di 8 bit.

I colori di sfondo brillanti possono causare lampeggiamento.

I tasti modificatori non sono supportati per l'uso del mouse, vedere **term caca (p. 251)**.

Il testo avanzato ruotato e la trasparenza non sono supportati. L'opzione **size** non è considerata per la visualizzazione sullo schermo.

Per disegnare correttamente la casella della chiave (key box), usare

```
set key width 1 height 1
```

L'allineamento del testo avanzato è sbagliato se contiene caratteri utf8. Il ridimensionamento della finestra della console di Windows non funziona correttamente a causa di un bug in libcaca. Chiudere la finestra del terminale facendo clic su "X" sulla riga del titolo terminerà wgnuplot. Premere "q" per chiudere la finestra.

Cairolatex

Il terminale **cairolatex** genera output incapsulato PostScript (*.eps), PDF o PNG usando le librerie di supporto cairo e pango, e utilizza LaTeX per l'output di testo usando le stesse routine del terminale **epslatex**.

Sintassi:

```
set terminal cairolatex
               {eps | pdf | png}
               {standalone | input}
               {blacktext | colortext | colourtext}
               {header <header> | noheader}
               {mono|color}
               {{no}transparent} {{no}crop} {background <rgbcolor>}
               {font <font>} {fontscale <scale>}
               {linewidth <lw>} {rounded|butt|square} {dashlength <dl>}
               {size <XX>{unit},<YY>{unit}}
               {resolution <dpi>}
```

Il terminale cairolatex stampa un grafico come **terminal epscairo** o **terminal pdfcairo** ma trasferisce i testi in LaTeX invece di includerli nel graph. Per il riferimento alle opzioni non spiegate qui vedere **pdfcairo (p. 285)**.

eps, **pdf**, o **png** selezionano il tipo di output grafico. Usare **eps** con latex/dvips e **pdf** per pdflatex. Se il proprio grafico ha un elevato numero di punti, usare **png** per mantenere le dimensioni del file ridotte. Quando si usa l'opzione **png**, il terminale accetta un'opzione **resolution** (risoluzione) extra per controllare la densità dei pixel del PNG risultante. L'argomento di **resolution** è un intero con l'unità implicita di DPI.

blacktext costringe tutto il testo ad essere scritto in nero anche in modalità colore;

Il driver **cairolatex** offre un modo speciale per controllare il posizionamento
del testo:

(a) Se una qualsiasi stringa di testo inizia con '{', è necessario includere anche una '}' alla fine del testo, e l'intero testo sarà centrato sia orizzontalmente che verticalmente da LaTeX. (b) Se la stringa di testo inizia con '[', è necessario continuare con: una specificazione di posizione (fino a due tra t,b,l,r,c), ']{', il testo stesso, e infine, '}'. Il testo stesso può essere qualsiasi cosa che LaTeX può comporre (typeset) come un LR-box. \rule{}{}' può aiutare per un miglior posizionamento. Vedere anche la documentazione per il driver del terminale **pslatex (p. 294)**. Per creare etichette multilinea, usare \shortstack, per esempio

```
set ylabel '[r]{\shortstack{first line \\ second line}}'
```

L'opzione **back** dei comandi **set label** è gestita in modo leggermente diverso rispetto agli altri terminali. Le etichette che usano 'back' sono stampate dietro tutti gli altri elementi del grafico, mentre le etichette che usano 'front' sono stampate sopra tutto il resto.

Il driver produce due file diversi, uno per la parte eps, pdf, o png della figura e uno per la parte LaTeX. Il nome del file LaTeX è preso dal comando **set output**. Il nome del file eps/pdf/png si ottiene sostituendo l'estensione del file (normalmente '.tex') con '.eps'/'.pdf'/'.png'. Non c'è nessun output LaTeX se non viene dato un file di output! Si ricorda di chiudere l'**output file** prima del grafico successivo, a meno che non sia in modalità **multiplot**.

Usare '\input{filename}' nei propri documenti LaTeX per includere la figura. Il file '.eps'/'.pdf'/'.png' è incluso dal comando \includegraphics{...}, quindi bisognerà includere anche \usepackage{graphicx} nel preambolo LaTeX. Se si vuole usare il testo colorato (opzione **colourtext**) è necessario includere anche \usepackage{color} nel preambolo LaTeX.

Il comportamento relativo alla selezione del font dipende dalla modalità di intestazione (header mode). In tutti i casi, la dimensione del font data viene usata per calcolare la corretta spaziatura. Quando non si usa la modalità **standalone**, vengono presi l'effetivo font LaTeX e la dimensione del font nel punto di inserimento, quindi bisogna usare i comandi LaTeX per modificare i font. Se si usa, ad esempio, 12pt come dimensione del font per il proprio documento LaTeX, bisogna usare '", 12"' come opzioni. Il nome del font è ignorato. Se si usa **standalone** vengono utilizzati il font e la dimensione del font dati, vedere sotto per una descrizione dettagliata.

Se il testo viene stampato a colori, è controllato dalle variabili booleane \ifGPcolor e \ifGPblacktext di TeX. Solo se \ifGPcolor è vero e \ifGPblacktext è falso, il testo è stampato a colori. È possibile cambiarle nel file TeX generato o fornirle globalmente nel proprio file TeX, per esempio usando

```
\newif\ifGPblacktext
\GPblacktexttrue
```

nel preambolo del proprio documento. L'assegnazione locale viene fatta solo se non viene dato un valore globale.

Quando si usa il terminale cairolatex è necessario dare il nome del file TeX nel comando **set output** includendo l'estensione del file (normalmente ".tex"). Il filename del grafico (graph) viene generato sostituendo l'estensione.

Se si usa la modalità **standalone** viene aggiunto un header LaTeX completo al file LaTeX ; e "-inc" viene aggiunto al filename del garfico (graph). La modalità **standalone** genera un file TeX che produce un output con la dimensione corretta quando si usa dvips, pdfTeX o VTeX. Il default, **input**, genera un file che deve essere incluso in un documento LaTeX usando il comando \input.

Se viene dato un font diverso da "" o "default", esso viene interpretato come nome del font LaTeX. Contiene fino a tre parti, separate da una virgola: 'fontname,fontseries,fontshape'. Se sono richiesti i fontshape o fontseries predefiniti, essi possono essere omessi. Quindi, la vera sintassi per il fontname è '{fontname}{,fontseries}{,fontshape}'. La convenzione di denominazione (naming convention) per tutte le parti è data dallo schema dei font LaTeX. Il fontname è lungo da 3 a 4 caratteri ed è costruito come segue: un carattere per il fornitore (vendor) del font, due caratteri per il nome del font, ed eventualmente un carattere aggiuntivo per i font speciali, ad es., 'j' per i font con numeri in vecchio stile (old-style numerals) o 'x' per i font esperti (expert). I nomi di molti font sono descritti in http://www.tug.org/fontname/fontname.pdf

Per esempio, 'cmr' sta per Computer Modern Roman, 'ptm' per Times-Roman, e 'phv' per Helvetica. La serie (series) del font denota lo spessore dei glifi, nella maggior parte dei casi 'm' per normale ("medium") e 'bx' o 'b' per i font in grassetto (bold). La forma (shape) del font è 'n' per upright, 'it' per italics (corsivo), 'sl' per slanted (inclinato), o 'sc' per small caps (maiuscoletto), in generale. Alcuni font possono fornire diverse serie o forme di font.

Esempi:

Usa Times-Roman in grassetto (con la stessa forma del testo circostante):

```
set terminal cairolatex font 'ptm,bx'
```

Usa Helvetica, grassetto, corsivo:

```
set terminal cairolatex font 'phv,bx,it'
```

Continua a usare il font circostante in forma inclinata:

```
set terminal cairolatex font ',,sl'
```

Usa il maiuscoletto:

```
set terminal cairolatex font ',,sc'
```

Con questo metodo, vengono cambiati solo i font del testo. Se si desidera cambiare anche i font matematici è necessario utilizzare il file "gnuplot.cfg" o l'opzione **header**, descritta più avanti.

In modalità **standalone**, la dimensione del font è presa dalla dimensione del font data nel comando **set terminal**. Per poter utilizzare una dimensione di font specificata, un file "size<size>.clo" deve risiedere nel percorso di ricerca di LaTeX. Per default, 10pt, 11pt, e 12pt sono supportate. Se viene installato il pacchetto "extsizes", sono aggiunte 8pt, 9pt, 14pt, 17pt, e 20pt.

L'opzione **header** prende una stringa come argomento. Questa stringa viene scritta nel file LaTeX generato. Se si usa la modalità **standalone**, essa viene scritta nel preambolo, subito prima del comando \begin{document}. Nella modalità **input**, è posta subito dopo il comando \begingroup per fare in modo che tutte le impostazioni siano locali al grafico.

Esempi:

Usa la codifica di font T1, cambia il font di testo e il font matematico in Times-Roman, e il font sans-serif in Helvetica:

```
set terminal cairolatex standalone header \
"\\usepackage[T1]{fontenc}\n\\usepackage{mathptmx}\n\\usepackage{helvet}"
```

Usa un font in grassetto nel grafico, senza influenzare il testo al di fuori del grafico:

```
set terminal cairolatex input header "\\bfseries"
```

Se il file "gnuplot.cfg" viene trovato da LaTeX, esso viene inserito nel preambolo del documento LaTeX, quando si usa la modalità **standalone**. Può essere usato per ulteriori impostazioni, ad es., cambiare il font del documento in Times-Roman, Helvetica e Courier, compresi i font matematici (gestiti da "mathptmx.sty"):

```
\usepackage{mathptmx}
\usepackage[scaled=0.92]{helvet}
\usepackage{courier}
```

Il file "gnuplot.cfg" viene caricato prima delle informazioni di intestazione date dal comando **header**. Perciò, è possibile usare **header** per sovrascrivere alcune delle impostazioni effettuate usando "gnuplot.cfg"

Canvas

Il terminale **canvas** crea un set di comandi javascript che disegnano sull'elemento canvas di HTML5. Sintassi:

```
set terminal canvas {size <xsize>, <ysize>} {background <rgb_color>}
                    {font {<fontname>}{,<fontsize>}} | {fsize <fontsize>}
                    {{no}enhanced} {linewidth <lw>}
                    {rounded | butt | square}
                    {dashlength <dl>}
                    {standalone {mousing} | name '<funcname>'}
                    {jsdir 'URL/for/javascripts'}
                    {title '<some string>'}
```

dove <xsize> e <ysize> impostano la dimensione dell'area del grafico in pixel. La dimensione predefinità in modalità standalone è 600 per 400 pixel. La dimensione del font predefinita è 10.

NB: È disponibile solo un font, la porzione ascii di Hershey simplex Roman fornita nel file canvastext.js. È possibile sostituirlo con il file canvasmath.js, che contiene anche la codifica UTF-8 di Hershey simplex Greek e simboli matematici. Per coerenza con altri terminali, è anche possibile usare **font "name,size"**. Attualmente il **name** (nome) del font viene ignorato, ma è probabile che arrivi eventualmente il supporto del browser per i font con nome.

La modalità predefinita **standalone** crea una pagina html contenente codice javascript che renderizza il grafico utilizzando l'elemento canvas di HTML 5. La pagina html crea un collegamento a due file javascript richiesti 'canvastext.js' e 'gnuplot_common.js'. Un file aggiuntivo 'gnuplot_dashedlines.js' è necessario per supportare le linee tratteggiate. Per default questi puntano a file locali, su sistemi unix-like solitamente nella directory directory /usr/local/share/gnuplot/<version>/js. Vedere le note di installazione per le altre piattaforme. È possibile cambiare ciò usando l'opzione **jsdir** per specificare o un'altra directory locale o un URL generale. Quest'ultimo è generalmente appropriato se il grafico viene esportato per la visualizzazione su macchine client remote.

Tutti i grafici prodotti dal terminale canvas sono utilizzabili con il mouse. La keyword aggiuntiva **mousing** fa sì che la modalità **standalone** aggiunga una casella di tracciamento del mouse sotto il grafico. Aggiunge anche un link a un file javascript 'gnuplot_mouse.js' e a un foglio di stile (stylesheet) per la casella del mouse 'gnuplot_mouse.css' nella stessa directory locale o URL directory di 'canvastext.js'.

L'opzione **name** crea un file contenente solo javascript. Sia la funzione javascript che contiene che l'id dell'elemento canvas su cui disegna sono presi dal seguente parametro stringa. I comandi

```
set term canvas name 'fishplot'
set output 'fishplot.js'
```

creeranno un file contenente una funzione javascript fishplot() che disegnerà su un canvas con id=fishplot. Una pagina html che invoca questa funzione javascript deve anche caricare la funzione canvastext.js come descritto sopra. Un file html minimo per racchiudere il fishplot creato sopra potrebbe essere:

```
<html>
<head>
    <script src="canvastext.js"></script>
    <script src="gnuplot_common.js"></script>
</head>
<body onload="fishplot();">
    <script src="fishplot.js"></script>
    <canvas id="fishplot" width=600 height=400>
        <div id="err_msg">No support for HTML 5 canvas element</div>
    </canvas>
</body>
</html>
```

I singoli grafici disegnati su questo canvas avranno i nomi fishplot_plot_1, fishplot_plot_2, e così via. Questi possono essere referenziati da routine javascript esterne, per esempio gnuplot.toggle_visibility("fishplot_plot_2").

Cgm

Il terminale **cgm** genera un Computer Graphics Metafile, Version 1. Questo formato di file è un sottoinsieme dello standard ANSI X3.122-1986 chiamato "Computer Graphics - Metafile for the Storage and Transfer of Picture Description Information".

Sintassi:

```
    set terminal cgm {color | monochrome} {solid | dashed} {{no}rotate}
                     {<mode>} {width <plot_width>} {linewidth <line_width>}
                     {font "<fontname>,<fontsize>"}
                     {background <rgb_color>}
[deprecated]         {<color0> <color1> <color2> ...}
```

solid disegna tutte le curve con linee continue, sovrascrivendo qualsiasi pattern tratteggiato; <mode> è **landscape**, **portrait**, o **default**; <plot_width> è la larghezza presunta del grafico in punti; <line_width> è lo spessore della linea in punti (default 1); <fontname> è il nome di un font (vedere l'elenco dei font più avanti) <fontsize> è la dimensione del font in punti (default 12).

Le prime sei opzioni possono essere in qualsiasi ordine. Selezionando **default** si impostano tutte le opzioni ai loro valori predefiniti.

Il meccanismo di impostazione dei colori delle linee nel comando **set term** è deprecato. Invece si dovrebbe impostare lo sfondo usando una keyword separata e impostare i colori delle linee usando **set linetype**. Il meccanismo deprecato accettava colori della forma 'xrrggbb', dove x è il carattere letterale 'x' e 'rrggbb' sono le componenti rosso, verde e blu in esadecimale. Il primo colore era usato per lo sfondo, i colori seguenti sono assegnati ai tipi di linea successivi.

Esempi:

```
set terminal cgm landscape color rotate dashed width 432 \
               linewidth 1  'Helvetica Bold' 12       # defaults
set terminal cgm linewidth 2  14  # linee più spesse & font più grande
set terminal cgm portrait "Times Italic" 12
set terminal cgm color solid      # niente trattini fastidiosi!
```

Cgm font

La prima parte di un Computer Graphics Metafile, la descrizione del metafile, include una tabella di font. Nel corpo dell'immagine, un carattere è designato da un indice in questa tabella. Di default, questo terminale genera una tabella con i seguenti 35 font, più altri sei con **italic** sostituito da **oblique**, o vice-versa (dato che almeno i filtri di importazione CGM di Microsoft Office e Corel Draw trattano **italic** e **oblique** come equivalenti):

Font CGM	
Helvetica	Hershey/Cartographic_Roman
Helvetica Bold	Hershey/Cartographic_Greek
Helvetica Oblique	Hershey/Simplex_Roman
Helvetica Bold Oblique	Hershey/Simplex_Greek
Times Roman	Hershey/Simplex_Script
Times Bold	Hershey/Complex_Roman
Times Italic	Hershey/Complex_Greek
Times Bold Italic	Hershey/Complex_Italic
Courier	Hershey/Complex_Cyrillic
Courier Bold	Hershey/Duplex_Roman
Courier Oblique	Hershey/Triplex_Roman
Courier Bold Oblique	Hershey/Triplex_Italic
Symbol	Hershey/Gothic_German
ZapfDingbats	Hershey/Gothic_English
Script	Hershey/Gothic_Italian
15	Hershey/Symbol_Set_1
	Hershey/Symbol_Set_2
	Hershey/Symbol_Math

I primi tredici di questi font sono necessari per WebCGM. Il filtro di importazione CGM Microsoft Office implementa i 13 font standard elencati sopra, e anche 'ZapfDingbats' e 'Script'. Tuttavia, il font script può essere accessibile solo con il nome '15'. Per saperne di più sulle sostituzioni di font del filtro di importazione di Microsoft, consultare il suo file help che può essere trovato qui:

```
C:\Program Files\Microsoft Office\Office\Cgmimp32.hlp
```

e/o il suo file di configurazione, che può essere trovato qui:

```
C:\Program Files\Common Files\Microsoft Shared\Grphflt\Cgmimp32.cfg
```

Nel comando **set term**, si può specificare un nome di font che non appare nella tabella dei font di default. In tal caso, viene realizzata una nuova tabella di font con il font specificato come prima entry. È necessario assicurarsi che l'ortografia, l'uso delle maiuscole e la spaziatura del nome siano appropriati per l'applicazione che leggerà il file CGM. (Gnuplot e qualsiasi applicazione conforme a MIL-D-28003A ignorano le maiuscole e le minuscole nei nomi dei font). Se si desidera aggiungere molti nuovi font, è necessario usare diversi comandi **set term**.

Esempio:

```
set terminal cgm 'Old English'
set terminal cgm 'Tengwar'
set terminal cgm 'Arabic'
set output 'myfile.cgm'
plot ...
set output
```

Non è possibile inserire un nuovo font in un comando **set label**.

Cgm fontsize

I font vengono ridimensionati supponendo che la pagina sia larga 6 pollici. Se il comando **size** viene usato per cambiare il rapporto d'aspetto della pagina o il file CGM viene convertito a una larghezza diversa, le dimensioni dei font risultanti verranno aumentate o diminuite di conseguenza. Per cambiare la larghezza presunta, usare l'opzione **width**.

Cgm linewidth

L'opzione **linewidth** imposta lo spessore delle linee in pt. Lo spessore predefinito è 1 pt. Il ridimensionamento è influenzato dalla larghezza effettiva della pagina, come discusso sotto le opzioni **fontsize** e **width**.

Cgm rotate

L'opzione **norotate** può essere usata per disabilitare la rotazione del testo. Per esempio, il filtro di input CGM per Word per Windows 6.0c può accettare il testo ruotato, ma l'editor DRAW all'interno di Word non può. Se si modifica un grafico (graph) (per esempio, per etichettare una curva), tutto il testo ruotato viene riportato in orizzontale. L'etichetta dell'asse Y si estenderà quindi oltre il confine del disegno (clip boundary). Con **norotate**, l'etichetta dell'asse Y inizia in una posizione meno esteticamente piacevole, ma la pagina può essere modificata senza danni. L'opzione **rotate** conferma il comportamento predefinito.

Cgm solid

L'opzione **solid** può essere usata per disabilitare gli stili di linea tratteggiata nei grafici. Ciò è utile quando il colore è abilitato e il tratteggio delle linee sminuisce l'aspetto del grafico. L'opzione **dashed** conferma il comportamento predefinito, che assegna un dash pattern diverso a ogni tipo di linea.

Cgm size

La dimensione predefinita di un grafico CGM è 32599 unità di larghezza e 23457 unità di altezza per il formato landscape, o 23457 unità di larghezza per 32599 unità di altezza per il formato portrait.

Cgm width

Tutte le distanze nel file CGM sono in unità astratte. L'applicazione che legge il file determina la dimensione del grafico finale. Per default, si presume che la larghezza del grafico finale sia 6 pollici (15.24 cm). Questa distanza è usata per calcolare la dimensione corretta dei font, e può essere cambiata con l'opzione **width**. La keyword deve essere seguita dalla larghezza in punti. (Qui, un punto è 1/72 di pollice, come in PostScript. Questa unità è conosciuta come "big point" (grande punto) in TeX). Le **expressions** (espressioni) di Gnuplot possono essere usate per convertire da altre unità.

Esempio:

```
set terminal cgm width 432           # default
set terminal cgm width 6*72          # come sopra
set terminal cgm width 10/2.54*72    # largo 10 cm
```

Cgm nofontlist

La tabella dei font predefinita include i font raccomandati per WebCGM, che sono compatibili con il filtro di input Computer Graphics Metafile per Microsoft Office e Corel Draw. Un'altra applicazione potrebbe usare font diversi e/o nomi di font diversi, che potrebbero non essere documentati. L'opzione **nofontlist** (sinonimo **winword6**) elimina la tabella dei font dal file CGM. In questo caso, l'applicazione di lettura dovrebbe usare

una tabella predefinita. Gnuplot userà comunque la sua tabella dei font predefinita per selezionare gli indici dei font. Quindi, 'Helvetica' darà un indice di 1, che dovrebbe fornire la prima entry nella tabella dei font predefinita della propria applicazione. 'Helvetica Bold' darà la sua seconda entry, ecc.

Context

ConTeXt è un pacchetto di macro per TeX, altamente integrato con Metapost (per disegnare figure) e destinato alla creazione di documenti PDF di alta qualità. Il terminale emette il sorgente Metafun, che può essere modificato manualmente, ma si dovrebbe essere in grado di configurare la maggior parte delle cose dall'esterno.

Per un utente medio del modulo ConTeXt + di gnuplot si raccomanda di fare riferimento a **Using ConTeXt** piuttosto che leggere questa pagina o leggere il manuale del modulo gnuplot per ConTeXt.

Il terminale **context** supporta le opzioni seguenti:

Sintassi:

```
set term context {default}
        {defaultsize | size <scale> | size <xsize>{in|cm}, <ysize>{in|cm}}
        {input | standalone}
        {timestamp | notimestamp}
        {noheader | header "<header>"}
        {color | colour | monochrome}
        {rounded | mitered | beveled} {round | butt | squared}
        {dashed | solid} {dashlength | dl <dl>}
        {linewidth | lw <lw>}
        {fontscale <fontscale>}
        {mppoints | texpoints}
        {inlineimages | externalimages}
        {defaultfont | font "{<fontname>}{,<fontsize>}"}
```

Hanno senso, nella grafica non-standalone (**input**), solo i parametri **size** per selezionare la dimensione della grafica, **fontscale** per ridimensionare tutte le etichette per un fattore <fontscale> e dimensione del font, il resto viene ignorato e dovrebbe essere configurato nel file .tex che immette la grafica. È altamente consigliato impostare la dimensione del font appropriata se il font del documento è diverso da 12pt, in modo che gnuplot sappia quanto spazio riservare alle etichette.

default ripristina tutte le opzioni ai loro valori predefiniti.

defaultsize imposta la dimensione del grafico a 5in,3in (pollici). **size** <scale> imposta la dimensione del grafico a <scale> per <default value>. Se vengono dati due argomenti (separati da ','), il primo stabilisce la dimensione orizzontale e il secondo la dimensione verticale. Le dimensioni possono essere date senza unità (in tal caso significa relative al valore predefinito), in pollici ('in') o centimetri ('cm').

input (default) crea una grafica che può essere inclusa in un altro documento ConTeXt. **standalone** aggiunge alcune linee, in modo che il documento possa essere compilato così com'è. In quel caso sarebbe meglio aggiungere anche **header**.

Usare **header** per qualsiasi impostazione/definizione/macro aggiuntiva che si potrebbe voler includere in una grafica indipendente (standalone). **noheader** è il default.

notimestamp impedisce di stampare l'ora di creazione nei commenti (se si usa il controllo versione, a volte si preferisce non impiegare la nuova versione quando cambia solo la data).

color è il default per realizzare grafici a colori, ma **monochrome** non fa ancora nulla di speciale. Se avete qualche buona idea su come il comportamento dovrebbe essere diverso per adattarsi meglio alle stampanti monocromatiche, i vostri suggerimenti sono ben accetti.

rounded (default), **mitered** e **beveled** controllano la forma delle giunzioni delle linee (line joins). **round** (default), **butt** e **squared** controllano la forma delle estremità delle linee (line caps). Vedere PostScript o

PDF Reference Manual per chiarimenti. Per le funzioni instabili (wild-behaving) e le linee spesse è meglio usare **rounded** e **round** per evitare angoli bruschi nelle giunzioni delle linee. (Dovrebbe essere aggiunto a Gnuplot un certo grado di sostegno per questo, in modo che le stesse opzioni possano essere impostate per ogni (stile) linea separatamente).

dashed (default) usa diversi dash pattern per diversi tipi di linea, **solid** disegna tutti i grafici con linee continue.

dashlength o **dl** ridimensiona la lunghezza dei segmenti di linea tratteggiata per <dl>. **linewidth** o **lw** ridimensiona tutti gli spessori di linea per <lw>. (lw 1 sta per 0.5bp, che è lo spessore predefinito della linea quando si disegna con Metapost.) **fontscale** ridimensiona le etichette di testo per il fattore <fontscale> rispetto al font predefinito del documento.

mppoints utilizza forme di punto predefinite, disegnate in Metapost. **texpoints** utilizza un set di simboli facilmente configurabile, definito con ConTeXt nel modo seguente:

```
\defineconversion[my own points][+,{\ss x},\mathematics{\circ}]
\setupGNUPLOTterminal[context][points=tex,pointset=my own points]
```

inlineimages scrive immagini binarie in una stringa e funziona solo in ConTeXt MKIV. **externalimages** scrive file PNG su disco e funziona anche con ConTeXt MKII. Gnuplot deve avere il supporto per le immagini PNG integrato affinché questo funzioni.

Con **font** è possibile impostare il nome e la dimensione del font nella grafica standalone. In modalità non-standalone (**input**) solo la dimensione del font è importante per lasciare abbastanza spazio alle etichette di testo. Il comando

```
set term context font "myfont,ss,10"
```

risulterà in

```
\setupbodyfont[myfont,ss,10pt]
```

Se, per esempio, si imposta **fontscale** a 0.8, allora il font risultante sarà grande 8pt e

```
set label ... font "myfont,12"
```

verrà fuori come 9.6pt.

È propria responsabilità fornire gli adeguati typescript (e header), altrimenti cambiare il font non avrà alcun effetto. Per un font standard in ConTeXt MKII (pdfTeX) si potrebbe usare:

```
set terminal context standalone header '\usetypescript[iwona][ec]' \
    font "iwona,ss,11"
```

Si prega di consultare la documentazione di ConTeXt, la wiki o la mailing list (archivi) per qualsiasi informazione aggiornata sull'uso dei font.

Esempi:

```
set terminal context size 10cm, 5cm     # 10cm, 5cm
set terminal context size 4in, 3in      # 4in, 3in
```

Per grafici standalone (pagina intera) con etichette in codifica UTF-8:

```
set terminal context standalone header '\enableregime[utf-8]'
```

Requisiti

È necessario il modulo gnuplot per ConTeXt http://ctan.org/pkg/context-gnuplot

e una versione recente di ConTeXt. Se si vuole chiamare gnuplot all'istante, è necessario abilitare anche write18. Nella maggior parte delle distribuzioni TeX questo può essere impostato con shell_escape=t in texmf.cnf.

Vedere http://wiki.contextgarden.net/Gnuplot

per i dettagli su questo terminale e per aiuto ed esempi più esaurienti.

Chiamare gnuplot da ConTeXt

Il modo più semplice per creare grafici in documenti ConTeXt è

```
\usemodule[gnuplot]
\starttext
\title{How to draw nice plots with {\sc gnuplot}?}
\startGNUPLOTscript[sin]
set format y "%.1f"
plot sin(x) t '$\sin(x)$'
\stopGNUPLOTscript
\useGNUPLOTgraphic[sin]
\stoptext
```

Questo eseguirà gnuplot automaticamente e includerà la figura risultante nel documento.

Corel

Legacy terminal per CorelDraw (circa 1995).

Sintassi:

```
set terminal corel {monochrome | color} {"<font>" {<fontsize>}}
                   {<xsize> <ysize> {<linewidth> }}
```

dove il fontsize (dimensione del font) e il linewidth (spessore della linea) sono specificati in punti e le dimensioni in pollici. I default sono monochrome, "SwitzerlandLight", 22, 8.2, 10 e 1.2.

Debug

Questo terminale viene fornito per permettere il debug di **gnuplot**. Probabilmente sarà utile solo agli utenti che stanno modificando il codice sorgente.

Domterm

Il terminale **domterm** viene eseguito sull'emulatore di terminale DomTerm, compresi i programmi domterm e qtdomterm. Supporta la grafica SVG incorporata direttamente nell'output del terminale. Vedere http://domterm.org .

Leggere help per il terminale **svg**.

Dumb

Il driver del terminale **dumb** plotta in un blocco di testo usando caratteri ascii. Ha una specificazione opzionale delle dimensioni e un flag di fine riga (linefeed).

Sintassi:

```
set terminal dumb {size <xchars>,<ychars>} {[no]feed}
                  {aspect <htic>{,<vtic>}}
                  {[no]enhanced}
                  {mono|ansi|ansi256|ansirgb}
```

dove <xchars> e <ychars> stabiliscono la dimensione del blocco di testo. Il default è 79 per 24. L'ultimo newline viene stampato solo se **feed** è abilitato.

L'opzione **aspect** può essere usata per controllare il rapporto d'aspetto del grafico impostando la lunghezza dei segni di tic orizzontali e verticali. Sono ammessi solo valori interi. Il default è 2,1 – corrispondente al rapporto d'aspetto dei font dello schermo comuni.

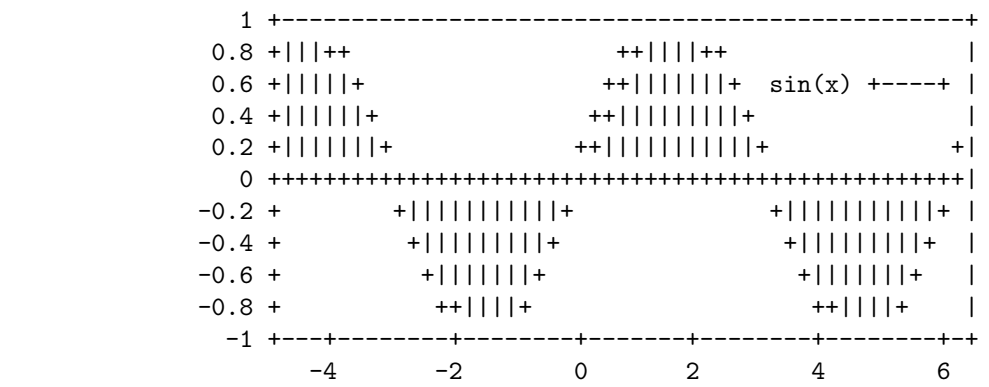
Le opzioni **ansi**, **ansi256**, e **ansirgb** includeranno sequenze di escape nell'output per gestire i colori. Notare che questi potrebbero non essere gestiti dal proprio terminale. Il default è **mono**. Per ottenere la migliore corrispondenza dei colori in modalità **ansi**, si dovrebbe usare **set colorsequence classic**. A seconda della modalità, il terminale **dumb** emetterà le seguenti sequenze

```
ESC [ 0 m           reset attributes to defaults
foreground color:
ESC [ 1 m           set intense/bold
ESC [ 22 m          intense/bold off
ESC [ <fg> m        with color code 30 <= <fg> <= 37
ESC [ 39 m          reset to default
ESC [ 38; 5; <c> m  with palette index 16 <= <c> <= 255
ESC [ 38; 2; <r>; <g>; <b> m  with components 0 <= <r,g,b> <= 255
background color:
ESC [ <bg> m        with color code 40 <= <bg> <= 47
ESC [ 49 m          reset to default
ESC [ 48; 5; <c> m  with palette index 16 <= <c> <= 231
ESC [ 48; 2; <r>; <g>; <b> m  with components 0 <= <r,g,b> <= 255
```

Vedere anche, per esempio, la descrizione su https://en.wikipedia.org/wiki/ANSI_escape_code#Colors

Esempio:

```
set term dumb mono size 60,15 aspect 1
set tics nomirror scale 0.5
plot [-5:6.5] sin(x) with impulse ls -1
```



Dxf

Driver del terminale **dxf** per l'esportazione in AutoCad (Release 10.x). Non ha opzioni. La dimensione predefinita è 120x80 unità AutoCad. **dxf** utilizza sette colori (bianco, rosso, giallo, verde, ciano, blu e magenta) che possono essere cambiati solo modificando il file sorgente. Se viene usato un dispositivo di plotting in bianco e nero, i colori sono mappati su diversi spessori di linea. Nota: per favore qualcuno aggiorni questo terminale allo standard DXF 2012!

Dxy800a

Nota: legacy terminal. Questo driver di terminale supporta il plotter Roland DXY800A. Non ha opzioni.

Eepic

Nota: Legacy terminal (non costruito di default). In versioni precedenti di gnuplot i terminali latex, emtex, eepic, e tpic fornivano un supporto minimo per gli stili grafici al di là di semplici linee e punti. Sono stati direttamente sostituiti dal terminale **pict2e**. Per tipi di terminale compatibili con TeX/LaTeX più abili, vedere **cairolatex (p. 252)**, **context (p. 259)**, **epslatex (p. 265)**, **mp (p. 280)**, **pstricks (p. 296)**, e **tikz (p. 302)**.

L'output di questo terminale è destinato ad essere utilizzato con il pacchetto di macro "eepic.sty" per LaTeX. Per usarlo, sono necessari "eepic.sty" "epic.sty" e un driver DVI che supporti i "tpic" \specials. Se il proprio driver non supporta questi \specials, "eepicemu.sty" permetterà di utilizzare alcuni di essi. dvips e dvipdfm supportano i "tpic" \specials , pdflatex invece no.

Sintassi:

```
set terminal eepic {default} {color|monochrome|dashed}
                   {rotate} {size XX,YY}
                   {small|tiny|<fontsize>}
```

color fa sì che gnuplot produca comandi \color{...} in modo che i grafici (graph) siano colorati. Usando questa opzione, è necessario includere \usepackage{color} nel preambolo del proprio documento latex.

dashed consentirà tipi di linea tratteggiati; senza questa opzione, saranno usate solo linee continue con spessore variabile. **dashed** e **color** si escludono a vicenda; se è specificato **color**, allora **dashed** sarà ignorato.

rotate abiliterà il testo realmente ruotato (di 90 gradi). Altrimenti, il testo ruotato sarà impaginato (typeset) con lettere impilate l'una sull'altra. Se si utilizza questa opzione si deve includere \usepackage{graphicx} nel preambolo.

small userà simboli \scriptsize come marcatori di punto. Il default è usare la dimensione matematica predefinita. **tiny** usa simboli \scriptscriptstyle.

La dimensione predefinita di un grafico eepic è 5x3 pollici. È possibile cambiare ciò usando l'opzione terminale **size**.

<fontsize> è un numero che specifica la dimensione del font all'interno dell'ambiente dell'immagine; l'unità è pt (punti), cioè, 10 pt equivale a circa 3.5 mm. Se il fontsize non è specificato, allora tutto il testo all'interno dell'immagine sarà impostato in \footnotesize.

default ripristina tutte le opzioni ai loro valori predefiniti = nessun colore, nessuna linea tratteggiata, testo pseudo-ruotato (impilato), simboli di punti large.

Note:

Ricordarsi di eseguire l'escape del carattere # (o altri caratteri significativi per (La-)TeX) con \\ (2 back-slash).

Sembra che le linee tratteggiate diventino linee continue quando i vertici di un grafico sono troppo vicini. (Non so se questo è un problema generale con i tpic specials, o se è causato da un bug in eepic.sty o dvips/dvipdfm).

I punti, tra le altre cose, sono disegnati utilizzando i comandi LaTeX "\Diamond", "\Box", ecc. Questi comandi non appartengono più al nucleo (core) di LaTeX2e; sono inclusi nel pacchetto latexsym, che fa parte della distribuzione base e quindi di qualsiasi implementazione di LaTeX. Si prega di non dimenticare di usare questo pacchetto. Invece di latexsym, è anche possibile includere il pacchetto amssymb.

Tutti i driver per LaTeX offrono un modo speciale per controllare il posizionamento del testo: Se una qualsiasi stringa di testo inizia con '{', è necessario includere anche una '}' alla fine del testo, e l'intero testo sarà centrato sia orizzontalmente che verticalmente. Se la stringa di testo inizia con '[', è necessario continuare con una specificazione di posizione (fino a due tra t,b,l,r), ']{', il testo stesso, e infine '}'. Il testo stesso può essere qualsiasi cosa che LaTeX può impaginare (typeset) come una LR-box. '\rule{}{}' può aiutare per un miglior posizionamento.

Esempi:

```
set term eepic
```

grafici (graph) in output come macro eepic all'interno di un ambiente di immagini; \input il file risultante nel proprio documento LaTeX.

```
set term eepic color tiny rotate 8
```

macro eepic con \color macro, \scripscriptsize marcatori di punti, testo realmente ruotato, e tutto il testo impostato con 8pt.

Sul posizionamento dell'etichetta: Usa i valori predefiniti di gnuplot (in genere opportuni, ma a volte non sono i più adatti):

```
set title '\LaTeX\ -- $ \gamma $'
```

Centratura forzata sia in orizzontale che in verticale:

```
set label '{\LaTeX\ -- $ \gamma $}' at 0,0
```

Specifica il proprio posizionamento (qui superiore):

```
set xlabel '[t]{\LaTeX\ -- $ \gamma $}'
```

L'altra etichetta - tiene conto delle ticlabel (etichette dei tic) lunghe:

```
set ylabel '[r]{\LaTeX\ -- $ \gamma $\rule{7mm}{0pt}}'
```

Emf

Il terminale **emf** genera un file Enhanced Metafile Format. Questo formato di file è riconosciuto da molte applicazioni Windows.

Sintassi:

```
set terminal emf {color | monochrome}
                 {enhanced {noproportional}}
                 {rounded | butt}
                 {linewidth <LW>} {dashlength <DL>}
                 {size XX,YY} {background <rgb_color>}
                 {font "<fontname>{,<fontsize>}"}
                 {fontscale <scale>}
```

In modalità **monochrome**, i tipi di linea che si susseguono si alternano attraverso dash pattern. **linewidth** <**factor**> moltiplica tutti gli spessori delle linee per questo fattore. **dashlength** <**factor**> è utile per le linee spesse. <fontname> è il nome di un font font; e <**fontsize**> è la dimensione del font in punti.

La dimensione nominale dell'immagine di output ha come default 1024x768 in unità arbitrarie. È possibile specificare una dimensione nominale diversa usando l'opzione **size**.

La modalità di testo avanzato cerca di approssimare la spaziatura proporzionale dei caratteri. Se si sta usando un font monospaziato, o se non piace l'approssimazione, è possibile disattivare questa correzione usando l'opzione **noproportional**.

Le impostazioni predefinite sono **color font "Arial,12" size 1024,768**. Selezionando **default** si impostano tutte le opzioni ai loro valori predefiniti.

Esempi:

```
set terminal emf 'Times Roman Italic, 12'
```

Emxvga

Nota: legacy terminal. I driver dei terminali **emxvga** e **emxvesa** supportano i PC con schede grafiche SVGA, VESA SVGA e VGA, rispettivamente. Sono destinati ad essere compilati con "emx-gcc" sotto DOS o OS/2. Hanno anche bisogno di VESA e SVGAKIT mantenuti da Johannes Martin (JMARTIN@GOOFY.ZDV.UNI-MAINZ.DE) con aggiunte di David J. Liu (liu@phri.nyu.edu).

Sintassi:

```
set terminal emxvga
set terminal emxvesa {vesa-mode}
```

L'unica opzione è la modalità vesa per **emxvesa**, che per default è G640x480x256.

Epscairo

Il terminale **epscairo** genera PostScript (*.eps) incapsulato usando le librerie di supporto cairo e pango. è richiesta una versione di cairo >= 1.6.

Leggere l'aiuto per il terminale **pdfcairo**.

Epslatex

Il driver **epslatex** genera output per un'ulteriore elaborazione da parte di LaTeX.

Sintassi:

```
set terminal epslatex   {default}
set terminal epslatex   {standalone | input}
                        {oldstyle | newstyle}
                        {level1 | leveldefault | level3}
                        {color | colour | monochrome}
                        {background <rgbcolor> | nobackground}
                        {dashlength | dl <DL>}
                        {linewidth | lw <LW>} {pointscale | ps <PS>}
                        {rounded | butt}
                        {clip | noclip}
                        {palfuncparam <samples>{,<maxdeviation>}}
                        {size <XX>{unit},<YY>{unit}}
                        {header <header> | noheader}
                        {blacktext | colortext | colourtext}
                        {{font} "fontname{,fontsize}" {<fontsize>}}
                        {fontscale <scale>}
```

Il terminale epslatex stampa un grafico come **terminal postscript eps** ma trasferisce i testi in LaTeX invece di includerli nel codice PostScript. Quindi, molte opzioni sono le stesse del **postscript terminal**.

L'aspetto del terminale epslatex è cambiato tra le versioni 4.0 e 4.2 per raggiungere una migliore coerenza con il terminale postscript: La dimensione del grafico è stata cambiata da 5 x 3 pollici a 5 x 3.5 pollici; la larghezza del carattere è ora stimata a essere il 60% della dimensione del font mentre il vecchio terminale epslatex usava il 50%; adesso, viene usato il maggior numero di tipi di linee e simboli postscript. Per ottenere un aspetto quasi identico a quello vecchio è necessario specificare l'opzione **oldstyle**. (In effetti rimangono alcune piccole differenze: le dimensioni dei simboli sono leggermente diverse, i tic sono grandi la metà di quelli del vecchio terminale, ciò può essere modificato usando **set tics scale**, e le frecce hanno tutte le funzionalità presenti nel terminale postscript).

Se appare il messaggio di errore

```
"Can't find PostScript prologue file ... "
```

Si prega di consultare e seguire le istruzioni nel **postscript prologue**.

L'opzione **color** abilita il colore, mentre **monochrome** preferisce elementi di disegno in bianco e nero. Inoltre, **monochrome** usa la **palette** di grigi ma non cambia il colore degli oggetti specificati con un esplicito **colorspec**. **dashlength** o **dl** ridimensiona la lunghezza dei segmenti di linea tratteggiata per <DL>, che è un numero in virgola mobile maggiore di zero. **linewidth** o **lw** ridimensiona tutti i linewidth (spessori delle linee) per <LW>.

Per default il codice PostScript generato usa le funzionalità del linguaggio che sono state introdotte in PostScript Level 2, in particolare i filtri e il pattern-fill di oggetti irregolari come i filledcurve. Le funzionalità di PostScript Level 2 sono condizionatamente protette in modo che gli interpreti di PostScript Level 1 non emettano errori ma, piuttosto, mostrino un messaggio o un'approssimazione di PostScript Level 1. L'opzione **level1** sostituisce le approssimazioni PostScript Level 1 di queste funzionalità e non usa il codice PostScript Level 2. Questo può essere richiesto da alcune vecchie stampanti e da vecchie versioni di Adobe Illustrator. Il flag **level1** può essere attivato in seguito modificando una singola linea nel file di output PostScript per forzare l'interpretazione PostScript Level 1. Nel caso di file contenenti codice di livello 2, le funzionalità di cui sopra non appariranno o saranno sostituite da una nota quando questo flag è impostato o quando il programma di interpretazione non indica che comprende PostScript di livello 2 o superiore. Il flag **level3** abilita la codifica PNG per le immagini bitmap, che può ridurre considerevolmente la dimensione dell'output.

rounded imposta le estremità delle linee e le giunzioni delle linee affinché siano arrotondate; **butt** è il default, estremità butt e giunzioni mitered.

clip dice a PostScript di ritagliare tutto l'output nel bounding box (riquadro di delimitazione); **noclip** è il default.

palfuncparam controlla come **set palette functions** sono codificate come gradienti nell'output. Le funzioni analitiche dei componenti del colore (impostate tramite **set palette functions**) sono codificate come gradienti lineari interpolati nell'output postscript: Le funzioni dei componenti del colore sono campionate in <samples> punti e tutti i punti sono rimossi da questo gradiente che può essere rimosso senza cambiare i colori risultanti di più di <maxdeviation> (deviazione massima). Per quasi tutte le palette utili si può tranquillamente lasciare inalterati i valori predefiniti di <samples>=2000 e <maxdeviation>=0.003.

La dimensione predefinita per l'output postscript è 10 pollici x 7 pollici. Il default per l'output eps è 5 x 3.5 pollici. L'opzione **size** la modifica in qualsiasi dimensione richiesta dell'utente. Per default le dimensioni X e Y sono considerate in pollici, ma sono possibili altre unità (attualmente solo cm). La BoundingBox del grafico viene regolata correttamente per contenere l'immagine ridimensionata. Le coordinate dello schermo vanno sempre da 0.0 a 1.0 lungo tutta la lunghezza dei bordi del grafico come specificato dall'opzione **size**. NB: **questo è un cambiamento rispetto al metodo consigliato in precedenza di usare il comando set size prima di impostare il tipo di terminale**. Il vecchio metodo lasciava la BoundingBox invariata e le coordinate dello schermo non corrispondevano ai limiti reali del grafico.

blacktext costringe tutto il testo ad essere scritto in nero anche in modalità color;

Il driver epslatex offre un modo speciale di controllare il posizionamento del testo: (a) Se una qualsiasi stringa di testo inizia con '{', è necessario includere anche una '}' alla fine del testo, e l'intero testo sarà centrato sia orizzontalmente che verticalmente da LaTeX. (b) Se la stringa di testo inizia con '[', è necessario continuare con una specificazione di posizione (fino a due tra t,b,l,r,c), ']{', il testo stesso, e infine '}'. Il testo stesso può essere qualsiasi cosa che LaTeX può impaginare (typeset) come una LR-box. '\rule{}{}' può aiutare per un miglior posizionamento. Vedere anche la documentazione per il driver del terminale **pslatex (p. 294)**. Per creare etichette multilinea, usare \shortstack, per esempio

```
set ylabel '[r]{\shortstack{first line \\ second line}}'
```

L'opzione **back** dei comandi **set label** è gestita in modo leggermente diverso rispetto agli altri terminali. Le etichette che usano 'back' sono stampate dietro tutti gli altri elementi del grafico, mentre le etichette che usano 'front' sono stampate sopra tutto il resto.

Il driver produce due file diversi, uno per la parte eps della figura e uno per la parte LaTeX. Il nome del file LaTeX è preso dal comando **set output**. Il nome del file eps si ottiene sostituendo l'estensione del file (normalmente **.tex**) con **.eps**. Non vi è alcun output LaTeX se non viene dato un file di output! Si ricorda

di chiudere l'**output file** prima del prossimo grafico, a meno che non sia in modalità **multiplot**.

Per includere la figura nei propri documenti LaTeX bisogna usare '\input{filename}'. Il file **.eps** è incluso dal comando \includegraphics{...}, quindi è necessario includere anche \usepackage{graphicx} nel preambolo LaTeX. Se si desidera utilizzare testo colorato (opzione **textcolour**) si deve includere anche \usepackage{color} nel preambolo LaTeX.

I file pdf possono essere ottenuti dal file eps usando 'epstopdf'. Se il pacchetto grafico è configurato correttamente, i file LaTeX possono anche essere elaborati da pdflatex senza modifiche, usando i file pdf invece dei file eps.

Il comportamento relativo alla selezione del font dipende dalla modalità header. In ogni caso, la dimensione del font fornita è usata per il calcolo della corretta spaziatura. Quando non si usa la modalità **standalone** vengono presi il font LaTeX attuale e la dimensione del font al punto di inclusione, quindi bisogna usare i comandi LaTeX per cambiare i font. Se si usa, ad esempio, 12pt come dimensione del font per il proprio documento LaTeX, bisogna usare '"" 12' come opzioni. Il nome del font è ignorato. Se si utilizza **standalone** vengono usati il font e la dimensione del font dati, vedere più avanti per una descrizione dettagliata.

Se il testo viene stampato a colori è controllato dalle booleane di TeX \ifGPcolor e \ifGPblacktext. Solo se \ifGPcolor è vera e \ifGPblacktext è falsa, il testo è stampato a colori. È possibile cambiarle nel file TeX generato o fornirle globalmente nel proprio file TeX, usando per esempio

```
\newif\ifGPblacktext
\GPblacktexttrue
```

nel preambolo del proprio documento. L'assegnazione locale viene fatta solo se non viene dato un valore globale.

Quando si usa il terminale epslatex si deve dare il nome del file TeX nel comando **set output** includendo l'estensione del file (normalmente ".tex"). Il filename eps viene generato sostituendo l'estensione con ".eps".

Se si usa la modalità **standalone** viene aggiunto un header LaTeX completo al file LaTeX; e "-inc" viene aggiunto al filename del file eps. La modalità **standalone** genera un file TeX che produce un output con la dimensione corretta quando si usa dvips, pdfTeX, o VTeX. Il default, **input**, genera un file che deve essere incluso in un documento LaTeX usando il comando \input.

Se viene dato un font diverso da "" o "default" esso viene interpretato come nome del font LaTeX. Contiene fino a tre parti, separate da una virgola: 'fontname,fontseries,fontshape'. Se sono richiesti i fontshape o fontseries predefiniti, possono essere omessi. Quindi, la vera sintassi per il fontname è '{fontname}{,fontseries}{,fontshape}'. La convenzione di denominazione (naming convention) per tutte le parti è data dallo schema dei font LaTeX. Il fontname è lungo da 3 a 4 caratteri ed è costruito come segue: un carattere per il fornitore (vendor) del font, due caratteri per il nome del font, ed eventualmente un carattere aggiuntivo per i font speciali, ad es., 'j' per i font con numeri in vecchio stile (old-style numerals) o 'x' per i font esperti (expert). I nomi di molti font sono descritti in http://www.tug.org/fontname/fontname.pdf

Per esempio, 'cmr' sta per Computer Modern Roman, 'ptm' per Times-Roman, e 'phv' per Helvetica. La serie (series) del font denota lo spessore dei glifi, nella maggior parte dei casi 'm' per normale ("medium") e 'bx' o 'b' per i font in grassetto (bold). La forma (shape) del font è 'n' per upright, 'it' per italics (corsivo), 'sl' per slanted (inclinato), o 'sc' per small caps (maiuscoletto), in generale. Alcuni font possono fornire diverse serie o forme di font.

Esempi:

Usa Times-Roman in grassetto (con la stessa forma del testo circostante):

```
set terminal epslatex 'ptm,bx'
```

Usa Helvetica, grassetto, corsivo:

```
set terminal epslatex 'phv,bx,it'
```

Continua a usare il font circostante in forma inclinata:

```
set terminal epslatex ',,sl'
```

Usa il maiuscoletto:

```
set terminal epslatex ',,sc'
```

Con questo metodo, vengono cambiati solo i font del testo. Se si desidera cambiare anche i font matematici è necessario utilizzare il file "gnuplot.cfg" o l'opzione **header**, descritta più avanti.

In modalità **standalone**, la dimensione del font è presa dalla dimensione del font fornita nel comando **set terminal**. Per poter utilizzare una dimensione di font specificata, un file "size<size>.clo" deve risiedere nel percorso di ricerca di LaTeX. Per default, 10pt, 11pt, e 12pt sono supportate. Se è installato il pacchetto "extsizes", vengono aggiunte 8pt, 9pt, 14pt, 17pt, e 20pt.

L'opzione **header** prende una stringa come argomento. Questa stringa viene scritta nel file LaTeX generato. Se si usa la modalità **standalone**, viene scritta nel preambolo, subito prima del comando \begin{document}. Nella modalità **input**, è posta subito dopo il comando \begingroup per fare in modo che tutte le impostazioni siano locali al grafico.

Esempi:

Usa la codifica dei font T1, cambia il font di testo e il font matematico in Times-Roman, e il font sans-serif in Helvetica:

```
set terminal epslatex standalone header \
"\\usepackage[T1]{fontenc}\n\\usepackage{mathptmx}\n\\usepackage{helvet}"
```

Usa un font in grassetto nel grafico, senza influenzare il testo al di fuori del grafico:

```
set terminal epslatex input header "\\bfseries"
```

Se il file "gnuplot.cfg" viene trovato da LaTeX, esso viene inserito nel preambolo del documento LaTeX, quando si usa la modalità **standalone**. Può essere usato per ulteriori impostazioni, ad es., cambiare il font del documento in Times-Roman, Helvetica e Courier, compresi i font matematici (gestiti da "mathptmx.sty"):

```
\usepackage{mathptmx}
\usepackage[scaled=0.92]{helvet}
\usepackage{courier}
```

Il file "gnuplot.cfg" viene caricato prima delle informazioni di intestazione date dal comando **header**. Perciò, è possibile usare **header** per sovrascrivere alcune delle impostazioni effettuate usando "gnuplot.cfg"

Epson_180dpi

Nota: disponibile solo se gnuplot è configurato con l'opzione –with-bitmap-terminals. Questo driver supporta una famiglia di stampanti Epson e derivati.

epson_180dpi e **epson_60dpi** sono driver per stampanti Epson stile LQ a 24 aghi con risoluzioni di 180 e 60 punti per pollice, rispettivamente.

epson_lx800 è un driver generico a 9 aghi appropriato per stampanti come la Epson LX-800, la Star NL-10 e NX-1000, la PROPRINTER, e così via.

nec_cp6 è un driver generico a 24 aghi che può essere usato per stampanti come la NEC CP6 e la Epson LQ-800.

Il driver **okidata** supporta le stampanti OKIDATA 320/321 Standard a 9 aghi.

Il driver **starc** è per la Star Color Printer.

Il driver **tandy_60dpi** è per la serie di stampanti Tandy DMP-130 a 9 aghi e 60 dpi (punti per pollice).

Il driver **dpu414** è per la stampante termica Seiko DPU-414.

nec_cp6 ha le opzioni:

Sintassi:

```
set terminal nec_cp6 {monochrome | colour | draft}
```

che di default è monochrome.

dpu414 ha le opzioni:

Sintassi:

```
set terminal dpu414 {small | medium | large} {normal | draft}
```

che di default è medium (=dimensione del font) e normal. Le combinazioni preferite sono **medium normal** e **small draft**.

Excl

Nota: legacy terminal. Il driver del terminale **excl** supporta le stampanti Talaris come la stampante EXCL Laser e la 1590. Non ha opzioni.

Fig

Il terminale **fig** genera output nel linguaggio grafico Fig per l'importazione (import) nello strumento di disegno interattivo xfig. Note:

```
Il terminale fig è stato considerevolmente rivisto nella versione 5.3
di gnuplot. Attualmente è supportata solo la versione 3.2 del formato
di file fig. L'uso dei dash pattern può richiedere Xfig 3.2.6 o più recente.
```

Sintassi:

```
set terminal fig {monochrome | color}
                 {small | big | size <xsize>{in|cm},<ysize>{in|cm}}
                 {landscape | portrait}
                 {font "<fontname>{,<fontsize>}"} {fontsize <size>}
                 {textnormal | {textspecial texthidden textrigid}}
                 {{linewidth|lw} <multiplier>}
```

Le impostazioni predefinite sono

```
set term fig color small landscape font "Times Roman,10" lw 1.0
```

size imposta la dimensione dell'area di disegno a <xsize>*<ysize> in unità di pollici (default) o centimetri. Il default è **size 5in,3in**. **small** è l'abbreviazione di **size 5in,3in** (3in,5in in modalità portrait). **big** è l'abbrieviazione di **size 8in,5in**.

font imposta la variante (face) del font del testo a <fontname> e la sua dimensione a <fontsize> punti. La scelta è limitata ai 35 font PostScript standard. **textnormal** azzera i flag di testo e seleziona i font postscript, **textspecial** imposta i flag di testo per gli specials LaTeX, **texthidden** imposta il flag nascosto e **textrigid** il flag rigid.

linewidth è un moltiplicatore per la proprietà linewidth di tutte le linee.

Sono disponibili anche altri simboli point-plot nel driver **fig**. I simboli possono essere usati tramite valori **pointtype** % 100 sopra 50, con diverse intensità di riempimento controllate da <pointtype> % 5 e contorni in nero (per <pointtype> % 10 < 5) o nel colore attuale. I simboli disponibili sono

```
50 - 59:  cerchi
60 - 69:  quadrati
70 - 79:  diamanti
80 - 89:  triangoli con la punta verso l'alto
90 - 99:  triangoli con la punta verso il basso
```

La dimensione di questi simboli varia con la dimensione del font.

I colori RGB saranno sostituiti con il grigio a meno che non siano stati definiti in un tipo di linea (linetype) prima di iniziare a plottare o corrispondano ad un nome di colore noto o ad un valore di palette. Vedere **colornames (p. 147)**. Per esempio

```
set linetype 999 lc rgb '#aabbcc'
plot $data with fillecurve fillcolor rgb '#aabbcc'
```

Ggi

Legacy terminal driver per il progetto GGI (General Graphics Interface). Sintassi:

```
set terminal ggi [acceleration <integer>] [[mode] {mode}]
```

In X la finestra non può essere ridimensionata usando gli handle del window manager, ma la modalità può essere data con la mode option, ad esempio:

```
- V1024x768
- V800x600
- V640x480
- V320x200
```

Fare riferimento alla documentazione di ggi per altre modalità. La keyword 'mode' è opzionale. Si raccomanda di selezionare il target (obiettivo) tramite variabili d'ambiente come spiegato nella pagina di manuale di libggi. Per mettere DGA su X, si dovrebbe scrivere per esempio

```
bash> export GGI_DISPLAY=DGA
csh>  setenv GGI_DISPLAY DGA
```

'acceleration' è usata solo per i target che riportano eventi relativi al movimento del puntatore (ad es. DGA) ed è un fattore moltiplicativo intero rigorosamente positivo per le distanze relative. Il default per acceleration è 7.

Esempi:

```
set term ggi acc 10
set term ggi acc 1 mode V1024x768
set term ggi V1024x768
```

Gif

Sintassi:

```
set terminal gif
       {{no}enhanced}
       {{no}transparent} {rounded|butt}
       {linewidth <lw>} {dashlength <dl>}
       {tiny | small | medium | large | giant}
       {font "<face> {,<pointsize>}"} {fontscale <scale>}
       {size <x>,<y>} {{no}crop}
       {background <rgb_color>}
       {animate {delay <d>} {loop <n>} {optimize}}
```

Le immagini PNG, JPEG e GIF sono create usando la libreria esterna libgd. I grafici GIF possono essere visualizzati interattivamente inviando (piping) l'output al programma 'display' dal pacchetto ImageMagick come segue:

```
set term gif
set output '| display gif:-'
```

È possibile visualizzare l'output dei comandi plot successivi in modo interattivo digitando <space> nella finestra di visualizzazione. Per salvare il grafico corrente in un file, bisogna cliccare con il tasto sinistro del mouse nella finestra di visualizzazione e scegliere **save**.

transparent ordina al driver di rendere il colore di sfondo trasparente. Il default è **notransparent**.

Le opzioni **linewidth** e **dashlength** sono fattori di scala che influenzano tutte le linee disegnate, cioè vengono moltiplicati per i valori richiesti nei vari comandi di disegno.

butt ordina al driver di usare un metodo di disegno della linea che non superi il punto finale di una linea desiderato. Questa impostazione è applicabile solo per spessori di linea maggiori di 1. Questa impostazione è particolarmente utile quando si disegnano linee orizzontali o verticali. Il default è **rounded**.

La dimensione del grafico in output <x,y> è data in pixel — il defaults è 640x480. Vedere informazioni aggiuntive sotto **canvas (p. 32)** e **set size (p. 205)**. Lo spazio vuoto ai bordi del grafico completato può essere tagliato usando l'opzione **crop**, ottenendo una dimensione finale dell'immagine più piccola. Il default è **nocrop**.

Animate

```
set term gif animate {delay <d>} {loop <n>} {{no}optimize}}
```

L'opzione **animate** del terminale gif crea un singolo file gif contenente più fotogrammi. Il ritardo tra la visualizzazione di fotogrammi successivi può essere specificato in unità di 1/100 di secondo (default 5), ma questo valore può essere rispettato accuratamente o meno da un programma usato per visualizzare l'animazione in seguito. È possibile specificare il numero di loop dell'animazione durante la riproduzione, con il valore predefinito di 0 che significa looping illimitato. Anche in questo caso il valore può essere rispettato o meno dal programma utilizzato in seguito per la visualizzazione. Una sequenza di animazione viene terminata dal prossimo comando **set output** o **set term**.

L'opzione **optimize** [DEPRECATA] viene passata alla libreria gd quando il file di output viene aperto. Ha due effetti sull'animazione.

1) Una singola color map è usata per l'intera animazione. Questo richiede che tutti i colori utilizzati in qualsiasi fotogramma dell'animazione siano già definiti nel primo fotogramma.

2) Se possibile, solo le porzioni di un fotogramma che differiscono dal precedente sono memorizzate nel file di animazione. Questo risparmio di spazio potrebbe non essere possibile se l'animazione usa la trasparenza.

Entrambe queste ottimizzazioni hanno lo scopo di produrre un file di output più piccolo, ma la diminuzione della dimensione è probabilmente significativa solo per le animazioni lunghe. Avvertimento: L'implementazione dell'ottimizzazione in libgd è nota per essere buggata. Pertanto l'uso di questa opzione in gnuplot non è raccomandato.

Esempio di rotazione continua:

```
set term gif animate loop 0
set output 'rotating_surface.gif'
do for [ang=1:359] {
    set view 60, ang
    splot f(x,y) with pm3d
}
unset output
```

Fonts

I dettagli della selezione del font sono complicati. Per maggiori informazioni, vedere la sezione separata sotto **fonts gd (p. 47)**.

Esempi:

```
set terminal gif medium noenhanced size 640,480 background '#ffffff'
```

Usa il font incorporato medium size, non scalabile e non ruotabile. La modalità di testo avanzato non funziona con questo font. Usa il bianco (24-bit RGB in esadecimale) per lo sfondo non trasparente.

```
set terminal gif font arial 14
```

Cerca un font con il nome della variante (face) 'arial' e imposta la dimensione del font a 14pt.

Gpic

Il driver di terminale **gpic** genera grafici (graph) GPIC nel pacchetto "groff" di Free Software Foundations. La dimensione predefinita è 5 x 3 pollici. L'unica opzione è l'origine, che per default è (0,0).

Sintassi:

```
set terminal gpic {<x> <y>}
```

dove **x** e **y** sono in pollici.

Un semplice grafico (graph) può essere formattato usando

```
groff -p -mpic -Tps file.pic > file.ps.
```

L'output di pic può essere inserito (pipe-lined) in eqn, quindi è possibile inserire funzioni complesse in un grafico (graph) con i comandi **set label** e **set {x/y}label**. Per esempio,

```
set ylab '@space 0 int from 0 to x alpha ( t ) roman d t@'
```

etichetterà l'asse y con un integrale se formattato con il comando:

```
gpic filename.pic | geqn -d@@ -Tps | groff -m[macro-package] -Tps
    > filename.ps
```

Le figure fatte in questo modo possono essere ridimensionate per essere inserite in un documento. Il linguaggio pic è facile da capire, quindi i grafici (graph) possono essere modificati a mano se necessario. Tutte le coordinate nel pic-file prodotto da **gnuplot** sono date come x+gnuplotx e y+gnuploty. Per default a x e y viene assegnato il valore 0. Se questa linea viene rimossa con un editor in un certo numero di file, in questo modo possono essere inseriti vari grafici (graph) in una figura (la dimensione predefinita è 5.0x3.0 pollici):

```
.PS 8.0
x=0;y=3
copy "figa.pic"
x=5;y=3
copy "figb.pic"
x=0;y=0
copy "figc.pic"
x=5;y=0
copy "figd.pic"
.PE
```

Questo produrrà una figura di 8 pollici di larghezza con quattro grafici (graph) disposti in due righe una sopra l'altra.

Si può ottenere la stessa cosa specificando x e y nel comando

```
set terminal gpic x y
```

Grass

Nota: legacy terminal. Il driver del terminale **grass** conferisce le capacità di **gnuplot** agli utenti del sistema informativo geografico GRASS. Contattare grassp-list@moon.cecer.army.mil per maggiori informazioni. Le pagine vengono scritte nel frame corrente della GRASS Graphics Window. Non esistono opzioni.

Hp2623a

Il driver del terminale **hp2623a** supporta Hewlett Packard HP2623A. Non ha opzioni.

Hp2648

Il driver del terminale **hp2648** supporta Hewlett Packard HP2647 e HP2648. Non ha opzioni.

Hp500c

Nota: disponibile solo se gnuplot è configurato con l'opzione –with-bitmap-terminals. Il driver del terminale **hp500c** supporta Hewlett Packard HP DeskJet 500c. Ha delle opzioni per la risoluzione e la compressione.

Sintassi:

```
set terminal hp500c {<res>} {<comp>}
```

dove **res** può essere 75, 100, 150 o 300 punti per pollice (dpi) e **comp** può essere "rle", o "tiff". Qualsiasi altro input è sostituito dalle impostazioni predefinite, che sono 75 dpi e nessuna compressione. La rasterizzazione a risoluzioni più elevate può richiedere una grande quantità di memoria.

Hpgl

Il driver del terminale **hpgl** produce un output HPGL per i dispositivi come il plotter HP7475A. Esistono due opzioni che possono essere impostate: il numero di penne ed **eject**, che dice al plotter di espellere una pagina quando ha finito. Il default è usare 6 penne e non espellere la pagina una volta finito.

I set di caratteri internazionali ISO-8859-1 e CP850 sono riconosciuti tramite **set encoding iso_8859_1** o **set encoding cp850**

Sintassi:

```
set terminal hpgl {<number_of_pens>} {eject}
```

La selezione

```
set terminal hpgl 8 eject
```

è equivalente al precedente terminale **hp7550**, e la selezione

```
set terminal hpgl 4
```

è equivalente al precedente terminale **hp7580b**. Le grafiche HPGL possono essere importate da molti pacchetti software.

Hpljii

Nota: disponibile solo se gnuplot è configurato con l'opzione –with-bitmap-terminals. Il driver del terminale **hpljii** supporta la stampante HP Laserjet Series II. Il driver **hpdj** supporta la stampante HP DeskJet 500. Questi driver consentono di scegliere tra diverse risoluzioni.

Sintassi:

```
set terminal hpljii | hpdj {<res>}
```

dove **res** può essere 75, 100, 150 or 300 punti per pollice; il default è 75. La rasterizzazione a risoluzioni più elevate può richiedere una grande quantità di memoria

Il terminale **hp500c** è simile a **hpdj**; **hp500c** supporta inoltre il colore e la compressione.

Hppj

Nota: disponibile solo se gnuplot è configurato con l'opzione –with-bitmap-terminals. Il driver del terminale **hppj** supporta le stampanti HP PaintJet e HP3630. L'unica opzione è la scelta del font.

Sintassi:

```
set terminal hppj {FNT5X9 | FNT9X17 | FNT13X25}
```

con il font (FNT9X17) di medie dimensioni come predefinito.

Imagen

Il driver del terminale **imagen** supporta le stampanti laser Imagen. È in grado di mettere più grafici (graph) su una sola pagina.

Sintassi:

```
set terminal imagen {<fontsize>} {portrait | landscape}
                    {[<horiz>,<vert>]}
```

dove **fontsize** ha come default 12 punti e il layout ha come default **landscape**. <**horiz**> e <**vert**> sono il numero di grafici (graph) nella direzione orizzontale e nella direzione verticale; questi sono predefiniti all'unità

Esempio:

```
set terminal imagen portrait [2,3]
```

colloca sei grafici (graph) sulla pagina in tre righe di due nell'orientamento portrait.

Jpeg

Sintassi:

```
set terminal jpeg
       {{no}enhanced}
       {{no}interlace}
       {linewidth <lw>} {dashlength <dl>} {rounded|butt}
       {tiny | small | medium | large | giant}
       {font "<face> {,<pointsize>}"} {fontscale <scale>}
       {size <x>,<y>} {{no}crop}
       {background <rgb_color>}
```

Le immagini PNG, JPEG e GIF sono create usando la libreria esterna libgd. Nella maggior parte dei casi, PNG è da preferire per i singoli grafici, e GIF per le animazioni. Entrambi sono formati d'immagine senza perdita, e producono una qualità d'immagine migliore del formato con perdita JPEG. Questo è particolarmente evidente per linee a tinta unita su uno sfondo a tinta unita, cioè esattamente il tipo di immagine creata tipicamente da gnuplot.

L'opzione **interlace** crea un'immagine JPEG progressiva. Il default è **nointerlace**.

Le opzioni **linewidth** e **dashlength** sono fattori di scala che influenzano tutte le linee disegnate, cioè vengono moltiplicati per i valori richiesti nei vari comandi di disegno.

butt ordina al driver di usare un metodo di disegno della linea che non superi il punto finale di una linea desiderato. Questa impostazione è applicabile solo per spessori di linea maggiori di 1. Questa impostazione è particolarmente utile quando si disegnano linee orizzontali o verticali. Il default è **rounded**.

I dettagli della selezione del font sono complicati. Due semplici esempi equivalenti sono dati di seguito:

```
set term jpeg font arial 11
set term jpeg font "arial,11"
```

Per maggiori informazioni, vedere la sezione separata sotto **font (p. 46)**.

La dimensione del grafico in output <x,y> è data in pixel — il default è 640x480. Vedere informazioni aggiuntive sotto **canvas (p. 32)** e **set size (p. 205)**. Lo spazio vuoto ai bordi del grafico completato può essere tagliato usando l'opzione **crop**, ottenendo una dimensione finale dell'immagine più piccola. Il default è **nocrop**.

Kyo

I driver dei terminali **kyo** e **prescribe** supportano la stampante laser Kyocera. L'unica differenza tra i due consiste nel fatto che **kyo** usa "Helvetica" mentre **prescribe** usa "Courier". Non esistono opzioni.

Latex

Nota: Legacy terminal (non costruito di default). In versioni precedenti di gnuplot i terminali latex, emtex, eepic, e tpic fornivano un supporto minimo per gli stili grafici al di là di semplici linee e punti. Sono stati direttamente sostituiti dal terminale **pict2e**. Per tipi di terminale compatibili con TeX/LaTeX più abili, vedere **cairolatex (p. 252)**, **context (p. 259)**, **epslatex (p. 265)**, **mp (p. 280)**, **pstricks (p. 296)**, e **tikz (p. 302)**.

Sintassi:

```
set terminal {latex | emtex} {default | {courier|roman} {<fontsize>}}
             {size <XX>{unit}, <YY>{unit}} {rotate | norotate}
             {color | monochrome}
```

Per default il grafico erediterà le impostazioni del font dal documento di incorporamento (embedding). L'utente ha la possibilità di forzare i font Courier (cmtt) o Roman (cmr). In questo caso si può anche specificare un fontsize. A meno che il proprio driver non sia in grado di costruire font di qualsiasi dimensione (ad es. dvips), attenersi alle dimensioni standard di 10, 11 e 12 punti.

Attenzione utenti METAFONT: a METAFONT non piacciono le dimensioni dispari.

Tutti i driver per LaTeX offrono un modo speciale per controllare il posizionamento del testo: Se una qualsiasi stringa di testo inizia con '{', è necessario includere anche una '}' alla fine del testo, e l'intero testo sarà centrato sia orizzontalmente che verticalmente. Se la stringa di testo inizia con '[', è necessario continuare con una specificazione di posizione (fino a due su t,b,l,r), ']{', il testo stesso, e infine '}'. Il testo stesso può essere qualsiasi cosa che LaTeX può impaginare (typeset) come una LR-box. '\rule{}{}' può aiutare per un miglior posizionamento.

I punti, tra le altre cose, sono disegnati utilizzando i comandi LaTeX "\Diamond" e "\Box". Questi comandi non appartengono più al nucleo (core) di LaTeX2e; sono inclusi nel pacchetto latexsym, che fa parte della distribuzione base e quindi di qualsiasi implementazione di LaTeX. Si prega di non dimenticare di usare questo pacchetto. Altri tipi di punto usano simboli dal pacchetto amssymb.

La dimensione predefinita del grafico è 5 pollici per 3 pollici. L'opzione **size** la modifica in qualsiasi dimensione richiesta dell'utente. Per default le dimensioni X e Y sono considerate in pollici, ma sono possibili altre unità (attualmente solo cm).

Se viene specificato **rotate**, è possibile ruotare il testo, specialmente un'etichetta ruotata sull'asse y, (sono necessari i pacchetti graphics o graphicx). Il meccanismo 'stacked' (impilato) dell'etichetta dell'asse y è quindi disattivato. Questo migliorerà significativamente la qualità della tracciatura delle linee, ed è il default dalla versione 5.3.

L'opzione **color** abilita il colore, mentre **monochrome** utilizza solo elementi di disegno in bianco e nero. È necessario caricare il pacchetto color o xcolor nel preambolo del proprio documento latex.

Esempi: Sul posizionamento dell'etichetta: Usa i valori predefiniti di gnuplot (in genere opportuni, ma a volte non sono i più adatti):

```
set title '\LaTeX\ -- $ \gamma $'
```

Centratura forzata sia in orizzontale che in verticale:

```
set label '{\LaTeX\ -- $ \gamma $}' at 0,0
```

Specifica il proprio posizionamento (qui superiore):

```
set xlabel '[t]{\LaTeX\ -- $ \gamma $}'
```

L'altra etichetta – tiene conto delle ticlabel (etichette dei tic) lunghe:

```
set ylabel '[r]{\LaTeX\ -- $ \gamma $\rule{7mm}{0pt}}'
```

Linux console

Le versioni precedenti di gnuplot richiedevano terminali speciali **linux** o **vgagl** per visualizzare la grafica sulla console linux, cioè in assenza di X11 o altri ambienti con finestre (windowing environment). Questi terminali sono stati deprecati.

Il metodo consigliato per eseguire gnuplot dalla console linux ora consiste nell'utilizzare un emulatore di terminale di console come yaft (https://github.com/uobikiemukot/yaft) che supporta la grafica sixel. Se si utilizza yaft come terminale di console, è possibile eseguire gnuplot e selezionare un terminale con l'output di sixel. Vedere **sixelgd (p. 298)**. Come opzione di ripiego si potrebbe usare **set term dumb**, ma la grafica sixel è molto più bella.

Lua

Il driver generico del terminale **lua** funziona insieme a uno script Lua esterno per creare un file di grafico specifico per il target. Attualmente l'unico target supportato è TikZ -> pdflatex.

Informazioni su Lua sono disponibili su http://www.lua.org .

Sintassi:

```
set terminal lua <target name> | "<file name>"
                    {<script_args> ...}
                    {help}
```

È obbligatorio un 'target name' o 'file name' (tra virgolette) per uno script. Se viene dato un 'target name' per lo script, il terminale cercherà "gnuplot-<target name>.lua" nella directory locale e, in caso di fallimento, nella variabile d'ambiente GNUPLOT_LUA_DIR.

Tutti gli argomenti verranno forniti allo script selezionato per un'ulteriore valutazione. Per esempio, 'set term lua tikz help' farà sì che lo script stesso stampi aiuto aggiuntivo sulle opzioni e le scelte per lo script.

Lua tikz

Il driver TikZ è una modalità di output del terminale generico Lua.

Sintassi:

```
set terminal lua tikz

 {latex | tex | context}
 {color | monochrome}
 {nooriginreset | originreset}
 {nogparrows | gparrows}
 {nogppoints | gppoints}
 {picenvironment | nopicenvironment}
 {noclip | clip}
 {butt}
```

```
{notightboundingbox | tightboundingbox}
{background "<colorpec>"}
{size <x>{unit},<y>{unit}}
{scale <x>,<y>}
{plotsize <x>{unit},<y>{unit}}
{charsize <x>{unit},<y>{unit}}
{font "<fontdesc>"}
{{fontscale | textscale} <scale>}
{dashlength | dl <DL>}
{linewidth | lw <LW>}
{nofulldoc | nostandalone | fulldoc | standalone}
{{preamble | header} "<preamble_string>"}
{tikzplot <ltn>,...}
{notikzarrows | tikzarrows}
{rgbimages | cmykimages}
{noexternalimages|externalimages}
{bitmap | nobitmap}
{providevars <var name>,...}
{createstyle}
{help}
```

Tutte le opzioni che richiedono lunghezze come argomenti saranno automaticamente impostate su 'cm' se non viene specificata alcuna unità. Si possono usare le seguenti unità per tutte le lunghezze: 'cm', 'mm', 'in' o 'inch', 'pt', 'pc', 'bp', 'dd', 'cc'. Gli spazi vuoti tra i numeri e le unità non sono ammessi.

'monochrome' disabilita la colorazione delle linee e passa ai riempimenti in scala di grigi.

'originreset' sposta l'origine dell'immagine TikZ nell'angolo inferiore sinistro del grafico. Può essere usata per allineare diversi grafici all'interno di un ambiente tikzpicture. Non è testata con i multiplot e i grafici pm3d!

'gparrows' utilizza la funzione interna di disegno delle frecce di gnuplot invece di quelle fornite da TikZ.

'gppoints' utilizza la funzione interna di disegno del plotmark di gnuplot invece di quelle fornite da TikZ.

'nopicenvironment' omette la dichiarazione dell'ambiente 'tikzpicture' in modo da impostarla manualmente. Questo permette di mettere qualche codice PGF/TikZ direttamente prima o dopo il grafico.

'clip' ritaglia (crop) il grafico alla dimensione del canvas indicata. Il default è 'noclip' con il quale viene impostato solo un bounding box (riquadro di delimitazione) minimo della dimensione del canvas. Non viene impostato né un bounding box fisso né un crop box se viene usata l'opzione 'plotsize' o 'tightboundingbox'.

'butt' cambia la proprietà linecap a "butt" e la proprietà linejoin a "miter". I default sono "round" e "round".

Se è impostato 'tightboundingbox', l'opzione 'clip' viene ignorata e il bounding box finale è il bounding box naturale calcolato da tikz.

'background' imposta il colore di sfondo al valore specificato nell'argomento <colorpec>. <colorspec> deve essere un nome di un colore valido o un codice RGB di 3 byte come un numero esadecimale con un segno numerico precedente ('#'). Ad es. '#ff0000' specifica rosso puro. Se è omesso lo sfondo è trasparente.

L'opzione 'size' si aspetta due lunghezze <x> e <y> come dimensione del canvas. La dimensione predefinita del canvas è 12.5cm x 8.75cm.

L'opzione 'scale' funziona in modo simile all'opzione 'size' ma si aspetta fattori di scala <x> e <y> invece di lunghezze.

L'opzione 'plotsize' permette di impostare la dimensione dell'area del grafico invece della dimensione del canvas, ovvero il comportamento abituale di gnuplot. L'uso di questa opzione può causare lunghezze di tic leggermente asimmetriche. Come 'originreset', questa opzione potrebbe non portare a risultati soddisfacenti se usata con multiplot o grafici pm3d. Un approccio alternativo è quello di impostare tutti i margini a zero

e usare l'opzione 'noclip'. L'area del grafico ha quindi le dimensioni del canvas dato.

L'opzione 'charsize' si aspetta la dimensione media orizzontale e verticale del font usato. Guardare il style file generato per un esempio di come utilizzarla dall'interno del proprio documento TeX.

'fontscale' o 'textscale' si aspetta un fattore di scala come parametro. Tutti i testi nel grafico sono poi scalati da questo fattore.

'dashlength' o 'dl' scala la lunghezza dei segmenti di linea tratteggiata per <DL>, che è un numero in virgola mobile maggiore di zero. 'linewidth' o 'lw' scala tutti i linewidth per <LW>.

Le opzioni 'tex', 'latex' e 'context' scelgono il formato di output TeX. LaTeX è il default. Per poter caricare lo style file, inserire la linea corrispondente all'inizio del proprio documento:

```
\input gnuplot-lua-tikz.tex    % (for plain TeX)
\usepackage{gnuplot-lua-tikz}  % (for LaTeX)
\usemodule[gnuplot-lua-tikz]   % (for ConTeXt)
```

'createstyle' ricava gli stili TeX/LaTeX/ConTeXt dallo script e li scrive nei file appropriati.

'fulldoc' o 'standalone' produce un documento LaTeX completo per la compilazione diretta.

'preamble' o 'header' può essere usato per mettere qualsiasi codice LaTeX aggiuntivo nel preambolo del documento in modalità standalone.

Con l'opzione 'tikzplot' verrà usato il comando '\path plot' invece che solo '\path'. La seguente lista di numeri di linetype (<ltn>,...) definisce le plotline (linee del grafico) interessate. Esiste un plotstyle per ogni linetype. Il plotstyle predefinito è 'smooth' per tutti i linetype >= 1.

Per default il terminale tikz produce semplici frecce LaTeX. Per produrre frecce conformi alle impostazioni 'arrowstyle' di gnuplot, utilizzare l'opzione 'gparrows'. L'opzione 'tikzarrows' è una terza alternativa che bypassa entrambe. Invece il parametro arrowstyle 'angle' è usato per indicizzare un set di 12 stili di freccia predefiniti da TikZ. Per esempio, uno stile di freccia con l'angolo '7' sarà mappato allo stile TikZ 'gp arrow 7' ignorando tutte le altre impostazioni arrowstyle.

Con 'cmykimages' il modello di colore CMYK sarà usato per i dati immagine inline invece del modello RGB. Tutti gli altri colori (come i colori delle linee, ecc.) non sono influenzati da questa opzione, poiché sono gestiti, per esempio, dal pacchetto xcolor di LaTeX. Questa opzione è ignorata se le immagini sono esternalizzate.

Usando l'opzione 'externalimages' tutte le immagini bitmap saranno scritte come immagini PNG esterne e incluse al momento della compilazione del documento. La generazione di file DVI e successivamente postscript richiede di convertire i PNG in file EPS in una fase separata, per esempio usando il **convert** di ImageMagick. Le immagini bitmap trasparenti sono sempre generate come PNG esterne.

L'opzione 'nobitmap' permette di renderizzare le immagini come rettangoli riempiti invece del formato immagine inline nativ PS o PDF. Questa opzione viene ignorata se le immagini sono esternalizzate.

L'opzione 'providevars' rende disponibili le variabili interne e le variabili utente di gnuplot usando il comando '\gpgetvar{<var name>}' all'interno dello script TeX. Usare il comando di gnuplot 'show variables all' per vedere la lista delle variabili valide.

La stringa <fontdesc> può contenere qualsiasi comando font TeX/LaTeX/ConTeXt valido, come ad esempio '\small'. Viene passato direttamente come parametro node (nodo) sotto forma di "font={<fontdesc>}". Questo può essere 'misused' (abusato) per aggiungere ulteriore codice a un nodo, ad es. '\small,yshift=1ex' o ',yshift=1ex' sono anch'essi validi, mentre quest'ultimo non modifica le impostazioni correnti del font. Un'eccezione è il secondo argomento della lista. Se è un numero della forma <number>{unit} sarà interpretato come un fontsize come in altri terminali e sarà aggiunto al primo argomento. Se l'unità è omessa, il valore viene interpretato come 'pt'. Come esempio la stringa '\sffamily,12,fill=red' imposta il font sans serif di LaTeX con una dimensione di 12pt e colore di sfondo rosso. Lo stesso vale per ConTeXt, ad es. '\switchtobodyfont[iwona],10' cambia il font in Iwona con una dimensione di 10pt. Gli utenti di Plain TeX devono cambiare la dimensione del font esplicitamente nel primo argomento. Il secondo dovrebbe essere impostato allo stesso valore per ottenere un corretto ridimensionamento delle caselle di testo.

Le stringhe devono essere messe tra virgolette singole o doppie. Le stringhe tra virgolette doppie possono

contenere caratteri speciali come newline '\n' ecc.

Mf

Il driver del terminale **mf** crea un file di input per il programma METAFONT. Perciò una figura può essere usata nel documento TeX allo stesso modo di un carattere.

Per usare un'immagine in un documento, il programma METAFONT deve essere eseguito con il file di output di **gnuplot** come input. Quindi, l'utente ha bisogno di una conoscenza di base del processo di creazione dei font e della procedura per includere un nuovo font in un documento. Tuttavia, se il programma METAFONT è impostato correttamente nel sito locale, un utente inesperto potrebbe eseguire l'operazione senza problemi particolari.

Il supporto del testo è basato su un set di caratteri METAFONT. Attualmente è inserito il set di font Computer Modern Roman, ma l'utente è libero di scegliere qualsiasi font di cui abbia bisogno. I file sorgente METAFONT per il font scelto devono essere disponibili. Ogni carattere è memorizzato in una variabile immagine separata in METAFONT. Queste variabili possono essere manipolate (ruotate, ridimensionate, ecc.) quando sono necessari i caratteri. L'inconveniente è il tempo di interpretazione nel programma METAFONT. Su alcune macchine (cioè PC) la quantità limitata di memoria disponibile può causare problemi se vengono memorizzate troppe immagini.

Il terminale **mf** non ha opzioni.

Istruzioni METAFONT

- Impostare il proprio terminale su METAFONT:
```
set terminal mf
```

- Selezionare un file di output, ad es.:
```
set output "myfigures.mf"
```

- Creare le proprie immagini. Ogni immagine genererà un carattere diverso. La sua dimensione predefinita sarà 5*3 pollici. È possibile cambiare la dimensione scrivendo **set size 0.5,0.5** o qualsiasi frazione della dimensione predefinita che si desidera avere.

- Chiudere **gnuplot**.

- Generare un file TFM e GF eseguendo METAFONT sull'output di **gnuplot**. Poiché l'immagine è piuttosto grande (5*3 in), sarà necessario utilizzare una versione di METAFONT che abbia un valore di almeno 150000 per memmax. Sui sistemi Unix questi sono istallati per convenzione sotto il nome di bigmf. Per quanto segue si supponga che il comando virmf stia per una grande versione di METAFONT. Per esempio:

- Invocare METAFONT:
```
  virmf '&plain'
```

- Selezionare il dispositivo di output: Al prompt ('*') di METAFONT, digitare:
```
  \mode:=CanonCX;     % o qualsiasi stampante venga utilizzata
```

- Selezionare eventualmente un ingrandimento:
```
  mag:=1;             % o qualsiasi si desidera
```

- Inserire il file **gnuplot**:
```
  input myfigures.mf
```

Su una tipica macchina Unix ci sarà solitamente uno script chiamato "mf" che esegue virmf '&plain', quindi probabilmente è possibile sostituire mf con virmf &plain. Ciò genererà due file: mfput.tfm e mfput.$$$gf (dove $$$ indica la risoluzione del proprio dispositivo). Quanto sopra può essere comodamente ottenuto digitando tutto sulla linea di comando, ad es.: virmf '&plain' '\mode:=CanonCX; mag:=1; input myfigures.mf' In questo caso i file di output saranno chiamati myfigures.tfm e myfigures.300gf.

- Generare un file PK dal file GF usando gftopk:

```
gftopk myfigures.300gf myfigures.300pk
```

Il nome del file di output per gftopk dipende dal driver DVI che si sta usando. Chiedere al proprio amministratore TeX locale riguardo alle convenzioni di denominazione. In seguito, installare i file TFM e PK nelle directory appropriate o impostare correttamente le proprie variabili d'ambiente. Di solito questo comporta impostare TEXFONTS per includere la directory corrente e fare la stessa cosa per la variabile d'ambiente che il proprio driver DVI adopera (nessun nome standard qui...). Questo passo è necessario affinché TeX trovi il file font metric e il proprio driver DVI trovi il file PK.

- Per includere le proprie immagini nel proprio documento si deve dire a TeX il font:

```
\font\gnufigs=myfigures
```

Ogni immagine che è stata realizzata viene memorizzata in un singolo carattere. La prima immagine è il character 0, la seconda è il character 1, e così via... Dopo aver eseguito il passaggio precedente, sarà possibile utilizzare le immagini proprio come qualsiasi altro carattere. Quindi, per posizionare le immagini 1 e 2 al centro del proprio documento, tutto ciò che si deve fare è:

```
\centerline{\gnufigs\char0}
\centerline{\gnufigs\char1}
```

in plain TeX. Per LaTeX è possibile, naturalmente, utilizzare l'ambiente immagine e posizionare l'immagine dove si desidera utilizzando le macro \makebox e \put.

Questa conversione permette di risparmiare molto tempo una volta generato il font; TeX gestisce le immagini come caratteri e usa un tempo minimo per posizionarle, e i documenti che vengono creati cambiano più spesso delle immagini. Si risparmia inoltre molta memoria di TeX. Un ultimo vantaggio di utilizzare il driver METAFONT consiste nel fatto che il file DVI rimane davvero indipendente dal dispositivo, perché non vengono utilizzati comandi \special come nei driver eepic e tpic.

Mif

Nota: Legacy terminal. Il driver del terminale **mif** produce il formato Frame Maker MIF versione 3.00. Plotta in MIF Frames con la dimensione 15*10 cm, e le primitive di un grafico con la stessa penna saranno raggruppate nello stesso gruppo MIF. Le primitive di un grafico in una pagina **gnuplot** saranno plottate in un MIF Frame, e diversi MIF Frame sono raccolti in un grande MIF Frame. Il font MIF usato per il testo è "Times".

Diverse opzioni possono essere impostate nel driver MIF 3.00.

Sintassi:

```
set terminal mif {color | colour | monochrome} {polyline | vectors}
                 {help | ?}
```

colour plotta le linee con tipi di linea >= 0 a colori (MIF sep. 2–7) e **monochrome** plotta tutti i tipi di linea in nero (MIF sep. 0). **polyline** plotta le curve come curve continue e **vectors** plotta le curve come collezioni di vettori. **help** e **?** stampano l'aiuto online sull'output di errore standard — entrambi stampano una breve descrizione dell'utilizzo; **help** elenca anche le opzioni.

Esempi:

```
set term mif colour polylines    # defaults
set term mif                     # defaults
set term mif vectors
set term mif help
```

Mp

Il driver **mp** produce un output destinato ad essere inserito nel programma Metapost. L'esecuzione di Metapost sul file crea file EPS che contengono i grafici. Per default, Metapost passa tutto il testo attraverso

TeX. Ciò comporta il vantaggio di poter utilizzare essenzialmente qualsiasi simbolo TeX nei titoli e nelle etichette.

Sintassi:

```
set term mp {color | colour | monochrome}
            {solid | dashed}
            {notex | tex | latex}
            {magnification <magsize>}
            {psnfss | psnfss-version7 | nopsnfss}
            {prologues <value>}
            {a4paper}
            {amstex}
            {"<fontname> {,<fontsize>}"}
```

L'opzione **color** fa sì che le linee siano disegnate a colori (su una stampante o uno schermo che lo supporti), **monochrome** (o nulla) seleziona linee nere. L'opzione **solid** disegna linee continue, mentre **dashed** (o nulla) seleziona linee con diversi pattern di trattini. Se viene selezionato **solid** ma non **color**, quasi tutte le linee saranno identiche. Questo può essere utile a volte, quindi è consentito.

L'opzione **notex** bypassa completamente Tex, quindi nessun codice TeX può essere usato in etichette sotto questa opzione. Questa è destinata all'uso su vecchi plot file o file che usano spesso caratteri comuni come **$** e **%** che richiedono un trattamento speciale in TeX.

L'opzione **tex** imposta il terminale in modo che emetta il suo testo affinché TeX lo elabori.

L'opzione **latex** imposta il terminale in modo che emetta il suo testo affinché venga elaborato da LaTeX. Questo permette cose come \frac per le frazioni che LaTeX conosce ma TeX no. Si noti che è necessario impostare la variabile d'ambiente TEX al nome del proprio file eseguibile LaTeX (normalmente latex) se si usa questa opzione o si usa **mpost –tex=<name of LaTeX executable> ...**. Altrimenti metapost cercherà di usare TeX per elaborare il testo e non funzionerà.

Cambiare le dimensioni del font in TeX non ha effetto sulla dimensione dei calcoli matematici, e non esiste un metodo infallibile per apportare tale modifica, se non impostando globalmente un fattore di ingrandimento. Questo è lo scopo dell'opzione **magnification**. Deve essere seguito da un fattore di scala. Tutto il testo (NON i grafici (graph)) sarà ridimensionato da questo fattore. Usare questa opzione se sono presenti calcoli matematici che si vogliono in una dimensione diversa da quella predefinita di 10pt. Sfortunatamente tutti i calcoli matematici avranno la stessa dimensione, ma si veda la discussione che segue riguardo a come editare l'output di MP. **mag** funzionerà anche sotto **notex** ma non sembra necessario usarla dato che l'opzione della dimensione del font (di seguito) funziona altrettanto bene.

L'opzione **psnfss** utilizza font postscript in combinazione con LaTeX. Poiché questa opzione è utile solo se si sta usando LaTeX, l'opzione **latex** viene selezionata automaticamente. Questa opzione include i seguenti pacchetti per LaTeX: inputenc(latin1), fontenc(T1), mathptmx, helvet(scaled=09.2), courier, latexsym e textcomp.

L'opzione **psnfss-version7** utilizza anche font postscript in LaTeX (anche l'opzione **latex** è selezionata automaticamente), ma utilizza i seguenti pacchetti con LaTeX: inputenc(latin1), fontenc(T1), times, mathptmx, helvet e courier.

L'opzione **nopsnfss** è il default e usa il font standard (cmr10 se non specificato altrimenti).

L'opzione **prologues** prende un valore come un argomento aggiuntivo e aggiunge, la linea **prologues:=<value>** al file metapost. Se è specificato un valore di **2** metapost usa dei font postscript per generare il eps-file, in modo che il risultato possa essere visualizzato usando ad es. ghostscript. Normalmente l'output di metapost usa dei font di TeX e quindi deve essere incluso in un file (La)TeX prima di poterlo guardare.

L'opzione **noprologues** è il default. Nessuna linea aggiuntiva che specifichi il prologo sarà aggiunta.

L'opzione **a4paper** aggiunge un **[a4paper]** al documentclass. Normalmente si utilizza letter paper (default). Poiché questa opzione è utilizzata solo quando si usa LaTeX, l'opzione **latex** è selezionata automaticamente.

L'opzione **amstex** selesiona automaticamente l'opzione **latex** e include i seguenti pacchetti LaTeX: amsfonts, amsmath(intlimits). Per default questi pacchetti non sono inclusi.

Un nome tra virgolette seleziona il font che sarà usato quando non viene dato un font esplicito in un **set label** o **set title**. Deve essere usato un nome riconosciuto da TeX (esiste un file TFM). Il default è `"cmr10"` a meno che non sia selezionato **notex**, allora è `"pcrr8r"` (Courier). Anche sotto **notex**, un file TFM è necessario per Metapost. Il file **pcrr8r.tfm** è il nome dato a Courier nel pacchetto psnfss di LaTeX. Se si cambia il font dal default **notex**, occorre scegliere un font che coincida con la codifica ASCII almeno nell'intervallo 32-126. **cmtt10** sembra adatto, ma ha un carattere non vuoto in posizione 32 (spazio).

La dimensione può essere qualsiasi numero tra 5.0 e 99.99. Se omessa, viene usato 10.0. È consigliabile usare le dimensioni **magstep**: 10 volte una potenza intera o semintera di 1.2, arrotondata a due decimali, perché queste sono le dimensioni più disponibili dei font nei sistemi TeX.

Tutte le opzioni sono facoltative. Se vengono date informazioni sul font, devono essere alla fine, con la dimensione (se presente) per ultima. La dimensione è necessaria per selezionare una dimensione per il font, anche se il nome del font include informazioni sulla dimensione. Per esempio, **set term mp "cmtt12"** seleziona cmtt12 rimpicciolito alla dimensione predefinita 10. Questo probabilmente non è quello che si desidera, altrimenti sarebbe stato usato cmtt10.

I seguenti caratteri ascii comuni hanno bisogno di un trattamento speciale in TeX:
`$, &, #, %, _;  |, <, >;  ^, ~,  \, {, e }`

I cinque caratteri \$, #, &, _, e % possono essere semplicemente evasi, ad es., **$**. I tre caratteri <, >, e | i caratteri possono essere racchiusi (wrap) in modalità matematica, ad es., **$**<**$**. Il resto richiede alcuni aggiustamenti di TeX. Qualsiasi buon libro su TeX darà alcune indicazioni.

Se si scrivono le etichette all'interno di doppie virgolette, i backslash nel codice TeX devono essere evasi (raddoppiati). Si può evitare questa operazione usando le virgolette singole ma poi non sarà possibile usare **n** per le interruzioni di linea. Al momento della stesura di questo manuale, la versione 3.7 di gnuplot elabora i titoli dati in un comando **plot** in modo diverso che in altri punti, e i backslash nei comandi TeX devono essere raddoppiati indipendentemente dallo stile delle virgolette.

Le immagini metapost sono tipicamente usate nei documenti TeX. Metapost gestisce i font più o meno nello stesso modo in cui lo fa TeX, che è diverso dalla maggior parte degli altri programmi di preparazione di documenti. Se l'immagine è inclusa in un documento LaTeX usando il pacchetto grafico, o in un documento plainTeX tramite epsf.tex, e poi convertita in PostScript con dvips (o un altro convertitore dvi-to-ps), il testo nel grafico sarà solitamente gestito correttamente. Tuttavia, il testo potrebbe non apparire se si invia l'output di Metapost così come è a un interprete PostScript.

Istruzioni Metapost

- Impostare il proprio terminale su Metapost, ad es.:
```
set terminal mp mono "cmtt12" 12
```

- Selezionare un file di output, ad es.:
```
set output "figure.mp"
```

- Creare le proprie immagini. Ogni grafico (o gruppo multiplot) genererà un gruppo Metapost beginfig...endfig separato. La sua dimensione predefinita sarà 5 per 3 pollici. È possibile cambiare la dimensione scrivendo **set size 0.5,0.5** o qualsiasi frazione della dimensione predefinita che si desidera avere.

- Chiudere gnuplot.

- Generare file EPS eseguendo Metapost sull'output di gnuplot:
```
mpost figure.mp  0  mp figure.mp
```

Il nome del programma Metapost dipende dal sistema, tipicamente **mpost** per una macchina Unix e **mp** su molte altre. Metapost genererà un file EPS per ogni immagine.

- Per includere le proprie immagini nel proprio documento si può usare il pacchetto grafico in LaTeX o epsf.tex in plainTeX:

```
\usepackage{graphics} % LaTeX
\input epsf.tex       % plainTeX
```

Se si utilizza un driver diverso da dvips per convertire l'output TeX DVI in PS, potrebbe essere necessario aggiungere la seguente linea nel proprio documento LaTeX:

```
\DeclareGraphicsRule{*}{eps}{*}{}
```

Ogni immagine che è stata realizzata si trova in un file separato. La prima immagine è in, ad es., figure.0, la seconda in figure.1, e così via... Per mettere la terza immagine nel proprio documento, per esempio, tutto quello che si deve fare è:

```
\includegraphics{figure.2} % LaTeX
\epsfbox{figure.2}         % plainTeX
```

Il vantaggio, se esiste, del terminale mp rispetto a un terminale postscript è l'output modificabile. È stato compiuto uno sforzo considerevole per rendere questo output il più chiaro possibile. Per coloro che sono esperti del linguaggio Metapost, i tipi di linea e i colori predefiniti possono essere cambiati modificando gli array **lt**[] e **col**[]. La scelta di linee continue o tratteggiate e di linee colorate o nere può essere cambiata modificando i valori assegnati alle booleane **dashedlines** e **colorlines**. Se l'opzione predefinita **tex** era attiva, i cambiamenti globali al testo delle etichette possono essere effettuati editando il blocco **vebatimtex...etex**. In particolare, si può aggiungere un preambolo LaTeX, se lo si desidera, e quindi si possono usare i comandi di modifica delle dimensioni incorporati di LaTeX per la massima flessibilità. Bisogna assicurarsi di impostare la variabile di configurazione MP appropriata per forzare Metapost ad eseguire LaTeX invece di plainTeX.

Pbm

Nota: disponibile solo se gnuplot è configurato con l'opzione –with-bitmap-terminals. Sintassi:

```
set terminal pbm {<fontsize>} {<mode>} {size <x>,<y>}
```

dove <fontsize> è **small**, **medium**, o **large** e <mode> è **monochrome**, **gray** o **color**. La dimensione predefinita del grafico è 640 pixel di larghezza e 480 pixel di altezza. La dimensione dell'output è riempita di spazi bianchi fino al più vicino multiplo di 8 pixel sia su x che su y. Questo spazio vuoto può essere rimosso in seguito, se necessario.

L'output del driver **pbm** dipende da <mode>: **monochrome** produce un bitmap portatile (un bit per pixel), **gray** un graymap portatile (tre bit per pixel) e **color** un pixmap portatile (colore, quattro bit per pixel).

L'output di questo driver può essere utilizzato con varie utility di conversione e manipolazione delle immagini fornite da NETPBM. Basato sul pacchetto PBMPLUS di Jef Poskanzer, NETPBM fornisce programmi per convertire i suddetti formati PBM in bitmap GIF, TIFF, MacPaint, Macintosh PICT, PCX, X11 e molti altri. Informazioni complete sono disponibili su http://netpbm.sourceforge.net/.

Esempi:

```
set terminal pbm small monochrome                # default
set terminal pbm color medium size 800,600
set output '| pnmrotate 45 | pnmtopng > tilted.png'  # usa NETPBM
```

Pcl5

Il driver **pcl5** supporta le stampanti PCL5e/PCL5c. Usa (principalmente) il formato vettoriale HP-GL/2.

Sintassi:

```
set terminal pcl5 {<mode>} {{no}enhanced}
    {size <plotsize> | size <width>{unit},<height>{unit}}
    {font "<fontname>,<size>"} {pspoints | nopspoints}
    {fontscale <scale>} {pointsize <scale>} {linewidth <scale}
    {rounded|butt} {color <number_of_pens>}
```

<mode> è **landscape** o **portrait**. <plotsize> è la dimensione fisica di plotting del grafico, che può essere uno dei seguenti formati: **letter** per i display standard (8 1/2" X 11"), **legal** per i display (8 1/2" X 14"), **noextended** per i display (36" X 48") (un rapporto di dimensioni letter), **extended** per i display (36" X 55") (quasi un rapporto di dimensioni legal), o **a4** per i display (296mm X 210mm). Inoltre è possibile specificare esplicitamente la dimensione del canvas usando le opzioni **width** e **height**. L'unità predefinita è **in** (pollici). La dimensione predefinita è **letter**.

<fontname> può essere uno tra stick, univers (default), albertus, antique_olive, arial, avant_garde_gothic, bookman, zapf_chancery, clarendon, coronet, courier, courier_ps, cg_times, garamond_antigua, helvetica, helvetica_narrow, letter_gothic, marigold, new_century_schlbk, cg_omega, palatino, times_new_roman, times_roman, zapf_dingbats, truetype_symbols, o wingdings. I nomi dei font non sono sensibili alle maiuscole e gli underscore possono essere sostituiti da spazi o trattini o possono essere omessi. <fontsize> è la dimensione del font in punti.

La selezione del tipo di punto può essere un default limitato impostato specificando **nopspoints**, o lo stesso set di tipi di punti fornito dal terminale postscript specificando **pspoints** (default).

L'opzione **butt** seleziona linee con estremità butt e giunzioni mitered (default), mentre **rounded** seleziona estremità e giunzioni della linea arrotondate.

Lo spessore delle linee e le dimensioni dei punti e dei font possono essere ridimensionati usando rispettivamente le opzioni **linewidth**, **pointscale**, o **fontscale**. **color** seleziona il numero di penne <number_of_pens> usate nei grafici. Il default è 8, il minimo è 2.

Si noti che il supporto integrato di alcune di queste opzioni dipende dalla stampante. Per esempio, tutti i font sono presumibilmente supportati dalla HP Laserjet IV, ma solo alcuni (ad es. univers, stick) possono essere supportati dalla HP Laserjet III e dalla Designjet 750C. Inoltre, il colore ovviamente non funzionerà su dispositivi monocromatici, ma quelli più recenti potranno fare la scala di grigio. Default: landscape, a4, 8 penne, univers, 12 punti, pspoints, butt, no scaling

Il terminale **pcl5** cercherà di richiedere i font che corrispondono all'**encoding** dell'utente. Si noti che ciò ha la massima priorità, quindi si potrebbe avere una variante di font (font face) diversa. L'**encoding** predefinito del terminale è **HP Roman-8**.

Limitazioni:

Questo terminale non supporta la trasparenza alpha. Il riempimento trasparente è emulato utilizzando modelli di ombreggiatura. Il testo racchiuso in un riquadro (boxed text) non è implementato.

Il supporto per UTF-8 è limitato. In mancanza della modalità label (label mode) per l'output UTF-8 in HP-GL/2, il driver ritorna al PCL per le stringhe che contengono caratteri a 8 bit. Il testo UTF-8 è limitato agli angoli di 0, 90, 180 e 270 gradi. Inoltre l'allineamento verticale potrebbe essere sbagliato a seconda del font.

Alcune funzionalità di testo avanzato (phantom box, overprinting) richiedono l'uso di funzionalità PCL in aggiunta a HP-GL/2. Questo è conforme alle specifiche ma potrebbe non funzionare con la propria stampante o software.

Pdf

[DEPRECATO] Questo terminale usa la libreria non-free PDFlib (GmbH Munchen) per produrre file in Portable Document Format. A meno che non si disponga di una licenza commerciale per PDFlib e si abbia bisogno di qualche caratteristica speciale che fornisce, sarebbe meglio usare invece il terminale pdf cairo. Gnuplot può anche esportare file PDF da sessioni di terminale interattivo wxt o qt.

Sintassi:

```
set terminal pdf {monochrome|color|colour}
                 {{no}enhanced}
                 {fname "<font>"} {fsize <fontsize>}
                 {font "<fontname>{,<fontsize>}"} {fontscale <scale>}
                 {linewidth <lw>} {rounded|butt}
```

```
{dl <dashlength>}}
{size <XX>{unit},<YY>{unit}}
```

L'impostazione predefinita prevede l'utilizzo di un colore diverso per ogni tipo di linea. Selezionando **monochome** si utilizzerà il nero per tutti i tipi di linea. Anche in modalità mono si possono comunque usare colori espliciti per aree riempite o stili di linea.

dove <font> è il nome del font predefinito da usare (default Helvetica) e <fontsize> è la dimensione del font (in punti, default 12). Per aiuto su quali font sono disponibili o su come installarne di nuovi, vedere la documentazione della propria installazione locale di pdflib.

L'opzione **enhanced** abilita le funzionalità di elaborazione del testo avanzato (pedici, indici e font misti). Vedere **enhanced (p. 35)**.

Lo spessore di tutte le linee del grafico può essere aumentato tramite il fattore <n> specificato in **linewidth**. Analogalmente, **dashlength** è un moltiplicatore per la spaziatura predefinita dei trattini.

rounded imposta le estremità delle linee e le giunzioni delle linee affinché siano arrotondate; **butt** è il default, estremità butt e giunzioni mitered.

La dimensione predefinita per l'output PDF è 5 pollici per 3 pollici. L'opzione **size** la modifica in qualsiasi dimensione richiesta dell'utente. Per default le dimensioni X e Y sono considerate in pollici, ma sono possibili altre unità (attualmente solo cm).

Pdfcairo

Il terminale **pdfcairo** genera output in pdf. Il disegno viene fatto tramite cairo, una libreria grafica 2D, e pango, una libreria per l'impaginazione e il rendering del testo.

Sintassi:

```
set term pdfcairo
             {{no}enhanced} {mono|color}
             {font <font>} {fontscale <scale>}
             {linewidth <lw>} {rounded|butt|square} {dashlength <dl>}
             {background <rgbcolor>}
             {size <XX>{unit},<YY>{unit}}
```

Questo terminale supporta una modalità di testo avanzato, che permette ai font e ad altri comandi di formattazione (pedici, apici, ecc.) di essere incorporati in etichette e altre stringhe di testo. La sintassi della modalità di testo avanzato è condivisa con altri tipi di terminale di gnuplot. Vedere **enhanced (p. 35)** per maggiori dettagli.

Lo spessore di tutte le linee del grafico può essere modificato dal fattore <lw> specificato in **linewidth**. Lo spessore di linea predefinito è di 0.5 punti. (1 punto "PostScript" = 1/72 pollice = 0.353 mm)

rounded imposta le estremità delle linee (line caps) e le giunzioni delle linee (line joins) affinchè siano arrotondate; **butt** è il default, estremità butt e giunzioni mitered (squadrate).

La dimensione predefinita dell'output è 5 pollici x 3 pollici. L'opzione **size** la modifica in qualsiasi dimensione richiesta dell'utente. Per default le dimensioni X e Y sono considerate in pollici, ma sono possibili altre unità (attualmente solo cm). Le coordinate dello schermo vanno sempre da 0.0 a 1.0 per tutta la lunghezza dei bordi del grafico, come specificato dall'opzione **size**.

<font> è nel formato "FontFace,FontSize", cioè la variante (face) e la dimensione separate da una virgola in una singola stringa. FontFace è un normale nome di variante di font, come 'Arial'. Se non si fornisce FontFace, il terminale pdfcairo userà 'Sans'. FontSize è la dimensione del font, in punti. Se non viene fornita, il terminale pdfcairo userà una dimensione nominale del font di 12 punti. Tuttavia, il parametro fontscale predefinito per questo terminale è 0.5, quindi la dimensione apparente del font è più piccola se l'output del pdf è visto a pieno formato.

Per esempio :

```
set term pdfcairo font "Arial,12"
set term pdfcairo font "Arial" # per cambiare solo la variante del font
set term pdfcairo font ",12" # per cambiare solo la dimensione del font
set term pdfcairo font "" # per reimpostare il nome e la dimensione del font
```

I font sono recuperati dai soliti sottosistemi di font. In Windows, questi font devono essere trovati e configurati nella entry "Fonts" del pannello di controllo. In UNIX, sono gestiti da "fontconfig".

Pango, la libreria usata per impaginare il testo, si basa su utf-8. Quindi, il terminale pdfcairo deve convertire dalla codifica dell'utente a utf-8. La codifica dell'input predefinita è basata sul proprio 'locale'. Se si desidera usare un'altra codifica, è necessario assicurarsi che gnuplot sappia quale si sta usando. Vedere **encoding (p. 154)** per ulteriori dettagli.

Pango può dare risultati inaspettati con i font che non rispettano la mappatura unicode. Con il font Symbol, per esempio, il terminale pdfcairo userà la mappa fornita da http://www.unicode.org/ per tradurre i codici dei caratteri in unicode. Notare che "il font Symbol" deve essere inteso come il font Symbol di Adobe, distribuito con Acrobat Reader come "SY_____.PFB". In alternativa, il font OpenSymbol, distribuito con OpenOffice.org come "opens___.ttf", offre gli stessi caratteri. Microsoft ha distribuito un font Symbol ("symbol.ttf"), ma ha un set di caratteri diversi con vari caratteri matematici mancanti o spostati. Se si verificano problemi con la configurazione predefinita (se la demo enhancedtext.dem non viene visualizzata correttamente, per esempio), probabilmente è necessario installare uno dei font Symbol di Adobe o OpenOffice, e rimuovere quello di Microsoft. È stato osservato che altri font non conformi, come "wingdings", funzionano.

Il rendering del grafico non può ancora essere modificato. Per ottenere il miglior output possibile, il rendering coinvolge due meccanismi: antialiasing e oversampling. L'antialiasing permette di visualizzare le linee non orizzontali e non verticali in modo più fluido. L'oversampling abbinato all'antialiasing fornisce una precisione subpixel, in modo che gnuplot possa disegnare una linea da coordinate non intere. Questo evita effetti di oscillazione (wobbling) sulle linee diagonali ('plot x', per esempio).

Pict2e

Il terminale **pict2e** utilizza la variante LaTeX2e dell'ambiente immagine. Sostituisce i terminali che erano basati sull'ambiente immagine originale di LaTeX: **latex**, **emtex**, **tpic**, ed **eepic**. (SPERIMENTALE)

tikz, **pstricks**, **cairolatex**, **pslatex**, **epslatex** e **mp** sono delle alternative a questo terminale con un supporto più completo delle funzionalità di gnuplot.

Sintassi:

```
set terminal pict2e
             {font "{<fontname>}{,<fontsize>}"}
             {size <XX>{unit}, <YY>{unit}}
             {color | monochrome}
             {linewidth <lw>} {rounded | butt}
             {texarrows | gparrows} {texpoints | gppoints}
             {smallpoints | tinypoints | normalpoints}
```

Questo terminale richiede i seguenti pacchetti standard di LaTeX: **pict2e**, **xcolor**, **graphics/graphicx** e **amssymb**. Per pdflatex, il pacchetto **transparent** è usato per supportare la trasparenza.

Per default il grafico erediterà le impostazioni del font dal documento incorporato (embedding). È possibile forzare un font con l'opzione **font**, come cmtt (Courier) o cmr (Roman), invece. In questo caso si può anche forzare uno specifico fontsize. Altrimenti l'argomento fontsize è usato per stimare lo spazio richiesto per il testo. A meno che il proprio driver non sia in grado di costruire font di qualsiasi dimensione (ad es. dvips), attenersi alle dimensioni standard di 10, 11 e 12 punti.

La dimensione predefinita per il grafico è 5 pollici per 3 pollici. L'opzione **size** la modifica in qualsiasi dimensione richiesta dell'utente. Per default le dimensioni X e Y sono considerate in pollici, ma sono possibili altre unità (attualmente solo cm).

Con **texpoints**, i punti sono disegnati usando comandi LaTeX come "\Diamond" e "\Box". Questi sono forniti dal pacchetto latexsym, che fa parte della distribuzione base e quindi di qualsiasi implementazione di LaTeX. Altri tipi di punti usano simboli del pacchetto amssymb. Con **gppoints**, il terminale userà invece le routine interne di gnuplot per disegnare i simboli dei punti.

Con l'opzione **texpoints**, è possibile selezionare tre diverse dimensioni del punto: **normalpoints**, **smallpoints**, e **tinypoints**.

color fa sì che gnuplot generi comandi \color{...} in modo che i grafici (graph) siano colorati. Usando questa opzione, è necessario includere \usepackage{xcolor} nel preambolo del proprio documento LaTeX. **monochrome** eviterà di usare qualsiasi comando color nell'output. Il riempimento di colore trasparente è disponibile se si usa pdflatex.

linewidth imposta il fattore di scala per lo spessore delle linee. **rounded** imposta le estremità delle linee e le giunzioni delle linee affinché siano arrotondate; **butt** imposta estremità butt e giunzioni mitered ed è il default.

pict2e supporta solo le linee punteggiate (dotted), ma non le linee tratteggiate (dashed). Tutti i tipi di linea predefiniti sono continui. Usare **set linetype** con la proprietà **dashtype** per modificare.

texarrows disegna **arrow** (frecce) usando i comandi LaTeX che sono più corti ma non offrono tutte le opzioni. **gparrows** seleziona invece di disegnare le frecce usando la routine propria di gnuplot per la piena funzionalità.

Pm

Il driver del terminale **pm** fornisce una finestra di OS/2 Presentation Manager in cui viene plottato il grafico (graph). La finestra si apre quando viene plottato il primo grafico (graph). Questa finestra ha il proprio aiuto online, oltre a installazioni per stampare e copiare negli appunti.

Sintassi:

```
set terminal pm      {{server} {n} | noserver}
                     {nopersist | persist}
                     {enhanced | noenhanced}
                     {font <fontspec>}
                     {nowidelines | widelines}
                     {fontscale <scale>}
                     {linewidth <scale>}
                     {pointscale <scale>}
                     {{title} "title"}
```

Se viene specificato **persist**, ogni grafico (graph) appare nella propria finestra e tutte le finestre rimangono aperte dopo l'uscita di **gnuplot**. Se viene specificato **server**, tutti i grafici (graph) appaiono nella stessa finestra, che rimane aperta quando **gnuplot** viene chiuso. Questa opzione prende un argomento numerico opzionale che specifica un'istanza del processo del server. Perciò più finestre del server possono essere utilizzate allo stesso tempo.

Se viene specificato **widelines**, tutti i grafici saranno disegnati con linee larghe. Se viene specificato **enhanced**, sono abilitati i pedici, gli apici e vari font (vedere **enhanced text (p. 35)** per i dettagli). I nomi dei font per i font PostScript di base possono essere abbreviati a una sola lettera (T/H/C/S per Times/Helvetica/Courier/Symbol).

linewidth, **fontscale**, **pointscale** possono essere usati per modificare lo spessore delle linee, la dimensione del testo, o la dimensione dei simboli dei punti.

Se viene specificato **title**, sarà usato come titolo della finestra del grafico. Sarà anche usato come nome dell'istanza del server, e sovrascriverà l'argomento numerico opzionale.

Il driver outboard di gnuplot, gnupmdrv.exe, viene cercato nella stessa directory di gnuplot stesso. È possibile sovrascrivere ciò definendo una delle variabili d'ambiente GNUPLOT_DRIVER_DIR o GNUPLOT. Come

ultima risorsa si cerca di individuare gnupmdrv.exe nella directory corrente e nel PATH.

Png

Sintassi:

```
set terminal png
       {{no}enhanced}
       {{no}transparent} {{no}interlace}
       {{no}truecolor} {rounded|butt}
       {linewidth <lw>} {dashlength <dl>}
       {tiny | small | medium | large | giant}
       {font "<face> {,<pointsize>}"} {fontscale <scale>}
       {size <x>,<y>} {{no}crop}
       {background <rgb_color>}
```

Le immagini PNG, JPEG e GIF sono create usando la libreria esterna libgd. I grafici PNG possono essere visualizzati interattivamente inviando (piping) l'output al programma 'display' dal pacchetto ImageMagick come segue:

```
set term png
set output '| display png:-'
```

È possibile visualizzare l'output dei comandi plot successivi in modo interattivo digitando <space> nella finestra di visualizzazione. Per salvare il grafico corrente in un file, bisogna cliccare con il tasto sinistro del mouse nella finestra di visualizzazione e scegliere **save**.

transparent ordina al driver di rendere il colore di sfondo trasparente. Il default è **notransparent**.

interlace ordina al driver di generare PNG interlacciati. Il default è **nointerlace**.

Le opzioni **linewidth** e **dashlength** sono fattori di scala che influenzano tutte le linee disegnate, cioè vengono moltiplicati per i valori richiesti nei vari comandi di disegno.

Per default le immagini png in output usano 256 colori indicizzati. L'opzione **truecolor** crea invece immagini TrueColor con 24 bit di informazioni di colore per pixel. Gli stili di riempimento trasparenti richiedono l'opzione **truecolor**. Vedere **fillstyle (p. 209)**. Uno sfondo trasparente è possibile sia nelle immagini indicizzate che in quelle TrueColor.

butt ordina al driver di usare un metodo di disegno della linea che non superi il punto finale di una linea desiderato. Questa impostazione è applicabile solo per spessori di linea maggiori di 1. Questa impostazione è particolarmente utile quando si disegnano linee orizzontali o verticali. Il default è **rounded**.

I dettagli della selezione del font sono complicati. Due semplici esempi equivalenti sono dati di seguito:

```
set term png font arial 11
set term png font "arial,11"
```

Per maggiori informazioni, vedere la sezione separata sotto **font (p. 46)**.

La dimensione del grafico in output <x,y> è data in pixel — il defaults è 640x480. Vedere informazioni aggiuntive sotto **canvas (p. 32)** e **set size (p. 205)**. Lo spazio vuoto ai bordi del grafico completato può essere tagliato usando l'opzione **crop**, ottenendo una dimensione finale dell'immagine più piccola. Il default è **nocrop**.

Esempi

```
set terminal png medium size 640,480 background '#ffffff'
```

Usa il font incorporato medium size, non scalabile e non ruotabile. Usa il bianco (24-bit RGB in esadecimale) per lo sfondo non trasparente.

```
set terminal png font arial 14 size 800,600
```

Cerca un font scalabile con il nome della variante (face) 'arial' e imposta la dimensione del font a 14pt. Vedere **fonts (p. 46)** per dettagli su come viene fatta la ricerca dei font.

```
set terminal png transparent truecolor enhanced
```

Usa 24 bit di informazioni di colore per pixel, con uno sfondo trasparente. Usa la modalità **enhanced text** (testo avanzato) per controllare il layout il layout delle stringhe da stampare.

Pngcairo

Il terminale **pngcairo** genera output in png. Il disegno viene fatto tramite cairo, una libreria grafica 2D, e pango, una libreria per l'impaginazione e il rendering del testo.

Sintassi:

```
set term pngcairo
            {{no}enhanced} {mono|color}
            {{no}transparent} {{no}crop} {background <rgbcolor>
            {font <font>} {fontscale <scale>}
            {linewidth <lw>} {rounded|butt|square} {dashlength <dl>}
            {pointscale <ps>}
            {size <XX>{unit},<YY>{unit}}
```

Questo terminale supporta una modalità di testo avanzato, che permette ai font e ad altri comandi di formattazione (pedici, apici, ecc.) di essere incorporati in etichette e altre stringhe di testo. La sintassi della modalità di testo avanzato è condivisa con altri tipi di terminale di gnuplot. Vedere **enhanced (p. 35)** per maggiori dettagli.

Lo spessore di tutte le linee del grafico può essere modificato dal fattore <lw>.

rounded imposta le estremità delle linee (line caps) e le giunzioni delle linee (line joins) affinchè siano arrotondate; **butt** è il default, estremità butt e giunzioni mitered (squadrate).

La dimensione predefinita dell'output è 640 x 480 pixel. L'opzione **size** la modifica in qualsiasi dimensione richiesta dell'utente. Per default le dimensioni X e Y sono considerate in pixel, ma sono possibili altre unità (attualmente cm e pollici). Una dimensione data in centimetri o pollici sarà convertita in pixel assumendo una risoluzione di 72 dpi. Le coordinate dello schermo vanno sempre da 0.0 a 1.0 per tutta la lunghezza dei bordi del grafico, come specificato dall'opzione **size**.

<font> è nel formato "FontFace,FontSize", cioè la variante (face) e la dimensione separate da una virgola in una singola stringa. FontFace è un normale nome di variante di font, come 'Arial'. Se non si fornisce FontFace, il terminale pngcairo userà 'Sans'. FontSize è la dimensione del font, in punti. Se non viene fornita, il terminale pngcairo userà una dimensione del font di 12 punti.

```
Per esempio :
   set term pngcairo font "Arial,12"
   set term pngcairo font "Arial" # to change the font face only
   set term pngcairo font ",12" # to change the font size only
   set term pngcairo font "" # to reset the font name and size
```

I font sono recuperati dai soliti sottosistemi di font. In Windows, questi font devono essere trovati e configurati nella entry "Fonts" del pannello di controllo. In UNIX, sono gestiti da "fontconfig".

Pango, la libreria usata per impaginare il testo, si basa su utf-8. Quindi, il terminale pdfcairo deve convertire dalla codifica dell'utente a utf-8. La codifica dell'input predefinita è basata sul proprio 'locale'. Se si desidera usare un'altra codifica, è necessario assicurarsi che gnuplot sappia quale si sta usando. Vedere **encoding (p. 154)** per ulteriori dettagli.

Pango può dare risultati inaspettati con i font che non rispettano la mappatura unicode. Con il font Symbol, per esempio, il terminale pngcairo userà la mappa fornita da http://www.unicode.org/ per tradurre i codici dei caratteri in unicode. Notare che "il font Symbol" deve essere inteso come il font Symbol di Adobe,

distribuito con Acrobat Reader come "SY_____.PFB". In alternativa, il font OpenSymbol, distribuito con OpenOffice.org come "opens___.ttf", offre gli stessi caratteri. Microsoft ha distribuito un font Symbol ("symbol.ttf"), ma ha un set di caratteri diversi con vari caratteri matematici mancanti o spostati. Se si verificano problemi con la configurazione predefinita (se la demo enhancedtext.dem non viene visualizzata correttamente, per esempio), probabilmente è necessario installare uno dei font Symbol di Adobe o OpenOffice, e rimuovere quello di Microsoft.

Il rendering usa l'oversampling, l'antialiasing, e il font hinting nella misura supportata dalle librerie cairo e pango.

Postscript

Diverse opzioni possono essere impostate nel driver **postscript**.

Sintassi:

```
set terminal postscript {default}
set terminal postscript {landscape | portrait | eps}
                        {enhanced | noenhanced}
                        {defaultplex | simplex | duplex}
                        {fontfile {add | delete} "<filename>"
                         | nofontfiles} {{no}adobeglyphnames}
                        {level1 | leveldefault | level3}
                        {color | colour | monochrome}
                        {background <rgbcolor> | nobackground}
                        {dashlength | dl <DL>}
                        {linewidth | lw <LW>} {pointscale | ps <PS>}
                        {rounded | butt}
                        {clip | noclip}
                        {palfuncparam <samples>{,<maxdeviation>}}
                        {size <XX>{unit},<YY>{unit}}
                        {blacktext | colortext | colourtext}
                        {{font} "fontname{,fontsize}" {<fontsize>}}
                        {fontscale <scale>}
```

Se appare il messaggio di errore

```
"Can't find PostScript prologue file ... "
```

Si prega di consultare e seguire le istruzioni nel **postscript prologue**.

landscape e **portrait** scelgono l'orientamento del grafico. La modalità **eps** genera output EPS (Encapsulated PostScript), che è solo un normale PostScript con alcune linee aggiuntive che permettono al file di essere importato in una varietà di altre applicazioni. (Le linee aggiuntive sono linee di commento di PostScript, quindi il file può ancora essere stampato da solo). Per ottenere output EPS, bisogna usare la modalità **eps** e realizzare un solo grafico per ogni file. In modalità **eps** l'intero grafico, compresi i font, è ridotto a metà della dimensione predefinita

enhanced abilita le funzionalità della modalità di testo avanzato (pedici, apici e font misti). Vedere **enhanced (p. 35)** per maggiori informazioni. **blacktext** costringe tutto il testo ad essere scritto in nero anche in modalità color;

Il duplexing in PostScript è la capacità della stampante di stampare su entrambi i lati dello stesso foglio di carta. Con **defaultplex**, viene utilizzata l'impostazione predefinita della stampante; con **simplex** viene stampato solo un lato; **duplex** stampa su entrambi i lati (ignorato se la propria stampante non è in grado di farlo).

"<**fontname**>" è il nome di un font PostScript valido; e <**fontsize**> è la dimensione del font in punti PostScript. Oltre ai font postscript standard, viene definita una versione obliqua del font Symbol, utile per la matematica. Si chiama "Symbol-Oblique".

default imposta tutte le opzioni ai loro default: **landscape**, **monochrome**, **dl 1.0**, **lw 1.0**, **defaultplex**, **enhanced**, "Helvetica" e 14pt. La dimensione predefinita di un grafico PostScript è 10 pollici di larghezza e 7 pollici di altezza. L'opzione **color** abilita il colore, mentre **monochrome** preferisce elementi di disegno in bianco e nero. Inoltre, **monochrome** usa la **palette** di grigi ma non cambia il colore degli oggetti specificati con un esplicito **colorspec**. **dashlength** o **dl** ridimensiona la lunghezza dei segmenti di linea tratteggiata per <DL>, che è un numero in virgola mobile maggiore di zero. **linewidth** o **lw** ridimensiona tutti i linewidth (spessori delle linee) per <LW>.

Per default il codice PostScript generato usa le funzionalità del linguaggio che sono state introdotte in PostScript Level 2, in particolare i filtri e il pattern-fill di oggetti irregolari come i filledcurve. Le funzionalità di PostScript Level 2 sono condizionatamente protette in modo che gli interpreti di PostScript Level 1 non emettano errori ma, piuttosto, mostrino un messaggio o un'approssimazione di PostScript Level 1. L'opzione **level1** sostituisce le approssimazioni PostScript Level 1 di queste funzionalità e non usa il codice PostScript Level 2. Questo può essere richiesto da alcune vecchie stampanti e da vecchie versioni di Adobe Illustrator. Il flag **level1** può essere attivato in seguito modificando una singola linea nel file di output PostScript per forzare l'interpretazione PostScript Level 1. Nel caso di file contenenti codice di livello 2, le funzionalità di cui sopra non appariranno o saranno sostituite da una nota quando questo flag è impostato o quando il programma di interpretazione non indica che comprende PostScript di livello 2 o superiore. Il flag **level3** abilita la codifica PNG per le immagini bitmap, che può ridurre considerevolmente la dimensione dell'output.

rounded imposta le estremità delle linee e le giunzioni delle linee affinché siano arrotondate; **butt** è il default, estremità butt e giunzioni mitered.

clip dice a PostScript di ritagliare tutto l'output nel bounding box (riquadro di delimitazione); **noclip** è il default.

palfuncparam controlla come **set palette functions** sono codificate come gradienti nell'output. Le funzioni analitiche dei componenti del colore (impostate tramite **set palette functions**) sono codificate come gradienti lineari interpolati nell'output postscript: Le funzioni dei componenti del colore sono campionate in <samples> punti e tutti i punti sono rimossi da questo gradiente che può essere rimosso senza cambiare i colori risultanti di più di <maxdeviation> (deviazione massima). Per quasi tutte le palette utili si può tranquillamente lasciare inalterati i valori predefiniti di <samples>=2000 e <maxdeviation>=0.003.

La dimensione predefinita per l'output postscript è 10 pollici x 7 pollici. Il default per l'output eps è 5 x 3.5 pollici. L'opzione **size** la modifica in qualsiasi dimensione richiesta dell'utente. Per default le dimensioni X e Y sono considerate in pollici, ma sono possibili altre unità (attualmente solo cm). La BoundingBox del grafico viene regolata correttamente per contenere l'immagine ridimensionata. Le coordinate dello schermo vanno sempre da 0.0 a 1.0 lungo tutta la lunghezza dei bordi del grafico come specificato dall'opzione **size**. NB: **questo è un cambiamento rispetto al metodo consigliato in precedenza di usare il comando set size prima di impostare il tipo di terminale**. Il vecchio metodo lasciava la BoundingBox invariata e le coordinate dello schermo non corrispondevano ai limiti reali del grafico.

I font elencati da **fontfile** o **fontfile add** racchiudono le definizioni dei font elencati da un file di font postscript Type 1 o TrueType direttamente nel file postscript di output di gnuplot. Quindi, il font racchiuso può essere usato in etichette, titoli, ecc. Vedere la sezione **postscript fontfile (p. 292)** per maggiori dettagli. Con **fontfile delete**, un fontfile viene cancellato dalla lista dei file incorporati. **nofontfiles** svuota la lista dei font incorporati.

Esempi:

```
set terminal postscript default       # old postscript
set terminal postscript enhanced      # old enhpost
set terminal postscript landscape 22  # old psbig
set terminal postscript eps 14        # old epsf1
set terminal postscript eps 22        # old epsf2
set size 0.7,1.4; set term post portrait color "Times-Roman" 14
set term post "VAGRoundedBT_Regular" 14 fontfile "bvrr8a.pfa"
```

Linewidth e pointsize possono essere cambiati con **set style line**.

Il driver **postscript** supporta circa 70 pointtype diversi, selezionabili tramite l'opzione **pointtype** su **plot**

e **set style line**.

Diversi file utili riguardanti il PostScript di **gnuplot** sono inclusi nella sottodirectory /docs/psdoc della distribuzione di **gnuplot** e nei siti della distribuzione. Essi sono "ps_symbols.gpi" (un file di comando **gnuplot** che, quando eseguito, crea il file "ps_symbols.ps" il quale mostra tutti i simboli disponibili attraverso il terminale **postscript**), "ps_guide.ps" (un file PostScript che contiene una sintesi della sintassi migliorata e una pagina che mostra cosa producono i codici ottali con font di testo e di simboli), "ps_file.doc" (un file di testo che contiene una discussione sull'organizzazione di un file PostScript scritto da **gnuplot**), e "ps_fontfile_doc.tex" (un file LaTeX che contiene una breve documentazione riguardante l'incapsulamento dei font LaTeX con una tabella dei glifi dei font matematici).

Un file PostScript è editabile, quindi una volta che **gnuplot** ne ha creato uno, è possibile modificarlo a proprio piacimento. Vedere la sezione **editing postscript (p. 292)** per alcuni suggerimenti.

Editing postscript

Il linguaggio PostScript è un linguaggio molto complesso, troppo complesso per essere descritto dettagliatamente in questo documento. Tuttavia ci sono alcune cose in un file PostScript scritto da **gnuplot** che possono essere cambiate senza il rischio di introdurre errori fatali nel file.

Per esempio, la dichiarazione PostScript "/Color true def" (scritta nel file in risposta al comando **set terminal postscript color**), può essere alterata in modo evidente per generare una versione in bianco e nero di un grafico. Allo stesso modo i colori delle linee, i colori del testo, i pesi delle linee e le dimensioni dei simboli possono essere modificati in maniera semplice. Il testo (titoli ed etichette) può essere editato per correggere gli errori di ortografia o per cambiare i font. Qualsiasi cosa può essere riposizionata, e naturalmente qualsiasi cosa può essere aggiunta o cancellata, ma modifiche come queste potrebbero richiedere una conoscenza più approfondita del linguaggio PostScript.

L'organizzazione di un file PostScript scritto da **gnuplot** è discussa nel file di testo "ps_file.doc" nella sottodirectory docs/ps della distribuzione sorgente di gnuplot.

Postscript fontfile

```
set term postscript ... fontfile {add|delete} <filename>
```

L'opzione **fontfile** o **fontfile add** prende un file name come argomento e incapsula questo file nell'output postscript per rendere questo font disponibile per gli elementi di testo (etichette, segni di tic, titoli, ecc.). L'opzione **fontfile delete** prende a sua volta un file name come argomento. Essa elimina questo file name dalla lista dei file incapsulati.

Il terminale postscript comprende alcuni formati di file di font: i font Type 1 nel formato di file ASCII (estensione ".pfa"), i font Type 1 nel formato di file binario (estensione ".pfb"), e i font TrueType (estensione ".ttf"). I file pfa vengono compresi direttamente, i file pfb e ttf vengono convertiti sul momento se sono installati strumenti di convesione appropriati (vedere di seguito). È necessario specificare il filename completo, compresa l'estensione. Ogni opzione **fontfile** prende esattamente un nome di file di font. Questa opzione può essere usata più volte per includere più di un file di font.

L'ordine di ricerca usato per trovare i file di font è (1) pathname assoluto o directory di lavoro corrente (2) una qualsiasi delle directory specificate da **set loadpath** (3) la directory specificata da **set fontpath** (4) la directory fornita nella variabile d'ambiente GNUPLOT_FONTPATH. NB: Questo è un CAMBIAMENTO rispetto alle versioni precedenti di gnuplot.

Per usare il file di font incapsulato si deve specificare il nome del font (che normalmente non è uguale al nome del file). Quando si incorpora un file di font usando l'opzione **fontfile** in modalità interattiva, il nome del font viene stampato sullo schermo. Ad es.

```
Font file 'p0520041.pfb' contiene il font 'URWPalladioL-Bold'. Posizione:
/usr/lib/X11/fonts/URW/p0520041.pfb
```

Quando si usano font pfa o pfb, è possibile scoprirlo anche guardando nel file del font. È presente una linea simile a "/FontName /URWPalladioL-Bold def". La stringa centrale senza la barra è il fontaname, qui "URWPalladioL-Bold". Per i font TrueType, questo non è così facile poiché il nome del font è memorizzato in un formato binario. Inoltre, spesso hanno degli spazi nei nomi dei font, il che non è supportato dai font Type 1 (in cui un TrueType è convertito al momento). I nomi dei font sono cambiati al fine di eliminare gli spazi nei fontname. Il modo più semplice per scoprire quale nome di font viene generato per l'uso con gnuplot, è quello di avviare gnuplot in modalità interattiva e digitare "set terminal postscript fontfile '<filename.ttf>'".

Per convertire i file di font (sia ttf che pfb) in formato pfa, lo strumento di conversione deve leggere il font da un file e scriverlo sullo standard output. Se l'output non può essere scritto sullo standard output, la conversione in tempo reale non è possibile.

Per i file pfb, "pfbtops" è uno strumento che può farlo. Se questo programma è installato sul proprio sistema, la conversione in tempo reale dovrebbe funzionare. Basta provare a incapsulare un file pfb. Se la chiamata non funziona correttamente, è possibile specificare come questo programma è chiamato definendo la variabile d'ambiente GNUPLOT_PFBTOPFA ad es. "pfbtops %s". Il **%s** sarà sostituito dal nome del file del font e quindi deve esistere nella stringa.

Se non si desidera fare la conversione in tempo reale ma ottenere un file pfa del font si può usare lo strumento "pfb2pfa" che è scritto in simple c e dovrebbe compilare con qualsiasi compilatore c. È disponibile su molti server ftp, ad es. ftp://ftp.dante.de/tex-archive/fonts/utilities/ps2mf/

Infatti, "pfbtopfa" e "pfb2ps" svolgono lo stesso lavoro. "pfbtopfa" colloca il codice pfa risultante in un file, mentre "pfbtops" lo scrive sullo standard output.

I font TrueType sono convertiti nel formato pfa Type 1, per esempio usando lo strumento "ttf2pt1" che è disponibile su http://ttf2pt1.sourceforge.net/

Se la conversione integrata non funziona, il comando di conversione può essere modificato dalla variabile d'ambiente GNUPLOT_TTFTOPFA. Per l'uso con ttf2pt1 può essere impostato a "ttf2pt1 -a -e -W 0 %s -". Anche qui rappresenta il nome del file.

Per scopi speciali è possibile utilizzare anche una pipe (se disponibile per il proprio sistema operativo). Perciò bisogna mettere il carattere "<" all'inizio della definizione del nome del file e aggiungere una chiamata di programma. Questo programma deve scrivere dati pfa sullo standard output. Quindi, è possibile accedere a un file pfa tramite **set fontfile "< cat garamond.pfa"**.

Per esempio, l'inclusione dei file di font Type 1 può essere usata per includere l'output postscript nei documenti LaTeX. Il font "european computer modern" (che è una variante del font "computer modern") è disponibile in formato pfb da qualsiasi server CTAN, ad es. ftp://ftp.dante.de/tex-archive/fonts/ps-type1/cm-super/

Per esempio, il file "sfrm1000.pfb" contiene i normali font verticali con serif nella dimensione design 10pt (font name "SFRM1000"). I font computer modern, che sono ancora necessari per la matematica, sono disponibili su ftp://ftp.dante.de/tex-archive/fonts/cm/ps-type1/bluesky

Con questi si può usare qualsiasi carattere disponibile in TeX. Tuttavia, i font computer modern hanno una strana codifica. (Questo è il motivo per cui non si dovrebbe usare cmr10.pfb per il testo, ma invece sfrm1000.pfb.) L'uso dei font TeX è mostrato in una delle demo. Il file "ps_fontfile_doc.tex" nella sottodirectory /docs/psdoc della distribuzione sorgente di **gnuplot** contiene una tabella con i glifi dei mathfont di TeX.

Se il font "CMEX10" è incorporato (file "cmex10.pfb") gnuplot definisce il font aggiuntivo "CMEX10-Baseline". È spostato verticalmente per adattarsi meglio agli altri glifi (la linea di base di CMEX10 è in cima ai simboli).

Postscript prologue

Ogni file di output PostScript include una sezione %%Prolog e possibilmente alcune sezioni aggiuntive definite dall'utente che contengono, per esempio, codifiche di caratteri. Queste sezioni sono copiate da un set di file prologo di PostScript che sono compilati nell'eseguibile di gnuplot o memorizzati altrove sul computer. Una

directory predefinita dove risiedono questi file è impostata al momento della costruzione di gnuplot. Tuttavia, è possibile sovrascrivere questo default utilizzando il comando di gnuplot **set psdir** o definendo una variabile d'ambiente GNUPLOT_PS_DIR. Vedere **set psdir (p. 203)**.

Postscript adobeglyphnames

Questa impostazione è rilevante solo per l'output PostScript con codifica UTF-8. Essa controlla i nomi usati per descrivere i caratteri con punti di ingresso Unicode superiori a 0x00FF. Cioè, tutti i caratteri non compresi nel set Latin1. In generale i caratteri unicode non hanno un nome univoco; essi hanno solo un codice di identificazione unicode. Tuttavia, Adobe ha uno schema raccomandato per assegnare i nomi a certi range di caratteri (extended Latin, Greek, ecc). Alcuni font usano questo schema, altri no. Per default, gnuplot userà i nomi dei glifi di Adobe. Per esempio, la lettera Greek minuscola alfa sarà chiamata /alpha. Se si specifica **noadobeglyphnames** allora gnuplot userà invece /uni03B1 per descrivere questo carattere. Se questa impostazione non viene eseguita correttamente, il carattere potrebbe non essere trovato anche se è presente nel font. Probabilmente è sempre opportuno usare il default per i font Adobe, ma per altri font forse sarà necessario provare entrambe le impostazioni. Vedere anche **fontfile (p. 292)**.

Pslatex e pstex

Il driver **pslatex** genera output per un'ulteriore elaborazione da parte di LaTeX, mentre il driver **pstex** genera output per un'ulteriore elaborazione da parte di TeX. **pslatex** utilizza \specials comprensibili da dvips e xdvi. Le figure generate da **pstex** possono essere incluse in qualsiasi formato basato su plain (compreso LaTeX).

Sintassi:

```
set terminal [pslatex | pstex] {default}
set terminal [pslatex | pstex]
                        {rotate | norotate}
                        {oldstyle | newstyle}
                        {auxfile | noauxfile}
                        {level1 | leveldefault | level3}
                        {color | colour | monochrome}
                        {background <rgbcolor> | nobackground}
                        {dashlength | dl <DL>}
                        {linewidth | lw <LW>} {pointscale | ps <PS>}
                        {rounded | butt}
                        {clip | noclip}
                        {palfuncparam <samples>{,<maxdeviation>}}
                        {size <XX>{unit},<YY>{unit}}
                        {<font_size>}
```

Se appare il messaggio di errore

```
"Can't find PostScript prologue file ... "
```

Si prega di consultare e seguire le istruzioni nel **postscript prologue**.

L'opzione **color** abilita il colore, mentre **monochrome** preferisce elementi di disegno in bianco e nero. Inoltre, **monochrome** usa la **palette** di grigi ma non cambia il colore degli oggetti specificati con un esplicito **colorspec**. **dashlength** o **dl** ridimensiona la lunghezza dei segmenti di linea tratteggiata per <DL>, che è un numero in virgola mobile maggiore di zero. **linewidth** o **lw** ridimensiona tutti i linewidth (spessori delle linee) per <LW>.

Per default il codice PostScript generato usa le funzionalità del linguaggio che sono state introdotte in PostScript Level 2, in particolare i filtri e il pattern-fill di oggetti irregolari come i filledcurve. Le funzionalità di PostScript Level 2 sono condizionatamente protette in modo che gli interpreti di PostScript Level 1 non

emettano errori ma, piuttosto, mostrino un messaggio o un'approssimazione di PostScript Level 1. L'opzione **level1** sostituisce le approssimazioni PostScript Level 1 di queste funzionalità e non usa il codice PostScript Level 2. Questo può essere richiesto da alcune vecchie stampanti e da vecchie versioni di Adobe Illustrator. Il flag **level1** può essere attivato in seguito modificando una singola linea nel file di output PostScript per forzare l'interpretazione PostScript Level 1. Nel caso di file contenenti codice di livello 2, le funzionalità di cui sopra non appariranno o saranno sostituite da una nota quando questo flag è impostato o quando il programma di interpretazione non indica che comprende PostScript di livello 2 o superiore. Il flag **level3** abilita la codifica PNG per le immagini bitmap, che può ridurre considerevolmente la dimensione dell'output.

rounded imposta le estremità delle linee e le giunzioni delle linee affinché siano arrotondate; **butt** è il default, estremità butt e giunzioni mitered.

clip dice a PostScript di ritagliare tutto l'output nel bounding box (riquadro di delimitazione); **noclip** è il default.

palfuncparam controlla come **set palette functions** sono codificate come gradienti nell'output. Le funzioni analitiche dei componenti del colore (impostate tramite **set palette functions**) sono codificate come gradienti lineari interpolati nell'output postscript: Le funzioni dei componenti del colore sono campionate in <samples> punti e tutti i punti sono rimossi da questo gradiente che può essere rimosso senza cambiare i colori risultanti di più di <maxdeviation> (deviazione massima). Per quasi tutte le palette utili si può tranquillamente lasciare inalterati i valori predefiniti di <samples>=2000 e <maxdeviation>=0.003.

La dimensione predefinita per l'output postscript è 10 pollici x 7 pollici. Il default per l'output eps è 5 x 3.5 pollici. L'opzione **size** la modifica in qualsiasi dimensione richiesta dell'utente. Per default le dimensioni X e Y sono considerate in pollici, ma sono possibili altre unità (attualmente solo cm). La BoundingBox del grafico viene regolata correttamente per contenere l'immagine ridimensionata. Le coordinate dello schermo vanno sempre da 0.0 a 1.0 lungo tutta la lunghezza dei bordi del grafico come specificato dall'opzione **size**. NB: **questo è un cambiamento rispetto al metodo consigliato in precedenza di usare il comando set size prima di impostare il tipo di terminale**. Il vecchio metodo lasciava la BoundingBox invariata e le coordinate dello schermo non corrispondevano ai limiti reali del grafico.

Se **rotate** è specificato, l'etichetta dell'asse y è ruotata. <font_size> è la dimensione (in punti) del font desiderato.

Se **auxfile** è specificato, ordina al driver di mettere i comandi PostScript in un file ausiliario invece che direttamente nel file LaTeX. Ciò è utile se le proprie immagini sono talmente grandi che dvips non può gestirle. Il nome del file PostScript ausiliario è derivato dal nome del file TeX dato nel comando **set output**; viene stabilito sostituendo il **.tex** finale (in realtà solo l'estensione finale nel nome del file) con **.ps** nel nome del file di output, o, se il file TeX non ha estensione, viene aggiunto **.ps**. Il **.ps** è incluso nel file **.tex** da un comando \special{psfile=...}. Si ricorda di chiudere l'**output file** prima del prossimo grafico, a meno che non sia in modalità **multiplot**.

Le versioni di Gnuplot precedenti alla versione 4.2 generavano grafici di dimensioni 5 x 3 pollici usando il terminale ps(la)tex mentre la versione attuale genera 5 x 3.5 pollici per essere coerente con il terminale postscript eps. Inoltre, la larghezza del carattere è ora stimata a essere il 60% della dimensione del font mentre il vecchio terminale epslatex usava il 50%. Per ottenere il vecchio formato è necessario specificare l'opzione **oldstyle**.

Il driver pslatex offre un modo speciale per controllare il posizionamento del testo: (a) Se una qualsiasi stringa di testo inizia con '{', è necessario includere anche una '}' alla fine del testo, e l'intero testo sarà centrato sia orizzontalmente che verticalmente da LaTeX. (b) Se la stringa di testo inizia con '[', è necessario continuare con: una specificazione di posizione (fino a due su t,b,l,r), ']{', il testo stesso, e infine '}'. Il testo stesso può essere qualsiasi cosa che LaTeX può impaginare (typeset) come una LR-box. '\rule{}{}' può aiutare per un miglior posizionamento.

Le opzioni non descritte qui sono identiche al **Postscript terminal**. Guardare quella parte se si desidera sapere cosa fanno.

Esempi:

```
set term pslatex monochrome rotate      # set to defaults
```

Per scrivere i comandi PostScript nel file "foo.ps":

```
set term pslatex auxfile
set output "foo.tex"; plot ...; set output
```

Sul posizionamento dell'etichetta: Usa i valori predefiniti di gnuplot (in genere opportuni, ma a volte non sono i più adatti):

```
set title '\LaTeX\ -- $ \gamma $'
```

Centratura forzata sia in orizzontale che in verticale:

```
set label '{\LaTeX\ -- $ \gamma $}' at 0,0
```

Specifica il proprio posizionamento (qui superiore):

```
set xlabel '[t]{\LaTeX\ -- $ \gamma $}'
```

L'altra etichetta – tiene conto delle ticlabel (etichette dei tic) lunghe:

```
set ylabel '[r]{\LaTeX\ -- $ \gamma $\rule{7mm}{0pt}}'
```

Linewidth e pointsize possono essere cambiate con **set style line**.

Pstricks

Il driver **pstricks** è destinato all'uso con il pacchetto di macro "pstricks.sty" per LaTeX. È un'alternativa ai driver **eepic** e **latex**. È necessario "pstricks.sty", e, naturalmente, una stampante che capisca PostScript, o un convertitore come Ghostscript.

PSTricks è disponibile su http://tug.org/PSTricks/.

Questo driver sicuramente non si avvicina ad utilizzare la piena capacità del pacchetto PSTricks.

Sintassi:

```
set terminal pstricks
               {unit | size <XX>{unit},<YY>{unit}}
               {standalone | input}
               {blacktext | colortext | colourtext}
               {linewidth <lw>} {rounded | butt}
               {pointscale <ps>}
               {psarrows | gparrows}
               {background <rgbcolor>}
```

L'opzione **unit** produce un grafico con dimensioni interne 1x1. Il default è un grafico **size 5in,3in**.

standalone produce un file LaTeX con possibilmente più grafici, pronto per essere compilato. Il default è **input** per produrre un file TeX che può essere incluso.

blacktext costringe tutto il testo ad essere scritto in nero. **colortext** abilita il testo a colori. Il default è **blacktext**.

rounded imposta le estremità delle linee e le giunzioni delle linee affinché siano arrotondate; **butt** è il default, estremità butt e giunzioni mitered.

linewidth e **pointscale** ridimensionano rispettivamente lo spessore delle linee e la dimensione dei simboli di punti.

psarrows disegna **arrow** (frecce) usando comandi PSTricks che sono più corti ma non offrono tutte le opzioni. **gparrows** seleziona invece di disegnare le frecce usando la routine propria di gnuplot per la piena funzionalità.

La vecchia opzione **hacktext** è stata sostituita dal nuovo formato default (%h), vedere **specificatori di formato (p. 158)**.

La trasparenza richiede il supporto di Ghostscript o la conversione in PDF.

Qms

Il driver del terminale **qms** supporta la stampante laser QMS/QUIC, la Talaris 1200 e altre. Non ha opzioni.

Qt

Il terminale **qt** genera l'output in una finestra separata con la libreria Qt. Sintassi:

```
set term qt {<n>}
            {size <width>,<height>}
            {position <x>,<y>}
            {title "title"}
            {font <font>} {{no}enhanced}
            {linewidth <lw>} {dashlength <dl>}
            {{no}persist} {{no}raise} {{no}ctrl}
            {close}
            {widget <id>}
```

Sono supportate più finestre di grafici: **set terminal qt** <**n**> dirige l'output alla finestra di grafico numero n.

Il titolo predefinito della finestra è basato sul numero della finestra. Questo titolo può anche essere specificato con la keyword "title".

Le finestre dei grafici rimangono aperte anche quando il driver **gnuplot** viene spostato su un dispositivo diverso. Una finestra di grafico può essere chiusa premendo la lettera 'q' mentre quella finestra ha il focus dell'input, scegliendo **close** da un menu window manager, o con **set term qt** <**n**> **close**.

La dimensione dell'area del grafico è data in pixel, il default è 640x480. Oltre a questo, la dimensione effettiva della finestra include anche lo spazio riservato alla barra degli strumenti e alla barra di stato. Quando si ridimensiona una finestra, il grafico viene immediatamente ridimensionato per adattarsi alle nuove dimensioni della finestra. Il terminale **qt** ridimensiona l'intero grafico, compresi i font e i linewidth, e mantiene costante il suo rapporto d'aspetto globale. Se si digita **replot**, si fa clic sull'icona **replot** nella barra degli strumenti del terminale o si digita un nuovo comando **plot**, il nuovo grafico si adatterà completamente alla finestra e la dimensione del font e gli spessori della linea saranno riportati ai loro valori predefiniti.

L'opzione position può essere usata per impostare la posizione della finestra del grafico. L'opzione position viene applicata solo al primo grafico dopo il comando **set term**.

La finestra del grafico attiva (quella selezionata da **set term qt** <**n**>) è interattiva. Il suo comportamento è condiviso con altri tipi di terminale. Vedere **mouse (p. 178)** per i dettagli. Possiede anche alcune icone extra, che dovrebbero essere autoesplicative.

Questo terminale supporta una modalità di testo avanzato, che permette ai font e ad altri comandi di formattazione (pedici, apici, ecc.) di essere incorporati in etichette e altre stringhe di testo. La sintassi della modalità di testo avanzato è condivisa con altri tipi di terminale di gnuplot. Vedere **enhanced (p. 35)** per maggiori dettagli.

<font> è nel formato "FontFace,FontSize", cioè la variante (face) e la dimensione separate da una virgola in una singola stringa. FontFace e un normale nome di variante di font, come 'Arial'. Se non si fornisce FontFace, il terminale qt userà 'Sans'. FontSize e la dimensione del font, in punti. Se non viene fornita, il terminale qt userà una dimensione di 9 punti.

```
Per esempio:
   set term qt font "Arial,12"
   set term qt font "Arial" # per cambiare solo la variante del font
   set term qt font ",12" # per cambiare solo la dimensione del font
   set term qt font "" # per reimpostare il nome e la dimensione del font
```

La lunghezza del trattino influenza solo i pattern personalizzati, non il set integrato di Qt.

Per ottenere il miglior output possibile, il rendering coinvolge tre meccanismi: antialiasing, oversampling e hinting. L'oversampling abbinato all'antialiasing fornisce una precisione subpixel, in modo che gnuplot possa disegnare una linea da coordinate non intere. Questo evita effetti di oscillazione sulle linee diagonali ('plot x', per esempio). Hinting evita la sfocatura sulle linee orizzontali e verticali causata dall'oversampling. Il terminale farà scattare queste linee alle coordinate intere in modo che una linea larga un pixel sia effettivamente disegnata su uno e un solo pixel.

Per default, la finestra viene portata in cima al desktop quando viene disegnato un grafico. Ciò può essere controllato con la keyword "raise". La keyword "persist" impedirà a gnuplot di uscire prima che vengano chiuse esplicitamente tutte le finestre dei grafici.

Il tasto <space> solleva la finestra della console di gnuplot [solo MS Windows]. Il tasto 'q' chiude la finestra del grafico. Questi tasti di scelta rapida possono essere cambiati in ctrl-space e ctrl-q usando la keyword "{no}ctrl" delle opzioni del terminale. Tuttavia il modo migliore per selezionare ctrl-q piuttosto che 'q' è quello di usare il toggle nel widget degli strumenti della finestra del grafico.

Il driver esterno di gnuplot, gnuplot_qt, viene cercato in un posto predefinito scelto quando il programma viene compilato. È possibile sovrascrivere ciò definendo la variabile d'ambiente GNUPLOT_DRIVER_DIR.

Regis

Nota: legacy terminal. Il terminale **regis** genera output nel linguaggio grafico REGIS. Ha la possibilità di usare 4 (il default) o 16 colori.

Sintassi:

```
set terminal regis {4 | 16}
```

Sixelgd

Sintassi:

```
set terminal sixelgd
       {{no}enhanced} {{no}truecolor}
       {{no}transparent} {rounded|butt}
       {linewidth <lw>} {dashlength <dl>}
       {tiny | small | medium | large | giant}
       {font "<face> {,<pointsize>}"} {fontscale <scale>}
       {size <x>,<y>} {{no}crop} {animate}
       {background <rgb_color>}
```

Il formato di output **sixel** era originariamente usato dai terminali e dalle stampanti DEC. Questo driver produce un flusso di output sixel convertendo un'immagine PNG creata internamente usando la libreria gd. Il flusso di output sixel può essere visualizzato nel terminale mentre viene creato o può essere scritto in un file in modo che possa essere riprodotto in seguito echeggiando (echoing) il file nel terminale.

L'opzione **animate** reimposta la posizione del cursore in alto a sinistra del terminale all'inizio di ogni grafico in modo che i grafici successivi sovrascrivano la stessa area sullo schermo piuttosto che avere i grafici precedenti che scorrono sullo schermo verso l'alto. Questo può essere desiderabile per creare un'animazione sul posto (in-place).

transparent ordina al driver di rendere il colore di sfondo trasparente. Il default è **notransparent**.

Le opzioni **linewidth** e **dashlength** sono fattori di scala che influenzano tutte le linee disegnate, cioè vengono moltiplicati per i valori richiesti nei vari comandi di disegno.

Per default l'output sixel usa 16 colori indicizzati. L'opzione **truecolor** invece crea un'immagine png TrueColor che viene mappata su 256 colori nell'immagine sixel in output. Gli stili di riempimento trasparenti richiedono l'opzione **truecolor**. Vedere **fillstyle (p. 209)**. Uno sfondo **trasparent** è possibile sia nelle immagini indicizzate che in quelle TrueColor.

butt ordina al driver di usare un metodo di disegno della linea che non superi il punto finale di una linea desiderato. Questa impostazione è applicabile solo per spessori di linea maggiori di 1. Questa impostazione è particolarmente utile quando si disegnano linee orizzontali o verticali. Il default è **rounded**.

I dettagli della selezione del font sono complicati. Per maggiori informazioni, vedere **font (p. 46)**.

La dimensione del grafico in output <x,y> è data in pixel — il default è 640x480. Vedere informazioni aggiuntive sotto **canvas (p. 32)** e **set size (p. 205)**. Lo spazio vuoto ai bordi del grafico completato può essere tagliato usando l'opzione **crop**, ottenendo una dimensione finale dell'immagine più piccola. Il default è **nocrop**.

Il terminale è stato testato con successo con i terminali xterm, mlterm e mintty. Gli ultimi due supportano la modalità **truecolor** usando 256 colori sixel fuori della scatola (out of box). Le copie distribuite di xterm possono essere state configurate o meno per supportare le grafiche sixel e possono essere limitate a 16 colori.

Svg

Questo terminale produce file nel formato W3C Scalable Vector Graphics.

Sintassi:

```
set terminal svg {size <x>,<y> {|fixed|dynamic}}
                 {mouse} {standalone | jsdir <dirname>}
                 {name <plotname>}
                 {font "<fontname>{,<fontsize>}"} {{no}enhanced}
                 {fontscale <multiplier>}
                 {rounded|butt|square} {solid|dashed} {linewidth <lw>}
                 {background <rgb_color>}
```

dove <x> e <y> sono le dimensioni del grafico SVG da generare, **dynamic** permette al visualizzatore di svg di ridimensionare il grafico, mentre l'impostazione predefinita, **fixed**, richiederà una dimensione assoluta.

linewidth <**w**> aumenta lo spessore di tutte le linee usate nella figura per un fattore di <w>.

<font> è il nome del font predefinito da usare (default Arial) e <fontsize> è la dimensione del font (in punti, default 12). I programmi di visualizzazione SVG possono sostituire altri font quando il file viene visualizzato.

La sintassi della modalità di testo avanzato è condivisa con altri tipi di terminale di gnuplot. Vedere **enhanced (p. 35)** per ulteriori dettagli.

L'opzione **mouse** dice a gnuplot di aggiungere il supporto per il tracciamento del mouse e per attivare/disattivare i singoli grafici cliccando sulla corrispondente key entry. Per default questo viene fatto includendo un link che punta ad uno script in una directory locale, di solito /usr/local/share/gnuplot/<version>/js. È possibile cambiare ciò usando l'opzione **jsdir** per specificare una diversa directory locale o un URL generale. Quest'ultimo è solitamente appropriato se si sta incorporando il svg in una pagina web. In alternativa, l'opzione **standalone** incorpora il codice del mouse nel documento svg stesso piuttosto che linkare a una risorsa esterna.

Quando un file SVG sarà usato insieme a file esterni, per esempio se è referenziato da codice javascript in una pagina web o in un documento padre, allora è necessario un nome univoco per evitare potenziali riferimenti conflittuali ad altri grafici SVG. Usare l'opzione **name** per assicurare l'unicità.

Svga

Legacy terminal. Il driver del terminale **svga** supporta i PC con grafica SVGA. In genere viene compilato solo con DJGPP e usa la libreria grafica GRX. Esiste anche una variante per Windows 32bit, che è usata principalmente per il debug. La libreria supporta anche X11, Linux console e SDL, ma questi obiettivi non sono attualmente supportati.

Sintassi:

```
set terminal svga {font "<fontname>"}
                  {{no}enhanced}
                  {background <rgb color>}
                  {linewidth|lw <lw>}
                  {pointscale|ps <scale>}
                  {fontscale|fs <scale>}
```

Il supporto di testo avanzato può essere attivato usando l'opzione **enhanced**, vedere **enhanced text (p. 35)**. Si noti che cambiare la dimensione del font del testo avanzato non è attualmente supportato. Quindi, gli apici e i pedici avranno la stessa dimensione.

Il parametro **linewidth** ridimensiona lo spessore delle linee. Il parametro **pointscale** imposta il fattore di scala per i simboli dei punti. È possibile utilizzare **fontscale** per ridimensionare il font bitmap. Questo potrebbe essere utile se si possiede un display ad alta risoluzione. Si noti che i fattori interi danno migliori risultati.

Tek40

Questa famiglia di driver di terminale supporta una varietà di terminali simili a VT. **tek40xx** supporta Tektronix 4010 e altri, così come la maggior parte degli emulatori TEK. **vttek** supporta emulatori di terminale tek40xx simili a VT. I seguenti sono presenti solo se selezionati quando gnuplot è stato creato: **kc-tek40xx** supporta gli emulatori di terminale MS-DOS Kermit Tek4010 a colori; **km-tek40xx** li supporta in monochrome. **selanar** supporta la grafica Selanar. **bitgraph** supporta i terminali BBN Bitgraph. Nessuno di essi ha delle opzioni.

Tek410x

Il driver del terminale **tek410x** supporta la famiglia 410x e 420x di terminali Tektronix. Non ha opzioni.

Texdraw

Il driver del terminale **texdraw** supporta l'ambiente texdraw (La)TeX. È destinato all'uso con il pacchetto texdraw, vedere https://www.ctan.org/tex-archive/graphics/texdraw/ .

```
set terminal texdraw
               {size <XX>{unit},<YY>{unit}}
               {standalone | input}
               {blacktext | colortext | colourtext}
               {linewidth <lw>} {rounded | butt}
               {pointscale <ps>}
               {psarrows | gparrows} {texpoints | gppoints}
               {background <rgbcolor>}
```

Nota: La grafica è solo in scala di grigi. Il testo è sempre nero. Box e poligoni vengono riempiti utilizzando solo livelli di grigio pieno. I pattern non sono disponibili.

I punti, tra le altre cose, sono disegnati utilizzando i comandi LaTeX "\Diamond" e "\Box". Questi comandi non appartengono più al nucleo (core) di LaTeX2e; sono inclusi nel pacchetto latexsym, che fa parte della distribuzione base e quindi di qualsiasi implementazione di LaTeX. Si prega di non dimenticare di usare questo pacchetto. Altri tipi di punti usano simboli del pacchetto amssymb. È necessario specificare l'opzione **gppoints** per la compatibilità con plain TeX.

standalone produce un file LaTeX con possibilmente più grafici, pronto per essere compilato. Il default è **input** per produrre un file TeX che può essere incluso.

blacktext costringe tutto il testo ad essere scritto in nero. **colortext** abilita il testo "colored" (a colori). Il default è **blacktext** e "color" significa in realtà scala di grigi.

rounded imposta le estremità delle linee e le giunzioni delle linee affinché siano arrotondate; **butt** imposta estremità butt e giunzioni mitered ed è il default.

linewidth e **pointscale** ridimensionano rispettivamente lo spessore delle linee e la dimensione dei simboli di punti. **pointscale**si applica solo a **gppoints**.

psarrows disegna **arrow** (frecce) usando comandi TeXdraw che sono più corti ma non offrono tutte le opzioni. **gparrows** seleziona invece di disegnare le frecce usando la routine propria di gnuplot per la piena funzionalità. Allo stesso modo, **texpoints** e **gppoints** selezionano i simboli LaTeX o le routine di disegno dei punti di gnuplot.

Tgif

Legacy terminal (presente solo se gnuplot è stato configurato con l'opzione –with-tgif). Tgif è/era uno strumento di disegno interattivo di grafica vettoriale 2-D basato su Xlib, capace anche di importare e marcare immagini bitmap.

Il driver **tgif** supporta la scelta del font e della dimensione del font, e più grafici (graph) sulla pagina. Le proporzioni degli assi non vengono modificate.

Sintassi:

```
set terminal tgif {portrait | landscape | default} {<[x,y]>}
                  {monochrome | color}
                  {{linewidth | lw} <LW>}
                  {solid | dashed}
                  {font "<fontname>{,<fontsize>}"}
```

dove <[x,y]> specifica il numero di grafici (graph) nelle direzioni x e y sulla pagina **color** abilita il colore, **linewidth** scala tutti i linewidth per <LW>, "<fontname>" è il nome di un font PostScript valido, e <fontsize> specifica la dimensione del font PostScript. **defaults** imposta tutte le opzioni ai loro default: **portrait**, **[1,1]**, **color**, **linewidth 1.0**, **dashed**, "**Helvetica,18**".

L'opzione **solid** è solitamente da preferire se le linee sono colorate, come spesso lo sono nell'editor. La copia cartacea sarà in bianco e nero, quindi per questo motivo dovrebbe essere scelta l'opzione **dashed**.

Multiplot è implementato in due modi diversi.

La prima implementazione multiplot è la funzione multiplot standard di gnuplot:

```
set terminal tgif
set output "file.obj"
set multiplot
set origin x01,y01
set size  xs,ys
plot ...
     ...
set origin x02,y02
plot ...
unset multiplot
```

Vedere **set multiplot (p. 180)** per ulteriori informazioni.

La seconda versione è l'opzione [x,y] per il driver stesso. Il vantaggio di questa implementazione è che tutto viene ridimensionato e posizionato automaticamente senza la necessità di impostare origini e dimensioni; i grafici (graph) mantengono le loro naturali proporzioni x/y di 3/2 (o qualunque sia fissata da **set size**).

Se sono selezionati entrambi i metodi multiplot, viene scelto il metodo standard e viene dato un messaggio di avvertimento.

Esempi di grafici singoli (o standard multiplot):

```
set terminal tgif                  # defaults
set terminal tgif "Times-Roman,24"
set terminal tgif landscape
set terminal tgif landscape solid
```

Esempi che utilizzano il meccanismo multiplot integrato:

```
set terminal tgif portrait [2,4]  # portrait; 2 grafici nella direzione
                                  # x e 4 nella direzione y
set terminal tgif [1,2]           # portrait; 1 grafico nella direzione
                                  # x e 2 nella direzione y
set terminal tgif landscape [3,3] # landscape; 3 grafici in entrambe
                                  # le direzioni
```

Tikz

Questo driver crea output per l'uso con il pacchetto TikZ di macro grafiche in TeX. Attualmente è implementato tramite uno script lua esterno, e **set term tikz** è una forma abbreviata del comando **set term lua tikz**. Vedere **term lua (p. 276)** per maggiori informazioni. Usare il comando **set term tikz help** per stampare le opzioni del terminale.

Tkcanvas

Questo driver di terminale genera i comandi widget Tk canvas in uno dei seguenti linguaggi di scripting: Tcl (default), Perl, Python, Ruby, o REXX.

Sintassi:

```
set terminal tkcanvas {tcl | perl | perltkx | python | ruby | rexx}
                      {standalone | input}
                      {interactive}
                      {rounded | butt}
                      {nobackground | background <rgb color>}
                      {{no}rottext}
                      {size <width>,<height>}
                      {{no}enhanced}
                      {externalimages | pixels}
```

Eseguire la seguente sequenza di comandi Tcl/Tk per visualizzare il risultato:

```
package require Tk
# le due linee seguenti sono necessarie solo per supportare immagini esterne
package require img::png
source resize.tcl
source plot.tcl
canvas .c -width 800 -height 600
pack .c
gnuplot .c
```

O, per Perl/Tk usare un programma come questo:

```
use Tk;
my $top = MainWindow->new;
my $c = $top->Canvas(-width => 800, -height => 600)->pack;
my $gnuplot = do "plot.pl";
$gnuplot->($c);
MainLoop;
```

O, per Perl/Tkx usare un programma come questo:

```
use Tkx;
my $top = Tkx::widget->new(".");
my $c = $top->new_tk__canvas(-width => 800, -height => 600);
$c->g_pack;
my $gnuplot = do "plot.pl";
$gnuplot->($c);
Tkx::MainLoop();
```

O, per Python/Tkinter usare un programma come questo:

```
from tkinter import *
from tkinter import font
root = Tk()
c = Canvas(root, width=800, height=600)
c.pack()
exec(open('plot.py').read())
gnuplot(c)
root.mainloop()
```

O, per Ruby/Tk usare un programma come questo:

```
require 'tk'
root = TkRoot.new { title 'Ruby/Tk' }
c = TkCanvas.new(root, 'width'=>800, 'height'=>600) { pack  { } }
load('plot.rb')
gnuplot(c)
Tk.mainloop
```

O, per Rexx/Tk usare un programma come questo:

```
/**/
call RxFuncAdd 'TkLoadFuncs', 'rexxtk', 'TkLoadFuncs'
call TkLoadFuncs
cv = TkCanvas('.c', '-width', 800, '-height', 600)
call TkPack cv
call 'plot.rex' cv
do forever
    cmd = TkWait()
    if cmd = 'AWinClose' then leave
    interpret 'call' cmd
end
```

Il codice generato da **gnuplot** (negli esempi precedenti, questo codice è scritto in "plot.<ext>") contiene le seguenti procedure:

gnuplot(canvas)

```
prende il nome di un canvas come suo argomento.
Quando viene chiamato, cancella il canvas, trova le dimensioni del
canvas e disegna il grafico in esso, ridimensionato per potersi adattare.
```

gnuplot_plotarea()

```
restituisce una lista contenente i bordi dell'area di plotting
(xleft, xright, ytop, ybot) nelle coordinate dello schermo del canvas.    Funziona solo per il plotti
```

gnuplot_axisranges()

```
restituisce gli intervalli dei due assi in coordinate del grafico
(x1min, x1max, y1min, y1max, x2min, x2max, y2min, y2max).
Funziona solo per il plotting bidimensionale (‘plot‘).
```

È possibile creare script autonomi e minimali usando l'opzione **standalone**. Il default è **input** il quale crea script che devono essere (source'd) reperiti (o caricati o chiamati o qualunque sia il termine adeguato per il linguaggio selezionato).

Se viene specificata l'opzione **interactive**, cliccando con il mouse su un segmento di linea verranno stampate le coordinate del suo punto medio su stdout. L'utente può sostituire questo comportamento fornendo una procedura user_gnuplot_coordinates che prende i seguenti argomenti:

```
win id x1s y1s x2s y2s x1e y1e x2e y2e x1m y1m x2m y2m,
```

ovvero il nome del canvas e l'id del segmento di linea seguito dalle coordinate del suo punto iniziale e finale nei due possibili intervalli degli assi; le coordinate del punto medio sono riempite solo per gli assi logaritmici.

Per default il canvas è **transparent**, ma si può impostare un colore di sfondo esplicito con l'opzione **background**.

rounded imposta le estremità delle linee e le giunzioni delle linee affinché siano arrotondate; **butt** è il default: estremità butt e giunzioni mitered.

Il testo ad angoli arbitrari può essere attivato con l'opzione **rottext**, che richiede Tcl/Tk 8.6 o successivo. Il default è **norottext**.

L'opzione **size** cerca di ottimizzare le dimensioni dei tic e dei font per la dimensione del canvas fornita. Per default si suppone che la dimensione di output sia 800 x 600 pixel.

enhanced seleziona l'elaborazione **enhanced text** (default), ma attualmente è disponibile solo per Tcl.

L'opzione **pixels** (default) seleziona il gestore dell'immagine pixel per pixel di sicurezza, vedere anche **image pixels (p. 78)**. L'opzione **externalimages** salva le immagini come immagini png esterne, che sono poi caricate e ridimensionate dal codice di tkcanvas. Questa opzione è disponibile solo per Tcl e la visualizzazione può essere lenta in alcune situazioni poiché il gestore di immagini Tk non fornisce un ridimensionamento arbitrario (arbitrary scaling). Gli script hanno bisogno di reperire il file rescale.tcl fornito.

La modalità interattiva non è ancora implementata per Python/Tk e Rexx/Tk. La modalità interattiva per Ruby/Tk non supporta ancora user_gnuplot_coordinates.

Tpic

Nota: Legacy terminal (non costruito di default). In versioni precedenti di gnuplot i terminali latex, emtex, eepic, e tpic fornivano un supporto minimo per gli stili grafici al di là di semplici linee e punti. Sono stati direttamente sostituiti dal terminale **pict2e**. Per tipi di terminale compatibili con TeX/LaTeX più abili, vedere **cairolatex (p. 252)**, **context (p. 259)**, **epslatex (p. 265)**, **mp (p. 280)**, **pstricks (p. 296)**, e **tikz (p. 302)**.

Il driver del terminale **tpic** supporta l'ambiente delle immagini LaTeX con tpic \specials. Le opzioni sono la dimensione del punto, lo spessore della linea e l'intervallo punto-trattino (dot-dash interval).

Sintassi:

```
set terminal tpic <pointsize> <linewidth> <interval>
```

dove **pointsize** e **linewidth** sono interi in milli-pollici e **interval** è un float in pollici. Se viene specificato un valore non positivo, viene scelto il valore predefinito: pointsize = 40, linewidth = 6, interval = 0.1.

Tutti i driver per LaTeX offrono un modo speciale per controllare il posizionamento del testo: Se una qualsiasi stringa di testo inizia con '{', è necessario includere anche una '}' alla fine del testo, e l'intero testo sarà centrato sia orizzontalmente che verticalmente da LaTeX. — Se la stringa di testo inizia con '[', è necessario continuare con: una specificazione di posizione (fino a due tra t,b,l,r), ']{', il testo stesso, e infine '}'. Il

testo stesso può essere qualsiasi cosa che LaTeX può impaginare (typeset) come una LR-box. '\rule{}{}' può aiutare per un miglior posizionamento.

Esempi: Sul posizionamento dell'etichetta: Usa i valori predefiniti di gnuplot (in genere opportuni, ma a volte non sono i più adatti):

```
set title '\LaTeX\ -- $ \gamma $'
```

Centratura forzata sia in orizzontale che in verticale:

```
set label '{\LaTeX\ -- $ \gamma $}' at 0,0
```

Specifica il proprio posizionamento (qui superiore):

```
set xlabel '[t]{\LaTeX\ -- $ \gamma $}'
```

L'altra etichetta – tiene conto delle ticlabel (etichette dei tic) lunghe:

```
set ylabel '[r]{\LaTeX\ -- $ \gamma $\rule{7mm}{0pt}}'
```

VWS

Nota: legacy terminal. Il driver del terminale **VWS** supporta il VAX Windowing System. Non ha opzioni. Rileverà il tipo di display (monochrome, scala di grigi o colore). Tutti gli stili di linea sono plottati come linee continue.

Windows

Il terminale **windows** è un driver di terminale interattivo veloce che usa Windows GDI per disegnare e scrivere il testo. I multipiattaforma **terminal wxt** e **terminal qt** sono supportati anche su Windows.

Sintassi:

```
set terminal windows {<n>}
                     {color | monochrome}
                     {solid | dashed}
                     {rounded | butt}
                     {enhanced | noenhanced}
                     {font <fontspec>}
                     {fontscale <scale>}
                     {linewidth <scale>}
                     {pointscale <scale>}
                     {background <rgb color>}
                     {title "Plot Window Title"}
                     {{size | wsize} <width>,<height>}
                     {position <x>,<y>}
                     {docked {layout <rows>,<cols>} | standalone}
                     {close}
```

Sono supportate più finestre di grafico: **set terminal win** <**n**> dirige l'output alla finestra di grafico numero n.

color e **monochrome** selezionano l'output colorato o mono, **dashed** e **solid** selezionano linee tratteggiate o continue. Notare che **color** ha come default **solid**, mentre **monochrome** ha come default **dashed**. **rounded** imposta le estremità delle linee e le giunzioni delle linee affinché siano arrotondate; **butt** è il default, estremità butt e giunzioni mitered. **enhanced** abilita le funzionalità della modalità di testo avanzato (pedici, indici e font misti, vedere **enhanced text (p. 35)** per maggiori informazioni). <**fontspec**> è nel formato "<fontface>,<fontsize>", dove "<fontface>" è il nome di un font Windows valido, e <fontsize> è la dimensione del font in punti ed entrambi i componenti sono opzionali. Si noti che nelle versioni precedenti di

gnuplot la dichiarazione **font** poteva essere omessa e <fontsize> poteva essere data come un numero senza le doppie virgolette. Questo non è più supportato. **linewidth**, **fontscale**, **pointscale** possono essere usati per modificare lo spessore delle linee, la dimensione del testo, o la dimensione dei simboli dei punti. **title** cambia il titolo della finestra del grafico (graph). **size** definisce la larghezza e l'altezza dell'area di disegno della finestra in pixel, **wsize** definisce la dimensione effettiva della finestra stessa e **position** definisce l'origine della finestra, ovvero la posizione dell'angolo superiore sinistro sullo schermo (sempre in pixel). Queste opzioni sovrascrivono qualsiasi impostazione predefinita presente nel file **wgnuplot.ini**.

docked incorpora la finestra del grafico (graph) nella finestra di testo di wgnuplot e le opzioni **size** e **position** vengono ignorate. Si noti che **docked** non è disponibile per gnuplot in modalità console. Impostando questa opzione si cambia il default per le nuove finestre. Il default iniziale è **standalone**. L'opzione **layout** permette di riservare un numero minimo di colonne e righe per i grafici (graph) in modalità docked. Se sono presenti più grafici (graph) di quelli che il layout dato può contenere, verranno aggiunte altre righe. I grafici (graph) sono ordinati in base all'id numerico, riempiendo prima le righe.

Altre opzioni possono essere cambiate usando il **graph-menu** o il file di inizializzazione **wgnuplot.ini**.

La versione di Windows normalmente termina non appena si raggiunge la fine di qualsiasi file dato come argomento della linea di comando (ovvero in modalità non interattiva), a meno che non si specifichi **-** come ultima opzione della linea di comando. Inoltre non verrà mostrata affatto la finestra di testo, in questa modalità, solo il grafico. Dando l'argomento opzionale **-persist** (lo stesso che per gnuplot sotto x11; le precedenti opzioni solo per Windows **/noend** o **-noend** sono ancora accettate), non verrà chiuso gnuplot. A differenza di gnuplot su altri sistemi operativi, la linea di comando interattiva di gnuplot è accessibile dopo l'opzione -persist.

La finestra del grafico rimane aperta quando il terminale di gnuplot viene cambiato con un comando **set term**. La finestra del grafico può essere chiusa con **set term windows close**.

gnuplot supporta diversi metodi per creare output stampato su Windows, vedere **windows printing (p. 307)**. Il terminale windows supporta lo scambio di dati con altri programmi tramite gli appunti e i file EMF, vedere **graph-menu (p. 306)**. È anche possibile utilizzare il **terminale emf** per creare file EMF.

Graph-menu

La finestra **gnuplot graph** dispone delle seguenti opzioni su un menu pop-up a cui si accede premendo il tasto destro del mouse(*) o selezionando **Options** dal menu di sistema o dalla barra degli strumenti:

Copy to Clipboard copia una bitmap e un'immagine metafile migliorata.

Save as EMF... permette all'utente di salvare la finestra del grafico (graph) corrente come metafile migliorato (EMF o EMF+).

Save as Bitmap... permette all'utente di salvare una copia del grafico (graph) come file bitmap.

Print... stampa le finestre grafiche usando un driver di stampa di Windows e permette di selezionare la stampante e il ridimensionamento dell'output. Vedere anche **windows printing (p. 307)**.

Bring to Top, se selezionata, solleva la finestra del grafico (graph) in cima dopo ogni grafico.

Color, se selezionata, abilita l'output a colori. Quando non è selezionata, obbliga tutto l'output ad essere in scala di grigi. Questo è utile, per esempio, per testare l'aspetto delle stampe monocromatiche.

Il **GDI backend** che usa la classica API GDI è deprecato ed è stato disabilitato in questa versione.

GDI+ backend disegna sullo schermo usando l'API GDI+ di Windows. Supporta l'antialiasing, l'oversampling, la trasparenza e i dash pattern personalizzati. Questo era il default nelle versioni 5.0 e 5.2.

Direct2D backend utilizza le API Direct2D & DirectWrite per disegnare. Utilizza l'accelerazione della scheda grafica ed è quindi tipicamente molto più veloce. Poiché Direct2D non può creare dati EMF, il salvataggio e la copia negli appunti dei dati EMF ricadono su GDI+ mentre i dati bitmap sono generati da D2d. Questo è il backend raccomandato e predefinito dalla versione 5.3.

Oversampling disegna linee diagonali in posizioni di pixel frazionarie per evitare effetti di "oscillazione" (wobbling). Le linee verticali o orizzontali sono ancora fissate su posizioni di pixel interi per evitare linee sfocate.

Antialiasing permette di smussare le linee e i bordi. Notare che questo rallenta le operazioni di disegno. **Antialiasing of polygons** è abilitato di default ma potrebbe rallentare le operazioni di disegno con il backend GDI+.

Fast rotation disattiva temporaneamente l'antialiasing mentre si ruota il grafico (graph) con il mouse. Questo accelera considerevolmente le operazioni di disegno a scapito di un ulteriore ridisegno dopo aver rilasciato il pulsante del mouse.

Background... imposta il colore di sfondo della finestra.

Choose Font... seleziona il font utilizzato nella finestra grafica.

Update wgnuplot.ini salva nel file di inizializzazione **wgnuplot.ini** le posizioni correnti delle finestre, le dimensioni delle finestre, il font della finestra di testo, la dimensione del font della finestra di testo, il font della finestra del grafico (graph), la dimensione del font della finestra del grafico (graph), il colore di sfondo.

(*) Si noti che questo menu è disponibile solo premendo il tasto destro del mouse con **unset mouse**.

Printing

In ordine di preferenza, i grafici (graph) possono essere stampati nei seguenti modi:

1. Usare il comando **set terminal** di **gnuplot** per selezionare una stampante e **set output** per reindirizzare l'output in un file.

2. Selezionare il comando **Print...** dalla finestra **gnuplot graph**. Un comando extra **screendump** consente di farlo dalla finestra di testo.

3. Se viene usato **set output "PRN"**, l'output andrà in un file temporaneo. Quando si esce da **gnuplot** o quando si cambia l'output con un altro comando **set output**, apparirà una finestra di dialogo che permetterà di selezionare la porta della stampante. Se si sceglie OK, l'output verrà stampato sulla porta selezionata, passando attraverso il gestore di stampa senza subire modifiche. È possibile inviare accidentalmente (o intenzionalmente) l'output della stampante (printer output) destinato a una determinata stampante a una stampante incompatibile.

Text-menu

La finestra **gnuplot text** dispone delle seguenti opzioni in un menu pop-up a cui si accede premendo il tasto destro del mouse o selezionando **Options** dal menu di sistema:

Copy to Clipboard copia il testo evidenziato negli appunti.

Paste copia il testo dagli appunti come se fosse stato digitato dall'utente.

Choose Font... seleziona il font usato nella finestra di testo.

System Colors, se selezionata, fa sì che la finestra di testo rispetti i System Colors (colori di sistema) impostati tramite il Control Panel (pannello di controllo). Quando non è selezionata, il testo è nero o blu su sfondo bianco.

Wrap long lines, se selezionata, fa sì che le linee più lunghe della larghezza della finestra attuale vengano mandate a capo (wrapped).

Update wgnuplot.ini salva le impostazioni correnti nel file di inizializzazione **wgnuplot.ini**, che si trova nella directory dei dati dell'applicazione dell'utente.

Wgnuplot.mnu

Se il file di menu **wgnuplot.mnu** si trova nella stessa directory di **gnuplot**, allora verrà caricato il menu specificato in **wgnuplot.mnu**. Comandi menu:

```
[Menu]      avvia un nuovo menu con il nome sulla linea seguente.
[EndMenu]   termina il menu attuale.
[--]        inserisce un separatore di menu orizzontale.
[|]         inserisce un separatore di menu verticale.
[Button]    mette la prossima macro su un pulsante invece che su un menu.
```

Le macro occupano due linee, il nome della macro (menu entry) sulla prima linea e la macro sulla seconda. Gli spazi iniziali sono ignorati. Comandi macro:

```
[INPUT]     Stringa di input con prompt terminato da [EOS] o {ENTER}
[EOS]       Terminatore End Of String. Non genera output.
[OPEN]      Ottenere il nome di un file da aprire, con il titolo
            della finestra di dialogo terminato da [EOS], seguito
            da un filename predefinito terminato da [EOS] o {ENTER}.
[SAVE]      Ottenere il nome di un file da salvare. Parametri come [OPEN]
[DIRECTORY] Ottenere il nome di una directory, con il titolo della
            finestra di dialogo terminato da [EOS] o {ENTER}
```

Sostituzioni di caratteri macro:

```
{ENTER}     Ritorno a capo '\r'
{TAB}       Tab '\011'
{ESC}       Escape '\033'
{^A}        '\001'
...
{^_}        '\031'
```

Le macro sono limitate a 256 caratteri dopo l'espansione.

Wgnuplot.ini

La finestra di testo di Windows e il terminale **windows** leggeranno alcune delle loro opzioni dalla sezione [**WGNUPLOT**] di **wgnuplot.ini**. Questo file si trova nella directory dei dati dell'applicazione dell'utente. Ecco un file **wgnuplot.ini** di esempio:

```
[WGNUPLOT]
TextOrigin=0 0
TextSize=640 150
TextFont=Consolas,9
TextWrap=1
TextLines=400
TextMaximized=0
SysColors=0
GraphOrigin=0 150
GraphSize=640 330
GraphFont=Tahoma,10
GraphColor=1
GraphToTop=1
GraphGDI+=1
GraphD2D=0
GraphGDI+Oversampling=1
GraphAntialiasing=1
```

```
GraphPolygonAA=1
GraphFastRotation=1
GraphBackground=255 255 255
DockVerticalTextFrac=350
DockHorizontalTextFrac=400
```

Queste impostazioni vengono applicate solo alla finestra di testo di wgnuplot. Le entry **TextOrigin** e **TextSize** specificano la posizione e le dimensioni della finestra di testo. Se **TextMaximized** è diverso da zero, la finestra sarà ingrandita.

L'entry **TextFont** specifica il font e la dimensione della finestra di testo.

L'entry **TextWrap** seleziona il ritorno a capo (wrapping) di lunghe linee di testo.

L'entry **TextLines** specifica il numero di linee (unwrapped) che il buffer interno della finestra di testo può contenere. Attualmente questo valore non può essere cambiato dall'interno di wgnuplot.

Vedere **text-menu (p. 307)**.

DockVerticalTextFrac e **DockHorizontalTextFrac** impostano la frazione della finestra riservata alla finestra di testo in permille del layout orizzontale e verticale.

L'entry **GraphFont** specifica il nome del font e la dimensione in punti.

Vedere **graph-menu (p. 306)**.

Wxt

Il terminale **wxt** genera output in una finestra separata. La finestra è creata dalla libreria wxWidgets, da cui proviene 'wxt'. Il disegno viene fatto tramite cairo, una libreria grafica 2D, e pango, una libreria per la stesura e il rendering del testo.

Sintassi:

```
set term wxt {<n>}
             {size <width>,<height>} {position <x>,<y>}
             {background <rgb_color> | nobackground}
             {{no}enhanced}
             {font <font>} {fontscale <scale>}
             {title "title"}
             {linewidth <lw>} {butt|rounded|square}
             {dashlength <dl>}
             {{no}persist}
             {{no}raise}
             {{no}ctrl}
             {close}
```

Sono supportate più finestre di grafici: **set terminal wxt <n>** dirige l'output alla finestra di grafico numero n.

Il titolo predefinito della finestra è basato sul numero della finestra. Questo titolo può anche essere specificato con la keyword `"title"`.

Le finestre dei grafici rimangono aperte anche quando il driver **gnuplot** viene spostato su un dispositivo diverso. Una finestra di grafico può essere chiusa premendo la lettera 'q' mentre quella finestra ha il focus dell'input, scegliendo **close** da un menu window manager, o con **set term wxt <n> close**.

La dimensione dell'area del grafico è data in pixel, per default è 640x384. Oltre a questo, la dimensione effettiva della finestra include anche lo spazio riservato alla barra degli strumenti e alla barra di stato. Quando si ridimensiona una finestra, il grafico viene immediatamente ridimensionato per adattarsi alle nuove dimensioni della finestra. A differenza di altri terminali interattivi, il terminale **wxt** ridimensiona l'intero grafico, compresi i font e i linewidth, e mantiene costante il suo rapporto d'aspetto globale, lasciando

uno spazio vuoto dipinto di grigio. Se si digita **replot**, si fa clic sull'icona **replot** nella barra degli strumenti del terminale o si digita un nuovo comando **plot**, il nuovo grafico si adatterà completamente alla finestra e la dimensione del font e gli spessori della linea saranno riportati ai loro valori predefiniti.

L'opzione position può essere usata per impostare la posizione della finestra del grafico. L'opzione position si applica solo al primo grafico dopo il comando **set term**.

La finestra del grafico attiva (quella selezionata da **set term wxt** <**n**>) è interattiva. Il suo comportamento è condiviso con altri tipi di terminale. Vedere **mouse (p. 178)** per i dettagli. Possiede anche alcune icone extra, che dovrebbero essere autoesplicative.

Questo terminale supporta una modalità di testo avanzato, che permette ai font e ad altri comandi di formattazione (pedici, apici, ecc.) di essere incorporati in etichette e altre stringhe di testo. La sintassi della modalità di testo avanzato è condivisa con altri tipi di terminale di gnuplot. Vedere **enhanced (p. 35)** per maggiori dettagli.

<font> è nel formato "FontFace,FontSize", cioè la variante (face) e la dimensione separate da una virgola in una singola stringa. FontFace è un normale nome di variante di font, come 'Arial'. Se non si fornisce FontFace, il terminale wxt userà 'Sans'. FontSize e la dimensione del font, in punti. Se non viene fornita, il terminale wxt userà una dimensione di 10 punti.

```
Per esempio:
   set term wxt font "Arial,12"
   set term wxt font "Arial" # to change the font face only
   set term wxt font ",12" # to change the font size only
   set term wxt font "" # to reset the font name and size
```

I font sono recuperati dai soliti sottosistemi di font. In Windows, questi font devono essere trovati e configurati nella entry "Fonts" del pannello di controllo. In UNIX, sono gestiti da "fontconfig".

Pango, la libreria usata per impaginare il testo, si basa su utf-8. Quindi, il terminale wxt deve convertire dalla codifica dell'utente a utf-8. La codifica dell'input predefinita è basata sul proprio 'locale'. Se si desidera usare un'altra codifica, è necessario assicurarsi che gnuplot sappia quale si sta usando. Vedere **encoding (p. 154)** per ulteriori dettagli.

Pango può dare risultati inaspettati con i font che non rispettano la mappatura unicode. Con il font Symbol, per esempio, il terminale wxt userà la mappa fornita da http://www.unicode.org/ per tradurre i codici dei caratteri in unicode. Pango farà del suo meglio per trovare un font contenente questo carattere, cercando il vostro font Symbol, o altri font con un'ampia copertura unicode, come i font DejaVu. Notare che "il font Symbol" deve essere inteso come il font Symbol di Adobe, distribuito con Acrobat Reader come "SY_____.PFB". In alternativa, il font OpenSymbol, distribuito con OpenOffice.org come "opens___.ttf", offre gli stessi caratteri. Microsoft ha distribuito un font Symbol ("symbol.ttf"), ma ha un set di caratteri diversi con vari caratteri matematici mancanti o spostati. Se si verificano problemi con la configurazione predefinita (se la demo enhancedtext.dem non viene visualizzata correttamente, per esempio), probabilmente è necessario installare uno dei font Symbol di Adobe o OpenOffice, e rimuovere quello di Microsoft. È stato osservato che altri font non conformi, come "wingdings", funzionano.

Il rendering del grafico può essere modificato tramite una finestra di dialogo disponibile dalla barra degli strumenti. Per ottenere il miglior output possibile, il rendering coinvolge tre meccanismi: antialiasing, oversampling e hinting. L'antialiasing permette di visualizzare le linee non orizzontali e non verticali in modo più fluido. L'oversampling abbinato all'antialiasing fornisce una precisione subpixel, in modo che gnuplot possa disegnare una linea da coordinate non intere. Questo evita effetti di oscillazione sulle linee diagonali ('plot x', per esempio). Hinting evita la sfocatura sulle linee orizzontali e verticali causata dall'oversampling. Il terminale farà scattare queste linee alle coordinate intere in modo che una linea larga un pixel sia effettivamente disegnata su uno e un solo pixel.

Per default, la finestra viene portata in cima al desktop quando viene disegnato un grafico. Ciò può essere controllato con la keyword "raise". La keyword "persist" impedirà a gnuplot di uscire prima che vengano chiuse esplicitamente tutte le finestre dei grafici. Infine, per default il tasto <space> solleva la finestra della console di gnuplot, e 'q' chiude la finestra del grafico. La keyword "ctrl" permette di sostituire questi

collegamenti con <ctrl>+<space> e <ctrl>+'q', rispettivamente. Queste tre keyword (raise, persist and ctrl) possono anche essere impostate e ricordate tra le sessioni attraverso la finestra di configurazione.

X11

Sintassi:

```
set terminal x11 {<n> | window "<string>"}
                 {title "<string>"}
                 {{no}enhanced} {font <fontspec>}
                 {linewidth LW}
                 {{no}persist} {{no}raise} {{no}ctrlq}
                 {{no}replotonresize}
                 {close}
                 {size XX,YY} {position XX,YY}
set terminal x11 {reset}
```

Sono supportate più finestre di grafici: **set terminal x11 <n>** dirige l'output alla finestra di grafico numero n. Se n non è 0, il numero del terminale sarà aggiunto al titolo della finestra (a meno che non sia stato fornito manualmente un titolo) e l'icona sarà etichettata **Gnuplot <n>**. La finestra attiva può essere contraddistinta da un cambiamento del cursore (da predefinito a mirino).

Il terminale **x11** può connettersi a finestre X precedentemente create da un'applicazione esterna tramite l'opzione **window** seguita da una stringa contenente l'ID X della finestra in formato esadecimale. Gnuplot usa quella finestra X esterna come contenitore, poiché X non permette a più client di selezionare l'evento ButtonPress. In questo modo, le funzioni del mouse di gnuplot funzionano all'interno della finestra di grafico contenuta.

```
set term x11 window "220001e"
```

Il terminale x11 supporta la modalità di testo avanzato (vedere **enhanced (p. 35)**), a seconda dei font disponibili. Affinché i comandi font size incorporati nel testo abbiano qualche effetto, il font predefinito x11 deve essere ridimensionabile. Quindi il primo esempio qui sotto funzionerà come previsto, ma il secondo no.

```
set term x11 enhanced font "arial,15"
set title '{/=20 Big} Medium {/=5 Small}'

set term x11 enhanced font "terminal-14"
set title '{/=20 Big} Medium {/=5 Small}'
```

Le finestre dei grafici rimangono aperte anche quando il driver **gnuplot** viene spostato su un dispositivo diverso. Una finestra di grafico può essere chiusa premendo la lettera 'q' mentre quella finestra ha il focus dell'input o scegliendo **close** da un menu window manager. Tutte le finestre dei grafici possono essere chiuse specificando **reset**, che termina effettivamente il sottoprocesso che mantiene le finestre (a meno che non sia specificato **-persist**). Il comando **close** può essere usato per chiudere singole finestre di grafico in base al numero. Tuttavia, dopo un **reset**, le finestre dei grafici rimaste a causa di persist non possono essere chiuse con il comando **close**. **close** senza un numero chiude tutte le finestre dei grafici attualmente attive.

Il driver esterno di gnuplot, gnuplot_x11, viene cercato in un posto predefinito scelto quando il programma viene compilato. È possibile sovrascrivere ciò definendo la variabile d'ambiente GNUPLOT_DRIVER_DIR in modo che punti a una posizione diversa.

Le finestre dei grafici saranno chiuse automaticamente alla fine della sessione a meno che non sia stata data l'opzione **-persist**.

Le opzioni **persist** e **raise** non sono impostate di default, il che significa che vengono presi i valori predefiniti (persist == no e raise == yes) o le opzioni della linea di comando -persist / -raise o le Xresources. Se sono specificati [no]persist o [no]raise, essi sovrascriveranno le opzioni della linea di comando e Xresources.

L'impostazione di una di queste opzioni avviene immediatamente, quindi il comportamento di un driver già in esecuzione può essere modificato. Se la finestra non viene sollevata, vedere la discussione in **raise** **(p. 134)**.

L'opzione **replotonresize** (attiva per default) reimposta (replot) i dati quando la finestra del grafico viene ridimensionata. Senza questa opzione, il ridimensionamento del rapporto di aspetto uniforme (even-aspect-ratio) può portare il grafico a riempire solo una parte della finestra dopo il ridimensionamento. Con questa opzione, gnuplot fa un replot completo su ogni evento di ridimensionamento (resize), portando a un migliore utilizzo dello spazio. Questa opzione è generalmente desiderabile, a meno che il replotting potenzialmente impegnativo per la CPU durante il ridimensionamento sia una preoccupazione. I replot possono essere avviati manualmente con il tasto di scelta rapida 'e' o con il comando 'replot'.

L'opzione **title "<title name>"** fornirà il nome del titolo della finestra per la finestra del grafico corrente o la finestra del grafico <n> se viene dato un numero. Dove (o se) questo titolo viene mostrato dipende dal proprio window manager X.

L'opzione size può essere usata per impostare la dimensione della finestra del grafico. L'opzione size verrà applicata solo alle finestre appena create.

L'opzione position può essere usata per impostare la posizione della finestra del grafico. L'opzione position verrà applicata solo alle finestre appena create.

La dimensione o il rapporto di aspetto di un grafico possono essere cambiati ridimensionando la finestra **gnuplot**.

Lo spessore delle linee e le dimensioni dei punti possono essere cambiati dall'interno di **gnuplot** con **set linestyle**.

Per il tipo di terminale **x11**, **gnuplot** accetta (quando inizializzate) le opzioni e le risorse standard di X Toolkit come geometria, font, e il nome dagli argomenti della linea di comando o da un file di configurazione. Vedere la pagina man di X(1) (o il suo equivalente) per una descrizione di tali opzioni.

Sono disponibili diverse altre opzioni di **gnuplot** per il terminale **x11**. Queste possono essere specificate sia come opzioni della linea di comando quando è invocato **gnuplot** o come risorse nel file di configurazione ".Xdefaults". Sono impostate al momento dell'inizializzazione e non possono essere modificate durante una sessione di **gnuplot**. (eccetto **persist** e **raise**)

X11_fonts

All'avvio, il font predefinito è preso dalle risorse di X11 come impostate nel file .Xdefaults di sistema o dell'utente o sulla linea di comando.

Esempio:

```
gnuplot*font: lucidasans-bold-12
```

Un nuovo font predefinito può essere specificato al driver x11 dall'interno di gnuplot usando

```
`set term x11 font "<fontspec>"`
```

Il driver prima richiede al server X un font con il preciso nome fornito. Se questa richiesta fallisce, allora cerca di interpretare <fontspec> come "<font>,<size>,<slant>,<weight>" e di costruire un nome di font X11 completo della forma

```
-*-<font>-<weight>-<s>-*-*-<size>-*-*-*-*-*-<encoding>
```

```
<font> è il nome base del font (ad es. Times o Symbol)
<size> è la dimensione del punto (predefinito a 12 se non è specificata)
<s> è `i` se <slant>=="italic", `o` se <slant>=="oblique", altrimenti `r`
<weight> è `medium` o `bold` se esplicitamente richiesto, altrimenti `*`
<encoding> è impostato in base al set di caratteri corrente (vedere `set encoding`).
```

Quindi **set term x11 font "arial,15,italic"** sarà tradotto in -*-arial-*-i-*-*-15-*-*-*-*-*-iso8859-1 (assumendo la codifica predefinita). Le specifiche <size>, <slant>, e <weight> sono tutte facoltative. Se non si specifica <slant> o <weight> allora si otterrà qualsiasi variante del font che il server di font offre per prima. È possibile impostare una codifica predefinita tramite la risorsa X11 corrispondente. Per esempio

```
gnuplot*encoding: iso8859-15
```

Il driver riconosce anche alcuni nomi comuni di font PostScript e li sostituisce con possibili equivalenti X11 o TrueType. Questa stessa sequenza è usata per processare le richieste per i font da **set label**.

Se il proprio gnuplot è stato costruito con l'opzione di configurazione –enable-x11-mbfonts, è possibile specificare i font multibyte usando il prefisso "mbfont:" sul nome del font. Può essere dato un font aggiuntivo, separato da un punto e virgola. Poiché le codifiche dei font multibyte sono interpretate in base all'impostazione del locale, è necessario assicurarsi che la variabile di ambiente LC_CTYPE sia impostata su qualche valore di locale appropriato come ja_JP.eucJP, ko_KR.EUC, o zh_CN.EUC.

Esempio:

```
set term x11 font 'mbfont:kana14;k14'
      # 'kana14' ed 'k14' sono alias di font X11 giapponesi, e ';'
      # è il separatore di nomi di font.
set term x11 font 'mbfont:fixed,16,r,medium'
      # anche la forma <font>,<size>,<slant>,<weight> è utilizzabile.
set title '(mb strings)' font 'mbfont:*-fixed-medium-r-normal--14-*'
```

La stessa sintassi si applica al font predefinito nelle impostazioni di Xresources, per esempio,

```
gnuplot*font: \
    mbfont:-misc-fixed-medium-r-normal--14-*-*-*-c-*-jisx0208.1983-0
```

Se gnuplot è costruito con –enable-x11-mbfonts, è possibile utilizzare due nomi di font PostScript speciali 'Ryumin-Light-*' e 'GothicBBB-Medium-*' (font PS giapponesi standard) senza il prefisso "mbfont:".

Command-line_options

Oltre alle opzioni di X Toolkit, le seguenti opzioni possono essere specificate sulla linea di comando quando si avvia **gnuplot** o come risorse nel proprio file ".Xdefaults" (si noti che **raise** e **persist** possono essere sovrascritte in seguito da **set term x11 [no]raise [no]persist)**:

'-mono'	forza il rendering monocromatico sui display a colori.
'-gray'	richiede il rendering in scala di grigi su display in scala di grigi o a colori. (I display in scala di grigi ricevono, per default, un rendering monocromatico.)
'-clear'	richiede che la finestra venga momentaneamente cancellata prima che venga visualizzato un nuovo grafico.
'-tvtwm'	richiede che le specifiche geometriche per la posizione della finestra siano fatte rispetto alla porzione attualmente visualizzata del root virtuale.
'-raise'	alza la finestra del grafico dopo ogni grafico.
'-noraise'	non alza la finestra del grafico dopo ogni grafico.
'-persist'	le finestre dei grafici sopravvivono dopo l'uscita del programma gnuplot principale.

Le opzioni sono mostrate sopra nella loro sintassi della linea di comando. Quando sono inserite come risorse in ".Xdefaults", richiedono una sintassi diversa.

Esempio:

```
gnuplot*gray:  on
gnuplot*ctrlq: on
```

gnuplot fornisce anche un'opzione di linea di comando (**-pointsize** <**v**>) e una risorsa, **gnuplot*pointsize:** <**v**>, per controllare la dimensione dei punti plottati con lo stile di plotting **points**. Il valore **v** è un numero reale (maggiore di 0 e minore o uguale a dieci) usato come fattore di scala per le dimensioni dei punti. Per esempio, **-pointsize 2** usa punti grandi il doppio della dimensione predefinita, e **-pointsize 0.5** usa punti grandi la metà della dimensione normale.

Lo switch **-ctrlq** cambia il tasto di scelta rapida che chiude una finestra di grafico da **q** a <**ctrl**>**q**. Questo è utile se si sta usando la funzione di cattura dei tasti (keystroke-capture) **pause mouse keystroke**, poiché permette di acquisire il carattere **q** come tutti gli altri caratteri alfanumerici. Lo switch **-ctrlq** sostituisce il tasto di scelta rapida <space> con <ctrl><space> per lo stesso motivo.

Color_resources

NB: QUESTA SEZIONE È SOSTANZIALMENTE IRRILEVANTE NELLA VERSIONE 5 DI GNUPLOT
Il terminale X11 rispetta le seguenti risorse (mostrate qui con i loro valori predefiniti) o le risorse in scala di grigi. Questi valori possono essere nomi di colori come elencati nel file X11 rgb.txt sul proprio sistema, specifiche esadecimali di colori RGB (vedere documentazione X11), o il nome di un colore seguito da una virgola e da un valore **intensity** (intensità) da 0 a 1. Per esempio, **blue, 0.5** significa un blu a mezza intensità.

```
gnuplot*background: white
gnuplot*textColor: black
gnuplot*borderColor: black
gnuplot*axisColor: black
gnuplot*line1Color: red
gnuplot*line2Color: green
gnuplot*line3Color: blue
gnuplot*line4Color: magenta
gnuplot*line5Color: cyan
gnuplot*line6Color: sienna
gnuplot*line7Color: orange
gnuplot*line8Color: coral
```

La sintassi della riga di comando per questi è semplice solo per lo sfondo, che corrisponde direttamente alla solita opzione del toolkit X11 "-bg". Tutti gli altri possono essere impostati solo sulla linea di comando usando l'opzione generica "-xrm" per sovrascrivere le risorse

Esempi:

```
gnuplot -background coral
```

per cambiare il colore dello sfondo.

```
gnuplot -xrm 'gnuplot*line1Color:blue'
```

per sovrescrivere il colore del primo linetype.

Grayscale_resources

Quando viene selezionato **-gray**, **gnuplot** rispetta le risorse seguenti per i display in scala di grigi o a colori (mostrati qui con i loro valori predefiniti). Si noti che lo sfondo predefinito è nero.

```
gnuplot*background: black
gnuplot*textGray: white
gnuplot*borderGray: gray50
gnuplot*axisGray: gray50
gnuplot*line1Gray: gray100
gnuplot*line2Gray: gray60
gnuplot*line3Gray: gray80
gnuplot*line4Gray: gray40
gnuplot*line5Gray: gray90
gnuplot*line6Gray: gray50
gnuplot*line7Gray: gray70
gnuplot*line8Gray: gray30
```

Line_resources

NB: QUESTA SEZIONE È SOSTANZIALMENTE IRRILEVANTE NELLA VERSIONE 5 DI GNUPLOT
gnuplot rispetta le seguenti risorse per impostare lo spessore (in pixel) delle linee del grafico (mostrate qui con i loro valori predefiniti). 0 o 1 significa uno spessore di linea minimo di 1 pixel. Un valore di 2 o 3 può migliorare l'aspetto di alcuni grafici.

```
gnuplot*borderWidth: 1
gnuplot*axisWidth: 0
gnuplot*line1Width: 0
gnuplot*line2Width: 0
gnuplot*line3Width: 0
gnuplot*line4Width: 0
gnuplot*line5Width: 0
gnuplot*line6Width: 0
gnuplot*line7Width: 0
gnuplot*line8Width: 0
```

gnuplot rispetta le seguenti risorse per impostare lo stile del trattino (dash) usato per le linee di plotting. 0 significa una linea continua. Un numero a due cifre **jk** (**j** e **k** sono >= 1 e <= 9) significa una linea tratteggiata con un pattern ripetuto di **j** pixel on seguito da **k** pixel off. Per esempio, '16' è una linea punteggiata con un pixel on seguito da sei pixel off. Pattern on/off più elaborati possono essere specificati con un valore a quattro cifre. Per esempio, '4441' è quattro on, quattro off, quattro on, uno off. I valori predefiniti mostrati qui sotto sono per i display monocromatici o rendering monocromatici su display a colori o in scala di grigi. I display a colori sono per default dashed:off

```
gnuplot*dashed: off
gnuplot*borderDashes: 0
gnuplot*axisDashes: 16
gnuplot*line1Dashes: 0
gnuplot*line2Dashes: 42
gnuplot*line3Dashes: 13
gnuplot*line4Dashes: 44
gnuplot*line5Dashes: 15
gnuplot*line6Dashes: 4441
gnuplot*line7Dashes: 42
gnuplot*line8Dashes: 13
```

X11 pm3d_resources

NB: QUESTA SEZIONE È SOSTANZIALMENTE IRRILEVANTE NELLA VERSIONE 5 DI GNUPLOT
Scegliere la visual class e il numero di colori appropriati è un punto cruciale nelle applicazioni X11 e un po' difficile, poiché X11 supporta sei visual type in diverse profondità (depth).

Per default **gnuplot** utilizza la visual (visualizzazione) predefinita dello schermo. Il numero di colori che possono essere assegnati dipende dalla visual class scelta. Su una visual class con una profondità > 12bit, gnuplot inizia con un numero massimo di 0x200 colori. Su una visual class con una profondità > 8bit (ma <= 12 bit) il numero massimo di colori è 0x100, su display <= 8bit il numero massimo di colori è 240 (16 sono lasciati per i colori delle linee)

Gnuplot inizia prima ad assegnare il numero massimo di colori come indicato sopra. Se questo non funziona, il numero di colori viene ridotto dal fattore 2 fino a quando gnuplot ottiene tutti i colori richiesti. Se dividendo ripetutamente **maxcolors** per 2 si ottiene un numero inferiore a **mincolors**, **gnuplot** cerca di installare una colormap privata. In questo caso il window manager è responsabile dello scambio delle colormap quando il puntatore viene spostato dentro e fuori la finestra del driver x11.

Il default per **mincolors** è maxcolors / (num_colormaps > 1 ? 2 : 8), dove num_colormaps è il numero di colormap attualmente utilizzate da gnuplot (di solito 1, se è aperta solo una finestra x11).

Alcuni sistemi supportano più visual class (diverse) su uno schermo. Su questi sistemi potrebbe essere necessario costringere gnuplot a usare una visual class specifica, ad esempio la visual predefinita potrebbe essere 8bit PseudoColor ma lo schermo supporterebbe anche 24bit TrueColor che sarebbe la scelta da preferire.

Le informazioni riguardanti le capacità di un Xserver possono essere ottenute con il programma **xdpyinfo**. È possibile scegliere uno dei seguenti nomi visual StaticGray, GrayScale, StaticColor, PseudoColor, TrueColor, DirectColor. Se un Xserver supporta un determinato visual type a diverse profondità, **gnuplot** sceglie la visual class con la profondità più alta (più profonda). Se la visual class richiesta corrisponde alla visual predefinita e sono supportate più classi di questo tipo, viene preferibile la visual predefinita.

Esempio: è possibile forzare una color map privata su una visual PseudoColor a 8bit specificando **gnuplot*maxcolors: 240** e **gnuplot*mincolors: 240**.

```
gnuplot*maxcolors: intero
gnuplot*mincolors: intero
gnuplot*visual: visual name
```

X11 other_resources

Per default il contenuto dell'attuale finestra di grafico viene esportato negli appunti di X11 in risposta agli eventi X nella finestra. Impostando la risorsa 'gnuplot*exportselection' su 'off' o 'false' disabiliterà questa funzione.

Per default la rotazione del testo viene eseguita utilizzando un metodo che è veloce, ma può corrompere i colori vicini a seconda dello sfondo. Se questo è un problema, è possibile impostare la risorsa 'gnuplot.fastrotate' su 'off'.

```
gnuplot*exportselection: off
gnuplot*fastrotate: on
gnuplot*ctrlq: off
```

Xlib

Il driver del terminale **xlib** supporta X11 Windows System. Genera comandi gnuplot_x11, ma li invia al file di output specificato da **set output '<filename>'**. **set term x11** equivale a **set output "|gnuplot_x11 -noevents"; set term xlib**. **xlib** prende lo stesso set di opzioni di **x11**.

Part V
Bug

Si prega di inviare segnalazioni di bug via e-mail alla mailing list gnuplot-bugs o caricare la segnalazione sul sito web di gnuplot su SourceForge. Per favore, fornite informazioni complete sulla versione di gnuplot che state usando e, se possibile, uno script di prova che dimostri il bug. Vedere **richiesta assistenza (p. 23)**.

Limitazioni note

Non è possibile utilizzare dati in linea (ad esempio, plot '-' ...) all'interno delle parentesi graffe di un ciclo **do** o **while**.

Terminale X11: È difficile selezionare i font UTF-8. Solo una palette di colori alla volta è attiva per qualsiasi finestra di grafico x11. Questo significa che i multiplot i cui grafici costitutivi usano diverse palette non verranno visualizzati correttamente in x11.

Terminale Qt: la rotazione 3D di poligoni e superfici può essere lenta; questa è fortemente influenzata dalla modalità di rendering di Qt (vedere documentazione Qt).

I comandi **raise** e **lower** sono inaffidabili.

Librerie esterne

Libreria esterna GD (usata dai terminali PNG/JPEG/GIF/sixelgd, pixmap): Le versioni di libgd fino alla 2.0.33 avevano vari bug nella mappatura dei caratteri del font Symbol di Adobe. I pattern punto-linea non sono supportati per questi terminali.

Libreria esterna PDFlib (usata dal terminale PDF, ma non da pdfcairo): Gnuplot può essere collegato alle versioni 4, 5 o 6 di libpdf. Tuttavia, queste versioni differiscono nella loro gestione del piped I/O. Perciò gli script di gnuplot che utilizzano l'output piped in PDF possono funzionare solo per alcune versioni di PDFlib.

Internazionalizzazione (impostazioni locali): Gnuplot usa la routine setlocale() della libreria C runtime per controllare la formattazione specifica al locale delle stringhe di numeri, orari e date in input e in output. I locale disponibili, e il livello di supporto per le funzionalità del locale come "thousands' grouping separator" (separatore di raggruppamento delle migliaia), dipendono dal supporto di internazionalizzazione fornito dalla propria macchina individuale.

Le versioni 1.8 1.9 1.10 della libreria esterna libcerf restituiscono risultati errati per la funzione voigt. Non usare.

Part VI
Index

Index