gnuplot

An Interactive Plotting Program

Thomas Williams & Colin Kelley

Version 3.7 organized by: David Denholm

Major contributors (alphabetic order):
Hans-Bernhard Broeker
John Campbell
Robert Cunningham
David Denholm
Gershon Elber
Roger Fearick
Carsten Grammes
Lucas Hart
Lars Hecking
Thomas Koenig
David Kotz
Ed Kubaitis
Russell Lang
Alexander Lehmann
Alexander Mai
Carsten Steger
Tom Tkacik
Jos Van der Woude
James R. Van Zandt
Alex Woo

Copyright (C) 1986 - 1993, 1998 Thomas Williams, Colin Kelley

Mailing list for comments: info-gnuplot@dartmouth.edu
Mailing list for bug reports: bug-gnuplot@dartmouth.edu

This manual was prepared by Dick Crawford.
3 December 1998

目 次

第 I 部　Gnuplot **1**

1　Copyright **1**

2　Introduction **2**

3　Seeking-assistance **3**

4　What's New in version 3.7 **4**

5　Batch/Interactive Operation **5**

6　Command-line-editing **6**

7　Comments **7**

8　Coordinates **7**

9　Environment **7**

10　Expressions **8**

10.1　Functions 9

10.2　Operators 11

10.2.1　Unary 11

10.2.2　Binary 12

10.2.3　Ternary 12

10.3　User-defined 13

11　Glossary **14**

12　Plotting **15**

13　Start-up **15**

14　Substitution **15**

15　Syntax **16**

16　Time/Date data **17**

第 II 部 Commands **18**

17 Cd **18**

18 Call **19**

19 Clear **19**

20 Exit **20**

21 Fit **20**

21.1 Adjustable parameters . . . 21

21.2 Beginner's guide . . . 22

21.3 Error estimates . . . 23

21.3.1 Statistical overview . . . 23

21.3.2 Practical guidelines . . . 24

21.4 Fit controlling . . . 25

21.4.1 Control variables . . . 25

21.4.2 Environment variables . . . 26

21.5 Multi-branch . . . 26

21.6 Starting values . . . 26

21.7 Tips . . . 27

22 Help **28**

23 If **28**

24 Load **29**

25 Pause **29**

26 Plot **30**

26.1 Data-file . . . 30

26.1.1 Every . . . 31

26.1.2 Example datafile . . . 32

26.1.3 Index . . . 32

26.1.4 Smooth . . . 33

26.1.4.1 Acsplines . . . 33

26.1.4.2 Bezier . . . 34

26.1.4.3 Csplines . . . 34
26.1.4.4 Sbezier . . . 34
26.1.4.5 Unique . . . 34
26.1.5 Special-filenames . . . 34
26.1.6 Thru . . . 35
26.1.7 Using . . . 36
26.2 Errorbars . . . 38
26.3 Parametric . . . 38
26.4 Ranges . . . 39
26.5 Title . . . 40
26.6 With . . . 41

27 Print **42**

28 Pwd **42**

29 Quit **42**

30 Replot **43**

31 Reread **43**

32 Reset **44**

33 Save **44**

34 Set-show **45**
34.1 Angles . . . 45
34.2 Arrow . . . 45
34.3 Autoscale . . . 47
34.3.1 Parametric mode . . . 48
34.3.2 Polar mode . . . 48
34.4 Bar . . . 48
34.5 Bmargin . . . 49
34.6 Border . . . 49
34.7 Boxwidth . . . 50
34.8 Clabel . . . 50
34.9 Clip . . . 51

34.10 Cntrparam 52
34.11 Contour 53
34.12 Data style 54
34.13 Dgrid3d 54
34.14 Dummy 55
34.15 Encoding 56
34.16 Format 56
34.16.1 Format specifiers 57
34.16.2 Time/date specifiers 58
34.17 Function style 59
34.18 Functions 59
34.19 Grid 59
34.20 Hidden3d 60
34.21 Isosamples 62
34.22 Key 62
34.23 Label 64
34.24 Linestyle 65
34.25 Lmargin 66
34.26 Locale 66
34.27 Logscale 67
34.28 Mapping 67
34.29 Margin 68
34.30 Missing 68
34.31 Multiplot 69
34.32 Mx2tics 70
34.33 Mxtics 70
34.34 My2tics 70
34.35 Mytics 70
34.36 Mztics 71
34.37 Offsets 71
34.38 Origin 71
34.39 Output 71
34.40 Parametric 72
34.41 Pointsize 73

34.42 Polar . . . 73
34.43 Rmargin . . . 74
34.44 Rrange . . . 74
34.45 Samples . . . 74
34.46 Size . . . 75
34.47 Style . . . 75
34.47.1 Boxerrorbars . . . 76
34.47.2 Boxes . . . 76
34.47.3 Boxxyerrorbars . . . 77
34.47.4 Candlesticks . . . 77
34.47.5 Dots . . . 77
34.47.6 Financebars . . . 77
34.47.7 Fsteps . . . 77
34.47.8 Histeps . . . 78
34.47.9 Impulses . . . 78
34.47.10 Lines . . . 78
34.47.11 Linespoints . . . 78
34.47.12 Points . . . 78
34.47.13 Steps . . . 78
34.47.14 Vector . . . 79
34.47.15 Xerrorbars . . . 79
34.47.16 Xyerrorbars . . . 79
34.47.17 Yerrorbars . . . 79
34.48 Surface . . . 79
34.49 Terminal . . . 80
34.49.1 Aed767 . . . 80
34.49.2 Aifm . . . 80
34.49.3 Amiga . . . 81
34.49.4 Apollo . . . 81
34.49.5 Atari ST (via AES) . . . 81
34.49.6 Atari ST (via VDI) . . . 82
34.49.7 Be . . . 82
34.49.7.1 Command-line_options . . . 83
34.49.7.2 Monochrome_options . . . 83

34.49.7.3 Color_resources . . . 83
34.49.7.4 Grayscale_resources . . . 84
34.49.7.5 Line_resources . . . 84
34.49.8 Cgi . . . 85
34.49.9 Cgm . . . 85
34.49.9.1 Font . . . 86
34.49.9.2 Fontsize . . . 88
34.49.9.3 Linewidth . . . 88
34.49.9.4 Rotate . . . 88
34.49.9.5 Solid . . . 88
34.49.9.6 Size . . . 88
34.49.9.7 Width . . . 89
34.49.9.8 Nofontlist . . . 89
34.49.10 Corel . . . 89
34.49.11 Debug . . . 89
34.49.12 Svga . . . 90
34.49.13 Dumb . . . 90
34.49.14 Dxf . . . 90
34.49.15 Dxy800a . . . 90
34.49.16 Eepic . . . 90
34.49.17 Emf . . . 92
34.49.18 Emxvga . . . 92
34.49.19 Epslatex . . . 93
34.49.20 Epson-180dpi . . . 93
34.49.21 Excl . . . 94
34.49.22 Hercules . . . 94
34.49.23 Fig . . . 94
34.49.24 Ggi . . . 96
34.49.25 Gif . . . 96
34.49.26 Unixplot . . . 97
34.49.27 Gpic . . . 97
34.49.28 Gpr . . . 98
34.49.29 Grass . . . 98
34.49.30 Hp2623a . . . 98

34.49.31 Hp2648 . . . 98
34.49.32 Hp500c . . . 99
34.49.33 Hpgl . . . 99
34.49.34 Hpljii . . . 100
34.49.35 Hppj . . . 100
34.49.36 Imagen . . . 100
34.49.37 Iris4d . . . 101
34.49.38 Kyo . . . 101
34.49.39 Latex . . . 102
34.49.40 Linux . . . 102
34.49.41 Lips . . . 103
34.49.42 Macintosh . . . 103
34.49.43 Mf . . . 104
34.49.43.1 METAFONT Instructions . . . 104
34.49.44 Mp . . . 105
34.49.44.1 Metapost Instructions . . . 107
34.49.45 Mgr . . . 107
34.49.46 Mif . . . 108
34.49.47 Mtos . . . 108
34.49.48 Next . . . 108
34.49.49 Next . . . 109
34.49.50 Pbm . . . 109
34.49.51 Dospc . . . 110
34.49.52 Pdf . . . 110
34.49.53 Pm . . . 110
34.49.54 Png . . . 111
34.49.55 Postscript . . . 111
34.49.55.1 Enhanced postscript . . . 112
34.49.55.2 Editing postscript . . . 113
34.49.55.3 Postscript plus . . . 113
34.49.56 Pslatex and pstex . . . 114
34.49.57 Pstricks . . . 115
34.49.58 Qms . . . 115
34.49.59 Regis . . . 115

34.49.60 Rgip . . . 115
34.49.61 Sun . . . 116
34.49.62 Svg . . . 116
34.49.63 Tek410x . . . 116
34.49.64 Table . . . 116
34.49.65 Tek40 . . . 116
34.49.66 Texdraw . . . 117
34.49.67 Tgif . . . 117
34.49.68 Tkcanvas . . . 118
34.49.69 Tpic . . . 119
34.49.70 Unixpc . . . 119
34.49.71 Unixplot . . . 119
34.49.72 Vx384 . . . 120
34.49.73 VWS . . . 120
34.49.74 Windows . . . 120
34.49.74.1 Graph-menu . . . 120
34.49.74.2 Printing . . . 121
34.49.74.3 Text-menu . . . 121
34.49.74.4 Wgnuplot.ini . . . 122
34.49.74.5 Windows3.0 . . . 122
34.49.75 X11 . . . 123
34.49.75.1 Command-line_options . . . 123
34.49.75.2 Monochrome_options . . . 124
34.49.75.3 Color_resources . . . 124
34.49.75.4 Grayscale_resources . . . 125
34.49.75.5 Line_resources . . . 125
34.49.76 Xlib . . . 126
34.50 Tics . . . 126
34.51 Ticslevel . . . 126
34.52 Ticscale . . . 127
34.53 Timestamp . . . 127
34.54 Timefmt . . . 127
34.55 Title . . . 128
34.56 Tmargin . . . 129

34.57 Trange . . . 129
34.58 Urange . . . 129
34.59 Variables . . . 129
34.60 Version . . . 129
34.61 View . . . 130
34.62 Vrange . . . 130
34.63 X2data . . . 130
34.64 X2dtics . . . 130
34.65 X2label . . . 130
34.66 X2mtics . . . 131
34.67 X2range . . . 131
34.68 X2tics . . . 131
34.69 X2zeroaxis . . . 131
34.70 Xdata . . . 131
34.71 Xdtics . . . 132
34.72 Xlabel . . . 132
34.73 Xmtics . . . 133
34.74 Xrange . . . 133
34.75 Xtics . . . 135
34.76 Xzeroaxis . . . 137
34.77 Y2data . . . 137
34.78 Y2dtics . . . 137
34.79 Y2label . . . 137
34.80 Y2mtics . . . 137
34.81 Y2range . . . 137
34.82 Y2tics . . . 137
34.83 Y2zeroaxis . . . 137
34.84 Ydata . . . 138
34.85 Ydtics . . . 138
34.86 Ylabel . . . 138
34.87 Ymtics . . . 138
34.88 Yrange . . . 138
34.89 Ytics . . . 138
34.90 Yzeroaxis . . . 138

34.91 Zdata . . . 138
34.92 Zdtics . . . 138
34.93 Zero . . . 139
34.94 Zeroaxis . . . 139
34.95 Zlabel . . . 139
34.96 Zmtics . . . 139
34.97 Zrange . . . 140
34.98 Ztics . . . 140

35 Shell **140**

36 Splot **140**

36.1 Data-file . . . 141
36.1.1 Binary . . . 142
36.1.2 Example datafile . . . 142
36.1.3 Matrix . . . 143
36.2 Grid_data . . . 143
36.3 Splot_overview . . . 144

37 Test **144**

38 Update **145**

第III部 Graphical User Interfaces **145**

第IV部 Bugs **145**

39 Old_bugs **146**

第 I 部

Gnuplot

1 Copyright

Copyright (C) 1986 - 1993, 1998 Thomas Williams, Colin Kelley

Permission to use, copy, and distribute this software and its documentation for any purpose with or without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation.

Permission to modify the software is granted, but not the right to distribute the complete modified source code. Modifications are to be distributed as patches to the released version. Permission to distribute binaries produced by compiling modified sources is granted, provided you

1. distribute the corresponding source modifications from the released version in the form of a patch file along with the binaries,
2. add special version identification to distinguish your version in addition to the base release version number,
3. provide your name and address as the primary contact for the support of your modified version, and
4. retain our contact information in regard to use of the base software.

Permission to distribute the released version of the source code along with corresponding source modifications in the form of a patch file is granted with same provisions 2 through 4 for binary distributions.

This software is provided "as is" without express or implied warranty to the extent permitted by applicable law.

AUTHORS

Original Software:
Thomas Williams, Colin Kelley.

Gnuplot 2.0 additions:
Russell Lang, Dave Kotz, John Campbell.

Gnuplot 3.0 additions:
Gershon Elber and many others.

(以下おおまかな訳; 訳は正しくないかも知れませんので詳しくは上記の原文を当たってください。訳者は責任を持ちません。)

Copyright (C) 1986 - 1993, 1998 Thomas Williams, Colin Kelley

このソフトウェアとその付属文書の使用、複製、配布の許可は、上記の著作権 (copyright) 表示が、全ての複製物に書かれていること、および著作権表示とこの許諾文の両方がその支援文書に書かれていることを条件とした上で、この文書により保証されます。

このソフトウェアの修正も認められています。しかし、修正を含む全ソースコードの配布の権利は認められません。修正はリリース版に対するパッチの形で配布しなければなりません。修正されたソースをコンパイルして作られたバイナリの配布は、以下の条件の元で認められます:

1. リリース版からのソースの修正部分を、パッチの形でバイナリと共に配布すること
2. ベースとなるリリース版と区別するために、そのバージョン番号に特別なバージョン指定子を付加すること
3. その修正版のサポート用に、あなたの名前とアクセス可能なアドレスとを提供すること
4. ベースとなるソフトウェアの使用に関しては、我々の連絡情報を保持し続けること

リリース版のソースコードを、パッチの形でのソースの修正と一緒に配布することは、バイナリ配布に関する条項 2 から 4 までの条件の元で許されます。

このソフトウェアは "あるがまま" 提供され、適用可能な法律で許められる範囲の保証を表明あるいは暗示していはいません。

著者

オリジナルソフトウェア:

Thomas Williams, Colin Kelley.

Gnuplot 2.0 追加:

Russell Lang, Dave Kotz, John Campbell.

Gnuplot 3.0 追加:

Gershon Elber and many others.

2 Introduction

gnuplot は，コマンド入力方式の対話的な関数描画プログラムです．コマンドや関数名は大文字小文字を区別します．いずれのコマンドも，あいまいさの無い限りにおいて省略することができます．1 行中にはセミコロン (;) で区切って複数のコマンドを書くことができます (ただし、**load** と **call** は最後のコマンドでなければなりません)．文字列は引用符を使って表します．引用符は，一重でも，二重でも構いません．例えば

```
load "filename"
cd 'dir'
```

しかし、両者には微妙な違いがあります (詳細は **syntax** を参照してください)。

コマンドラインでの引数は **gnuplot** 用のコマンドの書かれたファイルの名前であるものとします．但し標準の X11 の引数は例外で，まず最初に処理されます．各ファイルはコマンドライン上で指定された順に **load** コマンドでロードされます．**gnuplot** は，最後に指定されたファイルを処理し終ると終了します．ファイルが 1 つも指定されていない場合は，**gnuplot** は対話モードになります．特別なファイル名 "-" は標準入力を表します。詳細は "help batch/interactive" を参照してください。

gnuplot のコマンドの多くは複数のオプションを持っています。これらのオプションは、ほとんどの場合、不必要なものが省略できるよう、適切な順序で指定することになっています。よって、もしコマンドの全部の指定が"command a b c" である場合、"command a c" は多分うまくいくでしょうが、"command c a" はうまくいかないかもしれません。

コマンドは，複数行にまたがることができます．その場合は，最終行以外の全ての行の行末にバックスラッシュ (\) を書く必要があります．バックスラッシュは必ず各行 *最後* の文字でなくてはなりません．その結果としてバックスラッシュと，それに続く改行文字が存在しなかったかのように扱われます．つまり，改行文字がスペースの役をすることもありませんし，改行によってコメントが終了することもありません．ですから複数行にまたがる行の先頭をコメントアウトすると，そのコマンド全体がコメントアウトされることになります (**comment** 参照)．なお注意しますが、もし、複数行のコマンドのどこかでエラーが起きたとき、パーサはその場所を正確には指示することができませんし、また、正しい行に指示する必要もないでしょう。

このドキュメントにおいて，中括弧 ({}) は省略可能な引数を表すものとし、縦棒 (|) は，互いに排他的な引数を区切るものとします．**gnuplot** のキーワードや **help** における項目名は，バッククオート (`) または可能な場合には **boldface** (太字) で表します．角括弧 (<>) は，それに対応するものに置き換えられるべきものを表します．多くの場合、オプションの引数にはそれが省略されるとデフォルトの値が使用されます。しかし、これらの場合必ずしも角括弧が中括弧で囲まれて書かれているわけではありません。

ある項目についてのヘルプが必要なときには，**help** に続けてその項目名を入力して下さい．または単に **help** や **?** でもヘルプの項目のメニューが現われます。

初めて **gnuplot** を使う方は，**plotting** に関する説明から読みはじめると良いでしょう (現在使用中であれば **help plotting** と入力して下さい)．

3 Seeking-assistance

gnuplot ユーザのためのメーリングリストがあります。しかし、ニュースグループ

```
comp.graphics.apps.gnuplot
```

は、そのメーリングリストと同等であることに注意してください (どちらにも同じメッセージが流れます)。私達はメーリングリストに参加するより、むしろニュースグループのメッセージを読むことを勧めます。メーリングリスト管理者向けのメッセージは

```
majordomo@dartmouth.edu
```

へお送りください。詳細に関しては、メール本文 (Subject ではなく) に 1 語"help" (引用符はなしで) のみを書いたメッセージを送ってください。

メーリングリストメンバーへのメールアドレス:

```
info-gnuplot@dartmouth.edu
```

バグリポート、ソースの改良等は次のところへ:

```
bug-gnuplot@dartmouth.edu
```

β テスト版に関するメーリングリスト:

```
info-gnuplot-beta@dartmouth.edu
```

更新情報、既知のバグ情報を含む WWW ページもあります。

```
http://www.cs.dartmouth.edu/gnuplot_info.html
```

助けを求める前に、次をチェックしてください: FAQ (Frequently Asked Questions) list. もし FAQ のコピーを持っていなければ、email 経由で上記の Majordomo アドレスから、あるいは ftp 経由で次のアドレスから

```
ftp://ftp.ucc.ie/pub/gnuplot/faq,
ftp://ftp.gnuplot.vt.edu/pub/gnuplot/faq,
```

取得できますが、他は **gnuplot** の WWW ページを参照してください。

何か質問を投稿するときは、あなたが使用している **gnuplot** のバージョン、実行マシン、オペレーティングシステム、といった全ての情報を含むようにしてください。その問題を引き起こす _小さい_ スクリプトがあればなお良いです。その場合、データファイルのプロットよりも関数のプロットの方がより良いです。もし、info-gnuplot へメールをするなら、そのメーリングリストの購読をしているかどうかを述べてください。そうすれば、ニュースを見たユーザはあなたへの返事をメールで出せば良いことが分かるでしょうから。そのような記事のポストの form が WWW サイトにあります。

4 What's New in version 3.7

Gnuplot バージョン 3.7 は新しい機能をたくさん備えています。このセクションではそれらの一部分のリストやそれらに関する説明箇所について、順不同で示します。

1. **fit f(x) 'file' via** は Marquardt-Levenberg 法を使ってデータの当てはめを行います (これは version 3.5 に対する **gnufit** パッチとほんの少し違うだけです)。

2. **using** コマンドは大幅に拡張しました。詳しくは **plot using** を見てください。

3. **set timefmt** で、時系列データの入出力時に日付が使用できるようになりました。**Time/Date** の項目、および timedat.dem. を参照してください。

4. いくつかのドライバでの複数行ラベルとフォントの選択

5. 見出し付けされない小目盛り。**set mxtics** 参照

6. 描画ページ内のキー (グラフ見出し) ボックスの移動 (描画範囲の外にまで出せる)、そのタイトル、回りの枠等に関する **key** オプション。**set key** 参照

7. **set multiplot** による単一の描画ページ上での多重描画 (multiplot)

8. **postscript** ドライバの改良による上/下つき文字、フォントの変更 (これは 3.5 のパッチとして存在していた別なドライバ (**enhpost**) だったもの)

9. 第 2 軸: 上と右の軸を下と左の軸とは独立に使い、それぞれに対して描画、目盛りのラベル付けが可能。**plot** 参照

10. 特別なデータファイル名 '-' と "" のサポート。**plot special-filenames** 参照。

11. ラベルと矢 (arrow) に対する座標系を追加

12. **set size** でアスペクト比 (縦横比) の指定を可能に

13. 欠けているデータを正しく扱う **set missing**

14. コマンド **call**: 引数を持つ **load**

15. **reverse**, **writeback**, **restore** キーワードを持つより柔軟性のある **range**

16. 多国語エンコード用の **set encoding**

17. 持続性と複数のウィンドウをサポートした新しい **x11** ドライバ

18. 新しい描画スタイル: **xerrorbars**, **histeps**, **financebars** 等。**set style** 参照

19. 目盛りの見出しの新しい書式。"%l %L" は、見出しの与えられた単位に対する仮数部と指数部に使われます。**set format** 参照。

20. 新しいドライバ: MS-Office アプリケーションに張り込むための **cgm**、WEB 用の **gif** 等。

21. **plot** のグラフの平滑化、およびスプライン補間オプション。**plot smooth** 参照。

22. **set margin** と **set origin** は、描画範囲のどこにグラフを置くかをより良く制御します。

23. **set border** は各境界線を個々に制御可能になりました。

24. 新しいコマンド **if** と **reread** はコマンドループを可能にします。

25. 点のスタイルと大きさ、線の型と幅も **plot** コマンド上で指定できるようになりました。線の型と幅は、grids, borders, tics, arrows の各コマンドでも指定可能です。**plot with** 参照。さらに、それらの型は組み合わせることも可能ですし、再利用のために保存することも可能です。**set linestyle** 参照。

26. 出力ターミナルがサポートする限り、文字列 (ラベル、目盛り見出し、日付) は縦書きも可能。

5 Batch/Interactive Operation

gnuplot は多くのシステム上で、バッチ処理形式、あるいは対話型のどちらの形式でも実行でき、それらを組み合わせることも可能です。

コマンドライン引数は **gnuplot** コマンドを含むファイルのファイル名であると解釈されます (先に指定される標準的な X11 用コマンドの引数を除いて)。各ファイルは、指定された順に **load** コマンドで読み込まれます。最後のファイルを実行した後は **gnuplot** は終了します。ロードファイルを指定しない場合は、**gnuplot** は対話モードに入ります。特別なファイル名 "-" は標準入力を指定するのに使われます。

exit と **quit** はどちらも現在のコマンドファイルを終了し、まだ全てのファイルが終っていなければ、次のファイルを **load** するのに使われます。

例:

対話を開始する:

```
gnuplot
```

2 つのコマンドファイル "input1", "input2" を使ってバッチ処理を行なう:

```
gnuplot input1 input2
```

初期化ファイル "header" の後、対話型モードを起動し、その後別のコマンドファイル "tailer" を実行する:

```
gnuplot header - trailer
```

6 Command-line-editing

コマンドライン編集は Unix, Atari, VMS, MS-DOS and OS/2 上の **gnuplot** でサポートされています。履歴 (ヒストリ) 機能で、以前のコマンドを編集し再実行することも出来ます。コマンドラインの編集後は、カーソルがどこにいても改行や復帰キーによって行全体が入力されます。

(**gnuplot** における readline 関数は、GNU Bash や GNU Emacs で使われる readline 関数と全く同じではありません。もし、GNU 版を望むなら、コンパイル時に **gnuplot** 版の代わりに選択できます)

編集コマンドは以下の通りです:

コマンド行編集コマンド

文字	機能
	行編集
^B	1 文字前へ戻す
^F	1 文字先へ進める
^A	行の先頭へ移動
^E	行の最後へ移動
^H, DEL	直前の文字を削除
^D	現在位置の文字を削除
^K	現在位置から行末まで削除
^L, ^R	壊れた表示の行を再表示
^U	行全体の削除
^W	カーソル手前の単語から行末まで削除
	履歴
^P	前の履歴へ移動
^N	次の履歴へ移動

IBM PC では、行編集用に DOSEDIT とか CED などの TSR (常駐) プログラムを使いたいと思うかも知れません。デフォルトの makefile はこれを仮定していて、**gnuplot** はデフォルトでは行編集機能無しでコンパイルされます。もし **gnuplot** の行編集機能を使用したければ、makefile の READLINE をセットしてリンクファイルとして readline.obj を追加してください。IBM PC と Atari 版で readline を使う場合は以下のキーも使えます。

矢印キー	機能
左 (←)	^B と同じ
右 (→)	^F と同じ
Ctrl + 左	^A と同じ
Ctrl + 右	^E と同じ
上 (↑)	^P と同じ
下 (↓)	^N と同じ

Atari 版の readline は更にいくつかのエイリアスが定義されています:

キー	機能
Undo	`^L` と同じ
Home	`^A` と同じ
Ctrl Home	`^E` と同じ
Esc	`^U` と同じ
Help	'**help**' + return
Ctrl Help	'**help** '

7 Comments

コメントは次のように実装されています: 文字 '#' は，行中のたいていの場所に書くことができます．このとき **gnuplot** はその行の残りの部分を無視します．ただし，引用符の中，数 (複素数を含む) の中，コマンド置換 (command substitution) の中などではこの効果がありません．簡単に言うと，意味のあるような使い方をしさえすれば，正しく動作すると言うことです．

8 Coordinates

コマンド **set arrow**, **set key**, **set label** はグラフ上の任意の位置が指定できます。その位置は以下の書式で指定します:

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

各座標系指定 <system> には、**first**, **second**, **graph**, **screen** のいずれかが入ります。

first は左と下の軸で定義される x,y (3D の場合は z も) の座標系を使用します。**second** は第 2 軸 (上と右の軸) を使用します。**graph** はグラフ描画領域内の相対的位置を指定し、左下が 0,0 で 右上が 1,1 (splot の場合はグラフ描画領域内の左下が 0,0,0 で、土台の位置は負の z の値を使用します。**set ticslevel** 参照) となります。**screen** は表示範囲内 (範囲全体であり、**set size** で選択される一部分ではありません) を指定し、左下が 0,0 で 右上が 1,1 となります。

x の座標系が指定されていない場合は **first** が使われます。y の座標系が指定されていない場合は x に対する座標系が使用されます。

一つ (あるいはそれ以上) の軸が時間軸である場合、**timefmt** の書式文字列に従って、引用符で囲まれた時間文字列で適切な座標を指定する必要があります。**set xdata**, **set timefmt** を参照してください。また、**gnuplot** は整数表記も認めていて、その場合その整数は 2000 年 1 月 1 日からの秒数と解釈されます。

9 Environment

gnuplot は多くのシェル環境変数を認識します。必須のものはありませんが、使えば便利になるかも知れません。

GNUTERM が定義されている場合、それは使用される出力形式 (terminal) の名前として使われます。これは **gnuplot** が起動時に見つけた出力形式に優先して使用されますが、.gnuplot (またはそれに相当する) スタートアップファイル (**start-up** 参照) による指定や、当り前のことですが、その後に明示的に指定し

た物の方が優先されます。

Unix, AmigaOS, AtariTOS, MS-DOS, OS/2 では、GNUHELP にヘルプファイル (gnuplot.gih) のパス名を定義しておくことができます。

VMS では、論理名 GNUPLOT$HELP を **gnuplot** のヘルプライブラリの名前として定義します。**gnuplot** のヘルプは任意のシステムのヘルプライブラリに入れることができ、**gnuplot** の内部からでも外部からでも参照して構いません。

Unix においては、カレントディレクトリに .gnuplot というファイルがない場合には、HOME に定義されたディレクトリを探します。AmigaOS, AtariTOS, MS-DOS, OS/2 では GNUPLOT がその役割に使われます。VMS では SYS$LOGIN です。**help start-up** を参照してください。

Unix においては、PAGER がヘルプメッセージの出力用のフィルタとして使われます。

Unix, AtariTOS, AmigaOS では、SHELL が **shell** コマンドの際に使われます。MS-DOS, OS/2 では COMSPEC が **shell** コマンドの際に使われます。

MS-DOS で BGI または Watcom インターフェースが使われている場合、PCTRM が、使用するモニタがサポートする最大解像度を指示するのに使われます。PCTRM は S< 最大水平解像度 > のように指定します。例えば、モニタの最大解像度が 800x600 ならば、以下のように指定します:

```
set PCTRM=S800
```

PCTRM が設定されていなければ、標準的な VGA (640x480) が使われます。

FIT_SCRIPT は、当てはめ (fit) が中断されたときに実行する **gnuplot** コマンドの指定に使われます。**fit** を参照してください。FIT_LOG は当てはめによるログファイルのファイル名の指定に使われます。

10 Expressions

基本的には C, FORTRAN, Pascal, BASIC において利用可能な数学表現を使用できます。 演算子の優先順位は C 言語の仕様に従います。数式中の空白文字とタブ文字は無視されます。

複素数の定数は {<real>,<imag>} と表現します。ここで <real> と <imag> (実部、虚部) は数値定数である必要があります。例えば {3,2} は 3 + 2i をあらわし、{0,1} は 'i' 自身を表します。これらには明示的に中カッコを使う必要があります。

gnuplot は "実数" と "整数" 演算を FORTRAN や C のように扱うということに注意してください。"1", "-10" などは整数と見なされ、"1.0", "-10.0", "1e1", 3.5e-1 などは実数と見なされます。 これら 2 つのもっとも重要な違いは割算です。整数の割算は切り捨てられます: $5/2 = 2$。実数はそうではありません: $5.0/2.0 = 2.5$。それらが混在した式の場合、計算の前に整数は実数に "拡張" されます: $5/2e0 = 2.5$。負の整数を正の整数で割る場合、その値はコンパイラによって変わります。"print -5/2" として、あなたのシステムが -2 と -3 のどちらを答えとするかを確認してください。

数式 "1/0" は "未定義値 (undefined)" フラグを生成し、それによりその点は無視されます。**ternary** 演算子 (三項演算子) の項にその例があります。

複素数表現の実数部分、虚数部分は、どんな形で入力されても常に実数です: {3,2} の "3" と "2" は実数であり、整数ではありません。

10.1 Functions

gnuplot の関数は、Unix 数学ライブラリの関数とほぼ同じですが、特に注意がなければ全ての関数が整数、実数、複素数の引数を取ることができます。

度、あるいはラジアンのどちらかで角度を引数としたり戻り値としたりする関数 (sin(x), cos(x), tan(x), asin(x), acos(x), atan(x), atan2(x), arg(z)) に対しては、その単位は **set angles** で選択でき、デフォルトはラジアンです。

数学ライブラリ関数		
関数	引数	戻り値
abs(x)	任意	x の絶対値, $\|x\|$; 同じ型
abs(x)	複素数	x の長さ, $\sqrt{\mathrm{real}(x)^2 + \mathrm{imag}(x)^2}$
acos(x)	任意	$\cos^{-1} x$ (アークコサイン)
acosh(x)	任意	ラジアンでの $\cosh^{-1} x$ (逆双曲余弦)
arg(x)	複素数	x の偏角
asin(x)	任意	$\sin^{-1} x$ (アークサイン)
asinh(x)	任意	ラジアンでの $\sinh^{-1} x$ (逆双曲正弦)
atan(x)	任意	$\tan^{-1} x$ (アークタンジェント)
atan2(y,x)	整数または実数	$\tan^{-1}(y/x)$ (アークタンジェント)
atanh(x)	任意	ラジアンでの $\tanh^{-1} x$ (逆双曲正接)
besj0(x)	整数または実数	x の j_0 次ベッセル関数
besj1(x)	整数または実数	x の j_1 次ベッセル関数
besy0(x)	整数または実数	x の y_0 次ベッセル関数
besy1(x)	整数または実数	x の y_1 次ベッセル関数
ceil(x)	任意	$\lceil x \rceil$, x 以上の最小の整数 (real part)
cos(x)	任意	x のコサイン $\cos x$
cosh(x)	任意	$\cosh x$, x のハイパボリックコサイン
erf(x)	任意	erf(real(x)), x の 実部の誤差関数
erfc(x)	任意	erfc(real(x)), 1.0 - (x の実部の誤差関数)
exp(x)	任意	e^x, x の指数関数
floor(x)	任意	$\lfloor x \rfloor$, x (の実部) 以下の最大の整数
gamma(x)	任意	gamma(real(x)), x の実部のガンマ関数
ibeta(p,q,x)	任意	ibeta(real(p, q, x)), p,q,x の実部の不完全ベータ関数
inverf(x)	任意	x の実部の逆誤差関数
igamma(a,x)	任意	igamma(real(a, x)), a,x の実部の不完全ガンマ関数
imag(x)	複素数	x の虚数部分 (実数)
invnorm(x)	任意	x の実部の逆正規分布関数
int(x)	実数	x の実数部分 (0 に向かって丸め)
lgamma(x)	任意	lgamma(real(x)), x の実部のガンマ対数関数
log(x)	任意	$\log_e x$, x の自然対数 (底 e)
log10(x)	任意	$\log_{10} x$, x の対数 (底 10)
norm(x)	任意	x の実部の正規分布 (ガウス分布) 関数
rand(x)	任意	rand(real(x)), 疑似乱数生成器
real(x)	任意	x の実部
sgn(x)	任意	$x > 0$ なら 1, $x < 0$ なら -1, $x = 0$ なら 0. x の虚部は無視
sin(x)	任意	$\sin x$, x のサイン
sinh(x)	任意	$\sinh x$, x のハイパボリックサイン
sqrt(x)	任意	$\sqrt{x}$, x の平方根
tan(x)	任意	$\tan x$, x のタンジェント
tanh(x)	任意	$\tanh x$, x のハイパボリックタンジェント

さらにいくつかの関数が追加されています。

他の **gnuplot** の関数		
関数	引数	返り値
column(x)	整数	データファイル操作での x 列目
tm_hour(x)	整数	時
tm_mday(x)	整数	日
tm_min(x)	整数	分
tm_mon(x)	整数	月
tm_sec(x)	整数	秒
tm_wday(x)	整数	その週の何日目
tm_yday(x)	整数	その年の何日目
tm_year(x)	整数	西暦
valid(x)	整数	データ中の column(x) の正当性

10.2 Operators

gnuplot の演算子は、C 言語の演算子とほぼ同じですが、特に注意がなければ全ての演算子が整数、実数、複素数の引数を取ることができます。また、FORTRAN で使える ** (累乗) 演算子もサポートされています。

演算の評価の順序を変更するにはかっこを使います。

10.2.1 Unary

以下は、単項演算子とその使用法の一覧です:

単項演算子		
記号	例	説明
-	-a	マイナス符号
+	+a	プラス符号 (何もしない)
~	~a	* 1 の補数 (ビット反転)
!	!a	* 論理的否定
!	a!	* 階乗
$	$3	* ‘using‘ 内での引数/列指定

説明に星印 (*) のついた演算子の引数は整数でなければなりません。

演算子の優先順位は Fortran や C と同じです。それらの言語同様、演算の評価される順序を変えるためにかっこが使われます。よって -2**2 = -4 で、(-2)**2 = 4 です。

階乗演算子は、大きな値を返せるように実数を返します。

10.2.2 Binary

以下は、二項演算子とその使用法の一覧です:

二項演算子		
記号	例	説明
**	a**b	累乗
*	a*b	積
/	a/b	商
%	a%b	* 余り
+	a+b	和
-	a-b	差
==	a==b	等しい
!=	a!=b	等しくない
<	a<b	より小さい
<=	a<=b	以下
>	a>b	より大きい
>=	a>=b	以上
&	a&b	* ビット積 (AND)
^	a^b	* ビット排他的論理和 (XOR)
|	a|b	* ビット和 (OR)
&&	a&&b	* 論理的 AND
||	a||b	* 論理的 OR

説明に星印 (*) のついた演算子の引数は整数でなければなりません。

論理演算子の AND (&&) と OR (||) は C 言語同様に必要最小限の評価しかしません。すなわち、&& の第 2 引数は、第 1 引数が偽ならば評価されませんし、|| の第 2 引数は、第 1 引数が真ならば評価されません。

10.2.3 Ternary

一つだけ三項演算子があります:

三項演算子		
記号	例	説明
?:	a?b:c	三項演算子

三項演算子は C のものと同じ働きをします。最初の引数 (a) は整数でなければいけません。この値が評価され、それが真 (ゼロでない) ならば 2 番目の引数 (b) が評価されその値が返され、そうでなければ 3 番目の引数 (c) が評価され、その値が返されます。

三項演算子は、区分的に定義された関数や、ある条件が満たされた場合にのみ点を描画する、といったことを行なう場合に有用です。

例:

0 <= x < 1 では sin(x) に、1 <= x < 2 では 1/x に等しくて、それ以外の x では定義されない関数を描画:

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

gnuplot は未定義値に対しては何も表示せずにただ無視するので、最後の場合の関数 (1/0) は点を何も出力しないことに注意してください。また、この関数描画の描画スタイルが lines (線描画) の場合、不連続点 (x=1) の所も連続関数として線が結ばれてしまうことにも注意してください。その点を不連続になるようにするには、関数を 2 つの部分それぞれに分けてください (このような場合、媒介変数関数を使うのが便利です)。

ファイル 'file' のデータで、4 列目のデータが負でないときだけ、1 列目のデータに関する 2 列目と 3 列目のデータの平均値を描画:

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

using の書式の説明に関しては **plot data-file using** を参照してください。

10.3 User-defined

新たなユーザ定義変数と 1 個から 5 個までの引数を持つユーザ定義関数を、任意の場所で定義したり使ったりすることができます。それは **plot** コマンド上でも可能です。

ユーザ定義関数書式:

```
<func-name>( <dummy1> {,<dummy2>} ... {,<dummy5>} ) = <expression>
```

ここで <expression> は仮変数 <dummy1> から <dummy5> で表される数式です。

ユーザ定義変数書式:

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

例:

```
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)
```

円周率 **pi** は既に定義されています。しかしこれは決して手品のようなものではなく、好きなように再定義することができます。

変数名や関数名の命名規則は、大抵のプログラミング言語と同じで、先頭はアルファベットで、その後の文字はアルファベット、数字、"$", "_" が使えます。ただし、**fit** のサブルーチンでいくつか "FIT_" で始

まる変数を使用することに注意してください。よってそのような名前を使うのは避けるべきでしょう。しかし、**fit** の使用に際しては、例えば "FIT_LIMIT" のように再定義をする必要があるような変数はあります。詳しくは **fit** に関する説明を参照してください。

show functions, **show variables**, **fit** も参照してください。

11 Glossary

このドキュメント全体に渡って、用語に関する一貫性の維持が考えられています。しかしこの試みは完全には成功していません。それは **gnuplot** が時間をかけて進化してきたように、コマンドやキーワードの名前もそのような完全性を排除するかのように採用されて来ているからです。この節では、これらのキーワードのいくつかがどのように使われているかを説明します。

"ページ (page)" または "表示画面 (screen)" は **gnuplot** がアクセス可能な領域全体を指します。ディスプレイモニタでは、これは画面全体を指し、プロッタでは、一枚の紙全体になります。

表示画面は、一つ、またはそれ以上の "グラフ描画 (plot)" を含みます。グラフ描画は一つの横座標と一つの縦座標で定義されますが、余白 (margin) やその中に書かれる文字列 (text) 同様、それらは実際にその上に表示されている必要はありません。

グラフ描画は一つの "グラフ" を含みます。グラフは一つの横座標と一つの縦座標で定義されますが、これらは実際にその上に表示されている必要はありません。

グラフは一つまたはそれ以上の "曲線 (line)" を含みます。曲線は一つの関数、またはデータセットです。用語 "line" は描画スタイルとしても使われます。さらにこの用語は "文字列の一行 (a line of text)" のように使われることもあります。多分文脈からそれらは区別できるでしょう。

一つのグラフ上の複数の曲線はそれぞれ名前を持ちます。その名前は、その曲線の表現に使われる描画スタイルのサンプルとともに "(説明) key" 内に一覧表示されます。説明は、時には "(表題) legend" とも呼ばれます。

用語 "タイトル (title)" は **gnuplot** では複数の意味で使われます。このドキュメントではそれらを区別するために、形容詞として "描画の (plot)"、"曲線の (line)"、"説明の (key)" を頭につけたりもします。

一つのグラフは 4 つまでの見出し付けされる軸を持つことができます。そして **set xlabel** のように、そのコマンド名の中に一つの軸の名前を持つ様々なコマンドがあります。他に、**set logscale xy** のように、オプションに一つまたはそれ以上の軸の名前を持つコマンドもあります。これらの使われ方の中の 4 つの軸の名前はそれぞれ、グラフ描画の下の境界に沿う軸である "x"、左の境界に沿う軸 "y"、上の境界に沿う軸 "x2"、右の境界に沿う軸 "y2" となっています。3 次元描画では "z" もコマンド内に現われます。

データファイルに関する議論では、用語 "行 (record)" を復活し、ファイルの一行の文字列、すなわち、改行文字や行末文字同士の間の文字列、を指し示すのに使います。"点 (point)" は行から取り出した一つのデータです。"データブロック (datablock)" は、空行で区切られた連続した複数の行からなる点の集合です。データファイルの議論の中で "line" が参照される場合は、これはデータブロックの部分集合を指します。

(訳注: この日本語訳の中ではここに書かれているような用語の統一は考慮されてはおらず、よって混乱を引き起こす可能性があります。厳密には原文を参照すべきでしょう。)

12 Plotting

gnuplot には描画を生成する 3 つのコマンド、**plot**, **splot**, **replot** があります。**plot** は 2 次元描画を生成し、**splot** は 3 次元描画 (もちろん実際には 2 次元面への射影) を生成します。**replot** は与えられた引数を、直前の **plot** または **splot** コマンドに追加し、それを実行します。

描画に関する一般的な情報の大半は、**plot** に関する項で見つかります。3 次元描画に固有の情報は **splot** の項にあります。

plot は xy 直交座標系と極座標系が使えます。極座標系の詳細に関しては **set polar** を参照してください。**splot** は xyz 直交座標系のみしか扱えませんが、コマンド **set mapping** で他の 2, 3 の座標系を使用することが出来ます。さらに、オプション **using** を使えば、**plot** でも **splot** でもほとんどどんな座標系でもそれを定義して使うことが出来ます。

splot は点や線による描画に加えて、曲面や等高線を描くこともできます。3 次元の関数の格子定義に関する情報については、**splot** と **set isosamples** の項目を、3 次元データのファイルに必要な形態に関しては **splot datafile** の項目を、等高線に関する情報については **set contour** と **set cntrparam** の項目を参照してください。

13 Start-up

gnuplot が起動されるとき，初期設定ファイルを読み込もうとします . このファイルは Unix と AmigaOS では，**.gnuplot** であり，その他の処理系では **GNUPLOT.INI** となっています . このファイルがカレントディレクトリに無い場合，**gnuplot** はホームディレクトリを検索します (AmigaOS と Atari(single)TOS と MS-DOS と OS/2, MS-DOS では，環境変数 **gnuplot** にホームディレクトリに対応するディレクトリを指定します) . 注意: インストールの時に NOCWDRC を定義した場合には，**gnuplot** はカレントディレクトリからは読みません .

初期設定ファイルが見つかると，**gnuplot** はこのファイルに書かれているコマンドを実行します . ここには任意の正しい **gnuplot** コマンドを書くことが可能ですが、一般的には、出力装置の指定や、よく使う関数や変数の定義を設定する程度に抑えておきます。

14 Substitution

シェルコマンドをバッククォートで囲むことによってコマンド置換を行うことができます。このコマンドは子プロセスで実行され、その出力結果でコマンド (およびそれを囲んでいる引用符) を置き換えます。処理系によってはパイプがサポートされている場合もあります。**plot datafile special-filenames** を見てください。

子プロセスの出力中の改行文字は空白文字に置換されます。

コマンド置換は、単一引用符内の文字列以外は、**gnuplot** のコマンドライン中、どこででも使用可能です。

例:

以下の例は、**leastsq** というプログラムを実行し、その出力結果で、**leastsq** を (まわりの引用符こみで) 置き換えます:

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

ただし VMS では、

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

以下は現在の日付とユーザー名のラベルを生成します:

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

15 Syntax

gnuplot における記号や区切りの用法に関する一般的な規則は、キーワードとオプションは順序依存である、ということです。リストや座標がコンマ (,) 区切りであるのに対し、オプションやそれに伴うパラメータはスペース ( ) 区切りです。範囲はコロン (:) で区切ってかぎかっこ ([]) でくくりますし、文字列やファイル名は引用符でくくり、他にいくつかカッコ (()) でくくるものがあります。中カッコ ({}) は特別な目的で使われます。

コンマは以下の区切りで使用されます。**set** コマンドの **arrow**, **key**, **label** の座標; 当てはめ (fit) られる変数のリスト (コマンド **fit** のキーワード **via** に続くリスト); コマンド **set cntrparam** で指定されるとびとびの等高線の値やそのループパラメータのリスト; **set** コマンドの **dgrid3d dummy**, **isosamples**, **offsets**, **origin**, **samples**, **size**, **time**, **view** の引数; 目盛りの位置やそのループパラメータのリスト; タイトルや軸の見出しの位置; **plot**, **replot**, **splot** コマンドの x,y,z 座標の計算に使われる媒介変数関数のリスト; **plot**, **replot**, **splot** コマンドの複数の描画 (データ、または関数) のそれぞれの一連のキーワードのリスト。

(丸) カッコは、目盛りの見出しを (ループパラメータではなく) 明示的に集合与える場合の区切りとして、または **fit**, **plot**, **replot**, **splot** コマンドの **using** フィルタでの計算を指示するために使われます。

(カッコやコンマは通常の関数の表記でも使われます。)

かぎかっこは、**set**, **plot**, **splot** コマンド上で用いられた場合は範囲を区切るのに使われます。

コロンは **range** (範囲) 指定 (**set**, **plot**, **splot** コマンドで使われる) の両端の値を区切るのに、または **plot**, **replot**, **splot**, **fit** コマンドの **using** フィルタの各エントリを区切るのに使われます。

セミコロン (;) は、一行のコマンド行内で与えられる複数のコマンドを区切るのに使われます。

中カッコは、**postscript** のようないくつかの出力形式で特別に処理される文字列内で使用されます。または複素数を記述するのにも使われます: {3,2} = 3 + 2i となります。

文字列は単一引用符 (' ') または二重引用符 ("") で囲まれます。\n (改行) や \345 (8 進表記の文字コード) のような文字列でのバックスラッシュ (\) は、2 重引用符内の文字列では効力がありますが、単一引用符内では効力を持ちません。

1 つの複数行文字列に関する位置合わせは各行に同等に働きます。よって、中央に位置合わせされた文字列

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

は次のように表示されます:

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

しかし

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

だと次のようになります。

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

ファイル名は単一引用符、あるいは二重引用符内で囲みます。このマニュアルでは一般にコマンドの例示では、わかりやすくするためにファイル名は単一引用符でくくり、他の文字列は二重引用符でくくります。

postscript 出力形式 (terminal) の enhanced オプションを使う場合、現在は、{} の内部に \n を入れてはいけません。

EEPIC, Imagen, Uniplex, LaTeX, TPIC の各ドライバでは、単一引用符内の \\ または二重引用符内の \\\\ で改行を示すことが可能です。

バッククォート (``) は置換のためにシステムコマンドを囲むのに使います。

16 Time/Date data

gnuplot は入力データとして時間/日付情報の使用をサポートしています。この機能は **set xdata time**, **set ydata time** などのコマンドによって有効になります。

内部では全ての時間/日付は 2000 年からの秒数に変換されます。コマンド **set timefmt** は全ての入力書式を定義します。データファイル、範囲、軸の目盛りの見出し、ラベルの位置 – 手短に言えば、データの値を受けとる全てのものがこの書式にしたがって受けとらなければいけません。一時には一つの入力書式のみが有効なので、同じときに入力される全ての時間/日付のデータは同じ書式である必要があります。よって、ファイル内の x と y の両方が時間/日付データである場合は、それらは同じ書式でなければいけません。

秒数へ (秒数から) の変換は国際標準時 (UT; グリニッジ標準時 (GMT) と同じ) が使われます。各国標準時や夏時間への変換の機能は何も持ち合わせていません。もしデータがすべて同じ標準時間帯に従っているなら (そして全てが夏時間か、そうでないかのどちらか一方にのみ従うなら) これに関して何も心配することはありません。しかし、あなたが使用するアプリケーションで絶対的な時刻を厳密に考察しなければいけない場合は、あなた自身が UT に変換すべきでしょう。

show xrange のようなコマンドは、その整数値を **timefmt** に従って解釈し直します。**timefmt** を変更してもう一度 **show** でその値を表示させるとそれは新しい **timefmt** に従って表示されます。このため、もし機能を停止させるコマンド (**set xdata** のような) を与えると、その値は整数値として表示されることになります。

コマンド **set format** は、指定された軸が時間/日付であるなしに関わらず目盛りの見出しに使われる書式を定義します。

時間/日付情報がファイルから描画される場合、**plot**, **splot** コマンドでは **using** オプションを「必ず」使う必要があります。**plot**, **splot** では各行のデータ列の分離にスペースを使いますが、時間/日付データはその中にスペースを含み得るからです。もしタブ区切りを使用しているのなら、あなたのシステムがそれをどう扱うか確かめるために何度もテストする必要があるでしょう。

次の例は時間/日付データの描画の例です。

ファイル "data" は以下のような行からなるとします:

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

このファイルは以下のようにして表示されます:

```
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
```

ここで、x 軸の目盛りの見出しは "03/21" のように表示されます。

各コマンドの詳細はそれぞれの項の記述を参照してください。

第 II 部

Commands

このセクションでは **gnuplot** が受け付けるコマンドをアルファベット順に並べています。このドキュメントを紙に印刷したものは全てのコマンドを含んでいますが、オンラインドキュメントの方は完全ではない可能性があります。実際、この見出しの下に何のコマンドも表示されないシステムがあります。

ほとんどの場合、コマンド名とそのオプションは、紛らわしくない範囲で省略することが可能です。すなわち、"**plot f(x) with lines**" の代わりに"**p f(x) w l**" とすることができます。

書式の記述において、中カッコ ({}) は追加指定できる引数を意味し、縦棒 (|) は互いに排他的な引数を区切るものとします。

17 Cd

cd コマンドはカレントディレクトリを変更します .

書式:

```
cd '<ディレクトリ名>'
```

ディレクトリ名は引用符に囲まれていなければなりません .

例:

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

DOS では、二重引用符内 (") ではバックスラッシュ (\) が特別な意味を持ってしまうため、一重引用符 (') を用いなければなりません。 例えば

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

では失敗しますが、

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

なら期待通りに動くでしょう。

18 Call

call コマンドは，1 つの機能以外は **load** コマンドと等価です．その機能は，10 個までのパラメータをコマンドに追加できることです (パラメータは標準的な構文規則によって区切られます)．これらのパラメータは，ファイルから読まれる行に代入することができます．**call** した入力ファイルから各行が 読まれる時に，**$** (ドル記号) に続く数字 (0–9) の並びを走査します．もし見つかれば，その並びは **call** のコマンド行の対応するパラメータで置き換えられます．**call** の行でそのパラメータが文字列として指定されているならば，取り囲んでいる引用符が省かれて代入されます．数字以外の文字が後に続く **$** はその文字になります。例えば，一つの **$** を得るには **$$** を使います．**call** のコマンド行に 10 個より多いパラメータを与えるとエラーが起こります．与えられなかったパラメータは，何も無しとして扱われます．**call** 中のファイルの中にさらに **load** または **call** コマンドがあっても構いません．

call コマンドは，複数のコマンドからなる行の中では最後のコマンドでなければなりません．

書式:

```
call "<入力ファイル>" <パラメータ 0> <パ 1> ... <パ 9>
```

入力ファイル名は引用符で囲まなければなりません．そして、パラメータも引用符で囲むことを推奨します (gnuplot の将来のバージョンでは引用符で囲んである部分と囲んでない部分に対しては違う取り扱いをする予定です)。

例:

ファイル 'calltest.gp' は以下の行を含んでいるとすると:

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

次の行を入力すると:

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

以下のように表示されるでしょう:

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

注意: **using** を使用しているデータファイルでは文法的に重なってしまいます。その場合、**call** されたデータファイルからプロットするときは、データの n カラム目の指示には **$$n** または **column(n)** を使用してください。

19 Clear

clear コマンドは，**set output** で選択された画面または出力装置をクリアします．通常，ハードコピー装置に対しては改ページを行います．出力装置を選択するには **set terminal** を使用して下さい．

いくつかの出力装置は **clear** コマンドでは **set size** で定義された描画領域のみを消去します。そのため、**set multiplot** とともに使用することで挿入図を一つ作ることができます。

例:

```
set multiplot
plot sin(x)
set origin 0.5,0.5
```

```
set size 0.4,0.4
clear
plot cos(x)
set nomultiplot
```

これらのコマンドの詳細については **set multiplot**, **set size**, **set origin** を参照してください。

20 Exit

exit と **quit** の両コマンドと END-OF-FILE 文字は，現在の **gnuplot** コマンドファイルを終了し、次のファイルを **load** します。詳細は "help batch/interactive" を参照してください。

これらのコマンドは，出力装置を **clear** コマンドと同様にクリアしてしてから終了させます .

21 Fit

fit コマンドはユーザ定義関数を (x,y) または (x,y,z) の形式のデータ点の集合への当てはめを可能にします。それには Marquardt-Levenberg 法による非線形最小自乗法 (NLLS) の実装が用いられます。関数内部に現われるユーザ定義変数はいずれも当てはめのパラメータとして使うことができます。ただ、その関数の返り値は実数である必要があります。

書式:

```
fit {[xrange] {[yrange]}} <function> '<datafile>'
    {datafile-modifiers}
    via '<parameter file>' | <var1>{,<var2>,...}
```

範囲 (xrange,yrange) は、当てはめられるデータ点を一時的に制限するのに使うことができ、その範囲を超えたデータは全て無視されます。その書式は **plot** コマンド同様

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

です (**plot ranges** 参照)。

<function> は通常はあらかじめユーザ定義された f(x) または f(x,y) の形の関数ですが、**gnuplot** で有効な任意の数式を指定できます。

<datafile> は **plot** コマンドと同様に扱われます。**plot datafile** の修飾子 (**using**, **every**,...) は **smooth** を除いて全て **fit** に使うことができます。**plot datafile** を参照してください。

当てはめる 1 変数関数 y=f(x) へのデフォルトのデータの書式は {x:}y か x:y:s で、これらはデータファイルへの **using** 指定子で変更できます。この 3 番目の項目 (列番号、または数式) が与えられた場合は、それは対応する y の値の標準偏差として解釈され、それはそのデータへの重み (=1/s**2) を計算するのに使われます。そうでなければ、全てのデータは同じ重み (1) で計算されます。

2 変数関数 z=f(x,y) を当てはめる場合、データの書式は **using** による 4 つの項目 x:y:z:s が要求されます。これは完全に全てが与えられなければなりません–不足する項目に対してはどの列もデフォルトは仮定されていません。各データ点の重みは上と同様に 's' から計算されます。もし誤差評価を持っていなければ、定数値を定数式として指定すればいいでしょう (**plot datafile using** 参照)。例えば **using 1:2:3:(1)** のように。

複数のデータ集合も複数の 1 変数関数に同時に当てはめることも、y を ' 仮変数' とすれば可能です。例えばデータ行番号を使い、2 変数関数への当てはめ、とすればいいでしょう。**fit multibranch** を参照してください。

via 指定子はパラメータの調節を直接か、またはパラメータファイルを参照することによって行うかを指定します。

例:

```
f(x) = a*x**2 + b*x + c
g(x,y) = a*x**2 + b*y**2 + c*x*y
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 via a, b, c
fit g(x,y) 'surface.dat' using 1:2:3:(1) via a, b, c
```

反復の個々のステップの後で、当てはめの現在の状態についての詳細な情報が画面に表示されます。そし最初と最後の状態に関する同じ情報が "fit.log" というログファイルにも書き出されます。このファイルは前の当てはめの履歴を消さないように常に追加されていきます。これは望むなら削除、あるいは別な名前にできます。

当てはめの反復は Ctrl-C を押すことで中断できます (MSDOS と Atari マルチタスクシステムでは Ctrl-C 以外の任意のキー)。現在の反復が正常に終了した後、(1) 当てはめを止めて現在のパラメータの値を採用する, (2) 当てはめを続行する, (3) 環境変数 FIT_SCRIPT で指定した **gnuplot** コマンドを実行する、のいずれかを選ぶことができます。FIT_SCRIPT のデフォルトは **replot** であり、よってもしデータと当てはめ関数を一つのグラフにあらかじめ描画してあれば、現在の当てはめの状態を表示することができます。

fit が終了した後は、最後のパラメータの値を保存するのに **update** コマンドを使います。その値は再びパラメータの値として使うことができます。詳細は **update** を参照。

21.1 Adjustable parameters

via はパラメータを調節するための 2 つの方法を指定できます。一つはコマンドラインから直接指示するもので、もう一つはパラメータファイルを参照して間接的に行うものです。この 2 つは初期値の設定で違った方法を取ります。

調整するパラメータは、**via** キーワードの後ろにコンマで区切られた変数名のリストを書くことで指定できます。定義されていない変数は初期値 1.0 として作られます。しかし当てはめは、変数の初期値があらかじめ適切な値に設定されている方が多分速く収束するでしょう。

パラメータファイルは個々のパラメータを、個別に 1 行に一つずつ、初期値を次のような形で指定して書きます。

```
変数名 = 初期値
```

'#' で始まるコメント行や空行も許されます。特別な形式として

```
変数名 = 初期値          # FIXED
```

は、この変数が固定されたパラメータであることを意味し、それはこのファイルで初期化されますが、調

節はされません。これは、**fit** でレポートされる変数の中で、どれが固定された変数であるかを明示するのに有用でしょう。なお、# **FIXED** と言うキーワードは厳密にこの形でなくてはなりません。

21.2 Beginner's guide

fit は、与えられたデータ点を与えられたユーザ定義関数にもっとも良く当てはめるようなパラメータを見つけるのに使われます。その当てはめは、同じ場所での入力データ点と関数値との自乗誤差、あるいは残差 (SSR:Sum of the Squared Residuals) の和を基に判定されます。この量は通常χ (カイ) 自乗と呼ばれます。このアルゴリズムは SSR を 最小化することをしようとします。もう少し詳しく言うと、データ誤差 (または 1.0) の重みつき残差の自乗和 (WSSR) の最小化を行っています。詳細は **fit error_estimate** 参照。

これが、(非線形) 最小自乗当てはめ法と呼ばれるゆえんです。非線形 が何を意味しているのかを見るための例を紹介しますが、その前にいくつかの仮定について述べておきます。ここでは簡単のため、1 変数のユーザー定義関数は z=f(x), 2 変数の関数は z=f(x,y) のようにし、いずれも従属変数として z を用いることにします。パラメータとは **fit** が調整して適切な値を決定するユーザ定義変数で、関数の定義式中の未知数です。ここで言う、線形性/非線形性とは、従属変数 z と **fit** が調整するパラメータとの関係に対するものであり、z と独立変数 x (または x と y) との関係のことではありません (数学的に述べると、線形最小自乗問題では、当てはめ関数のパラメータによる 2 階 (そして更に高階の) 導関数は 0、ということになります)。

線形最小自乗法 (LLS) では、ユーザ定義関数は単純な関数の和であり、それぞれは一つのパラメータの定数倍で他のパラメータを含まない項になります。非線形最小自乗法 (NLLS) ではより複雑な関数を扱い、パラメータは色んな使われ方をされます。フーリエ級数は線形と非線形の最小自乗法の違いを表す一つの例です。フーリエ級数では一つの項は

```
z=a*sin(c*x) + b*cos(c*x).
```

のように表されます。もし、a と b が未知なパラメータで c は定数だとすればパラメータの評価は線形最小自乗問題になります。しかし、c が未知なパラメータならばそれは非線形問題になります。

線形の場合、パラメータの値は比較的簡単な線形代数の直接法によって決定できます。しかしそのような LLS は特殊な場合であり、'gnuplot' が使用する反復法は、もちろんそれも含めて、より一般的な NLLS 問題を解くことができます。**fit** は検索を行うことで最小値を探そうとします。反復の各ステップは、パラメータの新しい値の組に対して WSSR を計算します。Marquardt- Levenberg のアルゴリズムは次のステップのパラメータの値を選択します。そしてそれはあらかじめ与えた基準、すなわち、(1) 当てはめが "収束した" (WSSR の相対誤差が FIT_LIMIT より小さくなった場合)、または (2) あらかじめ設定された反復数の限界 FIT_MAXITER (**fit control variables** 参照) に達した場合、のいずれかを満たすまで続けられます。キーボードからその当てはめの反復は中断できますし、それに続いて中止することもできます (**fit** 参照)。

当てはめに使われる関数はしばしばあるモデル (またはある理論) を元にしていて、それはデータの振舞を記述したり、あるいは予測しようとします。よって **fit** は、データがそのモデルにどれくらいうまく当てはまっているのかを決定するため、そして個々のパラメータの誤差の範囲を評価するために、モデルの自由なパラメータの値を求めるのに使われます。**fit error_estimates** も参照してください。

そうでなければ、曲線による当てはめにおける関数は、モデルとは無関係に選ばれています (それは十分な表現力と最も少ない数のパラメータを持ち、データの傾向を記述しそうな関数として経験に基づいて選

ばれるでしょう)。

しかし、もしあなたが全てのデータ点を通るような滑らかな曲線を欲しいなら **fit** ではなく、むしろ **plot** の **smooth** オプションでそれを行うべきでしょう。

21.3 Error estimates

fit において "誤差" という用語は 2 つの異なった文脈で用いられます。一つはデータ誤差、もう一つはパラメータ誤差です。

データ誤差は、平方残差の重み付きの和 WSSR、すなわちχ 自乗を決定する際個々のデータ点の相対的な重みを計算するのに用いられます。それらはパラメータの評価に影響を与えます。それは、それらが、当てはめられた関数からの個々のデータ点の偏差が最終的な値に与える影響の大きさを決定することによります。正確なデータ誤差評価が与えられている場合には、パラメータの誤差評価等の **fit** が出力する情報はより役に立つでしょう。

'statistical overview' では **fit** の出力のいくつかを説明し、'practical guidelines' に対する背景を述べています。

21.3.1 Statistical overview

非線形最小自乗法 (Non-Linear Least-Squares; NLLS) の理論は、誤差の正規分布の点から一般的に記述されています。すなわち、入力データは与えられた平均とその平均に対する与えられた標準偏差を持つガウス (正規) 分布に従う母集団からの標本と仮定されます。十分大きい標本、そして母集団の標準偏差を知ることに対しては、χ 自乗分布統計を用いて、通常「χ 自乗」と呼ばれる値を調べることにより「当てはめの良さ」を述べることができます。減らされた自由度のχ 自乗 (χ 自乗の自由度は、データ点の数から当てはめられるパラメータの個数だけ引いた数) が 1.0 である場合は、データ点と当てはめられた関数との偏差の重みつき自乗和が、現在のパラメータ値に対する関数と与えられた標準偏差によって特徴付けられた母集団の、ランダムなサンプルに対する自乗和とが全く同じであることを意味します。

分散 = 総計である数え上げ統計学同様、母集団の標準偏差が定数でない場合、各点は観測される偏差の和と期待される偏差の和を比較するときに個別に重みづけされるべきです。

最終段階で **fit** は 'stdfit'、すなわち残差の RMS (自乗平均平方根) で求められる当てはめの標準偏差と、データ点が重みづけられている場合に ' 減らされたχ 自乗' とも呼ばれる残差の分散をレポートします。自由度 (データ点の数から当てはめパラメータの数を引いたもの) はこれらの評価で使用されます。なぜなら、データ点の残差の計算で使われるパラメータは同じデータから得られるものだからです。

パラメータに関する信頼レベルを評価することで、当てはめから得られる最小のχ 自乗と、要求する信頼レベルのχ 自乗の値を決定するためのχ 自乗の統計を用いることが出来ます。しかし、そのような値を生成するパラメータの組を決定するには、相当のさらなる計算が必要となるでしょう。

fit は信頼区間の決定よりむしろ、最後の反復後の分散-共分散行列から直ちに得られるパラメータの誤差評価を報告します。これらの評価は、標準偏差として計算される量の指定に関する統計上の条件が、一般には非線形最小自乗問題では保証されないのですが、線形最小自乗問題での標準誤差 (各パラメータの標準偏差) と同じ方法で計算されます。そしてそのため慣例により、これらは "標準誤差" とか "漸近標準誤差" と呼ばれています。漸近標準誤差は一般に楽観過ぎ、信頼レベルの決定には使うべきではありません

が、定性的な指標としては役に立つでしょう。

最終的な解は相関行列も生成します。それは解の範囲におけるパラメータの相関の表示を与えてくれます: もし一つのパラメータが変更されると、χ 自乗の増加が、他の補正の変更を行なう ? 主対角成分、すなわち自己相関はすべて 1 で、もし全てのパラメータが独立ならば他の成分はすべて 0 に近い値になります。完全に他を補いあう 2 つ変数は、大きさが 1 で、関係が正の相関か負の相関かによって正か負になる符号を持つ非対角成分を持ちます。非対角要素の大きさが小さいほど、各パラメータの標準偏差の評価は、漸近標準誤差に近くなります。

21.3.2 Practical guidelines

個々のデータ点への重みづけの割り当ての基礎を知っているなら、それが測定結果に対するより詳しい情報を使用させようとするでしょう。例えば、幾つかの点は他の点より当てになるということを考慮に入れることが可能です。そして、それらは最終的なパラメータの値に影響します。

データの重み付けは、最後の反復後の **fit** の追加出力に対する解釈の基礎を与えます。各点に同等に重み付けを行なうにしても、重み 1 を使うことよりもむしろ平均標準偏差を評価することが、χ 自乗が定義によりそうであるように、WSSR を 無次元変数とすることになります。

当てはめ反復の各段階で、当てはめの進行の評価に使うことが出来る情報が表示されます ('*' はより小さい WSSR を見つけられなかったこと、そして再試行していることを意味します)。'sum of squares of residuals' (残差の自乗和) は、'chisquare' (χ 自乗) とも呼ばれますが、これはデータと当てはめ関数との間の WSSR を意味していて、**fit** はこれを最小化しようとします。この段階で、重み付けされたデータによって、χ 自乗の値は自由度 (= データ点の数 - パラメータの数) に近付くことが期待されます。WSSR は補正されたχ 自乗値 (WSSR/ndf; ndf = 自由度)、または当てはめ標準偏差 (stdfit = sqrt(WSSR/ndf)) を計算するのに使われます。それらは最終的な WSSR に対してレポートされます。

データが重み付けされていなければ、stdfit は、ユーザの単位での、データと当てはめ関数の偏差の RMS (自乗平均平方根) になります。

もし妥当なデータ誤差を与え、データ点が十分多く、モデルが正しければ、補正χ 自乗値はほぼ 1 になります (詳細は、適当な統計学の本の ' χ 自乗分布' の項を参照してください)。この場合、この概要に書かれていること以外に、モデルがデータにどれくらい良く当てはっているかを決定するための追加の試験方法がいくつかあります。

補正χ 自乗が 1 よりはるかに大きくなったら、それは不正なデータ誤差評価、正規分布しないデータ誤差、システム上の測定誤差、孤立した標本値 (outliers)、または良くないモデル関数などのためでしょう。例えば **plot 'datafile' using 1:($2-f($1))** などとして残差を描画することは、それらのシステム的な傾向を知るための手がかりとなります。データ点と関数の両者を描画することは、他のモデルを考えための手がかりとなるでしょう。

同様に、1.0 より小さい補正χ 自乗は、WSSR が、正規分布する誤差を持つランダムなサンプルと関数に対して期待されるものよりも小さいことを意味します。データ誤差評価が大きすぎるのか、統計的な仮定が正しくないのか、またはモデル関数が一般的すぎて、内在的傾向に加えて特殊なサンプルによる変動の当てはめになっているのでしょう。最後の場合は、よりシンプルな関数にすればうまく行くでしょう。

標準的なエラーを、パラメータの不確定性に関する、あるより現実的な評価に関係付けること、および相関行列の重要性を評価することができるようになる前に、あなたは **fit** と、それを適用しようとするある

種の問題に慣れておく必要があるでしょう。

fit は、大抵の非線形最小自乗法の実装では共通して、距離の自乗 (y-f(x))**2 の重み付きの和を最小化しようとすることに注意してください。それは、x の値の "誤差" を計算に関してはどんな方法も与えてはおらず、単に y に関する評価のみです。また、"孤立点" (正規分布のモデルのから外れているデータ点) は常に解を悪化させる可能性があります。

21.4 Fit controlling

fit に影響を与えるために定義できるたくさんの **gnuplot** の変数があります。それらは **gnuplot** の動作中に一度定義できますが、それは **control_variable** で紹介し、**gnuplot** が立 ち上がる前に設定する変数は **environment_variables** で紹介します。

21.4.1 Control variables

デフォルトのもっとも小さい数字の限界 (1e-5) は、変数

```
FIT_LIMIT
```

で変更できます。残差の平方自乗和が 2 つの反復ステップ間で、この数値より小さい数しか変化しなかった場合、当てはめルーチンは、これを ' 収束した' と見なします。

反復数の最大値は変数

```
FIT_MAXITER
```

で制限されます。0 (または定義しない場合) は制限無しを意味します。

更にそのアルゴリズムを制御したい場合で、かつ Marquardt-Levenberg アルゴリズムを良く知っている場合は、さらにそれに影響を与える変数があります。**lambda** (λ) の最初の値は、通常 ML 行列から自動的に計算されますが、もしそれをあらかじめ用意した値にセットしたければ

```
FIT_START_LAMBDA
```

にセットしてください。FIT_START_LAMBDA を 0 以下にセットすると、自動的に計算されるようになります。変数

```
FIT_LAMBDA_FACTOR
```

は、χ 自乗化された関数が増加、あるいは減少するにつれて **lambda** が増加あるいは減少する因数を与えます。FIT_LAMBDA_FACTOR を 0 とすると、それはデフォルトの因数 10.0 が使用されます。

fit には FIT_ から始まる変数が他にもありますから、ユーザ定義変数としてはそのような名前で始まる変数は使わないようにするのが安全でしょう。

変数 FIT_SKIP と FIT_INDEX は、以前の版の **gnuplot** の、**gnufit** と呼ばれていた **fit** パッチで使われていたもので、現在は使用されていません。FIT_SKIP の機能はデータファイルに対する **every** 指定子で用意されています。FIT_INDEX は複数当てはめ法 (multi-branch fitting) で使われていたものですが、1 変数の複数当てはめ法は、今では 疑似 3 次元当てはめとして行なわれていて、そこでは枝の指定には 2 変数と **using** が使われています。**fit multi-branch** を参照してください。

21.4.2 Environment variables

環境変数は **gnuplot** が立ち上がる前に定義しなければなりません。その設定方法はオペレーティングシステムに依存します。

```
FIT_LOG
```

は、当てはめのログが書かれるファイル名 (およびパス) を変更します。デフォルトでは、作業ディレクトリ上の "fit.log" となっています。

```
FIT_SCRIPT
```

は、ユーザが中断した後に実行するコマンドを指定します。デフォルトでは **replot** ですが、**plot** や **load** コマンドとすれば、当てはめの進行状況の表示をカスタマイズするのに便利でしょう。

21.5 Multi-branch

複数当てはめ法 (multi-branch fitting) では、複数のデータ集合を、共通のパラメータを持つ複数の 1 変数の関数に、WSSR の総和を最小化することによって同時に当てはめることが出来ます。各データセットに対する関数とパラメータ (枝) は ' 疑似変数' を使うことで選択できます。例えば、データ行番号 (-1; ' データ列' の番号) またはデータファイル番号 (-2) を 2 つ目の独立変数とします。

例: 2 つの指数減衰形 z=f(x) が与えられていて、それぞれ異なるデータ集合を記述しているが、共通した減衰時間を持ち、そのパラメータの値を評価する。データファイルが x:z:s の形式であったとすると、この場合以下のようにすればよい。

```
f(x,y) = (y==0) ? a*exp(-x/tau) : b*exp(-x/tau)
fit f(x,y) 'datafile' using  1:-1:2:3  via a, b, tau
```

より複雑な例については、デモファイル "fit.dem" で使われる "hexa.fnc" を参照してください。

もし従属変数のスケールに差がある場合、単位の重み付けでは 1 つの枝が支配してしまう可能性があるので、適当な重み付けが必要になります。各枝をバラバラに当てはめるのに複数当てはめ法の解を初期値として用いるのは、全体を合わせた解の各枝に対する相対的な影響に関する表示を与えることになるでしょう。

21.6 Starting values

非線形当てはめは、大域的な最適値 (残差の自乗和 (SSR) の最小値を持つ解) への収束は保証はしませんが、局所的な極小値を与えることはできます。そのサブルーチンはそれを決定する方法を何も持ち合わせていないので、これが起こったかどうかを判断するのはあなたの責任となります。

fit は、解から遠くから始めると失敗するかも知れませんし、しばしばそれは起こり得ます。遠くというのは、SSR が大きく、パラメータの変化に対してその変化が小さい、あるいは数値的に不安定な領域 (例えば数値が大きすぎて浮動小数の桁あふれを起こす) に到達してしまって、その結果 "未定義値 (undefined value)" のメッセージか **gnuplot** の停止を引き起こしてしまうような場合を意味します。

大域的な最適値を見つける可能性を改善するには、最初の値をその解に少なくともほぼ近くに取るべきでしょう。例えば、もし可能ならば一桁分の大きさの範囲内で。最初の値が解に近いほど他の解で終了して

しまう可能性は低くなります。最初の値を見つける一つの方法は、データと当てはめ関数を同じグラフの上に描画して適当な近さに達するまで、パラメータの値を変更して **replot** することを繰り返すことです。その描画は、よくない当てはめの極小値で当てはめが終了したかどうかをチェックするのにも有用です。

もちろん、適度に良い当てはめが、"それよりよい" 当てはめ (ある改良された当てはめの良さの基準によって特徴付けられた統計学的な意味で、あるいはそのモデルのより適切な解である、という物理的な意味で) が存在しないことの証明にはなりません。問題によっては、各パラメータの意味のある範囲をカバーするような様々な初期値の集合に対して **fit** することが望ましいかも知れません。

21.7 Tips

ここでは、**fit** を最大限に利用するためにいくつか覚えておくべきヒントを紹介します。それらは組織的ではないので、その本質がしみ込むまで何回もよく読んでください。

fit の引数の **via** には、2 つの大きく異なる目的のための 2 つの形式があります。**via "file"** の形式は、バッチ処理 (非対話型での実行が可能) で最も良く使われ、そのファイルで初期値を与え、またその後で結果を他の (または 同じ) パラメータファイルにコピーするために **update** を使うことも出来ます。

via var1, var2, ... の形式は対話型の実行で良く使われ、コマンドヒストリの機構が使ってパラメータリストの編集を行い、当てはめを実行したり、あるいは新しい初期値を与えて次の実行を行なったりします。これは難しい問題に対しては特に有用で、全てのパラメータに対して 1 度だけ当てはめを直接実行しても、良い初期値でなければうまくいかないことが起こり得るからです。それを見つけるには、いくつかのパラメータのみに対して何回か反復を行ない、最終的には全てのパラメータに対する 1 度の当てはめがうまくいくところに十分近くなるまでそれを繰り返すことです。

当てはめを行なう関数のパラメータ間に共通の依存関係がないことは確認しておいてください。例えば、a*exp(x+b) を当てはめに使ってはいけません。それは a*exp(x+b)=a*exp(b)*exp(x) だからです。よってこの場合は a*exp(x) または exp(x+b) を使ってください。

技術的なお話: パラメータの大きさはあまり違いすぎてはいけません。絶対値が最も大きいパラメータと最も小さいパラメータの比が大きい程当てはめの収束は遅くなります。その比が、マシンの浮動小数の精度の逆数に近いか、またはそれ以上ならば、ほとんど永久に収束しないか、拒否されるでしょう。よってその関数をこれを避けるように改良しなければいけません。例えば、関数の定義で 'parameter' を '1e9*parameter' にするとか、または最初の値を 1e9 で割るとか。

もし、関数を、当てはめるパラメータを係数とする、単純な関数の線形結合で書けるなら、それはとてもいいので是非そうしてください。何故なら、問題がもはや非線形ではないので、反復は少ない回数で収束するでしょう。もしかしたらたった一回ですむかもしれません。

実際の実験の講義ではデータ解析に対するいくつかの指示が与えられ、それでデータへの最初の関数の当てはめが行なわれます。もしかすると、基礎理論の複数の側面にひとつずつ対応する複数回のプロセスが必要かも知れませんが、そしてそれらの関数の当てはめのパラメータから本当に欲しかった情報を取り出すでしょう。しかし、**fit** を使えば、求めるパラメータの視点から直接モデル関数を書くことにより、それはしばしば 1 回で済むのです。時々はより難しい当てはめ問題の計算コストがかかりますが、データ変換もかなりの割合で避けることが出来ます。もしこれが、当てはめ関数の単純化に関して、前の段落と矛盾してると思うなら、それは正解です。

"singular matrix" のメッセージは、この Marquardt-Levenberg アルゴリズムのルーチンが、次の反復に

対するパラメータの値の計算が出来ないことを意味します。この場合、別な初期値から始めるか、関数を別な形で書き直すか、より簡単な関数にしてみてください。

最後に、他の当てはめパッケージ (fudgit) のマニュアルから、これらの文書を要約するようないい引用を上げます: "Nonlinear fitting is an art! (非線形当てはめ法は芸術だ !)"

22 Help

help コマンドは，オンラインヘルプを表示します．ある項についての説明を指定したいときには，次の書式を使って下さい:

```
help {<項目名>}
```

もし < 項目名 > が指定されなかった場合は，**gnuplot** についての簡単な説明が表示されます．指定した項目についての説明が表示された後，それに対する細目のメニューが表示され、その細目名を入力することで細目に対するヘルプを続けることができます。そして，その細目の説明が表示された後に，さらなる細目名の入力を要求されるか、または 1 つ前の項目のレベルへ戻ります。これを繰り返すとやがて，**gnuplot** のコマンドラインへと戻ります．

また、疑問符 (?) を項目として指定すると、現在のレベルの項目のリストが表示されます。

23 If

if コマンドは，条件付でコマンドを実行させることができます．

書式:

```
if (<条件>) <コマンド行>
```

< 条件 > が評価され，もしそれが真 (ゼロでない) ならば，< コマンド行 > のコマンドが実行されます．もし，< 条件 > が偽 (ゼロ) ならば，< コマンド行 > の全部が無視されます．同じ行に複数のコマンド置くことを可能にする ';' を使えば，条件付のコマンドが終らないことに注意して下さい．

例:

```
pi=3
if (pi!=acos(-1)) print "?Fixing pi!"; pi=acos(-1); print pi
```

を実行すると，

```
?Fixing pi!
 3.14159265358979
```

と表示されます．また，

```
if (1==2) print "Never see this"; print "Or this either"
```

ならば，何も表示されません．

if と **reread** を使ってループを構成する例については **reread** を参照してください。

24 Load

load コマンドは，指定された入力ファイルの各行を，それが対話的に入力されたかのように実行します．**save** コマンドでつくられたファイルは，**load** することができます．有効なコマンドの書かれたテキストファイルをつくれば，それは，**load** コマンドによって，実行することができます．**load** 中のファイルの中にさらに **load** または **call** コマンドがあっても構いません．コマンド中のコメントについては，**comment** を参照して下さい．**load** するときに引数を与える方法については **call** を参照してください。

load コマンドは，複数のコマンドからなる行の中では最後のコマンドでなければなりません．

書式:

```
load "<入力ファイル名>"
```

入力ファイル名は引用符で囲まなければなりません．

load コマンドは、標準入力からのコマンドの入力のために、特別なファイル名 "-" を用意しています。これは、**gnuplot** のコマンドファイルが、いくつかのコマンドを標準入力から受け付けることを意味します。詳細については "help batch/interactive" を参照してください。

例:

```
load 'work.gnu'
load "func.dat"
```

gnuplot への引数として与えられたファイル名は，暗黙のうちに **load** コマンドによって実行されます．これらは，指定された順にロードされ，その後 **gnuplot** は終了します．

25 Pause

pause コマンドは，コマンドに続く任意の文字列を表示した後，指定された時間または，改行キーが押されるまで待ちます．**pause** コマンドは，**load** 用のファイルと共に使用すると，便利になるでしょう．

書式:

```
pause <時間> {"<文字列>"}
```

< 時間 > は，任意の整数の定数または式です．-1 を指定すると改行キーが押されるまで待ちます．0 を指定すると一切待たず，正整数を指定するとその秒数だけ待ちます．**pause 0** は **print** と同じです。

注意: **pause** コマンドは OS へのコマンドであり描画の一部ではないので，異なる出力装置では異なる動作をする可能性があります．(これは，テキストとグラフィックスが，どのように混在するかによります．)

例:

```
pause -1    # 改行キーが押されるまで待つ．
pause 3     # 3 秒待つ．
pause -1  "Hit return to continue"
pause 10  "Isn't this pretty?  It's a cubic spline."
```

26 Plot

plot は **gnuplot** で図を描くための基本的なコマンドです。 それは関数やデータを実に多くの方法で表示します。**plot** は 2 次元の関数やデータを描くのに使われ、**splot** は 3 次元の曲面やデータの 2 次元投影を描きます。**plot** と **splot** は多くの共通の特徴点を持ちますが、その違いについては **splot** の項を参照してください。特に注意しておきますが、**splot** の **binary** と **matrix** のオプションは **plot** には存在しません。

書式:

```
plot {<範囲 (ranges)>}
     { <関数 (function)>
       |  {"<データファイル (datafile)>" {データファイル修飾子}}}
     {axes <軸 (axes)>}{<表題 (title)>}{with <スタイル (style)>}
     {, {定義,} <関数> ...}
```

< 関数 > または引用符で囲まれたデータファイル名のどちらか一方を与えます。関数は一本の数式、または parametric mode においては 2 つの数式の組です。数式は完全に定義してもいいですし、前の方の **gnuplot** のコマンド列で部分的に定義してもいいです (**user-defined** の項目参照)。

関数とパラメータは **plot** コマンド自身の上で定義をすることも可能です。これは単に他の項目とコンマで分離して記述することでなされます。

軸は、4 種類の組が利用できます; キーワード < 軸 > は、特定の直線をどの軸に尺度を合わせるか、ということを選択するのに使われます。**x1y1** は下の軸と左の軸を指定; **x2y2** は上と右の軸の指定; **x1y2** は下と右の軸の指定; **x2y1** は上と左の軸の指定です。**plot** コマンドで指定された範囲は、この最初の軸の組 (下と左) にのみ適用されます。

例:

```
plot sin(x)
plot f(x) = sin(x*a), a = .2, f(x), a = .4, f(x)
plot [t=1:10] [-pi:pi*2] tan(t), \
     "data.1" using (tan($2)):($3/$4) smooth csplines \
              axes x1y2 notitle with lines 5
```

26.1 Data-file

ファイルに納められた離散的なデータは、**plot** コマンドライン上で、そのデータファイル名 (単一引用符または二重引用符で囲まれた) を指定することによって表示できます。

書式 :

```
plot '<ファイル名>' {index <index list>}
                    {every <every list>}
                    {thru <thru expression>}
                    {using <using list>}
                    {smooth <option>}
```

修正子の **index**, **every**, **thru**, **using**, **smooth** は、それぞれに分けて説明します。簡単に言うと、**index** はマルチデータセットファイルからどのデータセットを表示するのかを選び、**every** が、一つのデータセッ

トからどのポイントを表示するのかを選び、**using** は一行からどの列を解釈するのかを決定し (**thru** は、**using** の特別な場合である)、そして **smooth** が、単純な補間と近似を行います。(**splot** は、よく似た書式を持っていますが、**smooth** オプションと **thru** オプションはサポートしていません)

データファイルは、一行につき少なくとも一つのデータポイントを含む必要があります (**using** は一行から一つのデータポイントを選ぶことができます)。# (VMS では **!**) で始まる行は、コメントとして扱われ、無視されます。各データポイントは、(x,y) の組を表します。エラーバー付きの **plot** では (**set style errorbars** 参照)、各データポイントは、(x,y,ydelta), (x,y,ylow,yhigh), (x,y,xdelta), (x,y,xlow,xhigh), (x,y,xlow,xhigh, ylow,yhigh) のいずれかを意味します。全ての場合において、もし書式の指定子が **using** オプションによって与えられていなければ、データファイルの各行の数字は、ホワイトスペース (一つまたは複数の空白かタブ) によって区切られている必要があります。このホワイトスペースは、各行を列の項目に区切ります。

データは、指数部に e, E, d, D, q, Q の文字をつけた指数表記で書かれていても構いません。

必要であるのはただ一つの列 (y の値) のみです。もし x の値が省略されたら、**gnuplot** はそれを 0 で始まる整数値として用意します。

データファイルにおいて、ブランク行 (空白と改行、復帰以外に文字を含まない行) は重要です — - ブランク行の対は、**index** (**plot datafile index** 参照) を区切ります。2 つのブランク行で分離されたデータは、別々のデータファイルのデータであるかのように扱われます。

一つのブランク行は、**plot** に不連続を指示します; ブランク行によって区切られた点は線で結ばれることはありません (line style で書かれている場合には)。

もし autoscale の状態であれば (**set autoscale** 参照)、軸は全てのデータポイントを含むように自動的に引き伸ばされて、目盛りが書かれる状態ならば全ての目盛りがマークされます。これは、2 つの結果を引き起こします: i) **splot** では、曲面の角は底面の角に一致していないことがあります。この場合、縦の線は書かれることはありません。ii) 2 種類の軸での、同じ x の範囲のデータの表示の際、もし x2 の軸に対する目盛りが書かれていない場合は、x 座標があっていないことがあります。これは x 軸 (x1) は全ての目盛りにまで自動的に引き延ばされるのに対し、x2 軸はそうではないからです。次の例でその問題を見ることができます:

```
reset; plot '-', '-'
1 1
19 19
e
1 1
19 19
e
```

26.1.1 Every

キーワード **every** は、描画するデータをデータセットから周期的にサンプリングすることを可能にします。ここでは「ポイント」はファイル中の 1 つの行によって定義されるデータとし、ここでの「ブロック」は「データ・ブロック」(**glossary** 参照) と同じものを意味することとします。

書式：

```
plot 'file' every {<ポイント増分>}
```

```
{:{<ブロック増分>}
  {:{<開始ポイント>}
    {:{<開始ブロック>}
      {:{<終了ポイント>}
        {:<終了ブロック>}}}}}
```

プロットされるデータポイントは、< 開始ポイント > から < 終了ポイント > まで < ポイント増分 > の増加で選ばれ、ブロックは < 開始ブロック > から < 終了ブロック > まで < ブロック増分 > の増加で選ばれます。

各ブロックの最初のデータは、ファイル中の最初のブロックと同じように、「0 番」とされます。

プロットできない情報を含んでいる行もカウントされることに注意して下さい。

いくつかの数字は省略することができます; 増分のデフォルトは 1 、開始の値は最初のポイントか最初のブロック、そして終了の値は最後のポイントか最後のブロックに設定されます。**every** が指定されないなら、全ての行の全てのポイントがプロットされます。

例:

```
every :::3::3   # 4 番目のブロックだけ選ばれます (0 番が最初)
every :::::9    # 最初の 10 ブロックが選ばれます
every 2:2       # 1 つおきのブロックで 1 つおきのポイントが選ば
                # れます
every ::5::15   # それぞれのブロックでポイント 5 から 15 までが
                # 選ばれます
```

26.1.2 Example datafile

次の例は、ファイル "population.dat" 中のデータと理論曲線を図にするものです。

```
pop(x) = 103*exp((1965-x)/10)
plot [1960:1990] 'population.dat', pop(x)
```

ファイル "population.dat" は次のようなファイルです。

```
# Gnu population in Antarctica since 1965
   1965   103
   1970   55
   1975   34
   1980   24
   1985   10
```

26.1.3 Index

キーワード **index** はマルチデータセットファイルの中の、いくつかのデータセットのみを選び出すのに使われます。

書式:

```
plot 'file' index <m>{{:<n>}:<p>}
```

データセットは 2 行の空白で分離されています。**index <m>** は <m> 番目のセットだけを選択します; **index <m>:<n>** は <m> から <n> までのデータセットの選択; **index <m>:<n>:<p>** は、<m>, <m>+<p>, <m>+2<p>, など、<p> おきのセットを選択し、セット <n> で終了します。C 言語の添字 (index) の付け方に従い、index 0 はそのファイルの最初のデータセットを意味します。大きすぎる index の指定にはエラーメッセージが返されます。**index** が指定されない場合は、全てのデータセットが単一のデータセットとして描かれます。

例:

```
plot 'file' index 4:5
```

26.1.4 Smooth

gnuplot は、データの補間と近似を行う汎用的なルーチンをいくつか持っています。これ **smooth** オプションの中にグループ化されています。より洗練されたデータ処理をしたければ、外部においてデータの前処理をするか、または適切なモデルで fit を使うのがいいでしょう。

書式:

```
smooth {unique | csplines | acsplines | bezier | sbezier}
```

unique は、データを単調に揃えた後で、それらを plot します。他のルーチンはいずれも、データの両端の点の間を結ぶ、ある連続曲線の係数を決定するためにデータを使います。この曲線は、関数として同じ方法で描画されます。すなわち、それらの値は x 座標に沿う同じ幅の区間ごとに選ばれ (**set samples** 参照)、それらの点を線分でつなぐことにより (もし line style が選ばれているのならば) 描画されます。

もし **autoscale** の状態であれば、描画範囲はグラフの境界線の中に曲線が収まるように計算されます。

選択されたオプションを適用するのにデータの点数が少なすぎる場合は、エラーメッセージが表示されます。その最小のデータ数は **unique** では 1 つ、**acsplines** では 4 つ、他のオプションでは 3 つです。

smooth オプションは、関数の描画のときには無視されます。

26.1.4.1 Acsplines

acsplines オプションは「自然な滑らかなスプライン」でデータを近似します。データが x に関して単調にされた後 (**smooth unique** 参照)、1 つの曲線が、いくつかの 3 次多項式の一部分により区分的に構成されます。それらの 3 次式の係数は、いくつかのデータポイントの重み付けによって求められます。重みは、データファイルの 3 列目に与えます。そのデフォルトの値は、**using** の 3 番目の項目によって変更することができます。例えば次のようにします。

```
plot 'data-file' using 1:2:(1.0) smooth acsplines
```

性質上、重みの絶対的な大きさは、曲線を構成するのに使われる区分の数を決定します。もし重みが大きければ、個々のデータの影響は大きくなり、そしてその曲線は、隣り合う点同志を自然 3 次スプラインでつないで得られるものに近づきます。もし重みが小さければ、その曲線はより少ない区分で構成され、それによってより平滑的になります。その最も極端な場合はただ 1 つの区分からなる場合であり、それは全てのデータに重みの付き線形最小 2 乗近似によって作られます。誤差の立場から言えば、平滑さの重みは、その曲線に対する「平滑化因子」によって分割された各点への、統計的な重みと見ることができます。それにより、そのファイル中の (標準的な) 誤差は平滑さの重みとして使うことができます。

例:

```
sw(x,S)=1/(x*x*S)
plot 'data_file' using 1:2:(sw($3,100)) smooth acsplines
```

26.1.4.2 Bezier **bezier** オプションは、n 次 (データ点の個数) のベジェ曲線でデータを近似します。この曲線は両端の点をつなぎます。

26.1.4.3 Csplines **csplines** オプションはデータを単調に揃えた後で (**smooth unique** 参照) 自然 3 次スプライン曲線で引き続く点をつなぎます。

26.1.4.4 Sbezier **sbezier** オプションは、最初にデータを単調に揃え (**unique** 参照) そして **bezier** アルゴリズムを適用します。

26.1.4.5 Unique **unique** オプションは、データを x 方向に単調にします。同じ x を持つデータ点は y の値を平均して一つの点で置き換えます。そしてその結果として得られる点を線分で結びます。

26.1.5 Special-filenames

'-' という特別なファイル名は、データがインラインであることを指示します。すなわち、データをコマンドの後に続けて指定します。このときはデータのみがコマンドに続き得ます。よって、**plot** コマンドに対するのフィルター、タイトル、ラインスタイルといったオプションは、**plot** のコマンドラインの方に書かないといけません。これは、unix シェルスクリプトにおける << (ヒアドキュメント)、あるいは VMS DCL における $DECK と同様です。そのデータは、それらがファイルから読み込まれたかのように、1 行につき 1 つずつのデータ点が入力されます。そしてデータの終りは、1 列目の始めに文字 "e" を置くことで指示します。**using** オプションをこれらのデータに適用することは可能です – ある関数を通しデータをフィルターすることに使うのは意味があるでしょうが、列を選ぶのに使うことは多分意味がないでしょう。

'-' は、データとコマンドを一緒に持つことが有用である場合のためにあります。例えば、**gnuplot** があるフロントアプリケーションのサブプロセスとして起動される場合などがこれにあたります。例として、デモンストレーションでこの機能を使うものがあるでしょう。**index** や **every** のような **plot** のオプションが与えられていると、それらはあなたに使われることのないデータを入力する事を強要します。次の例を見てください。

```
plot '-' index 0, '-' index 1
2
4
6

10
12
14
e
2
4
6
```

```
10
12
14
e
```

これは、実際に動作しますが、

```
plot '-' , '-'
2
4
6
e
10
12
14
e
```

とタイプする方が楽でしょう。

もし、**replot** コマンドで '-' を使うなら、あなたは 1 度以上データを入力する必要があでしょう。

空のファイル名 ('') は、直前のファイル名が再び使われることを指示します。これは、

```
plot ' ある/とても/長い/ファイル名' using 1:2, '' using 1:3, '' using 1:4
```

のようなときに便利です。(もし同じ plot コマンド上で、'-' と '' の両方を使用すると、上の例にあるように、インラインデータの 2 つのセットを与える必要があります。)

popen 関数を持っているコンピュータシステム (Unix) の上では、データファイルは、'<' で始まるファイル名によって、シェルコマンドからパイプ入力することができます。例えば

```
pop(x) = 103*exp(-x/10)
plot "< awk '{print $1-1965, $2}' population.dat", pop(x)
```

は、最初の人口の例と同じ情報を描画します。ただし、x 座標は 1965 年からの経過年を表すようになります。この例を実行するときは、上のデータファイルのコメント行をすべて削除しなければなりませんが、または上のコマンドの最初の部分を次のように変えることもできます (コンマに続く部分):

```
plot "< awk '$0 !~ /^#/ {print $1-1965, $2}' population.dat"
```

このアプローチは最も柔軟性がありますが、**using** あるいは **thru** キーワードを用いた単純なフィルタリングで行うことも可能です。

26.1.6 Thru

thru 関数は前のバージョンとの互換性のために用意されています。

書式:

```
plot 'file' thru f(x)
```

これは次と同様です:

```
plot 'file' using 1:(f($2))
```

後者の方がより複雑に見えますが、この方が柔軟性を持っています。さらに自然な

```
plot 'file' thru f(y)
```

も動作します (すなわち、y をダミー変数として使うことができます)。

thru は **splot** と **fit** でも通りますが、何の効果も持ちません。

26.1.7 Using

最もよく使われるデータファイルの修飾子は **using** です。

書式:

```
plot 'file' using {<entry> {:<entry> {:<entry> ...}}} {'format'}
```

もし、フォーマット (format) が指定されれば、C のライブラリ関数 'scanf' を使ってデータファイルの各行をそのフォーマット文字列に従って読み込みます。そうでなければ、行はスペースまたはタブの所で列に分割されて読み込まれます。もし時系列フォーマットデータ (time-format data) を使っている場合は、フォーマットを指定することはできません (これは **set data time** で行わなければなりません)。

データは entry の指定に従った列に並び直されます。各 <entry> には、データを選び出すための単なる列の番号、カッコで囲まれた数式を指定するか、あるいは何も指定しません。数式中では、最初の列の値を読み込むために $1、2 番目の列の項目を使うために $2、といった書き方を使用できます。また、**column(x)** や **valid(x)** といったものも使うことができます。ここで、x は結果として整数になる任意の数式です。**column(x)** は x 番目のデータを返します。**valid(x)** は x 番目のデータが有効な値かをテストします。列番号の 0 は、各点毎に 0 から始まる番号を表し、それは 2 行の空行が来たところでリセットされます。列番号の -1 は 0 から始まるデータ行の番号を意味します。これは 1 行の空行毎に 1 ずつ増加し、2 行の空行が来たところでリセットされます。列番号の -2 は index を意味します。これは 2 行の空行が来たところで 1 ずつ増加します。<entry> に何も書かなければその entry のリストの順にデフォルトの値が使われます。例えば **using ::4** は **using 1:2:4** と解釈されます。

注: **call** コマンドも $ を特別な文字として使います。**call** の引数リストの中に列番号を含ませる方法の詳細については **call** の項目を参照してください。

using にただ一つの entry を指定した場合はその <entry> は y の値として使われ、データ点の番号が x として使われます。例えば **"plot 'file' using 1"** は **"plot 'file' using 0:1"** と同じ意味です。**using** に 2 つの entry を与えた場合、それらは x, y として使われます。 さらに entry を追加すると、それらは x および/または y の誤差に使われます。誤差情報を使った plot スタイルの詳細については **set style** を、そして、回帰曲線法での誤差情報の使用については **fit** を参照してください。

'scanf' 関数では色々なデータ形式の数値入力が使えますが、**gnuplot** は全ての入力データを倍精度浮動小数とみなしますから、**gnuplot** では **lf** が唯一の数値入力指定、ということになります。'scanf' は数と数の間にホワイトスペース – 空白、タブ ("\t")、改行 ("\n")、または改ページ ("\f") – があると期待します。それ以外の入力は明示的にスキップされるべきです。

"\t", "\n", "\f" を使うときはシングルクォートよりむしろダブルクォートを使うべきであることに注意してください。

例:

次の例は、1 番目のデータに対する 2 番目と 3 番目の和の値を plot します (書式文字列は、各列データがスペース区切りでなく、カンマ区切りであることを指示しています)。

```
plot 'file' using 1:($2+$3) '%lf,%lf,%lf'
```

次の例は、より複雑な書式指定でデータをファイル "MyData" から読み込みます。

```
plot 'MyData' using "%*lf%lf%*20[^\n]%lf"
```

この書式指定の意味は以下の通りです:

`%*lf`	数値を無視
`%lf`	倍精度浮動小数を読み込む（デフォルトでは x の値）
`%*20[^\n]`	20 個の改行以外の文字を無視
`%lf`	倍精度浮動小数を読み込む（デフォルトでは y の値）

3 項演算子 **?:** を使ってデータをフィルタする一つの芸当を紹介します。

```
plot 'file' using 1:($3>10 ? $2 : 1/0)
```

これは、1 列目のデータに対して、3 列目のデータが 10 以上であるような 2 列目のデータを plot します。**1/0** は未定義値であり、**gnuplot** は未定義の点を無視するので、よって適切でない点は隠されることになります

カッコで始まっていない限りは定数式を列番号として使うことができます。例えば **using 0+(**複雑な式**)** の様なことができます。そして、その数式は、カッコでスタートしていなければ数式の値が一度評価され、カッコでスタートしていれば個々のデータ点を読み込むためにその値が一度評価される、という点が重要です。

時系列フォーマットデータを使っている場合、その時間のデータは複数の列に渡らせることができます。その場合、他のデータの開始位置を計算するとき、時間のデータに空白が含まれていることに注意してください。例えば、データ行の最初の要素がスペースが埋め込まれた時間データであるならば、y の値は 3 列目の値として指定されるべきです。

plot 'file' と **plot 'file' using 1:2**、そして **plot 'file' using ($1):($2)** には微妙な違いがあることに注意してください。1) **file** が 1 列と 2 列のデータを持つ行をそれぞれ含んでいるとすると、データが 1 列のみの行に対しては、最初のものは x の値を作り出し、2 番目のものはその行は無視し、3 番目のものはそれを未定義の値として保存します (折れ線で plot している場合 (plot with lines)、その未定義の点を通過する線を結ばないように)。2) 1 列目に文字列を含んでいるような行がある場合、最初のものはエラーとして plot を中止しますが、2 番目と 3 番目のものはその不要な行を読みとばします。

実際、最初に単に

```
plot 'file' using 1:2
```

と指定することで、大抵の場合どんなにゴミのデータを含む行を持つファイルをも plot することが可能になります。しかし、どうしてもデータファイルに文字列を残しておきたいならば、そのテキスト行の第一列にコメント文字 (#) を置く方がより安全でしょう。

26.2 Errorbars

エラーバーは、1 から 4 個の追加されたデータを読む (またはエントリを **using** で追加選択する) ことにより、2 次元データの描画において実現されています。これら追加される値は、それぞれのエラーバースタイルで異なった形で使われます。

デフォルトでは、**gnuplot** はデータファイルの各行に以下のような 3 つ、4 つ、あるいは 6 つの列があることを期待しています:

```
(x, y, ydelta),
(x, y, ylow, yhigh),
(x, y, xdelta),
(x, y, xlow, xhigh),
(x, y, xdelta, ydelta),
(x, y, xlow, xhigh, ylow, yhigh)
```

x 座標は必ず指定しなければいけません。各数値を書く順序も上で挙げた通りでなくてはなりません。ただ、**using** 修飾子を使えばその順序を操作できますし、欠けている列の値も補うことは可能ですが。例えば、

```
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
```

最後の例は、相対的な x の誤差と絶対的な y の誤差、という、サポートされていない組のファイルに対するものです。**using** エントリが相対的な x の誤差から絶対的な x の最小値と最大値を生成しています。

y のエラーバーは、(x, ylow) から (x, yhigh) への鉛直な線として描かれます。ylow と yhigh の代わりに ydelta が指定されたときは、ylow = y - ydelta, yhigh = y + ydelta となります。ある行にデータが 2 つしかなければ、ylow と yhight はともに y となります。x エラーバーは同様に計算された水平線です。データの各点を結ぶ折れ線を引きたい場合は、with errorbars と with lines を指定して , 同じデータファイルを 2 回 **plot** して下さい (ただし、キーの中に 2 つのエントリを作らないように、その一方には **notitle** オプションを使うことを忘れないで下さい)。

エラーバーには、もし **set bar** を使っていなければ、そのそれぞれの端に垂直な線分がつきます (詳細は **set bar** をご覧下さい)。

自動範囲指定が有効であれば、その描画範囲はエラーバーも含むように調整されます。

さらなる情報に関して、**plot using**, **plot with**, **set style** も参照して下さい。

26.3 Parametric

媒介変数モード (**set parametric**) では、**plot** では 2 つの数式の組を、**splot** では 3 つの数式の組を与える必要があります。

例:

```
plot sin(t),t**2
splot cos(u)*cos(v),cos(u)*sin(v),sin(u)
```

データファイルは前と同じように描画されます。ただし、データファイルが描画のために与えられる前に、任意の媒介変数関数が先に完全に指定された場合を除いてです。言い換えると、x の媒介変数関数 (上の

例では **sin(t)**) と y の媒介変数関数 (上の例では **t**2**) との間に、他の修飾子やデータ関数をはさみこんではいけません。そのようなことをすると、構文エラーになり、媒介変数関数が完全には指定されていない、と表示されます。

with や **title** のような他の修飾子は、媒介変数関数の指定が完了した後に指定しなければいけません。

```
plot sin(t),t**2 title 'Parametric example' with linespoints
```

26.4 Ranges

オプションの範囲は、表示されるグラフの領域範囲を指定します。

書式:

```
[{<dummy-var>=}{{<最小値>}:{<最大値>}}]
[{{<最小値>}:{<最大値>}}]
```

最初の範囲指定は独立変数の範囲 (**xrange** またはパラメトリックモードでは **trange**) で、2 番目のものは従属変数の範囲 **yrange** (パラメトリックモードでは **xrange**) となります。<dummy-var> には独立変数の新しい別名を指定します (デフォルトの変数名は **set dummy** で変更できます)。< 最小値 >, < 最大値 > には定数式、あるいは * を書くことができます。

パラメトリックモードでなければ、与えられるべき範囲指定は **xrange**, **yrange** の順になります。

パラメトリックモードでは、**plot** コマンドに対してはその順序は **trange**, **xrange**, **yrange** になります。以下の **plot** コマンドは、**trange** を [-pi:pi], **xrange** を [-1.3:1.3], **yrange** を [-1:1] に設定する例です。

```
plot [-pi:pi] [-1.3:1.3] [-1:1] sin(t),t**2
```

x2 の範囲と y2 の範囲はここでは指定できないことに注意してください。それには **set x2range** や **set y2range** が使われます。

範囲は適切なモードに対して、上に示した順序で解釈されます。必要な範囲指定が一度全て指定されると、再び指定し直すことはありませんが、必要ない部分を全く指定しないようにはできません – その代わりそこに空の範囲指定 [] を置きます。

***** は、最小値や最大値に自動範囲指定 (autoscale) の機能を使うことを可能にします。**set autoscale** も参照してください。

plot や **splot** のコマンド行で指定された範囲はそのグラフにのみ影響を及ぼします。よって、その後のグラフのデフォルトの範囲を変更するには、**set xrange** や **set yrange** を使用してください。

時間データに対しては、範囲はクォートで囲んで指定する必要があります (データファイルに現われる時間データと同じ形式の)。**gnuplot** はその範囲を読みこむのに時間書式文字列 (**timefmt**) を使用します。詳しくは **set timefmt** を参照してください。

例:

以下は現在の範囲を使用します:

```
plot cos(x)
```

以下は x の範囲のみの指定です:

```
plot [-10:30] sin(pi*x)/(pi*x)
```

以下は上と同じですが、仮変数として t を使います:

```
plot [t = -10 :30]  sin(pi*t)/(pi*t)
```

以下は x と y の両方の範囲の指定です:

```
plot [-pi:pi] [-3:3]  tan(x), 1/x
```

以下は、y の範囲のみの指定で、両方の軸の自動範囲指定機能を無効にします:

```
plot [ ] [-2:sin(5)*-8] sin(x)**besj0(x)
```

以下は x の最大値と y の最小値のみの指定です。

```
plot [:200] [-pi:]  exp(sin(x))
```

以下は x の範囲を時系列データとして指定しています:

```
set timefmt "%d/%m/%y %H:%M"
plot ["1/6/93 12:00":"5/6/93 12:00"] 'timedata.dat'
```

26.5 Title

各関数やデータに対する曲線のタイトルは、その曲線のサンプル、および (または) それを表示されるのに使われる記号とともにキーの中に表示されます。それは **title** オプションで変更できます。

書式:

```
title "<title>" | notitle
```

ここで <title> はその曲線の新しいタイトルで、クォートで囲む必要があります。クォートはキーには表示されません。特殊文字も、バックスラッシュに続く 8 進値 (例えば "\345" のように) を使うことで用いることができます。タブ文字 "\t" は認識されます。バックスラッシュのそのような作用はダブルクォートで囲まれた文字列でしか効きません。逆にその作用を働かさないようにするにはシングルクォートを使ってください。改行文字 "\n" はどちらの型のクォートでもキーでは働きません。

曲線タイトルとサンプルは予約語 **notitle** を使うことでキーから削除できます。何もないタイトル (**title** '') は **notitle** と同じ意味を持ちます。サンプルだけが欲しいときは、一つ以上の空白をタイトルの後ろに入れてください (**tilte** ' ')。

デフォルトでは曲線のタイトルはその **plot** コマンドに現われる関数、またはデータファイル名です。ファイル名の場合は、指定される任意のデータファイル修飾子もそのデフォルトタイトルに含まれます。

位置やタイトルの位置揃えなどのキーのレイアウトは、**set key** で制御できます。詳細は **set key** の項目を参照してください。

例:

以下は y=x をタイトル 'x' で表示します:

```
plot x
```

以下は、x の 2 乗をタイトル "x^2" で、ファイル "data.1" をタイトル"measured data" で表示します:

```
plot x**2 title "x^2", 'data.1' t "measured data"
```

以下は、極座標グラフの周りに円形の境界を書き、タイトルなしで表示します:

```
set polar; plot my_function(t), 1 notitle
```

26.6 With

関数やデータの表示にはたくさんのスタイルのうちの一つを使うことができます。キーワード **with** がその選択のために用意されています。

書式:

```
with <style> { {linestyle | ls <line_style>}
               | {{linetype | lt <line_type>}
                  {linewidth | lw <line_width>}
                  {pointtype | pt <point_type>}
                  {pointsize | ps <point_size>}} }
```

ここで、<style> は **lines**, **points**, **linespoints**, **impulses**, **dots**, **steps**, **fsteps**, **histeps**, **errorbars**, **xerrorbars**, **yerrorbars**, **xyerrorbars**, **boxes**, **boxerrorbars**, **boxxyerrorbars**, **financebars**, **candlesticks**, **vector** の中のいずれかです。これらのいくつかに対してはデータを付け足す必要があります。それぞれのスタイルの詳細については **set style <style>** をご覧ください。

デフォルトのスタイルは **set function style** や **set data style** コマンドで決定されます。

デフォルトでは、それぞれの関数やデータファイルは、使うことができる型の最大数に達するまで異なる線種、点種を使います。すべての端末用ドライバは最低 6 つの異なる点種をサポートしていて、もしたくさん要求された場合、それらを順に再利用していきます。LaTeX ドライバは、それより 6 つ多く点種 (いずれも円の変種) を持っていて、よって点での曲線の描画は 12 種類の曲線が繰り返されるのみです。PostScript ドライバは (**postscript**) 全部で 64 種類の 点種を持っています。

一つの描画で線種や点種を選びたいならば、<line_type> や <point_type> を指定してください。これらの値は、その描画で使われる線種や点種を指定する正の整定数 (または数式) です。使用する端末で使える線種、点種を表示するには **test** コマンドを使ってください。

描画の線の幅や点の大きさは <line_width> や <point_size> で変更できます。これらはその各々の端末のデフォルトの値に対する相対的な値として指定します。点の大きさは全体に通用するように変更できます – 詳細は **set pointsize** を参照してください。しかし、ここでセットされる <point_size> と、**set pointsize** でセットされる大きさは、いずれもデフォルトのポイントサイズに掛けられることに注意してください – すなわち、それらの効果は累積はしません。例えば、**set pointsize 2; plot x w p ps 3** は、デフォルトのサイズの 3 倍であって、6 倍ではありません。

set linestyle を使って線種/線幅、点種/点幅の組を定義すれば、そのスタイルの番号を <line_style> にセットすることでそれらを使うことができます。

キーワードは暗示するような形で省略可能です。

linewidth と **pointsize** オプションは全ての端末装置でサポートされているわけではないことに注意してください。

例:

以下は、sin(x) を鉛直線で描画します:

```
plot sin(x) with impulses
```

以下は、x を点で描画し、x**2 をデフォルトの方式で描画します:

```
plot x w points, x**2
```

以下は、tan(x) を関数のデフォルトの方式で、"data.1" を折れ線で描画します:

```
plot [ ] [-2:5] tan(x), 'data.1' with l
```

以下は、"leastsq.dat" を鉛直線で描画します:

```
plot 'leastsq.dat' w i
```

以下は、データファイル "population" を矩形で描画します:

```
plot 'population' with boxes
```

以下は、"exper.dat" をエラーバー付きの折れ線で描画します (エラーバーは 3 列、あるいは 4 列のデータを必要とします):

```
plot 'exper.dat' w lines, 'exper.dat' notitle w errorbars
```

以下は、sin(x) と cos(x) をマーカー付きの折れ線で描画します。折れ線は同じ線種ですが、マーカーは異なったものを使います:

```
plot sin(x) with linesp lt 1 pt 3, cos(x) with linesp lt 1 pt 4
```

以下は "data" を点種 3 で、点の大きさを通常の 2 倍で描画します:

```
plot 'data' with points pointtype 3 pointsize 2
```

以下は、2 つのデータ集合に対して、幅のみ異なる線を用いて描画します:

```
plot 'd1' t "good" w l lt 2 lw 3, 'd2' t "bad" w l lt 2 lw 1
```

デフォルトの表示方法の変更方法については、**set style** を参照して下さい。

27 Print

print コマンドは < 式 > の値を画面に表示します . これは **pause 0** と同じです。< 式 > は、数を生成する **gnuplot** の数式か、または文字列です。

書式:

```
print <式>
```

expressions を参照して下さい .

28 Pwd

pwd コマンドはカレントディレクトリの名前を画面に表示します .

29 Quit

exit と **quit** の両コマンドと END-OF-FILE 文字は , **gnuplot** を終了させます . これらのコマンドは , 出力装置を (**clear** コマンドと同様に) クリアしてから終了させます .

30 Replot

replot コマンドを引数なしで実行すると，最後に実行した **plot** または **splot** コマンドを再実行します．これは，あるプロットを異なる **set** オプションでみたり，同じプロットを異なる装置に出力したりするときに便利でしょう．

replot コマンドに対する引数は最後に実行した **plot** または **splot** コマンドの引数に (暗黙の ',' と共に) 追加され，それから再実行されます．**replot** は，範囲 (range) を除いては，**plot** や **splot** と同じ引数をとることができます．よって、直前のコマンドが **splot** ではなく **plot** の場合は、関数をもう一つの軸刻でプロットするのに **replot** を使うことができます。そして同様に、直前のコマンドが **plot** でなく **splot** である場合、バイナリファイルからのプロットを追加するのに使うことができます。

注意:

```
plot '-' ; ... ; replot
```

という使い方は推奨されません。**gnuplot** はインラインデータを保存しないので、**replot** によって新たな情報が直前の **plot** に追加されて修正されたコマンドを実行することになったとしても、最初の plot の '-' は再びインラインデータを読もうとするからです。

replot コマンドは **multiplot** モードでは働きません。それは、それが画面全体にではなく直前のプロットのみを再実行するものだからです。

最後に実行した **plot** (**splot**) コマンドの内容を修正する方法については **command line-editing** を参照して下さい．

31 Reread

reread コマンドは，**load** コマンドまたはコマンドラインで指定した **gnuplot** のコマンドファイルを，その次のコマンドが読まれる前に，開始点に再設定します．これは，コマンドファイルの最初から **reread** コマンドまでのコマンドの無限ループを本質的に実装していることになります。(しかし、これは何も悪いことではありません。**reread** は **if** と組み合わせることでとても有用なコマンドとなります。詳細は **if** を参照してください。) 標準入力からの入力の場合は，**reread** コマンドは何も影響を与えません。

例:

ファイル "looper" が次のようなファイルで

```
a=a+1
plot sin(x*a)
pause -1
if(a<5) reread
```

そして、**gnuplot** から次のように実行するとします。

```
a=0
load 'looper'
```

すると、pause のメッセージで分割された 4 回のプロットが行われることになります。

ファイル "data" が、各行に、0 から 10 までの範囲 (yrange) の 6 つのデータ を持ち、最初が x 座標で、その他は 5 つの異なる関数の、その x での値であるとします。そして、ファイル "plotter" が

```
c_p = c_p+1
plot "$0" using 1:c_p with lines linetype c_p
if(c_p <  n_p) reread
```

で、**gnuplot** から次のように実行するとします。

```
n_p=6
c_p=1
set nokey
set yrange [0:10]
set multiplot
call 'plotter' 'data'
set nomultiplot
```

すると、5 つのプロットを合わせた 1 つのグラフができます。yrange は、multiplot モードで最初のものに続けて書かれる 5 つのグラフが、同じ軸を持つように、明示的に指定する必要があります。線種も指定しなければなりません。さもないと、全てのグラフが同じ線種で書かれることになります。

32 Reset

コマンド **reset** は **set** コマンドで定義できるほぼ全てのオプションをデフォルトの値に設定しますが、**set term** による出力形式の設定、および **set output** による出力ファイルの指定のみが例外でこれらは変化しません。このコマンドは、例えばコマンドファイルの最後にデフォルトの設定に復帰する、あるいはコマンドファイル内でたくさんの設定を行なった場合に元の状態に戻すときなどに便利です。様々なオプションの取るデフォルトの値を知るには、**set** コマンドの項を参照してください。

33 Save

save コマンドは，ユーザ定義関数，変数，**set** で設定するオプションのいずれかか，これらすべてと，それに加えて最後に実行した **plot** または **splot** コマンドを指定したファイルに保存します．

書式:

```
save  {<オプション>} '<ファイル名>'
```

ここで，< オプション > は，**functions**, **variables**, **set** のいずれかです．どれも指定されなかった場合には，**gnuplot** は，ユーザ定義関数，変数，set で設定するオプション，最後に実行した **plot** または **splot** コマンドの全てを保存します．

save は，テキスト形式で出力します．また，このファイルは **load** コマンドで読み込むことができます．

ファイル名は引用符に囲われていなければなりません．

例:

```
save 'work.gnu'
save functions 'func.dat'
save var 'var.dat'
save set 'options.dat'
```

34 Set-show

set コマンドは実に多くのオプションを設定するのに使われます。しかし、**plot**, **splot**, **replot** コマンドが与えられるまで何も表示しません。

show コマンドはそれらの設定値を表示します . **show all** でそれら全てを表示します .

もし変数が日時のデータを含むならば、**show** は、**set timefmt** によって現在設定されている書式に従って表示します。それは変数が最初に設定されていてその書式が効果を持たなかったとしてもです。

34.1 Angles

デフォルトでは **gnuplot** は極座標グラフの独立変数の単位はラジアンを仮定します。**set polar** の前に **set angles degrees** を指定すると、その単位は度になり、デフォルトの範囲は [0:360] となります。これはデータファイルの描画で特に便利でしょう。角度の設定は、**set mapping** コマンドを設定することにより 3 次元でも有効です。

書式:

```
set angles {degrees | radians}
show angles
```

set grid polar で指定される角度も、**set angles** で指定した単位で読まれ表示されます。

set angles は組み込み関数 sin(x), cos(x), tan(x) の引数や asin(x), acos(x), atan8x), atan2(x), arg(x) の出力にも影響を与えます。双曲線関数や、ベッセル関数の引数には影響を与えません。しかし、複素数を引数とする逆双曲線関数の出力には影響が出ます。それらの関数が使われるときは、**set angles radians** は入出力の引数の間に一貫性を持った管理を実現していなければなりません。

```
x={1.0,0.1}
set angles radians
y=sinh(x)
print y         #prints {1.16933, 0.154051}
print asinh(y)  #prints {1.0, 0.1}
```

しかし、

```
set angles degrees
y=sinh(x)
print y         #prints {1.16933, 0.154051}
print asinh(y)  #prints {57.29578, 5.729578}
```

34.2 Arrow

set arrow コマンドを使うことにより、グラフ上の任意の位置に矢印を表示することができます。

書式:

```
set arrow {<tag>} {from <position>} {to <position>} {{no}head}
```

```
          { {linestyle | ls <line_style>}
            | {linetype | lt <line_type>}
              {linewidth | lw <line_width>} }
set noarrow {<tag>}
show arrow
```

タグ <tag> は各矢印を識別する整数です。タグを指定しない場合は、その時点で未使用の最も小さい数が自動的に割り当てられます。タグを使うことで、特定の矢印を変更したり、削除したりできます。既に存在する矢印の属性を変更する場合は、タグを明示した set arrow コマンドで変更箇所を指定してください。

<position> は x,y あるいは x,y,z で指定します。そしてその前に座標系を選択するために **first**, **second**, **graph**, **screen** を置くことができます。座標を指定しなければデフォルトでは 0 と見なされます。矢印の端点は、四つの座標系 – **first** か **second** の軸、**graph** あるいは **screen** – のうちの 1 つを選択して指定できます。詳細は **coordinates** を参照して下さい。"from" の場所の座標系指定子は、"to" の場所に影響を及ぼすことはありません。グラフの枠をはみ出る矢印を書くこともできますが、出力端末によってはエラーを生ずることがあります。

nohead を指定することで、矢先のない矢 – すなわち線分を書くこともできます。これは描画の上に線分を描く別な方法を与えます。デフォルトでは矢先がついています。

線種はユーザの定義したラインスタイルのリストから選ぶこともできますし (**set linestyle** 参照)、用意されている <line_type> の値 (デフォルトのラインスタイルのリストの番号) そして <linewidth> (デフォルトの幅の乗数) を使ってここで定義することもできます。

しかし、ユーザー定義済のラインスタイルが選択された場合、その属性 (線種、幅) は、単に他の **set arrow** コマンドで適当な番号や **lt**, **lw** などを指定しても、変更はできないことに注意して下さい。

例:

原点から (1,2) への矢印をユーザ定義済のラインスタイル 5 で描くには:

```
set arrow to 1,2 ls 5
```

描画領域の左下角から (-5,5,3) へタグ番号 3 の矢印を描くには:

```
set arrow 3 from graph 0,0 to -5,5,3
```

矢印の端を 1,1,1 に変更し、矢先を外して幅を 2 にするには:

```
set arrow 3 to 1,1,1 nohead lw 2
```

x=3 の所へグラフの下から上まで鉛直線を描くには:

```
set arrow from 3, graph 0 to 3, graph 1 nohead
```

2 番の矢印を消すには:

```
set noarrow 2
```

全ての矢印を消すには:

```
set noarrow
```

全ての矢印の情報を (タグの順に) 見るには:

```
show arrow
```

34.3 Autoscale

自動縮尺機能 (autoscale) は x, y, z の各軸に対して独立に、または一括して指定できます。デフォルトでは全ての軸に対して自動縮尺設定を行います。

書式:

```
set autoscale {<axes>{min|max}}
set noautoscale {<axes>{min|max}}
show autoscale
```

ここで、<axes> (軸) は **x**, **y**, **z**, **x2**, **y2**, **xy** のいずれかです。**min** または **max** を軸に追加指定すると (**xy** では使えませんが) それは **gnuplot** にその軸の最小値、または最大値のみを自動縮尺させることになります。軸も何も指定されていない場合は全ての軸が対象となります。

自動縮尺機能を使うときは、描画範囲は自動的に割り出され、従属変数軸 (**plot** のときは y 軸、**splot** のときは z 軸) は、関数やデータの値域が収まるように設定されます。

従属変数軸 (y または z) の自動縮尺機能が指定されていない場合は、現在の y や z の描画範囲がそのまま使われます。

独立変数軸 (**plot** のときは x 軸、**splot** のときは x,y 軸) の自動縮尺機能が指定されている場合は、描画される全てのデータファイルの点が収まるように定義域をとるようになります。データファイルが 1 つも指定されていない場合は、自動縮尺機能はなんの効果もありません。つまり、関数のみが指定されていてデーターファイルを使わない場合は、x 軸の描画範囲 (z = f(x,y) を描画しているときは y 軸も) は影響をうけません。

範囲に関するより詳しい情報に関しては **set xrange** を見てください。

媒介変数モード (parametric) でも自動縮尺機能は有効です (**set parametric** 参照)。この場合、より多くの従属変数があるので、x, y, z 各軸に関して、より多くの制御が行われます。媒介変数モードでの独立変数 (仮変数) は **plot** では t で **splot** では u, v です。そして媒介変数モードでは、自動縮尺機能は (t, u, v, x, y, z) の全ての描画範囲を制御し、x, y ,z の範囲の自動設定を完全に行います。

自動縮尺機能は、極座標モード (polar mode) でも **plot** の媒介変数モードと同様に機能しますが、極座標モードでは **set dummy** で独立変数を t から変更するできる (**set dummy** 参照) という拡張があります。

目盛りが第二の軸に表示され、しかもこれらの軸に対する描画が行われなかった場合には、x2range と y2range は xrange と yrange の値を受け継ぎます。これは、xrange と yrange が整数個の目盛り幅に自動縮尺される「前」に行われますので、場合によって期待しない結果をもたらす可能性があります。

例:

以下は y 軸の自動縮尺機能を指定します (他の軸には影響を与えません):

```
set autoscale y
```

以下は y 軸の最小値に対してのみ自動縮尺機能を指定します (y 軸の最大値、および他の軸には影響を与えません):

```
set autoscale ymin
```

以下は x, y 両軸の自動縮尺機能を指定します:

```
set autoscale xy
```

以下は x, y, z, x2, y2 全軸の自動縮尺機能を指定します:

```
set autoscale
```

以下は x, y, z, x2, y2 全軸の自動縮尺機能を禁止します:

```
set noautoscale
```

以下は z 軸のみについて自動縮尺機能を禁止します:

```
set noautoscale z
```

34.3.1 Parametric mode

媒介変数表示モード (**set parametric**) においては , xrange も yrange と同様に縮尺を変えることができます。つまり、媒介変数モードにおいては、x 軸方向も自動的に縮尺が調整され、描こうとしている媒介変数表示の関数が収まるようになります。もちろん、y 軸方向も媒介変数モードでない時同様に自動的に縮尺を変えます。x 軸について自動縮尺機能が設定されていない場合は、現在の x の範囲が使われます。

データファイルは媒介変数モードでもそうでない状態でも同様に描画されます。しかし、データファイルと関数が混在している場合には、違いがあります: 媒介変数モードでなければ、x の自動縮尺機能は、関数の範囲をデータの描画範囲に合わせます。しかし媒介変数モードではデータの範囲は関数の範囲に影響しません。

それには、片手落ちにならないように **set autoscale t** というコマンドも用意されています。しかしその効果は非常に小さいものです。自動縮尺機能が設定されていると、**gnuplot** が t の範囲が無くなってしまうと判断した場合に範囲を少し調整します。自動縮尺機能が設定されていないとこのようなときにはエラーとなります。このような動作は実はあまり意味がなく、よって **set autoscale t** というコマンドは存在意義に疑問があります。

splot では上記の発想の元に拡張されています . 自動縮尺機能が設定されている場合、x, y, z の各描画範囲は計算結果が収まるように設定され縮尺調整されます。

34.3.2 Polar mode

極座標モード (**set polar**) では、xrange と yrange は極座標から求められ、それによって自動的に範囲設定がなされます。言いかえると、極座標モードでは描こうとしている極座標関数が収まるように x 軸、y 軸が自動的に縮尺が調整されます。

極座標モードで関数を描画する場合、rrange も自動範囲設定されます。データファイルを描画する場合はさらに trange も自動範囲設定がなされます。もし、trange がある象限 (四分円) に収まるならば、自動縮尺機能によりその象限のみの描画が行われることに注意してください。

1 つ、あるいは 2 つの範囲は明示的に設定してその他のものを指定しない場合は予期しない結果を引き起こすかも知れません。

34.4 Bar

コマンド **set bar** は誤差グラフ (errorbar) の両端のマークの幅を制御します。

書式 x:

```
set bar {small | large | <size>}
show bar
```

small は 0.0, **large** は 1.0 と同じです。サイズを指定しなければデフォルトの値は 1.0 です。

34.5 Bmargin

コマンド **set bmargin** は、下部の余白のサイズを設定します。詳細は **set margin** を参照してください。

34.6 Border

set border と **set noborder** は **plot** や **splot** でのグラフの枠の表示を制御します。

書式:

```
set border {<integer> { {linestyle | ls <line_style>}
                        | {linetype | lt <line_type> }
                          {linewidth | lw <line_width>} } }
set noborder
show border
```

枠は、12 ビットの整数に符号化されています: 下位 4 ビットは **plot** に対する外枠、**splot** に対しては底面の外枠、次の 4 ビットは **splot** の鉛直な外枠、そして上位 4 ビットは **splot** の天井面の外枠を制御します。その < 整数 > 値は次の表の対応する項目の数字の和になります:

グラフの外枠の符号化			
	選択するビットの整数値		
方向	plot の外枠 splot の底面	splot の 鉛直線	splot の 天井面
下 (南)	1	16	256
左 (西)	2	32	512
上 (北)	4	64	1024
右 (東)	8	128	2048

デフォルトの値は 31 で、これは **plot** では 4 方向の外枠全て、**splot** では底面の枠線全部と z 軸を描くことを意味します。

<line_style>, <line_type>, <line_width> を指定して、枠線の描画にそれらを反映させることができます (現在の出力装置がサポートするものに限定されます)。デフォルトでは、枠線は通常の 2 倍線幅で描かれます。<line_width> はそのデフォルトの値を伸縮させます。例えば **set border 15 lw 2** という指定により、枠の幅は通常の線幅の 4 倍になります。

軸は一つ一つ、あるいはいくつかをまとめて一緒にこのコマンドで付加できます。

下と左以外の枠に目盛りをつけるには、通常の目盛りを無効にしてから第二の軸を有効にします。

例:

以下は (デフォルトの) 全ての枠線を描きます:

```
set border
```

以下は南西方向 (下と左) の枠線を描きます:

```
set border 3
```

以下は **splot** で周りに完全な箱を描きます:

```
set border 4095
```

以下は、手前の垂直面を除いた箱を描きます:

```
set border 127+256+512
```

以下は北東方向 (上と右) の枠線のみ描きます:

```
set noxtics; set noytics; set x2tics; set y2tics; set border 12
```

34.7 Boxwidth

コマンド **set boxwidth** は **boxes** と **boxerrorbars** スタイルにおける棒のデフォルトの幅を設定するために使います

書式:

```
set boxwidth {<width>}
show boxwidth
```

第 3, 4, 5 番目の列 (または **using** による項目指定) のないデータファイルの描画、あるいは関数の描画では、各々の棒の幅は **set boxwidth** で設定します (もしファイルと **set boxwidth** の両方で幅が指定されている場合はファイルの方の幅が使われます)。いずれの方法でも指定されていない場合は、その隣合う棒に接するように各々の棒の幅が自動的に計算されます。4 列のデータの場合、第 4 列目の値が棒の幅として使われます。ただし、その幅が-2.0 の場合には、自動計算されます。詳細は **set style boxerrorbars** を参照してください。

棒の幅を自動的にセットするには

```
set boxwidth
```

とする、あるいは 4 列のデータに対しては以下のようにします。

```
set boxwidth -2
```

plot のキーワード **using** を使っても同じ効果を得ることができます:

```
plot 'file' using 1:2:3:4:(-2)
```

34.8 Clabel

gnuplot は、clabel が設定されている時には、各々の等高線のレベルに対して使う線種を変化させます。このオプションが有効である場合 (デフォ ルト)、凡例によって各々の線種を、それが表す z のレベルとともに表示されます。

書式:

```
set clabel {'<format>'}
set noclabel
show clabel
```

書式文字列のデフォルトは %8.3g で、小数部分は 3 桁表示されます。もし key がそのデフォルトの値から変更されていれば、その配置は不十分なものになるかもしれません。

最初の等高線の線種、または clabel が無効である場合の唯一つの等高線の線種は、(曲面の線種 +1) になります。等高線の点は曲面の点と同じものになります。

set contour も参照してください。

34.9 Clip

gnuplot はグラフの端の辺りのデータ点や線をクリッピングすることができます。

書式:

```
set clip <クリップ型>
set noclip <クリップ型>
show clip
```

クリップ型として **gnuplot** は **points**, **one**, **two** の 3 種を扱えます。ある描画に対して、これらのクリップ型は任意の組み合せで設定することができます。

クリップ型 **points** を設定すると，描画領域内にはあるけれど境界線に非常に近いような点をクリップする (実際には描画しないだけですが) ように **gnuplot** に指示します。これは点として大きなマークを使用したときに、そのマークが境界線からはみ出さないようにする効果があります。**points** をクリップしない場合、境界線の辺りの点が汚く見えるかもしれません。その場合、x や y の描画範囲 (xrange, yrange) を調整してみて下さい。

クリップ型 **one** を設定すると、一端のみが描画領域にあるような線分も描画するように **gnuplot** に指示します。この際、描画領域内にある部分のみが実際に描画される範囲です。設定しなかった場合、このような線分は描画対象とならず、どの部分も描画されません。

両端は共に描画範囲に無いが描画領域を通過するという線分もあります。クリップ型 **two** を設定することによって、このような線分の描画領域の部分を描画することができます。

どのような状況でも、描画範囲の外に線が引かれることはありません。

デフォルトでは、**noclip points**, **clip one**, **noclip two** となっています。

全てのクリップ型の設定状況を見るには以下のようにします:

```
show clip
```

過去のバージョンとの互換性のため以下の書式も使用可能です:

```
set clip
set noclip
```

set clip は **set clip points** と同義です。**set noclip** は 3 種のクリップ型全てを無効にします。

34.10 Cntrparam

set cntrparam は等高線の生成方法、およびそれを滑らかに描画する方法を制御します。**show contour** は現在の **contour** の設定だけでなく **cntrparam** の設定をも表示します。

書式:

```
set cntrparam {  {linear | cubicspline | bspline}
                 { points <n>} { order <n> }
                 { levels   auto {<n>} | <n>
                          | discrete <z1> {,<z2>{,<z3>...}}
                          | incremental <start>, <incr> {,<end>}
                  }
                }
show contour
```

このコマンドは 2 つの機能を持っています。一つは等高線上の点 (データ点の線形補間、あるいは関数の標本化 (isosample) による点) での z の値の設定で、もう一つは、そのように決定された z が等しい点同士を等高線で結ぶ方法の制御です。<n> は整数型の定数式、<z1>, <z2> ... は任意の定数式です。各オプション変数の意味は次の通りです:

linear, **cubicspline**, **bspline** — 近似 (補間) 方法を指定します。**linear** ならば，等高線は曲面から得られた値を区分的に直線で結びます。**cubicspline** (3 次スプライン) ならば、区分的な直線はいくぶんなめらかな等高線が得られるように補間されますが、多少波打つ可能性があります。**bspline** (B-spline) は、より滑らかな曲線を描くことが保証されますが、これは z の等しい点の位置を近似しているだけです。

points — 最終的には，全ての描画は，区分的な直線で行われます。ここで指定する数は、**bspline** または **cubicspline** での近似に使われる線分の数を制御します。実際には cubicspline と bspline の区間 (曲線線分) の数は **points** と線分の数の積に等しくなります。

order — bspline 近似の次数です。この次数が大きくなるにつれて、等高線はなめらかになります (もちろん、高次の bspline 曲線になるほど、元の区分的直線からは離れていきます)。このオプションは **bspline** モードでのみ有効です。指定できる値は、2 (直線) から 10 までの整数です。

levels — 等高線のレベルの数は、**auto** (デフォルト), **discrete**, **incremental** と等高線のレベル数 <n> で制御します。<n> の値は、plot.h の中で 定義されている (標準では 30) MAX_DISCRETE_LEVELS を上限としています。

auto では、<n> は仮のレベルの数であり、実際のレベルの数は、簡単なラベルを生成するように調節されます。曲面の z 座標が zmin から zman の範囲にあるとき、等高線はその間の dz の整数倍になるように生成されます。ここで、dz は 10 のあるべき乗の 1, 2, 5 倍、のいずれかです (2 つの目盛りの間を丁度割り切るように)。

levels discrete では、等高線は指定された z = <z1>, <z2> ... に対して生成されます。指定した個数が等高線のレベルの個数となります。**discrete** モードでは、**set cntrparams levels <n>** という指定は常に無視されます。

incremental では、等高線は z = <start> から始まり、<increment> ずつ増えて行き限界の個数に達するまで書かれます。<end> はその等高線の数を決定するのに使われますが、これは後の **set cntrparam levels <n>** によって常に変更されます。

コマンド **set cntrparam** が引数なしに呼ばれた場合は、次のデフォルトの値が使われます: linear, 5

points, order 4, 5 auto levels

例:

```
set cntrparam bspline
set cntrparam points 7
set cntrparam order 10
```

以下はレベルの基準が合えば 5 個のレベルがに自動的に選択されます:

```
set cntrparam levels auto 5
```

以下は .1, .37, .9 にレベルを設定します:

```
set cntrparam levels discrete .1,1/exp(1),.9
```

以下は 0 から 4 まで、1 ずつ増やすレベルを設定します:

```
set cntrparam levels incremental  0,1,4
```

以下はレベルの数を 10 に設定します (増加の最後の値 (end) または自動で設定されるレベルの数は変更されます):

```
set cntrparam levels 10
```

以下はレベルの数は保持したままレベルの開始値と増分値を設定します:

```
set cntrparam levels incremental 100,50
```

等高線を描く場所の制御に関しては **set contour** を、等高線のラベルの書式と線種の制御に関しては **set clabel** を参照してください。

34.11 Contour

コマンド **set contour** は曲面の等高線を引くことを指示します。このオプションは **splot** でのみ有効です。

書式:

```
set contour {base | surface | both}
set nocontour
show contour
```

これらの 3 つのオプションは等高線をどこに引くかを指定します。**base** では等高線を x/y 軸の刻みのある底面に描かれ、**surface** では等高線はその曲面自体の上に描かれ、**both** では底面と曲面上の両方に描かれます。オプションが指定されていない場合は **base** であると仮定されます。

等高線の描画に影響を与えるパラメータについては **set cntrparam** を、等高線のラベルの制御に関しては **set clabel** を参照してください。

等高線のみのグラフが得るために、曲面自身の描画をしないようにすることもできます (**set surface** 参照)。**set size** を使って、グラフを画面一杯に描画することも可能ですが、そういった出力形式よりも、等高線のデータをファイルに書き出し、それを再び 2 次元データとして読み込んで描画すればよりよい制御が可能になります:

```
set nosurface
set contour
set cntrparam ...
set term table
set out 'filename'
splot ...
set out
# contour info now in filename
set term <whatever>
plot 'filename'
```

等高線を描くためには、データは格子状データ ("grid data") である必要があります。そのようなファイルでは、一つの y-孤立線上の全ての点が順に並べられていきます。そして隣の y-孤立線上の点が順に並べられ、そして隣、と続いていきます。y-孤立線同士を分離するには一行の空行 (空白、復帰、改行以外の文字を含まない行) を挟みます。**splot datafile** も参照してください。

非格子上データで等高線を描きたい場合は、格子を生成するために **set dgrid3d** を使用します。詳しいことは **set dgrid3d** の項目を参照してください。

34.12 Data style

コマンド **set data style** は、データの描画のデフォルトの plot スタイルを変更します。

書式:

```
set data style <style-choice>
show data style
```

選択可能な plot スタイルに関しては **set style** を参照してください。何も選択を与えずに実行すると選択可能なものの一覧が表示されます。**show data style** によって、現在のデフォルトのデータファイルの plot スタイルが表示されます。

34.13 Dgrid3d

コマンド **set dgrid3d** は、非格子状データから格子状データへの写像機能を有効にし、そのためのパラメータを設定します。

書式:

```
set dgrid3d {<row_size>} {,{<col_size>} {,<norm>}}
set nodgrid3d
show dgrid3d
```

デフォルトでは **dgrid3d** は無効になっています。有効になると、ファイルから読み込まれる 3 次元のデータは「散在した」データ (非格子状データ) であると見なされます。格子は、グラフと等高線の描画のために、散在したデータを囲む矩形から得られる寸法と、row_size/col_size で指定される数の行と列を持つように生成されます。格子は x 方向 (行) と y 方向 (列) に等間隔です。z の値は散在するデータの z の値の重み付きの平均として計算されます。

3 番目のパラメータであるノルム (norm) は、重み付けを制御するもので、各点は格子点からの距離の norm 乗の逆数で重み付けされます。(実際には、dx, dy を各データ点と格子点との差の成分であるとすると、重みは dx^norm + dy^norm で与えられます。2 のべきのノルム、特に 4, 8, 16 に関しては、その重みの計算はユークリッド距離を使うことで (dx^2+dy^2)^norm/2 のように最適化されてますが、任意の負でない整数を使うことも可能です。)

格子点に近いデータ点程それはその格子点により大きい影響を与え、ノルムの値が大きい程格子点から離れた点の影響は小さくなります。

dgrid3d オプションは散在するデータから格子状データに変換する簡単なローパスフィルタです。この問題に対するより洗練された手法が存在しますので、この単純な方法が不十分であれば、**gnuplot** の外でそのような方法でデータを前処理するべきでしょう。

(z の値は全てのデータ点の重み付けとして求められ、隣接するデータ点を補間しているのではありません。よって、端の影響が予期しない、望ましくない結果を引き起こす可能性があります。ある場合では、小さいノルムの値により局所的な平均ではなくデータ点の距離の平均を反映したような格子点が生成され、一方、大きなノルムの値により、隣接するデータ点を滑らかに変化させるのでなく最も近くのデータ点と同じ値を持つ格子点による階段 ("steps") を作ってしまいます。格子の領域内部は任意の境界条件に関する補外法により埋めることができますが、変数は正規化されず、よって x と y の単位が x, y 各方向に関して、点の相対的な重みに影響をおよぼすことになります。)

例:

```
set dgrid3d 10,10,1     # デフォルト
set dgrid3d ,,4
```

最初のものは、構成する格子を 10x10 にし、重みの計算のノルムは 1 にします。2 番目の例はノルムのみ 4 に変更します。

34.14 Dummy

コマンド **set dummy** はデフォルトの仮変数名を変更します。

書式:

```
set dummy {<dummy-var>} {,<dummy-var>}
show dummy
```

デフォルトでは、**gnuplot** は **plot** では、媒介変数モード、あるいは極座標モードでは "t", そうでなければ "x" を独立変数 (仮変数) とし、同様に **splot** では、媒介変数モードでは (**splot** は極座標モードでは使えません) "u" と "v", そうでなければ "x" と "y" を独立変数とします。

仮変数は、物理的に意味のある名前、あるいはより便利な名前として使う方が便利でしょう。例えば、時間の関数を描画する場合:

```
set dummy t
plot sin(t), cos(t)
```

このコマンドでは、少なくとも一つの仮変数が設定される必要があります。**set dummy** だけだとエラーメッセージが表示されます。

例:

```
set dummy u,v
set dummy ,s
```

第二の例は、2 番目の変数を s とします。

34.15 Encoding

コマンド **set encoding** は文字のコード化を選択します。指定できる値はいくつかあり、**default** は出力装置のデフォルトの値を使用する指示、**iso_8859_1** (PostScript の世界でいう **ISO-Lain1**) は多くの UNIX ワークステーション、MS-Windows などで使われているもので、**cp850** は OS/2 用、**cp437** は MS-DOS 用のものです。

書式:

```
set encoding {<value>}
show encoding
```

このコード化はどんな出力装置でもサポートされているとは限らず、そして出力装置は要求されたどんな非標準文字も生成できなければいけません。

34.16 Format

座標軸の刻みの見出しは、コマンド **set format** で書式を設定できます。

書式:

```
set format {<axes>} {"<format-string>"}
set format {<axes>} {'<format-string>'}
show format
```

ここで、<axes> (軸) は **x**, **y**, **z**, **xy**, **x2**, **y2**、または何も指定しない (**xy** と同じ) かのいずれかです。刻みの見出しの文字列の長さ ('printf' で整形された後の) は 100 文字まで、と制限されています。書式文字列 (<format-string>) を省略した場合、それはデフォルトの "%g" になります。LaTeX ユーザにはよく "$%g$" が好まれます。空の文字列 "" を指定した場合、刻み自身は表示されますが見出しはつきません。すべての刻みを消すには **set noxtics** や **set noytics** を使用してください。

改行文字 (\n) も書式文字列で使えます。それを解釈させるには、単一引用符 (') でなく (") を使ってください。**syntax** の項も参照してください。

デフォルトの書式文字列は両軸とも "%g" ですが、"%.2f" や "%3.0em" などの書式が好まれることも多いでしょう。倍精度小数に対して 'printf' と出力装置が受けつけることができる書式であればそれは正しく動作するでしょう。他にもいくつかのオプションが追加されています。書式文字列が浮動小数に対するもののようであれば **gnuplot** は妥当な文字列に変換しようとします。

"%" が頭につかない文字はそのまま表示されます。よって、書式文字列内にスペースや文字列などを入れることができます。例えば "%g m" とすれば、数値の後に " m" が表示されます。"%" 自身を表示する場合には "%g %%" のように 2 つ重ねます。

刻みに関するより詳しい情報については **set xtics** を参照してください。

34.16.1 Format specifiers

使用可能な書式 (時間/日付モードでない場合) は以下の通りです:

目盛りラベルの数値書式表記	
書式	説明
%f	固定小数点表記
%e, %E	指数表記; 指数の前に "e", "E" をつける
%g, %G	%e (または %E) と %f の略記
%x, %X	16 進表記
%o, %O	8 進表記
%t	10 進の仮数部
%l	現在の対数尺の底を基数とする仮数部
%s	現在の対数尺の底を基数とする仮数部; 補助単位 (scientific power)
%T	10 進の指数部
%L	現在の対数尺の底を基数とする指数部
%S	補助単位の指数部 (scientific power)
%c	補助単位文字
%P	π の倍数

補助単位 ('scientific' power) は、指数が 3 の倍数であるようなものです。補助単位指数 ("%c") の文字への変換は -18 から +18 までの指数に対してサポートされています。この範囲外の指数の場合、書式は通常の指数形式に戻ります。

ほかに使うことのできる修飾詞 ("%" と書式指定子の間に書くもの) には、次のいくつかがあります: "-" は数字を左詰めにし、"+" は正の数にも符号をつけ、"#" は小数点以下の数字が 0 だけであっても小数点をつけ、正の整数は出力幅を定め、出力幅指定の直前の "0" (文字でなく数字) は先頭に空いた部分を空白で埋める代わりに 0 で埋め、小数点の後に非負の整数を書いたものは精度を意味します (整数の場合は最小桁、小数の場合は小数点以下の桁数)。

これらの全ての修飾詞をサポートしていないリリースの 'printf' もあるでしょうし、逆にこれ以外のものをもサポートする 'printf' もあるでしょう。疑わしい場合は、適切な資料を調べ、そして実験してみてください。

例:

```
set format y "%t"; set ytics (5,10)          # "5.0" と "1.0"
set format y "%s"; set ytics (500,1000)      # "500" と "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)
#"1.0", "1.0", "1.5" と表示される (3 は 1.5 * 2^1 なので)
```

丸めと指数が必要となるような書式で 9.999 の様な数字が書かれる場合は問題が起こることがあります。

軸のデータ型が日時データ (time/date) の場合、書式文字列は 'strftime' 関数 ('gnuplot' 外。"man strftime" としてみてください) に関する有効な指定を行う必要があります。使える入力書式指定の一覧に関しては **set timefmt** を参照してください。

34.16.2 Time/date specifiers

日時データモード (time/date mode) では、次の書式が使用できます:

Tic-mark label Date/Time Format Specifiers	
書式	説明
%a	曜日名の省略形 (Sun,Mon,...)
%A	曜日名 (Sunday,Monday,...)
%b, %h	月名の省略形 (Jan,Feb,...)
%B	月名 (January,February,...)
%d	日 (1–31)
%D	"%m/%d/%y" の簡略形
%H, %k	時 (0–24)
%I, %l	時 (0–12)
%j	その年の通算日 (1–366)
%m	月 (1–12)
%M	分 (0–60)
%p	"am" または "pm"
%r	"%I:%M:%S %p" の簡略形
%R	"%H:%M" の簡略形
%S	秒 (0–60)
%T	"%H:%M:%S" の簡略形
%U	その年の通算週 (週は日曜日からと数える)
%w	曜日番号 (0–6, 日曜 = 0)
%W	その年の通算週 (週は月曜日からと数える)
%y	西暦 (0-99)
%Y	西暦 (4 桁)

数字を表す書式では、これらの指定子 (% の後ろ、指定子の前) に "0" ("オー" でなく "ゼロ") をつけることで、先頭に空白ができる場合に空白の代わりに 0 で埋めることができ、また最小の出力幅を正の整数で指定することもできます (出力される数字を表示するのに指定した幅が足りない場合は無視されます)。表示する文字の長さは 24 文字まで、という制限があり、長すぎた部分は切り捨てられます。

例:

日時のデータが "76/12/25 23:11:11" の場合

```
set format x                # デフォルトでは "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"
```

日時のデータが "98/07/06 05:04:03" の場合

```
set format x "%1y/%2m/%3d %01H:%02M:%03S"  # "98/ 7/  6 5:04:003"
```

34.17 Function style

set function style は関数の描画スタイルを変更します。

書式:

```
set function style <style-choice>
show function style
```

選択できる項目 (style-choice) については **set style** を参照してください。何も指定しないと選択できる項目が一覧表示されます。**show function style** は現在のデフォルトでの関数の描画スタイルを表示します。

34.18 Functions

show functions コマンドはユーザーが定義した関数とその定義内容を表示します .

書式:

```
  show functions
```

gnuplot における関数の定義とその使い方については **expressions** の項を参照してください。

34.19 Grid

コマンド **set grid** は格子線を描きます。

書式:

```
set grid {{no}{m}xtics} {{no}{m}ytics} {{no}{m}ztics}
         {{no}{m}x2tics} {{no}{m}y2tics}
         {polar {<angle>}}
         { {linestyle <major_linestyle>}
           | {linetype | lt <major_linetype>}
             {linewidth | lw <major_linewidth>}
           { , {linestyle | ls <minor_linestyle>}
               | {linetype | lt <minor_linetype>}
                 {linewidth | lw <minor_linewidth>} } }
set nogrid
show grid
```

格子線は任意の軸の任意の主目盛/副目盛に対して有効/無効にでき、その主目盛りと副目盛りに対する線種、線幅も指定でき、現在の出力装置がサポートする範囲で、あらかじめ定義したラインスタイルを使用することもできます。

さらに、2 次元の描画では極座標格子も使うことができます — 定義可能な区間に対して、選択された目盛りを通る同心円と中心からの放射状の線が描かれます (その区間は **set angles** の設定にしたがって度、またはラジアンで指定します)。極座標格子は現在は極座標モードでは自動的には生成されないことに注意してください。

set grid が描く前に、必要な目盛りは有効になっていなければなりません。**gnuplot** は、存在しない目盛りに対する格子の描画の命令は単に無視します。しかし、後でその目盛りが有効になればそれに対する格子も描きます。

副格子線に対する線種を何も指定しなければ、主格子線と同じ線種が使われます。デフォルトの極座標の角度は 30 度です。

デフォルトでは、格子線の幅は通常の線幅の半分です。主線幅 (major linwidth) と副線幅 (minor linewidth) に対する修飾詞はこのデフォルトの値を縮尺します。例えば **set grid lw .5** とすれば、格子線幅は通常の線幅の 1/4 の幅になります。

z の格子線は描画の背景に描かれます。これは描画の周りに部分的な箱が描画されている場合にはいいでしょう — **set border** を参照してください。

34.20 Hidden3d

set hidden3d コマンドは曲面描画 (**splot** 参照) で隠線処理を行なうように指示します。その処理の内部アルゴリズムに関する追加機能もこのコマンドで制御できます。

書式:

```
set hidden3d {defaults} |
             { {{offset <offset>} | {nooffset}}
               {trianglepattern <bitpattern>}
               {{undefined <level>} | {noundefined}}
               {{no}altdiagonal}
               {{no}bentover} }
set nohidden3d
show hidden3d
```

gnuplot の通常の表示とは異なり、隠線処理では与えられた関数、またはデータの格子線を、実際の曲面がその曲面の背後にあって隠されている部分は見せないのと同じように処理します。これを可能にするためには、曲面は ' 格子状' (**splot datafile** 参照) である必要があり、またそれらは **with lines** か **with linespoints** で描かれていなければいけません。

hidden3d が有効なときは、格子線だけでなく、面部分や土台の上の等高線 (**set contour** 参照) も隠されます。複数の面を描画している場合は、各曲面は自分自身と他の曲面で隠される部分も持ちます。曲面上への等高線の表示 (**set contour surface**) は機能しません。見出しと矢印は常に表示され、影響を受けません。グラフの説明 (key) も曲面に隠されることはありません。

関数値は格子孤立線の交点で評価されます。見ることの出来る線分を求めるときは個々の関数値、あるいはデータ点の間はそのアルゴリズムによって線形補間されます。これは、**hidden3d** で描画する場合と **nohidden3d** で描画する場合で関数の見かけが異なることを意味します。なぜならば、後者の場合関数値は各標本点で評価されるからです。この違いに関する議論については、**set samples** と **set isosamples**

を参照してください。

曲面の隠される部分を消去するのに使われるアルゴリズムは、このコマンドで制御されるいくつかの追加オプションを持っています。**defaults** を指定すればそれらはすべて、以下で述べるようなデフォルトの値に設定されます。**defaults** が指定されなかった場合には、明示的に指定されたオプションのみが影響を受け、それ以外のものは以前の値が引き継がれます。よって、それらのオプションの値をいちいち修正することなく、単に **set {no}hidden3d** のみで隠線処理をオン/オフできることになります。

最初のオプション **offset** は ' 裏側' の線を描画する線の線種に影響を与えます。通常は曲面の表裏を区別するために、裏側の線種は、表側の線種より一つ大きい番号の線種が使われます。**offset <offset>** によって、その追加する値を、デフォルトの 1 とは異なる増分値に変更できます。**nooffset** オプションは **offset 0** を意味し、これは表裏で同じ線種を使うことになります。

次のオプションは **trianglepattern <bitpattern>** です。<bitpattern> は 0 から 7 までの数字で、ビットパターンと解釈されます。各曲面は三角形に分割されますが、このビットパターンの各ビットはそれらの三角形の各辺の表示を決定します。ビット 0 は格子の水平辺、ビット 1 は格子の垂直辺、ビット 2 は、元々の格子が 2 つの三角形に分割されるときの対角辺です。デフォルトのビットパターンは 3 で、これは全ての水平辺と垂直辺を表示し、対角辺は表示しないことを意味します。対角辺も表示する場合は 7 を指定します。

オプション **undefined <level>** は、定義されていない (欠けているデータまたは未定義の関数値) か、または与えられた x,y,z の範囲を超えているデータ点に適用させるアルゴリズムを指示します。そのような点は、それでも表示されてしまうか、または入力データから取り除かれます。取り除かれてしまう点に接する全ての曲面要素は同様に取り除かれ、よって曲面に穴が生じます。<level> = 3 の場合、これは **noundefined** と同じで、どんな点も捨てられません。これは他の場所であらゆる種類の問題を引き起こし得るので使わないべきです。<level> = 2 では未定義の点は捨てられますが、範囲を超えた点は捨てられません。<level> = 1 では、これがデフォルトですが、範囲を超えた点も捨てられます。

noaltdiagonal を指定すると、**undefined** が有効のとき (すなわち <level> が 3 でない場合) に起こる以下の場合のデフォルトでの取扱いを変更できます。入力曲面の各格子状の部分は一方の対角線によって 2 つの三角形に分割されます。通常はそれらの対角線の全てが格子に対して同じ方向を向いています。もし、ある格子の 4 つの角のうち一つが **undefined** 処理によりとり除かれていて、その角が通常の方向の対角線に乗っている場合は、その両方の三角形が取り除かれてしまいます。しかし、もしデフォルトの設定である **altdiagonal** が有効になっている場合、その格子については他方向の対角線が代わりに選択され、曲面の穴の大きさが最小になるようにします。

bentover オプションは今度は **trianglepattern** とともに起こる別のことを制御します。かなりしわくちゃの曲面では、下の ASCII 文字絵に書いたように、曲面の 1 つの格子が 2 つに分けられた三角形の表と裏の反対側が見えてしまう場合 (すなわち、元の四角形が折り曲げられている ('bent over') 場合) があります:

```
                                                          C----B
    元の 4 角形:      A--B      表示される 4 角形:        |\   |
 ("set view 0,0")     | /|    ("set view 75,75" perhaps)  | \  |
                      |/ |                                 |  \ |
                      C--D                                 |   \|
                                                           A    D
```

曲面の格子の対角辺が <bitpattern> の 2 bit によって見えるようにはなってはいない場合、上の対角辺

CB はどこにも書かれないことになり、それが結果の表示を理解しにくいものにします。デフォルトで定義される **bentover** オプションは、このような場合それを表示するようにします。もしそうしたくないなら、**nobentover** を選択してください。

34.21 Isosamples

関数を面として描画する場合の孤立線 (格子) の密度はコマンド **set isosamples** で変更できます。

書式:

```
set isosamples <iso_1> {,<iso_2>}
show isosamples
```

各曲面グラフは <iso_1> 個の u-孤立線と <iso_2> 個の v-孤立線を持ちます。<iso_1> のみ指定すれば、<iso_2> は <iso_1> と同じ値に設定されます。デフォルトでは、u, v それぞれ 10 本の標本化が行われます。標本数をもっと多くすればより正確なグラフが作られますが、時間がかかります。これらのパラメータは、データファイルの描画には何も影響を与えません。

孤立線とは、曲面の一つの媒介変数を固定して、もう一つの媒介変数によって描かれる曲線のことです。孤立線は、曲面を表示する単純な方法を与えます。曲面 s(u,v) の媒介変数 u を固定することで u-孤立線 c(v) = s(u0,v) が作られ、媒介変数 v を固定することで v-孤立線 c(u) = s(u,v0) ができます。

関数の曲面グラフが隠線処理なしで描かれている場合、**set samples** は各孤立線上で標本化される点の数を制御します。**set samples** と **set hidden3d** も参照してください。等高線描画ルーチンは、関数の点の標本化は各孤立線の交点で行われると仮定しているので、関数の曲面と等高線の解像度を変更するときは、**isosamples** と同じように **samples** を変更するのが望ましいでしょう。

34.22 Key

コマンド **set key** は描画された曲線の説明や表題を表示することを可能にします。

説明 (key) の内容、すなわち描画される個々のデータ集合や関数につける名前、およびそれらグラフの曲線とグラフ上の点を表す記号からなるサンプルは、**plot** (または **splot**) コマンドの **title**, **with** オプションにより決定されます。より詳しい情報については **plot title**, **plot with** を参照してください。

書式:

```
set key {  left | right | top | bottom | outside | below
         | <position>}
        {Left | Right} {{no}reverse}
        {samplen <sample_length>} {spacing <vertical_spacing>}
        {width <width_increment>}
        {title "<text>"}
        {{no}box { {linestyle | ls <line_style>}
                   | {linetype | lt <line_type>}
                     {linewidth | lw <line_width>}}}
set nokey
show key
```

デフォルトでは説明 (key) はグラフの右上の角に置かれます。説明をグラフの他の角、あるいはグラフの外の右や下に配置するために、**left**, **right**, **top**, **bottom**, **outside**, **below** といったキーワードが用意されています。これらは単独で、あるいは組み合わせて使います。

説明中のラベル (名前) の行揃えは **Left**, **Right** (デフォルト) で指示します。ラベル文字列と曲線のサンプルは左右入れ替えることができます (**reverse**) し、 全体を枠で囲むこともできます (**box {...}**)。その枠の線は、線種 (**lintype**), 線幅 (**linewidth**)、あるいは定義済のラインスタイル (**linestyle**) を指定することもできます。ただ、全ての出力装置が線幅の選択をサポートしているとは限らないことに注意してください。

グラフ曲線のサンプルの線分の長さは **samplen** で指定できます。その長さは目盛りの長さと、<sample_length>*(文字幅) の和として計算されます。**sapmlen** は、グラフ上の点のサンプルの位置にも (もしそれが書かれなくても) 影響を与えています。それは、点の記号はサンプル線分の中央に書かれるためです。<sample_length> は整数値でなければなりません。

行間の垂直スペースは、**spacing** で指定できます。その幅は、点のサイズ (pointsize) と垂直な目盛りのサイズと <vertical_spacing> の積になります。この垂直スペースは、文字の高さよりも小さくはならないことが保証されています。

<width_increment> は、文字列の長さに加えたり減らしたりする文字幅を表す数値です。これは、説明に外枠を書き、ラベル文字列の文字数を調節するときには役立つでしょう。**gnuplot** は外枠の幅を計算するときは、ラベル文字列の文字数を単純に数えるだけなので、それを修正するのに使えます。

表題 (title) を説明の上につけることもできます (**title "<text>"**) – 単一引用符 (') と二重引用符 (") の違いについては **syntax** を参照してください。説明の表題の行揃えは、グラフの表題の行揃えと同じものが使われます。

set key のデフォルトは、**right**, **top**, **Right**, **noreverse**, **samplen 4**, **spacing 1.25**, **title ""**, **nobox** です。説明の枠の線種はデフォルトではグラフ描画の外枠と同じものが使われます。**set key** にオプションをつけずに指定するとデフォルトの設定に戻ります。

説明の位置 (<position>) は、以前のバージョンと同様単に x,y,z を指定してもいいですが、その座標の座表系を選択するための 4 つのキーワード (**first**, **second**, **graph**, **screen**) を頭につけることもできます。詳細は **coordinates** を参照してください。

説明は、1 行に 1 曲線分ずつの数行のまとまりとして書かれます。各行の右側には (**reverse** を使っていれば左側には) その曲線と同じ種類の直線のサンプルが引かれ、他の側には **plot** コマンドから得られる文字列 (title) が置かれます。これらの行は、架空の直線が説明の左側と右側を分けるかのように垂直に整列されます。コマンド **set key** で指定する座標はこの架空の線分の上の端の座標です。**plot** では直線の位置を指定するために x と y だけが使われ、**splot** では、x, y, z の値全てを使い、グラフを 2 次元面へ投影するのと同じ方法を使って、架空の直線の 2 次元画面での位置を生成します。

場合によってはこの説明の一部または全部が境界の外にはみ出すこともあります。この場合見出しなどと重なってしまうこともあり、また出力装置によってはエラーを生じることもあります。**outside** や **below** を使って説明をグラフの外に出す場合は **gnuplot** は説明に余白を空けるためグラフは多少小さくなります。右の外に出す場合は、その説明の横幅が可能な限り小さくなるようにし、下の外に出す場合は、横幅が可能な限り大きくなるようにします (その幅はラベルの長さによります)。そのようにして、可能な限りグラフからは小さいスペースしか搾取しないようにしています。

TeX, PostScript, またはこれらと同等の、整形情報が文字列に埋め込まれる出力を使う場合は、**gnuplot**

は説明の位置合わせのための文字列の幅を正しく計算できません。よって説明を左に置く場合は **set key left Left reverse** という組合せを使うのがいいでしょう。そうすれば説明の枠と枠内のすき間は文字列そのままの幅に合わせられます。

splot で等高線を書く場合、説明には等高線のラベルも表示されます。これらのラベルの並び具合がうまくいかない、または別な位置に小数点を置きたい場合はそのラベルの書式を指定できます。詳細は **set clabel** を参照してください。

例:

以下はデフォルトの位置に説明を表示します:

```
set key
```

以下は説明を表示しなくします:

```
set nokey
```

以下はデフォルトの (第一の) 座標系での (2,3.5,2) の位置に説明を表示します:

```
set key 2,3.5,2
```

以下は説明をグラフの下に表示します:

```
set key below
```

以下は説明を左下角に表示し、テキストは左に行揃えで、タイトルをつけ、線種 3 の外枠を書きます:

```
set key left bottom Left title 'Legend' box 3
```

34.23 Label

set label コマンドを使うことによって任意の見出しをグラフ中に表示することができます。

書式:

```
set label {<tag>} {"<label_text>"} {at <position>}
          {<justification>} {{no}rotate} {font "<name><,size>"}
set nolabel {<tag>}
show label
```

位置 (<position>) は x,y か x,y,z のどちらかで指定し、座標系を選択するにはその座標の前に **first**, 'second', 'graph', 'screen' をつけるます。詳細は **coordinates** の項を参照してください。

タグ (<tag>) は見出しを識別するための整数値です。タグを指定しなかった場合未使用のもので最も小さい値が自動的に割り当てられます。現在の見出しを変更するときはそのタグと変更したい項目を指定して **set label** コマンドを使います。

デフォルトでは、指定した点 x,y,z に見出しの文章の左端が来るように配置されます。x,y,z を見出しのどこに揃えるかを変更するには変数 <justification> を指定します。これには、**left**, **right**, **center** のいずれかが指定でき、それぞれ文章の左、右、真中が指定した点に来るように配置さるようになります。描画範囲の外にはみ出るような指定も許されますが、座標軸の見出しや他の文字列と重なる場合があります。

rotate を指定するとラベルは縦書きになります (もちろん出力装置が対応
していれば、ですが)。

もし一つ (あるいはそれ以上の) 軸が時間軸である場合、座標は **timefmt** の書式にしたがって引用符で囲まれた文字列で与える必要があります。**set xdata** と **set timefmt** を参照してください。

EEPIC, Imagen, LaTeX, TPIC で出力する場合は、\\ を使うことで見出しを改行させることができます。

例:

(1,2) の位置に "y=x" と書く場合:

```
set label "y=x" at 1,2
```

Symbol フォントのサイズ 24 の "シグマ" (Σ) をグラフの真中に書く場合:

```
set label "S" at graph 0.5,0.5 center font "Symbol,24"
```

見出し "y=x^2" の右端が (2,3,4) に来るようにし、タグ番号として 3 を使う場合:

```
set label 3 "y=x^2" at 2,3,4 right
```

その見出しを中央揃えにする場合:

```
set label 3 center
```

タグ番号 2 の見出しを削除する場合:

```
set nolabel 2
```

全ての見出しを削除する場合:

```
set nolabel
```

全ての見出しをタグ番号順に表示する場合:

```
show label
```

x 軸が時間軸であるグラフに見出しを設定する例:

```
set timefmt "%d/%m/%y,%H:%M"
set label "Harvest" at "25/8/93",1
```

34.24 Linestyle

出力装置にはおのおのデフォルトの線種と点種の集合があり、それらはコマンド **test** で見ることができます。**set linestyle** は線種と線幅、点種と点の大きさを、個々の呼び出しで、それらの情報を全部指定する代わりに、単なる番号で参照できるようにあらかじめ定義するものです。

書式:

```
set linestyle <index> {linetype | lt <line_type>}
                      {linewidth | lw <line_width>}
                      {pointtype | pt <point_type>}
                      {pointsize | ps <point_size>}
set nolinestyle
show linestyle
```

線種と点種現在の出力装置が持つデフォルトの種類から選ばれます。線幅と点の大きさはデフォルトの幅、大きさに対する乗数です (しかし、ここでの <point_size> は、**set pointsize** で与えられる乗数には影響を受けないことに注意してください)。

線種と線幅のデフォルトの値はそのラインスタイル番号 (index) です。幅と大きさのデフォルトの大きさはどちらも 1 です。

このようにつくられるラインスタイルは、デフォルトの型 (線種, 点種) を別なものに置き換えることはしないので、ラインスタイル、デフォルトの型、どちらも使えます。

全ての出力装置が **linewidth** や **pointsize** をサポートしているわけではありません。もしサポートされていない場合はそれらのオプションは無視されます。

この機能は完全に実行されるとは限らないことに注意してください。このように定義されるラインスタイルは、'plot', 'splot', 'replot', 'set arrow' などでは使えますが、'set grid' のように、デフォルトの番号 (index) を使うことが許されているようなコマンドでは使えません。

例: 以下では、番号 1, 2, 3 に対するデフォルトの線種をそれぞれ赤、緑、青とし、デフォルトの点の形をそれぞれ正方形、十字、三角形であるとします。このとき以下のコマンド

```
set linestyle 1 lt 2 lw 2 pt 3 ps 0.5
```

は、新しいラインスタイルとして、緑でデフォルトの 2 倍の幅の線、および三角形で半分の幅の点を定義します。また、以下のコマンド

```
set function style lines
plot f(x) lt 3, g(x) ls 1
```

は f(x) はデフォルトの青線で、g(x) はユーザの定義した緑の線で描画します。同様に、コマンド

```
set function style linespoints
plot p(x) lt 1 pt 3, q(x) ls 1
```

は、p(x) を赤い線で結ばれたデフォルトの三角形で、q(x) は緑の線で結ばれた小さい三角形で描画します。

34.25 Lmargin

コマンド **set lmargin** は左の余白のサイズをセットします。詳細は **set margin** を参照してください。

34.26 Locale

locale の設定は **{x,y,z}{d,m}tics** が書く日付の言語を決定します。

書式:

```
set locale {"<locale>"}
```

<locale> にはインストールされたシステムで使うことの出来る任意の言語を指定できます。可能なオプションについてはシステムのドキュメントを参照してください。デフォルトの値は環境変数 LANG から決定されます。

34.27 Logscale

対数スケールは，x, y, z, x2, y2 の各軸について設定できます。

書式:

```
set logscale <軸> <底>
set nologscale <軸>
show logscale
```

ここで，< 軸 > は，**x**, **y**, **z** の任意の順序、または 'x2', 'y2' による組み合せが可能です．また，< 底 > は，対数スケールの底です．< 底 > が指定されなかった場合は，10 になります．もし，< 軸 > が指定されなかった場合は，全部が指定されたことになります．**set nologscale** は、指定した軸の対数スケールを解除します。

例:

x, z 両軸について対数スケールを設定する:

```
set logscale xz
```

y 軸について底 2 とする対数スケールを設定する:

```
set logscale y 2
```

z 軸の対数スケールを解除する:

```
set nologscale z
```

34.28 Mapping

データが **splot** に球面座標や円柱座標で与えられた場合、**set mapping** コマンドは **gnuplot** にそれをどのように扱うかを指定するのに使われます。

書式:

```
set mapping {cartesian | spherical | cylindrical}
```

デフォルトではカーテシアン座標 (通常の x,y,z 座標) が使われます。

球面座標では、データは 2 つか 3 つの列 (またはその個数の **using** エントリ) として与えられます。最初の 2 つは方位角 (theta) と仰角 (phi) とみなされます (**set angles** で設定された単位で)。半径 r は、もし 3 列目のデータがあればそれが使われ、もしなければ 1 に設定されます。各変数の x,y,z との対応は以下の通りです:

```
x = r * cos(theta) * cos(phi)
y = r * sin(theta) * cos(phi)
z = r * sin(phi)
```

これは、"極座標系" というより、むしろ "地学上の座標系" (緯度、経度) に相当することに注意してください。

円柱座標では、データはやはり 2 つか 3 つの列で与えられ、最初の 2 つは theta (**set angle** で指定された単位の) と z と見なされます。半径 r は球面座標の場合と同様、3 列目のデータがあればそれが、なければ 1 と設定されます。各変数の x,y,z との対応は以下の通りです:

```
x = r * cos(theta)
y = r * sin(theta)
z = z
```

mapping の効果は、**splot** コマンド上の **using** によるフィルタで実現することも可能ですが、多くのデータファイルが処理される場合は **mapping** の方が便利でしょう。しかし、**mapping** を使っていても、もしファイルのデータの順番が適切でなかったら結局 **using** が必要になってしまいます。

mapping は **plot** では何もしません。

34.29 Margin

自動的に計算される周囲の余白 (margin) はコマンド **set margin** で変更できます。**show margin** は現在の設定を表示します。

書式:

```
set bmargin {<margin>}
set lmargin {<margin>}
set rmargin {<margin>}
set tmargin {<margin>}
show margin
```

<margin> の単位には、適切と思われる、文字の高さと幅が使われます。正の値は余白の絶対的な大きさを定義し、負の値 (または無指定) は **gnuplot** によって自動計算される値を使うことになります。

描画の余白は通常目盛り、目盛りの見出し、軸の見出し、描画のタイトル、日付、そして境界の外にある場合の key (グラフ見出し) のサイズ等を元に計算されます。しかし、目盛りの刻みが境界でなく軸の方についている場合 (例えば **set xtics axis** によって)、目盛りの刻み自身とその見出しは余白の計算には含まれませんし、余白に書かれる他の文字列の位置の計算にも含まれません。これは、軸と境界が非常に近い場合、軸の見出しが他の文字列を上書きする可能性を示唆します。

34.30 Missing

コマンド **set missing** は、欠けているデータを記述するのに使われる文字を **gnuplot** に伝えるために使われます。

書式:

```
set missing {"<character>"}
show missing
```

例:

```
set missing "?"
```

は、データファイルが以下のような場合、

```
1 1
2 ?
3 2
```

真中の列を無視することを意味します。

missing に対するデフォルトの文字は設定されていません。

34.31 Multiplot

コマンド **set multiplot** は **gnuplot** を多重描画モードにします。これは複数の描画を同じページ、ウィンドウ、スクリーンに表示するものです。

書式:

```
set multiplot
set nomultiplot
```

出力形式 (terminal) によっては、コマンド **set nomultiplot** が与えられるまで何の描画も表示されないことがあります。この場合このコマンドによりページ全体の描画が行なわれ、**gnuplot** は標準の単一描画モードになります。それ以外の出力形式では、各 **plot** コマンドがそれぞれ描画を行ないますがその間で前の描画が消されてしまうことはありません。

定義済の見出しやベクトルは、各描画において、毎回現在のサイズと原点に従って書かれます (それらが **screen** 座表系で定義されていない場合)。それ以外の全ての **set** で定義されるものも各描画すべてに適用されます。もし 1 度の描画にだけ現われて欲しいものを作りたいなら、それが例えば日付 (timestamp) だとしたら、**set multiplot** と **set nomultiplot** で囲まれたブロック内の **plot** (または **splot**, **replot**) 命令の一つを **set time** と **set notime** ではさんでください。

コマンド **set origin** と **set size** 各描画で正しい位置に設定する必要があります。詳細は **set origin** と **set size** の項目を参照してください。

例:

```
set size 0.7,0.7
set origin 0.1,0.1
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)
set nomultiplot
```

は、cos(x) のグラフを、sin(x) の上に積み重ねて表示します。最初の **set size** と **set origin** に注意してください。これらはなくてもいいのですがそれを入れておくことを勧めます。ある種の出力ドライバは、描画が一つでも作られる前に全体の外枠の情報が参照できることを要求します。そして、上のやり方は、その外枠が最初の描画のための外枠ではなく、描画列全体を含む外枠であるということを保証しています。

set size と **set origin** は全体の描画領域を参照しそれは各描画で利用されます。描画境界を一列に揃えたいならば、**set margin** コマンドで、境界の外の余白サイズを同じサイズに揃えることが出来ます。その使用に関しては **set margin** を参照してください。余白サイズは文字サイズ単位の絶対的な数値単位を

使用することに注意してください。よって残ったスペースに描かれるグラフは表示するデバイスの表示サイズに依存します。例えば、プリンタとディスプレイの表示は多分違ったものになるでしょう。

34.32 Mx2tics

x2 (上) 軸の小目盛り刻みの印は **set mx2tics** で制御されます。**set mxtics** を参照してください。

34.33 Mxtics

x 軸の小目盛り刻みの印は **set mxtics** で制御されます。**set nomxtics** によってそれを表示させなくすることが出来ます。同様の制御コマンドが各軸毎に用意されています。

書式:

```
set mxtics {<freq> | default}
set nomxtics
show mxtics
```

これらの書式は **mytics**, **mztics**, **mx2tics**, **my2tics** に対しても同じです。

<freq> は大目盛り間の、小目盛りによって分割される区間の数 (小目盛りの数ではありません) です。通常の線形軸に対してはデフォルトの値は 10 で、よって大目盛り間に 9 つの小目盛りが入ることになります。**default** を指定することによって小目盛りの数はデフォルトの値に戻ります。

軸が対数軸である場合、分割区間の数はデフォルトでは有意な数にセットされます (10 個の長さを元にして)。<freq> が与えられていればそちらが優先されます。しかし、対数軸では通常の小目盛り (例えば 1 から 10 までの 2, 3,
..., 8, 9 の刻み) は、9 つの部分区間しかありませんが、<freq> の設定は

10 とすることでそうなります。

小目盛りは大目盛りが一様に配置されている場合にのみ使えます。大目盛りは **set {x|x2|y|y2|z}tics** コマンドによって任意の場所に配置できるので、それが明示的に指定された場合には小目盛りは使えないことになります。

デフォルトでは小目盛りの表示は、線形軸ではオフで、対数軸ではオンになっています。その設定は、大目盛りに対する **axis|border** と **{no}mirror** の指定を継承します。これらに関する情報については **set xtics** を参照してください。

34.34 My2tics

y2 (右) 軸の小目盛り刻みの印は **set my2tics** で制御されます。**set mxtics** を参照してください。

34.35 Mytics

y 軸の小目盛り刻みの印は **set mytics** で制御されます。**set mxtics** を参照してください。

34.36 Mztics

z 軸の小目盛り刻みの印は **set mztics** で制御されます。**set mxtics** を参照してください。

34.37 Offsets

オフセットは、自動縮尺されたグラフの中のデータの周りに境界を置く仕組みを提供します。

書式:

```
set offsets <left>, <right>, <top>, <bottom>
set nooffsets
show offsets
```

各オフセットは定数、または数式が使え、それらのデフォルトの値は 0 です。左右のオフセットは x 軸と同じ単位で指定し、上下のオフセットは y 軸と同じ単位で指定します。正のオフセットの値はグラフを指定された方向へ伸ばします。例えば正の下方向のオフセットは y の最小値をより小さな値にします。許されている範囲での負のオフセットは、自動縮尺、あるいはクリッピングとの思いもよらぬ結果を生む可能性があります。

オフセットは **splot** では無視されます。

例:

```
set offsets 0, 0, 2, 2
plot sin(x)
```

この sin(x) のグラフの y の範囲は [-3:3] になります。それは、関数の y の範囲は [-1:1] に自動縮尺されますが、垂直方向のオフセットがそれぞれ 2 であるためです。

34.38 Origin

コマンド **set origin** はスクリーン上で曲面描画の原点を指定 (すなわち、グラフとその余白) するのに使用します。その座標系はスクリーン座標系 (**screen**) で与えます。この座標系に関する情報については **coordinates** を参照してください。

書式:

```
set origin <x-origin>,<y-origin>
```

34.39 Output

デフォルトでは、グラフは標準出力に表示されます。コマンド **set output** はその出力を指定されたファイルやデバイスにリダイレクトします。

書式:

```
set output {"<filename>"}
show output
```

ファイル名は引用符で囲まなければなりません。ファイル名が省略された場合は、直前の **set output** で開かれたファイルがクローズされ、新たな出力が標準出力 (STDOUT) に送られます。(もし、**set output "STDOUT"** とすると出力は "STDOUT" という名前のファイルに送られるかもしれません！ ["かもしれない" というのは、例えば **x11** などの terminal (出力形式) では **set output** が無視されるからです。])

MSDOS のユーザは次のことに注意すべきです: 文字 '\' は 2 重引用符の中では特別な意味を持ちます。よって、別のディレクトリにあるファイル名を指定する場合は単一引用符を用いるべきでしょう。

set terminal と **set output** の両方を指定する場合、**set terminal** を先に指定する方が安全です。それは、ある種の terminal では、OS が必要とするフラグをセットすることがあるからです。例えば、OS がファイルを開くときに (礼儀良く) ファイルがフォーマットされているかどうかを知る必要があるような OS などがそれ該当します。

popen 関数を持つようなマシン (Unix 等) では、ファイル名の最初を '|' とすることにより、出力をシェルコマンドにパイプで渡すことが可能です。例えば以下の通りです:

```
set output "|lpr -Plaser filename"
set output "|lp -dlaser filename"
```

MSDOS では、**set output "PRN"** とすると標準のプリンタに出力されます。VMS では出力は任意のスプール可能なデバイスに送ることが出来ます。出力を DECnet 透過なタスクに送ることも可能で、それはある種の柔軟性を与えてくれます。

34.40 Parametric

set paramaetric コマンドは **plot** および **splot** の意味を通常の関数描画から媒介変数表示 (parametric) 関数描画に変更します . **set noparametric** を使えば元の描画モードに戻ります .

書式:

```
set parametric
set noparametric
show parametric
```

2 次元グラフにおいては , 媒介変数表示関数はひとつの媒介変数に対する 2 つの関数で定められます . 例としては plot sin(t),cos(t) とすることによって円が描けます (アスペクト比が正しく設定されていれば – **set size** 参照) . **gnuplot** は、両方の関数が媒介変数による **plot** のために与えられていなければエラーメッセージを出します。

3 次元グラフにおいては面は x = f(u,v), y = g(u,v), z = h(u,v) で定められます . よって 3 つの関数を組で指定する必要があります . 例としては , **cos(u)*cos(v),cos(u)*sin(v),sin(u)** とすることによって球面が描けます . **gnuplot** は、3 つ全部の関数が媒介変数による **splot** のために与えられていなければエラーメッセージを出します。

これによって表現できる関数群は , 単純な f(x) 型の関数群の内包することになります . なぜならば , 2 つ (3 つ) の関数は x , y (, z) の値を独立に計算する記述ができるからです . 実際 , t,f(t) のグラフは , 一番目の関数のような恒等関数を用いて x の値が計算される場合に f(x) によって生成されるグラフと等価です . 同様に、3 次元での u,v,f(u,v) の描画は f(x,y) と等価です。

媒介変数表示関数は , x の関数 , y の関数 (, z の関数) の順に指定し , それらは共通の媒介変数およびその変域で定義されることに留意して下さい .

さらに，**set parametric** の指定は，新しい変数変域を使用することを暗に宣言します．通常の f(x) や f(x,y) が xrange, yrange (, zrange) を使用するのに対して，媒介変数モードではそれに加えて， trange, urange, vrange を使用します．これらの変域は **set trange**, **set urange**, **set vrange** によって直接指定することも，**plot** や **splot** で指定することもできます．現時点では，これらの媒介変数のデフォルトの変域は [-5:5] となっています．将来的にはこれらのデフォルト値をもっと有意なものに変更する予定です．

34.41 Pointsize

コマンド **set pointsize** は描画で使われる点の大きさを変更します。

書式:

```
set pointsize <multiplier>
show pointsize
```

デフォルトは 1.0 倍です。画像データ出力では、大きいポイントサイズの方が見やすいでしょう。

一つの描画に対するポイントサイズは **plot** コマンドの上でも変更できます。詳細は **plot with** を参照してください。

ポイントサイズの設定は、必ずしも全ての出力形式でサポートされているわけではないことに注意してください。

34.42 Polar

コマンド **set polar** はグラフの描画方法を xy 直交座標系から極座標系に変更します。

書式:

```
set polar
set nopolar
show polar
```

version 3.7 では、極座標モードにおいていくつか変更がなされ、よって、version 3.5 やそれ以前の版用のスクリプトには修正が必要になるでしょう。主な変更は、仮変数 t が角度として使われるようになったことで、それにより x と y の値の範囲が独立に制御できるようになりました。その他の変更は以下の通りです: 1) 目盛りはもう 0 軸に自動的にはつきません– **set xtics axis nomirror**; **set ytics axis nomirror** を使ってください 2) 格子が選択されてもそれは自動的に極座標には従いません– **set grid polar** を使ってください 3) 格子は角度に関してはラベル付けされません– 必要なら **set label** を使ってください

極座標モードでは、仮変数 (t) は角度を表します。t のデフォルトの範囲は [0:2*pi] ですが、単位として度が選択されていれば [0:360] となります (**set angles** 参照)。

コマンド **set nopolar** は描画方法をデフォルトの xy 直交座標系に戻します。

set polar コマンドは **splot** ではサポートされていません。**splot** に対する同様の機能に関しては **set mapping** を参照してください。

極座標モードでは t の数式の意味は r=f(t) となり、t は回転角となります。trange は関数の定義域 (角度) を制御し、xrange と yrange はそれぞれグラフの x,y 方向の範囲を制御することになります。これらの範

囲と rrange は自動的に設定されるか、または明示的に設定できます。これらすべての **set range** コマンドの詳細に関しては **set xrange** の項を参照してください。

例:

```
set polar
plot t*sin(t)
plot [-2*pi:2*pi] [-3:3] [-3:3] t*sin(t)
```

最初の **plot** はデフォルトの角度の範囲の 0 から 2*pi を使います。半径とグラフのサイズは自動的に縮尺されます。2 番目の **plot** は角度の定義域を拡張し、グラフのサイズを x,y のいずれの方向にも [-3:3] に制限します。

set size square とすると **gnuplot** はアスペクト比 (縦横の比) を 1 にするので円が (楕円でなく) 円に見えるようになります。

34.43 Rmargin

コマンド **set rmargin** は右の余白のサイズをセットします。詳細は **set margin** を参照してください。

34.44 Rrange

コマンド **set rrange** は極座標モードのグラフの半径方向の範囲を設定します。詳細は **set xrange** を参照してください。

34.45 Samples

関数、またはデータの補間に関するサンプリング数はコマンド **set samples** で変更できます。

書式:

```
set samples <samples_1> {,<samples_2>}
show samples
```

デフォルトではサンプル数は 100 点と設定されています。この値を増やすとより正確な描画が出来ますが遅くなります。このパラメータはデータファイルの描画には何の影響も与えませんが、補間/近似のオプションが使われている場合はその限りではありません。2 次元描画については **plot smooth** を、3 次元描画に関しては **set dgrid3d** を参照してください。

2 次元のグラフ描画が行なわれるときは <samples_1> の値のみが関係します。

隠線処理なしで曲面描画が行なわれるときは、samples の値は孤立線毎に評価されるサンプル数の指定になります。各 v-孤立線は <samples_1> 個のサンプル点を持ち、u-孤立線は <samples_2> 個のサンプル数を持ちます。<samples_1> のみ指定すると、<samples_2> の値は <samples_1> と同じ値に設定されます。**set isosamples** の項も参照してください。

34.46 Size

コマンド **set size** は描画出力の大きさを拡大縮小します。

書式:

```
set size {{no}square | ratio <r> | noratio} {<xscale>,<yscale>}
show size
```

<xscale> と <yscale> は描画全体の拡大の倍率で、描画全体とはグラフと余白の部分を含みます。

ratio は、指定した <xscale>, <yscale> の描画範囲内で、グラフのアスペクト比 (縦横比) を <r> にします (<r> は x 方向の長さに対する y 方向の長さの比)。

<r> の値を負にするとその意味は違って来ます。<r>=-1 の場合、x 軸と y 軸の 1 が同じ長さになるようにグラフを縮小します (例えば地学的なデータでは有用でしょう)。<r>=-2 ならば y の 1 は x の 1 の 2 倍の長さになるようにスケール変換され、以下同様です。

gnuplot が指定されたアスペクト比のグラフをちゃんと書けるかは選択される出力形式に依存します。グラフの領域は出力の指定された部分にちゃんと収まり、アスペクト比が <r> であるような最大の長方形となります (もちろん適当な余白も残しますが)。

square は **ratio 1** と同じ意味です。

noratio と **nosquare** はいずれもグラフをその出力形式 (terminal) でのデフォルトのアスペクト比に戻しますが、<xscale> と <yscale> はそのデフォルトの値 (1.0) には戻しません。

ratio と **square** は 3 次元描画では意味を持ちません。

set size はデフォルトサイズに対する相対的な指定で、デフォルトサイズは出力形式毎に異なります。**gnuplot** はデフォルトで可能な限り使用可能な描画領域内一杯を使おうとしますから、**set size** は拡大する、という使い方でなくて縮小する、という使い方の方が安全です。

出力形式によっては、描画サイズの変更は、テキストの出力位置をおかしくさせる可能性があります。

例:

通常の大きさに設定します:

```
set size 1,1
```

グラフを通常の半分の大きさで正方形にします:

```
set size square 0.5,0.5
```

グラフの高さを横幅の 2 倍にします:

```
set size ratio 2
```

34.47 Style

デフォルトの描画スタイルは、**set function style** と **set data style** で設定できます。関数やデータのデフォルトの描画スタイルを個々に変更する方法については **plot with** を参照してください。

書式:

```
set function style <style>
```

```
set data style <style>
show function style
show data style
```

全ての線、または点の描画スタイルで使用される描画タイプ (すなわち、線では実線、点線、色など、点では円、四角、十字など) は、**plot** や **splot** コマンドで指定したものが使われるか、あるいはその出力形式 (terminal) で利用可能な描画タイプが順番に使われていきます。どんなものが利用可能かを知りたければ **test** コマンドを使ってください。

2 列より多くの情報を必要とするスタイル (例えば **errorbars**) はいずれも'splot'、あるいは関数の **plot** では使うことが出来ません。**boxes** とすべての **steps** スタイルは **splot** では使えません。不適当な描画スタイルが指定された場合、それは **points** に変更されます。

2 列より多くの情報を持つ 2 次元データに関しては **gnuplot** は使用可能な **errorbars** スタイルを適当に選択します。**plot** コマンドの **using** オプションは、描画するスタイルに適切な列を指示するのに使えます。(ここでは"列" (column) という言葉は、データファイル中の列と、**using** のパラメータリスト中のエントリの両方を指すのに使用します。)

3 列のデータに対しては **xerrorbars**, **yerrorbars** (または **errorbars**), **boxes**, **boxerrorbars** のみが使用可能です。他の描画スタイルが使われている場合はそれは **yerrorbars** に変更されます。**boxerrorbars** スタイルは箱の横幅を自動的に計算します。

4 列のデータに対しては **xerrorbars**, **yerrorbars** (または **errorbars**), **xyerrorbars**, **boxxyerrorbars**, **boxerrorbars** のみが使用可能です。他の描画スタイルは **yerrorbars** に変更されます。

5 列のデータに対しては **boxerrorbars**, **financebars**, **candlesticks** のみが使用可能です (最後の 2 つは主に金融相場の描画用のスタイルです)。他の描画スタイルは描画前に **boxerrorbars** に変更されます。

6 列、7 列のデータに対しては **xyerrorbars** と **boxxyerrorbars** のみが使用可能です。他の描画スタイルは描画前に **xyerrorbars** に変更されます。

誤差表示 (error bar) に関するより詳細な情報に関しては **plot errorbars** の項を参照してください。

34.47.1 Boxerrorbars

描画スタイル **boxerrorbars** は 2 次元のデータ描画でのみ利用可能です。これは **boxes** と **yerrorbars** スタイルの組合せです。y の誤差が"ydelta" の形式で与えられて、箱の横幅があらかじめ -2.0 に設定されて (**set boxwidth -2.0**) いなければ、箱の横幅は 4 列目の値で与えられます。y の誤差が "ylow yhigh" の形式で与えられる場合は箱の横幅は 5 列目の値で与えられます。特別な場合として、"ylow yhigh" の誤差形式の 4 列のデータに対する **boxwidth = -2.0** という設定があります。この場合箱の横幅は、隣接する箱にくっつくように自動的に計算されます。3 列のデータの場合も、横幅は自動的に計算されます。

箱の高さは、**yerrorbars** スタイル同様に y の誤差の値から決定されます – y-ydelta から y+ydelta まで、あるいは ylow から yhigh まで、これらは何列のデータが与えられているかによって決まります。

34.47.2 Boxes

boxes スタイルは 2 次元描画でのみ利用可能です。これは与えられた x 座標を中心とし、x 軸から (グラフの境界から、ではありません) 与えられた y 座標までの箱を書きます。箱の幅は 3 つのうち一つの方法

で決定されます。それがデータの描画で、かつファイルが 3 列目のデータを持っている場合はそれが箱の幅にセットされます。そうでなくて **set boxwidth** コマンドで箱の幅がセットされていた場合それが使われます。そのどちらでもない場合、箱の幅は、隣接する箱がくっつくように自動的に計算されます。

34.47.3 Boxxyerrorbars

boxxyerrorbars スタイルは 2 次元のデータ描画でのみ利用可能です。これは **boxes** と **xyerrorbars** スタイルの組合せです。

箱の幅と高さは **xyerrorbars** スタイル同様 x, y の誤差から決定されます – xlow から xhigh までと ylow から yhigh まで、または x-xdelta から x+xdelta までと y-ydelta から y+ydelta まで。これらは何列のデータが与えられているかによって決まります。

34.47.4 Candlesticks

candlesticks スタイルは、金融データの 2 次元のデータ描画でのみ利用可能です。5 列のデータが必要で、順に x 座標 (多分日付)、開始値、最安値、最高値、終値、となります。記号は、水平方向には x を中心とし、垂直方向には開始値と終値を境界とする長方形が使われます。そして、その x 座標のところに、長方形のてっぺんから最高値までと、長方形の底から最安値までの垂直線が引かれます (訳注: gnuplot-3.7 では実際には、開始値から最高値までと終値から最安値までのよう)。長方形の幅は **set bar** で変更できます。記号は最安値と最高値が入れ換わっても変わりませんし、開始値と終値が入れ換わっても変わりません (訳注: gnuplot-3.7 では実際には開始値が終値よりも大きい場合長方形内は 3 本の線で塗りつぶされるよう)。**set bar** と **financebars** を参照してください。

34.47.5 Dots

dots スタイルは各点に小さなドットを描画します。これはたくさんの点からなる散布図の描画に便利でしょう。

34.47.6 Financebars

financebars スタイルは金融データの 2 次元のデータ描画でのみ利用可能です。5 列のデータが必要で、順に x 座標 (多分日付)、開始値、最安値、最高値、終値、となります。記号は、水平方向にはその x 座標に置かれ、垂直方向には最高値と最安値を端とする線分が使われます。そして、その線分に水平左側の刻みが開始値の所に、水平右側の刻みが終り値の所につきます。その刻みの長さは **set bar** で変更できます。記号は最高値と最安値が入れ替わっても変わりません。**set bar** と **candlesticks** を参照してください。

34.47.7 Fsteps

fsteps スタイルは 2 次元描画でのみ利用可能です。これは 2 本の線分で隣り合う点をつなぎます: 1 本目は (x1,y1) から (x1,y2) まで、2 本目は (x1,y2) から (x2,y2) まで。

34.47.8 Histeps

histeps スタイルは 2 次元描画でのみ利用可能です。これはヒストグラムの描画での利用を意図しています。y の値は、x の値を中心に置くと考え、x1 での点は ((x0+x1)/2,y1) から ((x1+x2)/2,y1) までの水平線として表現されます。端の点では、その線はその x 座標が中心になるように延長されます。隣り合う点同士の水平線の端は、その両者の平均値のところでの鉛直線、すなわち ((x1+x2)/2,y1) から ((x1+x2)/2,y2) の線分で結ばれます。

autoscale が有効である場合、x の範囲は、その延長された水平線の範囲ではなく、データ点の範囲が選択されます。よって、端の点に関してはその水平線は半分しか描かれないことになります。

histeps は単なる描画スタイルにすぎず、**gnuplot** には、ヒストグラムの箱を生成する能力や、データ集合から母集団を決定する能力などはありません。

34.47.9 Impulses

impulses スタイルは、x 軸 (グラフの境界ではなく) から、**splot** では格子の土台からの垂直な線分を各点に対して表示します。

34.47.10 Lines

lines スタイルは隣接する点を真直な線分で結びます。

34.47.11 Linespoints

linespoints スタイルは **lines** と **points** の両方を行ないます。すなわち、各点に小さな記号をつけ、そして隣接する点を真直な線分で結びます。コマンド **set pointsize** を使って点 (point) の記号の大きさを変更できます。その使い方については **set pointsize** を参照してください。

linespoints は **lp** と略すことが出来ます。

34.47.12 Points

points スタイルは各点に小さな記号を表示します。その記号の大きさを変更するにはコマンド **set pointsize** が使えます。その使用法については **set pointsize** の項を参照してください。

34.47.13 Steps

steps スタイルは 2 次元描画でのみ利用可能です。これは 2 本の線分で隣り合う点をつなぎます: 1 本目は (x1,y1) から (x2,y1) まで、2 本目は (x2,y1) から (x2,y2) まで。

34.47.14 Vector

vector スタイルは (x,y) から (x+xdelta,y+ydelta) までのベクトルを書きます。よって 4 列のデータが必要です。ベクトルの先端には小さな矢先も書きます。

set clip one と **set clip two** はベクトルの描画に影響を与えます。詳細は **set clip** を参照してください。

34.47.15 Xerrorbars

xerrorbars スタイルは 2 次元のデータ描画のみで利用可能です。**xerrorbars** は、水平の誤差指示線 (error bar) が表示される以外は **dots** と同じです。各点 (x,y) において (xlow,y) から (xhigh,y) まで、または (x-xdelta,y) から (x+xdelta,y) までの線分が引かれますが、これらはいくつのデータ列が与えられるかによって変わります。誤差指示線の端には刻みの印が付けられます (**set bar** が使われていなければ。詳細に関しては **set bar** を参照してください)。

34.47.16 Xyerrorbars

xyerrorbars スタイルは 2 次元のデータ描画のみで利用可能です。**xyerrorbars** は、水平、垂直の誤差指示線 (error bar) も表示される以外は **dots** と同じです。各点 (x,y) において (x,y-ydelta) から (x,y+ydelta) までと (x-xdelta,y) から (x+xdelta,y) まで、または (x,ylow) から (x,yhigh) までと (xlow,y) から (xhigh,y) までの線分が引かれますが、これらはいくつのデータ列が与えられるかによって変わります。誤差指示線の端には刻みの印が付けられます (**set bar** が使われていなければ。詳細に関しては **set bar** を参照してください)。

データが、サポートされていない混合型の形式で与えられた場合、**plot** コマンドの **using** フィルタを使って適切な形に直さないといけません。例えばデータが (x,y,xdelta,ylow,yhigh) という形式である場合、以下のようにします:

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

34.47.17 Yerrorbars

yerrorbars (または **errorbars**) スタイルは 2 次元のデータ描画のみで利用可能です。**yerrorbars** は、垂直の誤差指示線 (error bar) が表示される以外は **dots** に似ています。各点 (x,y) において (x,y-ydelta) から (x,y+ydelta) まで、または (x,ylow) から (x,yhigh) までの線分が引かれますが、これらはいくつのデータ列が与えられるかによって変わります。誤差指示線の端には刻みの印が付けられます (**set bar** が使われていなければ。詳細に関しては **set bar** を参照してください)。

34.48 Surface

コマンド **set surface** は **splot** による曲面の表示を制御します。

書式:

```
set surface
```

```
set nosurface
show surface
```

曲面はデータや関数に対して、**with** で指定されたスタイル、あるいは他の適切なスタイルで書かれます。

set nosurface が実行されれば **splot** は関数やデータファイルの点に対する点や線を書きません。その場合でも **set contour** の設定により曲面の等高線は書かれます。**set nosurface; set contour base** は等高線を格子の土台に表示する際に便利です。**set contour** も参照してください。

34.49 Terminal

gnuplot は数多くのグラフィック形式をサポートしています。コマンド **set terminal** を使って **gnuplot** の出力の対象となる形式の種類を選んでください。出力先をファイル、または出力装置にリダイレクトするには **set output** を使ってください。

書式:

```
set terminal {<terminal-type>}
show terminal
```

<terminal-type> が省略されると **gnuplot** は利用可能な出力形式の一覧を表示します。<terminal-type> の指定には短縮形が使えます。

set terminal と **set output** の両方を使う場合、**set terminal** を最初にする方が安全です。それは、OS によっては、それが必要とするフラグをセットする出力形式があるからです。

いくつかの出力形式は追加オプションを持ちます。例えば **dump**, **iris4d**, **hpljii**, **postscript** などの項を参照してください。

この文書は、その出力書式に応じて、インストールされなかったために利用できない出力形式のドライバについては記述されておらず、利用できる全てのドライバについて記述されています。

34.49.1 Aed767

出力ドライバ **aed512** と **aed767** は AED グラフィック端末をサポートします。この 2 つのドライバは、水平方向の範囲のみが違っていて、それぞれ 512 ピクセルと 767 ピクセルです。垂直方向はいずれも 575 ピクセルです。これらのドライバにはオプションはありません。

34.49.2 Aifm

Adobe Illustrator 3.0+ ドライバ **aifm** には、いくつかのオプションを設定できます。

書式:

```
set terminal aifm {<color>} {"<fontname>"} {<fontsize>}
```

<color> は **color** か **monochrome**、"<fontname>" は正式な PostScript フォント名、<fontsize> はフォントサイズを PostScript のポイント数で与えます。これらは **set size** コマンドでスケール変換される前のものです。**default** を選択すると、全てのオプションがデフォルトの値: **monochrome**, "Helvetica", 14pt, に設定されます。

AI は複数のページをサポートしていないので、複数のグラフは直接他のグラフの上に書かれます。しかし、各グラフは個別にグループ化されていて、AI 上で簡単にそれらを分離することが出来ます (単にそれらを拾い上げて移動させるだけです)。

例:

```
set term aifm
set term aifm 22
set size 0.7,1.4; set term aifm color "Times-Roman" 14
```

34.49.3 Amiga

Commodore (コモドール) Amiga コンピュータ用の **amiga** 出力ドライバは、スクリーン (default)、または、Kickstart 3.0 またはそれ以上のものがインストールされていればカレントパブリックスクリーン上のウィンドウのどちらでも描画できます。フォントとそのサイズも選択できます。

書式:

```
set terminal amiga {screen | window} {"<fontname>"} {<fontsize>}
```

デフォルトのフォントは 8 ポイントの "topaz" です。

screen オプションは、仮想スクリーンを使うので、スクリーンより大きいグラフを作ることも可能です。

34.49.4 Apollo

apollo 出力ドライバは、Apollo グラフィックプリミティブリソースをサポートしウィンドウのリサイズ後のリスケーリングもサポートしています。オプションはありません。

固定サイズのウィンドウが必要ならば、**gpr** 出力を代わりに使用してください。

34.49.5 Atari ST (via AES)

atari 出力形式には、画面の色と文字のサイズを設定するオプションがあります。

書式:

```
set terminal atari {<fontsize>} {<col0> <col1> ... <col15>}
```

色を指定する場合、必ず文字サイズを指定しなければいけません。個々の (16 色までの) 色は、3 桁の 16 進数で与え、各桁は赤、緑、青 (RGB) の強度を (この順で) あらわします。0–15 の範囲は、実際にスクリーンが持つカラーの範囲に調整されます。例えば通常の ST スクリーンでは、奇数と偶数の強度は同じになります。

例:

```
set terminal atari 4    # 小さい (6x6) フォント
set terminal atari 6 0  # 黒背景に白の白黒画面
set terminal atari 13 0 fff f00 f0 f ff f0f ff0
           # 最初の 8 色を黒、白、赤、緑、青、シアン、紫、黄とし
           # 大きいフォント (8x16) を使用
```

また、環境変数 GNUCOLORS が設定されていれば、それもオプション文字列として解釈されますが、明示的なターミナルオプションの方が優先されます。

34.49.6 Atari ST (via VDI)

vdi 出力ドライバは、AES-Windows ではなく VDI 経由でスクリーンに出力することを除けば **atari** 出力ドライバと全く同じです。

vdi 出力形式には、画面の色と文字のサイズを設定するオプションがあります。

書式:

```
set terminal vdi {<fontsize>} {<col0> <col1> ... <col15>}
```

色を指定する場合、必ず文字サイズを指定しなければいけません。個々の (16 色までの) 色は、3 桁の 16 進数で与え、各桁は赤、緑、青 (RGB) の強度を (この順で) あらわします。0–15 の範囲は、実際にスクリーンが持つカラーの範囲に調整されます。例えば通常の ST スクリーンでは、奇数と偶数の強度は同じになります。

例:

```
set terminal vdi 4    # 小さい (6x6) フォント
set terminal vdi 6 0  # 黒背景に白の白黒画面
set terminal vdi 13 0 fff f00 f0 f ff f0f ff0
           # 最初の 8 色を黒、白、赤、緑、青、シアン、紫、黄とし
           # 大きいフォント (8x16) を使用
```

また、環境変数 GNUCOLORS が設定されていれば、それもオプション文字列として解釈されますが、明示的なターミナルオプションの方が優先されます。

34.49.7 Be

gnuplot は X サーバを利用する **be** 出力ドライバを提供します。この出力形式は、環境変数 **DISPLAY** がセットされているか、環境変数 **TERM** が **xterm** にセットされているか、またはコマンドラインオプションとして**-display** が使われていれば起動時に自動的に選択されます。

書式:

```
set terminal be {reset} {<n>}
```

複数のグラフ描画ウィンドウをサポートしています。**set terminal be <n>** は番号 n のウィンドウに出力します。n>0 の場合、その番号はウィンドウタイトルとアイコン名に **gplt <n>** として付けられます。現在のウィンドウはカーソル記号の変化で区別できます (デフォルトカーソルから十字カーソルへ)。

gnuplot ドライバが別な出力ドライバに変更されても、描画ウィンドウは開いたままになります。描画ウィンドウは、そのウィンドウにカーソルを置いて文字 q を押すか、ウィンドウマネージャのメニューの **close** を選択すれば閉じることができます。**reset** を実行すれば全てのウィンドウを一度に閉じれます。それは実際にウィンドウを管理している子プロセスを終了します (もし **-persist** が指定されていなければ)。

描画ウィンドウは **-persisit** オプションが与えられていなければ、対話の終了時に自動的に閉じられます。

描画サイズとアスペクト比は、**gnuplot** のウィンドウをリサイズすることでも変更できます。

線の幅と点のサイズは **gnuplot** の **set linestyle** で変更可能です。

出力ドライバ **be** に関しては、**gnuplot** は (起動時に)、コマンドライン、または設定ファイルから、geometry や font, name などの通常の X Toolkit オプションやリソースの指定を受け付けます。それらのオプションについては X(1) マニュアルページ (やそれと同等のもの) を参照してください。

他にも **be** 出力形式用の多くの **gnuplot** のオプションがあります。これらは **gnuplot** を呼ぶときにコマンドラインオプションとして指定するか、または設定ファイル "~ /.Xdefaults" のリソースとして指定できます。これらは起動時に設定されるので、**gnuplot** 実行時には変更できません。

34.49.7.1 Command-line_options X Toolkit オプションに加え、以下のオプションが gnuplot の立ち上げ時のコマンドラインで、またはユーザのファイル ".Xdefaults" 内のリソースとして指定できます:

'-mono'	カラーディスプレイ上で強制的に白黒描画
'-gray'	グレイスケールまたはカラーディスプレイ上でのグレイスケール描画 (デフォルトではグレイスケールディスプレイは白黒描画を受け付ける)
'-clear'	新しい描画を表示する前に (瞬間的に) 画面を消去
'-raise'	各描画後に描画ウィンドウを最前面へ出す
'-noraise'	各描画後に描画ウィンドウを最前面へ出すことはしない
'-persist'	gnuplot プログラム終了後も描画ウィンドウを残す

上記のオプション、はコマンドライン上での指定書式で、".Xdefaults" にリソースとして指定するときは異なる書式を使います。

例:

```
gnuplot*gray: on
```

gnuplot は描画スタイル **points** で描画する点のサイズの制御にも、コマンドラインオプション (**-pointsize <v>**) とリソース (**gnuplot*pointsize: <v>**) を提供しています。値 **v** は点のサイズの拡大率として使われる実数値 ($0 < v <= 10$) で、例えば **-pointsize 2** はデフォルトのサイズの 2 倍、**-pointsize 0.5** は普通のサイズの半分の点が使われます。

34.49.7.2 Monochrome_options 白黒ディスプレイに対しては **gnuplot** は描画色 (foreground) も背景色 (background) も与えません。デフォルトでは背景は白、描画は黒です。**-rv** や **gnuplot*reverseVideo: on** の場合には背景が黒で描画は白になります。

34.49.7.3 Color_resources カラーディスプレイに対しては、**gnuplot** は以下のリソース (ここではそのデフォルトの値を示します)、または白黒階調 (greyscale) のリソースを参照します。リソースの値はシステム上の BE rgb.txt ファイルに書かれている色名、または 16 進の色指定 (BE のマニュアルを参照) か、色名と強度 (0 から 1 の間の値) をコンマで区切った値を使用できます。例えば **blue, 0.5** は半分の強度の青、を意味します。

```
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
```

これらに関するコマンドラインの書式は、例えば以下の通りです。

例:

```
gnuplot -background coral
```

34.49.7.4 Grayscale_resources **-gray** を選択すると、**gnuplot** は、グレイスケールまたはカラーディスプレイに対して、以下のリソースを参照します (ここではそのデフォルトの値を示します)。デフォルトの背景色は黒であることに注意してください。

```
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
```

34.49.7.5 Line_resources **gnuplot** は描画の線の幅 (ピクセル単位) の設定のために以下のリソースを参照します (ここではそのデフォルトの値を示します)。0 または 1 は最小の線幅の 1 ピクセル幅を意味します。2 または 3 の値によってグラフの外観を改善できる場合もあるでしょう。

```
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 は線描画で使用する点線の形式の設定用に以下のリソースを参照します。0 は実線を意味します。2 桁の 10 進数 **jk** (**j** と **k** は 1 から 9 までの値) は、**j** 個のピクセルの描画に **k** 個の空白のピクセルが続くパターンの繰り返しからなる点線を意味します。例えば '16' は 1 個のピクセルの後に 6 つの空白が続くパターンの点線になります。さらに、4 桁の 10 進数でより詳細なピクセルと空白の列のパターンを指定できます。例えば、'4441' は 4 つのピクセル、4 つの空白、4 つのピクセル、1 つの空白のパターンを意味します。以下のデフォルトのリソース値は、白黒ディスプレイ、あるいはカラーや白黒階調 (grayscale) ディスプレイ上の白黒描画における値です。カラーディスプレイではそれらのデフォルトの値はほとんど 0 (実線) で、**axisDashes** のみがデフォルトで '16' の点線となっています。

```
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
```

34.49.8 Cgi

cgi と **hcgi** 出力ドライバは SCO CGI ドライバをサポートします。**hcgi** はプリンタ用で、環境変数 CGIPRNT が設定されている必要があります。**cgi** はディスプレイ、またはハードコピーに使え、環境変数 CGIDISP が設定されていればディスプレイを使用し、そうでなければ CGIPRNT を使用します。

これらの出力ドライバにはオプションはありません。

34.49.9 Cgm

cgm ドライバは CGM 出力 (Computer Graphics Metafile Version 1) を生成します。このファイルフォーマットは ANSI 規格書 X3.122-1986 "Computer Graphics - Metafile for the Storage and Transfer of

Picture Description Information" で定義されているものの一部分です。**cgm** には多くのオプションがあります。

書式:

```
set terminal cgm {<mode>} {<color>} {<rotation>} {solid | dashed}
                 {width <plot_width>} {linewidth <line_width>}
                 {"<font>"} {<fontsize>}
                 {<color0> <color1> <color2> ...}
```

ここで、<mode> は **landscape**, **portrait**, **default** のいずれか; <color> は **color**, **monochrome** のいずれか; <rotation> は **rotate**, **norotate** のいずれか; **solid** は全ての曲線を実線で描き、どんな点線パターンも塗りつぶします; <plot_width> はポイント単位でのグラフの仮定されている幅; <line_width> はポイント単位での線幅 (デフォルトは 1); <font> はフォントの名前; そして <fontsize> はポイント単位でのフォントのサイズ (デフォルトは 12) です。

デフォルトでは **cgm** は Y 軸の見出しに 90 度回転した文字を使用します。

最初の 6 つのオプションはどの順番で指定しても構いません。**default** を選択すると、全てのオプションをそのデフォルトの値にします。

各カラー (color) は 'xrrggbb' の形式です。ここで、x は文字 'x' そのもの、'rrggbb' は 16 進数での赤、緑、青の成分です。例えば 'x00ff00' は緑になります。背景色が最初に設定され、以下は描画色となります。

例:

```
set terminal cgm landscape color rotate dashed width 432 \
               linewidth 1  'Helvetica Bold' 12       # デフォルト
set terminal cgm linewidth 2  14  # やや広い線とやや大きいフォント
set terminal cgm portrait "Times Italic" 12
set terminal cgm color solid    # 面倒な点線など消えてしまえ !
```

34.49.9.1 Font CGM (Computer Graphics Metafile) ファイルの最初の部分、メタファイルの記述部分には、フォントリスト (font table) が含まれています。画像の本体部では、フォントはこのリストにある番号で指定されます。デフォルトではこのドライバは以下の 35 個のフォントリストを生成し、さらにこのリストの **Helvetica**, **Times**, **Courier** の各フォントの **italic** を **oblique** で置き換えたもの、およびその逆による 6 つの追加のフォントが含まれます (Microsoft Office と Corel Draw CGM の import フィルタは **italic** と **oblique** を同じものとして扱うからです)。

CGM fonts
Helvetica
Helvetica Bold
Helvetica Oblique
Helvetica Bold Oblique
Times Roman
Times Bold
Times Italic
Times Bold Italic
Courier
Courier Bold
Courier Oblique
Courier Bold Oblique
Symbol
Hershey/Cartographic_Roman
Hershey/Cartographic_Greek
Hershey/Simplex_Roman
Hershey/Simplex_Greek
Hershey/Simplex_Script
Hershey/Complex_Roman
Hershey/Complex_Greek
Hershey/Complex_Script
Hershey/Complex_Italic
Hershey/Complex_Cyrillic
Hershey/Duplex_Roman
Hershey/Triplex_Roman
Hershey/Triplex_Italic
Hershey/Gothic_German
Hershey/Gothic_English
Hershey/Gothic_Italian
Hershey/Symbol_Set_1
Hershey/Symbol_Set_2
Hershey/Symbol_Math
ZapfDingbats
Script
15

これらのフォントの最初の 13 個は WebCGM で要求されているものです。Microsoft Office の CGM import フィルタはその 13 個の標準フォントと'ZapfDingbats' と 'Script' をサポートしています。しかし、そのスクリプト (script) フォントは '15' という名前でしかアクセスできません。Microsoft の import フィルタの font の置き換えに関するより詳しい情報については、

```
C:\Program Files\Microsoft Office\Office\Cgmimp32.hlp
```

のヘルプファイル、または

```
C:\Program Files\Common Files\Microsoft Shared\Grphflt\Cgmimp32.cfg
```

の設定ファイルなどをチェックしてください。

set term コマンドでデフォルトのフォントリストにないフォント名を指定することも可能です。その場合、その指定したフォントが最初に現われる新しいフォントリストが作られます。そのフォント名に関して、スペル、単語の先頭の大文字化やどこにスペースが入るかなどが、作られる CGM ファイルを読むアプリケーションにとって適切なものであるかをちゃんと確認する必要があります。(gnuplot と任意の MIL-D-28003A 準拠アプリケーションは、フォント名の大文字小文字の違いは無視します。) 新しいフォントをいくつも追加したい場合は、**set term** コマンドを繰り返し使用してください。

例:

```
set terminal cgm 'Old English'
set terminal cgm 'Tengwar'
set terminal cgm 'Arabic'
set output 'myfile.cgm'
plot ...
set output
```

set label コマンドでは新しいフォントを導入することはできません。

34.49.9.2 Fontsize フォントは、ページが 6 インチの幅であると仮定して縮尺されます。**size** コマンドでページの縦横比が変更されていたり、CGM ファイルが異なる幅に変換されている場合、結果としてフォントのサイズも拡大されたり縮小されたりすることになります。仮定されている幅を変更するには、**width** オプションを使用してください。

34.49.9.3 Linewidth **linewidth** オプションは線の幅をポイント単位 (pt) で設定します。デフォルトの幅は 1 pt です。**fontsize** や **width** オプションのところで説明されているように、ページの実際の幅によってその縮尺は影響を受けます。

34.49.9.4 Rotate **norotate** オプションはテキストの回転をしないようにします。例えば Word for Windows 6.0c 用の CGM 入力フィルタは回転された文字列を受け付けますが、Word に付属する DRAW エディタはそれを受け付けることができず、グラフを編集すると (例えば曲線に見出しをつける)、全ての回転された文字列は水平方向になって保存されてしまい、Y 軸の見出しはクリップされる境界線を越えてしまうでしょう。**norotate** オプションを使えば、見栄えの良くない場所から Y 軸の見出しが始まってしまいますが、編集によってダメージを受けることはなくなります。**rotate** オプションはデフォルトの挙動を保証します。

34.49.9.5 Solid **solid** オプションは描画の点線の線描画スタイルを無効するのに使います。これは、カラーが有効である場合、また点線にすることでグラフが見にくくなる場合に有用でしょう。**dashed** オプションはデフォルトの挙動を保証し、この場合個々のグラフに異なる点線のパターンが与えられます。

34.49.9.6 Size CGM グラフのデフォルトのサイズは、横置き (landscape) では幅 32599, 縦 23457、縦置き (portrait) では幅 23457, 縦 32599 です。

34.49.9.7 Width CGM ファイルの全ての長さは抽象的な単位を持ち、そのファイルを読むアプリケーションが最終的なグラフのサイズを決定します。デフォルトでは最終的なグラフの幅は 6 インチ (15.24 cm) であると仮定されています。この幅は正しいフォントサイズを計算するのに使われ、**width** オプションで変更できます。キーワード **width** の後に幅をポイント単位で指定します。(ここで、ポイントは PostScript と同様 1/72 インチを意味します。この単位は TeX では"big point" と呼ばれています。) 他の単位から変換するには、gnuplot の数式が使えます。

例:

```
set terminal cgm width 432            # デフォルト
set terminal cgm width 6*72           # 上と同じ値
set terminal cgm width 10/2.54*72     # 10 cm の幅
```

34.49.9.8 Nofontlist デフォルトのフォントリスト (font table) は WebCGM で勧告されているフォントを含んでいて、これは Microsoft Office と Corel Draw の CGM (Computer Graphics Metafile) 入力フィルタに適合しています。他のアプリケーションは異なるフォント、あるいは異なるフォント名を使用するかも知れませんが、それはマニュアルには書かれていないかも知れません。そのための回避策として、オプション **nofontlist** を使用して CGM ファイルからフォントリストを削除するという方法があります。この場合、読み込んだアプリケーションはデフォルトのフォントリストを使用するでしょう。gnuplot はその場合でもフォント番号の選択のために自分のデフォルトのフォントリストを使用します。よって、'Helvetica' が 1 番になり、それがあなたの使用するアプリケーションのデフォルトフォントリストの最初のものになります。'Helvetica Bold' がそのフォントリストの 2 番目のフォントに対応し、他も同様となります。

以前の **winword6** オプションは今は **nofontlist** の同意語に成り下がりました。**winword6** オプションが回避策として使われていた、色やフォントリストに関する問題は、結局 gnuplot のバグであるとわかり、それは今では修正されています。

34.49.10 Corel

corel 出力形式は CorelDraw 用の出力です。

書式:

```
set terminal corel {  default
                    | {monochrome | color
                        {<fontname> {"<fontsize>"
                           {<xsize> <ysize> {<linewidth> }}}}}
```

ここで、フォントサイズ (**fontsize**) と線幅 (**linewidth**) はポイント単位、横幅 (**xsize**) と縦幅 (**ysize**) はインチ単位です。デフォルトの値はそれぞれ、monochrome, "SwitzerlandLight", 22, 8.2, 10, 1.2 です。

34.49.11 Debug

このドライバは **gnuplot** のデバッグのために提供されているものです。おそらくソースコードを修正するユーザのみが使用するものでしょう。

34.49.12 Svga

svga ドライバは SVGA グラフィックの PC をサポートします。これは DJGPP でコンパイルされた場合のみ使うことができます。オプションはフォントに関するもののみです。

書式:

```
set terminal svga {"<fontname>"}
```

34.49.13 Dumb

ダム端末 (**dumb**) ドライバにはサイズの指定と改行制御用のオプションがあります。

書式:

```
set terminal dumb {[no]feed} {<xsize> <ysize>}
```

<xsize>, <ysize> はダム端末 のサイズを設定し、デフォルトはそれぞれ 79, 24 となっています。最後の改行は、**feed** オプションが設定されている場合のみ出力されます。

例:

```
set term dumb nofeed
set term dumb 79 49 # VGA screen---何故こうしたい人がいるんだろう ?
```

34.49.14 Dxf

dxf ドライバは、AutoCad (リリース 10.x) に取り込むことができる画像を生成します。このドライバ自身にはオプションはありませんが、描画に関するいくつかの特徴は他の方法で変更できます。デフォルトの大きさは AutoCad の単位での 120x80 で、これは **set size** で変更できます。**dxf** は 7 色 (白、赤、黄、緑、水色、青、赤紫) を使いますが、これを変更するにはドライバソースファイルを修正する必要があります。白黒の出力装置を使う場合、それらの色は線の太さの違いで表現されます。詳細は AutoCad の印刷/プロッタ出力コマンドに関する記述を参照してください。

34.49.15 Dxy800a

このドライバは Roland DXY800A プロッタをサポートします。オプションはありません。

34.49.16 Eepic

eepic ドライバは LaTeX picture 環境を拡張するものをサポートします。これは **latex** ドライバに代わる別な選択肢です。

このドライバによる出力は、LaTeX 用の "eepic.sty" マクロパッケージと共に使われることを仮定しています。それを使うには、"eepic.sty" と"epic.sty"、および "tpic" \special 命令群をサポートするプリンタドライバが必要です。もし、あなたの使うプリンタドライバがそれらの "tpic" \special 命令をサポートし

ていない場合でも、"eepicemu.sty" を使うことでそれらのうちのいくつかを使えるようになります。dvips と dvipdfm は "tpic" \special をサポートしています。

書式:

```
set terminal eepic {color, dashed, rotate, small, tiny, default, <fontsize>}
```

オプション: オプションは任意の順番で与えることができます。'color' は gnuplot に \color{...} コマンドを生成させ、それによりグラフをカラーにします。このオプションを使用する場合は、latex 文書のプリアンブルに \usepackage{color} を入れる必要があります。'dashed' は線種に点線を使用することを許可します。このオプションを指定しないと、色々な太さの実線のみが使われます。'dashed' と 'color' は一方のみが意味を持ち、'color' が指定された場合、'dashed' は無視されます。'rotate' は本当に回転 (90 度) された文字列を使用するようになります。指定しない場合は、1 文字 1 文字、上に積み上げていく方法で回転された文字列を作ります。このオプションを使う場合は、\usepackage{graphicx} をプリアンブルに入れる必要があります。'small' は point スタイルでのグラフ描画の印として \scriptsize の記号を使用します (多分これは TeX ではだめで、LaTeX2e でしか使えないでしょう)。デフォルトでは標準の数式のサイズを使用します。'tiny' は、それに \scriptscriptstyle の記号を使用します。'default' は全てのオプションをデフォルトの値にリセットします。デフォルトは、color はなし、dashed line はなし、疑似回転 (積み上げ) 文字列の使用、大きなサイズの記号の使用、です。<fontsize> は picture 環境内でのフォントサイズを指定する数字です。単位は pt (ポイント) で、10 pt はほぼ 3.5 mm です。フォントサイズを指定しない場合、全てのグラフ内の文字は \footnotesize に設定されます。

注意: 文字 # (およびその他 (La)TeX で特別な意味を持つその他の文字) を \\ (バックスラッシュ 2 つ) でエスケープすることを忘れないでください。グラフの角が近すぎると点線は実線のようになります。(これが tpic specials の一般的な問題なのか、eepic.sty や dvips/dvipdfm のバグが原因なのかは私にはわかりません。) デフォルトの eepic グラフの大きさは 5x3 インチで、これは 'set size a,b' で縮尺可能です。数ある中で、点 (point) は、LaTeX のコマンド "\Diamond", "\Box" などを使って描かれます。これらのコマンドは現在は LaTeX2e のコアには存在せず latexsym パッケージに含まれていますが、このパッケージ基本配布の一部であり、よって多くの LaTeX のシステムの一部になっています。このパッケージを使うことを忘れないでください。latexsym の代わりに amssymb パッケージを使うことも可能です。LaTeX に関する全てのドライバは文字列の配置の制御に特別な方法を提供します: '{' で始まる文字列は、'}' で閉じる必要がありますが、その文字列全体が水平方向にも垂直方向にもセンタリングされます。'[' で始まる文字列の場合は、位置の指定をする文字列 (t,b,l,r のうち 2 つまで) が続き、次に']{'、文字列本体、で最後に '}' としますが、この文字列は LaTeX が LR-box として整形します。'\rule{}{}' を使えばさらに良い位置合わせが可能でしょう。

例: set term eepic

は、グラフを `picture` 環境に含まれた `eepic` マクロとして出力します。
そのファイルを `LaTeX` 文書に `\input` で取り込んでください。

set term eepic color tiny rotate 8

`eepic` マクロを `\color` マクロ、`point` 印は `\scripscriptsize` の大きさ、
本当に回転された文字の使用、および全ての文字を `8pt` にセットして出力
します。

見出しの位置合わせに関して: gnuplot のデフォルト (大抵それなりになるが、そうでないこともある):

```
set title '\LaTeX\ -- $ \gamma $'
```

水平方向にも垂直方向にもセンタリング:

```
set label '{\LaTeX\ -- $ \gamma $}' at 0,0
```

位置を明示的に指定 (上に合わせる):

```
set xlabel '[t]{\LaTeX\ -- $ \gamma $}'
```

他の見出し – 目盛りの長い見出しに対する見積り:

```
set ylabel '[r]{\LaTeX\ -- $ \gamma $\rule{7mm}{0pt}}'
```

34.49.17 Emf

emf ドライバは EMF (Enhanced Metafile Format) ファイルを生成します。このファイルフォーマットは MS Win32 システムの標準メタファイルです。

書式:

```
set terminal emf {<color>} {solid | dashed}
                 {"<font>"} {<fontsize>}
```

<color> は **color** (カラー) か **monochrome** (白黒)、**solid** は全ての線を実線で書き、点線のパターンは塗りつぶします。<font> はフォント名、<**fontsize**> はポイント単位でのフォントの大きさです。

最初の 2 つのオプションはどちらが先でも構いません。**default** を選択すると全てのオプションがそのデフォルトの値になります。

例:

```
set terminal emf 'Times Roman Italic' 12
set terminal emf color solid    # 面倒な点線など消えてしまえ !
```

34.49.18 Emxvga

emxvga, **emxvesa**, **vgal** の各ドライバはそれぞれ SVGA, vesa SVGA, VGA グラッフィックボードの PC をサポートします。これらは DOS、または OS/2 上で、"emx-gcc" でコンパイルされたることを意図しています。これらにはさらに VESA パッケージと SVGAKIT が必要です。これらは Johannes Martin (JMARTIN@GOOFY.ZDV.UNI-MAINZ.DE) が保守し David J. Liu (liu@phri.nyu.edu) が拡張しているライブラリです。

書式:

```
set terminal emxvga
set terminal emxvesa {vesa-mode}
set terminal vgal
```

唯一のオプションは **emxvesa** に対する vesa のモードで、デフォルトではそれは G640x480x256 となっています。

34.49.19 Epslatex

epslatex ドライバでは 2 つのオプションが設定できます。

書式:

```
set terminal epslatex {default}
                        {color | monochrome} {solid | dashed}
                        {"<fontname>"} {<fontsize>}
```

default モードは全てのオプションを以下のデフォルトの値に設定します: **monochrome**, **dashed**, "default", 11pt。
グラフのデフォルトのサイズは幅が 5 インチ、高さが 3 インチです。

solid は全てのグラフを実線で描き、点線のパターンを塗りつぶします。"<**fontname**>" はフォントの名前、<**fontsize**> は PostScript ポイント単位でのフォントの大きさです。フォントの選択はまだサポートされていません。フォントの大きさの選択は適切なスペーシングの計算に対してのみサポートされています。挿入の間際では実際には LaTeX のフォントが使用されるので、フォントの変更は LaTeX のコマンドを使用してください。例えば LaTeX の文書で 12pt のフォントを使うのなら、オプションとして '"default" 12' と指定してください。

LaTeX に関する全てのドライバは文字列の配置の制御に特別な方法を提供します: '{' で始まる文字列は、'}' で閉じる必要がありますが、その文字列全体が LaTeX によって水平方向にも垂直方向にもセンタリングされます。'[' で始まる文字列の場合は、位置の指定をする文字列 (t,b,l,r のうち 2 つまで) が続き、次に ']{'、文字列本体、で最後に '}' としますが、この文字列は LaTeX が LR-box として整形します。\rule{}{} を使えばさらに良い位置合わせが可能でしょう。pslatex ドライバに関する項目も参照してください。複数行の見出しを作成するには \shortstack を使用してください。例えば、

```
set ylabel '[r]{\shortstack{first line \\ second line}}'
```

このドライバは 2 つの別のファイルを作ります。1 つは LaTeX の部分で、もう一つは図の eps の部分です。LaTeX ファイルの名前は **set output** コマンドで与えられる eps ファイルの名前から導かれ、そのファイル名の最後の **.eps** (そのファイル名の最後の拡張子部分 – それがない場合 LaTeX ファイルは生成されません) を **.tex** で置き換えたものになります。**gnuplot** を終了する前にファイルをクローズするのを忘れないでください。出力ファイルを指定しなければ LaTeX 出力は行なわれません！LaTeX の文書で図を取り込むには '\input{filename}' としてください。\usepackage{graphics} をプリアンブルに入れてください！'epstopdf' で (例えば teTeX パッケージには含まれていて ghostscript が必要) eps 出力から pdf ファイルを作ることができます。graphics パッケージが適切に設定されているなら、LaTeX のファイルは変更なしに pdflatex によっても処理でき、eps ファイルの代わりに pdf ファイルが取り込まれます。

34.49.20 Epson-180dpi

このドライバはエプソンプリンタのいくつかとそれに類似するものをサポートします。

epson-180dpi と **epson-60dpi** はそれぞれ 180dpi (ドット/インチ), 60dpi の解像度の Epson LQ 型 24 ピンプリンタ用のドライバです。

epson-lx800 は Epson LX-800, Star の NL-10 や NX-1000, PROPRINTER などの適当なプリンタに流用できる、一般的な 9 ピンプリンタドライバです。

nec-cp6 は NEC CP6 や Epson LQ-800 などのプリンタで使える、一般的な 24 ピンプリンタ用のドライバです。

okidata ドライバは 9 ピンの OKIDATA 320/321 標準プリンタをサポートします。

starc ドライバは Star カラープリンタ用です。

tandy-60dpi ドライバは 9 ピン 60dpi の Tandy DMP-130 シリーズ用です。

nec-cp6 にのみオプションがあります。

書式:

```
set terminal nec-cp6 {monochrome | colour | draft}
```

デフォルトでは白黒 (monochrome) です。

これらのドライバのそれぞれで PC 上で印刷する場合はバイナリコピーが必要です。**print** を使ってはいけません。その代わりに **copy file /b lpt1:** としてください。

34.49.21 Excl

excl ドライバは EXCL レーザープリンターや 1590 のような Talaris プリンタをサポートします。オプションはありません。

34.49.22 Hercules

これらのドライバは PC のモニタをサポートし、グラフィックボードを自動検出します。これらは、gcc, または Zortec C/C++ でコンパイルされた場合のみ利用可能です。

34.49.23 Fig

fig ドライバは Fig グラフィック言語での出力を生成します。

書式:

```
set terminal fig {monochrome | color} {small | big}
                 {pointsmax <max_points>}
                 {landscape | portrait}
                 {metric | inches}
                 {fontsize <fsize>}
                 {size <xsize> <ysize>}
                 {thickness <units>}
                 {depth <layer>}
```

monochrome と **color** は 画像を白黒にするか **color** にするかを決定します。**small** と **big** は、デフォルトの **landscape** モードではグラフを 5x3 インチにするか 8x5 インチにするか、**portrait** モードでは 3x5 インチにするか 5x8 インチにするかを決定します。<max_points> は折れ線の最大点数を設定します。"xfig" での編集に対するデフォルトの単位を **metric** か **inches** のいずれかに選択できます。**fontsize** は

テキストフォントのサイズを <fsize> ポイントに設定します。**size** は描画範囲を <xsize>*<ysize> に設定 (変更) します。この場合の単位は、**inches** か **metric** かの設定によってそれぞれインチかセンチメートルになります。**depth** は全ての線と文字列に対する、重なりに関するデフォルトの深さ (depth layer) を設定します。デフォルトの深さは 10 で、"xfig" でグラフの上に何かを上書きするための余地を残しています。

tickness はデフォルトの線の太さを設定し、指定されなければ 1 になります。太さの変更は、**plot** コマンドの **linetype** の値に 100 倍の数を加えることで実現できます。同様に、(デフォルトの深さに対する) 各描画要素の深さの値は <linetype> に 1000 倍の数を加えることで制御できます。よってその深さは <layer>+<linetype>/1000 になり、線の太さは (<linetype>%1000) /100 となりますが、その値が 0 の場合はデフォルトの線の太さになります。

plot コマンドの **point** スタイルによる描画の際の記号を **fig** ドライバで追加することもできます。記号の指定は (**pointtype** の値) % 100 の 50 以上の値が使われ、その塗りつぶしの濃さは <pointtype> % 5 の値で制御し、その輪郭は黒 (<pointtype> % 10 < 5 の場合) または現在の色で書かれます。利用可能な記号は以下の通りです。

```
50 - 59:  円
60 - 69:  正方形
70 - 79:  ひし形
80 - 89:  上向きの三角形
90 - 99:  下向きの三角形
```

これらの記号の大きさはフォントの大きさと関係しています。デフォルトでは記号の深さは、良いエラーバーを実現するために、線の深さより 1 だけ小さい値になっています。<pointtype> が 1000 より大きい場合、深さは <layer> +<pointtype>/1000-1 になります。<pointtype>%1000 が 100 より大きい場合塗りつぶし色は (<pointtype>%1000)/100-1 になります。

有効な塗りつぶし色 (1 から 9) は、黒、青、緑、水色、赤、紫、黄、白、暗い青 (白黒モードでは 1 から 6 までは黒で 7 から 9 までは白) です。

<linetype> と <pointtype> の詳細については **plot with** を参照してください。

big オプションは以前のバージョンの **bfig** ドライバの代用品ですが、このドライバは今はもうサポートされていません。

例:

```
set terminal fig monochrome small pointsmax 1000  # デフォルト

plot 'file.dat' with points linetype 102 pointtype 759
```

は、黄色で塗りつぶされた円を生成し、それら輪郭は幅 1 の青い線です。

```
plot 'file.dat' using 1:2:3 with err linetype 1 pointtype 554
```

は黒い線によるエラーバーと赤で塗りつぶされた円を生成します。この円は線よりも 1 層だけ上になります (デフォルトでは深さは 9)。

円の上にエラーバーを書くには以下のようにしてください。

```
plot 'file.dat' using 1:2:3 with err linetype 1 pointtype 2554
```

34.49.24 Ggi

GGI ドライバは GGI (General Graphics Interface) の出力を生成します。

34.49.25 Gif

gif ドライバは GIF フォーマットの出力を生成します。これは Thomas Boutell の gd ライブラリを使用します (http://www.boutell.com/gd/ にあります)。

デフォルトでは **gif** ドライバは Web に合うパレットの割り当てを使用します。

書式:

```
set terminal gif {transparent} {interlace}
                 {tiny | small | medium | large | giant}
                 {size <x>,<y>}
                 {<color0> <color1> <color2> ...}
```

transparent はドライバに透明化 (transparent) GIF を生成するよう指示します。この場合最初の指定色が透明化される色になります。

interlace はドライバにインターレース GIF を生成するよう指示します。

フォントは **tiny** (5x8 ピクセル)、**small** (6x12 ピクセル), **medium** (7x13 Bold)、**large** (8x16)、**giant** (9x15 ピクセル) から選択します。

サイズ <x,y> はピクセル単位で与えます。デフォルトは 640x480 です。ピクセル数は **set size** コマンドで縮尺することでも変更できます。

個々の色は 'xrrggbb' の形式で指定します。ここで、x はそのまま文字 'x' を意味し、'rrggbb' は赤、緑、青の 16 進数での成分です。例えば 'x00ff00' は緑を表します。背景色を最初に指定し、その後ろに境界色、X 軸と Y 軸の色、グラフの描画色、と続きます。色数は最大 256 色まで設定できます。

例:

```
set terminal gif small size 640,480 \
                 xffffff x000000 x404040 \
                 xff0000 xffa500 x66cdaa xcdb5cd \
                 xadd8e6 x0000ff xdda0dd x9500d3    # デフォルト
```

この例では透明化されない背景色が白で、境界は黒、軸は灰色、そして描画色として赤、橙色、青緑色 (midium aquamarine)、アザミ色 (thistle 3)、明青色 (light blue)、青、かきつばた色 (plum)、暗いすみれ色 (dark violet) の 8 色を使います。

```
set terminal gif transparent xffffff \
                 x000000 x202020 x404040 x606060 \
                 x808080 xA0A0A0 xC0C0C0 xE0E0E0
```

これは透明化される背景が白で、境界は黒、軸は暗い灰色、描画色はグレイスケールで 6 色を使うことになります。

ページサイズは 640x480 ピクセルです。**gif** ドライバはカラー、または白黒の出力を生成しますが、どちらを生成するかの制御はできません。

現在のバージョンの **gif** ドライバは、アニメーション GIF はサポートしていません。

34.49.26 Unixplot

unixplot ドライバは、デバイスに依存しない GNU の plot グラフィック言語の出力を生成します。それを "plot2ps" で PostScript 形式の出力にした大きさのデフォルトは 5 インチ x 3 インチです。これは **set size** で約 8.25 x 8.25 まで増加させることが可能です。

書式:

```
set terminal unixplot {"<fontname>"} {<fontsize>}
```

デフォルトは 10 ポイントの "Courier" です。

非 GNU 版の **unixplot** ドライバもありますが、それは GNU 版のを取り除かないとコンパイルはされません。

34.49.27 Gpic

gpic ドライバは FSF (the Free Software Foundations) の "groff" パッケージの中の GPIC 形式のグラフを生成します。デフォルトの大きさは 5 x 3 インチです。オプションは原点に関するもののみで、デフォルトでは (0,0) です。

書式:

```
set terminal gpic {<x> <y>}
```

ここで **x** と **y** の単位はインチです。

単純なグラフを整形するには以下のようにします。

```
groff -p -mpic -Tps file.pic > file.ps
```

pic からの出力はパイプで eqn に渡すこともできるので、'set label' と **set {x/y}label** コマンドでグラフに複雑な関数の式を入れることも可能です。例えば、

```
set ylab '@space 0 int from 0 to x alpha ( t ) roman d t@'
```

とすれば、以下のコマンドによって y 軸に綺麗な積分が見出し付けされます。

```
gpic filename.pic | geqn -d@@ -Tps | groff -m[macro-package] -Tps
    > filename.ps
```

このようにして作られた図は文書に綺麗に当てはまるように縮尺することができます。pic 言語は簡単に理解できるので、必要なら容易にグラフを直接編集できます。**gnuplot** で作られる pic ファイルの全ての座標は x+gnuplotx, y+gnuploty の形で与えられます。デフォルトでは x, y の値は 0 です。いくつかのファイルに対してその x,y を 0 と設定している行を削除すれば、以下のようにして複数のグラフを一つの図の中に入れてしまうこともできます (デフォルトの大きさは 5.0x3.0 インチ):

```
.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
```

これは、横に 2 つ、縦に 2 つずつ並んだ 4 つのグラフからなる、8 インチの広さの図を生成します。

以下のコマンドでも同じことができます。

```
set terminal gpic x y
```

これを例えば以下のように使います。

```
.PS 6.0
copy "trig.pic"
.PE
```

34.49.28 Gpr

gpr ドライバは Apollo Graphics Primitive Resource の固定サイズウィンドウをサポートします。オプションはありません。

色々なサイズのウィンドウが必要なら、**apollo** ドライバを代わりに使用してください。

34.49.29 Grass

grass ドライバは GRASS 地理情報システムのユーザが **gnuplot** を利用することを可能にします。詳しい情報については grassp-list@moon.cecer.army.mil に連絡を取ってください。ページは GRASS グラフウィンドウの現在のフレームに書かれます。オプションはありません。

34.49.30 Hp2623a

hp2623a ドライバはヒューレットパッカード (Hewlett Packard) HP2623A をサポートします。オプションはありません。

34.49.31 Hp2648

hp2648 ドライバはヒューレットパッカード (Hewlett Packard) HP2647 と HP2648 をサポートします。オプションはありません。

34.49.32 Hp500c

hp500c ドライバはヒューレットパッカード (Hewlett Packard) 社の HP DeskJet 500c をサポートします。これには解像度と圧縮に関するオプションがあります。

書式:

```
set terminal hp500c {<res>} {<comp>}
```

ここで **res** は 75, 100, 150, 300 のいずれかの解像度 (DPI; dots per inch) で、**comp** は "rle" か "tiff" です。他の設定をするとそれはデフォルトの値になります。デフォルトは 75 dpi で圧縮はなしです。高解像度でのラスタライズはたくさんのメモリを必要とします。

34.49.33 Hpgl

hpgl ドライバは HP7475A プロッタのような装置用の HPGL 出力を行ないます。これは 2 つの設定可能なオプションを持ちます: それはペンの数と **eject** オプションで、"eject" は描画後にプロッタにページを排出させるよう指示しデフォルトでは 6 つのペンを使い、描画後のページの排出は行ないません。

国際的文字セット ISO-8859-1 と CP850 を **set encoding iso_8859_1** や **set encoding cp850** で認識させることができます (詳細は **set encoding** を参照してください)。

書式:

```
set terminal hpgl {<number_of_pens>} {eject}
```

以下の設定

```
set terminal hpgl 8 eject
```

は、以前の **hp7550** ドライバと同等で、設定

```
set terminal hpgl 4
```

は、以前の **hp7580b** ドライバと同等です。

pcl5 ドライバは、Hewlett-Packard Designjet 750C、Hewlett-Packard Laserjet III, Hewlett-Packard Laserjet IV のようなプロッタをサポートします。これは実際には HPGL-2 を使用しているのですが、装置間で名前の衝突があります。このドライバにはいくつかのオプションがありますが、それらは以下に示した順序で指定しなければいけません:

書式:

```
set terminal pcl5 {mode <mode>} {<plotsize>}
    {{color {<number_of_pens>}} | monochrome} {solid | dashed}
    {font <font>} {size <fontsize>} {pspoints | nopspoints}
```

<mode> は **landscape** か **portrait** です。<plotsize> はグラフの物理的な描画サイズで、それは以下のうちのいずれかです: **letter** は標準の (8 1/2" X 11") 出力、**legal** は (8 1/2" X 14") 出力、**noextended** は (36" X 48") 出力 (letter サイズ比)、**extended** は (36" X 55") 出力 (ほぼ legal サイズ比)。**color** は複数のペン (すなわちカラー) 描画用で <number_of_pens> はカラー出力で使用されるペンの本数 (すなわち色数) です。**monochrome** は 1 本のペン (例えば黒) の描画です。**solid** は全ての線を実線で描

き、**dashed** は異なる点線や鎖線パターンで線を描き分けます。<font> は **stick**, **univers**, **cg_times**, **zapf_dingbats**, **antique_olive**, **arial**, **courier**, **garamond_antigua**, **letter_gothic**, **cg_omega**, **albertus**, **times_new_roman**, **clarendon**, **coronet**, **marigold**, **truetype_symbols**, **wingdings** のいずれかです。<fontsize> はポイント単位でのフォントの大きさです。点の種類 (point type) は、**nopspoints** を指定することで標準的なデフォルトの設定から選択できるようになりますが、**pspoints** を指定すると postscript terminal と同じ点の種類の設定から選択できるようになります。

これらのオプションのいくつかの組み込まれたサポートは、プリンタに依存することに注意してください。例えば全てのフォントは恐らく HP Laserjet IV ではサポートされているでしょうが、HP Laserjet III と Designjet 750C では 2,3 (例えば univers, stick) がサポートされているのみでしょう。また、laserjet は白黒の出力装置なので、それらではカラーも明らかに使えません。

デフォルト: landscape, noextended, color (6 pens), solid, univers, 12
point, nopspoints

pcl5 では国際的文字セットはプリンタで扱われますので、テキスト文字列に適切な 8-bit 文字コードを入れるだけで、わざわざ **set encoding** で邪魔をする必要はありません。

HPGL グラフィックは多くのソフトウェアパッケージで取り込むことが可能です。

34.49.34 Hpljii

hpljii ドライバは HP Laserjet Sries II プリンタを、**hpdj** ドライバは HP DeskJet 500 プリンタをサポートします。これらのドライバでは、解像度の選択が可能です。

書式:

```
set terminal hpljii | hpdj {<res>}
```

ここで **res** は 75, 100, 150, 300 のいずれかの解像度 (DPI; dots per inch) で、デフォルトは 75 です。高解像度でのラスタライズはたくさんのメモリを必要とします。

hp500c ドライバは **hpdj** とほぼ同じですが、**hp500c** は加えてカラーと圧縮もサポートしています。

34.49.35 Hppj

hppj ドライバは HP PaintJet と HP3630 プリンタをサポートします。オプションはフォントを選択するものがあるのみです。

書式:

```
set terminal hppj {FNT5X9 | FNT9X17 | FNT13X25}
```

中間サイズフォント (FNT9X17) がデフォルトです。

34.49.36 Imagen

imagen ドライバは Imagen レーザプリンタをサポートします。これは 1 ページに複数のグラフを配置することも可能です。

書式:

```
set terminal imagen {<fontsize>} {portrait | landscape}
                    {[<horiz>,<vert>]}
```

fontsize はデフォルトでは 12 ポイントで、レイアウトのデフォルトは **landscape** です。<**horiz**> と <**vert**> はグラフを横方向と縦方向に何列置くかを指定します。これらのデフォルトは 1 です。

例:

```
set terminal imagen portrait [2,3]
```

これは、1 ページに 6 つのグラフを横に 2 列、縦に 3 列、縦置き (portrait) で配置します。

34.49.37 Iris4d

iris4d ドライバは Silicon Graphics IRIS 4D コンピュータをサポートします。唯一のオプションは 8-bit か 24-bit かの色の深さの指定で、デフォルトでは 8 です。

書式:

```
set terminal iris4d {8 | 24}
```

色の深さは自由に選べるものではなく、ハードウェアに適切な値を選択すべきです。

24-bit モードを使う場合、その色は直接ファイル .gnuplot_iris4d で指定することができますが、このファイルは現在のディレクトリ、HOME 環境変数に設定されるホームディレクトリの順に検索されます。このファイルには、背景、境界、見出し、そして 9 個の描画色の RGB 値を、この順に書きます。例えば以下にデフォルトの色を持つファイルを記します:

```
85   85   85     Background   (暗い灰色; dark gray)
0    0    0      Boundary     (黒; black)
170  0    170    Labeling     (赤紫; magenta)
85   255  255    Plot Color 1 (明るい水色; light cyan)
170  0    0      Plot Color 2 (赤; red)
0    170  0      Plot Color 3 (緑; green)
255  85   255    Plot Color 4 (明るい赤紫; light magenta)
255  255  85     Plot Color 5 (黄色; yellow)
255  85   85     Plot Color 6 (明るい赤; light red)
85   255  85     Plot Color 7 (明るい緑; light green)
0    170  170    Plot Color 8 (水色; cyan)
170  170  0      Plot Color 9 (茶色; brown)
```

このファイルは RGB の 3 つの組が丁度 12 行なくてはいけません。空行も許されませんが、各行の 3 つ目以降の文字は全て無視されます。

34.49.38 Kyo

kyo と **prescribe** のドライバは Kyocera (京セラ) レーザープリンタをサポートします。この両者の唯一の違いは、**kyo** が "Helvetica" を使うのに対して **prescribe** が "Courier" を使うことだけです。オプションはありません。

34.49.39 Latex

latex と **emtex** ドライバは 2 つのオプションを持ちます。

書式:

```
set terminal latex | emtex {courier | roman | default} {<fontsize>}
```

fontsize には任意のサイズを指定します。デフォルトは、それを埋め込む文書のフォントの設定を引き継ぎます。

あなたの DVI ドライバが任意のサイズのフォントを作り出すことができない場合 (例えば dvips)、標準的な 10, 11, 12 ポイントサイズでなんとかしのいでください。

METAFONT ユーザへの警告: METAFONT は妙なサイズは好みません。

LaTeX に関する全てのドライバは文字列の配置の制御に特別な方法を提供します: '{' で始まる文字列は、'}' で閉じる必要がありますが、その文字列全体が水平方向にも垂直方向にもセンタリングされます。'[' で始まる文字列の場合は、位置の指定をする文字列 (t,b,l,r のうち 2 つまで) が続き、次に']{'、文字列本体、で最後に '}' としますが、この文字列は LaTeX が LR-box として整形します。'\rule{}{}' を使えばさらに良い位置合わせが可能でしょう。

数ある中で、点 (point) は、LaTeX のコマンド "\Diamond" と "\Box" を使って描かれます。これらのコマンドは現在は LaTeX2e のコアには存在せず、latexsym パッケージに含まれていますが、このパッケージ基本配布の一部であり、よって多くの LaTeX のシステムの一部になっています。このパッケージを使うことを忘れないでください。

例: 見出しの位置合わせに関して: gnuplot のデフォルト (大抵それなりになるが、そうでないこともある):

```
set title '\LaTeX\ -- $ \gamma $'
```

水平方向にも垂直方向にもセンタリング:

```
set label '{\LaTeX\ -- $ \gamma $}' at 0,0
```

位置を明示的に指定 (上に合わせる):

```
set xlabel '[t]{\LaTeX\ -- $ \gamma $}'
```

他の見出し – 目盛りの長い見出しに対する見積り:

```
set ylabel '[r]{\LaTeX\ -- $ \gamma $\rule{7mm}{0pt}}'
```

34.49.40 Linux

linux ドライバには指定するオプションは何もありません。それはデフォルトのモード用に環境変数 GSVGAMODE を参照します。もしそれが設定されていなければ 1024x768x256 をデフォルトとして使用しますが、それができなければ 640x480x16 (標準の VGA) とします。

34.49.41 Lips

lips ドライバでは色々なオプションが設定できます。

書式:

```
set terminal lips {<version>} {auto | manual} {<mode>}
                  {color | monochrome}  {solid | dashed}
                  {<duplexing>}
                  {"<fontname>"} {<fontsize>}
```

<version> は **iii** か **iv**、<mode> は **landscape** か **portrait** か **default** です。**manual** はプリンタ用紙を MP トレイから給紙し、**auto** は自動モードで給紙します。**solid** は全ての点線のパターンの描画を実線にします。<duplexing> は **defaultplex** か **simplex** か **duplex** で、LIPS の "duplexing" は、プリンタに 1 ページの両面印刷を行なわせます。"<**fontname**>" は有効なプリンタフォント名を、<**fontsize**> はポイント単位でのフォントサイズを指定します。

default モードは全てのオプションをデフォルト値にします。デフォルトは **iv**, **landscape**, **auto**, **monochrome**, **dashed**, **defaultplex**, "Swiss", 14pt です。LIPS 出力のデフォルトのサイズは A4 です。

34.49.42 Macintosh

macintosh ドライバではいくつかのオプションを設定できます。

書式:

```
set terminal macintosh {singlewin | multiwin} {vertical | novertical}
                       {size <width>, <height> | default}
```

'singlewin' は単一ウィンドウへの制限であり、アニメーション用に有用です。'multiwin' は複数のウィンドウが使えるようになります。'vertical' は gx オプションの元でのみ有効です。このオプションでは回転した文字列を垂直方向に書きます。`novertical` はこのオプションを無効にします。

`size <width>, <height>` は `preferences dialog` で設定されたグラフサイズを書き換え、`'set term mac size default'` か `'set term mac default'` が指定されるまでその値は保持されます。

`'set term mac size default'` はウィンドウサイズの設定を `preferences dialog` で設定したものにします。

`'set term mac default'` は全てのオプションをデフォルトの値に設定します。デフォルト値は `nogx, multiwin, novertical` です。

`multiwin` オプションの元でグラフを作成し、その後 `singlewin` にスイッチした場合、次の `plot` コマンドでは更に一つウィンドウが作られ、この新しいウィンドウでは `singlewin` が有効な間使用されます。そして `multiwin` に戻り、いくつかのグラフを生成し、また再び `singlewin` にスイッチすると、もし元の `'singlewin'` ウィンドウがまだ開いていればそれが使われ、そうでなければ新しい `'singlewin'` ウィンドウが作られます。`'singlewin'` ウィンドウは番号付けはされません。

34.49.43 Mf

mf ドライバは METAFONT プログラムへの入力ファイルを作ります。よってその図は TeX の文書中では文字と同じように使うことができます。

文書中で図を使うには、**gnuplot** の出力するファイルを入力として METAFONT プログラムを実行する必要があります。よって、ユーザはフォントが作られるプロセスと新しく作ったフォントをドキュメントに取り込むための基礎知識が必要となります。しかし、使用するサイトで METAFONT プログラムが適切に設定されていれば、経験のないユーザでもそう問題なく操作はできるでしょう。

グラフ中の文字は METAFONT の文字セットに基づいてサポートされます。現状では Computer Modern Roman フォントセットが入力ですが、ユーザは必要なフォントを何でも自由に選ぶことができます。ただしその選んだフォントの METAFONT ソースファイルが使える状態になっている必要があります。個々の文字は METAFONT の中で別々のピクチャー変数に保存され、文字が必要になったときにこれらの変数が操作 (回転、伸縮等) されます。欠点は、METAFONT プログラムが解釈に要する時間です。ある計算機 (つまり PC) では、ピクチャー変数をたくさん使用しすぎることで、使えるメモリの量の限界が問題を起こすこともあります。

mf ドライバにはオプションはありません。

34.49.43.1 METAFONT Instructions

- 出力形式 (terminal) を METAFONT にセット:

```
set terminal mf
```

- 出力ファイル名を設定。例えば:

```
set output "myfigures.mf"
```

- グラフの描画。各グラフは別々の文字を生成し、そのデフォルトの大きさは 5x3 インチですが、この大きさは **set size 0.5,0.5** のようにしてどんなサイズにでも自由に変更できます。

- **gnuplot** を終了

- **gnuplot** の出力ファイルに対して METAFONT を実行し、TFM ファイルと GF ファイルを作ります。グラフは割と大きい (5x3 インチ) ので、memmax の値が少なくとも 150000 である METAFONT を使う必要があるでしょう。Unix では、それは通常 bigmf という名前でインストールされているでしょう。以下では、virmf コマンドが big 版の METAFONT であると仮定し、実行例を示します:

- METAFONT の立ち上げ:

```
virmf '&plain'
```

- 出力装置の選択: METAFONT プロンプト ('*') 上で次のように打ちます:

```
\mode:=CanonCX;       % あなたの使用するプリンタを指定
```

- 拡大率 (magnification) の選択 (オプション):

```
mag:=1;               % あなたの好みの値を指定
```

- **gnuplot** で作ったファイルを入力:

```
input myfigures.mf
```

典型的な Unix マシンでは、virmf '&plain' を実行するスクリプト "mf" があるので、virmf &plain の代わりに mf を使えます。これにより mfput.tfm と mfput.$$$gf ($$$ は出力装置の解像度) の 2 つのファイル

が作られます。上の作業は、すべてをコマンドライン上で簡単に実行することもできます: virmf '&plain' '\mode:=CanonCX; mag:=1; input myfigures.mf' この場合、作られるファイル名は myfigures.tfm と myfigures.300gf という名前になります。

- gftopk を使って GF ファイルから PK ファイルを生成:

```
gftopk myfigures.300gf myfigures.300pk
```

gftopk が作るファイルの名前はあなたが使用する DVI ドライバに依存しますので、サイトの TeX の管理者にフォント名の規則について聞いてください。次に TFM ファイルと PK ファイルを適当なディレクトリにインストールするかまたは環境変数を適切な値に設定します。通常それは、TEXFONTS にカレントディレクトリを含めることと、あなたが使用する DVI ドライバが使用している環境変数 (標準的な名前はありませんが ...) に対して同じことをやれば済みます。これは TeX がフォントメトリック (TFM) ファイル を見つけ、DVI ドライバが PK ファイルを見つけられるようにするために必要な作業です。

- 文書にそのグラフを入れるために TeX にそのフォント名を指示:

```
\font\gnufigs=myfigures
```

各グラフは、最初のグラフが文字 0、2 番目のグラフが文字 1 というように、それぞれ一つの文字として保存されています。上記の作業を行なうと、グラフはその他の文字と同じように使うことができ、例えばグラフ 1 と 2 を文書中にセンタリングして置くために plain TeX ファイル中ですべきことは:

```
\centerline{\gnufigs\char0}
\centerline{\gnufigs\char1}
```

だけです。もちろん LaTeX では picture 環境を使って \makebox と \put マクロで任意の位置にグラフを配置することができます。

このやり方は、一度フォントを生成してしまえば、大幅に時間の節約になります: TeX はグラフを文字として使い、それを配置するにはごく少ない時間しか使用しませんし、グラフよりも文書の方が修正することが多いでしょうから。そしてこれは TeX のメモリの節約にもなります。METAFONT ドライバを使うもう最後の一つの利点は、生成される DVI ファイルが本来のデバイス非依存な形になるということです。それは eepic や tpic ドライバのような \special コマンドを全く使わないからです。

34.49.44 Mp

mp ドライバは Metapost プログラムへ入力することを意図した出力を生成し、そのファイルに対して Metapost を実行するとグラフを含む EPS ファイルが作られます。デフォルトでは Metapost は全ての文字列を TeX に通します。これはタイトルや見出しに任意の TeX の記号を本質的に使うことができる、という利点を持つことを意味します。

mp ドライバは以下の形式のコマンドとして実行されます:

```
set term mp {color} {solid} {notex} {mag <magsize>} {"<name>"} {<size>}
```

オプション **color** は線をカラーで書くことを意味し (それをサポートするプリンタやディスプレイ上で)、**monochrome** (または何も指定しない場合) は黒の線が選択されます。オプション **solid** は線を実線で描き、**dashed** (または無指定) は線を異なるパターンの点線で描き分けます。**solid** が指定されてかつ **color** が指定されなかった場合、ほとんど全ての線が同じものになりますが、これも何かの場合には有用でしょうから認められています。

オプション **notex** は完全に TeX を迂回しますので、このオプションの元では見出しには TeX のコードは使うことができません。これは、古いグラフファイル、あるいは TeX では特殊記号として解釈されてしまう $ や % のような一般的な文字をたくさん使うファイルのために用意されています。

TeX におけるフォントサイズの変更は数式には効果がなく、そのような変更を行なうとても簡単な方法は、大域的に拡大率 (magnification factor) を設定する以外にはありません。それがオプション **magnification** の意味です。その場合は拡大率を後ろに指定する必要があります。全ての文字 (グラフではなく) はこの率で拡大されます。数式をデフォルトの 10pt 以外の他のサイズで書きたい場合はこれを使用してください。ただ残念なことに全ての数式が同じサイズになってしまいますが、しかし、以下の MP 出力の編集に関する説明を参照してください。**mag** は **notex** の元でも働きますが、それを行なう意味がないくらい (以下に述べる) フォントサイズオプションはうまく働きます。

引用符で囲まれた名前はフォント名を表し、**set label** や **set title** で明示的にフォントが与えられない場合はこのフォントが使われます。フォントは TeX が認識できる (TFM ファイルが存在する) ものを使う必要があります。デフォルトでは **notex** が選択されていなければ "cmr10" が、そうでなければ "pcrr8r" (Courier) が使われます。**notex** の元でも、Metapost には TFM ファイルは必要です。**pcrr8r.tfm** は LaTeX psnfss パッケージの Courier フォント名として与えられています。**notex** のデフォルトからフォントを変更する場合は、少なくとも 32-126 のコード範囲は ASCII エンコーディングに一致するものを選んでください。**cmtt10** もほぼ使えますが、しかしこれはコード 32 (スペース) にスペースではない文字が入っています。

サイズは 5.0 から 99.99 の間の任意の数字を指定でき、省略された場合は 10.0 が使われます。なるべく **magstep** サイズ、つまり 1.2 の整数かまたは 0.5 乗の 10 倍を小数以下 2 桁未満を丸めた値を使用することをお勧めします。それはそれが TeX のシステムで最もよく使われるフォントのサイズだからです。

全てのオプションは省略可能です。フォントを指定する場合はそれは (必要ならサイズもつけて) 一番最後に指定する必要があります。フォント名にそのサイズ情報が含まれていたとしても、サイズを変えるにはフォントサイズを指定する必要があります。例えば **set term mp "cmtt12"** は cmtt12 をデフォルトのサイズである 10 に縮めて使います。それは多分望まないことでしょうし cmtt10 を使う方が良いでしょう。

以下の ascii 文字は、TeX では特別に扱われます:

```
$, &, #, %, _;   |, <, >;   ^, ~,   \, {, }
```

$, #, &, _, % の 5 つは、例えば \$ とすることで容易にそれをエスケープできます。<, >, | の 3 つは、例えば $<$ のように数式モードに入れてやればうまくいきます。残りのものに関しては少し TeX の回避策が必要になりますが、適当なよい TeX の本がそれを指導してくれるでしょう。

見出しを二重引用符で囲む場合、TeX コードのバックスラッシュはエスケープする (2 つ書く) 必要があります。単一引用符を使えばそれを避けることはできますが、今度は改行として \n を使えなくなります。これを書いている現在、gnuplot 3.7 は **plot** コマンドで与えられたタイトルは、別な場所で与えられた場合とは異なる処理をしますし、引用符のスタイルにかかわらず TeX コマンドのバックスラッシュは二重化した方が良さそうです。

Metapost の画像は TeX の文書内で一般に使われています。Metapost はフォントを TeX が行なうのと全く同じ方法で扱い、それは他の大抵の文書整形プログラムとは異なっています。グラフが LaTeX の文書に graphics パッケージで取り込まれ、あるいは epsf.tex を使って plainTeX に取り込まれ、そして dvips (または他の dvi から ps への変換ソフト) で PostScript に変換される場合、そのグラフ内の文字は大抵は正しく扱われているでしょう。しかし、Metapost 出力をそのまま PostScript インタプリタに送っても、グラフ内の文字は出力されないでしょう。

34.49.44.1 Metapost Instructions

- まず terminal ドライバを Metapost に設定、例えば:

```
set terminal mp mono "cmtt12" 12
```

- 出力ファイルを選択、例えば:

```
set output "figure.mp"
```

- グラフを作成。各 plot (または multiplot の各グループ) はそれぞれ別な Metapost beginfig...endfig グループに分けられます。そのデフォルトのサイズは 5x3 インチですが、それは **set size 0.5,0.5** とか、そうしたいと思う適当な割合をそのように指定することで変更できます。

- gnuplot を終了。

- gnuplot の出力ファイルに対して Metapost を実行して EPS ファイルを作成:

```
mpost figure.mp  OR  mp figure.mp
```

Metapost プログラム名はシステムに依存し、Unix では通常 **mpost** で、他の多くのシステムでは **mp** です。Metapost は各グラフに対して 1 つずつの EPS ファイルを生成します。

- そのグラフを文書に取り込むには LaTeX graphics パッケージや、plainTeX では epsf.tex を使用:

```
\usepackage{graphics} % LaTeX
\input epsf.tex       % plainTeX
```

TeX DVI 出力を PS に変換するのに、dvips 以外の DVI ドライバを使う場合は、LaTeX ファイルに以下の行を入れる必要があるかも知れません:

```
\DeclareGraphicsRule{*}{eps}{*}{}
```

作られた各グラフは分離したファイルになっていて、最初のグラフのファイルは、例えば figure.0, 2 つ目は例えば figure.1 のような名前になります。よって、3 つ目のグラフを文書に取り込むためにあなたがしなければいけないことは以下のみです:

```
\includegraphics{figure.2} % LaTeX
\epsfbox{figure.2}         % plainTeX
```

mp ドライバの postscript ドライバに代わる利点は、もしあるとすれば、それは編集可能な出力であるということでしょう。この出力を可能な限り綺麗にするための、かなりの努力が払われました。Metapost 言語に関するそういった知識のおかげで、デフォルトの線種や色は配列 **lt**[] や **col**[] を編集することで変更できるようになりました。実線/点線、カラー/白黒といった選択も、真偽値として定義されている **dashedlines** や **colorlines** を変更することで行なえます。デフォルトの **tex** オプションが有効な場合、ブロック **vebatimtex...etex** を編集することで、ラベル文字フォントに対する大域的な変更が行なえます。特に、もし望むなら LaTeX のプリアンブルを追加することもでき、その場合 LaTeX の持つサイズ変更コマンドを使えるので最大の柔軟性を発揮できるでしょう。ただし、Metapost に plainTeX でなく LaTeX を実行させるよう、適切な MP 設定変数を設定することを忘れないでください。

34.49.45 Mgr

mgr ドライバは Mgr ウィンドウシステムをサポートします。オプションはありません。

34.49.46 Mif

mif ドライバは Frame Maker MIF フォーマット (version 3.00) の出力を生成します。これは 15*10 cm のサイズの MIF フレームを出力し、同じペンで書かれるグラフの基本要素は同じ MIF グループにグループ化されます。**gnuplot** の 1 ページにおけるグラフの基本要素は一つの MIF フレームに描画され、いくつかの MIF フレームは一つの大きな MIF フレーム内に集められます。文字列で使われる MIF フォントは "Times" です。

MIF 3.00 ドライバではいくつかのオプションが設定できます。

書式:

```
set terminal mif {colour | monochrome} {polyline | vectors}
                 {help | ?}
```

colour は線種 (line type) >= 0 の線をカラー (MIF separation 2–7) で、**monochrome** は全ての線を黒 (MIF separation 0) で描画します。**polyline** は曲線を連続曲線として描画し、**vectors** は曲線をベクトルの集まりとして描画します。**help** と ? はオンラインヘルプを標準エラー出力に表示します。両者はその使用法の短い説明を出力し、**help** は更にオプションも表示します。

例:

```
set term mif colour polylines    # デフォルト
set term mif                     # デフォルト
set term mif vectors
set term mif help
```

34.49.47 Mtos

mtos ドライバにはオプションはありません。これはデータを GPCLIENT という外部プログラムにパイプ経由で渡します。それは MULTIOS, Magic 3.x, MagicMAC, MiNT 上で動作します。もし GPCLIENT を見つけられなかったら、dirk@lstm.uni-erlangen.de へメールしてください。

34.49.48 Next

next ドライバには設定のためのいくつかのオプションがあります。

書式:

```
set terminal next {<mode>} {<type> } {<color>} {<dashed>}
           {"<fontname>"} {<fontsize>} title {"<newtitle>"}
```

<mode> は **default** のみ指定でき、その場合全てのオプションがデフォルトになります。<type> は **new** か **old** で、**old** は古い単一ウィンドウを要求します。<color> は **color** (カラー) か **monochrome** (白黒)、<dashed> は **solid** (実線のみ) か **dashed** (点線が有効)、"<fontname>" は有効な PostScript フォントの名前を、<fontsize> は PostScript ポイント単位でのフォントのサイズを、<title> は GnuTerm ウィンドウのタイトルをそれぞれ設定します。デフォルトは **new**, **monochrome**, **dashed**, "Helvetica", 14pt です。

例:

```
set term next default
set term next 22
set term next color "Times-Roman" 14
set term next color "Helvetica" 12 title "MyPlot"
set term next old
```

点の大きさは **set linestyle** で変更できます。

34.49.49 Next

next ドライバには設定のためのいくつかのオプションがあります。

書式:

```
set terminal next {<mode>} {<type> } {<color>} {<dashed>}
           {"<fontname>"} {<fontsize>} title {"<newtitle>"}
```

<mode> は **default** のみ指定でき、その場合全てのオプションがデフォルトになります。<type> は **new** か **old** で、**old** は古い単一ウィンドウを要求します。<color> は **color** (カラー) か **monochrome** (白黒)、<dashed> は **solid** (実線のみ) か **dashed** (点線が有効)、"<fontname>" は有効な PostScript フォントの名前を、<fontsize> は PostScript ポイント単位でのフォントのサイズを、<title> は GnuTerm ウィンドウのタイトルをそれぞれ設定します。デフォルトは **new**, **monochrome**, **dashed**, "Helvetica", 14pt です。

例:

```
set term next default
set term next 22
set term next color "Times-Roman" 14
set term next color "Helvetica" 12 title "MyPlot"
set term next old
```

点の大きさは **set linestyle** で変更できます。

34.49.50 Pbm

pbm ドライバ — PBMplus 用ドライバ — にはいくつかのオプションが設定できます。

書式:

```
set terminal pbm {<fontsize>} {<mode>}
```

<fontsize> は **small** か **medium** か **large** で、<mode> は **monochrome** か **gray** か **color** です。デフォルトの描画サイズは 640 ピクセルの幅で 480 ピクセルの高さですが、これは **set size** で変更できます。

pbm ドライバの出力は <mode> によります: **monochrome** は portable bitmap (PBM; 1 ピクセル 1 ビット) を、**gray** は portable graymap (PGM; 1 ピクセル 3 bit) を、**color** は portable pixmap (PPM; 1 ピクセル 4 ビット) を出力します。

このドライバの出力は Jef Poskanzer の優れた PBMPLUS パッケージで使うことができ、このパッケージは上記の PBMPLUS フォーマットから GIF, TIFF, MacPaint Macintosh PICT, PCX, X11 ビットマップ、

その他多くの形式に変換するプログラムを提供します。PBMPLUS は ftp.x.org から取得できます。ファイル名は "netpbm-1mar1994.p1" で始まり、/contrib/utilities にあります。このパッケージは多分 ftp.x.org の他の多くのミラーサイトからも取得できるでしょう。

例:

```
set terminal pbm small monochrome             # デフォルト
set size 2,2; set terminal pbm color medium
```

34.49.51 Dospc

dospc ドライバは PC の任意のグラフィックボードをサポートし、それらを自動検出します。これは、gcc, または Zortec C/C++ コンパイラを使っていないときのみ利用可能です。

34.49.52 Pdf

このドライバは Adobe PDF (Portable Document Format) 出力を生成し、それは Acrobat Reader のようなツールで表示、印刷ができます。

書式:

```
set terminal pdf {fname "<font>"} {fsize <fontsize>}
```

<font> はデフォルトで使われるフォント名 (デフォルトでは Helvetica) で <fontsize> はポイント単位でのフォントサイズ (デフォルトでは 12) です。

34.49.53 Pm

pm ドライバは、グラフが描画される OS/2 プレゼンテーションマネージャウィンドウを提供します。そのウィンドウは最初のグラフが描画されたときに開かれます。このウィンドウは印刷、クリップボードへのコピー、いくつかの線種や色の調整のための機能、そしてそれ自身のオンラインヘルプを持っています。**multiplot** オプションもサポートされています。

書式:

```
set terminal pm {server {n}} {persist} {widelines} {enhanced} {"title"}
```

persist が指定されると、各グラフはそれぞれ自身のウィンドウを持ち、そのすべてのウィンドウは **gnuplot** が終了した後も開いたままになります。**server** が指定されると、全てのグラフは同じウィンドウ内に現われ、それは **gnuplot** 終了後も開いたままになります。このオプションは、さらに追加の数引数を取り、その数字はサーバプロセスのインスタンスになります。よって、同時に複数のサーバウィンドウを使うことができます。

widelines が指定されると、全てのグラフは幅の広い線で描かれます。**enhanced** が指定されると **enhanced postscript** オプション (詳細は **set terminal postscript enhanced** 参照) と同じ書式で上付き文字や下付き文字、複数のフォントを使うことができます。基本的な PostScript フォントのフォント名は 1 文字に省略できます。

title が指定されると、それは描画ウィンドウのタイトルとして使われます。それはサーバインスタンス名としても使われ、それは追加の数引数を上書きします。

線の幅は **set linestyle** で変更できます。

34.49.54 Png

png ドライバは PNG (Portable Network Graphics) をサポートします。これをコンパイルするには、公開ライブラリである "libpng" と "zlib" が必要です。いずれも http://www.cdrom.com/pub/png/ にあります。**png** には 4 つのオプションがあります。

デフォルトでは **png** ドライバは Web に合うパレットの割り当てを使用します。

書式:

```
set terminal png {small | medium | large}
                 {transparent | notransparent}
                 {monochrome | gray | color}
                 {<color0> <color1> <color2> ...}
```

transparent はドライバに透明化 (transparent) PNG を生成するよう指示します。この場合最初の指定色が透明化される色になります。

デフォルトは small (のフォントサイズ) でカラーです。デフォルトの出力サイズは 640*480 ピクセルです。

個々の色は 'xrrggbb' の形式で指定します。ここで、x はそのまま文字 'x' を意味し、'rrggbb' は赤、緑、青の 16 進数での成分です。例えば 'x00ff00' は緑を表します。背景色を最初に指定し、その後ろに境界色、X 軸と Y 軸の色、グラフの描画色、と続きます。色数は現在のところ最大 99 色まで設定できます。

34.49.55 Postscript

postscript ドライバではいくつかのオプションが設定できます。

書式:

```
set terminal postscript {<mode>}
                        {enhanced | noenhanced | plus | noplus}
                        {color | monochrome} {solid | dashed}
                        {<duplexing>}
                        {"<fontname>"} {<fontsize>}
```

<mode> は **landscape** か **portrait** か **eps** か **default** で、**solid** は全てのグラフの点線のパターンを実線で上書きします。<duplexing> は **defaultplex** か **simplex** か **duplex** です (PostScript の "duplexing" はプリンタの 1 ページ両面印刷の機能を有効にします。あなたのプリンタがそれを行なえないなら設定しないでください)。**enhanced** は "拡張 PostScript" (上付き文字、下付き文字、および複数のフォントの利用) の機能を有効にします。"<**fontname**>" は有効な PostScript フォントの名前で、<**fontsize**> は PostScript ポイント単位でのフォントの大きさです。

default モードは全てのオプションを以下のデフォルトの値に設定します: **landscape**, **monochrome**, **dashed**, **defaultplex**, **noenhanced**, "Helvetica", 14pt。PostScript のグラフのデフォルトの大きさは、10 インチの幅で 7 インチの高さです。

eps モードは EPS (Encapsulated PostScript) 出力を生成しますが、これは通常の PostScript に、それを他の多くのアプリケーションで取り込むことができるようにいくつかの行を追加したものです (追加される行は PostScript のコメント行なので、よってそれ自身もちゃんと印刷できます)。EPS 出力を得るには **eps** モードを使用し、1 つのファイルには 1 つのグラフのみ、としてください。**eps** モードではフォントも含めてグラフ全体がデフォルトの大きさの半分に縮められます。

例:

```
set terminal postscript default       # 以前の postscript
set terminal postscript enhanced      # 以前の enhpost
set terminal postscript landscape 22  # 以前の psbig
set terminal postscript eps 14        # 以前の epsf1
set terminal postscript eps 22        # 以前の epsf2
set size 0.7,1.4; set term post portrait color "Times-Roman" 14
```

線の幅と点の大きさは **set linestyle** で変更できます。

postscript ドライバは約 70 種類の異なる点種をサポートしていて、これは **plot** や **set linestyle** の **pointtype** オプションで選択できます。

gnuplot と Postscript に関する多分有用と思われるファイルが **gnuplot** の配布物、またはその配布サイトの /docs/ps サブディレクトリ内にいくつか含まれています。そこには "ps_symbols.gpi" (実行すると **postscript** ドライバで使える全ての記号を紹介する "ps_symbols.ps" というファイルを生成する **gnuplot** のコマンドファイル)、"ps_guide.ps" (拡張された書式に関する要約と、文字列内で 8 進コードで生成されるもの、symbol フォント等を含む PostScript ファイル)、"ps_file.doc" (**gnuplot** で作られる PostScript ファイルの構造の説明を含むテキストファイル) があります。

PostScript ファイルは編集可能で、一度 **gnuplot** でそれを作れば、それを望むように修正することは自由に行なえます。"editing postscript" の節を参照すればそのためのヒントがいくつか得られるでしょう。

34.49.55.1 Enhanced postscript

拡張テキスト制御記号		
制御記号	例	説明
^	a^x	上付き文字
_	a_x	下付き文字
@	@x or a@^b_c	空ボックス (幅がない)
&	&{space}	指定した長さのスペースを挿入

中カッコは一文字が期待される箇所に複数の文字を書く場合に使われます (例えば 2^{10})。フォント、およびそのサイズを変更するには、以下の形式を使ってください: {/[fontname][=fontsize | *fontscale] text} よって、例えば {/Symbol=20 G} は 20 ポイントのガンマに、{/*0.75 K} は現在の有効なフォントサイズの 3/4 の大きさの K になります (文字 '/' は、必ず '{' の後ろの最初の文字でなくてはなりません)。

set encoding によって文字エンコード系が変更されている場合、'/' の次に '-' を書くとデフォルトの文字エンコード系が代わりに使われます。しかし、これは Symbol フォントを使う場合は必要ありません。それは /Symbol は自分自身のエンコード系を使い、**gnuplot** は他のエンコーディング系をそれには適用しないからです。

空ボックス (phantom box) は a@^b_c の上付き文字と下付き文字を揃えるときに有用ですが、文字にアクセント記号を重ねる場合にはうまく働きません (後者を行なうには **set encoding iso_8859_1** で ISO Latin-1 エンコーディング系に変更し、そこに含まれるたくさんの種類のアクセントや他のダイアクリティカルマークのついた文字を利用する方がいいでしょう)。そのボックスはスペーシングが行なわれないので、ボックス内 (つまり @ の後ろ) の上付き文字や下付き文字を短く出力するのに適しています。

ある文字列と同じ長さのスペースを文字 '&' を使うことで入れることができます。すなわち、

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

は以下を生成します (abc と ghi の間は 3 文字分の空白):

```
'abc   ghi'
```

\ 文字コード (8 進数) を指定することで特殊な記号を指定することができます。例えば、{/Symbol \245} は無限大の記号になります。

制御文字は、 \\ や \{ などのように \ を使ってエスケープできます。

しかし、二重引用符内の文字列は単一引用符内の文字列とは異なって解釈されることを知っておいてください。主な違いは、二重引用符内の文字列ではバックスラッシュは 2 つ重ねる必要があることです。

例 (これらは言葉で説明するのは難しいのでとりあえずやってみてください):

```
set xlabel 'Time (10^6 {/Symbol m}s)'
set title '{/Symbol=18 \362@_{/=9.6 0}^{/=12 x}} \
          {/Helvetica e^{-{/Symbol m}^2/2} d}{/Symbol m}'
```

gnuplot ソース配布物内の /docs/ps サブディレクトリにあるファイル"ps_guide.ps" に、拡張された書式に関する例が更に載っています。

34.49.55.2 Editing postscript

PostScript 言語はとても複雑な言語で、ここで詳細を記述することはとてもできません。それでも、**gnuplot** で作られる PostScript ファイルには、致命的なエラーをそのファイルに導入してしまう危険性のない変更を行なうことが可能な部分があります。

例えば、PostScript の文 "/Color true def" (**set terminal postscript color** コマンドに答えてファイルに書き込まれます) を変更して、その描画を白黒のものにする方法はおわかりでしょう。同様に、線の色、文字の色、線の太さ (weight)、記号のサイズも、本当に簡単に書き換えられるでしょう。タイトルや見出しなどの文字列の誤植や、フォントの変更も編集可能でしょう。任意のものの配置も変更できますし、もちろん、任意のものを追加したり、削除したりもできますが、それらの修正は PostScript 言語の深い知識が必要でしょう。

gnuplot によって作られる PostScript ファイルの構成に関しては、/docs/ps サブディレクトリのテキストファイル "ps_file.doc" に述べられています。

34.49.55.3 Postscript plus

plus オプションが有効な場合、フォントの変更、および上付き文字、下付き文字、ギリシャ文字等は LaTeX と同様の書式が使えます。

拡張テキスト制御記号		
制御記号	例	説明
^	a^x	上付き文字
_	a_x	下付き文字

例:

```
set label 1 '\rotate=-20{\size=20\bf \sqrt{2\beta}}'
set xlabel 'Time $(10^6 \mu s)$'
set ylabel 'Length $(\AA)$'
set title '${\sm\size=18 \362}_{\small 0}^{\size=12 xi} \
                                    {\rm e^{-\alpha^2} d}\mu$'
```

34.49.56 Pslatex and pstex

pslatex と **pstex** ドライバは、それぞれ LaTeX と TeX で後処理される出力を生成する。**pstex** で生成される図は、任意の plain-TeX ベースの TeX (LaTeX もそうです) で取り込むことができます。

書式:

```
set terminal pslatex | |pstex {<color>} {<dashed>} {<rotate>}
                              {auxfile} {<font_size>}
```

<color> は **color** か **monochrome**、<rotate> は **rotate** か **norotate** で、それは y-軸の見出しを回転するかどうかを決定します。<font_size> は希望するフォントの (ポイント単位での) 大きさです。

auxfile が指定されると、ドライバは PostScript コマンドを、LaTeX ファイルに直接出力する代わりに、補助ファイルに書き出すようになります。これは、dvips がそれを扱えないくらい大きいグラフである場合に有用です。補助 PostScript ファイルの名前は、**set output** コマンドで与えられる TeX ファイルの名前から導かれるもので、それはその最後の **.tex** の部分 (実際のファイル名の最後の拡張子の部分) を **.ps** で置き換えたもの、または、TeX ファイルに拡張子がないならば **.ps** を最後に付け足したものになります。**gnuplot** を終了する前にファイルを閉じることを忘れないでください。

LaTeX に関する全てのドライバは文字列の配置の制御に特別な方法を提供します: '{' で始まる文字列は、'}' で閉じる必要がありますが、その文字列全体が LaTeX によって水平方向にも垂直方向にもセンタリングされます。'[' で始まる文字列の場合は、位置の指定をする文字列 (t,b,l,r のうち 2 つまで) が続き、次に ']{'、文字列本体、で最後に '}' としますが、この文字列は LaTeX が LR-box として整形します。\rule{}{} を使えばさらに良い位置合わせが可能でしょう。

例:

```
set term pslatex monochrome dashed rotate       # デフォルトに設定
```

PostScript コマンドを "foo.ps" に書き出す:

```
set term pslatex auxfile
set output "foo.tex"; plot ...: set output
```

見出しの位置合わせに関して: gnuplot のデフォルト (大抵それなりになるが、そうでないこともある):

```
set title '\LaTeX\ -- $ \gamma $'
```

水平方向にも垂直方向にもセンタリング:

```
set label '{\LaTeX\ -- $ \gamma $}' at 0,0
```

位置を明示的に指定 (上に合わせる):

```
set xlabel '[t]{\LaTeX\ -- $ \gamma $}'
```

他の見出し – 目盛りの長い見出しに対する見積り:

```
set ylabel '[r]{\LaTeX\ -- $ \gamma $\rule{7mm}{0pt}}'
```

線幅と点の大きさは **set linestyle** で変更できます。

34.49.57 Pstricks

pstricks ドライバは LaTeX の "pstricks.sty" マクロパッケージと共に使われることを意図しています。これは **eepic** や **latex** ドライバに代わる選択肢の一つです。"pstricks.sty" は必要ですが、もちろん PostScript を解釈するプリンタ、または Ghostscript のような変換ソフトも必要です。

PSTricks は anonymous ftp で Princeton.EDU の /pub ディレクトリから取得できます。このドライバは、PSTricks パッケージの全ての能力を使おうとなどとは全く考えてはいません。

書式:

```
set terminal pstricks {hacktext | nohacktext} {unit | nounit}
```

最初のオプションは、あまり綺麗ではない方法で数字のより良い出力を生成するもので、2 つ目のオプションはグラフを縮尺する際には必要です。デフォルトでは **hacktext** と **nounit** です。

34.49.58 Qms

qms ドライバは QMS/QUIC レーザープリンタ、Talaris 1200、その他をサポートします。オプションはありません。

34.49.59 Regis

regis ドライバは REGIS グラフィック言語での出力を生成します。このドライバには色を 4 色使うか (デフォルト) 16 色使うかのオプションがあります。

書式:

```
set terminal regis {4 | 16}
```

34.49.60 Rgip

rgip と **uniplex** ドライバは RGIP メタファイルをサポートします。それらはいくつかのグラフを一つのページの中に入れることができますが、一つの出力ファイル内には 1 つのページしか許されません。

書式:

```
set terminal rgip | uniplex {portrait | landscape}
                            {[<horiz>,<vert>]} {<fontsize>}
```

フォントサイズは 1–8 間での値が許されていて、デフォルトでは 1 です。デフォルトの方向 (layout) は横置き (landscape) です。グラフはそのページに **horiz**x**vert** 個が格子状に配置されます。デフォルトは [1,1] です。

例:

```
set terminal uniplex portrait [2,3]
```

これは 1 ページ 6 つのグラフを横に 2 つ縦に 3 つ並べ、縦置き (portrait) に出力します。

34.49.61 Sun

sun ドライバは SunView ウィンドウシステムをサポートしています。オプションはありません。

34.49.62 Svg

このドライバは W3C SVG (Scalable Vector Graphics) フォーマットを生成します。

書式:

```
set terminal svg {size <x> <y>}
                 {fname "<font>"} {fsize <fontsize>}
```

ここで <x> と <y> は生成される SVG グラフのサイズ、<font> はデフォルトとして使われるフォント名 (デフォルトでは Arial)、<fontsize> はポイント単位でのフォントサイズ (デフォルトは 12) です。

34.49.63 Tek410x

tek410x ドライバは Tektronix 410x, 420x ファミリーをサポートしています。オプションはありません。

34.49.64 Table

table ドライバはグラフを作成する代わりに、グラフがその上に載る各点を出力します。それは **plot** や **splot** コマンドの処理の結果で、X Y {Z} R の複数列の値からなる ASCII 出力です。R の部分は次の 3 つのうちの一つの値になります: その点が有効な範囲内にある場合は "i"、その点が範囲外である場合は "o"、その点が未定義値である場合は "u" です。データの書式は、対応する軸の見出しのフォーマットにより決定され (**set format** 参照)、各列は一つのスペースで区切られます。

その数字を参照したい場合のために、それをスクリーンに表示したりファイルにセーブしたりすることができます。これは、等高線を生成してそれを後々のため、恐らくは **plot** で表示するために保存する場合などに有用です。一つ例がありますので **set contour** を参照してください。同じ方法は、補間データを保存するのにも使われます (**set samples** と **set dgrid3d** を参照してください)。

34.49.65 Tek40

このドライバ群は VT-ライクな端末のいくつかをサポートします。**tek40xx** は Tektronix 4010 とその他ほとんどの TEK エミュレータをサポートします。**vttek** は VT-ライクな tek40xx 端末エミュレータをサポートします。**kc-tek40xx** は、カラーの MS-DOS Kermit Tek4010 ターミナルエミュレータを、**km-tek40xx** はその白黒版をサポートします。**selanar** は Selanar グラフィック端末をサポートします。**bitgraph** は BBN Bitgraph 端末をサポートします。いずれもオプションはありません。

34.49.66 Texdraw

texdraw ドライバは LaTeX texdraw 環境をサポートします。それは texdraw パッケージの "texdraw.sty" と "texdraw.tex" と共に使用されることを仮定しています。

オプションはありません。

34.49.67 Tgif

Tgif は X11 ベースのドローツールです — ただし、これは GIF に関して何かするわけではありません。

tgif ドライバは複数の点の大きさ (**set pointsize** で)、複数の見出し用のフォント、フォントサイズ (例えば **set label "Hallo" at x,y font "Helvetica,34**")、そして 1 ページ内の複数のグラフ描画をサポートします。軸の比率は変更されません。

書式:

```
set terminal tgif {portrait | landscape} {<[x,y]>}
                  {solid | dashed}
                  {"<fontname>"} {<fontsize>}
```

<[x,y]> にはそのページ内の x 方向、y 方向のグラフの数を指定し、"<fontname>" には有効な PostScript フォント名、<fontsize> はその PostScript フォントの大きさを指定します。デフォルトでは **portrait**, [**1,1**], **dashed**, "**Helvetica**", **18** です。

solid オプションは、編集作業中にそうであるように、線がカラーである場合に普通使われます。ハードコピーは白黒になることが多いので、その場合は **dashed** を選択すべきでしょう。

多重描画 (multiplot) は 2 種類の方法で実装されています。

その一つは、標準的な gnuplot の多重描画のやり方です:

```
set terminal tgif
set output "file.obj"
set multiplot
set origin x01,y01
set size  xs,ys
plot ...
     ...
set origin x02,y02
plot ...
set nomultiplot
```

より詳しい情報については **set multiplot** を参照してください。

もう一つの方法はドライバの [x,y] オプションです。この方法の長所は、原点 (origin) や大きさ (size) の設定をしなくても全てのグラフが自動的に縮尺され配置されることです。グラフの比 x/y は、自然な比 3/2 (または **set size** で設定されたもの) が保持されます。

両方の多重描画の実装が選択された場合、標準的なやり方の方が選択され、警告のメッセージが表示されます。

単一描画 (または標準的な多重描画) の例:

```
set terminal tgif                    # デフォルト
set terminal tgif "Times-Roman" 24
set terminal tgif landscape
set terminal tgif landscape solid
```

ドライバの持つ多重描画の仕組みを利用する例:

```
set terminal tgif portrait [2,4]  # 縦置、x-方向に 2 つ、y-方向
                                  # に 4 つのグラフ描画
set terminal tgif [1,2]           # 縦置、x-方向に 1 つ、y-方向
                                  # に 2 つのグラフ描画
set terminal tgif landscape [3,3] # 横置、両方の方向に 3 つのグ
                                  # ラフ描画
```

34.49.68 Tkcanvas

このドライバは Tcl/Tk ベース (デフォルト)、または Perl ベースの Tk canvas widget コマンドを生成します。これを使うには、"term.h" のこのドライバに対応する行のコメント記号を外すか適当な行を書き加えて **gnuplot** を make し直して、以下のように実行します。

```
gnuplot> set term tkcanvas {perltk} {interactive}
gnuplot> set output 'plot.file'
```

そして "wish" を起動した後で、以下の Tcl/Tk コマンド列を実行してください:

```
% source plot.file
% canvas .c
% pack .c
% gnuplot .c
```

Perl/Tk の場合は以下のようにしてこのプログラムを使います:

```
use Tk;
my $top = MainWindow->new;
my $c = $top->Canvas->pack;
my $gnuplot = do "plot.pl";
$gnuplot->($c);
MainLoop;
```

gnuplot によって生成されたコードは "gnuplot" と呼ばれる手続きを作り、それはその引数を canvas の名前とします。その手続きが呼ばれると、それは canvas をクリアし、canvas のサイズを見つけ、その中に丁度収まるようにグラフを書きます。

2 次元の描画 (**plot**) では 2 つの手続きが追加されて定義されます: "gnuplot_plotarea" は描画範囲の境界を含むリスト "xleft, xright, ytop, ybot" を canvas スクリーン座標で返し、2 つの対の軸の描画座標での範囲"x1min, x1max, y1min, y1max, x2min, x2max, y2min, y2max" は手続き"gnuplot_axisranges" を呼べば得られます。"interactive" オプションを指定すると、canvas の線分上でマウスをクリックするとその線分の中点の座標が標準出力に出力されるようになります。さらに、"user_gnuplot_coordinates" という

手続きを定義することで、それに代わる新たな動作を起こすことも可能です。その手続きには以下の引数が渡されます: "win id x1s y1s x2s y2s x1e y1e x2e y2e x1m y1m x2m y2m"、これらは、canvas の名前、線分の id、2 つの座標系でのその線分の開始点の座標、終了点の座標、そして中点の座標です。中点の座標は対数軸に対してのみ与えられます。

tkcanvas の現在の版では **multiplot** も **replot** もサポートしていません。

34.49.69 Tpic

tpic ドライバは tpic \special での LaTeX picture 環境の描画をサポートします。 これは **latex** や **eepic** ドライバに代わる別な選択肢です。点の大きさ (pointsize)、線の幅 (linewidth)、点線の点の間隔 (interval) に関するオプションがあります。

書式:

```
set terminal tpic <pointsize> <linewidth> <interval>
```

pointsize と **linewidth** は整数でミリインチ単位、**interval** は実数で単位はインチです。正でない値を指定するとデフォルトの値が使われます。デフォルトでは pointsize = 40, linewidth = 6, interval = 0.1 です。

LaTeX に関する全てのドライバは文字列の配置の制御に特別な方法を提供します: '{' で始まる文字列は、'}' で閉じる必要がありますが、その文字列全体が LaTeX によって水平方向にも垂直方向にもセンタリングされます。'[' で始まる文字列の場合は、位置の指定をする文字列 (t,b,l,r のうち 2 つまで) が続き、次に ']{'、文字列本体、で最後に '}' としますが、この文字列は LaTeX が LR-box として整形します。\rule{}{} を使えばさらに良い位置合わせが可能でしょう。

例: 見出しの位置合わせに関して: gnuplot のデフォルト (大抵それなりになるが、そうでないこともある):

```
set title '\LaTeX\ -- $ \gamma $'
```

水平方向にも垂直方向にもセンタリング:

```
set label '{\LaTeX\ -- $ \gamma $}' at 0,0
```

位置を明示的に指定 (上に合わせる):

```
set xlabel '[t]{\LaTeX\ -- $ \gamma $}'
```

他の見出し – 目盛りの長い見出しに対する見積り:

```
set ylabel '[r]{\LaTeX\ -- $ \gamma $\rule{7mm}{0pt}}'
```

34.49.70 Unixpc

unixpc ドライバは AT&T 3b1 と AT&T 7300 Unix PC をサポートします。オプションはありません。

34.49.71 Unixplot

unixplot ドライバは Unix の "plot" グラフィック言語の出力を生成します。オプションはありません。

このドライバは、GNU 版の plot を使っている環境ではコンパイルされません。その場合は代わりに **gnugraph** ドライバ (terminal) を利用してください。

34.49.72 Vx384

vx384 ドライバは Vectrix 384 と Tandy カラープリンタをサポートします。オプションはありません。

34.49.73 VWS

VWS ドライバは VAX ウィンドウシステムをサポートします。オプションはありません。このドライバはディスプレイの状態 (白黒か、グレイスケールかカラーか) を自動検知します。全ての線種は実線で描画されます。

34.49.74 Windows

windows ドライバでは 3 つのオプションを設定することができます。

書式:

```
set terminal windows {<color>} {"<fontname>"} {<fontsize>}
```

<color> は **color** か **monochrome** のいずれか、"**<fontname>**" は有効な Windows のフォント名で、**<fontsize>** はポイント単位でのフォントの大きさです。

他のオプションもグラフメニュー、初期化ファイル、または **set linestyle** で設定できます。

Windows 版は、非対話型モードでは通常、コマンドラインから与えたファイルの最後に達すると直ちに終了し、このモードではテキストウィンドウは表示せず、グラフのみの表示となります。オプションとして **/noend** か **-noend** を指定することでこの挙動は無効にできます。

34.49.74.1 Graph-menu **gnuplot graph** ウィンドウでマウスの右ボタンを押すか、システムメニューから **Options** を選択すると以下のオプションを持つポップアップメニューが現われます:

Bring to Top チェックを入れるとグラフウィンドウを他の全ての描画ウィンドウの手前に表示

Color チェックを入れるとカラーの線種が有効、チェック無しでは白黒

Copy to Clipboard クリップボードにビットマップやメタファイルの画像をコピー

Background... ウィンドウ背景色の設定

Choose Font... グラフィックウィンドウで使うフォントの選択

Line Styles... 線の色や種類のカスタマイズ

Print... グラフィックウィンドウを Windows プリンタドライバでプリントアウト。プリンタと拡大率の選択が可能ですが、この **Print** オプションによる印刷結果は **gnuplot** の持つプリンタドライバによるもの程良くはありません。

Update wgnuplot.ini 現在のウィンドウの位置、ウィンドウの大きさ、テキストウィンドウのフォントとそのフォントサイズ、グラフウィンドウのフォントとそのフォントサイズ、背景色、線種を初期化ファ

イル **WGNUPLOT.INI** に保存

34.49.74.2 Printing 好みにより、グラフは以下のような方法で印刷できます。

1. gnuplot のコマンド **set terminal** でプリンタを選択し、**set output** で出力をファイルにリダイレクト

2. gnuplot graph ウィンドウから **Print...** コマンドを選択。テキストウィンドウからこれを行なう特別なコマンド **screendump** もある。

3. set output "PRN" とすると出力は一時ファイルに出力され、**gnuplot** を終了するかまたは **set output** コマンドで出力を他のものへ変更すると、ダイアログ (対話) ボックスが現われ、そこでプリンタポートを選択。そこで OK を選択すると、出力はプリントマネージャでは加工されずにそのまま選択されたポートでプリントアウトされる。これは偶然 (または故意) に、あるプリンタ用の出力を、それに対応していないプリンタに送り得ることを意味する。

34.49.74.3 Text-menu **gnuplot text** ウィンドウでマウスの右ボタンを押すか、システムメニューから **Options** を選択すると以下のオプションを持つポップアップメニューが現われます:

Copy to Clipboard マークしたテキストをクリップボードにコピー

Paste 打ち込んだのと同じようにクリップボードからテキストをコピー

Choose Font... テキストウィンドウで使うフォントの選択

System Colors 選択するとコントロールパネルで設定したシステムカラーをテキストウィンドウに与える。選択しなければ白背景で文字は黒か青。

Update wgnuplot.ini 現在のテキストウィンドウの位置、テキストウィンドウの大きさ、テキストウィンドウのフォントとそのフォントサイズを初期化ファイル **WGNUPLOT.INI** に保存

MENU BAR

メニューファイル **WGNUPLOT.MNU** が WGNUPLOT.EXE と同じディレクトリにある場合、**WGNUPLOT.MNU** に書かれているメニューが読み込まれます。メニューコマンドは以下の通りです:

[Menu] 次の行の名前で新しいメニューを開始

[EndMenu] 現在のメニューを終了

[–] 水平なメニューの仕切りを入れる

[|] 垂直なメニューの仕切りを入れる

[Button] メニューにプッシュボタンを入れ、それに次のマクロを当てはめる

マクロは 2 行で書き、最初の行はマクロ名 (メニューの見出し)、2 行目がマクロ本体です。先頭の空白列は無視されます。マクロコマンドは以下の通りです:

[INPUT] — [EOS] か {ENTER} までをプロンプトとして出力し文字列を入力

[EOS] — 文字列の終り (End Of String)。何も出力しない

[OPEN] — 開くファイル名をリストボックスから取得。[EOS] までがリストボックスのタイトル、その後から次の [EOS] か {ENTER} までがデフォルトのファイル名 (Windows 3.1 の COMMDLG.DLL を使用)。

[SAVE] — セーブファイル名を取得 ([OPEN] 同様)

マクロ文字の置き換えは以下の通りです:

{ENTER} — 復帰 '\r'

{TAB} — タブ '\011'

{ESC} — エスケープ '\033'

{^A} — '\001'

...

{^_} — '\031'

マクロは展開後の文字数が最大 256 文字に制限されています。

34.49.74.4 Wgnuplot.ini Windows 版の **gnuplot** は Windows ディレクトリにある **WGNUPLOT.INI** の [**WGNUPLOT**] セクションからいくつかのオプションを読み込みます。**WGNUPLOT.INI** ファイルの例です:

```
[WGNUPLOT]
TextOrigin=0 0
TextSize=640 150
TextFont=Terminal,9
GraphOrigin=0 150
GraphSize=640 330
GraphFont=Arial,10
GraphColor=1
GraphToTop=1
GraphBackground=255 255 255
Border=0 0 0 0 0
Axis=192 192 192 2 2
Line1=0 0 255 0 0
Line2=0 255 0 0 1
Line3=255 0 0 0 2
Line4=255 0 255 0 3
Line5=0 0 128 0 4
```

GraphFont の項にはフォント名とポイント単位のフォントサイズを指定します。**Border**, **Axis**, **Line** には、赤の強度 (0–255)、緑の強度、青の強度、カラーの線種、白黒の線種の 5 つの数字を指定します。この線種 (**linestyle**) は、0=実線、1=鎖線、2=点線、3=一点鎖線、4=二点鎖線、となっています。上の例の **WGNUPLOT.INI** では、Line 2 はカラーモードでは緑の実線で、白黒モードでは鎖線になります。デフォルトの線幅は 1 ピクセルです。線種が負の値の場合、その絶対値がピクセル単位での線幅である実線を意味します。Line1 と **points** スタイルで使われる線種は 1 ピクセル幅の実線でなければなりません。

34.49.74.5 Windows3.0 Windows 3.1 の方が良いのですが、以下のような制限の元で Windows 3.0 上でも WGNUPLOT は動作します: **1.** COMMDLG.DLL と SHELL.DLL (Windows 3.1、または Borland C++ 3.1 にある) が windows ディレクトリにあること

2. Borland C++ 3.1 で作られた WGNUPLOT.HLP は Windows 3.1 フォーマットなので、それには Borland C++ 3.1 とともに提供される WINHELP.EXE を使う必要あり

3. リアルモードではメモリ不足のため動作しない

4. TrueType フォントはグラフウィンドウ内では利用できない

5. ドラッグ-ドロップは働かない

34.49.75 X11

gnuplot は X サーバを利用する **x11** 出力ドライバを提供します。この出力形式は、環境変数 **DISPLAY** がセットされているか、環境変数 **TERM** が **xterm** にセットされているか、またはコマンドラインオプションとして**-display** が使われていれば起動時に自動的に選択されます。

書式:

```
set terminal x11 {reset} {<n>}
```

複数のグラフ描画ウィンドウをサポートしています。**set terminal x11** <**n**> は番号 n のウィンドウに出力します。n>0 の場合、その番号はウィンドウタイトルとアイコン名に **gplt** <**n**> として付けられます。現在のウィンドウはカーソル記号の変化で区別できます (デフォルトカーソルから十字カーソルへ)。

gnuplot ドライバが別な出力ドライバに変更されても、描画ウィンドウは開いたままになります。描画ウィンドウは、そのウィンドウにカーソルを置いて文字 q を押すか、ウィンドウマネージャのメニューの **close** を選択すれば閉じることができます。**reset** を実行すれば全てのウィンドウを一度に閉じれます。それは実際にウィンドウを管理している子プロセスを終了します (もし **-persist** が指定されていなければ)。

描画ウィンドウは **-persisit** オプションが与えられていなければ、対話の終了時に自動的に閉じられます。

描画サイズとアスペクト比は、**gnuplot** のウィンドウをリサイズすることでも変更できます。

線の幅と点のサイズは **gnuplot** の **set linestyle** で変更可能です。

出力ドライバ **x11** に関しては、**gnuplot** は (起動時に)、コマンドライン、または設定ファイルから、geometry や font, name などの通常の X Toolkit オプションやリソースの指定を受け付けます。それらのオプションについては X(1) マニュアルページ (やそれと同等のもの) を参照してください。

他にも **x11** 出力形式用の多くの **gnuplot** のオプションがあります。これらは **gnuplot** を呼ぶときにコマンドラインオプションとして指定するか、または設定ファイル "~ /.Xdefaults" のリソースとして指定できます。これらは起動時に設定されるので、**gnuplot** 実行時には変更できません。

34.49.75.1 Command-line_options X Toolkit オプションに加え、以下のオプションが gnuplot の立ち上げ時のコマンドラインで、またはユーザのファイル ".Xdefaults" 内のリソースとして指定できます:

‘-mono‘	カラーディスプレイ上で強制的に白黒描画
‘-gray‘	グレイスケールまたはカラーディスプレイ上でのグレイスケール描画 (デフォルトではグレイスケールディスプレイは白黒描画を受け付ける)
‘-clear‘	新しい描画を表示する前に (瞬間的に) 画面を消去
‘-tvtwm‘	geometry オプションによる位置の指定を、仮想ルートウィンドウ中の現在の表示部分に対する相対的な位置にする
‘-raise‘	各描画後に描画ウィンドウを最前面へ出す
‘-noraise‘	各描画後に描画ウィンドウを最前面へ出すことはしない
‘-persist‘	gnuplot プログラム終了後も描画ウィンドウを残す

上記のオプション、はコマンドライン上での指定書式で、".Xdefaults" にリソースとして指定するときは異なる書式を使います。

例:

```
gnuplot*gray: on
```

gnuplot は描画スタイル **points** で描画する点のサイズの制御にも、コマンドラインオプション (**-pointsize <v>**) とリソース (**gnuplot*pointsize: <v>**) を提供しています。値 **v** は点のサイズの拡大率として使われる実数値 ($0 < v <= 10$) で、例えば **-pointsize 2** はデフォルトのサイズの 2 倍、**-pointsize 0.5** は普通のサイズの半分の点が使われます。

34.49.75.2 Monochrome_options 白黒ディスプレイに対しては **gnuplot** は描画色 (foreground) も背景色 (background) も与えません。デフォルトでは背景は白、描画は黒です。**-rv** や **gnuplot*reverseVideo: on** の場合には背景が黒で描画は白になります。

34.49.75.3 Color_resources カラーディスプレイに対しては、**gnuplot** は以下のリソース (ここではそのデフォルトの値を示します)、または白黒階調 (greyscale) のリソースを参照します。リソースの値はシステム上の X11 rgb.txt ファイルに書かれている色名、または 16 進の色指定 (X11 のマニュアルを参照) か、色名と強度 (0 から 1 の間の値) をコンマで区切った値を使用できます。例えば **blue, 0.5** は半分の強度の青、を意味します。

```
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
```

これらに関するコマンドラインの書式は、例えば以下の通りです。

例:

```
gnuplot -background coral
```

34.49.75.4 Grayscale_resources **-gray** を選択すると、**gnuplot** は、グレイスケールまたはカラーディスプレイに対して、以下のリソースを参照します (ここではそのデフォルトの値を示します)。デフォルトの背景色は黒であることに注意してください。

```
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
```

34.49.75.5 Line_resources **gnuplot** は描画の線の幅 (ピクセル単位) の設定のために以下のリソースを参照します (ここではそのデフォルトの値を示します)。0 または 1 は最小の線幅の 1 ピクセル幅を意味します。2 または 3 の値によってグラフの外観を改善できる場合もあるでしょう。

```
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 は線描画で使用する点線の形式の設定用に以下のリソースを参照します。0 は実線を意味します。2 桁の 10 進数 **jk** (**j** と **k** は 1 から 9 までの値) は、**j** 個のピクセルの描画に **k** 個の空白のピクセルが続くパターンの繰り返しからなる点線を意味します。例えば '16' は 1 個のピクセルの後に 6 つの空白が続くパターンの点線になります。さらに、4 桁の 10 進数でより詳細なピクセルと空白の列のパターンを指定できます。例えば、'4441' は 4 つのピクセル、4 つの空白、4 つのピクセル、1 つの空白のパターンを意味します。以下のデフォルトのリソース値は、白黒ディスプレイ、あるいはカラーや白黒階調 (grayscale) ディスプレイ上の白黒描画における値です。カラーディスプレイではそれらのデフォルトの値はほとんど 0 (実線) で、**axisDashes** のみがデフォルトで '16' の点線となっています。

```
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
```

34.49.76 Xlib

xlib ドライバは X11 Windows System をサポートしています。このドライバはコマンド gnuplot_x11 用の出力を行ないます。**set term x11** は、**set terminal xlib; set output "|gnuplot_x11"** とほぼ同様です。**xlib** にはオプションはありませんが、**x11** を参照してみてください。

34.50 Tics

コマンド **set tics** は目盛りの刻みを外向きに書かれるように変更するのに使われます。

書式:

```
set tics {<direction>}
show tics
```

ここで <direction> は **in** (デフォルト) または **out** です。

大目盛り (ラベルのつく) の他の制御に関しては **set xtics** を、小目盛りの制御に関しては **set mxtics** もそれぞれ参照してください。

34.51 Ticslevel

splot において、**set ticslevel** によって垂直軸 (Z) の相対的な高さを調整できます。数値引数を与えるとそれは xy 平面から見た目盛りの一番下の位置 (z の範囲に対する比) を指定したことになります。デフォルトの値は 0.5 です。負の値も許されていますが、そうすると 3 つの軸の目盛りの見出しが重なる可能性があります。

xy 平面を z 軸の 'pos' の位置に置くには、**ticslevel** の値を (pos - zmin) / (zmin - zmax) としてください。

書式:

```
set ticslevel {<level>}
show tics
```

set view も参照してください。

34.52 Ticscale

目盛りの刻みの大きさは **set ticscale** で調節できます。

書式:

```
set ticscale {<major> {<minor>}}
show tics
```

もし <minor> が指定されなければそれは 0.5*<major> となります。デフォルトの大目盛り (major tics) の刻みのサイズは 1.0 で、小目盛り (minor tics) は 0.5 です。負の値を指定することで、刻みを外側に向けることが可能であることに注意してください。

34.53 Timestamp

コマンド **set timestamp** は描画の日付と時刻を左の余白に表示します。

書式:

```
set timestamp {"<format>"} {top|bottom} {{no}rotate}
              {<xoff>}{,<yoff>} {"<font>"}
set notimestamp
show timestamp
```

書式文字列 (format) を使って、書かれる日付と時刻の書式を選択することができます。デフォルトは asctime() が使用する "%a %b %d %H:%M:%S %Y" です (曜日、月名、日、時、分、秒、4 桁の西暦)。**top** と **bottom** を使って日付を左の余白の上に配置するか、下に配置するかを選択できます (デフォルトは下)。**rotate** は、もし出力形式がサポートしていればですが、日付を垂直方向の文字列にします。定数 <xoff>、<yoff> は文字スクリーン座標系での、デフォルトの位置からのずれを表します。<font> は日付が書かれるフォントを指定します。

timestamp の代わりに省略名 **time** を使っても構いません。

例:

```
set timestamp "%d/%m/%y %H:%M" 80,-2 "Helvetica"
```

日付の書式文字列に関する詳しい情報については **set timefmt** を参照してください。

34.54 Timefmt

このコマンドは、データが日時の形式になっている場合に、その時系列データに適用されます。これはコマンド **set xdata time** も与えられていないと意味がありません。

書式:

```
set timefmt "<format string>"
show timefmt
```

文字列引数 (<format string>) は **gnuplot** に日時データをデータファイルからどのように読むかを指示します。有効な書式は以下の通りです:

Time Series timedata Format Specifiers	
書式	説明
%d	何日, 1–31
%m	何月, 1–12
%y	何年, 0–99
%Y	何年, 4 桁
%j	1 年の何日目, 1–365
%H	何時, 0–24
%M	何分, 0–60
%S	何秒, 0–60
%b	月名 (英語) の 3 文字省略形
%B	月名 (英語)

任意の文字を文字列中で使用できますが、規則に従っている必要があります。\t (タブ) は認識されます。バックスラッシュ + 8 進数列 (\nnn) はそれが示す文字に変換されます。日時要素の中に分離文字がない場合、%d, %m, %y, %H, %M, %S はそれぞれ 2 桁の数字を読み込み、%Y は 4 桁、%j は 3 桁の数字を読み込みます。%b は 3 文字を、%B は必要な分だけの文字を要求します。

空白 (スペース) の扱いはやや違います。書式文字列中の 1 つの空白は、ファイル中の 0 個、あるいは 1 つ以上の空白文字列を表します。すなわち、"%H %M" は "1220" や "12 20" を "12 20" と同じように読みます。

データ中の非空白文字の集まりそれぞれは、**using n:n** 指定の一つ一つの列とカウントされます。よって **11:11 25/12/76 21.0** は 3 列のデータと認識されます。混乱を避けるために、日時データが含まれる場合 **gnuplot** は、あなたの **using** 指定が完璧なものであると仮定します。

gnuplot は数字でない文字列を読めないので、日付データが曜日、月の名前を含んでいる場合、書式文字列でそれを排除しなければいけません。しかし、"%a", "%A", "%b", "%B" でそれらを表示することはできます: これら、及び日時データの出力の他のオプションの詳細に関しては **set format** を参照してください (**gnuplot** は数値から月や曜日を正しく求めます)。

他の情報については **set xdata** と **Time/date** の項も参照してください。

例:

```
set timefmt "%d/%m/%Y\t%H:%M"
```

は、**gnuplot** に日付と時間がタブで分離していることを教えます (ただし、あなたのデータをよーく見てください。タブだったものがどこかで複数のスペースに変換されていませんか ? 書式文字列はファイル中に実際にある物と一致していなければなりません)。

34.55 Title

コマンド **set title** は、描画の上の真中に書かれる描画タイトルを生成します。**set title** は **set label** の特殊なもの、とみなせます。

書式:

```
set title {"<title-text>"} {<xoff>}{,<yoff>} {"<font>,{<size>}"}
show title
```

定数 <xoff>, <yoff> を指定することで、タイトルを <xoff>, <yoff> 文字スクリーン座標だけ動かすことができます (グラフ座標ではありません)。例えば **"set title ,-1"** は、タイトルの y 方向の位置のみ変更し、大ざっぱに言って 1 文字分の高さだけタイトルを下に下げます。

<font> はタイトルが書かれるフォントを指定するのに使われます。<size> の単位は、どの出力形式 (terminal) を使っているかによって変わります。

set title をパラメータなしで使うとタイトルを消去します。

バックスラッシュ文字列の作用、及び文字列を囲む単一引用符と二重引用符の違いについては **syntax** を参照してください。

34.56 Tmargin

コマンド **set tmargin** は上の余白のサイズをセットします。詳細は **set margin** を参照してください。

34.57 Trange

コマンド **set trange** は、媒介変数モード、あるいは極座標モードでの x, y の値を計算するのに使われる媒介変数の範囲を設定します。詳細は **set xrange** を参照してください。

34.58 Urange

set urange と **set vrange** は、**splot** の媒介変数モードで x, y, z の値を計算するのに使われる媒介変数の範囲を設定します。詳細は **set xrange** を参照してください。

34.59 Variables

show variables コマンドは全てのユーザ定義変数とその値の一覧を表示します .

書式:

```
show variables
```

34.60 Version

コマンド **show version** は現在起動している gnuplot のバージョン、最終修正日、著作権者と、FAQ や info-gnuplot メーリングリスト、バグレポート先のメールアドレスを表示します。対話的にプログラムが呼ばれているときはスクリーン上にその情報を表示します。

書式:

```
show version {long}
```

long オプションを与えると、さらにオペレーティングシステム、**gnuplot** インストール時のコンパイルオプション、ヘルプファイルの置き場所、そして (再び) 有用なメールアドレスを表示します。

34.61 View

コマンド **set view** は **splot** の視線の角度を設定します。これは、グラフ描画の 3 次元座標をどのように 2 次元の画面 (screen) に投影するかを制御します。これは、描画されたデータの回転と拡大縮小の制御を与えてくれますが、正射影しかサポートしていません。

書式:

```
set view <rot_x> {,{<rot_z>}{,{<scale>}{,<scale_z>}}}
show view
```

ここで <rot_x> と <rot_z> は、画面に投影される仮想的な 3 次元座標系の回転角 (単位は度) の制御で、最初は (すなわち回転が行なわれる前は) 画面内の水平軸は x, 画面内の垂直軸は y, 画面自身に垂直な軸が z となっています。最初は x 軸の周りに <rot_x> だけ回転されます。次に、新しい z 軸の周りに <rot_z> だけ回転されます。

<rot_x> は [0:180] の範囲に制限されていて、デフォルトでは 60 度です。<rot_z> は [0:360] の範囲に制限されていて、デフォルトでは 30 度です。<scale> は **splot** 全体の拡大縮小率を制御し、<scale_z> は z 軸の拡大縮小のみを行ないます。スケールのデフォルトはどちらも 1.0 です。

例:

```
set view 60, 30, 1, 1
set view ,,0.5
```

最初の例は 4 つの全てをデフォルトの値にしています。2 つめの例は縮小率のみを 0.5 に変更しています。

set ticslevel も参照してください。

34.62 Vrange

コマンド **set urange** と **set vrange** は、**splot** の媒介変数 (パラメータ) モードで x, y, z の値を計算するのに使われる媒介変数の範囲を設定します。 詳細は **set xrange** を参照してください。

34.63 X2data

コマンド **set x2data** は x2 (上) 軸のデータを時系列 (日時) 形式に設定します。詳細は **set xdata** を参照してください。

34.64 X2dtics

コマンド **set x2dtics** は x2 (上) 軸の目盛りを曜日に変更します。詳細は **set xdtics** を参照してください。

34.65 X2label

コマンド **set x2label** は x2 (上) 軸の見出しを設定します。詳細は **set xlabel** を参照してください。

34.66 X2mtics

コマンド **set x2mtics** は、x2 (上) 軸を 1 年の各月に設定します。詳細は **set xmtics** を参照してください。

34.67 X2range

コマンド **set x2range** は x2 (上) 軸の表示される水平範囲を設定します。詳細は **set xrange** を参照してください。

34.68 X2tics

コマンド **set x2tics** は x2 (上) 軸の、見出し付けされる大目盛りの制御を行ないます。詳細は **set xtics** を参照してください。

34.69 X2zeroaxis

コマンド **set x2zeroaxis** は、原点を通る x2 (上) 軸 (y2 = 0) を描きます。詳細は **set zeroaxis** を参照してください。

34.70 Xdata

このコマンドは x 軸のデータ形式を日時データにセットします。同様のコマンドが他の軸それぞれに用意されています。

書式:

```
set xdata {time}
show xdata
```

ydata, **zdata**, **x2data**, **y2data** にも同じ書式が当てはまります。

time オプションはデータが日時データであることを伝えます。オプションをつけない場合、データ型は通常のものに戻ります。

gnuplot にどのように日時データを読みこませるかについては、**set timefmt** を参照してください。日時データは今世紀の始まり (訳注: 厳密には 2000 年 1 月 1 日の始まり) からの秒数に変換されます。時間書式 (timefmt) は現在はただ一つだけしか使えません。それは、全ての日時データ項目がこの書式に一致しなければならないことを意味します。また、範囲の指定は、日時指定が数式と解釈されるのを避けるために、その書式に従った文字列を引用符で囲んで指定すべきです。

目盛り刻みの見出し (label) を表示するのには関数 'strftime' (unix でそれを調べるには "man strftime" とタイプしてください) が使われます。**set format x "string"** で、10 進数の書式ではなさそうなもの (2 つ以上の '%'、または %f でも %g でもないもの) が与えられていなければ、**gnuplot** はこれを適当に意味のある書式で計算して表示します。

他の情報については **Time/date** も参照してください。

34.71 Xdtics

コマンド **set xdtics** は x 軸の目盛りの刻みを曜日に変換します (0=Sun, 6=Sat)。6 を越える場合は 7 による余りが使われます。**set noxdtics** はその見出しをデフォルトの形式に戻します。他の軸にも同じことを行なう同様のコマンドが用意されています。

書式:

```
set xdtics
set noxdtics
show xdtics
```

ydtics, **zdtics**, **x2dtics**, **y2dtics** にも同じ書式が当てはまります。

set format コマンドも参照してください。

34.72 Xlabel

コマンド **set xlabel** は x 軸の見出しを設定します。他の軸にも見出しを設定する同様のコマンドがあります。

書式:

```
set xlabel {"<label>"} {<xoff>}{,<yoff>} {"<font>{,<size>}"}
show xlabel
```

同じ書式が **x2label**, **ylabel**, **y2label**, **zlabel** にも適用されます。

見出しの追加のずれを表す定数 <xoff>, <yoff> を指定すると、見出しを <xoff> の文字幅、または <yoff> の文字の高さ分だけずらします。例えば**"set set xlabel -1"** は xlabel の x の位置だけ変更し、大ざっぱに言って 1 文字分の文字幅だけ左に移動します。文字のサイズは、フォントと使用する出力形式 (terminal) に依存します。

<font> は見出しが書かれるフォントを指定するのに使われます。フォントの <size> (大きさ) の単位は、どんな出力形式を使うかに依存します。

見出しを消去するには、オプションをつけずに実行します。例: **"set y2label"**

軸の見出しのデフォルトの位置は以下の通りです:

xlabel: x 軸の見出しは下の軸の下の真中

ylabel: y 軸の見出しは出力形式依存で、以下の 3 つのいずれか:

1. 水平方向の文字列で描画の左上に左端に合わせて配置されます。文字列の回転を行なえない出力形式では多分これが選択されます。**set x2tics** が同時に使われている場合、ylabel は x2 軸の見出しの左端と重なるかも知れません。これは ylabel の位置か左の余白を調整することで対処できるでしょう。

2. 垂直方向の文字列で、描画の左で垂直方向に中央揃えされます。文字列を回転できる出力形式では多分これが選択されます。

3. 水平方向の文字列で、描画の左で垂直方向に中央揃えされます。EEPIC, LaTeX, TPIC ドライバではこれが選択されます。ylabel が描画に上書きするのを避けるには、\\ を使ってユーザが改行を入れる必要があります。垂直方向に並んだ文字列を作りたいなら、全ての文字の間に \\ を入れてください (しかしこれは綺麗ではありません)。

zlabel: z 軸の見出しは軸の表示範囲より上で、見出しの真中が z 軸の真上

y2label: y2 軸の見出しは y2 軸の右。その位置は、出力形式依存で y 軸と同様の規則で決定。

x2label: x2 軸の見出しは上の軸の上で、描画タイトルよりは下。これは、改行文字を使えば、それによる複数の行からなる描画タイトルで x2 軸の見出しを生成することも可能。例:

```
set title "This is the title\n\nThis is the x2label"
```

これは二重引用符を使うべきであることに注意してください。この場合、もちろん 2 つの行で同じフォントが使われます。

もし軸の位置のデフォルトの位置が気に入らないならば、代わりに **set label** を使ってください。このコマンドは文字列をどこに配置するかをもっと自由に制御できます。

バックスラッシュ文字列の作用、及び文字列を囲む単一引用符と二重引用符の違いに関するより詳しい情報については **syntax** を参照してください。

34.73 Xmtics

コマンド **set xmtics** は x 軸の目盛りの見出しを月に変換します。1=Jan (1 月)、12=Dec (12 月) となります。12 を越えた数字は、12 で割ったあまりの月に変換されます。**set noxmtics** で目盛りはデフォルトの見出しに戻ります。他の軸に対しても同じ役割をする同様のコマンドが用意されています。

書式:

```
set xmtics
set noxmtics
show xmtics
```

x2mtics, **ymtics**, **y2mtics**, **zmtics** にも同じ書式が適用されます。

コマンド **set format** も参照してください。

34.74 Xrange

コマンド **set xrange** は表示される水平方向の範囲を指定します。他の軸にも同様のコマンドが存在しますし、極座標での半径 r, 媒介変数 t, u, v にも存在します。

書式:

```
set xrange { [{{<min>}:{<max>}}] {{no}reverse} {{no}writeback} }
           | restore
show xrange
```

ここで <min> と <max> には定数、数式、または '*' で、'*' は自動縮尺機能を意味します。日時データの場合、範囲は **set timefmt** の書式に従った文字列を引用符で囲む必要があります。省略された値は変更されません。

yrange, **zrange**, **x2range**, **y2range**, **rrange**, **trange**, **urange** **vrange** は同じ書式を使用します。

オプション **reverse** は軸の方向を逆にします。例えば **set xrange [0:1] reverse** は、1 が左、0 が右であるような軸にします。これは、もちろん **set xrange [1:0]** と同じですが、**reverse** は主に自動縮尺 (**autoscale**) で用いられることを意図しています。

オプション **writeback** は、**set xrange** で占められているバッファの中に自動縮尺機能により作られた範囲を保存します。これは、いくつかの関数を同時に表示し、しかしその範囲はそのうちのいくつかのものから決定させたい場合に便利です。**writeback** の作用は、**plot** の実行中に機能するので、そのコマンドの前に指定する必要があります。最後に保存した水平方向の範囲は **set xrange restore** で復元できます。例を上げます。

```
set xrange [-10:10]
set yrange [] writeback
plot sin(x)
set yrange restore
replot x/2
```

この場合、y の範囲 (yrange) は sin(x) の値域として作られた [-1:1] の方になり、x/2 の値域 [-5:5] は無視されます。上記のそれぞれのコマンドの後に **show yrange** を実行すれば、上で何が行なわれているかを理解する助けになるでしょう。

2 次元描画において、**xrange** と **yrange** は軸の範囲を決定し、**trange** は、媒介変数モードの媒介変数の範囲、あるいは極座標モードの角度の範囲を決定します。同様に 3 次元媒介変数モードでは、**xrange**, **yrange**, **zrange** が軸の範囲を管理し、**urange** と **yrange** が媒介変数の範囲を管理します。

極座標モードでは、**rrange** は描画される半径の範囲を決定します。<rmin> は半径への追加の定数として作用し、一方 <rmax> は半径を切り捨てる (clip) ように作用し、<rmax> を越えた半径に対する点は描画されません。**xrange** と **yrange** は影響されます。これらの範囲は、グラフが r(t)-rmin のグラフで、目盛りの見出しにはそれぞれ rmin を加えたようなものであるかのようにセットされます。

全ての範囲は部分的に、または全体的に自動縮尺されますが、データの描画でなければ、パラメータ変数の自動縮尺機能は意味がないでしょう。

範囲は **plot** のコマンドライン上でも指定できます。コマンドライン上で与えられた範囲は単にその **plot** コマンドでだけ使われ、**set** コマンドで設定された範囲はその後の描画で、コマンドラインで範囲を指定していないもの全てで使われます。これは **splot** も同じです。

例:

x の範囲をデフォルトの値にします:

```
set xrange [-10:10]
```

y の範囲が下方へ増加するようにします:

```
set yrange [10:-10]
```

z の最小値には影響を与えずに (自動縮尺されたまま)、最大値のみ 10 に設定します:

```
set zrange [:10]
```

x の最小値は自動縮尺とし、最大値は変更しません:

```
set xrange [*:]
```

34.75 Xtics

x 軸の (見出しのつく) 大目盛りは コマンド **set xtics** で制御できます。目盛りは **set noxtics** で消え、**set xtics** で (デフォルトの状態の) 目盛りがつきます。y,z,x2,y2 軸の大目盛りの制御を行なう同様のコマンドがあります。

書式:

```
set xtics {axis | border} {{no}mirror} {{no}rotate}
          {  autofreq
           | <incr>
           | <start>, <incr> {,<end>}
           | ({"<label>"} <pos> {,{"<label>"} <pos>}...) }
set noxtics
show xtics
```

同じ書式が **ytics**, **ztics**, **x2tics**, **y2tics** にも適用されます。

axis と **border** は **gnuplot** に目盛り (目盛りの刻自身とその見出し) を、それぞれ軸につけるのか、境界につけるのかを指示します。軸が境界にとても近い場合、**axis** を使用すると目盛りの見出し文字が余白に書かれている他の文字に重ってしまう可能性があります。

mirror は **gnuplot** に反対側の境界の同じ位置に、見出しのない目盛りを出力するよう指示します。**nomirror** は、あなたが思っている通りのことを行ないます。

rotate は、文字列を 90 度回転させて出力させようとします。これは、文字列の回転をサポートしている出力ドライバ (terminal) では実行されます。**norotate** はこれをキャンセルします。

x と y 軸の大目盛りのデフォルトは **border mirror norotate** で、x2, y2 軸は **border nomirror norotate** がデフォルトです。z 軸には、**{axis | border}** オプションは無効で、デフォルトは **nomirror** です。z 軸の目盛りをミラー化したいなら、多分 **set border** でそのための空間をあける必要があるでしょう。

オプションなしで **set xtics** を実行すると、目盛りが表示される状態であれば、それはデフォルトの境界、または軸を復元し、そうでなければ何もしません。その前に指定した目盛りの間隔、位置 (と見出し) は保持されます。

目盛りの位置は、デフォルト、またはオプション **autofreq** が指定されていれば自動的に計算されます。そうでなければ、次の 2 つの形式で指定されます:

暗示的な <start>, <incr>, <end> 形式は、目盛りの列を <start> から <end> の間を <incr> の間隔で表示します。<end> を指定しなければ、それは無限大とみなされます。<incr> は負の値も可能です。<start> と <end> の両方が指定されていない場合、<start> は -∞、<end> は +∞ とみなされ、目盛りは <incr> の整数倍の位置に表示されます。軸が対数軸の場合、目盛りの間隔 (増分) は、積因子として使用されます。

例:

目盛りを 0, 0.5, 1, 1.5, ..., 9.5, 10 の位置に生成

```
set xtics 0,.5,10
```

目盛りを ..., -10, -5, 0, 5, 10, ... に生成

```
set xtics 5
```

目盛りを 1, 100, 1e4, 1e6, 1e8 に生成

```
set logscale x; set xtics 1,100,1e8
```

明示的な ("<label>" <pos>, ...) の形式は、任意の目盛りの位置、あるいは数字でない見出しの生成も可能にします。目盛りの一揃いは、位置とその見出しからなる組の集合です。見出しは二重引用符で囲まれた文字列であることに注意してください。それは、"hello" のような固定文字列でも構いませんし、"%3f clients" のようにその位置を数字に変換する書式文字列を含んでも構いませんし、空文字列 "" でも構いません。より詳しい情報については **set format** を参照してください。もし、文字列が与えられなければ、デフォルトの数字の見出しが使用されます。この形式では、目盛りは位置の数字の順に与える必要はありません。

例:

```
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)
```

2 番目の例では、全ての目盛りが見出し付けされます。3 番目の例では、端のものだけが見出し付けされます。

しかし指定しても、表示されるのはあくまで描画範囲のものだけです。

目盛りの見出しの書式 (または省略) は **set format** で制御されます。ただしそれは **set xtics** (<label>) の形式の明示的な見出し文字列が含まれていない場合だけです。

(見出し付けされない) 小目盛りは **set mxtics** コマンドで追加することが出来ます。

時系列データの場合、位置の値は **timefmt** の書式にしたがった日付、または時刻を引用符で囲んで与えなければいけません。<start>, <incr>, <end> 形式を使う場合、<start> と <end> は **timefmt** に従って与えますが、<incr> は秒単位で与える必要があります。その時刻は実際には **set format** で与えた書式に従って表示されます。

例:

```
set xdata time
set timefmt "%d/%m"
set format x "%b %d"
set xrange ["01/12":"06/12"]
set xtics "01/12", 172800, "05/12"

set xdata time
set timefmt "%d/%m"
set format x "%b %d"
set xrange ["01/12":"06/12"]
set xtics ("01/12", "" "03/12", "05/12")
```

これらは両方とも "Dec 1", "Dec 3", "Dec 5", の目盛りを生成しますが、2 番目の例 "Dec 3" の目盛りは見出し付けされません。

34.76 Xzeroaxis

コマンド **set xzeroaxis** は y = 0 の直線を描きます。詳細に関しては、**set zeroaxis** を参照してください。

34.77 Y2data

コマンド **set y2data** は y2 (右) 軸のデータを時系列 (日時) 形式に設定します。詳細は **set xdata** を参照してください。

34.78 Y2dtics

コマンド **set y2dtics** は y2 (右) 軸の目盛りを曜日に変更します。詳細は **set xdtics** を参照してください。

34.79 Y2label

コマンド **set y2label** は y2 (右) 軸の見出しを設定します。詳細は **set xlabel** を参照してください。

34.80 Y2mtics

コマンド **set y2mtics** は y2 (右) 軸の目盛りを 1 年の各月に変更します。詳細は **set xmtics** を参照してください。

34.81 Y2range

コマンド **set y2range** は y2 (右) 軸の表示される垂直範囲を設定します。詳細は **set xrange** を参照してください。

34.82 Y2tics

コマンド **set y2tics** は y2 (右) 軸の、見出し付けされる大目盛りの制御を行ないます。詳細は **set xtics** を参照してください。

34.83 Y2zeroaxis

コマンド **set y2zeroaxis** は、原点を通る y2 (右) 軸 (x2 = 0) を描きます。詳細は **set zeroaxis** を参照してください。

34.84 Ydata

コマンド **set ydata** は y 軸のデータを時系列 (日時) 形式に設定します。**set xdata** を参照してください。

34.85 Ydtics

コマンド **set ydtics** は y 軸の目盛りを曜日に変更します。詳細は **set xdtics** を参照してください。

34.86 Ylabel

このコマンドは y 軸の見出しを設定します。**set xlabel** を参照してください。

34.87 Ymtics

コマンド **set ymtics** は、y 軸の目盛りを月に変更します。詳細は **set xmtics** を参照してください。

34.88 Yrange

コマンド **set yrange** は、y 方向の垂直範囲を設定します。詳細は **set xrange** を参照してください。

34.89 Ytics

コマンド **set ytics** は y 軸の (見出し付けされる) 大目盛りを制御します。詳細は **set xtics** を参照してください。

34.90 Yzeroaxis

コマンド **set yzeroaxis** は $x = 0$ の直線 (y 軸) を書きます。詳細は **set zeroaxis** を参照してください。

34.91 Zdata

コマンド **set zdata** は z 軸のデータを時系列 (日時) 形式に設定します。**set xdata** を参照してください。

34.92 Zdtics

コマンド **set zdtics** は z 軸の目盛りを曜日に変更します。詳細は **set xdtics** を参照してください。

34.93 Zero

zero の値は、0.0 に近いデフォルトの閾値を表します。

書式:

```
set zero <expression>
show zero
```

gnuplot は、(複素数値を持つ点の描画においては) その値の虚数部分の絶対値が **zero** 閾値より大きい場合 (つまり実数でない値を持つ点) は、その点を描画しません。この閾値は **gnuplot** の他の様々な部分においてその (大まかな) 数値誤差の閾値としても使われています。デフォルトの **zero** の値は 1e-8 です。1e-3 (= 典型的なビットマップディスプレイの解像度の逆数) より大きい **zero** の値は設定すべきではないでしょうが、**zero** を 0.0 と設定するのは意味のないことではありません。

34.94 Zeroaxis

x 軸は **set xzeroaxis** によって描かれ、**set noxzeroaxis** によって削除されます。同様の y, x2, y2 軸用のコマンドが同様の働きをします。

書式:

```
set {x|x2|y|y2|}zeroaxis { {linestyle | ls <line_style>}
                           | { linetype | lt <line_type>}
                             { linewidth | lw <line_width>}}
set no{x|x2|y|y2|}zeroaxis
show {x|y|}zeroaxis
```

デフォルトでは、これらのオプションはオフになっています。選択された 0 の軸は <line_type> の線の型と <line_width> の線の幅 (現在使用している出力形式がサポートしていれば) で、あるいはあらかじめ定義された <line_style> のスタイルで描かれます。

線の型を指定しなければ、軸は通常の軸の線の型 (型 0) で描かれます。

set zeroaxis l は **set xzeroaxis l; set yzeroaxis l** と同等で、**set nozeroaxis** は **set noxzeroaxis; set noyzeroaxis** と同等です。

34.95 Zlabel

このコマンドは z 軸の見出しを設定します。**set xlabel** を参照してください。

34.96 Zmtics

コマンド **set zmtics** は z 軸の目盛りを月に変更します。詳細は **set xmtics** を参照してください。

34.97 Zrange

コマンド **set zrange** は z 軸方向に表示される範囲を設定します。このコマンドは **splot** にのみ有効で、**plot** では無視されます。詳細は **set xrange** を参照してください。

34.98 Ztics

コマンド **set ztics** は z 軸の (見出し付けされる) 大目盛りを制御します。詳細は **set ztics** を参照してください。

35 Shell

shell コマンドは対話的なシェルを起動します . **gnuplot** に戻るには , VMS では **logout** を , Unix ならば **exit** もしくは END-OF-FILE 文字を , AmigaOS では **endcli** を , MS-DOS か OS/2 ならば **exit** を入力して下さい .

単一のシェルコマンドならば , コマンドラインの行頭に ! の文字 (VMS では$) をつけることによっても実現できます . この場合コマンドが終了するとすぐに制御は **gnuplot** に戻ってきます . 例えば VMS, AmigaOS, MS-DOS, OS/2 では ,

```
! dir
```

とするとディレクトリの一覧を表示して **gnuplot** に戻ってきます .

Atari では , ! コマンドは , 最初にシェルが既にロードされているか調べて , 有効ならばそれを使います . 例えば , **gnuplot** が **gulam** から起動されている場合に , これは実用的です .

36 Splot

splot は 3 次元描画のためのコマンドです (もちろんご存知でしょうが、実際にはその 2 次元への射影です)。それは関数、またはデータファイルから、**plot** コマンドととても良く似た方法でその描画を作ります。

plot コマンドと共通の仕様については **plot** を参照して下さい。ここではそれと異なるものだけ詳細に取り上げます。なお、**binary** や **matrix** オプションは ("datafile-modifiers" 以下で取り上げます) **plot** にはないことに特に注意して下さい。

書式:

```
splot {<ranges>}
      <function> | "<datafile>" {datafile-modifiers}}
      {<title-spec>} {with <style>}
      {, {definitions,} <function> ...}
```

ここで、関数 <function>、またはクォートでくくられたデータファイル名のどちらかが必要です。関数は、一本の数式、あるいは媒介変数モードでは 3 つの数式の組です。

デフォルトでは、splot は描画されるデータの下に完全な xy 面を描きます。z の一番下の目盛りと xy 平面の位置関係は set ticslevel で変更できます。splot の射影の向きは set view で制御できます。詳細は set view, set ticslevel を参照して下さい。

splot コマンドの範囲の指定の書式は plot の場合と同じです。媒介変数モードでなければ範囲は xrange, yrange, zrange の順であり、媒介変数モードでは urange, vrange, xrange, yrange, zrange の順です。

title オプションも plot と同じです。with も plot とほぼ同じですが、splot では利用可能な描画スタイルは lines, points, linespoints, dots, impulses に限られています。plot で使えるエラーバーの機能は splot にはありません。

データファイルのオプションはさらに違いがあります。

36.1 Data-file

plot と同じように、ファイルに含まれる離散的なデータは、そのファイル名をクォートで囲んで指定することで描画できます。

書式:

```
splot '<file_name>' {binary | matrix}
                    {index <index list>}
                    {every <every list>}
                    {using <using list>}
```

"" や "-" といった特別なファイル名も plot のときと同様に許されます。

手短にいうと、binary や matrix はそのデータが特別な形であることを、index は多重データ集合ファイルからどのデータ集合を選んで描画するかを、every は各データ集合からどのデータ行 (部分集合) を選んで描画するかを、using は各データ行からどのように列を選ぶかを指定します。

index と every オプションは plot の場合と同じように振舞います。using も、using のリストが 2 つでなく 3 つ必要であるということを除いては同様です。

thru や smooth といった plot のオプションは splot では利用できません。しかし、cntrparams や dgrid3d が、制限されてはいますが平滑化のために用意されています。

データファイルの形式は、各点が (x,y,z) の 3 つ組である以外は、本質的に plot と同じです。もし一つの値だけが与えられれば、それは z として使われ、データブロック番号が y として、そして x はそのデータブロック内での番号が使われます。もし 2 つの値が与えられれば、gnuplot はエラーメッセージを出します。3 つの値は (x,y,z) の組と見なされます。他に値があれば、それは一般に誤差と見なされます。それは fit で使うことが可能です。

splot のデータファイルでは、1 行の空行はデータブロックの分離子です。splot は個々のデータブロックを、関数の y-孤立線と同じものとして扱います。1 行の空行で分離されている点同士は線分で結ばれることはありません。全てのデータブロックが全く同じ点の数を持つ場合、gnuplot はデータブロックを横断し、対応する点同士を結ぶ孤立線を描きます。これは "grid data" と呼ばれ、曲面の描画、等高線の描画 (set contour)、隠線処理 (set hidden3d) では、この形のデータであることが必要となります。splot grid data も参照して下さい。

3 列の splot データにおいては、媒介変数モード (parametric) を指定することはもはや不要です。

36.1.1 Binary

splot はある特別なフォーマットで書かれたバイナリファイルを (そしてバイナリファイルの表現に互換性を持つシステムの上で) 読むことができます。

以前のバージョンでは、**gnuplot** は動的にバイナリデータかどうかを判断していましたが、現在は、ファイル名の後ろに **binary** キーワードを直接指定することが必要です。

単精度浮動小数の数値が次のように保存されています:

```
<N+1>  <y0>   <y1>   <y2>  ...  <yN>
 <x0> <z0,0> <z0,1> <z0,2> ... <z0,N>
 <x1> <z1,0> <z1,1> <z1,2> ... <z1,N>
  :       :       :       :   ...    :
```

これらは以下のような 3 つの数字の組に変換されます:

```
<x0> <y0> <z0,0>
<x0> <y1> <z0,1>
<x0> <y2> <z0,2>
 :    :     :
<x0> <yN> <z0,N>

<x1> <y0> <z1,0>
<x1> <y1> <z1,1>
 :    :     :
```

そして、これらの 3 つの数字の組は **gnuplot** の孤立線に変換され、その後 **gnuplot** が通常の方法で描画の残りを行います。

行列やベクトルの操作のサブルーチン (C による) が **binary.c** に用意されています。バイナリデータを書くルーチンは

```
int fwrite_matrix(file,m,nrl,nrl,ncl,nch,row_title,column_title)
```

です。これらのサブルーチンを使う例が **bf_test.c** として用意されていて、これはデモファイル **demo/binary.dem** 用に複数のバイナリファイルを生成します。

index キーワードは、ファイルフォーマットが 1 つのファイルにつき 1 つの曲面しか許さないため、サポートされません。**every** や **using** フィルタはサポートされます。**using** は、データがあたかも上の 3 つ組の形で読まれたかのように働きます。

36.1.2 Example datafile

以下は 3 次元データファイルの描画の単純な一つの例です。

```
splot 'datafile.dat'
```

ここで、"datafile.dat" は以下を含むとします:

```
# 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
```

この "datafile.dat" は 4*3 の格子 (それぞれ 3 点からなるブロックの 4 つの行) を定義することに注意して下さい。行 (データブロック) は 1 行の空行で区切られます。

x の値はそれぞれのデータブロックの中で定数になっていることに注意して下さい。もし y を定数の値とし、隠線処理が有効な状態で描画すると、その曲面は裏返しで書かれることになります。

格子状データ (grid data) に対して、個々のデータブロック内で x の値を定数としておく必要はありませんし、同じ場所の y の値を同じ値に揃えておく必要もありません。**gnuplot** は個々のデータブロック内の点の数が等しいということを必要としているだけです。

しかし、等高線を導くのに用いられる曲面の網目は、対応する点を列的に選んで結ぶため、不揃いの格子データに対する曲面の描画への影響は予想できません。それはケースバイケースの原理でテストすべきでしょう。

36.1.3 Matrix

matrix 指定子は ASCII データが配列形式で保存されていることを指示します。各ブロックの z の値は一行で一度に読まれます。すなわち、

```
z11 z12 z13 z14 ...
z21 z22 z23 z24 ...
z31 z32 z33 z34 ...
```

など。その行や列の番号は、x や y の値として使われます。

36.2 Grid_data

3 次元描画のためのルーチンは、個々の網目の格子においては一つの標本点と一つのデータ点がある、という形の格子状データ用に設計されています。各データ点は、関数の値を評価すること (**set isosample** 参照)、またはデータファイルを読み込むこと (**splot datafile** 参照) によって生成されます。"孤立線" と

いう言葉は関数に対しても、データに対してもその網目の線を表すものとして用いられます。網目は、必ずしも x, y に関する長方形でなくてもよく、u,v で媒介変数表示されても構わないことに注意して下さい。**set isosamples** を参照して下さい。

しかし、**gnuplot** はそのような形式を必ずしも必要とはしません。例えば関数の場合は、**samples** は **isosamples** と違っていても構いません。すなわち、x-孤立線のうち、1 本の y-孤立線と交わないものがいくつかあることがあります。データファイルの場合は、個々のデータブロックのばらついた点の個数が全て同じであれば、"孤立線は" はデータブロックの点を結び、"横断孤立線" は各データブロックの対応する点同士を結び、"曲面" を作ろうとします。どちらの場合でも、等高線、および隠線処理モードは点が意図したフォーマットであった場合とは違った描画を与えることになります。ばらつきのあるデータは **set dgrid3d** によって { 異なる } 格子状データに変換することができます。

等高線に関するコードは、y-孤立線の点と、それに対応する隣の y-孤立線上の点の間の線分に沿っての z の張力を計測します。よって、x-孤立線に、y-孤立線との交点とはならないような標本点があるような曲面に対しては、**splot** の等高線はそのような標本点を無視することになります。以下を試してみて下さい:

```
set xrange [-pi/2:pi/2]; set yrange [-pi/2:pi/2]
set function style 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
```

36.3 Splot_overview

splot は点の集まりとして、あるいは、それらの点を結ぶことによって曲面を表示することができます。**plot** と同様に、点はデータファイルから読むこともできますし、指定された区間で関数の値を評価して得ることもできます。**set isosamples** を参照して下さい。曲面は、各点を線分で結ぶことで近似的に作られます。**set surface** を参照して下さい。そしてその場合曲面は **set hidden3d** で不透明にもできます。3 次元曲面を眺める向きは、**set view** で変更できます。

さらに、格子上のデータ点に対しては、**splot** は同じ高さを持つ点を補間することができ (**set contour** 参照)、そしてそれらを結んで等高線を描くことができます。さらに、その結び方には真直な線分や滑らかな線を使うことができます (**set cntrparams** 参照)。関数は、常に **set isosamples** と **set samples** で決定される格子状データとして評価されます。一方、ファイルのデータは、**data-file** に書かれているような格子状データフォーマットにするか、あるいは格子データを生成する (**set dgrid3d** 参照) ということをしなければそうはなりません。

等高線は曲面の上に表示することもできるし、底面に射影することもできる。底面への射影は、ファイルに書き出すこともでき、そしてそれを **plot** で再び読み込んで **plot** のより強い整形能力を生かすこともできる。

37 Test

test は使用している出力形式で使える線の種類、点の種類、または有用なその他の描画を生成します。

書式:

```
test
```

38 Update

このコマンドは当てはめ (fit) のパラメータの現在の値を、初期値のファイルの形式で (**fit** の項で説明されている)、与えられたファイルに書き出します。これは、現在の値を、後で使うために、あるいは終了/中断した当てはめを再実行するために保存しておくのに有用です。

書式:

```
update <filename> {<filename>}
```

2 番目のファイル名を指定すると、元のパラメータファイルは変更せずに 2 番目のファイルの方に更新された値を書き出します。

そうでなければ、指定したファイルが存在すれば **gnuplot** はそのファイル名に **.old** をつけてファイル名を変更し、指定したファイル名のファイルを新たに開き直します。つまり、"**update 'fred'**" とすると、それは "**!rename fred fred.old; update 'fred.old' 'fred'**" としたことと同じことになります。["filename.ext" の 12 文字しか使えない DOS や他のシステムでは、"ext" が "**old**" になって "filename" は最初のファイルに関係するもの (多分どれかがすぐに分かるでしょう) が使われます。VMS では、ファイルのバージョン管理システムが使われるため、名前の変更は行なわれません。]

より詳しい情報に関しては **fit** を参照してください。

第 III 部

Graphical User Interfaces

gnuplot のために色々なグラフィカルユーザインターフェースが書かれてきました。その win32 用の物はこの配布版に含まれています。そして、Macintosh に対するインターフェースは

```
ftp://ftp.ee.gatech.edu/pub/mac/gnuplot
```

に置いてあります。X11 に対するインターフェースは、通常の Tcl/Tk の配布場所に置かれている 3 つの Tcl/Tk プログラムがあります。

第 IV 部

Bugs

浮動小数計算例外 (浮動小数値が大きすぎる (または小さすぎる) 場合、0 で割算した場合など) は、ユーザ定義関数において時折発生します。特に、いくつかのデモで、浮動小数の範囲を越える数値を生成することが起こるようです。システムがそのような例外を無視する (**gnuplot** はそのような点を定義できない

もの、と見なします) か、または **gnuplot** の実行を中止するかは、コンパイル時 (あるいは実行時) の環境によります。

ベッセル関数は複素数引数に対しては動作しません。

ガンマ関数は複素数引数に対しては動作しません。

gnuplot の現在のバージョン 3.7 では、全ての開発は ANSI C コンパイラが使われています。現在のオペレーティングシステム、コンパイラ、ライブラリ、バージョン 3.5 のドキュメントに上げられた各 OS 毎のバグに関しては、ここではなく、今は **old_bugs** の方にあげられています。

現在の版以降、報告されたバグは以下の公式配布サイトに置かれます。

```
ftp://ftp.dartmouth.edu/pub/gnuplot
http://www.cs.dartmouth.edu/gnuplot_info.html
```

全てのバグに関しては、bug-gnuplot@dartmouth.edu に e-mail してください。

39 Old_bugs

古い Sun の OS (SunOS Sys4-3.2) には、標準入出力 (stdio) ライブラリにバグがあります。'printf' の書式 "%g" は時々正しくない表示を行ないます (例えば "2" を 200000.0 と表示する)。よって、目盛りの見出しは、Sun4 上の **gnuplot** では不正になる場合があります。回避方法としては、データのスケール変換 (定数倍する) を行なうこと、あるいはコマンド **set format** で目盛りの見出しの書式を "%7.0f" などの適当な物に変えることなどがあります。これは SunOS 4.0 では修正されているようです。

他のバグ: Sun3 上の SunOS 4.0, Sun4 上の Sys4-3.2 と SunOS 4.0 で、'sscanf' ルーチンが、書式 "%f %f" で "00 12" を不正に解釈し、0 と 12 ではなく、0 と 0 のように読んでしまいます。これはデータの入力に影響を与えます。もし、データファイルが x 座標として '00' や '000' のように書かれたゼロを含んでいる場合、y の値は不正なものとなるでしょう。データファイルをチェックするか SunOS をバージョンアップしてください。これは SunOS 4.1.1 では修正されているようです。

Sun は、大きな x に対する exp(-x) の計算で桁溢れを起こすようで、よって **gnuplot** は未定義値を取得することになります。一つの回避策は、ユーザ定義関数として e(x) = x<-500 ? 0 : exp(x) のようなものを使うことです。これは、例えばガウス関数 (exp(-x*x)) の描画に影響を与えます。x*x は非常に早く大きくなるからです。

Microsoft C 5.1 では 'printf' の %g 書式に関するひどいバグがあります。"%.2g", "%.1g", "%.0g", "%.g" のような書式を使うと、'printf' は 1e-4 から 1e-1 の間の不正な数値を表示するでしょう。%e の書式で表示されるべき数値は、 %f では小数点の後に間違った数のゼロがついた不正な数値になります。この問題を回避するには、%e や %f の書式を明示的に使用してください。

Microsoft C でコンパイルされた **gnuplot** は、テストでは 2 つの VGA ディスプレイで正しく動作しませんでした。多分 CGA, EGA, VGA ドライバは Microsoft C グラフィックライブラリを使うように書き直すべきでしょう。Borland C++ でコンパイルされた **gnuplot** は Turbo C グラフィックドライバを使用し、これは VGA ディスプレイでちゃんと動作します。

VAX/VMS 4.7 C コンパイラ (release 2.4) の 'printf' の書式 %g の実装は貧弱です。数は数値としては正しく表示されますが、要求した書式でないものになり得ます。K&R 第 2 版には、書式 %g は、指数部が -4 より小さい場合、あるいは指定精度以上になる場合は %e を使う、と書いてあります。しかし VAX は

指数部が -1 より小さい場合に %e を使います。VAX では、1 より小さい数に対して、%e と %f のどちらを使うのかを決定するときに精度が評価されていないようです。この問題を回避するには、%e, %f を明示的に使ってください。VAX C 2.4 のリリースノートより: e,E,f,F,g,G の結果は常に小数点を含みます。g と G に対しては、末尾のゼロは取り除かれません。

VAX/VMS 5.2 C コンパイラ (release 3.0) は release 2.4 よりも多少ましな書式 %g の実装を持ちますが、さほどではありません。末尾の小数点は取り除かれるようになりましたが、末尾のゼロは相変わらず %g の指数表記の数からは取り除かれていません。

これらの問題はコンパイラの問題ではなく、実際にはライブラリの問題です。よって、DEC コンパイラ、または他のもの (例えば最新の gcc) を使って **gnuplot** をコンパイルしてももこの問題は起こるでしょう。

ULTRIX X11R3 は、X11 ドライバがグラフを "一つおき" に表示する、ということを引き起こすバグがあります。このバグは、DEC の X11R4 では修正されたようで、新しいリリースの ULTRIX ではこの問題を起こさないようです。古いサイトでの解決策は、X11 ライブラリをバージョンアップする (DEC から、あるいは MIT から直接) か、または x11.trm ファイルのコンパイル時に ULTRIX_KLUDGE を定義してください。しかし、kludge は理想的な回避策ではないことに注意してください。

NeXT OS 2.0 では、定数 HUGE が不正な値に定義されています。HUGE は plot.h で 1e38 にセットされるのが正しいです。このエラーは NeXT OS のバージョン 2.1 では修正されています。

HP プロッタの古いモデルの中にはページの排出コマンド 'PG' を持たないものがあります。現在の HPGL ドライバはこのコマンドを HPGL_reset で使用していますが、そのようなプロッタには、このコマンドを取り除く必要があるでしょう。現在の PCL5 ドライバは、グラフィックと同様にテキストにも HPGL/2 を使用しています。これはスケーラブル PCL フォントを使うように修正されるべきでしょう。

Atari 版では、プリンタに直接出力を送れません (出力ファイルを **/dev/lp** としては)。それはバイナリ出力の LF (改行) に CR (復帰) が追加されるからです。回避策としては、出力をファイルに書き出し、それをその後でシェルコマンドを使ってプリンタにコピーしてください。

AIX 4 では、データファイル中の文字 'NaNq' は、内部の ' 未定義' フラグとしては処理されず、内部の特殊な 非数値 として保存されてしまいます。回避するには **set missing 'NaNq'** を使ってください。

リリース後のバグの最新のリストは以下の WWW ページにあります:

```
http://www.cs.dartmouth.edu/gnuplot_info.html
```

バグがあったら bug-gnuplot@dartmouth.edu に報告してください。