

Gnuplot 入門

第 0.08 版

半場 滋¹

2007 年 8 月 31 日 (水)

¹琉球大学工学部電気電子工学科

この文書について

この文書は琉球大学工学部電気電子工学科の学生向けの講義や実験などの補助教材として作成されています。作成にあたって、読者が UNIX の基本的な操作に習熟していることを想定しています。

本文書は基本的にチュートリアルです。読者が自分がやりたいことに似た例題を探しながら読み進めてゆくことを想定しています。部分的にはリファレンスとしても使えるように配慮してある箇所もありますが、全般に説明は網羅的ではないので注意して下さい。

この文書はコピーフリーとします。本文書に一切改変を加えないという条件の下で、自由な再配布を許可します。

著者は本文書の内容に関して正確を期してはいますが、本文書の内容の正しさに関しては一切保証しません。また、本文書の内容に関する質問は一切受け付けません。ただし、誤りの報告については歓迎します。

この文書の使い方

この文書は全 10 章から成ります。

第 1 章は gnuplot の最低限の操作を覚えるための簡単なチュートリアルです。gnuplot の操作に慣れていない読者はこのチュートリアルをすべて読んで下さい。

第 3 章では gnuplot の基本的なコマンドが説明されています。また、第 4 章では gnuplot のコマンドの基本的な構文と、gnuplot の組み込み関数、gnuplot で新しく関数を宣言する方法が説明されています。ここで説明されている内容は gnuplot の基礎ですので、gnuplot に不慣れな読者はざっと目を通しておくといよいでしょう。ただし、これらの記述を記憶する努力をする必要はありません。

第 5 章ではターミナルの切り換えについて説明します。この章は必要に応じて参照すればよく、読者があらかじめ目を通しておく必要はありません。

第 6 章には gnuplot のグラフの各部の名称とグラフの各部を調整するためのコマンドの一覧が記載されています。この章は、読者が第 7 章以降のチュートリアルを見ていて用語やコマンドの意味について混乱したときに参照することを意図して書かれています。通読するときには、gnuplot のグラフの各部の名称に関する記述の部分のみに目を通せば十分です。

第 7 章、第 8 章、第 9 章はそれぞれ 2 次元グラフ、3 次元グラフおよびデータファイルからグラフを作成する方法のチュートリアルです。これらの章のうち、第 7 章の関する記述はかなり詳細なのですが、第 8 章、第 9 章では、第 7 章と同じ使い方ができる事項に関する記述はある程度省略されています。これらの章は、読者が自分の希望する操作に類似した例を探して使うことを意図して書かれています。通読する必要はありません。

第 10 章は plus モード に関するリファレンスです。この章は、 \LaTeX の知識を持たない読者にも plus モードが使えるように書かれています。この章はリファレンスですから、通読する必要はありません。

本文書では、原則として、ユーザが直接入力する文字列をタイプ文字で標記しています。なお、コマンド入力後には、行末で Enter キー を押す必要があります。本文中のコマンドの例では、`[Enter]` という記号で Enter キーを押す箇所を標記しています。

このドキュメントに含まれないもの

gnuplot が取り扱える多数のターミナルのうち取り扱われているのは x11, postscript, tgif, latex, png, pbm のみです。多くのコンピュータシステムのターミナルについては、筆者の手元に動作チェックできる環境がないので説明を省略しました。また、複数用意されている \TeX 系のターミナルの中では、latex のみを簡単に取り上げました。これは、機能がすくない \TeX の picture 環境で作図をおこなう必然性が筆者にはあまり感じられなかったからです。

2 次元グラフ作成に関する機能のうち、financebars などの工学系の学生が使うことはまずないと思われる事項に関する説明は省略しました。

gnuplot の 3 次元グラフ作成に関する機能に関する説明は、gnuplot の 3 次元グラフ作成機能にやや不安定な部分があるのと、3 次元グラフそのものの利用頻度が 2 次元グラフと比較して高いとは思えないので、ある程度省略してあります。

gnuplot でアニメーションを作成することもできるのですが、これに関する記述も省略しました。

目 次

第 1 章	gnuplot とは	1
第 2 章	チュートリアル	2
2.1	gnuplot を起動する	2
2.2	gnuplot を終了する	2
2.3	基本的な操作法	3
2.4	1 変数関数を描画する	3
2.5	eps ファイルに保存する	3
2.6	非対話的な gnuplot の実行	4
2.7	ファイルからコマンドを実行する	5
2.8	コマンドのファイルへの保存	5
第 3 章	gnuplot の基本的なコマンド	6
3.1	ファイルとディレクトリの操作	7
3.1.1	現在のディレクトリを表示する	7
3.1.2	ディレクトリを移動する	7
3.1.3	ファイルをロードする	8
3.1.3.1	オプションなしでファイルをロードする -- load --	8
3.1.3.2	オプションつきでファイルをロードする -- call --	9
3.1.4	ファイルをもう 1 回読む	9
3.1.5	関数や変数などを保存する	10
3.2	グラフの描画	12
3.2.1	一時停止	12
3.2.2	テスト画面を描画する	12
3.3	数値やオプションなどの設定と表示	13
3.3.1	オプションを設定する	13
3.3.2	オプションの状態を表示する	13
3.3.3	変数の値を表示する	13
3.3.4	オプションをすべてクリアする	13
3.4	オンラインマニュアルを見る	14
3.5	その他のコマンド	20
3.5.1	条件分岐	20
3.5.2	シェルに戻る	20
3.6	コマンドの省略記法	21
第 4 章	gnuplot の構文と関数	22
4.1	構文	22
4.2	数式と組み込み演算子	23
4.3	数値の演算の結果を画面に表示する	24

4.4	ユーザ変数	24
4.4.1	ユーザ変数の一覧を見る	25
4.4.2	ユーザ変数を新しく定義する	25
4.4.3	ユーザ変数をファイルに保存する	26
4.5	複素数の表現	27
4.6	数学関数	27
4.6.1	指数関数, 対数関数, 三角関数	27
4.6.2	算術関数	28
4.6.3	その他の関数	28
4.7	ユーザ関数の定義	28
4.7.1	ユーザ関数の一覧を見る	29
4.7.2	ユーザ関数を新しく定義する	30
4.7.3	ユーザ関数を使う	30
4.7.4	関数を使うときに変数名を変更する	31
4.7.5	ユーザ関数をファイルに保存する	32
第 5 章	グラフの印刷とターミナルの切り換え	33
5.1	グラフの確認から印刷までの一連の流れ	33
5.2	ターミナルと出力ファイルの指定	34
5.2.1	ターミナルの切り換え	34
5.2.2	出力ファイルの指定	35
5.3	いろいろなターミナル	36
5.3.1	x11	36
5.3.2	postscript	37
5.3.2.1	モード	37
5.3.2.2	オプション	38
5.3.3	tgif	38
5.3.4	latex	39
5.3.5	table	39
5.3.6	png	40
5.3.7	pbm	40
第 6 章	gnuplot のグラフの構造	41
6.1	グラフの各部の名称	41
6.1.1	2次元グラフの各部の名称	41
6.1.2	3次元グラフの各部の名称	42
6.2	グラフを調整する -- set と with --	42
6.2.1	キーワード with を使う	43
6.2.2	コマンド set と show を使う	43
第 7 章	2次元グラフを描画する	47
7.1	関数のグラフのプロット	47
7.1.1	簡単な2次元グラフを描画する	47
7.1.2	y 軸や x 軸の範囲を変える	47
7.1.3	y 軸と x 軸にラベルをつける	48
7.1.4	関数のグラフの線のスタイルを変える	49

7.1.4.1	グラフを点で表示	49
7.1.4.2	線と点を重ねる	49
7.1.4.3	インパルスで表示	50
7.1.4.4	階段グラフ	51
7.1.4.5	線の太さを変える	52
7.1.4.6	線の種類を変える	53
7.1.4.7	点の種類を変える	54
7.1.4.8	点の大きさを変える	57
7.1.4.9	点と線の種類を同時に変える	57
7.1.5	描画するときのサンプル点の間隔を調整する	57
7.1.6	キー（関数など説明）の操作	58
7.1.6.1	グラフを説明するテキストを変更する	58
7.1.6.2	キーのスタイルと位置の制御	58
7.1.7	図の標題の制御	59
7.1.8	詳細な目盛りの制御	60
7.1.8.1	目盛りの表示	60
7.1.8.2	目盛りの数値の書式を変更する	61
7.1.8.3	副目盛りの表示と目盛りの調整	62
7.1.8.4	x 軸や y 軸の目盛りを曜日で表示する	63
7.1.9	座標軸の対数目盛りと通常の日盛りの切り換え	64
7.1.10	枠や座標軸，グリッドの表示と非表示	65
7.1.11	グラフの大きさの指定	65
7.1.12	グラフの起点の指定	66
7.1.13	グラフ上下左右の余白の調整	67
7.1.14	グラフとボックスの間隔の調整	68
7.1.15	グラフに作成日時を入れる	68
7.1.16	グラフィックスウィンドウに描画されたグラフを消す	69
7.2	2 個以上の関数を同時にプロットする	70
7.3	グラフをどんどん重ね書きしてゆく	70
7.4	直交座標を用いたパラメータ付き曲線のプロット	73
7.4.1	パラメータ表示モードの起動と終了	73
7.4.2	基本的な描画コマンド	73
7.4.3	パラメータ付き曲線の例	75
7.4.4	パラメータの範囲や描画範囲の指定	76
7.5	極座標を用いたパラメータ付き曲線のプロット	78
7.5.1	極座標モードの起動と終了	78
7.6	簡単な例	79
7.6.1	変数と描画範囲の指定	79
第 8 章	いろいろな 3 次元グラフを描画する	81
8.1	直交座標系を用いた 3 次元グラフの描画	81
8.1.1	単純な 3 次元グラフの表示	81
8.1.2	描画範囲の調整	82
8.1.3	グラフにかかった網目の大きさを変える	82
8.1.4	隠れ面の表示の切り換え	83
8.1.5	視点の変更	83

8.1.6	水平面（座標が書かれる面）の位置の変更	85
8.1.7	網のかかった面の表示をやめる	87
8.1.8	等高線の表示	87
8.1.8.1	等高線をプロットする	87
8.1.8.2	等高線の高さ表示のフォーマットを変える	89
8.1.8.3	等高線のよりきめ細かい制御	89
8.1.9	グラフの表示スタイルを網かけや等高線以外にする	91
8.1.10	PostScript ファイルの BoundingBox について	91
8.2	パラメータ表示された 3 次元グラフ	92
8.2.1	パラメータ表示モードの起動と終了	92
8.2.2	簡単な例題	93
8.2.3	描画範囲の調整	94
8.2.4	隠れ面や網の切り方の処理	94
8.2.5	その他の各種のオプション	95
8.2.6	3 次元のパラメータ付き曲線を表示する	95
8.2.7	注意事項	96
第 9 章	データファイルのプロット	97
9.1	2 次元データのプロット	97
9.1.1	単純なデータファイルのプロット	97
9.1.2	データファイルの書式	99
9.1.3	データファイルの例	100
9.1.3.1	不連続点を含むデータファイル	101
9.1.3.2	複数のグラフのデータを含むデータファイル	101
9.1.4	複数のグラフが含まれるデータファイルからグラフを抜き出す	102
9.1.5	データファイルのどの列を使ってプロットするかを変更する	104
9.1.6	データを間引く	107
9.1.7	グラフのスタイルを変更する	113
9.1.7.1	点や線のスタイルを変更する	113
9.1.7.2	エラーバーを表示する	113
9.1.7.2.1	縦軸のみに誤差を含むグラフ	113
9.1.7.2.2	横軸のみに誤差を含むグラフ	114
9.1.7.2.3	横軸と縦軸誤差を含むグラフ	114
9.1.7.2.4	誤差の表記が混在したデータファイルからグラフを作りたいとき	115
9.1.8	データ点のあいだを結ぶ線のなめらかさを変える	116
9.1.8.1	準備	116
9.1.8.2	キーワード unique を指定したグラフ	117
9.1.8.3	キーワード csplines を指定したグラフ	117
9.1.8.4	キーワード bezier を指定したグラフ	118
9.1.8.5	キーワード sbezier を指定したグラフ	118
9.1.8.6	キーワード acsplines を指定したグラフ	119
9.1.9	デフォルトのデータファイル描画のスタイルの確認と変更	121
9.2	3 次元データファイルのプロット	122
9.2.1	単純な 3 次元データファイルのプロット	122
9.2.2	網かけグラフのプロット	122
9.2.3	x 座標と y 座標の自動割り付け	123

9.2.4	グラフのスタイルの変更やデータファイルからのデータの抜き出し	124
9.2.5	球座標と円柱座標	124
9.2.5.1	座標系の切り換えのためのコマンド	124
9.2.5.2	球座標によるデータファイル	124
9.2.5.3	円柱座標によるデータファイル	125
9.3	非線形最小 2 乗法によるデータへの曲線のあてはめ	126
9.3.1	簡単な例題	126
9.3.2	コマンド <code>fit</code> の使い方	129
9.3.3	データに非線形関数を当てはめるときの注意	130
第 10 章	plus モード: 数式の組版とテキストのより細かい制御	131
10.1	plus モードの開始と終了	131
10.2	plus モードのコマンド	132
10.3	文字の大きさや字体の指定と回転	132
10.3.1	文字の大きさの指定	133
10.3.2	字体の指定	134
10.3.3	テキスト全体の回転	134
10.4	特殊記号および各種の記号	135
10.4.1	特殊記号	135
10.4.2	各種記号, 外国語の文字	136
10.5	数式の表現	136
10.6	グラフの標題に簡単な数式を入れる	137
10.7	数式を組むためのいくつかのコマンド	138
10.8	利用可能な数学記号	139
10.8.1	矢印	139
10.8.2	ギリシャ文字	139
10.8.3	数式中のアクセント記号	140
10.8.4	関数記号	140
10.8.5	2 項演算子と関係記号	140
10.8.6	その他の数学記号	140
10.8.7	大きい括弧	141
10.8.8	いくつかの例	141
10.8.9	機能の制限	141

第1章 gnuplot とは

gnuplot は、2 次元および 3 次元のグラフを描画するためのフリーウェアです。

gnuplot のユーザインターフェースは利用者がコマンドを打ち込んでゆく形態 (CUI) です。ですから、gnuplot を使うためには、最低限のコマンドをいくつか覚えておく必要があります。一方、コマンド `help` で閲覧することができるオンラインマニュアルが充実しているため、最低限のコマンドさえ覚えてしまえば、より高級な使い方をするとときに困ることはほとんどありません。

gnuplot の 2 次元グラフ描画機能は極めて強力であり、各種の関数やデータのグラフが自由自在に作成できます。非常に多様な画像の形式をサポートしていることも gnuplot の特徴です。gnuplot で作成したグラフは、PostScript 形式や EPS 形式、`tgif` のオブジェクトファイル、PNG フォーマット、PBM フォーマットなど、各種のフォーマットで保存することができます。gnuplot には 3 次元グラフを作成する機能もありますが、こちらは 2 次元ほど強力ではありません。

本文書では、gnuplot の基本的な使い方について、チュートリアル形式で 2 次元の場合を中心に解説してゆきます。

第2章 チュートリアル

以下のコマンドの例では， [Enter] という記号で Enter キーを押す箇所を標記しています．

2.1 gnuplot を起動する

gnuplot を起動するには， kterm など

gnuplot [Enter]

と入力します． すると， 図 2.1 のように gnuplot の起動メッセージが画面に表示され， gnuplot が立ち上がります．

```
% gnuplot

G N U P L O T
Unix version 3.7
patchlevel 1 (+1.2.0 2001/01/11)
last modified Fri Oct 22 18:00:00 BST 1999

Copyright (C) 1986 - 1993, 1998, 1999
Thomas Williams, Colin Kelley and many others

Type 'help' to access the on-line reference manual
The gnuplot FAQ is available from
<http://www.ucc.ie/gnuplot/gnuplot-faq.html>

Send comments and requests for help to <info-gnuplot@dartmouth.edu>
Send bugs, suggestions and mods to <bug-gnuplot@dartmouth.edu>

Terminal type set to 'x11'
gnuplot> █
```

図 2.1: gnuplot の起動画面

2.2 gnuplot を終了する





gnuplot を終了するには， gnuplot の画面内で

quit [Enter]

と入力します．

2.3 基本的な操作法

gnuplot の基本的な操作法は (t)csch や bash などとほぼ同様です。キーボードからコマンドを打ち込んで `Enter` キーを押すと、打ち込んだ内容が実行されます。コマンド行は編集可能で、編集には以下のキーが使えます。

キー 1	キー 2	効果
[Ctrl]+ <code>f</code>		1 文字前進
[Ctrl]+ <code>b</code>		1 文字後退
[Ctrl]+ <code>p</code>		1 個前に入力した行を表示
[Ctrl]+ <code>n</code>		1 個次に入力した行を表示
[Ctrl]+ <code>e</code>		行末に移動
[Ctrl]+ <code>a</code>		行頭に移動
[Ctrl]+ <code>d</code>	[Del]	カーソル位置の文字を消す
[Ctrl]+ <code>h</code>	[BackSpace]	カーソルの手前の文字を消す
[Ctrl]+ <code>k</code>		カーソルの位置から行末までを消す
[Ctrl]+ <code>l</code>		画面を再描画する
[Ctrl]+ <code>u</code>		行全体を消す
[Ctrl]+ <code>w</code>		最後の単語を消す

2.4 1 変数関数を描画する

関数のグラフを描画するにはコマンド `plot` を使います。たとえば、gnuplot のウィンドウ内で

```
plot sin(x) [Enter]
```

と入力すると、図 2.2 のように正弦関数のグラフが描画されます。横軸の範囲と縦軸は適当に調整されます。

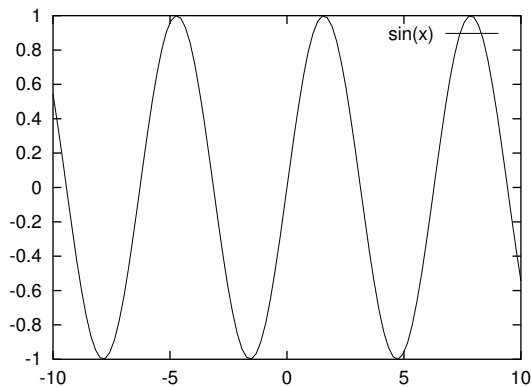


図 2.2: 正弦関数のグラフ

2.5 eps ファイルに保存する

描画したグラフを \LaTeX で処理可能な eps (encapsulated PostScript) 形式で保存するには、まず `set terminal` というコマンドを使って出力データの形式を `postscript` に変更します。

```
set terminal postscript eps [Enter]
```

キーボードからの入力に誤りがなければ、

```
Options are 'eps noenhanced monochrome  
dashed defaultplex "Helvetica-Ryumin" 14'
```

どのようなメッセージが画面に表示されます。

続いて、`set output` というコマンドを使って出力ファイルの名前を指定します。ここでは、`test.eps` としておきましょう。

```
set output 'test.eps' [Enter]
```

ここまでの手順が終わった状態で

```
replot [Enter]
```

と入力すると、グラフが `test.eps` というファイルに保存されます。ここまでの作業が終わったときの `gnuplot` のウィンドウの内容を図 2.3 に示します。

```
GNUPLOT  
Unix version 3.7  
patchlevel 1 (+1.2.0 2001/01/11)  
last modified Fri Oct 22 18:00:00 BST 1999  
  
Copyright (C) 1986 - 1993, 1998, 1999  
Thomas Williams, Colin Kelley and many others  
  
Type 'help' to access the on-line reference manual  
The gnuplot FAQ is available from  
<http://www.ucc.ie/gnuplot/gnuplot-faq.html>  
  
Send comments and requests for help to <info-gnuplot@dartmouth.edu>  
Send bugs, suggestions and mods to <bug-gnuplot@dartmouth.edu>  
  
Terminal type set to 'x11'  
gnuplot> plot sin(x)  
gnuplot> set terminal postscript eps  
Terminal type set to 'postscript'  
Options are 'eps noenhanced monochrome dashed defaultplex "Helvetica-Ryumin" 14'  
gnuplot> set output 'test.eps'  
gnuplot> replot  
gnuplot> █
```

図 2.3: ファイルへの保存が終わったときの `gnuplot` のウィンドウの状態

いったん出力データの形式を `postscript` 形式に変更すると、それ以降は、利用者が出力データの形式を切り換えるコマンドを実行するまで、出力データの形式はずっと `postscript` のままになります。ところで、出力データの形式が `postscript` になっていると、描画したグラフのようすを画面で確認することはできません。再びグラフを画面で確認できる状態に戻したいときには、

```
set terminal x11 [Enter]
```

というコマンドを実行する必要があります。

2.6 非対話的な `gnuplot` の実行

`gnuplot` コマンドをファイルにあらかじめ書いておいて、そのファイルを実行することもできます。

図 2.4 のような内容のファイルがカレントディレクトリに用意されているものとしましょう。ファイル名は

```
plot sin(x)
pause -1
```

図 2.4: gnuplot のスクリプトファイルの例

test.gnuplot とします。

このようなファイルを準備した上で kterm など

```
gnuplot test.gnuplot [Enter]
```

と入力すると、先ほどと同じように正弦関数のグラフが画面に描画されます。

なお、非対話的に gnuplot を実行した場合には、gnuplot はすべてのコマンドを実行し終わるとただちに終了してしまいます。上の例で

```
pause -1
```

とある部分は、ユーザが [Enter] キーを押すまで gnuplot が終了しないようにするためのものです。

2.7 ファイルからコマンドを実行する

gnuplot を起動し、gnuplot のウィンドウ内でコマンド load を実行することでも、あるファイルに書かれたコマンド群を実行することができます。コマンド load を使うときには、引数としてファイル名を単一引用符「`'`」で囲って指定します。

例として、先ほどの例と同じファイル test.gnuplot を実行する場合を考えましょう。

このためには、

```
load 'test.gnuplot' [Enter]
```

と入力します。

2.8 コマンドのファイルへの保存

コマンド save を実行すると、最も最近のコマンド plot に関連した一連のコマンド群をファイルに保存できます。

コマンド save を使うときも、引数としてファイル名を単一引用符「`'`」で囲って指定します。

例として、今実行したコマンドを test2.gnuplot というファイルに保存してみます。

```
save 'test2.gnuplot' [Enter]
```

コマンド save を実行した場合には、gnuplot が標準で設定するいろいろなパラメータの内容がすべてファイルに書き込まれるため、ファイルの内容はかなり複雑になります。

第3章 gnuplot の基本的なコマンド

表 3.1 に gnuplot の基本的なコマンドを示します。本節ではこれらの中の代表的なものについて簡単に説明してゆきます。

表 3.1: gnuplot の基本的なコマンド

機能	コマンド
ファイルとディレクトリの操作	
終了する	exit, quit
現在のディレクトリを表示する	pwd
ディレクトリを変更する	cd
ファイルをロードする	call, load
ファイルをもう 1 回読む	reread
ファイルにデータを保存する	save
グラフの描画	
2 次元グラフをプロットする	plot
3 次元グラフをプロットする	splot
グラフが表示されるウィンドウをクリアする	clear
グラフを再描画する	replot
グラフの一時停止	pause
テスト画面を表示する	test
数値やオプションなどの設定と表示	
変数の値などを画面に表示する	print
各種オプションを設定する	set
各種設定の内容を表示する	show
すべての設定をクリアする	reset
データファイルの操作	
データに直線や曲線をあてはめる	fit
直線あてはめ時にパラメータをファイルに書き出す	update
その他	
条件分岐	if
シェルに戻る	shell
オンラインマニュアルを見る	help

3.1 ファイルとディレクトリの操作

3.1.1 現在のディレクトリを表示する

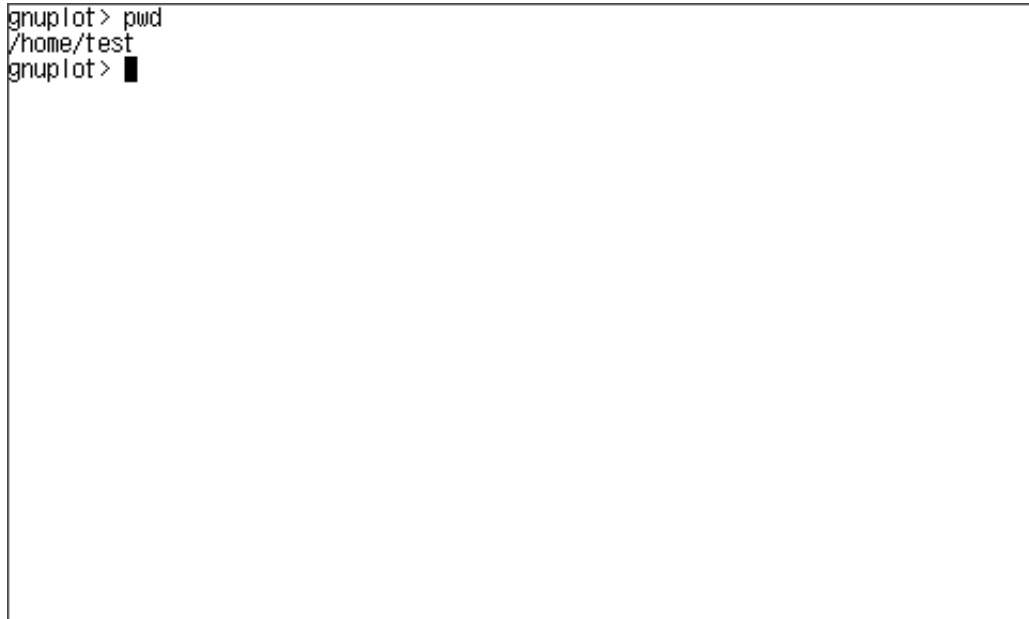
現在のディレクトリを表示するには、`pwd` というコマンドを使います。

コマンド `pwd` の使い方は UNIX のそれと同じで、

```
pwd [Enter]
```

です。

コマンド `pwd` を実行すると、たとえば図 3.1 のように、現在作業をしているディレクトリが表示されます。



```
gnuplot> pwd
/home/test
gnuplot> █
```

図 3.1: コマンド `pwd` を実行したところ

3.1.2 ディレクトリを移動する

ディレクトリを移動するには `cd` というコマンドを使います。

コマンド `cd` の使い方は UNIX とは異なり、オプションとしてディレクトリ名を単一引用符で囲ったものを与えます。

たとえば、`/home/test` というディレクトリに移動したいときには、

```
cd '/home/test' [Enter]
```

と入力します。

ディレクトリを指定するときには、文字「`.`」(ピリオド 1 個)で現在のディレクトリを、文字列「`..`」(ピリオド 2 個)で現在のディレクトリの 1 個根本のディレクトリをあらわすことができます。また、記号「`/`」がディレクトリ名の先頭についている場合は絶対パス(ルートディレクトリが起点)、ディレクトリ名の先頭に記号「`/`」がついていない場合は相対パス(現在作業中のディレクトリが起点)として解釈されます。

これ以外の UNIX の標準的な記法、たとえば記号「`~`」などは使えません。

3.1.3 ファイルをロードする

gnuplot のコマンドや変数などが書かれたファイルを読むためのコマンドには、load と call の 2 種類があります。

3.1.3.1 オプションなしでファイルをロードする – load –

コマンド load は、単純に gnuplot のコマンドなどが書かれたファイルを読み込むコマンドです。あらかじめファイルに連の gnuplot のコマンドを書いておいてからそのファイルを load コマンドで読み込むと、そのファイルに書かれたコマンドすべてがキーボードから打ち込んだ場合と同じように上から下に順に実行されてゆきます。

コマンド load の使い方は、

```
load 'ファイルの名前' [Enter]
```

です。ここに、「ファイルの名前」と書かれた部分には、gnuplot のコマンドなどが書かれたファイルが入ります。

たとえば、図 3.2 のような内容のファイルを用意しておいて（ファイル名を sample1.gp とします）、gnuplot のウィンドウ内で

```
load 'sample1.gp' [Enter]
```

というコマンドを実行することは、キーボードから

```
set terminal x11 [Enter]
set xrange [0:10] [Enter]
set yrange [0:10] [Enter]
plot x+1 [Enter]
set terminal postscript eps [Enter]
set output 'test.eps' [Enter]
replot [Enter]
```

という内容を 1 行ずつ打ち込んでゆくことと同一の効果があります。

```
set terminal x11 set xrange [0:10] set yrange [0:10] plot x+1 set
terminal postscript eps set output 'test.eps' replot
```

図 3.2: ファイル sample1.gp の内容

なお、コマンド load は、読み込まれるファイルがカレントディレクトリにあると仮定します。カレントディレクトリ以外にあるファイルを読み込みたいときには、絶対パスあるいは相対パスでディレクトリと合わせてファイル名を指定するか、あるいはコマンド cd を使ってそのファイルがあるディレクトリまで移動して下さい。

たとえば、絶対パスを使って読み込むファイル名を指定するときには

```
load '/home/test/sample1.gp' [Enter]
```

のようにします。

3.1.3.2 オプションつきでファイルをロードする – call –

コマンド `call` のはたらきは `load` とほぼ同様なのですが、コマンド `load` にはパラメータがないのに対してコマンド `call` は 10 個までのパラメータを持つことができるという点が違います。ですから、コマンド `call` を使った場合は、あらかじめファイルに書き込んでおく `gnuplot` のコマンドをまとめたファイルにいくつか可変な部分を持たせておいて、そのファイルを読み込むときにいくつかパラメータを指定する、ということができます。

コマンド `call` を使って読み込まれるファイル中では、パラメータが \$0 から \$9 までの記号で指示されます。

例として、やや人工的ですが、図 3.3 のようなファイルを考えましょう。ただし、ファイル名を `sample2.gp` とします。

```
plot [$0:$1] sin(x) with $2
```

図 3.3: ファイル `sample2.gp` の内容

このファイルを `call` を使って読み込む場合は、パラメータとしてグラフの初期値、グラフの終了値とグラフの書き方を指定することができます。

なお、ここで出て来たコマンド `plot` の使い方といろいろなオプションの指定法については第 7 章で詳しく説明されています。ここでは、\$0 と書かれたところに横軸の左端、\$1 と書かれたグラフの横軸の右端の座標を書き込み、\$2 と書かれたところに「`lines`」という文字を書き込むとグラフが線で描画され、\$2 と書かれたところを「`points`」という文字に変えるとグラフが点で描画されるということだけ理解しておいて下さい。

さて、図 3.3 のようなファイルを用意した上で、`gnuplot` のウィンドウで

```
call 'sample2.gp' 0 10 lines [Enter]
```

と入力すると図 3.4 のようなグラフが、

```
call 'sample2.gp' 5 25 points [Enter]
```

と入力すると図 3.5 のようなグラフが得られます。

横軸の読みとグラフの形状から、今指定したパラメータが確かにグラフに反映されていることが理解できるでしょう。

コマンド `call` も、読み込まれるファイルがカレントディレクトリにあると仮定します。カレントディレクトリ以外にあるファイルを読み込みたいときには、絶対パスあるいは相対パスでディレクトリと合わせてファイル名を指定するか、あるいはコマンド `cd` を使ってそのファイルがあるディレクトリまで移動して下さい。

なお、`gnuplot` のコマンドが書かれたファイルでパラメータがまったくないものを読み込むときにもコマンド `call` を使うことができます。この場合の挙動はコマンド `load` と完全に同一になります。

3.1.4 ファイルをもう 1 回読む

コマンド `load` を使って読み込まれたファイルをもう 1 回読み直して実行するには `reread` というコマンドを使います。

コマンド `reread` の使い方は、

```
reread [Enter]
```

です。

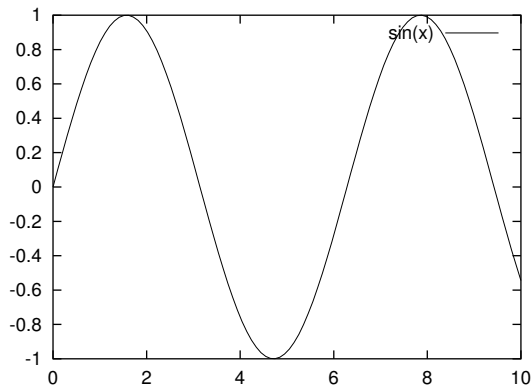


図 3.4: コマンド call の使用例 (1)

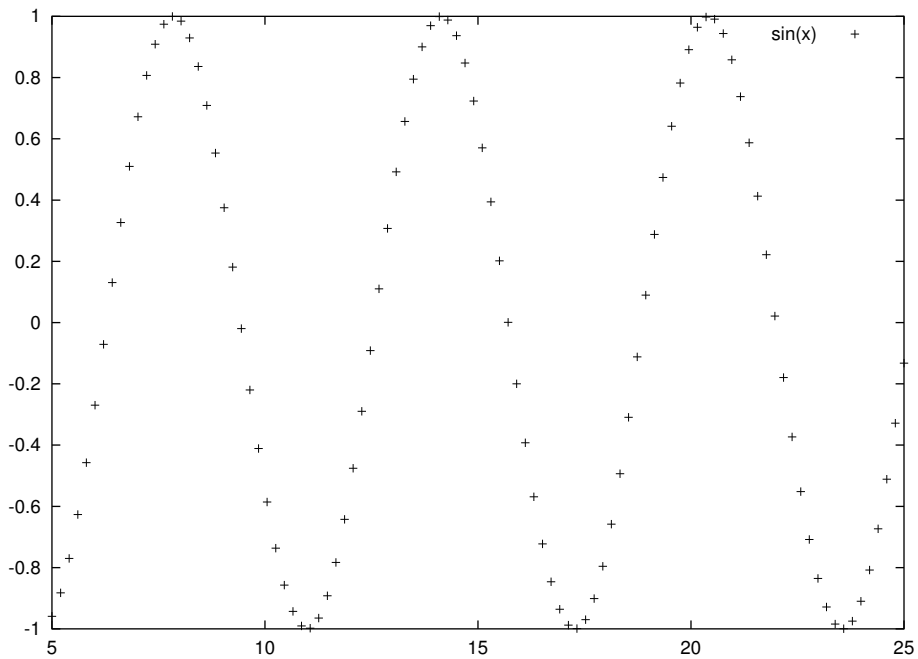


図 3.5: コマンド call の使用例 (2)

表 3.2: コマンド save の使い方

機能	コマンド
最も最近のコマンド plot とすべての関数, 変数および設定を保存	save 'ファイル' [Enter]
すべての関数を保存	save functions 'ファイル' [Enter]
すべての変数を保存	save var 'ファイル' [Enter]
すべての設定を保存	save set 'ファイル' [Enter]

3.1.5 関数や変数などを保存する

いろいろな関数や変数などを保存するときには, コマンド save を使います. コマンド save には 4 種類の使い方があります. これらを表 3.2 にまとめておきます.

なお、「ファイル」と書かれた部分には適当なファイル名が入ります。

コマンド `save` を実行すると `gnuplot` を起動した標準で設定される膨大なパラメータがすべてファイルの保存されるため、ファイルの大きさがかなり大きくなります。

3.2 グラフの描画

グラフの描画に関連したコマンドについては次章以降で詳しく説明されているので、ここでは、以下であまり触れる機会がないコマンドについて簡単に紹介しておきます。

3.2.1 一時停止

グラフが表示されたところで一旦作業を止めるコマンドが `pause` です。このコマンドは `gnuplot` のコマンドをファイルから読み込んで実行していて、グラフが表示されるごとに一旦実行を停止したい場合に有効です。

コマンド `pause` の使い方は2種類あります。

ひとつめは、

```
pause 数 [Enter]
```

というものです。これは、「数」の部分に指定された秒だけ `gnuplot` の実行を中断します。

ふたつめは、

```
pause -1 [Enter]
```

というものです。これは、`[Enter]` キーが押されるまで実行を中断します。

実行を中断しているときに何かメッセージを表示したいときには、

```
pause 数 '表示するメッセージ' [Enter]
```

あるいは

```
pause -1 '表示するメッセージ' [Enter]
```

のように、表示したいメッセージを単一引用符で囲って指定します。

3.2.2 テスト画面を描画する

今グラフが描画されるウィンドウがどのような状態になっているかをユーザに知らせるために、テスト画面を表示するコマンドが用意されています。このコマンドの名前が `test` です。

```
test [Enter]
```

のようにして実行します。

3.3 数値やオプションなどの設定と表示

3.3.1 オプションを設定する

オプションを設定するには `set` というコマンドを使います。 `set` の使い方は

```
set キーワード1 キーワード2 ... [Enter]
```

のように、コマンド `set` に続いて必要なキーワードをたくさん並べてゆく、というものです。具体的な使い方については次章以降で説明します。

3.3.2 オプションの状態を表示する

いろいろなオプションを設定するコマンド `set` に対応して、オプションの状態を表示するコマンドがあります。それが `show` です。

コマンド `show` の基本的な使い方は、

```
show オプションの名前 [Enter]
```

です。具体的な使い方は次章以降で説明します。

3.3.3 変数の値を表示する

変数の値などを画面に表示するコマンドが `print` です。

`gnuplot` を起動した段階では、円周率をあらわす変数 `pi` だけが設定されています。

この数値を表示するには、

```
print pi [Enter]
```

とします。すると、図 3.6 に示すように、変数 π の値（円周率）が表示されます。



```
gnuplot> print pi  
3.14159265358979  
gnuplot> █
```

図 3.6: 変数 `pi` の値が表示されたところ

3.3.4 オプションをすべてクリアする

コマンド `set` で設定できるすべてのオプションをクリアするためのコマンドが `reset` です。使い方は

```
reset [Enter]
```

です。

3.4 オンラインマニュアルを見る

オンラインマニュアルを見るためのコマンドが `help` です。このコマンドの基本的な使い方は

```
help トピック [Enter]
```

です。

コマンド `help` の使い方は、

```
help help [Enter]
```

とすることで参照できます。

マニュアル表示をやめるときには、

```
q
```

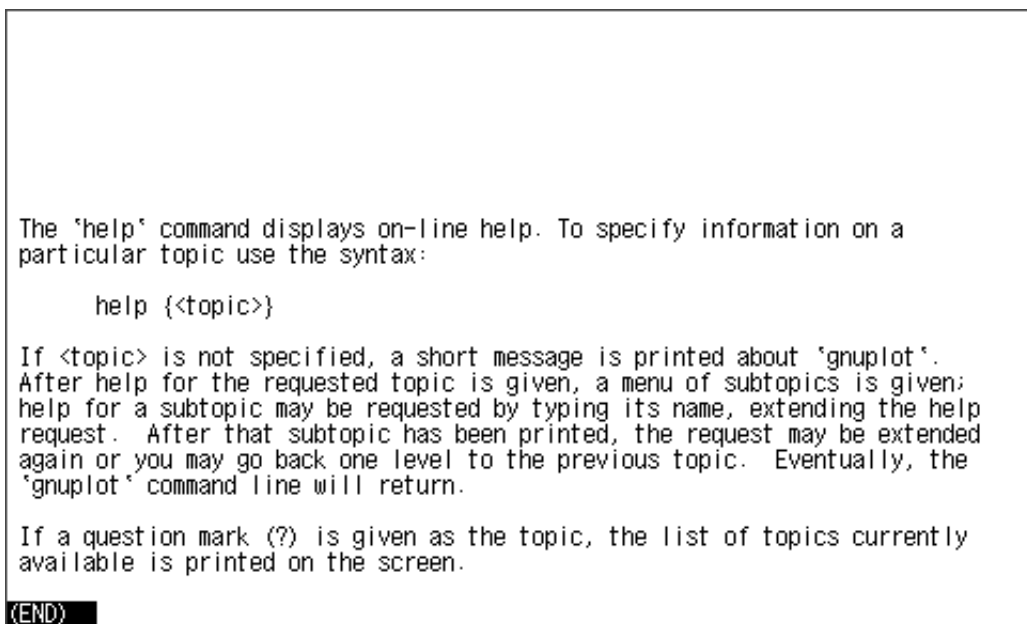
と入力します。

コマンド `help` を実行すると、そのトピックに関する簡単な説明に続いて、もしあればサブトピックの一覧が表示されます。

たとえば、`gnuplot` のウィンドウで、

```
help help [Enter]
```

と入力すると、画面は図 3.7 のようになります。



```
The 'help' command displays on-line help. To specify information on a
particular topic use the syntax:

    help {<topic>}

If <topic> is not specified, a short message is printed about 'gnuplot'.
After help for the requested topic is given, a menu of subtopics is given;
help for a subtopic may be requested by typing its name, extending the help
request. After that subtopic has been printed, the request may be extended
again or you may go back one level to the previous topic. Eventually, the
'gnuplot' command line will return.

If a question mark (?) is given as the topic, the list of topics currently
available is printed on the screen.

(END)
```

図 3.7: コマンド `help` のオンラインマニュアル

コマンド `help` のオンラインマニュアルはサブトピックを含まず、かつマニュアル本文が短いので、マニュアルが 1 画面におさまっていて、マニュアルの末尾に (END) という文字が白黒反転して表示されています。マニュアルが表示された状態からは、

```
q
```

と入力すれば抜けられます。

これに対し、

```
help plot [Enter]
```

と入力すると、画面は図 3.8 のようになります。

```
'plot' is the primary command for drawing plots with 'gnuplot'. It creates
plots of functions and data in many, many ways. 'plot' is used to draw 2-d
functions and data; 'splot' draws 2-d projections of 3-d surfaces and data.
'plot' and 'splot' contain many common features; see 'splot' for differences.
Note specifically that 'splot's 'binary' and 'matrix' options do not exist
for 'plot'.

Syntax:
  plot {<ranges>}
      {<function> | {"<datafile>" {datafile-modifiers}}}}
      {axes <axes>} {<title-spec>} {with <style>}
      {, {definitions,} <function> ...}

where either a <function> or the name of a data file enclosed in quotes is
supplied. A function is a mathematical expression or a pair of mathematical
expressions in parametric mode. The expressions may be defined completely or
in part earlier in the stream of 'gnuplot' commands (see 'user-defined').

It is also possible to define functions and parameters on the 'plot' command
itself. This is done merely by isolating them from other items with commas.

There are four possible sets of axes available; the keyword <axes> is used to
select the axes for which a particular line should be scaled. 'x1y1' refers
:
```

図 3.8: コマンド plot のオンラインマニュアル

図 3.8 において、画面の最下段の行に

```
\texttt{:})\prompt
```

というプロンプトが出ていることに注意して下さい。この状態では、ページャ(たいてい less, jless, more のいずれか、環境変数 PAGER によって決まる)が自動的に起動して、マニュアルを表示しています。ですから、そのページャのコマンドはすべて利用可能です。

さて、[Space] キーを押すなどしてマニュアルの最後尾まで画面をスクロールさせると画面は図 3.9 のようになります。図 3.9 では

```
Subtopics available for plot:
```

という文字列に続いてキーワードの一覧が表示されていることに注意して下さい。

There are four possible sets of axes available; the keyword <axes> is used to select the axes for which a particular line should be scaled. 'x1y1' refers to the axes on the bottom and left; 'x2y2' to those on the top and right; 'x1y2' to those on the bottom and right; and 'x2y1' to those on the top and left. Ranges specified on the 'plot' command apply only to the first set of axes (bottom left).

Examples:

```
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
```

Subtopics available for plot:

acsplines	bezier	csplines	datafile
errorbars	every	example	index
parametric	ranges	sbezier	smooth
special-filenames	style	thru	title
unique	using	with	

(END)

図 3.9: コマンド plot のオンラインマニュアルの末尾

図 3.9 の状態で

9

と入力すると、画面は図 3.10 のようになります。

```

There are four possible sets of axes available; the keyword <axes> is used to
select the axes for which a particular line should be scaled. 'x1y1' refers
to the axes on the bottom and left; 'x2y2' to those on the top and right;
'x1y2' to those on the bottom and right; and 'x2y1' to those on the top and
left. Ranges specified on the 'plot' command apply only to the first set of
axes (bottom left).

Examples:
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

Subtopics available for plot:
acsplines      bezier      csplines      datafile
errorbars      every       example       index
parametric     ranges      sbezier       smooth
special-filenames style     thru          title
unique         using      with

Subtopic of plot: █

```

図 3.10: コマンド plot のオンラインマニュアルにおける subtopic の選択

ここで、画面下段の

Subtopic of plot: \prompt

というプロンプトに注意して下さい。 図 3.10 の状態からは、3 種類の操作が可能です。 これらを表 3.3 に示します。

表 3.3: オンラインマニュアルの subtopic 選択画面における操作

機能	コマンド
オンラインマニュアルを抜ける	[Enter]
サブトピックの一覧を表示する	? [Enter]
特定のトピックのマニュアルを見る	トピック [Enter] ただし、「トピック」と書かれたところには、 キーワード一覧で表示された用語のいずれか が入る

たとえば、図 3.10 の状態で、キーワード `with` のマニュアルを見たいときには、

`with` [Enter]

と入力します。すると、画面には、図 3.11 のようにキーワード `with` のマニュアルが表示されます。

```
Functions and data may be displayed in one of a large number of styles.
The 'with' keyword provides the means of selection.

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

where <style> is either 'lines', 'points', 'linespoints', 'impulses', 'dots',
'steps', 'fsteps', 'histeps', 'errorbars', 'xerrorbars', 'yerrorbars',
'xyerrorbars', 'boxes', 'boxerrorbars', 'boxxyerrorbars', 'financebars',
'candlesticks' or 'vector'. Some of these styles require additional
information. See 'set style <style>' for details of each style.

Default styles are chosen with the 'set function style' and 'set data style'
commands.

By default, each function and data file will use a different line type and
point type, up to the maximum number of available types. All terminal
drivers support at least six different point types, and re-use them, in
order, if more are required. The LaTeX driver supplies an additional six
:
```

図 3.11: キーワード `with` のマニュアルを表示しているところ

マニュアルを最後までスクロールさせて

q

と入力すると、図 3.12 のようにサブトピック選択画面に戻ります。

```
This plots "leastsq.dat" with impulses:
plot 'leastsq.dat' w i

This plots the data file "population" with boxes:
plot 'population' with boxes

This plots "exper.dat" with errorbars and lines connecting the points
(errorbars require three or four columns):
plot 'exper.dat' w lines, 'exper.dat' notitle w errorbars

This plots sin(x) and cos(x) with linespoints, using the same line type but
different point types:
plot sin(x) with linesp lt 1 pt 3, cos(x) with linesp lt 1 pt 4

This plots file "data" with points of type 3 and twice usual size:
plot 'data' with points pointtype 3 pointsize 2

This plots two data sets with lines differing only by weight:
plot 'd1' t "good" w l lt 2 lw 3, 'd2' t "bad" w l lt 2 lw 1

See 'set style' to change the default styles.

Subtopic of plot: ■
```

図 3.12: サブトピック選択画面に戻ったところ

3.5 その他のコマンド

3.5.1 条件分岐

コマンド `if` は条件分岐のために用いられます。基本的な構文は

```
if(条件) コマンド
```

というものです。条件全体をカッコで囲む必要があります。

3.5.2 シェルに戻る

`gnuplot` を一旦抜けてシェルでコマンドを実行するために、`shell` というコマンドが用意されています。使い方は

```
shell [Enter]
```

です。

シェルから抜けるときには

```
exit [Enter]
```

と入力します。

3.6 コマンドの省略記法

gnuplot のいろいろなコマンドにはいろいろな省略記法が用意されています。その代表的なものは、「最初の数文字だけを書く」という書き方です。たとえば、

```
plot sin(x) [Enter]
```

と入力しても、

```
p sin(x) [Enter]
```

と入力しても、

```
pl sin(x) [Enter]
```

と入力しても、同一の結果が得られます。すなわち、「p」や「pl」はコマンド `plot` の省略記法とみなされるのです。

上記からわかるように、曖昧さが生じない範囲で、多くのコマンドにいろいろな省略記法が用意されています。ただし、あまり省略記法を多用すると、あとで見直してみると自分でも何をやっているかわからない、ということになることがあるので、省略はほどほどにした方がよいでしょう。

第4章 gnuplot の構文と関数

4.1 構文

gnuplot は、ユーザが入力するコマンドを 1 行ずつ解釈して実行してゆきます。コマンド行には、コマンドおよびオプションやキーワードなどを 1 個以上の空白（スペース）で区切って並べます。キーボードからコマンドを入力しているときには [Tab] キーは効果がありませんが、gnuplot のコマンドをファイルに書いておくときには空白のかわりにタブ文字（[Tab] キーを押すと入力される文字で、4 文字から 8 文字分の空白に見える）も使えます。空白が 2 個以上あるときの効果は空白 1 個の場合と変わりません。なお、いわゆる全角スペースを空白として使うことはできません。

入力するオプションなどに文字列が含まれるときには、その文字列を単一引用符（「'」）で囲みます。たいていの場合は単一引用符のかわりに 2 重引用符（「"」）を使うこともできますが、例外もあるので、つねに単一引用符を使うほうが良いでしょう。

オプションなどで x 座標と y 座標を指定する必要があるときには、それらをコンマで区切って並べます。同様に、x 座標と y 座標および z 座標を指定する必要があるときにも、それらをコンマで区切って並べます。

gnuplot のコマンドは改行キーを押したときに確定します。

行頭が文字「#」で始まる行は注釈とみなされます。gnuplot は注釈を無視するので、3.1.3 節に述べたようにあらかじめ gnuplot のコマンドが書かれたファイルを作っておいてそれを実行する場合には、文字「#」で始まる行にいろいろなコメントを書いておいて、あとでそのファイルを読み直すときに便利できるようにしておくことができます。

行の途中に注釈を入れることもできます。そのためには、行中に文字「#」を入れます。すると、gnuplot は文字「#」から改行文字までの部分をすべて無視します。

先に述べたように、gnuplot のコマンドは改行キーを押したときに確定するのですが、gnuplot のコマンド中に記号「\」（バックスラッシュ）を入力してそれに続いて [Enter] キーを押すと、コマンドは実行されず、かわりに「>」という記号が行の先頭にあらわれます。

たとえば、

```
plot \ [Enter]
```

と入力すると、gnuplot の画面は図 4.1 のような状態になります。図 4.1 の 2 行目で、



図 4.1: 記号 \ を使ったときの状態

```
gnuplot>
```

というプロンプトのかわりに

```
>
```

というプロンプトが出ていることに注意して下さい。

この状態で、次に

```
sin(x) [Enter]
```

と入力すると、

```
plot sin(x) [Enter]
```

と入力した場合と同じ内容が実行されます。すなわち、

```
\ [Enter]
```

とすると、改行文字の入力確定という意味をなくすことができます。

この機能は、3.1.3 節に述べたように、あらかじめ gnuplot のコマンドが書かれたファイルを作っておいてコマンド load を使ってそれを実行する場合に便利です。ファイルに書くコマンドが長くなってしまった場合は、コマンド中に適宜「\」を挿入して、ファイルを見やすくすることができます。

図 4.2 に、長すぎるコマンドを「\」を使って見やすくしている例を示します。

```
plot [-2*pi:2*pi] [-1:1] sin(x)
title ' 正弦関数'
with linespoints
linetype 3 linewidth 4
pointtype 4 pointsize 4
```

図 4.2: コマンドの途中における改行

なお、記号「\」（バックスラッシュ）は端末の状態によっては「¥」記号に表示されることもあり、キーボードにも「¥」記号の方が書いてあることがあります。

4.2 数式と組み込み演算子

gnuplot には、表 4.1 のような演算子があります。

演算子の優先順位が変わる部分に横線が引いてあります。横線が引かれた部分の上側は優先順位が高く、下側は優先順位が低くなります。

たとえば、

```
x**4- 2*x + 3
```

と書いた場合、演算子**の方が優先順位が高いので、これは

```
(x**4)- 2*x + 3,
```

すなわち

$$x^4 - 2x + 3$$

表 4.1: gnuplot の演算子

記号	意味
**	べき [†] (x**2 は x の 2 乗)
-	単項の負演算子 [†]
~	ビットごとの否定
!	論理否定
!	階乗
*	乗算 [†]
/	除算 [†]
%	剰余
+	加算 [†]
-	減算 [†]
==	等値
!=	非等値
&	ビットごとの論理積 (and)
^	ビットごとの排他論理和 (xor)
	ビットごとの論理和 (and)
&&	論理積 (and)
	論理和 (or)
?:	条件演算子

と解釈されます。

優先順位はカッコ () をつけることで変更できます。演算子の優先順位はあまり記憶しやすいものではないので、いちいち記憶するよりは、積極的にカッコを使って優先順位を明示した方がよいでしょう。

「意味」と書かれた欄に記号[†]の付いた演算子は整数、実数、複素数に対して適用可能ですが、それ以外は整数に対してのみ適用可能です。

4.3 数値の演算の結果を画面に表示する

数値の演算の結果を画面に表示するには、3.3.3 節で紹介されたコマンド `print` を使います。

たとえば、2 の平方根の数値を画面に表示したいときには、

```
print sqrt(2) [Enter]
```

と入力します。

結果は図 4.3 のようになります。

4.4 ユーザ変数

ユーザ変数とは利用者が自分で値を定義する変数のことをいいます。本節ではユーザ変数の操作の仕方について説明します。

```
gnuplot> print sqrt(2)
1.4142135623731
gnuplot> █
```

図 4.3: 2 の平方根の値が画面に表示されたところ

4.4.1 ユーザ変数の一覧を見る

ユーザ変数の定義の方法の説明に入る前に、現在定義されているユーザ変数の一覧を表示する方法を説明しておきましょう。

ユーザ変数の一覧を表示するには、gnuplot のウィンドウで

```
show variables [Enter]
```

と入力します。

gnuplot を立ち上げた直後の状態では、上記を実行した結果は図 4.4 のようになります。初期状態では `pi` と

```
gnuplot> show variables
      Variables:
      pi = 3.14159265358979
gnuplot> █
```

図 4.4: ユーザ変数の初期状態

いうユーザ変数が定義されていて、それが円周率 π をあらわしていることがわかります。

4.4.2 ユーザ変数を新しく定義する

次に、新しくユーザ変数を定義する方法を説明しましょう。

新しいユーザ変数を定義するには、

```
変数名 = 数値 [Enter]
```

という構文を用います。

変数名の中で使える文字は、英文字の大文字と小文字および文字「`_`」(アンダースコア)です。大文字と小文字は区別されます。空白は使えません。変数名は、これらの利用可能な文字を有限個並べたものになります。

たとえば、ネイピアの数 e (約 2.71828) を定義したいときには、

```
e=2.71828 [Enter]
```

とします。

```
e=2.71828 [Enter]
show variables [Enter]
```

を実行すると、結果は図 4.5 のようになります。図 4.5 からユーザ変数 e が追加されていることがわかります。

```
gnuplot> e=2.71828
gnuplot> show variables

Variables:
pi = 3.14159265358979
e = 2.71828

gnuplot> █
```

図 4.5: ユーザ変数 e が追加された状態

グラフを描くときによく使うユーザ変数があるときには、キーボードから毎回読み込むのは無駄ですし、誤入力の危険もあるので、あらかじめユーザ変数をまとめたファイルを用意しておいて、3.1.3 節で述べられている手順にしたがってファイルを読み込むとよいでしょう。

4.4.3 ユーザ変数をファイルに保存する

現在定義されているユーザ変数をすべてファイルに保存することもできます。このためには、`save var` というコマンドを使います。

コマンド `save var` の使い方は、

```
save var 'ファイル' [Enter]
```

です。ここに、「ファイル」と書かれた部分には適当なファイルの名前が入ります。ファイルの名前に日本語や空白などを含めると問題が生じることがあります。ファイル名には英文字と記号「`_`」、「`-`」、「`.`」以外は使わないようすれば安全です。

たとえば、

```
e=2.71828 [Enter]
save var 'test.gp' [Enter]
quit [Enter]
```

とすると、現在作業中のディレクトリに `test.gp` というファイルができます。このファイルの内容は図 4.6 のようになります。

図 4.6 において、最初の部分はこのファイルが `gnuplot` のコマンドであることを示す部分、その下の行頭が文字「`#`」で始まる部分はすべて注釈で、最後から 2 行目の部分にユーザ変数の情報が書かれています。最後の行には「`# EOF`」というコメントが書かれています。

このようにして保存されたファイルは第 3.1.3 節で解説されているコマンド `load` を使って読み込むことができます。ファイルを読み込むと、そこで定義されているユーザ変数が使えるようになります。

```
% pwd
/home/hamba/text/education/00gnuplot
% jless test.gp
#!/usr/local/bin/gnuplot -persist
#
#
#      G N U P L O T
#      Unix version 3.7
#      patchlevel 1 (+1.2.0 2001/01/11)
#      last modified Fri Oct 22 18:00:00 BST 1999
#
#      Copyright (C) 1986 - 1993, 1998, 1999
#      Thomas Williams, Colin Kelley and many others
#
#      Type 'help' to access the on-line reference manual
#      The gnuplot FAQ is available from
#      <http://www.ucc.ie/gnuplot/gnuplot-faq.html>
#
#      Send comments and requests for help to <info-gnuplot@dartmouth.edu>
#      Send bugs, suggestions and mods to <bug-gnuplot@dartmouth.edu>
#
e = 2.71828
# EOF
test.gp (END)
```

図 4.6: コマンド `save var` によって生成されるファイルの内容

4.5 複素数の表現

gnuplot は複素数を扱うことができます。複素数を表現するときには、中括弧 `{}` の中に実部と虚部をコンマで区切って指定します。

例えば、 $4+3i$ は gnuplot では `{4,3}` と表記されます。

複素数を用いてグラフを描画するときには虚数単位がよく使われるので、第 4.4 節の手順にしたがって虚数単位に対応するユーザ変数を用意しておくといよいでしょう。

たとえば、 I (大文字の i) で虚数単位をあらわすことにする場合には、

`I = {0,1}`

とします。

4.6 数学関数

gnuplot にはいろいろな数学関数が用意されています。この節では、これらを簡単に紹介します。

4.6.1 指数関数, 対数関数, 三角関数

gnuplot に用意されている指数関数および対数関数の一覧を表 4.2 にまとめておきます。

表 4.2 の関数の意味は名前から明らかだとは思われますが、代表的なものについて簡単に説明します。

指数関数は `exp` という名前です。引数としては実数あるいは複素数を取ることができます。

対数関数には、自然対数 `log` と常用対数 `log10` の 2 種類があります。

三角関数の名前はそれぞれ `sin` (正弦関数), `cos` (余弦関数), `tan` (正接関数) です。gnuplot を起動したときには単位としてラジアンが指定されていますが、単位を度に変更することができます。

逆三角関数としては、`asin`, `acos`, `atan` の 3 種類が用意されています。逆三角関数は返却値の単位をラジアンあるいは度に変更できます。

単位を度とラジアンで変更するには、`set angles` というコマンドを使います。

表 4.2: 指数関数, 対数関数, 三角関数

関数	引数の型	意味
$\exp(x)$	任意	指数関数
$\log(x)$	任意	自然対数
$\log_{10}(x)$	任意	常用対数
$\sin(x)$	任意	正弦関数
$\cos(x)$	任意	余弦関数
$\tan(x)$	任意	正接関数
$\operatorname{asin}(x)$	任意	逆正弦関数
$\operatorname{acos}(x)$	任意	逆余弦関数
$\operatorname{atan}(x)$	任意	逆正接関数
$\operatorname{atan2}(y, x)$	整数, 実数	$\tan^{-1} \frac{y}{x}$ の実部 x の実部 を返す

```
set angles degrees [Enter]
```

と入力すると単位が度に変更され、

```
set angles radians [Enter]
```

と入力すると単位がラジアンに変更されます。

1 回単位をラジアンと度のあいだで変更すると、その後は（自分でもう 1 回単位を変更するまで）ずっとその単位が有効になります。

なお、正割関数、余割関数、余接関数とその逆関数を用意されていません。

4.6.2 算術関数

gnuplot には 実数や複素数の演算や整数への変換のためのいろいろな関数が用意されています。 これらを表 4.3 にまとめておきます。

4.6.3 その他の関数

gnuplot には、上に述べたもの以外にもいろいろな数学関数が用意されています。これらを表 4.4 にまとめておきます。

以下に書いたもののうち、双曲線関数とベッセル関数の単位はラジアンに固定されているので注意して下さい。

4.7 ユーザ関数の定義

ユーザ関数とは利用者が自分で値を定義する関数のことをいいます。本節ではユーザ関数の操作の仕方について説明します。

表 4.3: 実数, 複素数の演算と整数への変換

関数	引数の型	意味
<code>abs(x)</code>	任意	絶対値関数. 引数の絶対値を返す
<code>sgn(x)</code>	任意	符号関数. 引数の実部が正なら 1, 負なら 0 を返す. 引数の実部が 0 のときは 0 を返す
<code>arg(x)</code>	複素数	複素数の偏角を返す. 返却値の単位はラジアンか度のいずれか. <code>set angles</code> によって返却値の単位をラジアンにするか度にするかを選択する
<code>sqrt(x)</code>	任意	引数の平方根を返す
<code>ceil(x)</code>	任意	引数以上の最小の整数を返す. 引数が複素数のときにはその実部に対して作用する
<code>floor(x)</code>	任意	引数以下の最大の整数を返す
<code>int(x)</code>	実数	引数が正のときは引数以下の最大の整数を, 引数が負のときは引数以上の最小の整数を返す
<code>real(x)</code>	任意	複素数の引数の実部を返す
<code>imag(x)</code>	複素数	複素数の引数の虚部を返す

表 4.4: その他の関数

関数	引数の型	意味
<code>sinh(x)</code>	任意	双曲正弦関数.
<code>cosh(x)</code>	任意	双曲余接関数
<code>tanh(x)</code>	任意	双曲正接関数
<code>asinh(x)</code>	任意	逆双曲正弦関数
<code>acosh(x)</code>	任意	逆双曲余弦関数
<code>atanh(x)</code>	任意	逆双曲正接関数
<code>gamma(x)</code>	任意	引数の実部にガンマ関数を作用させたものを返す
<code>igamma(x)</code>	任意	引数の実部に不完全ガンマ関数を作用させたものを返す
<code>lgamma(x)</code>	任意	引数の実部にガンマ関数を作用させてから自然対数を取ったものを返す
<code>ibeta(x)</code>	任意	引数の実部に不完全ベータ関数を作用させたものを返す
<code>besj0(x)</code>	整数, 実数	0 次のベッセル J 関数
<code>besj1(x)</code>	整数, 実数	1 次のベッセル J 関数
<code>besy0(x)</code>	整数, 実数	0 次のベッセル Y 関数
<code>besy1(x)</code>	整数, 実数	1 次のベッセル Y 関数
<code>rand(x)</code>	任意	引数の実部を種と, 0 から 1 のあいだに分布する擬似乱数を返す
<code>erf(x)</code>	任意	引数の実部にオイラー関数を作用させたものを返す
<code>erfc(x)</code>	任意	$1 - \text{erf}(x)$ を返す
<code>inverf(x)</code>	任意	引数の実部にオイラー関数の逆関数を作用させたものを返す
<code>norm(x)</code>	任意	引数の実部に正規分布関数を作用させたものを返す
<code>invnorm</code>	任意	引数の実部に逆正規分布関数を作用させたものを返す

4.7.1 ユーザ関数の一覧を見る

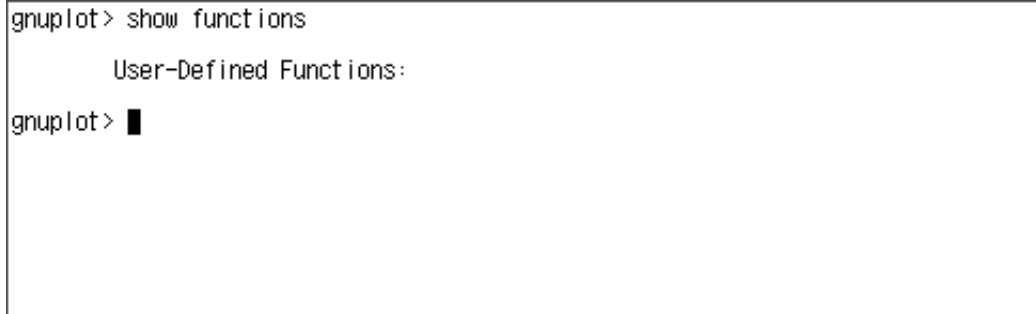
ユーザ関数の定義の方法の説明に入る前に, 現在定義されているユーザ関数の一覧を表示する方法を説明しておきましょう.

ユーザ関数の一覧を表示するには、gnuplot のウィンドウで

```
show functions [Enter]
```

と入力します。

gnuplot を立ち上げた直後の状態では、上記を実行した結果は図 4.7 のようになります。初期状態ではユー



```
gnuplot> show functions
      User-Defined Functions:
gnuplot> █
```

図 4.7: ユーザ関数の初期状態

ザ関数は何も定義されていないことがわかります。

4.7.2 ユーザ関数を新しく定義する

次に、新しくユーザ関数を定義する方法を説明しましょう。

ユーザ関数を定義するときには、

```
関数名 (x)=定義式 [Enter]
```

```
関数名 (x,y)=定義式 [Enter]
```

というような内容を入力します。

関数名の中で使える文字は、英文字の大文字と小文字および文字「_」(アンダースコア)です。大文字と小文字は区別されます。空白は使えません。変数名は、これらの利用可能な文字を有限個並べたものになります。

変数の部分には x , y , z 以外の文字を使ってもよいのですが、特別な理由がある場合を除き x , y , z と指定した方がよいでしょう。

たとえば、余接関数 $\cot x$ を定義するときには、

```
cot(x)=1/tan(x) [Enter]
```

とします。

```
cot(x)=1/tan(x) [Enter]
```

```
show functions [Enter]
```

というコマンドを実行した結果を図 4.8 に示します。

4.7.3 ユーザ関数を使う

一旦定義されたユーザ関数は、gnuplot に標準で用意されている関数と同じように使えます。ただし、関数を使うときには、その関数を定義するときになどどのような変数を使ったかにかかわらず、変数としてはその文脈に適した文字を使う必要があります。

たとえば、

```
gnuplot> cot(x)=1/tan(x)
gnuplot> show functions

      User-Defined Functions:
      cot(x)=1/tan(x)

gnuplot> █
```

図 4.8: 新しい関数が定義された状態

```
cot(w)=1/tan(w) [Enter]
```

と定義した上で、ここで定義された関数 `cot` のグラフをプロットするときには、

```
plot cot(x) [Enter]
```

とします。

```
plot cot(w) [Enter]
```

とするとエラーになるので注意して下さい。関数を使うときには、その関数の変数名はつねにその文脈（2次元か3次元か、パラメータ付きか否か、これらについては後述）に応じて適合したものに置き換えられてゆくのです。

4.7.4 関数を使うときに変数名を変更する

関数を使うときにその関数の変数名を変更したい場合には、`set dummy` というコマンドを使います。

コマンド `set dummy` の使い方は、

```
set dummy 変数名 [Enter]
```

です。2変数関数を使うときには、

```
set dummy 変数1, 変数2 [Enter]
```

とします。

たとえば

```
set dummy time [Enter]
```

```
plot sin(time) [Enter]
```

とすると、

```
plot sin(x)
```

とした場合と同様に正弦波のグラフが描画できます。

同様に、

```
set dummy h,v [Enter]
```

```
splot sin(h+v) [Enter]
```

とすると、

```
splot sin(x+y) [Enter]
```

とした場合と同様のグラフが描画されます。

4.7.5 ユーザ関数をファイルに保存する

現在定義されているユーザ関数をすべてファイルに保存することもできます。このためには、`save functions` というコマンドを使います。

コマンド `save functions` の使い方は、

```
save functions 'ファイル' [Enter]
```

です。ここに、「ファイル」と書かれた部分には適当なファイルの名前が入ります。ファイルの名前に日本語や空白などを含めると問題が生じることがあります。ファイル名には英文字と記号「_」、「-」、「.」以外は使わないようすれば安全です。

このようにして保存されたファイルはコマンド `load` を使って読み込むことができます。ファイルを読み込むと、そこで定義されているユーザ関数が使えるようになります。

第5章 グラフの印刷とターミナルの切り換え

5.1 グラフの確認から印刷までの一連の流れ

gnuplot でグラフを確認してから印刷するまでの作業の流れは、

1. 関数をプロットする
2. ターミナルを印刷用データのものに切り変える
3. 出力ファイルを指定する
4. 画像をファイルに保存する
5. ターミナルを標準的な画面に戻す

というものです。

参考のために、

1. 正弦関数をプロットする
2. ターミナルを PostScript の eps モードに切り換える
3. 出力を test.eps というファイルに変える
4. 画像をファイルに保存する
5. ターミナルを X Window System に戻す

という一連の作業を実行している例を図 5.1 に示しておきます。

```
gnuplot> plot sin(x)
gnuplot> set terminal postscript eps
Terminal type set to 'postscript'
Options are 'eps noenhanced monochrome dashed defaultplex "Helvetica-Ryumin" 14'
gnuplot> set output 'test.eps'
gnuplot> replot
gnuplot> set terminal x11
Terminal type set to 'x11'
Options are '0'
gnuplot> █
```

図 5.1: gnuplot の標準的な印刷の流れ

関数やコマンド、ファイル名やターミナルなどが変わることがあっても、図 5.1 に示された作業の流れはだいたい共通です。ですから、この作業の流れは記憶しておいて下さい。

5.2 ターミナルと出力ファイルの指定

gnuplot が起動したときには、ターミナル（画像を表示する装置）が X Window System になっています。グラフを印刷する作業は、

- ターミナルを X Window System から印刷用のものに切り換える
- 出力ファイルなどを指定する

という 2 工程から成ります。

ターミナルの切り換えには `set terminal` というコマンドを使い、出力ファイルなどの指定には `set output` というコマンドを使います。

5.2.1 ターミナルの切り換え

コマンド `set terminal` は、

`set terminal` ターミナルの名前 [Enter]

のようにして使います。「ターミナルの名前」の部分にて指定できるものの中で代表的なものを表 5.1 にまとめておきます。

表 5.1: ターミナルの種類

ターミナルの名前	説明
x11	X Window System
postscript	PostScript 形式, 代表的な印刷用フォーマットのひとつ
tgif	tgif のオブジェクトファイル, 描画ツール tgif で使うための形式
latex	文書作成用ツール L ^A T _E X に取り込むための形式
table	関数の値を表にして作成
png	png 形式, 代表的な印刷用フォーマットのひとつ
pbm	pbm 形式, 代表的な印刷用フォーマットのひとつ

これらの使い方については次節以降で説明することにして、ここではターミナルを PostScript の `eps` モードに切り換える例を示すにとどめます。

ターミナルを PostScript の `eps` モードに切り換えるには、

`set terminal postscript eps` [Enter]

とすると、gnuplot のウィンドウは図 5.2 のような状態になり、表示されたメッセージからターミナルが PostScript の `eps` モードになっていることが確認できます。

この状態のとき、グラフは画像として確認できるフォーマットではなく、PostScript 形式（印刷用のフォーマットで）作成されます。

印刷用のフォーマットのデータをファイルに保存するためには、これに続いて次節で述べるコマンド `set output` で出力ファイルを指定する必要があります。コマンド `set output` を使うのを忘れると、作成されたデータは標準出力 (gnuplot のウィンドウ) にそのままテストとして表示されてしまいます。

上に述べたように、ターミナルが `postscript` になっていると、いくらグラフを描画しても作成された図は画面上では確認できません。

ファイルの保存が終わってもう 1 回画面で画像が確認できる状態に戻りたいときには、gnuplot のウィンドウ内で

```
gnuplot> set terminal postscript eps
Terminal type set to 'postscript'
Options are 'eps noenhanced monochrome dashed defaultplex "Helvetica-Ryumin" 14'
gnuplot> █
```

図 5.2: ターミナルを PostScript の eps モードに切り換えた状態

```
set terminal x11 [Enter]
```

と入力します。すると、gnuplot のウィンドウは図 5.2 のような状態になり、ターミナルが x11 になったことが確認できます。

```
gnuplot> set terminal x11
Terminal type set to 'x11'
Options are '0'
gnuplot> █
```

図 5.3: ターミナルを X11 の eps モードに切り換えた状態

ターミナルが x11 になっているときには、出力ファイルを指定する必要はありません。また、コマンド `set output` で出力ファイルの名前が指定されていても無視されます。

5.2.2 出力ファイルの指定

出力ファイルを指定するときには、コマンド `set output` を使います。

コマンド `set output` の使い方は、

```
set output 'ファイル' [Enter]
```

とします。ここに、「ファイル」と書かれた部分には適当なファイル名が入ります。

現在の出力の設定を見るには、コマンド `show output` を使います。

コマンド `show output` の使い方は、

```
show output 'ファイル' [Enter]
```

です。

ターミナルと出力ファイルが正しく設定された状態でコマンド `plot` あるいは コマンド `splot` を実行すると、グラフが指定されたファイルに保存されます。

また、1 回コマンド `plot` あるいは `splot` を実行してグラフを画面で確認してからファイルに保存する場合には、ターミナルと出力ファイルを指定したあとでコマンド `replot` を実行する、すなわち

```
replot [Enter]
```

と入力すると、先ほど確認したものと同じグラフがファイルに保存されます。

ターミナル が `x11` 以外になっているときには、コマンド `set output` で出力ファイルを設定することが必須です。忘れないようにして下さい。

5.3 いろいろなターミナル

本節では、いろいろなターミナルとそのオプションについて説明してゆきます。

5.3.1 x11

X Window System でグラフを画面上で確認するためのターミナルです。

```
set terminal x11 [Enter]
```

とするとこのウィンドウを選択できます。

グラフを表示するウィンドウを複数作りたいときには、

```
set terminal x11 番号 [Enter]
```

とします。ここに、「番号」と書かれた部分には 0 以上の数を指定します。ただし、数 0 を指定することと番号を指定しないことの効果は同じです。

たとえば、

```
set terminal x11
plot sin(x)
set terminal x11 1
plot cos(x)
set terminal x11 2
plot exp(x)
```

とすると、画面には「Gnuplot」「Gnuplot 1」「Gnuplot 2」と書かれた 3 枚のウィンドウが開きます (図 5.4)。

それぞれのウィンドウにマウスカーソルを合わせて

```
q
```

と入力すると、そのウィンドウを閉じることができます。また、

```
set terminal x11 reset
```

と入力すると、開いているすべての画像ウィンドウを閉じることができます。

`gnuplot` を終了すると、`gnuplot` が作成した画像のウィンドウもすべて閉じます。このウィンドウを閉じないようにする (`gnuplot` が終了してから画面に残っている) ように変更したいときには、`gnuplot` を起動するときに

```
gnuplot -persist [Enter]
```

のように、`gnuplot` に `persist` というオプションを付けます。

ターミナル `x11` では日本語を問題なく取り扱うことができます。

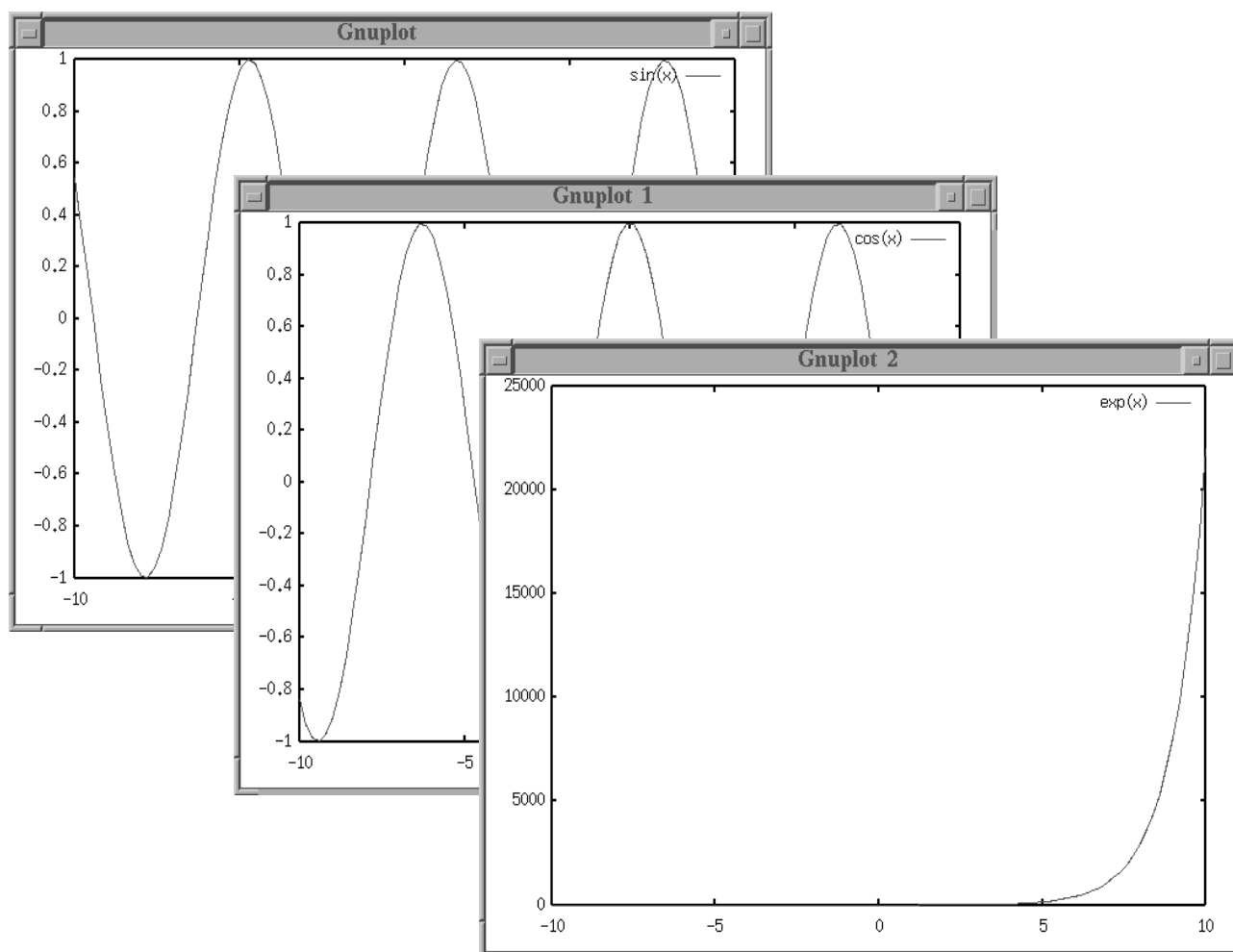


図 5.4: gnuplot で複数の画像ウィンドウを開いたところ

5.3.2 postscript

PostScript は文字や画像などのデータを保存する形式の代表的なもののひとつです。gnuplot では、PostScript 形式は、postscript という名前のターミナルで利用できます。

ターミナルを postscript に切り換えるには、

```
set terminal postscript モード オプション [Enter]
```

と入力します。ここに、「モード」と「オプション」には後で説明するモードとオプションに対応する文字が入ります。モードおよびオプションはすべて省略できますが、ほかの文書に取り込む画像を作成しているときにはモードを eps にしておいた方がよいでしょう。

ターミナルが postscript になっているときは、コマンド set output で出力ファイルを設定することが必須です。忘れないようにして下さい。postscript 形式のファイルの拡張子は通常 eps あるいは ps です。

5.3.2.1 モード

postscript には eps モード (他の文書に取り込む図の作成用)、portrait モード (用紙縦置き)、landscape (用紙横置き)、default (デフォルト) の 4 種類のモードがあります。これらを表 5.2 にまとめておきます。

表 5.2: ターミナル postscript のモード

モードの名称	意味
eps	encapsulated PostScript モード (他の文書に取り込む図の作成用)
portrait	用紙縦置きモード
landscape	用紙横置きモード
default	標準

5.3.2.2 オプション

postscript モードにはいくつかオプションがあるのですが、これらの中で利用頻度が高いと思われるものを以下にまとめておきます。表の中で、横線を引いてまとめてあるオプションを同時に指定することはできませんが、他のグループにあるオプションについては複数組み合わせられます。なお、表 5.3 の左端に「標準」と書いてある項目がターミナルを postscript に変更したときに自動的に選択されるオプションです。これらの値を変更する必要がないときには、オプションを指定しなくてもよいです。

なお、表 5.3 にある plus モードは図の縦軸や横軸に自由に数式などを入れることができる強力なオプションです。この plus モードは gnuplot に対するパッチ (差分ファイル) として gnuplot の開発元とは別の開発者が提供しているもので、標準の gnuplot には含まれません。ですから、gnuplot がインストールされた状態によっては、plus モードの機能が使えないこともあります。plus モードに関する詳しい説明をすると本節の主題から逸脱してしまうので、これについては第 10 章で改めて取り上げることにします。

表 5.3: ターミナル postscript の代表的なオプション

オプション	意味
noplus	数式などを取り扱うための拡張モード (10 章参照) を無効にする
plus	数式などを取り扱うための拡張モードを有効にする
nocolor	白黒画像を生成
color	カラー画像を生成
dashed	点線をそのまま保存する
solid	点線を実線で置き換える (カラー画像で使われることがある)
数	フォントの大きさを指定

ターミナル postscript では日本語を問題なく取り扱うことができます。

5.3.3 tgif

描画ツール tgif の形式のターミナルが tgif です。ターミナルを tgif に変更するには、

```
set terminal tgif [Enter]
```

と入力します。

ただし、ターミナルを tgif にする場合で、図に日本語の文字が含まれているときには、必ず

```
set terminal tgif 'Ryumin-Light-EUC-H' [Enter]
```

のように入力して下さい。この理由は以下で述べます。

また、ターミナルが tgif になっているときは、コマンド set output で出力ファイルを設定することが必須です。忘れないようにして下さい。tgif のファイルの拡張子は obj です。

ターミナル `tgif` にもいろいろなオプションがあります。これらを表 5.4 にまとめておきます。

表 5.4: ターミナル `tgif` のオプション

オプション	意味	
<code>portrait</code>	用紙縦置き	標準
<code>landscape</code>	用紙横置き	
<code>dashed</code>	点線をそのまま保存する	標準
<code>solid</code>	点線を実線で置き換える (カラー画像で使われることがある)	
'フォント'	フォントの名前を指定	
数	フォントの大きさを指定	

ここで1点注意すべきことがあります。ターミナル `tgif` では、フォントを正しく設定しないと、日本語が表示できず、文字化けします。日本語を表示するためのフォントは、やや名前が長いのですが、`Ryumin-Light-EUC-H` です。ですから、ターミナルを `tgif` にする場合で、図に日本語の文字が含まれているときには、日本語フォントの指定は必須です。このような場合には、必ず

```
set terminal tgif 'Ryumin-Light-EUC-H,Helvetica' [Enter]
```

のように明示的にフォントを指定して下さい。他のオプションを指定するか否かは任意です。

なお、これはおそらくバグと思われるのですが、`terminal` を `tgif` にした状態では、日本語の文字と英数字が混在したグラフを正しく保存することができません。また、`tgif` 形式で保存したグラフを `tgif` で編集した場合、

- 日本語の部分の編集が正しくできない
- PostScript 形式で保存すると文字化けする

という問題が生じることがあります。グラフを `tgif` 形式で保存する場合には、`gnuplot` で描画するときにはグラフには日本語を入れずにおき、日本語入力および編集作業はすべて `tgif` でおこなうのが良いでしょう。

5.3.4 latex

文書作成ツール \LaTeX のソースコードを生成するターミナルが `latex` です。ターミナルを `latex` にすると、グラフが \LaTeX の `picture` 環境という環境の形式で生成されます。このようにして生成されたファイルを \LaTeX で作成された文書にそのまま挿入することで、図入りの文書が作成できます。

ターミナルが `latex` になっているときは、コマンド `set output` で出力ファイルを設定することが必須です。忘れないようにして下さい。 `latex` のファイルの拡張子は `tex` です。

\LaTeX のピクチャー環境については \LaTeX のマニュアルを参照して下さい。

5.3.5 table

ターミナルを `table` にすると、グラフを描くかわりにグラフを描くために計算されたいろいろな点の座標と、それに対応する関数の値が、テキスト形式の表の形で書き出されます。

ターミナルが `table` になっているときは、コマンド `set output` で出力ファイルを設定することが必須です。忘れないようにして下さい。

5.3.6 png

ターミナルを png にすると、代表的な画像フォーマットのひとつである Portable Network Graphics(png) フォーマットで画像を保存することができます。

ターミナルが png になっているときは、コマンド `set output` で出力ファイルを設定することが必須です。忘れないようにして下さい。ファイルの拡張子は通常 png です。

ターミナル png には表 5.5 のようなオプションがあります。ただし、大きいフォントを選択すると日本語が

表 5.5: ターミナル png のオプション

オプション	意味
small	小さいフォント
medium	中サイズのフォント
large	大きいフォント
monochrome	白黒
gray	グレースケール
color	カラー

表示されなくなるので注意して下さい。

5.3.7 pbm

ターミナルを pbm にすると、代表的な画像フォーマットのひとつである Portable BitMap(pbm) フォーマットで画像を保存することができます。

ターミナルが pbm になっているときは、コマンド `set output` で出力ファイルを設定することが必須です。忘れないようにして下さい。ファイルの拡張子は通常 pbm です。

ターミナル pbm のオプションには png ファイルと同様のオプションがあります。これについては表 5.5 を参照して下さい。なお、大きいフォントを選択すると日本語が表示されなくなるので注意して下さい。

第6章 gnuplot のグラフの構造

6.1 グラフの各部の名称

6.1.1 2次元グラフの各部の名称

図 6.1 に gnuplot が描画した 2 次元グラフの主要部の名称を示します。

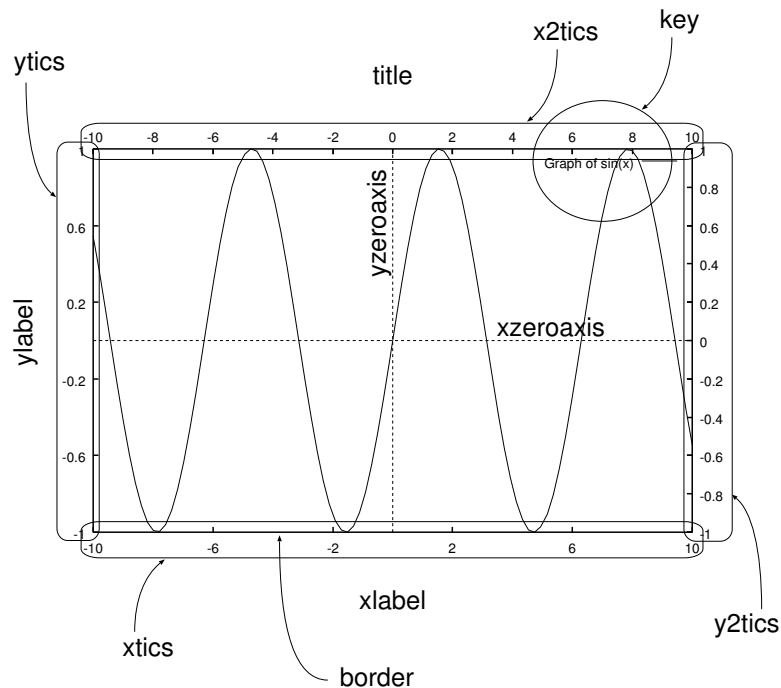


図 6.1: 2 次元グラフの各部の名称

図 6.1 の各項目の簡単な説明などを表 6.1 にまとめておきます。なお、図中には記載がありませんが、`zeroaxis` というキーワードは x 軸と y 軸の双方を指します。

表 6.1: 2 次元グラフの各部の名称の説明

項目	説明	項目	説明	項目	説明
title	グラフの標題	xlabel	x 軸の説明	ylabel	y 軸の説明
key	点や曲線の説明	xtics	x 軸の主目盛り	ytics	y 軸の主目盛り
border	グラフ全体を囲む箱	x2tics	x 軸の第 2 目盛り	y2tics	y 軸の第 2 目盛り
zeroaxis	座標軸 (x 軸と y 軸)	xzeroaxis	座標軸 (x 軸)	yzeroaxis	座標軸 (y 軸)

6.1.2 3次元グラフの各部の名称

図 6.2 に gnuplot が描画した 3 次元グラフの主要部の名称を示します。

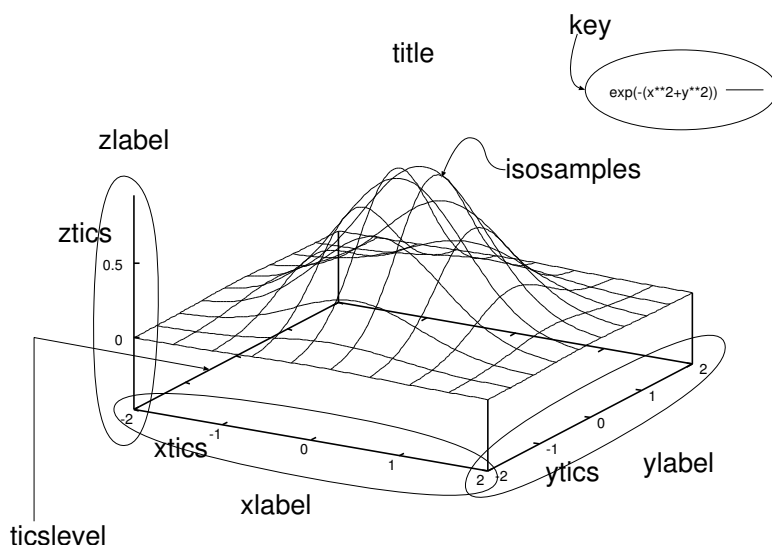


図 6.2: 3 次元グラフの各部の名称

図 6.2 の各項目の簡単な説明などを表 6.2 にまとめておきます。なお、見にくくなるために図中には示されていませんが、2 次元グラフと同様に、グラフを囲む箱 (3 次元の場合はグラフの上部は囲われない) を `border` といいます。

表 6.2: 3 次元グラフの各部の名称の説明

項目	説明	項目	説明	項目	説明
<code>title</code>	グラフの標題	<code>xlabel</code>	x 軸の説明	<code>zlabel</code>	z 軸の説明
<code>key</code>	点や曲線の説明	<code>xticks</code>	x 軸の目盛り	<code>zticks</code>	z 軸の目盛り
<code>border</code>	グラフを囲む箱	<code>ylabel</code>	y 軸の説明		
<code>isosamples</code>	グラフの網かけ	<code>yticks</code>	y 軸の目盛り		

2 次元グラフは x 軸と y 軸にそれぞれ 2 種類の目盛り (`xticks` と `x2ticks`, `yticks` と `y2ticks`) を持つのに対し、3 次元グラフでは各座標軸にそれぞれ 1 種類の目盛りしか存在しないことを注意しておきます。

6.2 グラフを調整する – `set` と `with` –

グラフはいろいろな目的で作成されますし、利用者が作成したいグラフの形状は目的によって大きく異なります。このようないろいろな利用者の要求に答えるために、gnuplot にはグラフを調整する様々なオプションが用意されています。これらのオプションを設定するには、

- キーワード `with` に続いてオプションを指定する
- コマンド `set` を用いて設定する

という 2 種類の方法のいずれかを用います。

コマンド `set` を用いて設定された事項を確認するときには、コマンド `show` を使います。

6.2.1 キーワード with を使う

キーワード `with` を使うと、

- グラフのスタイル
- 線の種類
- 線の太さ
- 点の種類
- 点の大きさ

を指定することができます。グラフのスタイルに関する説明は後回しにして、まず、線の種類、線の太さ、点の種類、点の大きさについて説明しておきます。

線の種類、線の太さ、点の種類、点の大きさを指定するときには、表 6.3 に示したようなキーワードを使います。これらを指定するときには、たとえば

表 6.3: 線と点を指定するキーワード

指定する事項	キーワード	省略記法
線の種類	linetype	lt
線の太さ	linewidth	lw
点の種類	pointtype	pt
点の大きさ	pointsize	ps

```
plot sin(x) with lines linetype 3 linewidth 10 [Enter]
```

のようにキーワード `with` に続いて必要なキーワードおよびそのキーワードに対応した数値を入力してゆくのですが、ここでひとつ注意すべきことがあります。それは何かというと、

指定するキーワードの順番は決まっている

ということです。具体的に言うと、表 6.3 のキーワードを指定するときには、表 6.3 と同じ順番を守らなければなりません。指定する必要のないキーワードを飛ばすことはできるのですが、順番を入れ換えることはできません。ですから、たとえば

```
plot sin(x) with lines linewidth 10 linetype 3 [Enter]
```

と入力するとエラーメッセージが出ます。次に、先ほど予告したグラフのスタイルについて説明しましょう。gnuplot では、表 6.4 に示されているようなグラフのスタイルが使えます。ただし、これらの中には、2 次元グラフを描画するときのみ有効なものがあります。

6.2.2 コマンド set と show を使う

コマンド `set` を使うと、作図にかかわるいろいろな事項を設定することができます。コマンド `set` を用いて設定された内容を確認するには、コマンド `show` を用います。

コマンド `set` を使うときには、

```
set 事項の名前 設定の内容 [Enter]
```

表 6.4: グラフのスタイル

スタイル名	機能
lines	データ点のあいだを線で結ぶ (データ点には小さい点を打つ).
points	データ点に適切な図記号を配置する. データ点のあいだには線を引かない.
linespoints	データ点に適切な図記号を配置し, データ点のあいだを線で結ぶ.
impulses	各データ点について, x 軸からデータ点まで垂直に伸びる直線を引く.
dots	データ点に小さい点を打つ.
steps	グラフを階段関数で表示する.
errorbars	y 軸にエラーバーを表示する.
xerrorbar	x 軸にエラーバーを表示する.
xyerrorbars	x 軸と y 軸にエラーバーを表示する.
boxes	各データ点について, x 軸からデータ点まで垂直に伸びる長方形を描く.
boxerrorbars	各データ点について, x 軸からデータ点まで垂直に伸びる長方形を描き, y 軸にエラーバーを付ける.
boxxyerrorbars	各測定点について, データ点の値とその誤差 $(x, y, x_\delta, y_\delta)$ が与えられているときには, 4 個の頂点 $(x - x_\delta, y - y_\delta)$, $(x - x_\delta, y + y_\delta)$, $(x + x_\delta, y - y_\delta)$, $(x + x_\delta, y + y_\delta)$ を持つ長方形を描画する. また, データ点の値とその最小値, 最大値 $(x, y, x_{min}, x_{max}, y_{min}, y_{max})$ が与えられているときには, 4 個の頂点 (x_{min}, y_{min}) , (x_{min}, y_{max}) , (x_{max}, y_{min}) , (x_{max}, y_{max}) を持つ長方形を描画する.

のように, コマンド `set` に続いていろいろな文字列を入力します. 以下に, コマンド `set` のあとに指定する文字列を作業内容に応じて分類したものを表にして示します. 表 6.5 には作図全般に関連したコマンドが, 表 6.6 には 3 次元グラフに関連したコマンドが, 表 6.7 には座標系 (曲線座標など) や目盛りの表示に関連したコマンドが, 表 6.8 には座標軸の表示に関連したコマンドがまとめられています.

ここで, 表の見方について説明しておきましょう.

まず, これらのコマンドはすべてコマンド `set` に続いて指定されます.

次に, 表中で

`[no]border`

などのように各括弧付きで表示されている部分は, 各括弧で囲われた部分を書くときと書かないときに機能が切り換わることを意味します. たとえば, 上の例は `set border` というコマンドと `set noborder` というコマンドを同時にあらわしていて, 前者が枠を表示するコマンド, 後者が座標軸を表示しないコマンドになります.

さいごに, 表中で `<>` で囲まれた部分は, そのパラメータが省略可能であることを意味しています. また,

A OR B

という記法を「キーワード A か B かのいずれか」という意味で使っている部分があります. たとえば, 表 6.5 の `mxgrid OR mygrid` という記述は, 「キーワード `mxgrid` と `xygrid` のいずれか一方を指定する」という意味になります.

これらのコマンドの具体的な使い方については次節以降を参照して下さい.

表 6.5: 作図全般にかかわるコマンド一覧

機能	コマンド
関数のプロットのスタイルを設定	function style <style-choice>
図を囲む枠の表示切り換え	[no]border
図の端の点や直線などを切り抜き表示の制御	[no]clip <clip-type>
データファイルのプロットのスタイルを設定	data style <style-choice>
媒介変数を指定する	dummy <dummy1>,<dummy2>...
角度の単位を設定する	angles [degrees radians]
始点から終点まで矢印を引く (すでにグラフが描画されているときに有効)	arrow [<tag>] [from <sx>,<sy>,<sz>] [to <ex>,<ey>,<ez>] [nohead]
グラフ中央からのオフセットの設定	offsets<left>,<right>,<top>,<bottom>
関数の標本点 (プロットに使う点) の数を指定	samples <expression>
図の縮尺を指定	size <xsize>,<ysize>
角度の範囲を設定	rrange [<rmin>:<rmax>]
端末切り換え	terminal <device>
図に作成日時を入れるかどうかを指定	[no]time
図の標題を指定	title "title-text" <xoff>,<yoff>

表 6.6: 3 次元グラフィックスに関連したコマンド一覧

機能	コマンド
等高線の制御	cntrparam [spline] [points] [order] [levels]
等高線の描画	[no]contour [base surface both]
隠れ面の表示切り換え	[no]hidden3d
グラフにかかった網目のきめを設定する	isosamples <expression>
水平面の位置を調整する	ticslevel <level>
視点を指定	view <rot_x>,<rot_z>,<scale>,<scale_z>
グラフの網かけの表示の切り換え	[no]surface
曲線座標との切り換え	mapping[cartesian spherical cylindrical]
パラメータ付き曲面の範囲指定	urange OR vrangle

表 6.7: 座標系や目盛りの制御などに関連したコマンド一覧

機能	コマンド
パラメータ付き曲線モードのオン, オフ	[no]parametric
グラフ中に方眼目盛りを描く	[no]grid [mxgrid OR mygrid]
目盛りの数値のフォーマットを指定	format [<axes>] ["format-string"]
パラメータ付き曲線のパラメータ範囲指定	trange [<tmin>:<tmax>]
座標軸の範囲を自動設定する	autoscale [<axes>]
キーの表示切り換え	key <x>,<y>,<z>
座標軸を対数目盛りにする	logscale <axes> [<base>]
2 次元極座標との切り換え	[no]polar
目盛りの切り方の指定	tics <direction>
目盛り記号の大きさを調整する	ticscale [<size>]

表 6.8: 座標軸の調整に関連したコマンド一覧

機能	コマンド
x 軸のラベルを設定	xlabel "<label>" <xoff>,<yoff>
x 軸の描画範囲を指定	xrange [<xmin>:<xmax>]
x 軸の目盛りを変更	xtics <start>,<incr>,<end>,"<label>" <pos>
x 軸の目盛りの鏡像 (通常上) の表示切り換え	xtics [no]mirror
x 軸の副目盛りの調整	[no]mxtics OR [no]mytics [<freq>]
x 軸の表示切り換え	[no]xzeroaxis
y 軸のラベルを設定	ylabel "<label>" <xoff>,<yoff>
y 軸の描画範囲を指定	yrange [<ymin>:<ymax>]
y 軸の目盛りを変更	yticks <start>,<incr>,<end>,"<label>" <pos>
y 軸の目盛りの鏡像 (通常右) の表示切り換え	yticks [no]mirror
y 軸の表示切り換え	[no]yzeroaxis
零付近のしきい値の設定	zero <expression>
座標軸の描画切り換え	[no]zeroaxis
z 軸のラベルを設定	zlabel "<label>" <xoff>,<yoff>
z 軸の描画範囲を指定	zrange [<zmin>:<zmax>]
z 軸の目盛りを変更	zticks <start>,<incr>,<end>,"<label>" <pos>
z 軸の表示切り換え	[no]zzeroaxis

第7章 2次元グラフを描画する

7.1 関数のグラフのプロット

7.1.1 簡単な2次元グラフを描画する

簡単な2次元グラフを描画するには、

```
plot 関数名 [Enter]
```

と入力します。関数のデフォルトの変数(引数)は x です。

たとえば、正弦関数 $\sin(x)$ を描画するときには、

```
plot sin(x) [Enter]
```

と入力します。すると、図の x 軸と y 軸が自動的に調整され、正弦関数のグラフが描画されます。

プロットが終わったグラフを `eps` ファイルの形式で保存するには、

```
set terminal postscript eps [Enter]
```

```
set output 'ファイル' [Enter]
```

```
replot [Enter]
```

とします。ここに、「ファイル」と書かれた部分には適当なファイル名が入ります。

7.1.2 y 軸や x 軸の範囲を変える

gnuplot ではグラフの x 軸や y 軸の範囲の初期値は自動的に設定されます。しかし、標準的な設定で適切なグラフが得られないときには、これらを自分で設定し直す必要があります。

このような場合には、コマンド `plot` に続けて x 軸と y 軸の範囲を指定します。 y 軸、 x 軸のそれぞれについてそれぞれ

- 指定なし
- 下限のみ指定
- 上限のみ指定
- 上限と下限を指定

の4種類の指定ができます。指定されていない部分は自動的に調整されます。以下にいくつか例を示します。

これとは別に、グラフを描く前に描画範囲を指定したり描画範囲の自動調整の有無を切り換えたりするコマンドもあります。これらについて表 7.2 にまとめておきます。

表 7.1: 描画範囲の設定例

機能	コマンド
x 軸, y 軸とも指定しない	<code>plot exp(x) [Enter]</code>
x 軸の上限のみ指定, y 軸は指定なし	<code>plot [:9] exp(x) [Enter]</code>
x 軸の下限のみ指定, y 軸は指定なし	<code>plot [1:] exp(x) [Enter]</code>
x 軸の上限と下限を指定, y 軸は指定なし	<code>plot [1:9] exp(x) [Enter]</code>
x 軸は指定なし, y 軸の上限と下限を指定	<code>plot [] [1:4] exp(x) [Enter]</code>
x 軸と y 軸の上限と下限を指定	<code>plot [1:9] [1:4] exp(x) [Enter]</code>

表 7.2: 描画範囲の調整コマンド

機能	コマンド
x 軸の範囲を指定	<code>set xrange [x0:x1] [Enter]</code> x0 と x1 の部分には初期値と終了値 (数値) を指定する; 片方を省略することもできる
y 軸の範囲を指定	<code>set yrange [y0:y1] [Enter]</code> y0 と y1 の部分には初期値と終了値 (数値) を指定する; 片方を省略することもできる
全座標軸を自動調整する	<code>set autoscale [Enter]</code>
全座標軸を自動調整しない	<code>set noautoscale [Enter]</code>
x 軸を自動調整する	<code>set autoscale x [Enter]</code>
x 軸を自動調整しない	<code>set noautoscale x [Enter]</code>
y 軸を自動調整する	<code>set autoscale y [Enter]</code>
y 軸を自動調整しない	<code>set nonoautoscale x [Enter]</code>
x 軸と y 軸を自動調整する	<code>set autoscale xy [Enter]</code>
x 軸と y 軸を自動調整しない	<code>set noautoscale xy [Enter]</code>

7.1.3 y 軸と x 軸にラベルをつける

x 軸や y 軸にラベルをつけるときには, `set xlabel` および `set ylabel` というコマンドを使います. x 軸のラベルを設定するには,

```
set xlabel 'ラベルの名前' [Enter]
```

とします. 同様に, y 軸のラベルを設定するには,

```
set ylabel 'ラベルの名前' [Enter]
```

のようにします.

以下に, x 軸に「時間」, y 軸に「人口」というラベルを付けて指数関数を描画する例を示します.

```
set xlabel '時間' [Enter]
set ylabel '人口' [Enter]
plot exp(x) [Enter]
```

結果は図 7.1 のようになります.

ラベルを付けたくないときには, `set nolabel`, `set noxlabel`, `set noylabel` というコマンドが使えます. これらについて表 7.3 にまとめておきます.

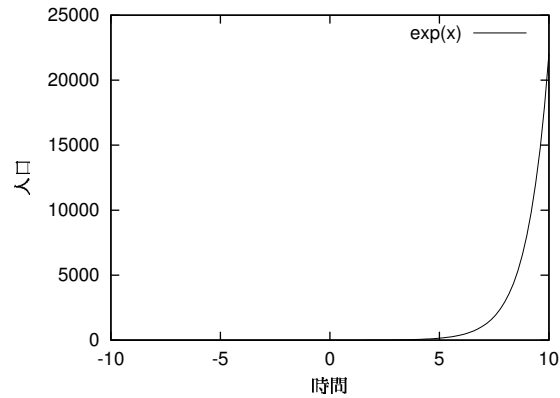


図 7.1: y 軸と x 軸のラベルを指定したグラフ

表 7.3: ラベル関連のコマンド

機能	コマンド
x 軸のラベルを設定	set xlabel 'ラベル' [Enter]
y 軸のラベルを設定	set ylabel 'ラベル' [Enter]
ラベルをすべて表示する	set label [Enter]
ラベルをすべて表示しない	set nolabel [Enter]
x 軸のラベルを表示する	set xlabel [Enter]
x 軸のラベルを表示しない	set noxlabel [Enter]
y 軸のラベルを表示する	set ylabel [Enter]
y 軸のラベルを表示しない	set noylabel [Enter]

7.1.4 関数のグラフの線のスタイルを変える

関数のグラフを表示するときには、線以外にもいろいろなものが使えます。ここでは、これらについて順に説明してゆきます。

7.1.4.1 グラフを点で表示

グラフを線でむすぶのをやめて、かわりに点で表示するには、

```
plot 関数 with points [Enter]
```

のようにします。

以下に三角関数を点を使って描画している例を示します。

```
plot sin(x) with points [Enter]
```

結果は図 7.2 のようになります。

7.1.4.2 線と点を重ねる

線と点を重ねることもできます。このような場合には、

```
plot 関数 with linespoints [Enter]
```

のようにします。

以下に例を示します。

```
plot sin(x) with linespoints [Enter]
```

結果は図 7.3 のようになります。

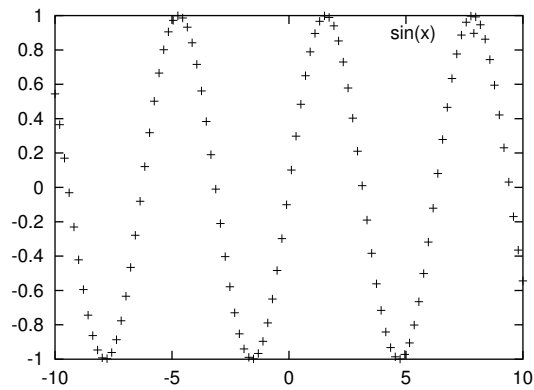


図 7.2: 点を使ってプロット

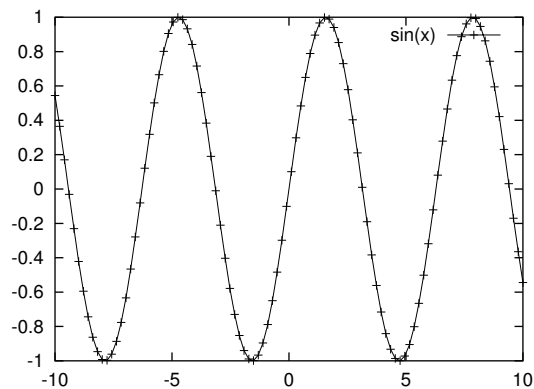


図 7.3: 点と線でプロット

7.1.4.3 インパルスで表示

グラフを線や点で表示するかわりに、x 軸から伸びるインパルスで関数を表示することもできます。このような場合には、

```
plot 関数 with impulses [Enter]
```

のようにします。

以下に例を示します。

```
plot sin(x) with impulses [Enter]
```

結果は図 7.4 のようになります。

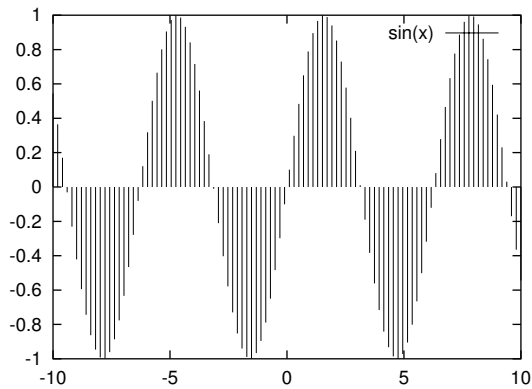


図 7.4: インパルスでプロット

7.1.4.4 階段グラフ

階段グラフを作ることができます。このような場合には、

`plot 関数 with steps [Enter]`

のようにします。

以下に例を示します。

`plot sin(x) with steps [Enter]`

結果は図 7.5 のようになります。

x 軸から伸びる長方形のボックスで関数を表示するには、

`plot 関数 with boxes [Enter]`

のようにします。

以下に例を示します。

`plot sin(x) with boxes [Enter]`

結果は図 7.6 のようになります。

階段グラフには、`steps`、`fsteps`、`histeps` の 3 種類があり、それぞれ挙動が違います。2 個の点 (x_1, y_1) と (x_2, y_2) が与えられているとき、`steps`、`fsteps`、`histeps` はそれぞれ

steps (x_1, y_1) から (x_2, y_1) への水平な線と (x_2, y_1) から (x_2, y_2) への垂直な線を引く

fsteps (x_1, y_1) から (x_1, y_2) への垂直な線と (x_1, y_2) から (x_2, y_2) への水平な線を引く

histeps $((x_0 + x_1)/2, y_1)$ から点 $((x_1 + x_2)/2, y_1)$ への水平な線と $((x_1 + x_2)/2, y_1)$ から点 $((x_1 + x_2)/2, y_2)$ への垂直な線を引く

という挙動を示します。図 7.7 のようなデータファイル（ファイル名を `steps.dat` とします）が与えられたときの `steps`、`fsteps`、`histeps` の挙動をそれぞれ図 7.8、図 7.9 および図 7.1.4.4 に示します。挙動の違いを見やすくするために、データ点を大きく表示してあります。

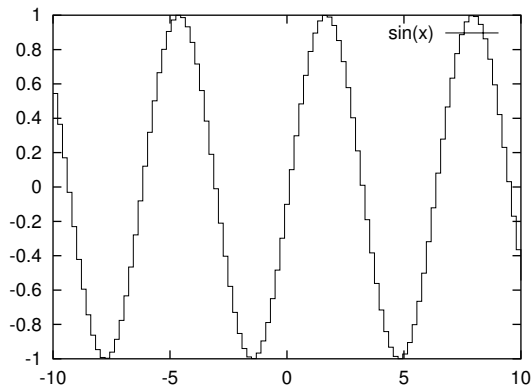


図 7.5: 階段グラフでプロット

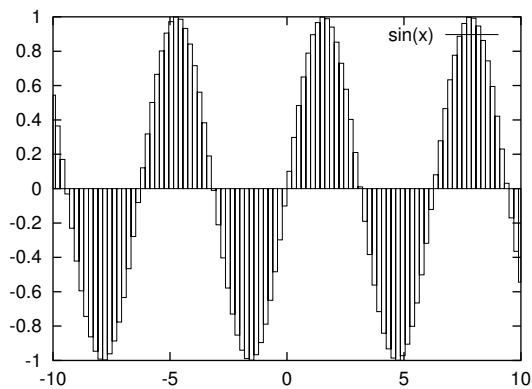


図 7.6: x 軸から伸びる長方形の箱でプロット

7.1.4.5 線の太さを変える

グラフの線の太さを変えるには

`plot 関数 with lines linewidth 数値`

のように、キーワード `linewidth` のあとに適当な値（整数）に指定します。

以下に線の太さを通常の 10 倍にしている例を示します。

`plot sin(x) with lines linewidth 10 [Enter]`

結果は図 7.11 のようになります。

なお、`linewidth` の省略形である `lw` というキーワードも使えます。ですから、

`plot sin(x) with lines lw 10 [Enter]`

0	10
10	30
20	100
30	40

図 7.7: データファイル `steps.dat` の内容

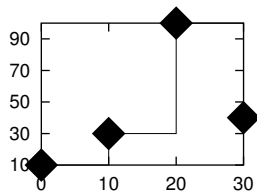


図 7.8: steps の挙動

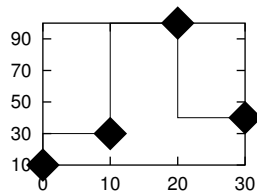


図 7.9: fsteps の挙動

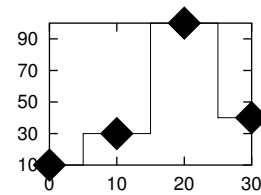


図 7.10: histeps の挙動

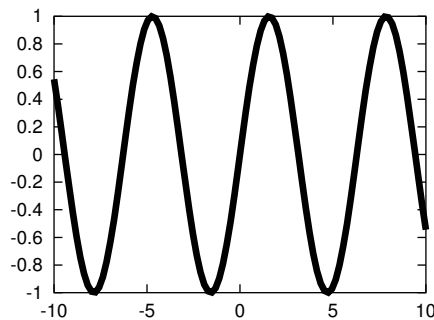


図 7.11: 線の太さを 10 倍にしたもの

としても同じ結果が得られます。

7.1.4.6 線の種類を変える

線の種類を変えるには、

```
plot 関数 with lines linestyle 数値 [Enter]
```

linestyle のあとに適当な数値を指定します。

以下に例を示します。

```
plot sin(x) with lines linestyle 4 [Enter]
```

結果は図 7.12 のようになります。なお、linestyle の省略形である lt というキーワードも使えます。ですから、

```
plot sin(x) with lines lt 4 [Enter]
```

としても同じ結果が得られます。

上記の操作をおこなうと、画面にはマゼンタ色の線が表示されるのですが、白黒の PostScript ファイルに保存するとこの線は点線で置き換えられます。

一般に、線の種類を変更した場合には、白黒のデータを作成すると、線を色で区別することができなくなるので、かわりにいろいろな点線が使われます。結果として、画面に見えている線の見掛けとファイルに保存した線の見掛けは変わります。

図 7.13 に利用可能な線の種類の一覧を示します。線の種類は 1 から 8 までの 8 種類が指定でき、それぞれ X Window System の画面では赤、緑、青、マゼンタ (赤紫)、シアン、シエナ (赤茶色)、橙、コラル (珊瑚色) で表示されますが、白黒の PostScript で保存すると図 7.13 に示したような実線や点線に変わります。なお、他の色が使える形式 (たとえば tgif) で保存した場合、表示される色が X Window System のそれと一致しないことがあります。

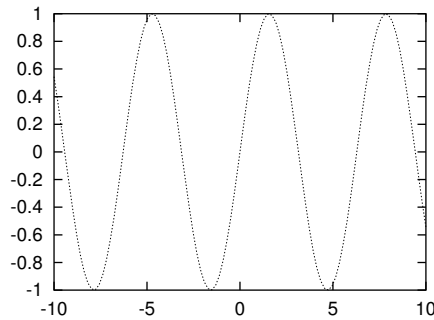


図 7.12: 線の種類を変更したもの

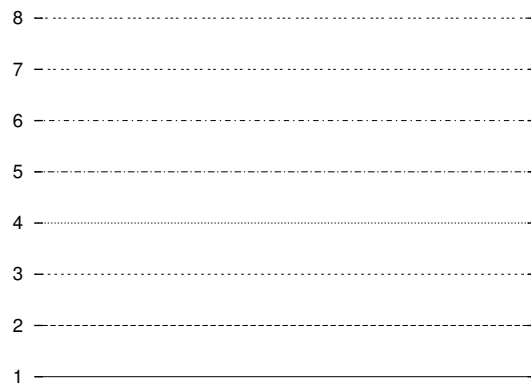


図 7.13: いろいろな線の種類

7.1.4.7 点の種類を変える

グラフの点の種類を変えるには

`plot 関数 with points pointtype 数値`

のように、キーワード `pointtype` のあとに適当な値（整数）に指定します。

以下に例を示します。

```
plot sin(x) with points pointtype 2 [Enter]
```

結果は図 7.14 のようになります。

なお、`pointtype` の省略形である `pt` というキーワードも使えます。ですから、

```
plot sin(x) with points pt 2 [Enter]
```

としても同じ結果が得られます。

点の種類を変えた場合も、画面に表示される点の種類とグラフを保存したときにそのファイルで使われる点の種類は一致しないことがあるので、注意が必要です。指定可能な 75 種類の点の X Window System 上での見え方図 7.15 に、対応する PostScript ファイルにおける点の見え方を図 7.16 に示します。いずれも、点の番号は、図の右端で 0 から始まり、反時計回りに 75 まで増加してゆきます。

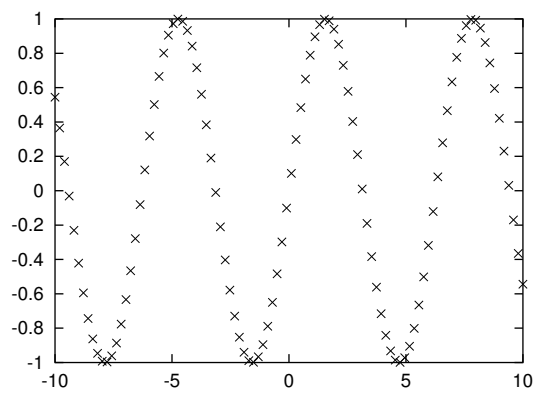


図 7.14: 点の種類を変更したもの

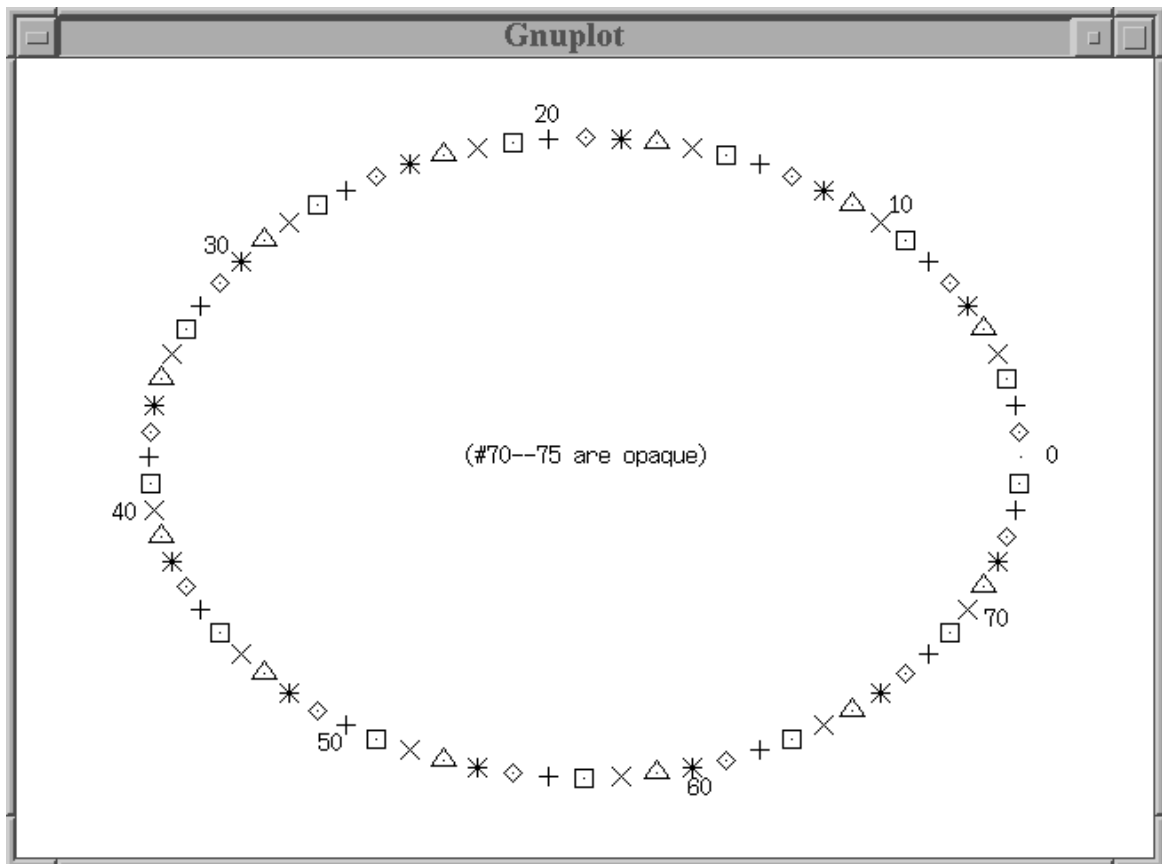


図 7.15: いろいろな点の見え方 (X Window System 上の表示)

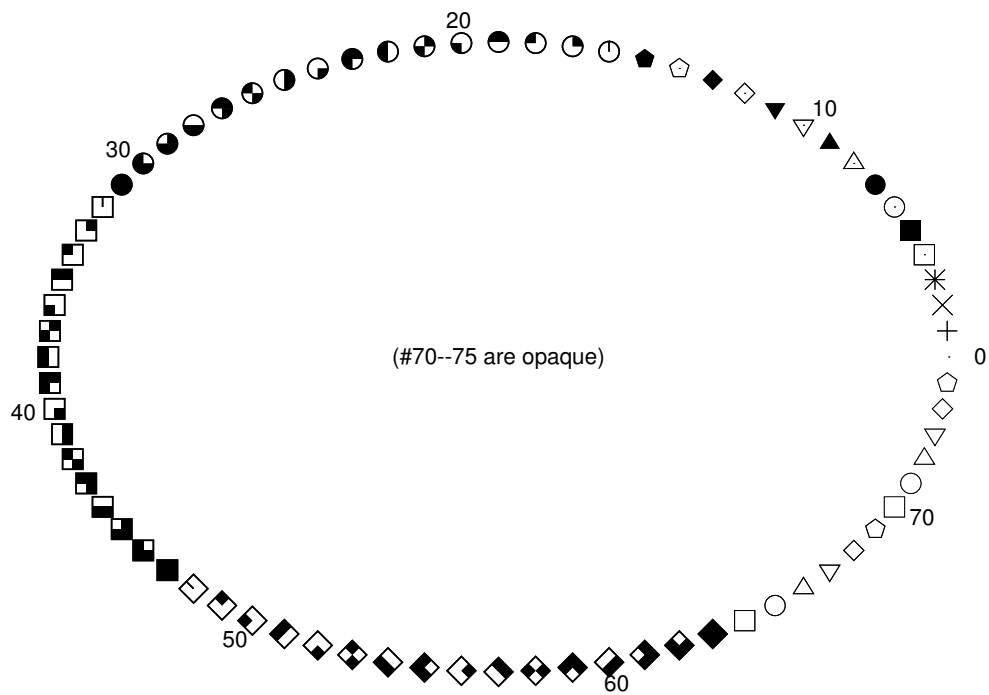


図 7.16: いろいろな点の見え方 (PostScript ファイルの表示)

7.1.4.8 点の大きさを変える

グラフの点の大きさを変えるには

`plot 関数 with points pointsize 数値`

のように、キーワード `pointsize` のあとに適当な値（整数）に指定します。以下に例を示します。

```
plot sin(x) with points pointsize 10 [Enter]
```

なお、`pointtype` の省略形である `ps` というキーワードも使えます。ですから、

```
plot sin(x) with points ps 2 [Enter]
```

としても同じ結果が得られます。出力例は省略します。

点を使ってプロットしている場合は、点の種類と点の大きさの指定を併用できます。このような場合には、

```
plot 関数 with points pt 数値 ps 数値 [Enter]
```

のようにします。

7.1.4.9 点と線の種類を同時に変える

線と点を両方使ってプロットしている場合には、

```
plot 関数 with linespoints lw 数値 pt 数値 [Enter]
```

とすれば、線の太さと点の種類を両方変更できます。

7.1.5 描画するときのサンプル点の間隔を調整する

さて、ここで

```
plot [0:5] sin(exp(x)) [Enter]
```

と入力してみます。表示されたグラフは図 7.17 のようになります。本来はこの関数の値は 0 と 1 のあいだで振動するはずなのですが、変なふうに値が欠落してしまっています。何が悪いのでしょうか？

`gnuplot` は、 x 軸のいくつかの点について対応する y 軸の値を計算して、そのあいだをなめらかな線で結ぶ、ということを実行しています。

```
plot 関数 with points [Enter]
```

としたときに表示される点の数が、標準で計算される点の数です。この点の数が少な過ぎると、グラフが欠けてしまいます。

x 軸に取る点の数を指定するには、

```
set samples 数値 [Enter]
```

と入力します。

x 軸に取る点の数を 10000 個に変更してからグラフを書き直している例を以下に示します。

```
set samples 10000 [Enter]
plot [0:7] sin(exp(x)) [Enter]
```

こんどは、表示されたグラフは図 7.18 のようになり、データの欠落はなくなります。

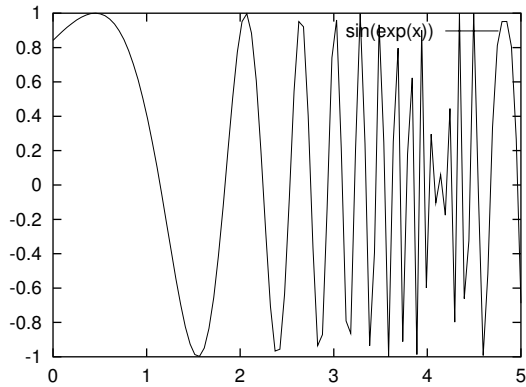


図 7.17: サンプルが少なすぎるグラフ

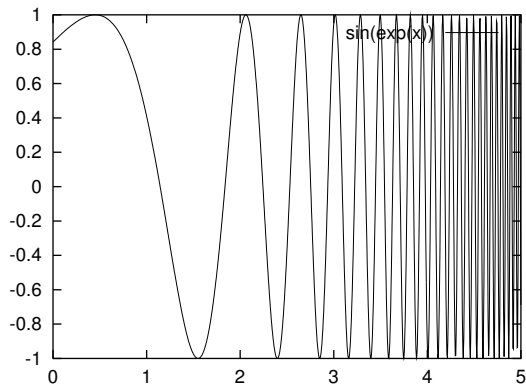


図 7.18: サンプルが十分にあるグラフ

7.1.6 キー (関数など説明) の操作

gnuplot で図を作成すると、標準では図の右上にグラフと同じスタイルの水平な線が表示され、その左側にグラフの説明として関数の名前が表示されます。この部分をキー (key) といいます。本項ではこの部分を変更する方法を説明します。

7.1.6.1 グラフを説明するテキストを変更する

線の横の関数名が表示されている部分を変更するには、`title` というキーワードを使います。

```
plot 関数 title 'グラフの説明' [Enter]
```

とすると、関数名が表示されていた部分が「」で囲まれた文字列で置き換えられます。

たとえば、

```
plot sin(x) title '正弦関数' [Enter]
```

のようにすると、図 7.19 のように図の右上に「正弦関数」という文字が入ります。

この部分に数式などを入れることもできます。数式などを入れる方法については、10 章を参照して下さい。

7.1.6.2 キーのスタイルと位置の制御

キーを制御するには、`set key` というコマンドに続いていろいろなオプションを指定してゆきます。

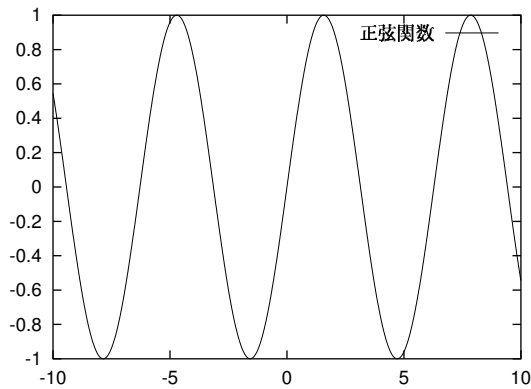


図 7.19: 線の説明付きのグラフ

まず、基本的なものとして、表 7.4 に 3 個のコマンドを挙げておきます。

表 7.4: グラフの説明の制御 (基本)

機能	コマンド
キーを表示する	set key
キーを表示しない	set nokey
キーの状態を見る	show key

次に、キーの表示場所や表示のされかたを変更するためのコマンド `set key` のさまざまなオプションを表 7.5 にまとめておきます。表 7.5 のオプションはすべてコマンド `set key` のあとに続けて指定されます。たとえば、表 7.5 の最初の「キーの場所を左上にする」という機能を使うためには、

```
set key left top [Enter]
```

と入力します。

表 7.5 において、文字が斜体になっている部分には、適当な数値が入ります。特に断ってある場合を除き、正の数と負の数のいずれも指定可能です。

また、表 7.5 のオプションの中で種類が違うものについてはいくつでも組み合わせて指定することができます。表 7.5 では、オプションの種類が変わるごとに横線を引いて区別してあります。

線の説明を箱で囲むときには、さらに囲む線のスタイルなどを変更することもできます。指定の仕方は 7.1.4 と同じです。

7.1.7 図の標題の制御

図に標題をつけるには、

```
set title '図の標題'
```

とします。上で「図の標題」と書いてある部分に適当なグラフの題目を指定すると、その題目が図の上側に表示されます。

ただし、図の標題を図の上に入れることは、技術文書の書き方としては、必ずしも標準的ではありません。たとえば、レポートを作成するときに、図の標題を必ず図の下に入れるように指示されることもあります。

ですから、`gnuplot` を使って図に標題をつけるよりは、`gnuplot` で作図するときには標題をつけずにおいて、他の文書作成ソフトウェアを使って図入りの文書を作成するときに改めて標題をつけ直す方がよいでしょう。

表 7.5: コマンド `set key` のさまざまなオプション

機能	オプション
キーの場所を左上にする	left top
キーの場所を左下にする	left bottom
キーの場所を右上にする	right top
キーの場所を右下にする	right bottom
キーの場所を画面外上段にする	top outside
キーの場所を画面外下段にする	bottom outside
キーの場所をグラフの下側にする	below
キーが表示される場所の座標 x と y を指定する	x, y
線の見本の長さに m 文字分の長さを追加する	samplen m
説明を線の右側に付ける	noreverse
説明を線の左側に付ける	reverse
線の説明を箱で囲む	box
線の説明を箱で囲まない	nobox
文字が表示される部分の前側に m 文字分の空白を追加する	width w
グラフが複数あるとき (7.2 節参照), 説明の文字を左にそろえる	Left
グラフが複数あるとき説明の文字を右にそろえる	Right
グラフが複数あるとき説明のあいだの空白の幅を標準の m 倍にする ($m > 0$)	spacing m
キー全体に標題をつける	title 'キーの標題'

7.1.8 詳細な目盛りの制御

7.1.8.1 目盛りの表示

x 軸の目盛りを切り直すには,

```
set xtics 初期値, 増分, 終了値 [Enter]
```

とします. 同様に, y 軸の目盛りを等間隔で切り直すには,

```
set ytics 初期値, 増分, 終了値 [Enter]
```

とします.

gnuplot は, 標準では, x 軸についてはグラフの上下に, y 軸についてはグラフの左右に目盛りを打ちます. この表示切り換え (両方に目盛りを打つか片方だけにするか) には, `[no]mirror` というキーワードを使います.

たとえば, x 軸に関し, 上下双方に目盛りを打つのをやめたいときには,

```
set xtics nomirror [Enter]
```

と入力し, 逆に上下双方に目盛りを打つようにしたいときには

```
set xtics mirror [Enter]
```

とします. y 軸に関しても同様で, 左右双方に目盛りを打つのをやめたいときには,

```
set ytics nomirror [Enter]
```

と入力し, 逆に左右双方に目盛りを打つようにしたいときには

```
set ytics mirror [Enter]
```

とします。

目盛りを不等間隔で切りたいときには、() 内に目盛りを打つ場所の座標を指定してゆきます。たとえば、グラフ下側の x 軸の座標 1, 4, 9, 16 という点に目盛りを打ちたいときには、

```
set xtics (1,4,9,16) [Enter]
```

とします。同様に、グラフ左側の y 軸の座標 1, 4, 9, 16 という点に目盛りを打ちたいときには、

```
set ytics (1,4,9,16) [Enter]
```

とします。

目盛りを表示するかしないかを切り換えることもできます。たとえば、下の目盛りを表示するときには

```
set xticsk [Enter]
```

とし、たとえば、下の目盛りを表示しないときには、

```
set noxtics [Enter]
```

とします。

ひとつの目盛りの中により細かい副目盛りを刻むこともできます。このためには、

```
set mxtics 数 [Enter]
```

```
set mytics 数 [Enter]
```

のように指定します。「数」と書かれた部分には、ひとつの目盛りの中にいくつ小さい目盛りを打つかを整数で指定します。

目盛りの表示/非表示の切り換えに関連したコマンドを表 7.6 にまとめておきます。

表 7.6: 目盛りの表示, 非表示関連のコマンド

機能	コマンド
x 軸の目盛りを表示	set xtics [Enter]
x 軸の目盛りを非表示	set noxtics [Enter]
x 軸の目盛りを上下の片方だけに	set xtics nomirror [Enter]
x 軸の目盛りを上下の両方に	set xtics mirror [Enter]
y 軸の目盛りを表示	set ytics [Enter]
y 軸の目盛りを非表示	set noytics [Enter]
y 軸の目盛りを左右の片方だけに	set ytics nomirror [Enter]
y 軸の目盛りを左右の両方に	set ytics mirror [Enter]
目盛りの全状態を報告	show tics [Enter]

7.1.8.2 目盛りの数値の書式を変更する

座標軸の目盛りに表示される数値の書式を変更するには、

```
set format 座標軸 "書式" [Enter]
```

という構文を使います。「座標軸」と書かれた部分にはなにも書かれないか、 x, y, xy, x^2, y^2 のいずれかを入れます。座標軸を指定しないと、指定した書式が x 軸と y 軸に適用されます。 x^2, y^2 の意味についてはひとつ上の項を参照して下さい。

「書式」の部分には C 言語の浮動小数点数の書式指定と同じ様式で書式を指定します。具体的には、

```
%数. 数 e
%数. 数 f
%g
```

などのように、文字%のあとに e, f, g の 3 文字のいずれかを書き、そのあとに必要なに応じて 2 個の数を小数点で区切ってならべます。2 個の数を書いた場合、小数点の左側が全桁数、小数点の右側が小数点以下の桁数をあらわします。これらは片方あるいは両方とも省略することができます。文字 e, f, g の意味については表 7.7 にまとめておきます。gnuplot の標準の書式は g です。

表 7.7: 書式指定で使える文字

意味	文字
指数表示	e
ふつうの小数表示	f
自動	g

なお、現在の目盛りの数値の書式を確認したいときには、

```
show format
```

とします。

いくつか例を示しましょう。

表 7.8: 書式切り換えの例

機能	コマンド
全座標軸に書式 g を設定	<code>set format "%g" [Enter]</code>
x 軸に書式 f を設定, 全 2 桁, 小数点以下 1 桁	<code>set format x "%2.1f" [Enter]</code>
y 軸に書式 e を設定, 全 2 桁, 小数点以下 1 桁	<code>set format y "%2.1e" [Enter]</code>
x 軸と y 軸に書式 f を設定, 全 4 桁, 小数点以下 2 桁	<code>set format xy "%4.2f" [Enter]</code>
現在の目盛りの数値の書式を確認	<code>show format [Enter]</code>

7.1.8.3 副目盛りの表示と目盛りの調整

gnuplot は、グラフの x 軸や y 軸に、定規についているような主目盛りよりも細かい副目盛りを打つことができます。

たとえば、 x 軸に副目盛りを打ちたいときには、

```
set mxtics [Enter]
```

とします。また、 x 軸の副目盛りを消したいときには、

```
set nomxtics [Enter]
```

と入力します。

副目盛りはデフォルトでは 1 個の主目盛を 10 等分するように打たれます。 これを変更するには、

```
set mxtics 数 [Enter]
```

と入力します。 ただし、「数」と書かれた部分では、1 個の主目盛りを何等分するかを指定します。 x2tics, ytics などについても同様です。

gnuplot はデフォルトでは目盛りをグラフ全体を囲むボックスの内側に表示しますが、この目盛りの表示位置をボックスの外側に変更することもできます。 目盛りの表示位置をグラフの外側にするには

```
set tics out [Enter]
```

と入力し、目盛りの表示位置をグラフの内側にするには

```
set tics in [Enter]
```

と入力します。

目盛りの制御に関連したコマンドを表 7.9 にまとめておきます。 なお、表中で「[数]」と書かれている部分では、「数」という部分には適当な数値が入ります。 また、記号「[]」はその部分の記述が省略可能であることを意味しています。 たとえば、表 7.9 の 1 段目の記述は、set mxtics というコマンドが

```
set mxtics [Enter]
```

あるいは

```
set mxtics 10 [Enter]
```

などのようにして使われることを意味しています。

表 7.9: グラフの大きさの指定

機能	コマンド
x 軸の副目盛りを表示, 設定	set mxtics [数] [Enter]
x 軸の副目盛りを非表示	set nomxtics [Enter]
x 軸の第 2 副目盛りを表示, 設定	set mx2tics [数] [Enter]
x 軸の第 2 副目盛りを非表示	set nomx2tics [Enter]
y 軸の目盛りを表示, 設定	set mytics [数] [Enter]
y 軸の目盛りを非表示	set nomytics [Enter]
y 軸の第 2 副目盛りを表示, 設定	set my2tics [数] [Enter]
y 軸の第 2 副目盛りを非表示	set nomy2tics [Enter]
z 軸の目盛りを表示, 設定	set mztics [数] [Enter]
z 軸の目盛りを非表示	set nomztics [数] [Enter]
目盛りをボックスの内側にする	set tics in [Enter]
目盛りをボックスの外側にする	set tics out [Enter]
目盛りの状態を表示	show tics [Enter]

7.1.8.4 x 軸や y 軸の目盛りを曜日で表示する

gnuplot には、x 軸や y 軸の目盛りの表示を曜日に切り換える機能があります。 x 軸の目盛りを曜日に切り換えるには


```
set xdtics [Enter]
```

とします。

目盛り表示が曜日になっているときには、 x 軸や y 軸の読みが 0 のときは日曜日 (Sun), 1 のときは月曜日 (Mon), ..., 6 のときは土曜日 (Sat) と解釈されます。7 になると日曜日に戻ります。

例として、

```
set xdtics [Enter]
set ydtics [Enter]
plot [0:13] sin(x) [Enter]
```

とした場合の出力を図 7.20 に示します。

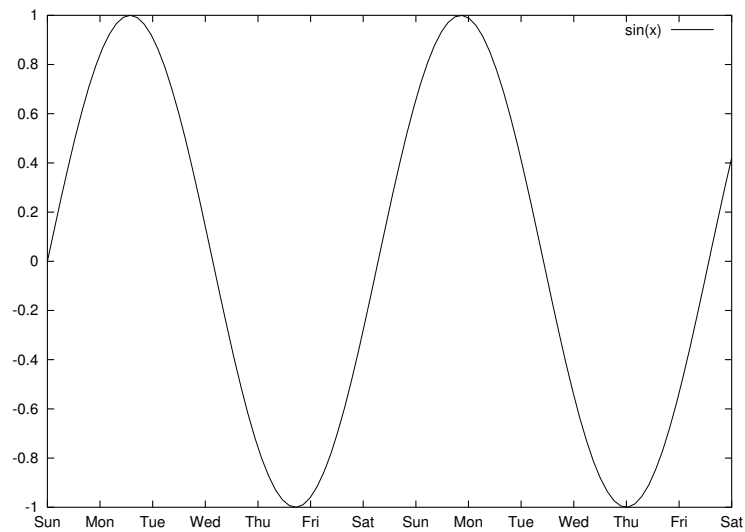


図 7.20: 横軸を曜日表示にしたグラフ

表 7.10: 座標軸の読みを曜日に切り換えるコマンド一覧

機能	コマンド
x 軸の読みを曜日表示にする	set xdtics
x 軸の読みの曜日表示をやめる	set notdtics
x 軸の読みの状態を表示する	show xdtics
y 軸の読みを曜日表示にする	set ydtics
y 軸の読みの曜日表示をやめる	set notdtics
y 軸の読みの状態を表示する	show ydtics

7.1.9 座標軸の対数目盛りと通常の日盛りの切り換え

x 軸と y 軸のそれぞれについて、座標軸を対数目盛りに変えたり、ふつうの日盛りに戻したりできます。表 7.11 にこれらのコマンドをまとめておきます。

表 7.11: 対数目盛りの切り換えコマンド

機能	コマンド
座標軸を対数目盛りにする	set logscale [Enter]
x 軸と y 軸の両方の対数目盛りを解除する	set nologscale [Enter]
x 軸のみ対数目盛りに変える	set logscale x [Enter]
x 軸のみ対数目盛りをやめる	set nologscale x [Enter]
y 軸のみ対数目盛りに変える	set logscale y [Enter]
y 軸のみ対数目盛りをやめる	set nologscale y [Enter]

7.1.10 枠や座標軸、グリッドの表示と非表示

枠（グラフを囲う長方形）や座標軸、グリッド（格子）の表示と非表示を切り換えるコマンドを以下にまとめます。座標軸を表示する設定に変更すると、点線で座標軸が表示されるようになります。x 軸と y 軸の一方だけを表示することもできます。

表 7.12: 枠や座標軸の表示切り換えコマンド

機能	コマンド
枠の表示	set border [Enter]
枠の非表示	set noborder [Enter]
座標軸の表示	set zeroaxis [Enter]
座標軸の非表示	set nozeroaxis [Enter]
グリッドの表示	set grid [Enter]
グリッドの非表示	set nogrid [Enter]
x 軸の表示	set xzeroaxis [Enter]
x 軸の非表示	set noxzeroaxis [Enter]
y 軸の表示	set yzeroaxis [Enter]
y 軸の非表示	set noyzeroaxis [Enter]

7.1.11 グラフの大きさの指定

グラフの大きさを指定するときには、

```
set size
```

というコマンドを使います。指定する数値は、グラフ表示用のウィンドウから見た相対的な数値になります。指定できる内容は、

- グラフを正方形にする
- グラフの縦横の比率を指定する
- グラフの横の単位長さと縦の単位長さの比率を指定する
- グラフの表示比率を標準に戻す

表 7.13: グラフの大きさの指定

機能	コマンド
描画範囲を指定する	set size r_x, r_y [Enter] r_x, r_y は 0 以上 1 未満の数で, それぞれウィンドウの横幅とグラフの横幅, ウィンドウの高さとグラフの高さの比率をあらわす
グラフを正方形にする	set size square [Enter]
グラフの正方形表示をやめる	set size nosquare [Enter]
グラフの横縦の比率を r にする	set size ratio r [Enter]
y 軸の単位長さを x 軸の r 倍にする	set size ratio $-r$ [Enter]
比率の指定をやめる	set size noratio [Enter]
現在の大きさ指定を表示する	show size [Enter]

の 4 種類です. これらの使い方を表 7.13 に示します. なお, r_x, r_y と r の部分には適当な数値が入ります.

「縦横の比率を指定する」ことと「横の単位長さと縦の単位長さの比率を指定する」の違いに注意して下さい. 前者ではグラフの見た目が指定されているのに対し, 後者では y 軸の単位長さを x 軸の単位長さの何倍に取るかが指定されているだけで, グラフの見た目は指定されていません.

比較のために,

```
set size ratio 1 [Enter]
plot [0:2] sin(x) [Enter]
```

というコマンドを実行した結果を図 7.21 に,

```
set size ratio -1 [Enter]
plot [0:2] sin(x) [Enter]
```

というコマンドを実行した結果を図 7.22 に示します.

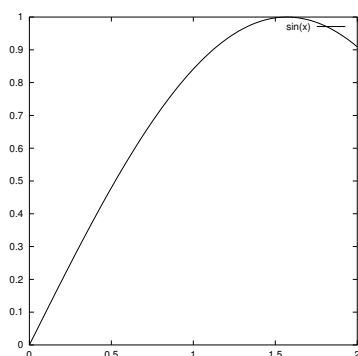


図 7.21: 縦横の比率を指定

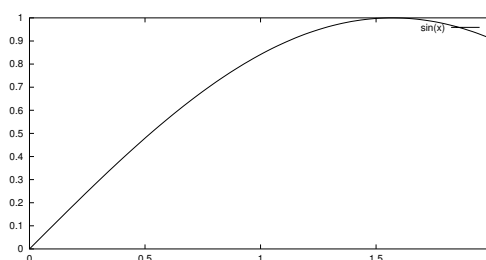


図 7.22: 横の単位長さと縦の単位長さの比率を指定

7.1.12 グラフの起点の指定

画面に描画されるグラフの起点 (左端の点) を指定するには, set origin というコマンドを使います.

具体的な指定の仕方は、

```
set origin x,y [Enter]
```

です。ここに、 x,y と書かれた部分には適当な数値が入ります。コマンド `set origin` はグラフの起点（左端の点）をグラフが表示されるウィンドウから見た相対的な位置で指定するので、 x と y は 0 以上 1 以下の値でなければなりません。

たとえば、

```
set origin 0.5,0.5 [Enter]
```

とすると、グラフを描き始める点がグラフが表示されるウィンドウの中央になります。

```
set origin 0.5,0.5 [Enter]
```

```
plot sin(x) [Enter]
```

とした場合に X Window System の画面に表示されるグラフの例を図 7.23 に示します。

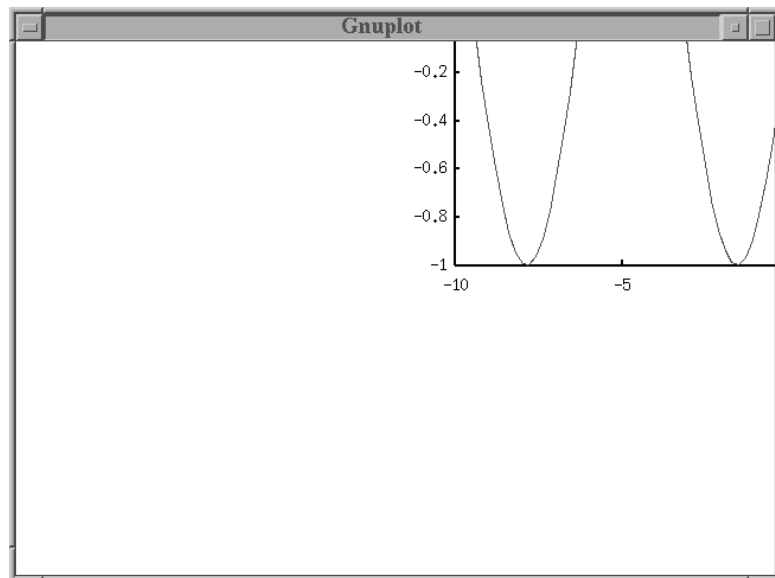


図 7.23: 原点を指定したグラフ

なお、たとえば図 7.23 のようなグラフを描いてからそのグラフを PostScript 形式で保存した場合には、原点の情報は失われます。図 7.23 のような不完全なグラフがそのまま保存されることはありません。

7.1.13 グラフ上下左右の余白の調整

グラフの上下左右の余白を調整するためには、たとえば

```
set lmargin
```

というコマンドを使います。また、現在の空白の設定を見るためには、

```
show margin
```

というコマンドを使います。これらを表 7.14 にまとめておきます。なお、表 7.14 で「数」と書かれた部分には適当な数値が入ります。値が大きいほど余白が大きくなります。グラフを PostScript 形式で保存した場合にも、余白の情報はそのまま保存されます。

表 7.14: グラフの大きさの指定に関連したコマンド

機能	コマンド
左の欄外の幅を設定	set lmargin 数
右の欄外の幅を設定	set rmargin 数
上の欄外の幅を設定	set tmargin 数
下の欄外の幅を設定	set bmargin 数
欄外の設定を表示	show margin

7.1.14 グラフとボックスの間隔の調整

グラフをボックスで囲むときには、グラフとボックスのあいだの間隔を調整することができます。この間隔はデフォルトでは0になっていて、グラフの描画範囲に一致するボックスが生成されるようになっています。

グラフとボックスのあいだの間隔を調整するには、set offsets というコマンドに続いてグラフの左、右、上、下の空白の大きさをコンマで区切って指定します。

```
set offsets 0,1,2,4 [Enter]
plot sin(x) [Enter]
```

と入力したときに描画されるグラフを図 7.24 に示します。

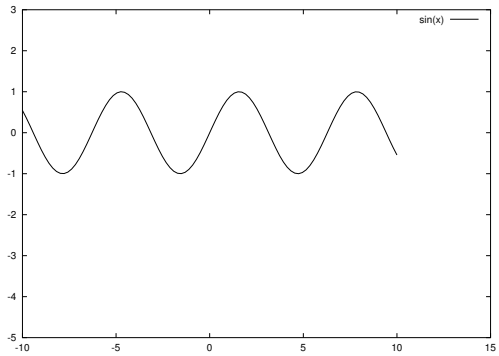


図 7.24: グラフとボックスのあいだの間隔の調整

表 7.15: グラフとボックスのあいだの間隔の調整に関連したコマンド

機能	コマンド
グラフとボックスのあいだの間隔を指定	set offsets 左, 右, 上, 下 [Enter]
グラフとボックスのあいだの間隔をなくす	set nooffsets [Enter]
現在のオフセットの情報を表示	show offsets [Enter]

7.1.15 グラフに作成日時を入れる

gnuplot には作成したグラフに作成日時を入れる機能があります。これらのコマンドを表 7.16 にまとめておきます。

```
set timestamp [Enter]
```

と入力してからグラフを描くと、グラフの左下に作成日時が入るようになります。

表 7.16: 作成日時に関連したコマンド一覧

機能	コマンド
グラフに作成日時を入れる	set timestamp [Enter]
グラフに作成日時を入れるのをやめる	set notimestamp [Enter]
作成日時の状態を表示する	show timestamp [Enter]

7.1.16 グラフィックスウィンドウに描画されたグラフを消す

グラフィックスウィンドウに描画されたグラフを消す（クリアする）には、

```
clear [Enter]
```

と入力します。グラフィックスウィンドウそのものは消えません。このコマンドは第 7.3 節で述べる 1 個のウィンドウ複数のグラフを重ね書きしてゆく場合に有用です。1 個のウィンドウにグラフを 1 個しか描いていないときにはあまり使い道がありません。

7.2 2 個以上の関数を同時にプロットする

2 個以上の関数を同時にプロットするときには、

```
plot 関数 1, 関数 2 [Enter]
```

のようにします。線のスタイルはこれらの関数が区別できるように自動的に調整されます。

各関数について線のスタイルなどのパラメータを指定することもできます。このような場合は、

```
plot 関数 1 with オプション,  
      関数 2 with オプション [Enter]
```

などのように指定します。上で「オプション」と書かれた部分にはこれまで説明してきたいろいろなコマンドが入ります。なお、紙面の都合でコマンドの途中で改行されていますが、これは上記の通りに入力するという意味ではありません。

以下に例を示します。

```
plot sin(x),cos(x) [Enter]
```

結果は図 7.25 のようになります。

線のスタイルなどのパラメータを指定する場合は、たとえば次のようになります。

```
plot sin(x) with lines lw 4, cos(x) with points pt 5 ps 3 [Enter]
```

結果は図 7.26 のようになります。

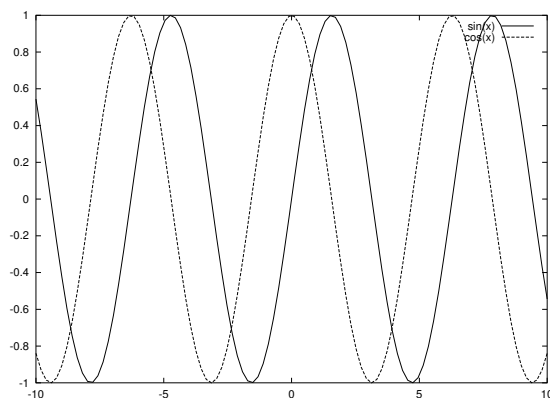


図 7.25: 2 個の関数の重ね書き

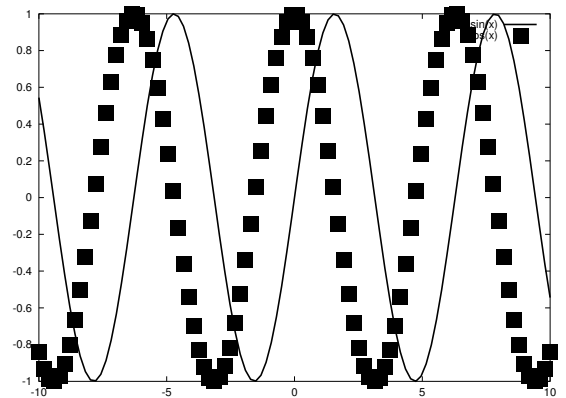


図 7.26: 2 個の関数の重ね書き (パラメータを指定)

7.3 グラフをどんどん重ね書きしてゆく

gnuplot でグラフを描くときには、plot が実行されるたびに以前描いてあったグラフが消え、新しいグラフが作成されました。では、1 枚の図にグラフをどんどん追加してゆくにはどうしたらよいのでしょうか？

このためには、multiplot というコマンドを使います。

multiplot の使い方を説明する前に、multiplot モードでは画面に描画したグラフをあらためてファイルに保存することはできないということを注意しておきます。gnuplot では描画したグラフを保存するときには

terminal と output の切り換え

という操作をおこなうわけですが、一旦 `multiplot` モードに入ってしまうと、この切り換えができません。ですから、`multiplot` モードで描画したグラフをファイルに保存するためには、

- `multiplot` モードに入る前に `terminal` を `postscript` などに指定し、さらに `output` に適当なファイルを指定しておく
- `xwd`, `xwpick`, `gimp` などのツールを使って描画したグラフをキャプチャする

のいずれかの操作をおこなう必要があります。前者の操作をおこなう場合は、ふつうは事前に試し書きをしてグラフを調整してから改めて描画をすべてやり直すことになり、若干面倒です。後者の操作では、得られるグラフの品質はかなり低くなります。

では、`multiplot` の使い方を説明しましょう。

```
set multiplot [Enter]
```

と入力すると、`gnuplot` のウィンドウは図 7.27 のような状態になり、さらに何も描かれていないグラフィックスウィンドウが画面にあらわれます。プロンプトが



図 7.27: `multiplot` モード

```
gnuplot>
```

から

```
multiplot>
```

に変わったことで、グラフの重ね書きモードに入っていることが確認できます。これ以降コマンド `plot` を実行するたびに、グラフがこのウィンドウにどんどん追加されてゆきます。ただし、描画範囲の自動調整などはうまくいかないことがあるので、`set origin`, `set size` といったコマンドを実行してグラフの原点や大きさを調整する必要があります。

では、例として

```
set multiplot [Enter]
set size 0.5 [Enter]
set origin 0,0 [Enter]
plot sin(x) [Enter]
set origin 0.5,0 [Enter]
plot cos(x) [Enter]
set origin 0,0.5 [Enter]
plot x**2 [Enter]
set origin 0.5,0.5 [Enter]
plot x**3 [Enter]
```

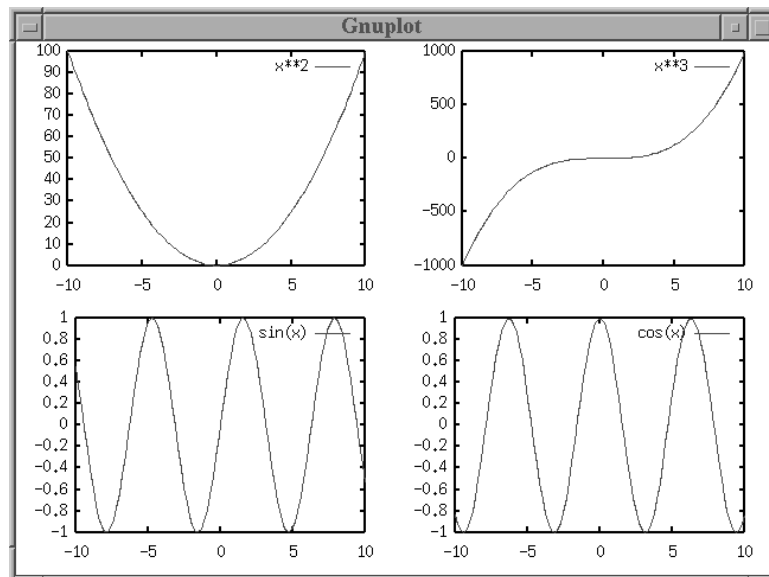



図 7.28: multiplot によって作成されたグラフの例

というコマンドを実行した場合に作成されるグラフを図 7.28 に示します． なお，先に述べたように，このようなグラフをそのまま PostScript 形式で保存するためのコマンドは gnuplot には用意されていません． ですから，グラフを保存したいときには，X Window System に用意されている xwd や xwpick などのコマンドを使う必要があります． これらのコマンドの使い方については UNIX システムのオンラインマニュアルを参照して下さい．

multiplot モードを抜けて通常モードに戻りたいときには，

```
set nomultiplot [Enter]
```

と入力します．

ここまでの結果をまとめておきましょう．

機能	コマンド
multiplot モードに入る	set multiplot [Enter]
multiplot モードを抜ける	set nomultiplot [Enter]

表 7.17: multiplot モードへの切り換え

7.4 直交座標を用いたパラメータ付き曲線のプロット

パラメータ付き曲線とは、変数 t を連続的に変化させたとき、

$$x = f(t), \quad y = g(t)$$

を満たす点が描く軌跡をいいます。

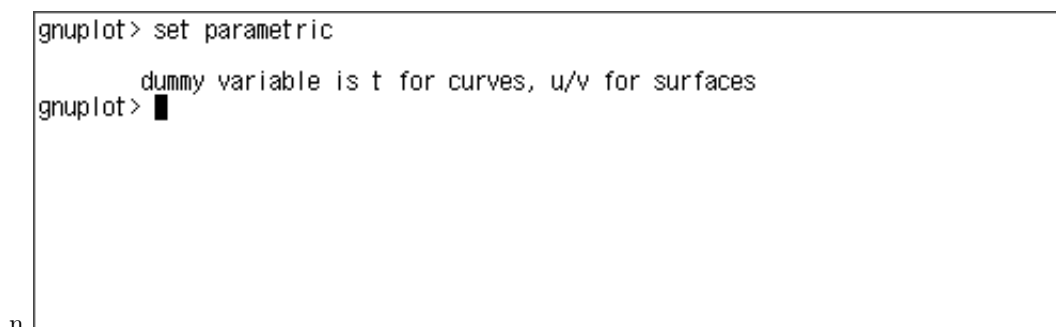
本節では 2 次元のパラメータ付き曲線を描画する方法について説明します。

7.4.1 パラメータ表示モードの起動と終了

パラメータ付きの曲線をプロットしたいときには、

```
set parametric [Enter]
```

と入力します。すると、gnuplot のウィンドウは図 7.29 のような状態になります。画面に表示される



```
gnuplot> set parametric
          dummy variable is t for curves, u/v for surfaces
gnuplot> █
```

図 7.29: parametric モードへの切り換え

```
dummy variable is t for curves, u/v for surfaces
```

というメッセージから、描画モードがパラメータ付き曲線モードに切り換わったことが確認できます。パラメータ付き曲線モードでは、変数は x から t に変わります。

パラメータ付き曲線モードを抜けるには

```
set noparametric [Enter]
```

と入力します。すると、gnuplot のウィンドウは図 7.30 のような状態になります。画面に表示される

```
dummy variable is x for curves, x/y for surfaces
```

というメッセージから、描画モードが通常のモードに戻ったことが確認できます。変数も t から x に戻ります。

7.4.2 基本的な描画コマンド

直交座標を用いてパラメータ付き曲線を描画するときには、コマンド `plot` に続いて、 x 軸の座標と y 軸の座標をあらわす関数を t の関数として指定します。

例として、円をパラメータ付き曲線としてプロットしてみましょう。この場合、 x 座標は $\cos t$ 、 y 座標は $\sin t$ です。

まず、直交座標によるパラメータ付き曲線描画モードに移行します。

```
gnuplot> set parametric
      dummy variable is t for curves, u/v for surfaces
gnuplot> set noparametric
      dummy variable is x for curves, x/y for surfaces
gnuplot> █
```

図 7.30: parametric モードを抜けたところ

```
set parametric [Enter]
```

続いて描画コマンドを入力します。

```
plot cos(t),sin(t) [Enter]
```

すると、図 7.31 のように楕円が描かれます。これは本来は円なのですが、gnuplot は標準ではグラフを横長で作成するためにこのようなグラフになってしまっています。

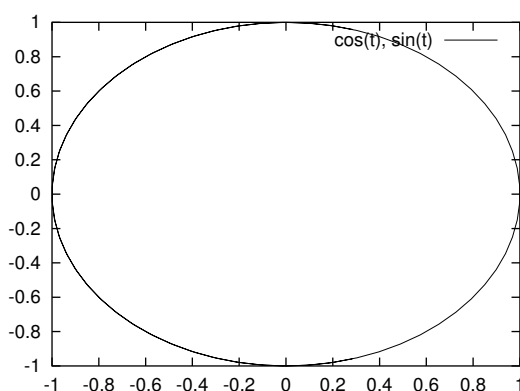


図 7.31: 円が楕円に見えているグラフ

上記の例では、真円を描画したいのに、描画結果は楕円になってしまいました。これは、画面の縦横の比率を指定しなかったからです。一般に、ふつうの関数のグラフは画面が横長の方が見やすいですが、パラメータ付き曲線の場合は、どのような画面を使うとグラフが見やすくなるかは曲線の形に大きく依存します。ですから、パラメータ付き曲線を描画するときには、試行錯誤によって見やすい画面の縦横の比率を自分で見付ける必要があります。画面の縦横の比率の指定の方法は第 7.1.11 節に記載されているので、ここでは例 7.31 のグラフを画面の縦横比率を 1 に変えて再描画する例を挙げるにとどめます。

上記のようにするためのコマンドは、以下の通りです。

```
set size ratio -1 [Enter]
plot cos(t),sin(t) [Enter]
```

とします。このようにすると、図 7.32 のように真円が描画されます。

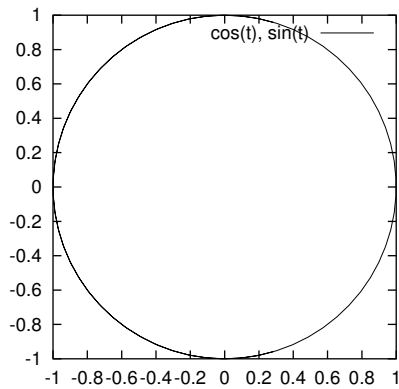


図 7.32: 真円の描画

7.4.3 パラメータ付き曲線の例

パラメータ付き曲線の例として、リサージュ図形と、アストロイドと呼ばれる図形を示しておきます。

リサージュ図形とは、

$$x = A_x \cos(\omega_x t + \psi_x), y = A_y \sin(\omega_y t + \psi_y)$$

なるパラメータ表示であらわされる曲線です。電気系の学生にとっては、オシロスコープの取り扱いの際にお馴染みの図形だと思います。まず、

$$x = \cos 3t, y = \sin t$$

の場合についてプロットしてみます。

描画コマンドは

```
plot cos(3*t),sin(t) [Enter]
```

です。描画結果は図 7.33 のようになります。 w_x と w_y の比率が複雑になる程、図形の形も複雑になります。

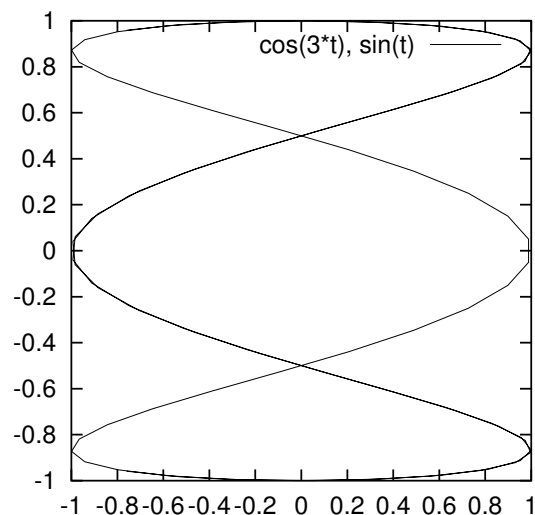


図 7.33: リサージュ図形

例として、

```
plot cos(11*t),sin(13*t) [Enter]
```

によって描画した結果 ($x = \cos(11t), y = \sin(13t)$) を図 7.34 に示します. w_x と w_y の比率が有理数でない場合,

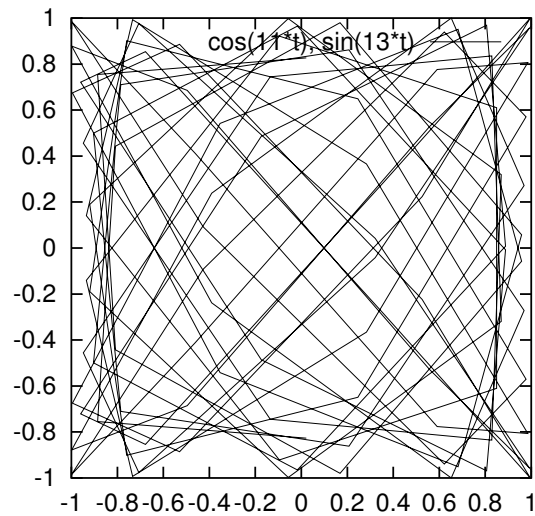


図 7.34: 複雑なリサージュ図形

リサージュ図形は閉曲線になりません.

次に、アストロイドと呼ばれる図形をプロットします. これは,

$$x = \cos^3 t, y = \sin^3 t$$

なるパラメータ表示を持つ曲線です.

描画コマンドは

```
plot cos(t)**3,sin(t)**3 [Enter]
```

です. 描画結果を図 7.35 に示します.

7.4.4 パラメータの範囲や描画範囲の指定

パラメトリック付き曲線の描画モードでは、コマンド `plot` に続けて、パラメータの範囲、横軸と縦軸の 3 種類が範囲を指定できます.

パラメータ、縦軸、横軸のそれぞれについてそれぞれ

- 指定なし
- 下限のみ指定
- 上限のみ指定
- 上限と下限を指定

の 4 種類の指定ができます. 指定されていない部分は自動的に調整されます. 以下にいくつか例を示します.

これとは別に、あらかじめ変数や半径の範囲を指定しておくこともできます.

変数の動く範囲を指定するには、

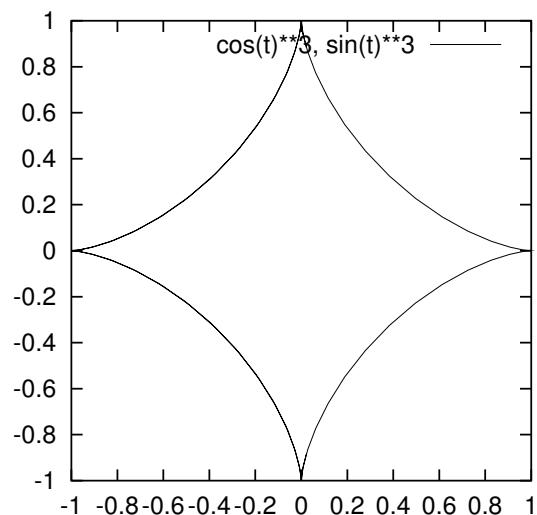


図 7.35: アストロイド

表 7.18: 描画範囲の設定例

機能	コマンド
パラメータ, 横軸, 縦軸とも指定しない	<code>plot cos(t),sin(t) [Enter]</code>
パラメータの上限のみ指定	<code>plot [:2*pi] cos(t),sin(t) [Enter]</code>
パラメータの下限のみ指定	<code>plot [0:] cos(t),sin(t) [Enter]</code>
パラメータの上限と下限を指定	<code>plot [0:2*pi] cos(t),sin(t) [Enter]</code>
パラメータは指定なし, 横軸の上限と下限を指定	<code>plot [] [-1:1] cos(t),sin(t) [Enter]</code>
横軸と縦軸の上限と下限を指定	<code>plot [] [-1:1] [-1:1] cos(t),sin(t) [Enter]</code>
パラメータ, 横軸, 縦軸の上限と下限を指定	<code>plot [0:2*pi] [-1:1] [-1:1] cos(t),sin(t) [Enter]</code>

`set trange [下限:上限] [Enter]`

などします。横軸と縦軸の範囲を指定するには、それぞれ

`set xrange [下限:上限] [Enter]`

`set yrange [下限:上限] [Enter]`

などします。trange の初期値は [-10:10] に、xrange の初期値は [-10:10] になっています。

7.5 極座標を用いたパラメータ付き曲線のプロット

極座標を用いたパラメータ付き曲線のプロットとは、図 7.36 に示すように、曲線の軌跡を、原点と曲線上の点を結ぶ半直線の角度 t とその半直線の長さ $r(t)$ の関係によって表現したものをいいます。

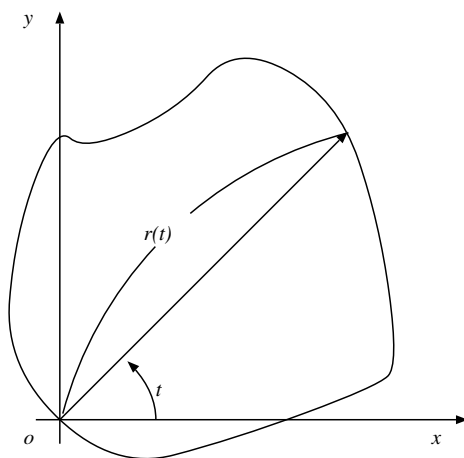


図 7.36: パラメータ付き曲線の極座標表示

本節では、極座標を用いてパラメータ付き曲線をプロットする方法について説明してゆきます。

7.5.1 極座標モードの起動と終了

パラメータ付きの曲線をプロットしたいときには、まず

```
set polar [Enter]
```

と入力します。すると、描画モードが極座標を用いたパラメータ付き曲線モードに切り換わります。7.4 節と同様に、独立変数は x から t に変わります。

パラメータ付き曲線モードを抜けるには

```
set nopolar [Enter]
```

と入力します。すると、独立変数が t から x に戻ります。

```
set polar [Enter]
```

として極座標を用いたパラメータ付き曲線描画モードにした状態で、

$$r = f(t)$$

なる曲線（たとえば $f(t) = \sin t$, $f(t) = t^2$ など）をプロットするには、

```
plot f(t) [Enter]
```

とします。すると、変数 t を連続的に変化させたときに原点からの角度が t で長さが $f(t)$ の弦の先頭が描く軌跡が画面に描画されます。

7.6 簡単な例

いくつか例を見てみましょう。
まず、

$$r = \sin 2t$$

なる図形をプロットしてみます。これは、4つ葉のクローバーの形の図形です。ただし、描画にあたって、画面の縦横比率を1に固定しておくことにします。

まず

```
set size ratio [Enter]  
set polar [Enter]
```

と入力して画面の縦横比率を1にしてから極座標を用いたパラメータ付き曲線描画モードに移行し、続いて

```
plot sin(2*t) [Enter]
```

と入力すると、図 7.37 のようなグラフが得られます。

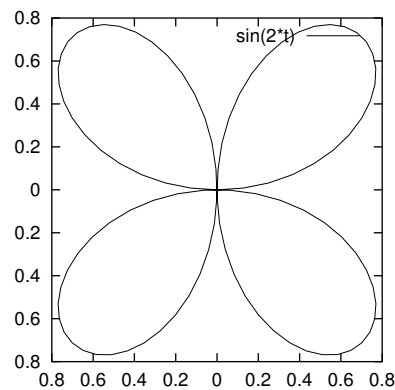


図 7.37: 4つ葉のクローバーの形の図形

上のグラフにおいて正弦関数の周期を変えると葉の数が変わります。たとえば、葉の数を8枚にするには、

$$r = \sin 4t$$

なる極座標表示を用います。実際、

```
plot sin(4*t) [Enter]
```

というコマンドを実行すると、図 7.38 のような図形が得られます。

7.6.1 変数と描画範囲の指定

変数や描画範囲を指定する方法は直交座標を用いた場合と同じです。いくつか例を挙げます。

これとは別に、あらかじめ変数や半径の範囲を指定しておくこともできます。

変数の動く範囲を指定するには、

```
set trange [下限:上限] [Enter]
```

などとします。また、描画される半径の範囲を指定するには、

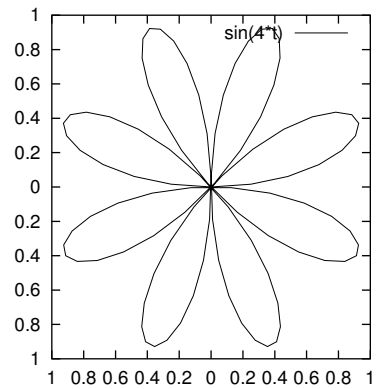


図 7.38: 4 つ葉のクローバーの形の図形

表 7.19: 変数や描画範囲の指定

機能	コマンド
変数の動く範囲のみを指定	plot [0:pi] sin(2*t) [Enter]
半径の範囲のみを指定	plot [0:pi] sin(2*t) [Enter]

set rrange [下限:上限] [Enter]

などします。極座標モードに入ったときには、trange の初期値は $[0:2\pi]$ に、rrange の初期値は $[0:10]$ になっています。

なお、描画にあたってサンプル点の数を十分多く取っておかないと、サンプル点が少ないために予想に反する奇妙な図形が得られることがあります。極座標を用いたパラメータ付き曲線描画モードを使うときは、あらかじめ

set samples サンプル点の数 [Enter]

と自分で指定しておくといいでしょう（「サンプル点の数」のところには適当な自然数が入ります）。サンプル点の数の初期値は 100 ですが、100 では足りないことも多いです。

第8章 いろいろな3次元グラフを描画する

8.1 直交座標系を用いた3次元グラフの描画

3次元グラフをプロットするときにはコマンド `plot` のかわりにコマンド `splot` を使います。

8.1.1 単純な3次元グラフの表示

まず単純なグラフを描画してみましょう。

```
splot exp(-(x**2+y**2)/20) [Enter]
```

と入力すると、図 8.1 のような結果が得られます。

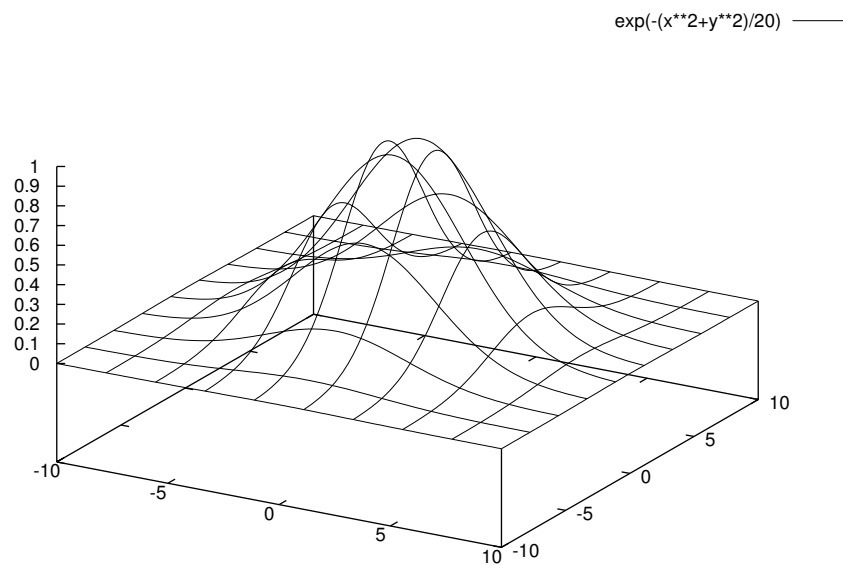


図 8.1: 関数 $\exp[-(x^2 + y^2)/20]$ のプロット

プロットが終わったグラフを `eps` ファイルの形式で保存するには、

```
set terminal postscript eps [Enter]
set output 'ファイル' [Enter]
replot [Enter]
```

とします。ここに、「ファイル」と書かれた部分には適当なファイル名が入ります。

8.1.2 描画範囲の調整

3次元グラフを描画するときには、x軸、y軸およびz軸の範囲をそれぞれ指定できます。基本的な構文は

```
splot [xの下限:xの上限] [yの下限:yの上限] [zの下限:zの上限] 関数 [Enter]
```

というものです。ただし、後半の部分を省略して、最初の1個だけ、あるいは最初の2個だけのパラメータを指定することもできます。途中の部分を省略するときには省略したい部分に [] と書きます。

いくつか例を示します。

表 8.1: 各軸の範囲の指定の例

機能	コマンド
xの範囲のみ指定	splot [0:1] sin(x+y) [Enter]
xとyの範囲を指定	splot [0:4] [0:1] sin(x+y) [Enter]
x,y,zの範囲を指定	splot [0:4] [0:1] [0:4] sin(x+y) [Enter]
x,zの範囲を指定	splot [0:4] [] [0:1] sin(x+y) [Enter]

ここで注意してほしいのは、splotにおける各軸の範囲指定は、先に実行されたコマンドの影響を受けるということです。gnuplotを起動したときにはどの軸の範囲もデフォルトで verb?[-10:10]?になっているのですが、たとえば

```
splot [0:1] sin(x+y) [Enter]
```

というコマンドを実行すると、x軸のデフォルトの範囲は[0:1]に変わってしまいます。なお、コマンド plot にはこのようなことはありません。

あらかじめx軸、y軸、z軸の範囲を指定しておくこともできます。このためには、

```
set xrange [下限:上限] [Enter]
```

```
set yrange [下限:上限] [Enter]
```

```
set zrange [下限:上限] [Enter]
```

のように入力します。ただし、「下限」と書かれた部分と「上限」と書かれた部分には適当な数が入ります。このようにして指定した場合も、先の例のように

```
splot [0:1] sin(x+y) [Enter]
```

と明示的に範囲指定して splot を使うと、先ほど指定した範囲は上書きされてしまいます。

8.1.3 グラフにかかった網目の大きさを変える

splot は標準では3次元グラフに横軸に10個、縦軸に10個の目がある網をかけて表示しています。しかし、この状態では、網の目が粗すぎて関数の様子をうまく表示できないことがあります。

このような場合には、set isosamples というコマンドを使うことで網目の数を調整することができます。

コマンド set isosamples の使い方は、

```
set isosamples 横軸の網目, 縦軸の網目 [Enter]
```

です。ここに、「横軸の網目」、「縦軸の網目」と書かれた部分には適当な整数が入ります。

例題として、図 8.1 のグラフにおいて、網目を横軸 50、縦軸 50 に切り直してみます。

```
set isosamples 50,50 [Enter]
```

```
splot exp(-(x**2+y**2)/20) [Enter]
```

結果は図 8.2 のようになります。

$$\exp(-(x^2+y^2)/20)$$

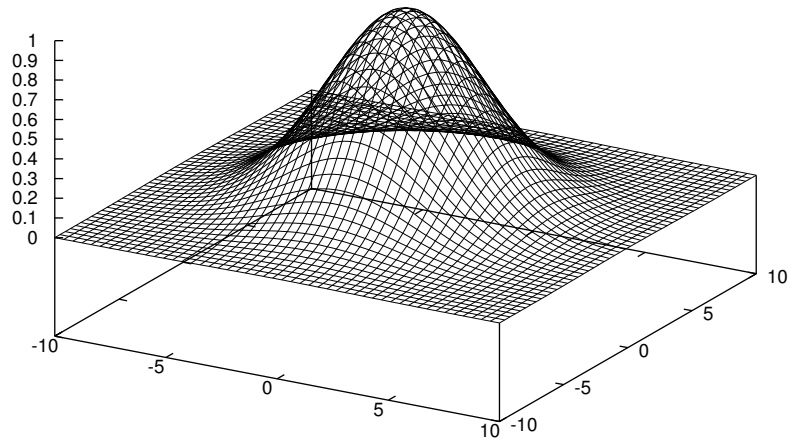


図 8.2: 網目の切り直しの例

8.1.4 隠れ面の表示の切り換え

splot で 3 次元グラフを網目を使って表示しているとき、画面に描画されるのは、網目のある方向から見た図、すなわち、ある方向に射影した像です。さて、現実の世界で 3 次元図形を見るときには、目に見えるのは一番近い図形のみで、その影に隠れた図形（隠れ面）は見えません。ところが、gnuplot は、標準状態では他の網目に隠された面也表示します。この隠れ面を表示しないようにするには、set hidden3d というコマンドを使います。隠れ面の表示と非表示の切り換えのコマンドを表 8.2 にまとめておきます。

表 8.2: 隠れ面の表示切り換え

機能	コマンド
隠れ面を表示しない	set hidden3d [Enter]
隠れ面を表示する	set nohidden3d [Enter]

例として、

```
set hidden3d [Enter]
splot exp(-(x**2+y**2)/20) [Enter]
```

とした場合の図を 8.3 に示します。図 8.1 と見比べてみて下さい。

8.1.5 視点の変更

3 次元グラフを画面に表示するときには、視点の位置によってグラフが見やすくなったり見にくくなったりします。

そこで、グラフの見やすさを調整するために、set view というコマンドが用意されています。

$$\exp(-(x^2+y^2)/20) \text{ ———}$$

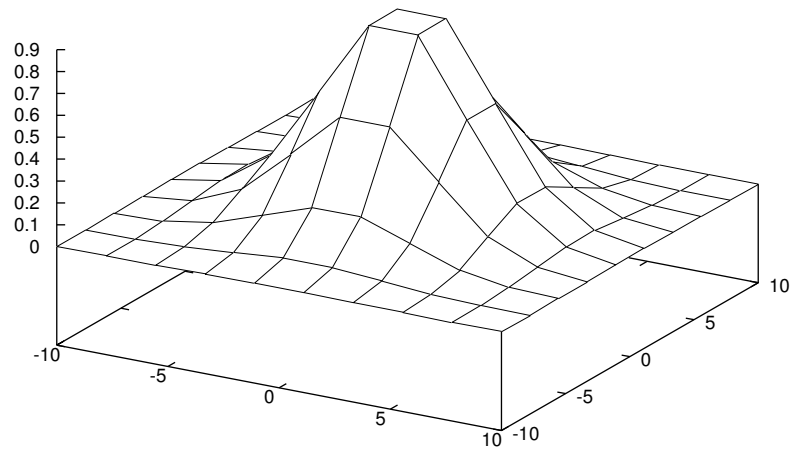


図 8.3: 隠れ面を表示しない場合のグラフ

われわれの視点（モニタの正面）から見たグラフが描かれる座標系は，初期状態ではモニタの左右方向が x 軸，モニタの上下方向が y 軸，モニタに立てた垂線方向が z 軸になっています．

この座標系を最初に x 軸のまわりで回転させ，さらに新しい z 軸のまわりで回転した座標系がグラフが実際に描画される座標系になります（図 8.4）．

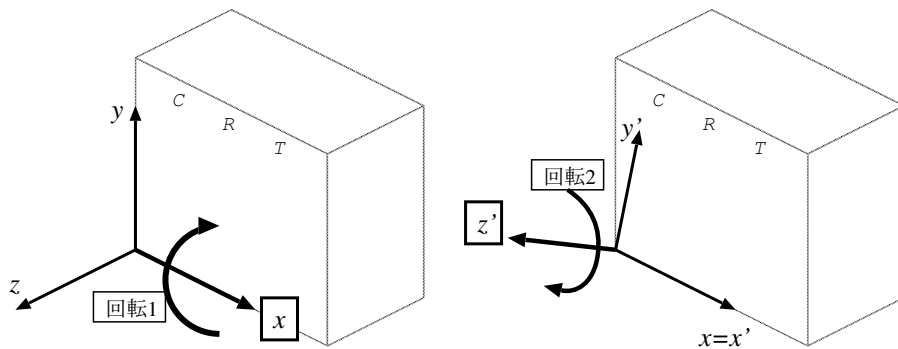


図 8.4: 座標系の回転

我々の視点はずねにモニタの正面に固定されていて，グラフが描画される座標系の方が回転しているということに注意して下さい． コマンド名 `set view` からは視点の方を移動することが連想されますが，これは実際の挙動とは異なります．

さらに，グラフ全体の縮尺と z 軸の縮尺を指定することで，実際にグラフがどのように描かれるかが決まります．

以上の説明をふまえた上で，コマンド `set view` の使い方を紹介しましょう．

コマンド `set view` は

`set view` 回転 1, 回転 2, グラフの拡大率, z 軸の拡大率 [Enter]

のようにして使います。回転 1 は x 軸のまわりの回転の角度で、0 度から 180 度のあいだの値を取ります。回転 2 は少しわかりにくいのですが、回転 1 で回転したあとの座標系に関する z 軸まわりの回転で、0 度から 360 度の値を取ります。回転 1 と回転 2 の意味を図 8.4 に示します。

拡大率には任意の数が指定できます。また、回転 2 以降は省略可能です。

gnuplot を起動したときには、座標系は

```
set view 60, 30, 1, 1 [Return]
```

を実行した場合と同じ位置にあります。

以下に、座標軸を回転させたときの例を示します。

まず最初に、回転角 1、回転角 2 をともに 0、0 に指定してみましょう。

```
set view 0,0,1,1 [Return]
splot exp(-(x**2+y**2)/20) [Return]
```

結果は図 8.5(a) のようになります。 z 軸がモニタの画面に垂直で、グラフを真上から見ていることになるので、関数の形はまったくわかりません。続いて、回転角 1 を 30 度ずつ増やし、回転角 2 は零のまま固定しててみます。結果を図 8.5(b), (c), (d) に示します。結果を図 8.5(d) では、グラフを真横から見た図が表示されています。

続いて、回転角 1 を 10 度に固定して、回転角 2 を 0 度、30 度、60 度、90 度とした場合のグラフをプロットしてみましょう。

8.1.6 水平面 (座標が書かれる面) の位置の変更

gnuplot の 3 次元グラフは、適当な高さに座標平面 (xy 平面) が描かれ、その上あるいは下に関数の形が網のかかった面の形で表示される、という構造になっています。そして、表示される座標平面の高さが適切でないこともあります。

このような場合に座標平面の表示される高さを調整するためのコマンドが `set ticslevel` です。

コマンド `ticslevel` の使い方は、

```
set ticslevel p [Enter]
```

です。ここに、 p と書かれた部分には数値が入ります。

数値 p がどのようにして決まるかというと、

$$p = \frac{\text{座標平面の } z \text{ 座標} - z \text{ 軸の最小値}}{z \text{ 軸の最小値} - z \text{ 軸の最大値}}$$

という式にしたがって決まります。

この式の分母において z 軸の最小値から z 軸の最大値が引かれていることに注意して下さい。このため、 p が正の値のときには、座標平面は必ず z 軸の最小値より下側に書かれます。座標平面を z 軸の最小値より上に描画したいときには、 p を負の値に設定する必要があります。

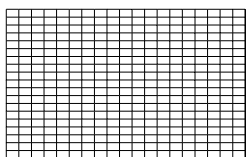
gnuplot を起動したときには `ticslevel` は 0.5 に設定されています。

現在の `ticslevel` の設定を知りたいときには、

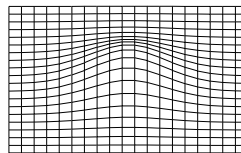
```
show tics [Enter]
```

と入力します。

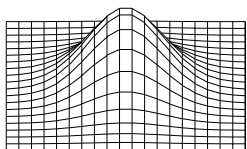
座標平面の表示に関連したコマンドを表 8.3 にまとめておきます。



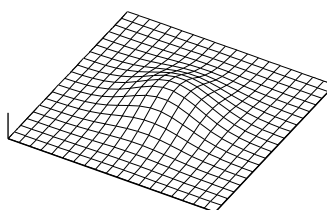
(a) 回転角 1=0, 回転角 2=0



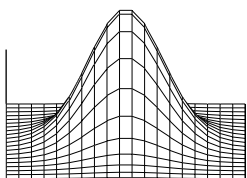
(a) 回転角 1=10, 回転角 2=0



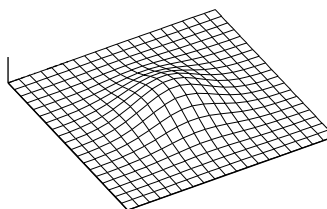
(b) 回転角 1=30, 回転角 2=0



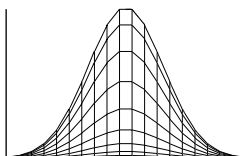
(b) 回転角 1=10, 回転角 2=30



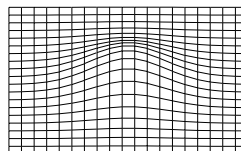
(c) 回転角 1=60, 回転角 2=0



(c) 回転角 1=10, 回転角 2=60



(d) 回転角 1=90, 回転角 2=0



(d) 回転角 1=10, 回転角 2=90

図 8.5: 回転角 1 の指定

図 8.6: 回転角 2 の指定

表 8.3: 座標平面の表示に関連したコマンド

機能	コマンド
座標平面の場所を設定する	set ticslevel 位置 [Enter]
座標平面の設定を調べる	show tics [Enter]

8.1.7 網のかかった面の表示をやめる

網のかかった面の表示をやめる方法もあります。これは、次の 8.1.8 節で述べる等高線の表示と組み合わせで使われます。

網のかかった面を表示するには `set surface` というコマンドを、表示をやめるには `set nosurface` というコマンドを使います。これらを表 8.4 にまとめておきます。

表 8.4: 網のかかった面の表示に関連したコマンド

機能	コマンド
等高線を表示する	set surface [Enter]
等高線を表示しない	set nosurface [Enter]

8.1.8 等高線の表示

8.1.8.1 等高線をプロットする

グラフを等高線を表示することもできます。このためには、`set contour` というコマンドを使います。

`set contour` を実行してから関数をプロットすると、その関数の等高線が表示されます。しかし、これにはいくつか問題があります。

```
set contour [Enter]
plot exp(-(x**2+y**2)/20) [Enter]
```

とすると、結果は図 8.7 のようになります。

図 8.7 には、

- 等高線を斜めの視点から見ると見にくい
- 網かけされたグラフが邪魔

という問題があります。

より見やすい等高線グラフを作成するには、8.1.5 節の視点変更と 8.1.7 節の網かけ面の非表示を組み合わせる必要があります。さらに、このままでは図が粗いので、8.1.3 節の網目の切り直しを併用した方がよいでしょう。さらに、見やすいようサイズも適当に調整すべきです。

以上を踏まえた上で、

```
set nosurface [Enter]
set contour [Enter]
set size 0.66,1 [Enter]
set view 0,0,1,1 [Enter]
```

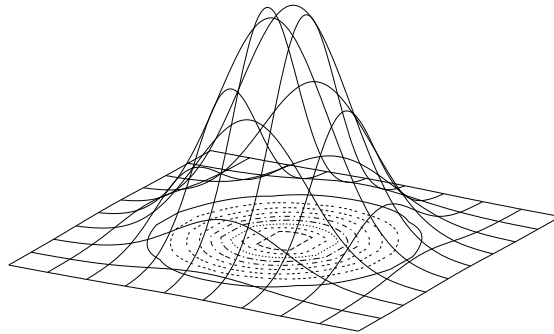



図 8.7: 等高線のプロット (望ましくない例)

```
set isosamples 50,50 [Enter]
splot exp(-(x**2+y**2)/20) [Enter]
```

という一連のコマンドを実行すると、図 8.8 のような等高線グラフが得られます。

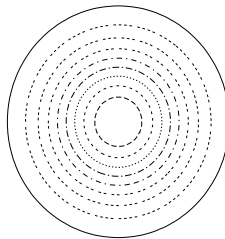


図 8.8: 等高線のプロット

ここまでに出てきた等高線の表示に関連したコマンドを表 8.5 にまとめておきます。

表 8.5: 等高線の表示に関連したコマンド

機能	コマンド
等高線を表示する	<code>set contour [Enter]</code>
等高線を表示しない	<code>set nocontour [Enter]</code>
等高線の高さを指定した書式で表示する	<code>set clabel '書式' [Enter]</code>
等高線の高さを表示しない	<code>set noclabel [Enter]</code>
現在の等高線の高さの表示形式を見る	<code>show clabel [Enter]</code>

8.1.8.2 等高線の高さ表示のフォーマットを変える

等高線グラフを描いたときには、図の右側に各等高線の数値が表示されます。この数値を変更するときには、`set clabel` というコマンドを使います。

`set clabel` の使い方は、

```
set clabel 'フォーマット' [Enter]
```

というものです。ただし、'フォーマット'の部分には、C言語と同じ構文の浮動小数点数の書式指定が入ります。書式指定のしかたについては7.1.8.2節に説明されているものと同じなので、そちらを参照して下さい。標準の書式は`%8.3g`になっています。

等高線の高さの数値を表示したくないときには、

```
set noclablel [Enter]
```

とします。

また、現在の現在の等高線の高さの表示形式を見たいときには

```
show clabel [Enter]
```

とします。

8.1.8.3 等高線のよりきめ細かい制御

等高線をよりきめ細かく制御したいときには、`set cntrparam` というコマンドを使います。このコマンドを使う前には、等高線を描画するときには `gnuplot` で標準的に設定されている値が使われます。

等高線のグラフを描くときには、等高線を描くべき高さにある標本点をいくつか取ってからこれらのあいだを線で結んでゆくのですが、この結び方や標本点の高さの取り方がこのオプションによって決まります。

表 8.6 に、`set cntrparam` で指定できる事項の一覧を示します。

表 8.6: 等高線の制御のためのコマンド

機能	コマンド
標本点のあいだを直線でむすぶ	set cntrparam linear [Enter]
標本点のあいだをなめらかな直線でむすぶ	set cntrparam cubicspline [Enter]
標本点を適当に避けるなめらかな線を引く	set cntrparam bspline [Enter]
cubicspline と bspline で線のなめらかさを指定する (注 1)	set cntrparam points 数 (注 2) [Enter]
bspline において線のなめらかさを指定する (注 3)	set cntrparam order 数 [Enter] (注 4)
等高線の数を実動調整する	set cntrparam levels auto [Enter]
自動調整で指定された数の等高線を引く	set cntrparam levels auto 数 [Enter] (注 5)
指定された本数の等高線を引く	set cntrparam levels 数 [Enter] (注 6)
指定された高さの点に等高線を引く	set cntrparam levels discrete z_1, z_2, \dots, z_n [Enter] (注 7)
指定された間隔で等高線を引く	set cntrparam levels incremental 初期値, 刻み, 終了値 [Enter] (注 8)
等高線の設定を見る	show contour [Enter]

注 1 gnuplot は、曲線を描画するときには、曲線を小さな直線の集まりで近似している。このパラメータは、近似直線をどのくらい細かく取るかを決定する。

注 2 「数」の部分には適当な整数が入る。値が大きいくほど曲線はなめらかになる。

注 3 set cntrparam points の方は曲線が標本点の近くを通るときは影響を与えないが、set cntrparam order の方はこの通り方に影響を与える。

注 4 「数」の部分には 2 から 10 までの整数が入る；値が大きいくほど等高線がなめらかになるが、標本点からは大きく外れる

注 5 「数」の部分には適当な整数が入る。ただし、正確に指定された数の等高線が引かれるわけではない。

注 6 「数」の部分には適当な整数が入る

注 7 z_1, z_2, \dots, z_n の部分には等高線が引かれる高さ (z 座標) をコンマで区切ってならべて指定する；等高線の数はいくつでもよい

注 8 「初期値」、「刻み」、「終了値」には適当な数値が入る；初期値から終了値まで刻みで指定された間隔で等高線が引かれる；終了値は省略できる

8.1.9 グラフの表示スタイルを網かけや等高線以外にする

7.1.4 節と同様に、グラフの表示スタイルを網かけ以外のものに変更することもできます。方法は 7.1.4 節と同様なので、ここではいくつか簡単な例を紹介します。まず、グラフをインパルスで表示してみましょう。

```
splot exp(-(x**2+y**2)/20) with impulses [Enter]
```

とすると、図 8.9 のような結果が得られます。

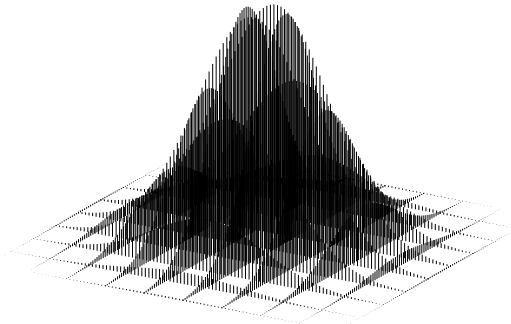


図 8.9: 3 次元グラフのインパルス表示

次に、グラフを点のみで表示してみましょう。

```
splot exp(-(x**2+y**2)/20) with dots [Enter]
```

とすると、図 8.10 のようなグラフが得られます。

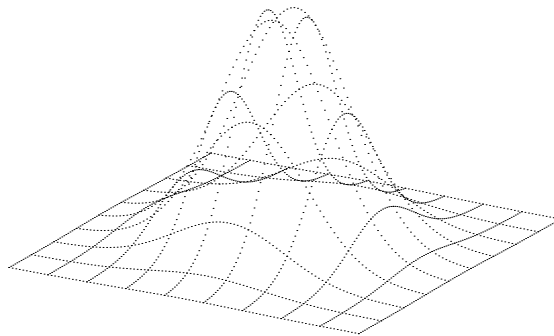


図 8.10: 3 次元グラフの点を使った表示

8.1.10 PostScript ファイルの BoundingBox について

gnuplot で 3 次元グラフを描画し、グラフを PostScript ファイルに保存して文書に取り込むと、グラフのまわりに大きい空白ができてしまうことがあります。このような場合、PostScript ファイル中の BoundingBox というパラメータを手動で編集すれば、空白を小さくすることができます。

BoundingBox は、PostScript ファイルで、図を描画するために確保する領域の左下の点と右上の点の座標を指定するパラメータです。通常、PostScript ファイルの上から数行のところで指定されています。パラメータの指定の順番は、

%%BoundingBox: 左下隅の x 座標 左下隅の y 座標 右上隅の x 座標 右上隅の y 座標

のようになります。座標軸の原点は画面左下隅で、座標軸は x 軸が画面横方向、y 軸が画面縦方向です。BoundingBox を試行錯誤によって図が実際に描画されている領域に合わせれば、余分な空白を小さくすることができます。ただし、BoundingBox に変な値を指定してしまうと、文書に図を取り込んだときに適切に表示されなくなります。BoundingBox を手動で編集するにはそれなりのリスクがあることを認識して下さい。

図 8.11 に、実際の PostScript ファイルで Bounding Box が指定されている部分を示します。

```
%!PS-Adobe-2.0 EPSF-2.0
%%Title: gnuplot-3d-contour-2.eps
%%Creator: gnuplot 3.7 patchlevel 1 (+1.2.0 2001/01/11)
%%CreationDate: Tue May 25 16:07:57 2004
%%DocumentFonts: (atend)
%%BoundingBox: 100 130 237 232 << ここが BoundingBox
%%Orientation: Portrait
%%EndComments
```

図 8.11: BoundingBox の指定箇所

8.2 パラメータ表示された 3 次元グラフ

パラメータ表示された 3 次元グラフとは、適当な関数 $f(u, v)$, $g(u, v)$, $h(u, v)$ に対して

$$x = f(u, v), \quad y = g(u, v), \quad z = h(u, v)$$

を満たす点の全体のことを言います。この図形は多くの場合は 3 次元中の曲面になります。

本節ではパラメータ表示された 3 次元グラフを描画する方法について説明してゆきます。

8.2.1 パラメータ表示モードの起動と終了

パラメータ表示された 3 次元グラフを表示したいときには、パラメータ表示された 2 次元グラフを表示するときと同様に、

```
set parametric [Enter]
```

と入力します。すると、画面は図 8.12 のような状態になります。図 8.12 からわかるように、パラメータ付き曲線および曲面モードは、2 次元と 3 次元で共通です。そして、コマンド `plot` を使うと 2 次元のパラメータ付き曲線が、コマンド `splot` を使うと 3 次元のパラメータ付き曲線や曲面が描画されます。`plot` でパラメータ付き曲線を描画するときには独立変数として `t` を使用しましたが、`splot` でパラメータ付き曲線や曲面を描画するときには独立変数として `u` と `v` を使用します。

パラメータ表示モードを抜けるには、

```
set noparametric [Enter]
```

と入力します。

```
gnuplot> set parametric
      dummy variable is t for curves, u/v for surfaces
gnuplot> █
```

図 8.12: パラメータ付き曲面描画モード

8.2.2 簡単な例題

この節では、例題として、いくつかの簡単なパラメータ表示された 3 次元グラフを描画してみます。まず、トーラス（ドーナツ型）をプロットしてみます。トーラスのパラメータ表示は、たとえば

$$x = \cos u(3 + \cos v), \quad y = \sin u(3 + \cos v), \quad z = \sin v$$

によって与えられます。これをプロットしてみましょう。

あらかじめ

```
set parametric [Enter]
```

とした上で、

```
splot cos(u)*(3+cos(v)),sin(u)*(3+cos(v)),sin(v) [Enter]
```

と入力すると、図 8.13 のように、トーラスが描画されます。残念ながら、図 8.13 の描画は見やすいとはい

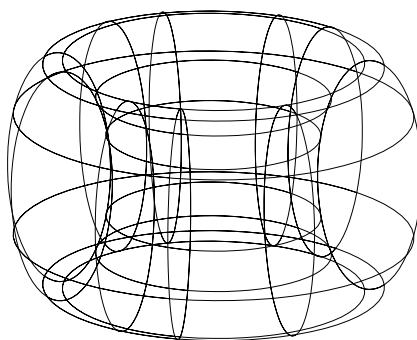


図 8.13: トーラス

難しいです。この理由は、パラメータの範囲が適切でないこと、隠れ面の処理がなされていないこと、網目の切り方が粗すぎることにあります。これらを調整する方法を以下で順に見てゆきます。

8.2.3 描画範囲の調整

パラメータ付きの3次元グラフを描画するときには、変数 u, v および x 軸, y 軸および z 軸の範囲をそれぞれ指定できます。基本的な構文は

```
splot [u の下限:u の上限] [v の下限:v の上限]  
      [x の下限:x の上限] [y の下限:y の上限] [z の下限:z の上限] 関数 [Enter]
```

というものです。ただし、後半の部分を省略することもできます。途中の部分を省略するときには省略したい部分に `[]` と書きます。なお、紙面の都合で途中で余分な改行が入っていますが、この改行は実際に入力するわけではありません。

具体的な指定の仕方については、8.1.2 節と同様ですので説明を省略します。

なお、あらかじめ変数 u, v および x 軸, y 軸, z 軸の範囲を指定しておくこともできます。このためには、

```
set urange [下限:上限] [Enter]  
set vrange [下限:上限] [Enter]  
set xrange [下限:上限] [Enter]  
set yrange [下限:上限] [Enter]  
set zrange [下限:上限] [Enter]
```

のように入力します。ただし、「下限」と書かれた部分と「上限」と書かれた部分には適当な数が入ります。

トーラスを変数 u と v の範囲をそれぞれ 0 から 2π までに設定してプロットしている例を以下に示します。

```
splot [0:2*pi] [0:2*pi] cos(u)*(3+cos(v)),sin(u)*(3+cos(v)),sin(v) [Enter]
```

結果は図 8.14 のようになります。これだけでも、図 8.13 より幾分かは見やすくなっています。

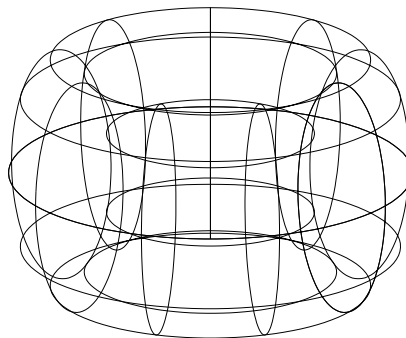


図 8.14: トーラス:変数の範囲を適切に設定

8.2.4 隠れ面や網の切り方の処理

図 8.14 でも、隠れ面が処理されていないのと、網の切り方が粗すぎるために、まだあまり見やくありません。これを修正するにはどうしたらよいでしょうか？

隠れ面を処理する方法や網の切り方をより細かくする方法は 8.1.3 節, 8.1.4 節で述べられている通りです. すなわち, パラメータ `isosamples` の値を大きくすれば網目は細かくなりますし, `set hidden3d` とすれば隠れ面は表示されなくなります.

では, これらを併用してみましょう.

```
set hidden3d [Enter]
set isosamples 40,40 [Enter]
splot [0:2*pi] [0:2*pi] cos(u)*(3+cos(v)),sin(u)*(3+cos(v)),sin(v) [Enter]
```

としてみます. すると, 結果は図 8.15 のようになります. こんどは非常に見やすいドーナツ型の図が得られ

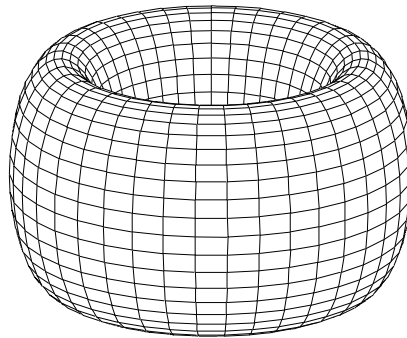


図 8.15: トーラス:隠れ面の処理とより細かい描画

ています.

8.2.5 その他の各種のオプション

図の標題の付け方などは通常の 2 次元グラフや 3 次元グラフと同じです. 詳細については第 6 章, 第 7 章および第 8 章などを参照して下さい.

8.2.6 3 次元のパラメータ付き曲線を表示する

3 次元のパラメータ付き曲線 (曲面ではなく) を表示したいときには, 変数 u, v のどちらか一方だけを使えばよいです.

例として螺旋をプロットしてみましょう.

```
set parametric [Enter]
set isosamples 40,40 [Enter]
splot [0:10*pi] u,sin(u),cos(u) [Enter]
```

結果は図 8.16 のようになります.

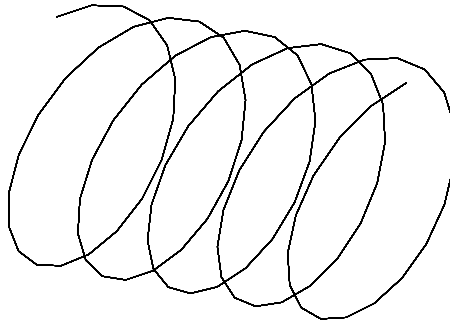


図 8.16: 螺旋の描画

8.2.7 注意事項

gnuplot-3.7.1 におけるコマンド `splot` はパラメータ表示モードでは誤動作あるいは暴走することがあります。

第9章 データファイルのプロット

データファイルとは、実験の測定値や統計データのような数値が規則正しく並べられたファイルのことをいいます。本節では、データファイルに記載されたデータ点からグラフを作成する方法について説明します。

9.1 2次元データのプロット

9.1.1 単純なデータファイルのプロット

データファイルをプロットするときには、

```
plot 'ファイル' [Enter]
```

とします。ただし、「ファイル」と書かれた部分にはデータファイルの名前が入ります。

例として、図 9.1 のような単純なデータファイルがあった場合を考えます。ファイル名を 'sample1.dat' とします。



```
20
60
80
60
100
```

図 9.1: データファイル sample.dat の内容

このデータファイルをプロットするには、

```
plot 'sample.dat' [Enter]
```

とします。すると、図 9.2 のようなグラフが描画されます。ただし、データ点の表示が小さすぎると、データ点が記号「+」で標記されていて、座標軸と重なると見えなくなってしまうため、あまり見やすくありません。

第 6 章および第 7 でも説明したように、データ点の表示を切り換えるには、`pointtype` (略称 `pt`) を指定します。どのような指定をするとどのような点が表示されるかについては、第 7 章の図 7.15 と図 7.16 (56 ページ) を参照して下さい。また、`pointsize` (略称 `ps`) を指定すると、点の大きさを変えることができます。ここでは、`+` 印を使ってデータ点を表示し直してみることになります。`+` 印に対応するのは、`pointtype 6` (略称 `pt 6`) です。また、点の大きさを 2 に指定 `pointsize 2` (略称 `ps 2`) とすることになります。

上記のような設定で描画するには、

```
plot 'sample.dat' with points pt 6 ps 2 [Enter]
```

とします。すると、図 9.3 のようなグラフが得られます。

次に、データ点のあいだを線で結んでみます。このためには、

```
plot 'sample.dat' with lines [Enter]
```

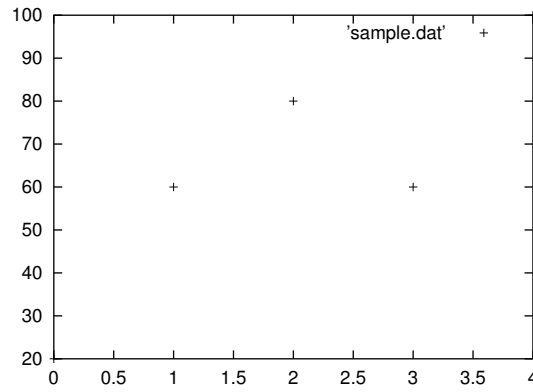


図 9.2: データファイル sample.dat のプロット (1)

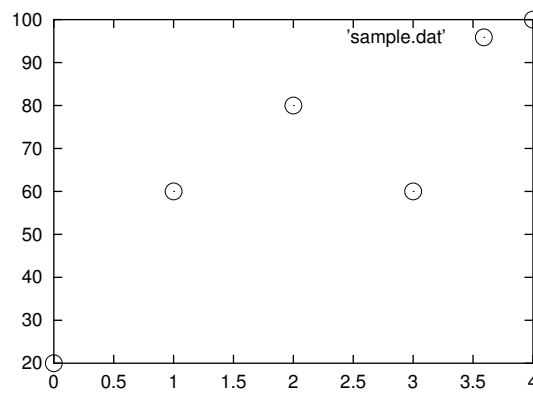


図 9.3: データファイル sample.dat のプロット (2)

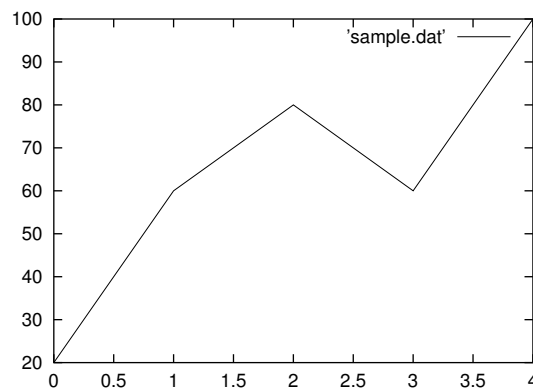


図 9.4: データファイル sample.dat のプロット (3)

とします．ここに、`with lines` というのは、データ間を線で結ぶことを意味するキーワードです．描画結果は図 9.2 のようになります．

データ点のあいだを線で結び、かつデータ点自身も明示したいときには、`linespoints` というキーワードを利用します．キーワード `linespoints` を利用するときには、`pointtype` , `pointtype` も指定できます．たとえば、

```
plot 'sample.dat' with linespoints pt 6 ps 2 [Enter]
```

とすると、描画結果は図 9.5 のようになります。

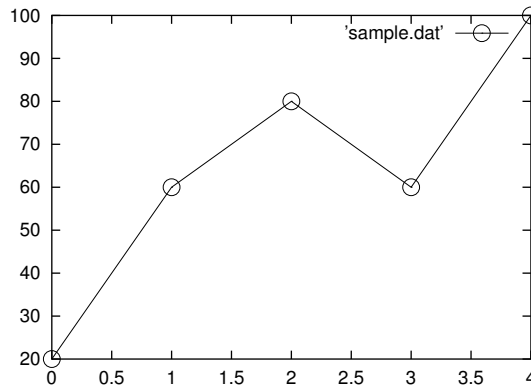


図 9.5: データファイル sample.dat のプロット (4)

データ点に点を打ち、あいだに線を引かない場合は、

```
plot 'sample.dat' with points [Enter]
```

です。これがデフォルトの挙動です。描画におけるオプション指定については、第 6 章および第 7 でも説明しましたが、データファイル特有のオプションもあるので、第 9.1.7.1 節でも改めて議論します。

プロットが終わったグラフを eps ファイルの形式で保存するには、

```
set terminal postscript eps [Enter]
set output 'ファイル' [Enter]
replot [Enter]
```

とします。ここに、「ファイル」と書かれた部分には適当なファイル名が入ります。

これ以降の例では、描画結果を見やすくするため、座標軸の目盛り (xtics, ytics) や線の説明 (key) などを必要に応じて省略します。また、データ点のスタイルや大きさも必要があれば調整します。ですから、みなさんが例と同じコマンドを打ち込んでも、描画結果がここに記載された例と若干異なることがあります。これらの調整の仕方については、第 6 章および第 7 を参照して下さい。

9.1.2 データファイルの書式

データファイルは以下に述べるような書式および規則にしたがって作成され、gnuplot によって解釈されます。

- 1 行中に複数の数値が空白で区切って並べられる。
- 空白は英語のスペースあるいはタブ文字を 1 個以上 (個数はいくつでもよい) 並べたもので構成される。
- 空白に日本語のスペースを使うことはできない。
- 数値のフォーマットとしては、ふつうの整数および小数の表記と指数形式が使える。
- 指数形式でデータを表記するときには、たとえば 2.998E+8 あるいは 2.24e-2 などのように、

仮数部、文字 E あるいは e、10 のべきの桁数

という順番で数値および文字を並べる． 文字 E あるいは e は指数部の始まりを示すために用いられる． 文字 E と文字 e の意味は同一である． なお、先に示した第 1 の例は 2.998×10^8 という意味になり、第 2 の例は 2.24×10^{-2} という意味になる． 指数部については、たとえば 2.998E+8 と書くかわりに 2.998E+08 や 2.998E+008 などのように冒頭に数字 0 を追加した書き方をしても、意味は変わらない．

- 標準的な書式は 1 行にグラフの x 座標と y 座標を空白で区切って並べるというものである．
- x 座標を省略して 1 行に数値を 1 個ずつ書いていった場合、gnuplot はこのデータファイルの x 座標を 0 から始まる整数であると解釈して実行する．
- 1 行中にデータ点を 3 個以上ならべることができる．
- 1 行中にたくさんデータ点がある場合、gnuplot を実行中にデータのどの列を使ってグラフを 描画するかを指示することができる．
- gnuplot が作成するグラフの中には 1 行中に含まれるデータ点の数を 3 個あるいは 4 個要求するものもある．
- 文字#以降はコメントとみなされ、無視される．
- データが記載された行のあいだに空白行（改行文字のみからなる行）が 1 個ある場合、gnuplot はこのデータファイルの該当部分をデータの不連続点と解釈する． データ点のあいだを線で結ぶ場合にも、不連続点の部分に線が引かれることはない．
- データが記載された行のあいだに空白行が 2 個以上あった場合（2 個以上ならいくつでもよい）、gnuplot は、データファイルには複数のグラフをプロットするためのデータが含まれていて、2 個以上の空白行の部分から新しいグラフのデータが始まっているものとみなす．
- 同一のデータファイルに複数のグラフのデータを含ませる場合、1 個のファイルに含めることができるグラフの数には制限はない．

9.1.3 データファイルの例

いくつかデータファイルの例を示しておきます．

図 9.6 に、1 行あたりに含まれるデータ点が 1 個のデータファイルと、それをプロットした結果を示します．ただし、データ点を記号「○」であらわし、かつデータ点のあいだを直線で結んでいます．

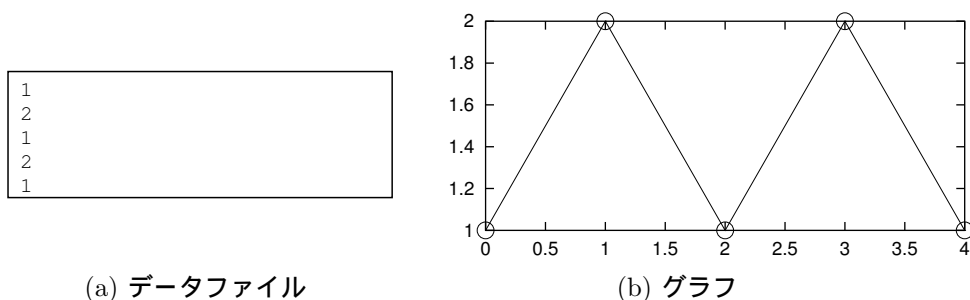


図 9.6: 1 行あたりのデータ点が 1 個のデータファイルとそのグラフ

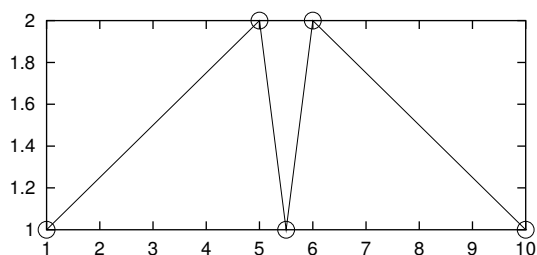
図 9.7 に、1 行あたりに含まれるデータ点が 2 個含まれる場合は、1 番目の数値が x 座標、2 番目の数値が y 座標と解釈されます． 図 9.7 に、そのようなデータファイルとそれをプロットした結果を示します． 先の例と同様に、データ点を記号「○」であらわし、かつデータ点のあいだを直線で結んでいます．

```

1 1
5 2
5.5 1
6 2
10 1

```

(a) データファイル



(b) グラフ

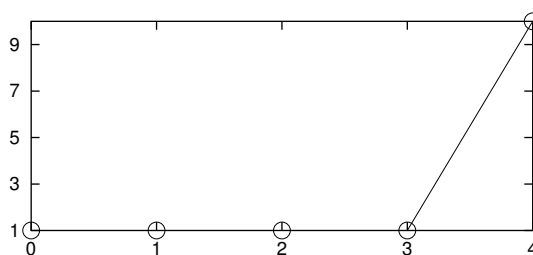
図 9.7: 1 行あたりのデータ点が 2 個のデータファイルとそのグラフ

```

1.0E+0.0
1.0E+0.3
1.0E+0.6
1.0E+0.9
1.0E+1.2

```

(a) データファイル



(b) グラフ

図 9.8: 指数形式で表現されたデータファイルとそのグラフ

また、図 9.8 に、データ点が指数形式で標記されたデータファイルとそれをプロットしたグラフを示します。先と同様に、データ点を記号「 \circ 」であらわし、かつデータ点のあいだを直線で結んでいます。

なお、先に述べたように、べきの部分であらわすときには、英文字 e の大文字、小文字の双方が使えます。また、 $E-03$ のような表記において、符号の直後の 0 は省略することができます。

9.1.3.1 不連続点を含むデータファイル

先に述べたように、データが記載された行のあいだに単一の空白行があると、その部分はグラフにおける不連続点であると解釈されます。第 9.1.3.2 節で述べるように、データファイルに空白行が 2 個あると、gnuplot はこれを不連続点ではなく異なるデータだと解釈します。単一の空白行と 2 個の空白行で意味がまったく異なるので注意して下さい。

図 9.9 に、2 個の不連続点を含むデータファイルと、そのグラフを線付きでプロットした結果を示します。先と同様に、データ点を記号「 \circ 」であらわし、かつデータ点のあいだを直線で結んでいます。図 9.9 から、データに空白行がある部分で補間直線が途切れていることがわかります。

9.1.3.2 複数のグラフのデータを含むデータファイル

先に述べたように、データが記載された行のあいだ 2 個以上の連続した空白行があると、データファイルには複数のグラフをプロットするためのデータが含まれていて、2 個以上の空白行の部分から新しいグラフのデータが始まっているものと解釈されます。

図 9.10 に、2 個の不連続点を含むデータファイルと、そのグラフを線付きでプロットした結果を示します。先と同様に、データ点を記号「 \circ 」であらわし、かつデータ点のあいだを直線で結んでいます。図 9.10 において、グラフが 3 本表示されていることに注意して下さい。図 9.10(a) では、4 行目および 5 行目と 9 行目および 10

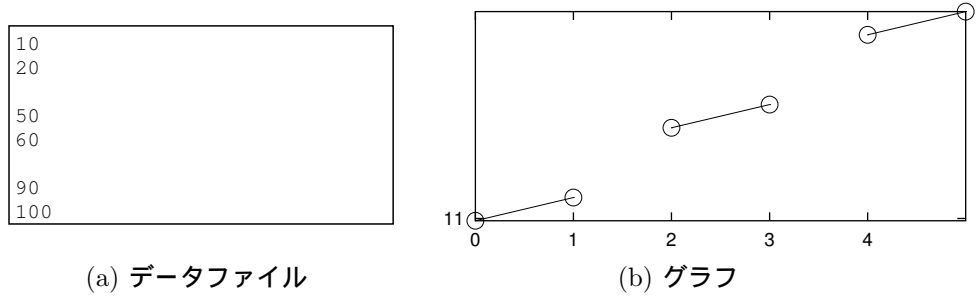


図 9.9: 不連続点を含むデータファイルとそのグラフ

行目それぞれに 2 行分の空白行があります。 ですから, `gnuplot` は, このデータファイルには 3 個のグラフを描くためのデータが含まれていると解釈します。

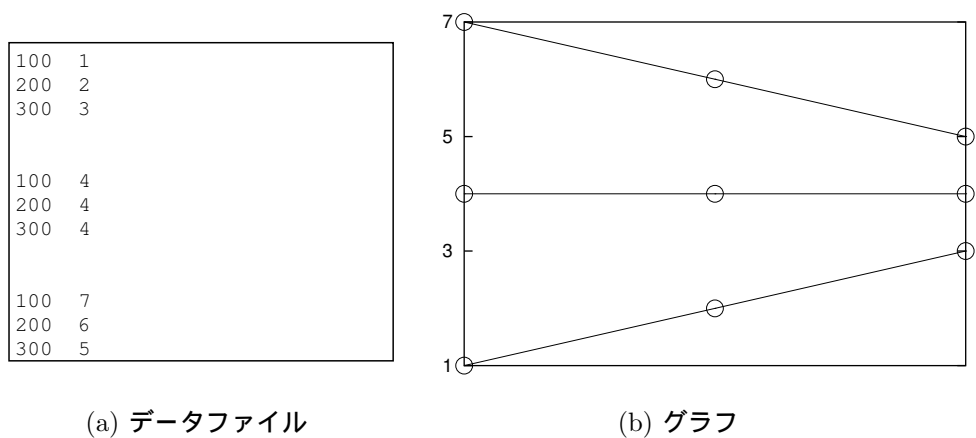


図 9.10: 3 個のグラフに対応したデータが含まれたデータファイルとそのグラフ

9.1.4 複数のグラフが含まれるデータファイルからグラフを抜き出す

複数のグラフが含まれるデータファイルからグラフを抜き出すには, コマンド `plot` に `index` というキーワードを付けます。 キーワード `index` の使い方は,

```
plot 'ファイル' index グラフの番号 [Enter]
```

です。 ここに, 「グラフの番号」と書かれた部分には 0 以上の整数を指定します。

データファイルに複数のグラフのためのデータが含まれている場合, 最初のグラフのためのデータには 0, 2 番目のグラフのためのデータには 1 などといった番号が付きます (番号が 0 から始まることに注意)。 このグラフの番号を指定することで, 特定のグラフだけを抜き出すことができます。

先に図 9.10(a) で挙げたファイル (ファイル名を `multi-graphs.dat` とします) から, 最初のグラフだけ抜き出してプロットしてみましょう。 ただし, グラフを見やすくするために, データ点を記号「`o`」であらわし, かつデータ点のあいだを直線で結ぶことにします。

```
plot 'multi-graphs.dat' index 0 with lines [Enter]
```

と入力すると, 図 9.11 のように, 図 9.10 のグラフの中から最初の 1 個を抜き出したグラフを描画することができます。

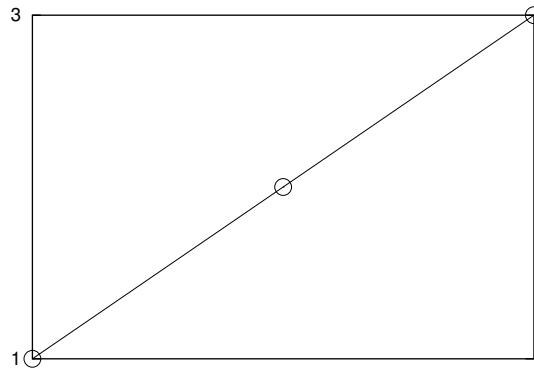


図 9.11: 複数のグラフを含むデータファイルから 1 個のグラフを抜き出してプロットしたもの

データファイルから 2 個以上のグラフを抜き出して、それぞれのスタイルを変えるということもできます。たとえば、

```
plot 'multi-graphs.dat' index 0 with linespoints pt 6 ps 2\ [Enter]
'multi-graphs.dat' index 2 with linespoints pt 4 ps 2[Enter]
```

とすると、図 9.12 のようなグラフが得られます。ただし、上のコマンドにおいて、pt は pointtype の省略形です。pointtype は点の種類を指定するキーワードでした（第 7.1.4.7 節の図 7.16 を参照）。なお、記号 \ はコマンド中に改行を入れなければならないときに使われます。

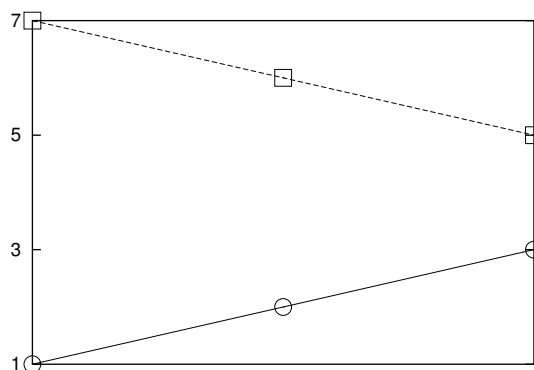


図 9.12: 複数のグラフを含むデータファイルから 2 個のグラフを抜き出してプロットしたもの

実は、キーワード index そのものにも、複数のグラフを選択するための、2 通りの使い方があります。そのうちのひとつめは、

```
index m: n
```

というものです。この場合、第 m 番目のグラフから第 n 番目のデータまでが描画されます。もちろん、実際には、 m および n と書かれた部分に適当な自然数を指定します。ふたつめは、そのうちのひとつめは、

```
index m: n: p
```

というものです。この場合、第 $m + kp$ 番目のグラフ（ただし $k \geq 0$, $m + kp \leq n$ ）がすべて重ね書きされます。たとえば、


```
plot 'multi-graphs.dat' index 1:2 with linespoints pt 6 [Enter]
```

と入力すると、図 9.13 のようなグラフが得られます。範囲指定が 1:2 となっているので、1 番目と 2 番目のグ

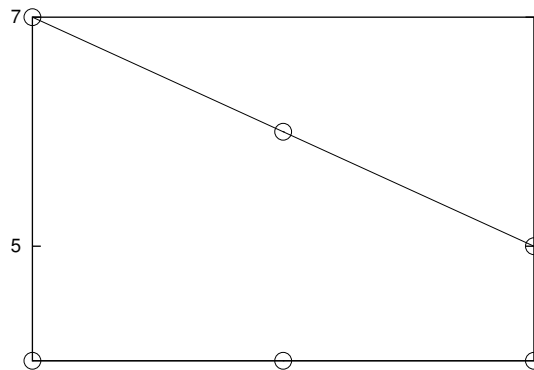


図 9.13: 複数のグラフを含むデータファイルから 2 番目と 3 番目のグラフを抜き出している例

ラフが選択されています。また、

```
plot 'multi-graphs.dat' index 0:2:2 with linespoints pt 6 [Enter]
```

と入力すると、図 9.14 のようなグラフが得られます。範囲指定が 0:2:2 となっているので、抜き出されるグラ

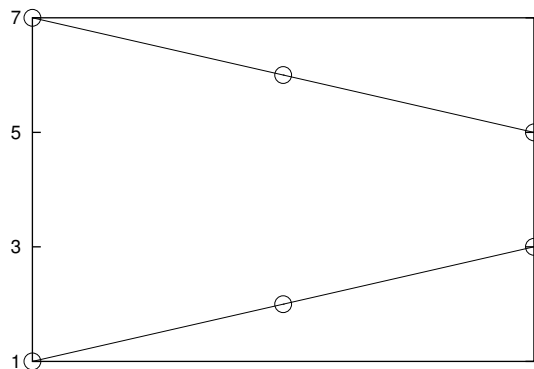


図 9.14: 複数のグラフを含むデータファイルから、0 番目以降のグラフを 1 個飛ばして抜き出している例

フの番号は、第 0 番、第 2 番、... となるのですが、第 2 番が選択されたときにデータの番号が 2 になって終了条件が満たされるので、ここでプロットは終わります。

9.1.5 データファイルのどの列を使ってプロットするかを変更する

この節ではデータファイルのどの列を使ってプロットするかを変更する方法について説明します。

説明のために、図 9.15 のようなファイルを考えます。このファイルには、

```
multi-data.dat
```

という名前がついているものとします。このデータの第 1 列、第 2 列、第 3 列の値はそれぞれ x , $\sin x$, $\cos x$ になっています。

さて、図 9.15 に示されたファイル `multi-data.dat` を使い、

0.0000	0.0000	1.0000
0.3140	0.3089	0.9511
0.6280	0.5875	0.8092
0.9420	0.8087	0.5882
1.2560	0.9509	0.3096
1.5700	1.0000	0.0008
1.8840	0.9514	-0.3081
2.1980	0.8097	-0.5869
2.5120	0.5888	-0.8083
2.8260	0.3104	-0.9506
3.1400	0.0016	-1.0000
3.4540	-0.3074	-0.9516
3.7680	-0.5862	-0.8101
4.0820	-0.8078	-0.5895
4.3960	-0.9504	-0.3111
4.7100	-1.0000	-0.0024
5.0240	-0.9518	0.3066
5.3380	-0.8106	0.5856
5.6520	-0.5901	0.8073
5.9660	-0.3119	0.9501

図 9.15: ファイル multi-data.dat の内容

```
plot 'multi-data.dat' with lines [Enter]
```

というコマンドによりグラフをプロットすると、図9.16のようなグラフが得られます。 ファイル multi-data.dat

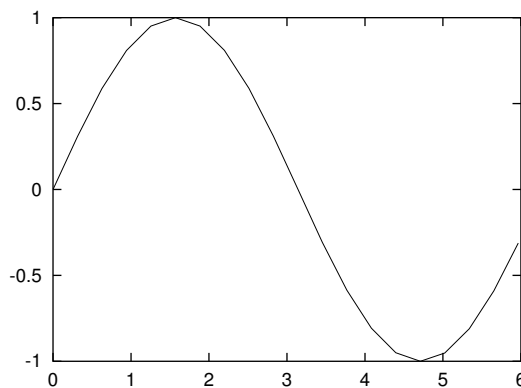


図 9.16: ファイル multi-data.dat を単純にプロットした結果

の第 1 列を横軸，第 2 列を縦軸に取ったグラフがプロットされているわけです。

では，第 1 列と第 3 列を使って， $\cos x$ のグラフがプロットできるようにするにはどうしたらよいでしょうか？

このためには，コマンド plot に using というキーワードを付けて実行します。

キーワード using がついたコマンド plot の基本的な構文は，

```
plot 'ファイル' using 横軸に使う列の番号:縦軸に使う列の番号 [Enter]
```

です。ここに，「横軸に使う列の番号」と書かれた部分と「縦軸に使う列の番号」と書かれた部分には，それぞれデータファイル中で描画に使いたい列の番号を書きます。ただし，データの最初の列が第 1 列になります。

例題として，ファイル multi-data.dat の第 1 列と第 3 列を使ってグラフを描いてみましょう。ただし，グラフを見やすくするために，コマンド plot に with lines というキーワードを付け，データ点のあいだを線で結んでおくことにします。

```
plot 'multi-data.dat' using 1:3 with lines [Enter]
```

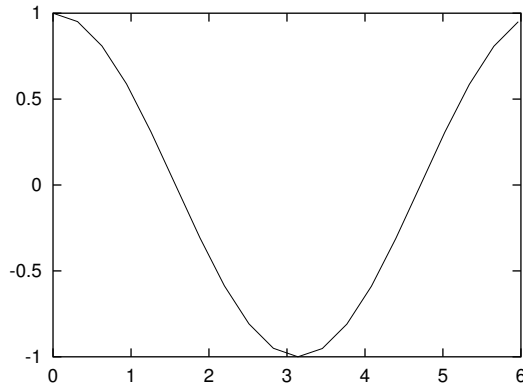


図 9.17: ファイル multi-data.dat の第 1 列と第 3 列を使ってプロットした結果

というコマンドを実行すると、図 9.17 に示すように、希望通り $\cos x$ のグラフがプロットされます。

もうひとつの例題として、横軸にファイル multi-data.dat の第 2 列、縦軸に第 3 列を取ったグラフを描いてみましょう。ただし、この場合は、横軸が $\sin x$ で縦軸が $\cos x$ なので、描画結果が円になることがわかります。ですから、あらかじめグラフの縦と横を一定にするコマンド `set size ratio -1` を実行しておくことにします。

```
set size ratio -1 [Enter]
plot 'multi-data.dat' using 2:3 with lines [Enter]
```

というコマンドを実行すると、図 9.18 に示すように、円が描画されます。ただし、サンプルの数が少ないのと、データファイルをプロットするときには標準では標本点のあいだは直線で結ばれるので、描画結果はやや角張ってしまっています。

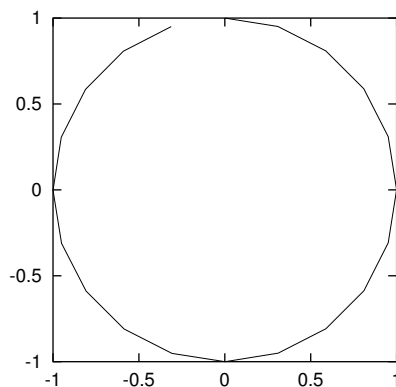


図 9.18: ファイル multi-data.dat の第 2 列と第 3 列を使ってプロットした結果

実は、キーワード `using` にはもうすこし高級な使い方があります。これについて説明するためには、`gnuplot` が読み込んだデータをどのようにして解釈するかについてもう少し詳しく見ておく必要があります。

キーワード

```
using m:n
```

を使うときには、 m および n にはデータの列の番号が入る（最左端が 1 列目と数える）のですが、この部分に数 0 を指定することもできます。列の番号として 0 が指定された場合、`gnuplot` は、第 0 列目として $0, 1, 2, 3, \dots$ というふうにつづく整数を自動的に補います。

さらに、第 m の実際に読み込まれたデータを、「 m 」という記号で参照することができます。そして、「 m 」という記号を関数の引数として使うことができます。ですから、

```
plot 'ファイル' using  $m$ :(関数($ $m$ )) [Enter]
```

とすると、横軸を指定されたデータファイルの第 m 列に、縦軸を指定されたデータファイルの第 n 列に関数を作作用させたものにとったグラフが描画されます。ここに、「ファイル」と書かれた部分には適当なファイルの名前が、「関数」と書かれた部分には gnuplot に用意されている数学関数かユーザが自分で定義した関数が入ります。「(関数(m))」のように全体を括弧「()」で囲う必要があることに注意して下さい。

いくつか例を見てみましょう。

まず、キーワード `using` で列 0 を指定した場合の例を見ます。

```
plot 'multi-data.dat' using 0:2 with lines [Enter]
```

というコマンドを実行すると、図 9.19 に示すようなグラフが得られます。このグラフの形は図 9.16 と同じです

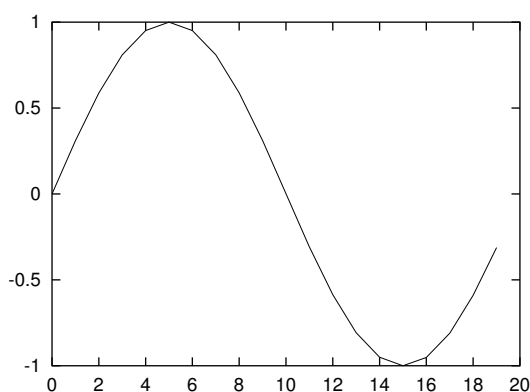


図 9.19: ファイル `multi-data.dat` の第 0 列と第 2 列を使ってプロットした結果

が、横軸の目盛りが違います。この違いがどこから来ているかというと、図 9.16 では横軸の数値としてファイル `multi-data.dat` (図 9.15) の第 1 列目が使われているのに対し、図 9.19 では横軸の数値として $0, 1, 2, \dots$ が使われていることから来ています。

次に、「 $\$n$ 」という記号を使った例を見ます。横軸をファイル `multi-data.dat` の第 1 列目に、横軸をファイル `multi-data.dat` の第 1 列目に関数 `sqrt` (平方根を取る関数) を作用させた結果にとったグラフを表示してみます。この場合、「第 1 列目に関数 `sqrt` を作用させた結果」は

```
(sqrt($1))
```

のように記述されます。「`sqrt($1)`」と書かれた部分を括弧で囲む必要があることをに注意して下さい。

では例を見てみましょう。

```
plot 'multi-data.dat' using 1:(sqrt($1)) with lines [Enter]
```

というコマンドを実行すると、図 9.20 をのようなグラフが得られます。この例の場合、グラフの横軸がファイル `multi-data.dat` の第 1 列に、縦軸がファイル `multi-data.dat` の第 1 列に関数 `sqrt` を作用させたものになっています。

9.1.6 データを間引く

データファイルからグラフを作るときに、適切なグラフを作るのに必要なデータ点の数と比較して実際のデータ点の数が極端に多いと、グラフを作成して保存したときにできるファイルが無駄に大きくなってしまいます。で

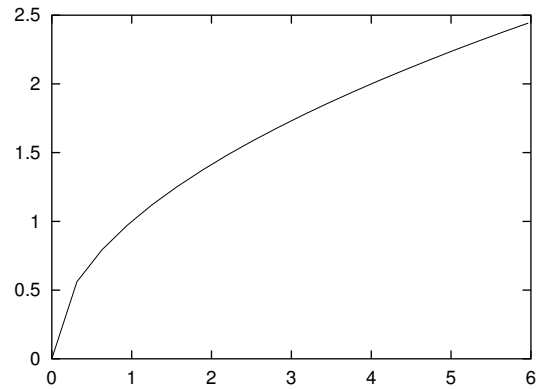


図 9.20: ファイル multi-data.dat の第 1 列を横軸に、それに関数 sqrt を作用させたものを縦軸に取ったグラフ

すから、このような場合には、データを適当に間引く、すなわち何個かに 1 個データ点を残して残りは捨ててしまってからグラフを作ったほうが好都合です。これとは別に、グラフが不連続点を含んでいるときに、グラフの連続な部分だけを取り出してプロットしたいという状況もあります。

このようにデータを適当に間引いてからプロットしたいときには、コマンド plot に every というキーワードを付けて実行します。

キーワード every の使い方はややこしいので、まず一番簡単な使い方から見ることにします。

再び図 9.15 のデータファイル multi-data.dat を考えます。このグラフで、データ点を 2 個ずつ飛ばしてプロットしてゆくにはどうしたらよいでしょうか？

```
plot 'multi-data.dat' every 3 with linespoints [Enter]
```

とします。なお、データが間引かれていることを明示するために、with linespoints というキーワードを使用しています。

上のようなコマンドを打ち込んだときに得られるグラフは図 9.21 のようになります。

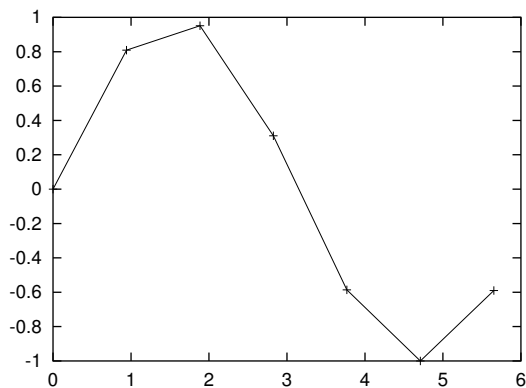


図 9.21: ファイル multi-data.dat のデータ点を 2 個ずつ飛ばしてプロットしたグラフ

図 9.16 と図 9.21 を比較すると、図 9.21 ではデータ点が間引かれている分だけグラフがぎざぎざになっていることがわかるでしょう。

データ点を間引くときの一般的な記法は、

```
plot 'multi-data.dat' every n [Enter]
```

です。ここに、「 n 」の部分には適当な正の整数が入ります。上記のように指定されると、gnuplot はデータファイルのデータ点を 1 個使うたびに、それに続いた $n-1$ 個のデータ点を読み飛ばしてゆきます。

キーワード `every` のより詳しい使い方を説明する前に、gnuplot がデータファイルの各行を解釈する仕方についてもう少し詳しく理解しておく必要があります。

例として、図 9.22(a) のような、いくつかの空白行を含むデータファイルを考えます。ファイル名を `block-data.dat` としておきましょう。

このファイルを、gnuplot は次のように解釈します。

- 単一の空白行を境に、データファイル全体は「ブロック」と呼ばれる単位に分割される
- 各ブロックには上から順番に 0 から始まる整数の番号が付く
- ブロック中のデータ行には、そのブロックの最初の行から順番に 0 から始まる整数の番号が付く

なお、データファイルに空白行がない場合には、データは 1 個のブロックから成るものとみなされ、最初の行から順番に 0, 1, 2, ... という番号が振られてゆきます。

ファイル `block-data.dat` が実際にどのように解釈されるかについては、図 9.22(b) を参照して下さい。

ブロック番号 行番号		
ブロック0	データ0	→10 100
	データ1	→11 110
	データ2	→12 120
	データ3	→13 130
ブロック1	データ0	→14 200
	データ1	→15 210
	データ2	→16 220
	データ3	→17 230
ブロック2	データ0	→18 300
	データ1	→19 310
	データ2	→20 320
	データ3	→21 330
ブロック3	データ0	→22 400
	データ1	→23 410
	データ2	→24 420
	データ3	→25 430

(a) ファイル `block-data.dat`

(b) gnuplot による解釈

図 9.22: ファイル `block-data.doc` とその gnuplot による解釈

さて、図 9.22 をふまえた上で、キーワード `every` のより詳しい説明に移りましょう。

キーワード `every` の完全な使い方は、

`every 行刻み:ブロック刻み:初期行:初期ブロック:終了行:終了ブロック`

のようになります。上記において、「行刻み」、「ブロック刻み」、「初期点」、「初期ブロック」、「終点」および「終了ブロック」と書かれた部分には、それぞれ 0 以上の整数が入ります。これらの意味について表 9.1 にまとめておきます。

これらのオプションはすべて省略可能です。ただし、省略の仕方は少し変わっていて、

- 指定する最後のオプション以降には記号「:」も含めて何も書かない

表 9.1: キーワード every におけるオプション指定

	意味
行刻み	データ行を何行ごとに読み出すか指定, 何も指定しなければ 1 になる (すべてのデータ行を読む)
ブロック刻み	データのブロックを何ブロックごとに読み出すか指定, 何も指定しなければ 1 になる (すべてのブロックを読む)
初期行	データ行をどの行から読み始めるかの指定, 何も指定しなければ 0 になる (最初の行から読む)
初期ブロック	データブロックをどのブロックから読み始めるかの指定, 何も指定しなければ 0 になる (最初のブロックから読む)
終了行	データ行をどの行で読み終わるかの指定, 何も指定しなければデータファイルの最後の行まで読む
終了ブロック	データブロックをどのブロックで読み終わるかの指定, 何も指定しなければデータファイルの最後の行まで読む

- 最後のオプション以前に省略するものがある場合には, オプションそのものは指定しないが, オプションに対応する記号「:」だけは残しておく

という規則にしたがいます.

これだけではわかりにくいと思われるので, 表 9.2 にいくつか例を示しておきます. オプションの最初あるい

表 9.2: オプションの省略のしかた

省略する項目	記法
完全なオプションの指定	行刻み:ブロック刻み:初期行:初期ブロック:終了行:終了ブロック
終了ブロックを省略	行刻み:ブロック刻み:初期行:初期ブロック:終了行
終了行と終了ブロックを省略	行刻み:ブロック刻み:初期行:初期ブロック
終了行のみを省略	行刻み:ブロック刻み:初期行:初期ブロック::終了ブロック
初期ブロックと終了行を省略	行刻み:ブロック刻み:初期行::終了ブロック
終了行以降を省略	行刻み:ブロック刻み:初期行:初期ブロック
終了行以降とブロック刻みを省略	行刻み::初期行:初期ブロック
終了行以降とブロック刻み, 初期行を省略	行刻み:::初期ブロック
行刻み以外をすべて省略	行刻み
終了ブロック以外をすべて省略	:::::終了ブロック

は途中で省略部分がある場合には記号「:」がいくつも続いていることと, オプションを指定している部分の最後に記号「:」が付くことはないということを確認して下さい.

では, 実際に図 9.22(a) に示されたデータファイル block-data.dat を使って, いくつかのグラフをプロットしてみましょう.

まず参考のために, どのデータ点も間引かれていないグラフをプロットしておきます.

```
plot 'block-data.dat' pt 6 ps 2 [Enter]
```

とすると, 図 9.23 のようなグラフが得られます.

続いて,

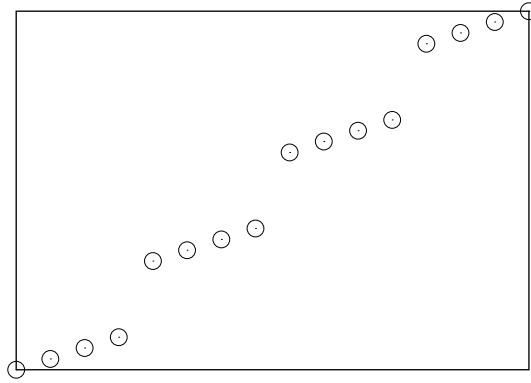


図 9.23: データファイル block-data.dat の省略のないプロット

```
plot 'block-data.dat' every :::1 pt 6 ps 2 [Enter]
```

としてみます。これは、「グラフを第 1 ブロックまでプロットする」という意味になります。得られるグラフは図 9.24 のようなものになります。

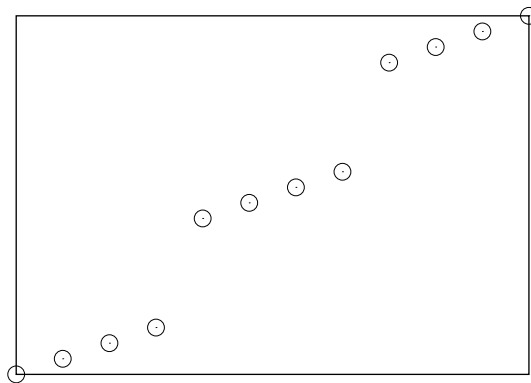


図 9.24: 終了ブロックを指定したデータファイル block-data.dat のプロット

さらに、

```
plot 'block-data.dat' every :::2:3 pt 6 ps 2 [Enter]
```

としてみます。これは、「第 3 ブロックまで、各ブロックの番号 2 の行までをプロットする」という意味になります。結果として、図 9.25 のようなグラフが得られます。

続いて、

```
plot 'block-data.dat' every 2:2 pt 6 ps 2 [Enter]
```

としてみます。これは、「2 個のブロックごとに 1 個のブロックを使う、各ブロック内でも 2 個のデータ行ごとに 1 個のデータ行を使う」、すなわち 1 ブロック読むごとに 1 ブロック読み飛ばし、各ブロック内部でも 1 行読むごとに 1 行読み飛ばすという意味に解釈されます。上記のコマンドを実行すると、図 9.26 のようなグラフが得られます。

途中のデータブロックから描画を始める例も見てください。

```
plot 'block-data.dat' every ::1:1:3:2 pt 6 ps 2 [Enter]
```

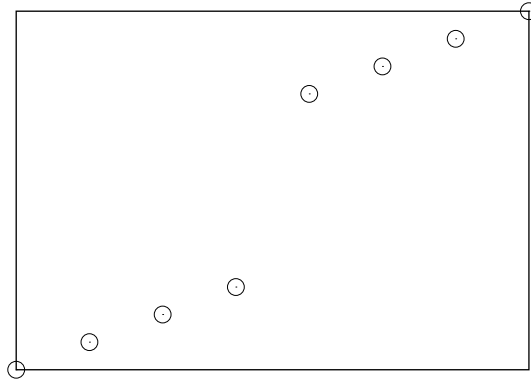



図 9.25: 終了行と終了ブロックを指定したデータファイル block-data.dat のプロット

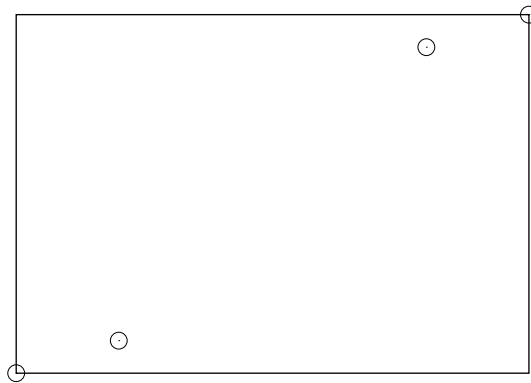


図 9.26: ブロックの読み飛ばしとブロック内の読み飛ばしを指定したデータファイル block-data.dat のプロット

とすると、「第 1 ブロック (2 番目のブロック: 番号は 0 から始まっていることに注意) から第 2 番目のデータブロックまで、番号 1 のデータ行から番号 3 のデータ行までをプロットする」という意味になります。これをプロットすると、図 9.27 のようなグラフが得られます。

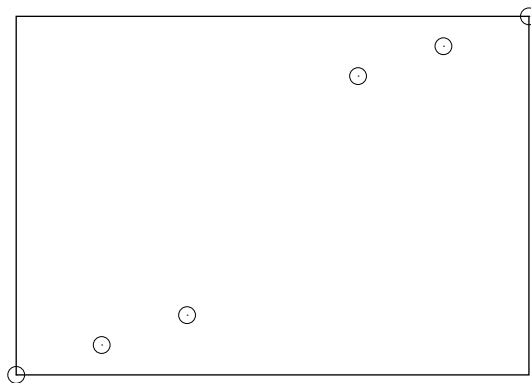


図 9.27: 途中から描画を始めたデータファイル block-data.dat のプロット

9.1.7 グラフのスタイルを変更する

グラフのスタイルを変える方法は、第 7.1.4 節とほとんど同じなのですが、第 7.1.4 節に述べられたもの以外にデータファイルに固有のスタイルがいくつかあります。

9.1.7.1 点や線のスタイルを変更する

第 1 次、第 7.1.4 節と共通のものについて表 9.3 にまとめておきます。例題については第 7.1.4 節を参照して下さい。なお、標準では「データ点を点で表示する」グラフの作成法が選択されています。標準のままが良い場合にはオプションを指定する必要はありません。

表 9.3: グラフのスタイルを変更するためのオプション

機能	オプションおよび説明
データ点を点で表示する	with points 標準
データ点のあいだを線で結ぶ	with lines
データ点を点で表示し、あいだを線で結ぶ	with linespoints
線の種類を変える	linetype 数
	省略形 lt, with lines か with linespoints の後に指定、linewidth の後には指定できない
線の太さを変える	linewidth 数
	省略形 lw, with lines か with linespoints の後に指定
点の種類を変える	pointtype 数
	省略形 pt, with points か with linespoints の後に指定、pointsize の後には指定できない
点の大きさを変える	pointsize 数
	省略形 ps, with points か with linespoints の後に指定
データをインパルスで表示する	with impulses
データを箱で表示する	with boxes
データを階段グラフで表示する	with steps あるいは with fsteps あるいは histeps

9.1.7.2 エラーバーを表示する

誤差が含まれるデータをプロットする場合、グラフにエラーバーを付けて表示することがあります。本節ではグラフにエラーバーを付ける方法について説明します。

9.1.7.2.1 縦軸のみに誤差を含むグラフ ここでは、まず横軸の数値には誤差がなく縦軸の数値にのみ誤差が含まれる場合を考えます。

縦軸のみに誤差を含むデータファイルの書式としては、図 9.28 の 2 種類のうちのいずれかが想定されています。そして、データのエラーバーを付ける場合には、コマンド plot に with errorbars というキーワードを使います。なお、with yerrorbars というキーワードを使っても同じことができます。ですから、典型的な縦軸のみにエラーバーを付ける方法は、

```
plot 'ファイル' with errorbars [Enter]
```

x_1	y_1	$\delta_{y,1}$	
x_2	y_2	$\delta_{y,2}$	
.....			

形式 1

x_1	y_1	y_1^{\min}	y_1^{\max}
x_2	y_2	y_2^{\min}	y_2^{\max}
.....			

形式 2

図 9.28: 縦軸にエラーバーを付けるグラフが前提としているデータファイルの書式

となります。ここに、「ファイル」と書かれた部分には適当なファイル名が入ります。

コマンド `plot` が `with errorbars` というオプション付きで起動された場合、gnuplot は、データファイルが図 9.28 の形式 1 の場合は点 (x_i, y_i) に点を打ってから、 $(x_i, y_i - \delta_{y,i})$ から $(x_i, y_i + \delta_{y,i})$ に伸びる直線を引き、さらに点 $(x_i, y_i - \delta_{y,i})$ および点 $(x_i, y_i + \delta_{y,i})$ を中心とする短い横方向の線分を引きます。

一方、コマンド `plot` が `with errorbars` というオプション付きで起動され、データファイルが図 9.28 の形式 2 だった場合には、gnuplot は点 (x_i, y_i) に点を打ってから、 (x_i, y_i^{\min}) から (x_i, y_i^{\max}) に伸びる直線を引き、さらに点 (x_i, y_i^{\min}) および点 (x_i, y_i^{\max}) を中心とする短い横方向の線分を引きます。

図 9.29(a) に示すような形式 1 のデータファイル（ファイル名を `eb1.dat` とします）を使い、

```
plot 'eb1.dat' with errorbars pt 6 ps 2[Enter]
```

とした結果を図 9.29(b) に、図 9.30(a) に示すような形式 2 のデータファイル（ファイル名を `eb2.dat` とします）

```
plot 'eb2.dat' with errorbars pt 6 ps 2[Enter]
```

を使い、とした結果を図 9.30(b) に示します。形式 1 と形式 2 の違いを明確にするために、図 9.30(a) のデータでは、エラーバーのデータ点より上の部分がデータ点より下の部分よりかなり長くなるように取ってあります。

9.1.7.2.2 横軸のみに誤差を含むグラフ 次に、縦軸の数値には誤差がなく横軸の数値にのみ誤差が含まれる場合を考えます。

横軸のみに誤差を含むデータファイルの書式としては、図 9.31 の 2 種類のうちのいずれかが想定されています。

そして、データのエラーバーを付ける場合には、コマンド `plot` に `with xerrorbars` というキーワードを使います。ですから、典型的な横軸のみにエラーバーを付ける方法は、

```
plot 'ファイル' with xerrorbars [Enter]
```

となります。ここに、「ファイル」と書かれた部分には適当なファイル名が入ります。

コマンド `plot` が `with xerrorbars` というオプション付きで起動された場合、gnuplot は、データファイルが図 9.28 の形式 1 の場合は点 (x_i, y_i) に点を打ってから、 $(x_i - \delta_{x,i}, y_i)$ から $(x_i + \delta_{x,i}, y_i)$ に伸びる直線を引き、さらに点 $(x_i - \delta_{x,i}, y_i)$ および点 $(x_i + \delta_{x,i}, y_i)$ を中心とする短い縦方向の線分を引きます。

一方、コマンド `plot` が `with errorbars` というオプション付きで起動され、データファイルが図 9.28 の形式 2 だった場合には、gnuplot は点 (x_i, y_i) に点を打ってから、 (x_i^{\min}, y_i) から (x_i^{\max}, y_i) に伸びる直線を引き、さらに点 (x_i^{\min}, y_i) および点 (x_i^{\max}, y_i) を中心とする短い縦方向線分を引きます。

9.1.7.2.3 横軸と縦軸誤差を含むグラフ 横軸と縦軸に縦軸のみに誤差を含むデータファイルの書式としては、図 9.31 の 2 種類のうちのいずれかが想定されています。

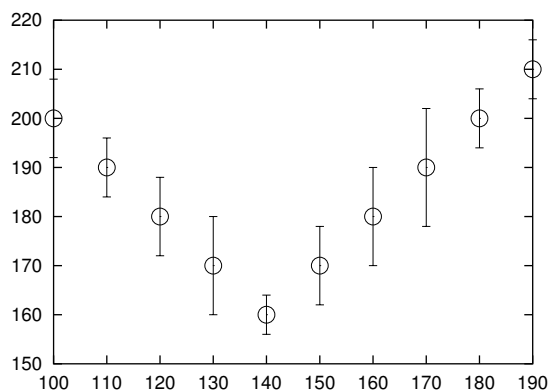
横軸は形式 1、縦軸は形式 2、というふうになんらかの形式を混ぜて使うことはできません。

グラフの横および縦方向のエラーバーを付ける場合には、コマンド `plot` に `with xyerrorbars` というキーワードを付けます。ですから、典型的な横軸のみにエラーバーを付ける方法は、

```
plot 'ファイル' with xyerrorbars [Enter]
```

100	200	8
110	190	6
120	180	8
130	170	10
140	160	4
150	170	8
160	180	10
170	190	12
180	200	6
190	210	6

(a) データファイル

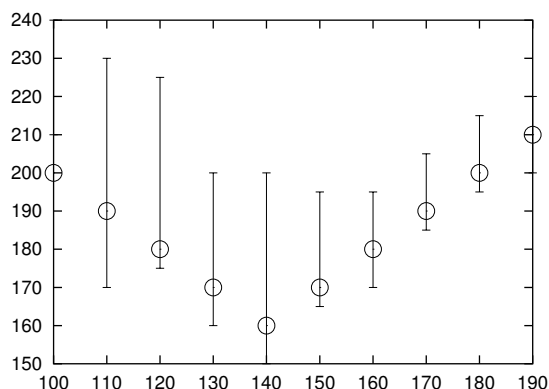


(b) グラフ

図 9.29: 誤差を含む形式 1 のデータファイルとそのグラフ

100	200	210	240
110	190	170	230
120	180	175	225
130	170	160	200
140	160	150	200
150	170	165	195
160	180	170	195
170	190	185	205
180	200	195	215
190	210	200	220

(a) データファイル



(b) グラフ

図 9.30: 誤差を含む形式 2 のデータファイルとそのグラフ

x_1	y_1	$\delta_{x,1}$
x_2	y_2	$\delta_{x,2}$
.....		

形式 1

x_1	y_1	x_1^{\min}	x_1^{\max}
x_2	y_2	x_2^{\min}	x_2^{\max}
.....			

形式 2

図 9.31: 横軸にエラーバーを付けるグラフが前提としているデータファイルの書式

x_1	y_1	$\delta_{x,1}$	$\delta_{y,1}$
x_2	y_2	$\delta_{x,2}$	$\delta_{y,2}$
.....			

形式 1

x_1	y_1	x_1^{\min}	x_1^{\max}	y_1^{\min}	y_1^{\max}
x_2	y_2	x_2^{\min}	x_2^{\max}	y_2^{\min}	y_2^{\max}
.....					

形式 2

図 9.32: 横軸と縦軸にエラーバーを付けるグラフが前提としているデータファイルの書式

となります。ここに、「ファイル」と書かれた部分には適当なファイル名が入ります。

エラーバーがどのように描かれるかは、横軸のみ、あるいは縦軸のみにエラーバーを付ける場合と同じです。

9.1.7.2.4 誤差の表記が混在したデータファイルからグラフを作りたいとき いろいろな誤差表記が混在したデータファイルからエラーバー付きのグラフを付けるには、9.1.5 で述べたキーワード `using` を使います。実

は、キーワード `using` の後に指定できる列の種類は 2 種類に限られるわけではなく、

```
using m:n:p:q
```

などのように何種類でも列を指定できます。そして、上の例の場合は、「第 m 列と第 n 列と第 p 列と第 q 列」というふうに解釈されます。ですから、いろいろな誤差表記が混在したデータファイルからエラーバー付きのグラフを作成するときには、キーワード `using` を使ってグラフから必要な部分を抜き出す作業が必要になります。

9.1.8 データ点のあいだを結ぶ線のなめらかさを変える

データファイルをプロットするときには、`gnuplot` は標準ではデータ点のあいだを直線で結びます。

この部分を変更して、データ点のあいだを曲線で結ぶようにするには、キーワード `smooth` に続いてオプションを指定します。指定できるオプションは `unique`, `csplines`, `acsplines`, `bezier`, `sbezier` の 5 種類です。このそれぞれの意味について表 9.4 にまとめておきます。複数のデータ点が同一の x 座標を持つことがない場合に

表 9.4: キーワード `smooth` のオプション

オプション	意味
<code>unique</code>	まずデータファイルを横軸の座標に関してソートする。 x 座標が同一で y 座標が異なる複数のデータ点 $(x_i, y_{i1}), \dots, (x_i, y_{im})$ がある場合は、これらの y 座標を平均したデータ点 $\left(x_i, \frac{y_{i1} + \dots + y_{im}}{m}\right)$ をグラフにプロットし、もともとのデータ点は捨てる。最後に、プロットされたデータ点のあいだを折れ線で結ぶ。
<code>csplines</code>	まず <code>unique</code> と同一の処理を実行してからデータを近似する 3 次の自然スプラインによる補間をおこなう。
<code>acsplines</code>	まず <code>unique</code> と同一の処理を実行してから重み付きの自然スプライン補間をおこなう。重みはデータファイルの第 3 列で指定される。
<code>bezier</code>	データを近似する、データ点の数と同じ次数のベジエ曲線を引く。
<code>sbezier</code>	まず <code>unique</code> と同一の処理を実行してからデータを近似するベジエ曲線を引く。

は、上記のうち `unique` と `csplines` を使った場合にはデータ点をすべて通る曲線が引かれますが、`acsplines`, `bezier`, `sbezier` を使った場合には曲線は場合によってはデータ点を避けます。

これらの例を以下で見てゆくことにします。ただし、`acsplines` は必要とするデータファイルの形式が他と異なるので、この紹介を最後に回します。

9.1.8.1 準備

図 9.33(a) のようなデータファイルを考えます。ファイル名を `for-smooth.dat` としておきましょう。

```
plot 'for-smooth.dat' with points pt 6 ps 2 [Enter]
```

とした結果を図 9.33(b) に示します。このデータファイルでは、同一の x 座標データ点が 3 個ずつあることがわかります。

まず最初に、

```
plot 'for-smooth.dat' with linespoints, 'for-smooth.dat' with points pt 6 ps 2 [Enter]
```

として、単純にデータ点のあいだを線で結んでみます。ただし、上の例で、

```
'for-smooth.dat' with points pt 6 ps 2
```

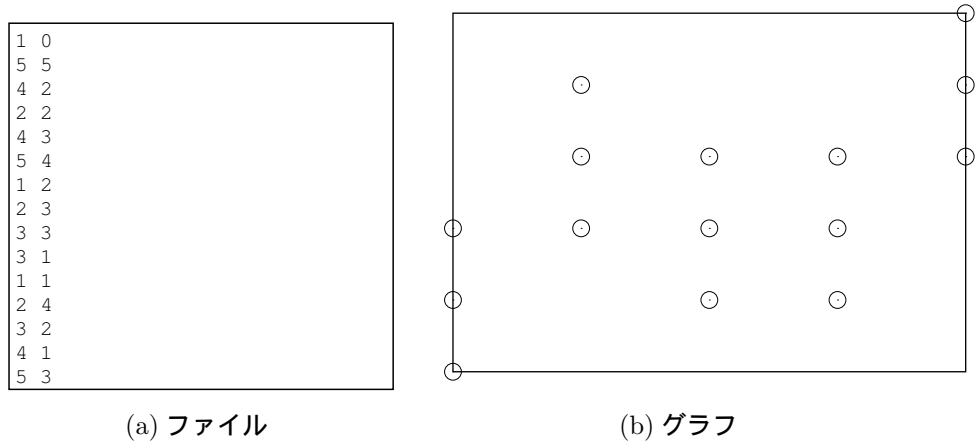


図 9.33: x 座標が小さい順に並んでいないデータファイルとそのグラフ

となっている部分は、図を見やすくするためにグラフにデータ点そのものを重ね書きしている部分であり、本来は必要ではありません。以下の例では、データ点と曲線の関係を見やすくするために、グラフにデータ点をつねに重ね書きしておくことにします。

さて、上のようなコマンドを実行した結果は結果は図 9.34 のようになります。データが x 座標の小さい順に

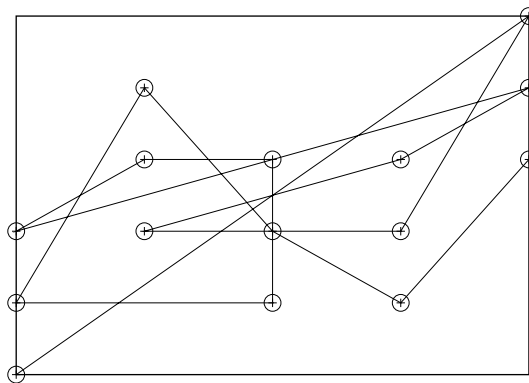


図 9.34: データファイル for-smooth.dat を単純に線で結んだもの

並べられていないことと同じ x 座標を持つ点が複数あることから、滅茶苦茶な線が引かれています。

9.1.8.2 キーワード unique を指定したグラフ

次に、キーワード unique を指定してみます。

```
plot 'for-smooth.dat' smooth unique, 'for-smooth.dat' with points pt 6 ps 2 [Enter]
```

と入力すると、結果は図 9.35 のようになります。図 9.35 から、グラフの縦軸の平均値を通る折れ線が引かれていることがわかります。

9.1.8.3 キーワード csplines を指定したグラフ

続いて、キーワード csplines を指定してみます。

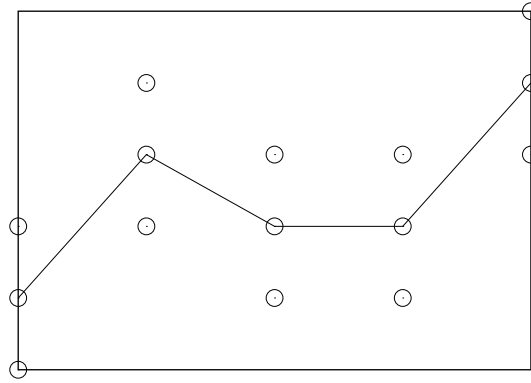


図 9.35: キーワード unique を適用した結果

```
plot 'for-smooth.dat' smooth csplines, 'for-smooth.dat' with points pt 6 ps 2 [Enter]
```

と入力しましょう。結果は図 9.36 のようになります。図 9.35 の折れ線が図 9.36 ではなめらかな曲線で置き

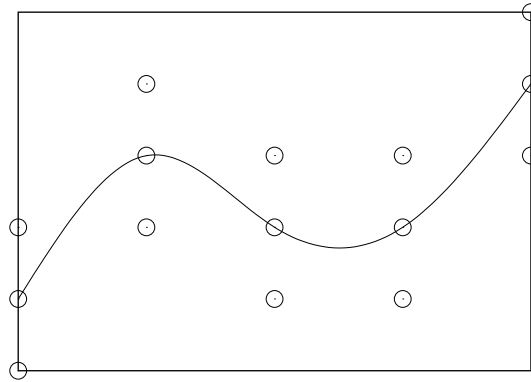


図 9.36: キーワード csplines を適用した結果

換えられています。

9.1.8.4 キーワード bezier を指定したグラフ

さらに、キーワード bezier を指定してみます。

```
plot 'for-smooth.dat' smooth bezier, 'for-smooth.dat' with points pt 6 ps 2 [Enter]
```

と入力しましょう。結果は図 9.37 のようになります。キーワード bezier を指定した場合には、x 軸の値に関して整理されていないデータ点のあいだを単純に線で結んでいった場合と同様、ねじれた曲線が描かれることがあります。図 9.37 もそのようになっています。

9.1.8.5 キーワード sbezier を指定したグラフ

次はキーワード sbezier を指定してみます。

```
plot 'for-smooth.dat' smooth sbezier, 'for-smooth.dat' with points pt 6 ps 2 \rt
```

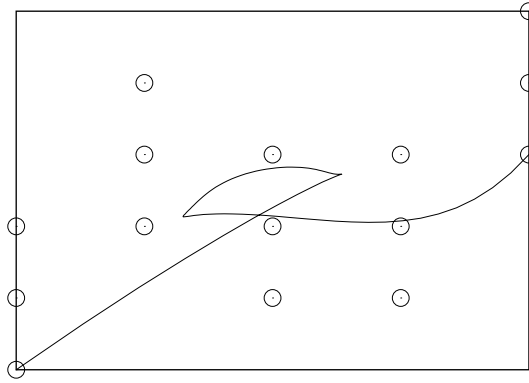


図 9.37: キーワード bezier を適用した結果

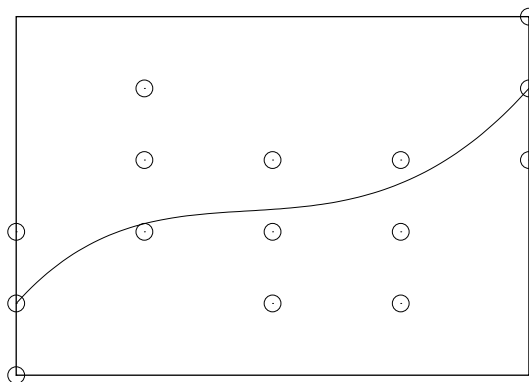


図 9.38: キーワード sbezier を適用した結果

と入力しましょう。結果は図 9.38 のようになります。キーワード sbezier を指定した場合には、まず unique と同じ処理がおこなわれるため、曲線がねじれることはありません。先に述べた通り、csplines と sbezier の違いは、csplines は計算された点をすべて通過する曲線を引くのに対し、sbezier はそれらの点を適当に避けた点を引くことです。

9.1.8.6 キーワード acsplines を指定したグラフ

さいごに、キーワード acsplines を使った例を示します。acsplines を使う場合には、データファイルには「重み」という追加情報が必要になります。acsplines を使った場合にも引かれる曲線はデータ点を適当に避けるのですが、重みの数値が大きいところほど曲線がデータ点に近付き、数値が小さいところほど曲線はデータ点から離れます。

キーワード acsplines の使い方は unique その他と同じで、

```
plot 'ファイル' smooth acsplines [Enter]
```

です。

例として、図 9.39(a) のようなデータファイル（ファイル名を ac1.dat とします）をコマンド

```
plot 'ac1.dat' smooth acsplines, 'ac1.dat' pt 6 ps 2 [Enter]
```


によってプロットしたものの図 9.39(b) に、図 9.39(b) のようなデータファイル（ファイル名を ac2.dat とします）をコマンド

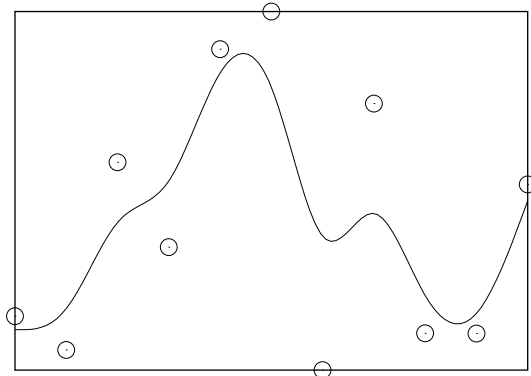
```
plot 'ac2.dat' smooth acsplines, 'ac2.dat' pt 6 ps 2 [Enter]
```

プロットしたものの図 9.40(b) に示します。

ファイル ac1.dat と ac2.dat の違いは、ac2.dat の方は重みを指定している第 3 列の数値がデータの後半で大きくなっていることです。結果として、ac2.dat をプロットした図 9.40(b) では、図の右側の方で曲線がデータ点に近付いています。

100	0.4127	0.01
110	0.3610	0.01
120	0.6479	0.01
130	0.5184	0.01
140	0.8210	0.01
150	0.8784	0.01
160	0.3303	0.01
170	0.7377	0.01
180	0.3865	0.01
190	0.3862	0.01
200	0.6141	0.01

(a) ファイル ac1.dat

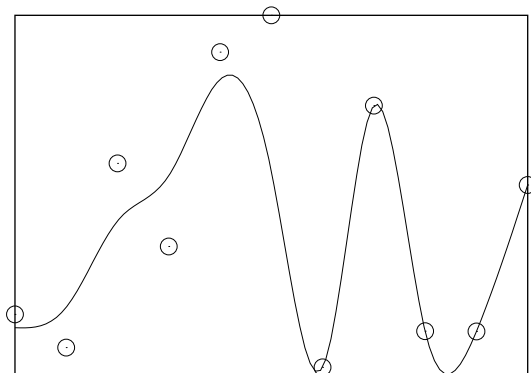


(b) グラフ

図 9.39: キーワード acsplines を使った例 (1)

100	0.4127	0.01
110	0.3610	0.01
120	0.6479	0.01
130	0.5184	0.01
140	0.8210	0.01
150	0.8784	0.01
160	0.3303	1000
170	0.7377	1000
180	0.3865	1000
190	0.3862	1000
200	0.6141	1000

(a) ファイル ac2.dat



(b) グラフ

図 9.40: キーワード acsplines を使った例 (2)

なお、重みをデータファイルで指定するかわりに gnuplot のコマンド行で直接指定することもできます。たとえば、図 9.39(a) のデータファイルで、重みに第 3 列目のデータを使うのをやめ、かわりにすべての行について数値 10.0 を使うことにする場合には、

```
plot 'ac1.dat' using 1:2:(10.0) smooth acsplines [Enter]
```

とします。この場合、キーワード using のあとの部分の表記は

横軸に使う列番号:縦軸に使う列番号:(重み)

のようになっています。3 番目の部分を括弧 () で囲む必要があることに注意して下さい。

9.1.9 デフォルトのデータファイル描画のスタイルの確認と変更

デフォルトのデータファイルの描画のスタイルを確認するには、

```
show data style [Enter]
```

と入力します。また、デフォルトのデータファイルの描画のスタイルを変更したいときには、たとえば

```
set data style lines [Enter]
```

などのように、

```
set data style
```

というキーワードに続けて希望するスタイルを指定します。

デフォルトのスタイルを変更すると、その変更は `gnuplot` を終了するまで有効になります。

9.2 3次元データファイルのプロット

9.2.1 単純な3次元データファイルのプロット

3次元データファイルをプロットするときには、コマンド `plot` のかわりに コマンド `splot` を使います。

データ点を単純にプロットしてゆくだけの場合、3次元データをプロットするやり方は2次元データファイルをプロットする場合と変わりません。

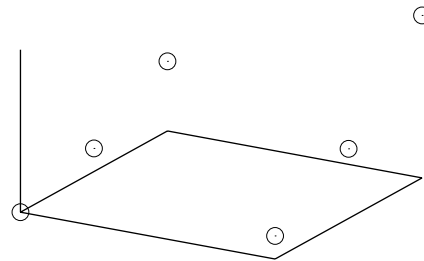
また、データファイルの記法も、データ点の座標として x 座標, y 座標, z 座標の3種類を指定する必要があること以外は2次元のものと同じです。

例として、図 9.41(a) に簡単な3次元データのデータファイルを示します。ファイルの名前は `3d0.dat` としておきます。このデータファイルを使い、

```
splot '3d0.dat' pt 6 ps 2 [Enter]
```

というコマンドを実行すると、図 9.41(b) のようなグラフが得られます。ただし、データ点を見やすくするためにデータ点の表示スタイルを変更した上でデータ点の大きさを大きくしています。この例からもわかるよう

1	1	250
1	2	300
1	3	400
2	1	300
2	2	400
2	3	600



(a) データファイル `3d0.dat`

(b) グラフ

図 9.41: 簡単な3次元データとそのグラフ

に、点の種類や大きさの変更法は2次元グラフを描画するときと同一です。

9.2.2 網かけグラフのプロット

ところで、3次元の関数をプロットするときには関数のグラフに網かけをすることで関数の形状が表現されたわけですが、同じことをデータファイルに対して実行するときには、いくつか注意が必要です。

そもそも網かけグラフとはどのようなものかという、 xy 平面に置かれた格子を曲面 $z = f(x, y)$ に投影した影、すなわち、曲面 $z = f(x, y)$ が光を半分透過するスクリーンであり、 xy 平面に格子が置かれていて、 xy 平面の下の方から拡散しない光をあてたときに、格子が曲面 $z = f(x, y)$ 上に作る像を斜めから見ているものです(図 9.42)。関数のグラフを作成するときにはこの格子が自動的に作成されたのですが、3次元データファイルから網かけグラフを作るときには、この格子を自分で作成しておく必要があります。

具体的には、たとえば図 9.43(a) に示すように、

- x 軸に平行な第1番目の格子の格子点の y 座標とその格子点における関数値から成るブロック
- x 軸に平行な第2番目の格子の格子点の y 座標とその格子点における関数値から成るブロック
-

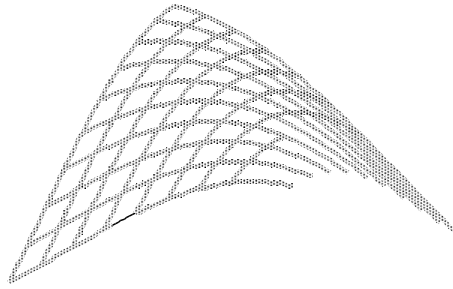


図 9.42: 網かけグラフの作り方

のようなデータを作成しておきます（ブロックとは、データファイル中の空白行で区切られたかたまりのことでした）。なお、どのブロックも同じ数のデータ点を含むようにしないといけません。

図 9.43(a) のファイルの名前を 3d1.dat としましょう。このファイルを

```
splot '3d1.dat' with linespoints pt 6 ps 2 [Enter]
```

によってプロットすると、図 9.43(b) のような網かけグラフが作成できます。

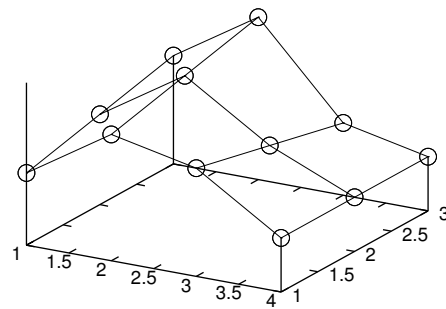
```
1 1 20
1 2 30
1 3 40

2 1 50
2 2 60
2 3 70

3 1 40
3 2 30
3 3 20

4 1 10
4 2 10
4 3 10
```

(a) データファイル 3d1.dat



(b) グラフ

図 9.43: 3 次元データの網かけグラフそのグラフ

実は、図 9.43 の例は実際には網かけグラフを作成しているわけではなく、同じブロックに含まれるデータ点のあいだを線で結んでから各々のブロックを結ぶ線を補っているだけです。

9.2.3 x 座標と y 座標の自動割り付け

2 次元データファイルでは、縦軸の値だけを書いたファイルを作っておくと、gnuplot が横軸の値として 0, 1, 2, ... という整数を自動的に割り振ってくれました。3 次元データファイルについても同様のことができます。

z 座標だけが書かれたファイルがあると、gnuplot は以下のような規則にしたがって x 座標と y 座標を補ってゆきます。

- データの最初の行では、gnuplot は x 座標および y 座標の両方に 0 を割り振る。
- データが 1 行下にゆくにしたがって x 座標を 1 ずつ増やす。y 座標は変えない。
- 空白行が見付かると、空白行の次のデータ行では y 座標を 1 増やす。x 座標は 0 に戻す。

ですから，たとえば，図 9.44(a) のデータファイルと図 9.44(b) のデータファイルを gnuplot によってプロットした場合，同じグラフが得られます．

<pre> 1 1 1 2 2 2 3 3 3 </pre>	<pre> 0 0 1 1 0 1 2 0 1 0 1 2 1 1 2 2 1 2 0 2 3 1 2 3 2 2 3 </pre>
(a)x 座標と y 座標を省略	(b) 全座標を記載

図 9.44: x 座標と y 座標が省略されたデータファイルと省略されていないデータファイル

9.2.4 グラフのスタイルの変更やデータファイルからのデータの抜き出し

2 次元データファイルのプロットで使えたキーワード `index`，`every`，`using` は 3 次元グラフでも使えます．使い方については 2 次元データファイルの対応する箇所を参照して下さい．

9.2.5 球座標と円柱座標

データファイルから 3 次元グラフを描くときには，データの座標系として球座標（3 次元極座標）や円柱座標を使うことができます．

9.2.5.1 座標系の切り換えのためのコマンド

直交座標と球座標，円柱座標の切り換えには `set mapping` というコマンドを使います．座標変換に関連したのコマンドの一覧を表 9.5 にまとめておきます．

表 9.5: 座標系の変換

機能	コマンド
球座標を選択	<code>set mapping spherical [Enter]</code>
円柱座標を選択	<code>set mapping cylindrical [Enter]</code>
直交座標を選択	<code>set mapping cartesian [Enter]</code>
現在選択されている座標系を表示	<code>show mapping</code>

球座標や円柱座標を用いたグラフのプロットはデータファイルのプロットのときのみ有効です．関数をプロットするときには使えません．

9.2.5.2 球座標によるデータファイル

球座標用のデータファイルは 1 行あたり 2 個あるいは 3 個の数値を含みます．

1 行あたりの数値が 3 個のデータファイルでは、 i 番目のデータ点の極座標が (r_i, θ_i, ϕ_i) であるとき、すなわち直交座標に直すと

$$x_i = r_i \cos \theta_i \cos \phi_i, \quad y_i = r_i \sin \theta_i \cos \phi_i, \quad z_i = r_i \sin \phi_i$$

という座標にデータ点があるとき、対応するデータファイルの行には、数値が

$$\theta_i, \phi_i, r_i$$

という順で並びます。 r_i が第 3 番目に来ることに注意して下さい。

1 行あたりの数値が 2 個のデータファイルでは、データ点の原点からの距離はつねに 1 であると解釈されます。すなわち、データファイルの第 i 行目に

$$\theta_i, \phi_i$$

という数値が記載されているとき、gnuplot はこのデータ点を極座標 $(1, \theta_i, \phi_i)$ を持つ点であると解釈します。

例として、半球面上にある点の極座標がいくつか保存されたファイル `spherical.dat` のプロットを示します。ファイル `spherical.dat` の内容は図 9.45(a) のようなものであるとします。ファイル `spherical.dat` には 250 個のデータ点が含まれるのですが、図 9.45(a) にはそのうち最初の 5 個だけが記載されています。

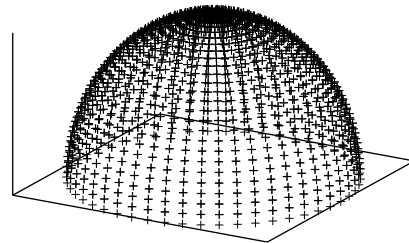
図 9.45(a) のようなデータファイルを準備した上で

```
set mapping spherical [Enter]
splot 'spherical.dat' [Enter]
```

というコマンドを実行すると、図 9.45(b) のようなグラフが描画されます。

0.000	0.000
0.000	0.063
0.000	0.126
0.000	0.188
0.000	0.251
(以下略)	

(a) データファイル



(b) グラフ

図 9.45: 半球面上のデータ点の描画例

9.2.5.3 円柱座標によるデータファイル

円柱座標用のデータファイルは 1 行あたり 2 個あるいは 3 個の数値を含みます。

1 行あたりの数値が 3 個のデータファイルでは、 i 番目のデータ点の極座標が (r_i, θ_i, z_i) であるとき、すなわち直交座標に直すと

$$x_i = r_i \cos \theta_i, \quad y_i = r_i \sin \theta_i, \quad z_i = z_i$$

という座標にデータ点があるとき、対応するデータファイルの行には、数値が

$$\theta_i, z_i, r_i$$

という順で並びます。球座標の場合と同様に r_i が第 3 番目に来ることに注意して下さい。

1 行あたりの数値が 2 個のデータファイルでは、データ点の原点からの距離はつねに 1 であると解釈されます。すなわち、データファイルの第 i 行目に

$$\theta_i, z_i$$

という数値が記載されているとき、gnuplot はこのデータ点を球座標 $(1, \theta_i, z_i)$ を持つ点であると解釈します。

例として、円筒にある点の円柱座標がいくつか保存されたファイル `cylindrical.dat` のプロットを示します。ファイル `cylindrical.dat` の内容は図 9.46(a) のようなものであるとします。ファイル `cylindrical.dat` には 250 個のデータ点が含まれるのですが、図 9.46(a) にはそのうち最初の 5 個だけが記載されています。

図 9.46(a) のようなデータファイルを準備した上で

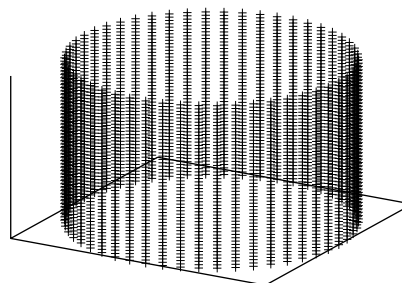
```
set mapping cylindrical [Enter]
```

```
splot 'cylindrical.dat' [Enter]
```

というコマンドを実行すると、図 9.46 のようなグラフが描画されます。

0.000	0.000
0.000	0.020
0.000	0.040
0.000	0.060
0.000	0.080
(以下略)	

(a) データファイル



(b) グラフ

図 9.46: 円筒上のデータ点の描画

9.3 非線形最小 2 乗法によるデータへの曲線のあてはめ

gnuplot には、与えられたデータファイルに記載されたデータに曲線や曲面をあてはめる機能があります。本節ではこの機能について簡単に紹介します。

9.3.1 簡単な例題

最小 2 乗法でデータに曲線や曲面をあてはめる手順は、

- パラメータを含む関数を定義する
- コマンド `fit` を使って先に定義された関数をデータファイルに記載されたデータにあてはめる

というものです。

さっそく、例題を見てみましょう。

図 9.47(a) のようなデータファイルを考えます。ファイル名は `f0.dat` とします。

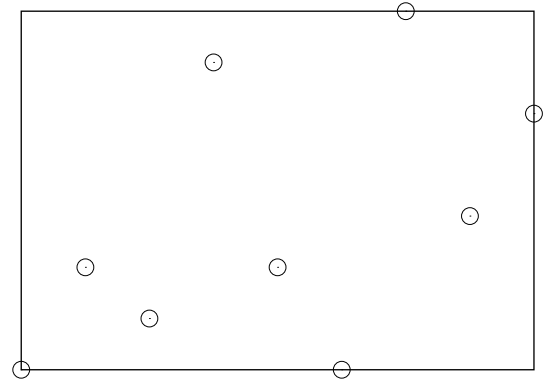
このデータに、 $f(x) = ax + b$ という直線をあてはめてみることにします。このためには、

```

10 10
20 30
30 20
40 70
50 30
60 10
70 80
80 40
90 60

```

(a) データファイル f0.dat



(b) f0.dat をプロットしたもの

図 9.47: データファイル f0.dat とそのプロット

```

f(x)=a*x+b [Enter]
fit f(x) 'f0.dat' via a,b [Enter]

```

と入力します。すると、gnuplot のウィンドウは図 9.48 のような状態になります。このウィンドウには、パラ

```

a          = 0.483333
b          = 14.7222

After 5 iterations the fit converged.
final sum of squares of residuals : 3887.22
rel. change during last iteration : -3.03736e-10

degrees of freedom (ndf) : 7
rms of residuals      (stdfit) = sqrt(WSSR/ndf)      : 23.5652
variance of residuals (reduced chisquare) = WSSR/ndf : 555.317

Final set of parameters          Asymptotic Standard Error
=====
a          = 0.483333          +/- 0.3042          (62.94%)
b          = 14.7222           +/- 17.12           (116.3%)

correlation matrix of the fit parameters:

          a      b
a          1.000
b         -0.889  1.000
gnuplot> █

```

図 9.48: コマンド fit の実行が終わったときの gnuplot のウィンドウの状態

メータの値、計算が終了するまでにおこなわれた繰り返し計算の回数およびいくつかの統計情報が表示されます。では、これに続いて、計算された直線とデータを重ね書きしてみましょう。

```

plot 'f0.dat' pt 6 ps 2, f(x) [Enter]

```

というコマンドを実行し、データファイル f0.dat と関数 f(x) を重ね書きしてみます。すると、図 9.49 のような結果が得られます。

次に、このグラフに

$$f(x) = ax + b + c \sin(dx)$$

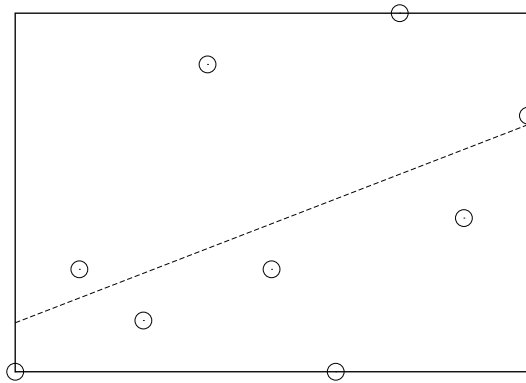


図 9.49: データ f0.dat に直線を当てはめた結果

と関数をあてはめてみます．ただし，計算を始める前に，パラメータ a は 0.5 b は 15, c は 10, d は 0.1 で初期化しておくことにします．

```
a=0.5 [Enter]
b=15 [Enter]
c=10 [Enter]
d=0.1 [Enter]
e=1 [Enter]
f(x)=a*x+b + c *sin(d*x) [Enter]
fit f(x) 'f0.dat' via a,b,c,d [Enter]
plot 'f0.dat' pt 5 ps 3, f(x) [Enter]
```

というコマンドを実行すると，すると，図 9.50 のような結果が得られます．

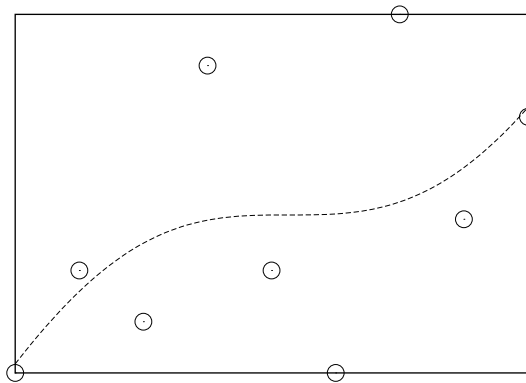


図 9.50: データ f0.dat に非線形関数を当てはめた結果

計算が終わったときのパラメータをファイルに保存しておきたいときには，

```
save variables 'パラメータファイル' [Enter]
```

とします．ここに，「パラメータファイル」と書かれた部分には適当なファイル名が入ります．上記のコマンドを実行すると，たとえば図 9.51 のようなファイルが作成されます．

```

#!/usr/local/bin/gnuplot -persist
#
#
#      G N U P L O T
#      Unix version 3.7
#      patchlevel 1 (+1.2.0 2001/01/11)
#      last modified Fri Oct 22 18:00:00 BST 1999
#
#      Copyright (C) 1986 - 1993, 1998, 1999
#      Thomas Williams, Colin Kelley and many others
#
#      Type 'help' to access the on-line reference manual
#      The gnuplot FAQ is available from
#      <http://www.ucc.ie/gnuplot/gnuplot-faq.html>
#
#      Send comments and requests for help to <info-gnuplot@dartmouth.edu>
#      Send bugs, suggestions and mods to <bug-gnuplot@dartmouth.edu>
#
a = 0.873873120212733
b = -5.19530356445324
c = 14.6621983960157
d = 0.0596332775518248
e = 1
#      EOF
(END)

```

図 9.51: パラメータが保存されたファイル

9.3.2 コマンド fit の使い方

コマンド `fit` の基本的な使い方は 2 種類あります。

ひとつめは、

`fit 関数 'データファイル' via 変数 1, 変数 2, ... [Enter]`

というものです。この場合、「変数 1」、「変数 2」と書かれた部分には、当てはめに使う関数に含まれる調整可能なパラメータの名前が入ります。

ふたつめの使い方は、

`fit 関数 'データファイル' via 'パラメータファイル' [Enter]`

というものです。この使い方をする場合には、あらかじめパラメータの初期値が書かれたファイルを用意しておく必要があります。

パラメータファイルの作るときには、たとえば図 9.52 に示すように、変数の名前とその初期値を等号をはさんで並べます。先ほどの例で見た、図 9.51 のファイルも、註釈行（文字「`#`」で始まる行）を除くと、この形をしています。

```

a=0.5
b=10

```

図 9.52: パラメータファイルの記法

ふたつめの使い方をした場合、コマンド `fit` の実行が終了した段階で、

`update 'パラメータファイル' [Enter]`

というコマンドを実行すると、パラメータファイルの内容が更新されます。

パラメータの中に値を変えてはいけないものがあるときには、たとえば図 9.53 のパラメータ a の部分のように、パラメータ値の右側に「#FIXED」という文字を書いております。そうすると、パラメータの値が更新されることはありません。

```
a=0.5 \#FIXED  
b=10
```

図 9.53: 固定パラメータを含むパラメータファイルの記法

9.3.3 データに非線形関数を当てはめるときの注意

データに当てはめる関数が線形有的时候には、パラメータの初期値をどのように取っても最終的に最適な当てはめが達成されることが保証されます。これに対し、非線形関数を当てはめるときには、最終的な当てはめの結果の良し悪しがパラメータの初期値に大幅に依存します。

- 線形関数を当てはめることができる場合には、可能な限り線形関数を使う
- 非線形関数を当てはめに使うときには初期値の選択に注意する

ということを心掛けて下さい。

第10章 plus モード：数式の組版とテキストのより細かい制御

gnuplot には、plus モードという、簡単な数式を組んだり、いくつかの図記号を入れたり、文字の大きさや書体などを細かく制御したりするためのモードがあります。

本節ではこの plus モードについて説明してゆきます。

10.1 plus モードの開始と終了

先に述べた通り、plus モードでは、簡単な数式を組んだり、いくつかの図記号を入れたり、文字の大きさや書体などを細かく制御したりすることができます。

ただし、この機能は、ターミナルが `postscript` になっている場合、すなわち PostScript 形式でファイルに保存する場合にしか有効になりません。

さらに、ただの PostScript モードでも、数式は図記号などが取り扱えるわけではありません。ですから、plus モードの機能を使いたいときには、ターミナルを `postscript` の plus モードにしておく必要があります。

このためには、

```
set terminal postscript eps plus [Enter]
```

あるいは

```
set terminal postscript plus [Enter]
```

とします。

印刷結果をプリンタに直接送る場合には後者でもよいのですが、作成した図をほかの文書に取り込む予定ならば前者のほうが良いでしょう。この節の説明の内容は前者でも後者でも変わらないのですが、説明の便宜上本節では前者に限定して議論を進めます。

plus モードを起動すると、作成される PostScript ファイル には日本語のデータが入ります。ですから、手持ちのプリンタが日本語に対応していないなどの理由で日本語のデータを含まない PostScript ファイルを作成しなければならないときには、

```
set locale "" [Enter]
```

と指定する必要があります。逆に、日本語のデータを入力できるモードに戻るには、

```
set locale "ja_JP.EUC" [Enter]
```

とします。

ターミナルが `postscript` の状態で plus モードを無効にしたいときには、

```
set terminal postscript eps noplus [Enter]
```

とします。

ここまでに出て来たコマンドを表 10.1 にまとめておきます。

表 10.1: plus モードの起動と終了に関連したコマンド

機能	コマンド
plus モードを有効にする	set terminal postscript eps plus [Enter]
plus モードを無効にする	set terminal postscript eps noplus [Enter]
日本語を有効にする	set locale "ja_JP.EUC" [Enter]
日本語を無効にする	set locale "" [Enter]

なお、plus モードでは、文字列は必ず単一引用符「'」で囲まなければなりません。これ以外のモードでは「"」と「'」のどちらを使っても良かったのですが、plus モードで「"」を使うと誤動作が起きることがあります。また、いわゆる半角カナは使えません。

- 文字列は単一引用符「'」で囲まなければならない
- 半角カナは使えない

ということを記憶しておいて下さい。

10.2 plus モードのコマンド

plus モードにおけるコマンドの記法は基本的に \TeX と同じで、最初の文字が「\」（バックスラッシュ）で始まります。コンピュータの状態によっては記号「\」が画面には円記号（「¥」のような形）として表示されることがあるので注意して下さい。また、キーボードの表面にも、記号「\」のかわりに記号「¥」が表示されていることがあります。

コマンドの有効範囲を明示する必要が生じることもあります。この典型的な例が、後の 10.3.2 節で述べるフォントの選択です。

コマンドの有効範囲を指示するためには、記号「{」「}」を使います。「{」が有効範囲の始まり、「}」が有効範囲の終わりです。たとえば、コマンド `\it` (10.3.2 節参照) の有効範囲を限定するには、

```
{\it }
```

などのようにします。

コマンドの中には引数を取るものもあります。コマンドの引数をあらわすのにも、多くの場合は記号「{」「}」を使います。

たとえば、コマンド `\sqrt` は平方根記号を表示するコマンドなのですが（後述、表 10.5）、数 2 に平方根の記号を付けるには、

```
\sqrt{2}
```

のように、コマンド `\sqrt` の引数として 2 を指定します。

これ以外に、引数を取るコマンドで、

```
\command=値
```

という書き方を要求するものもあります（後述、10.3 節）。

10.3 文字の大きさや字体の指定と回転

plus モードでは、文字の大きさや字体を指定したり、テキスト全体を指定した角度だけ回転したりすることができます。

10.3.1 文字の大きさの指定

文字の大きさを指定するときには、文字列のどこかに

`\size=数`

と書きます。ここに、「数」と書いてある部分には文字の大きさが入ります。単位はポイントです。

文字の大きさをテキストの最初で指定すると、次に`\size` コマンドがあらわれるまでは先に指定した大きさの文字が使われます。テキストの途中でも文字の大きさを変更できます。

図中でいろいろな大きさの文字を使う例として、

```
set terminal postscript eps plus? [Enter]
set output 'gnuplot-plus-size.eps'? [Enter]
set label '\size=30 30 ポイント \size=32 32 ポイント' at 0.2,0.8 [Enter]
set label '\size=26 26 ポイント \size=28 28 ポイント' at 0.2,0.7 [Enter]
set label '\size=22 22 ポイント \size=24 24 ポイント' at 0.2,0.6 [Enter]
set label '\size=18 18 ポイント \size=20 20 ポイント' at 0.2,0.5 [Enter]
set label '\size=14 14 ポイント \size=16 16 ポイント' at 0.2,0.4 [Enter]
set label '\size=10 10 ポイント \size=12 12 ポイント' at 0.2,0.3 [Enter]
set label '\size=6 6 ポイント \size=8 8 ポイント' at 0.2,0.2 [Enter]
set label '\size=2 2 ポイント \size=4 4 ポイント' at 0.2,0.1 [Enter]
plot [0:2] [0:1.0] 0 [Enter]
```

というコマンドを実行したときの出力を図 10.1 に示します。

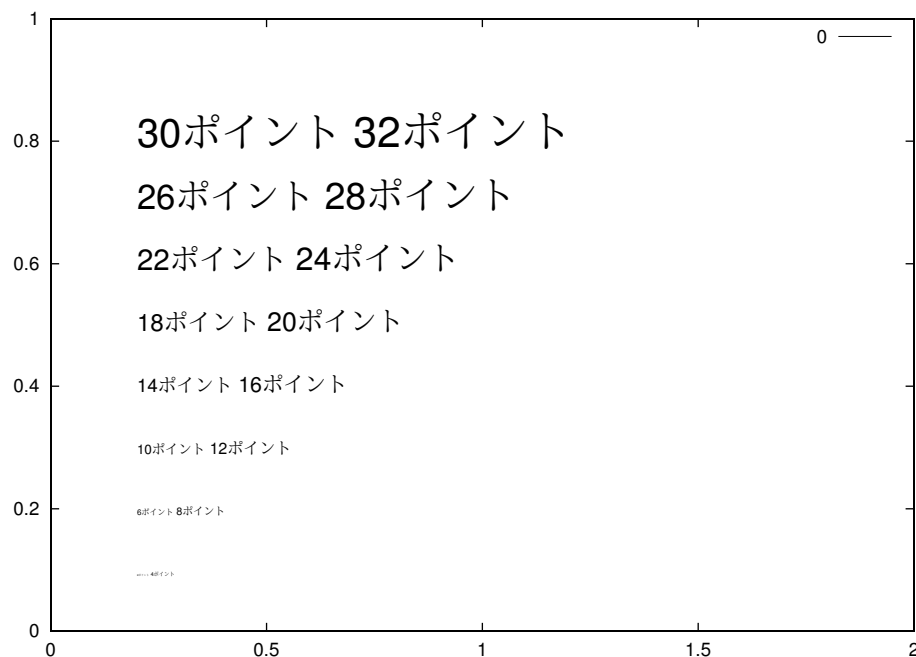


図 10.1: いろいろな文字の大きさの指定

10.3.2 字体の指定

字体を指定するためのコマンドとして、`\rm`、`\it`、`\tt`、`\sf`、`\bf` があります。これらの意味を表 10.2 に示します。

ボールド体を指定するとローマン体に対応する太字の字体が選択されますが、`\it`、`\tt`、`\sf` と組み合わせることで、それぞれ対応する字体の太字が得られます。

表 10.2: 利用可能な字体一覧

記号	コマンド	記号	コマンド	記号	コマンド
<code>\rm</code>	ローマン体	<code>\it</code>	イタリック体	<code>\tt</code>	タイプ文字
<code>\sf</code>	サンセリフ体	<code>\bf</code>	太字		

以下にいろいろな字体を選択する例を示します。ただし、グラフをみやすくするために、図の大きさを縦横とも標準の半分にしてあります（最初の行）。

```
set terminal postscript eps plus? [Enter]
set output 'gnuplot-plus-font.eps' [Enter]
set label '{\sf Sanserif} {\tt\bf BoldSanserif} ' at 0.5,0.7 [Enter]
set label '{\tt TypeWriter} {\tt\bf BoldTypeWriter}' at 0.5,0.5 [Enter]
set label '{\it Italic} {\it\bf BoldItalic} ' at 0.5,0.3 [Enter]
set label '{\rm Roman} {\bf Bold} ' at 0.5,0.1 [Enter]
plot [0:2] [0:1.0] 0 [Enter]
```

結果は図 10.2 のようになります。

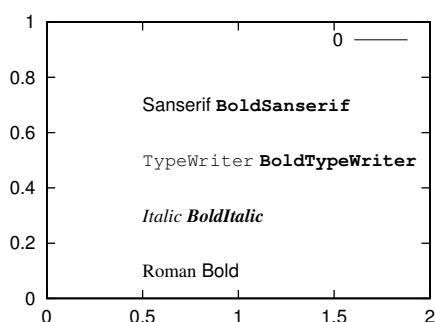


図 10.2: いろいろな字体の選択

10.3.3 テキスト全体の回転

テキスト全体を回転させるには、文字列の開始をあらわす単一引用符の直後に

`\rotate=角度`

と書きます。ここに、「角度」と書いてある部分には回転の角度が入ります。単位は度です。

テキストを回転させたいときには、必ず文字列の最初（文字列の開始をあらわす単一引用符の直後）でコマンド `\rotate` を使わなければなりません。それ以外の場所ではエラーになります。

例として、文字列をいろいろな角度回転させたものを含む図を作成してみます。ただし、グラフをみやすくするために、図の大きさを縦横とも標準の半分にしておきます（最初の行のコマンド）。

```
set size 0.5,0.5 [Enter]
set terminal postscript eps plus [Enter]
set output 'gnuplot-plus-rotate.eps' [Enter]
set label '\rotate=315 315 度回転' at 1.1,0.4 [Enter]
set label '\rotate=270 270 度回転' at 1.0,0.4 [Enter]
set label '\rotate=225 225 度回転' at 0.9,0.4 [Enter]
set label '\rotate=180 180 度回転' at 0.9,0.5 [Enter]
set label '\rotate=135 135 度回転' at 0.9,0.6 [Enter]
set label '\rotate=90 90 度回転' at 1.0,0.6 [Enter]
set label 'rotate=45 45 度回転' at 1.1,0.6 [Enter]
set label 'rotate=0 0 度回転' at 1.1,0.5 [Enter]
plot [0:2] [0:1.0] 0 [Enter]
```

というコマンドを実行すると、図 10.3 のような結果が得られます。

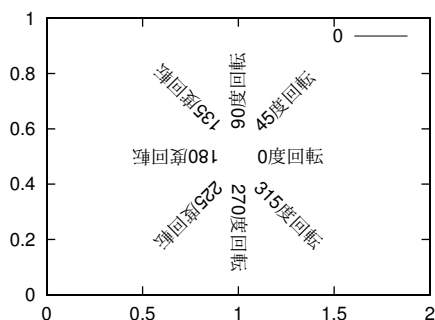


図 10.3: いろいろな角度の回転

10.4 特殊記号および各種の記号

10.4.1 特殊記号

コマンドの一部として使われることが多いいくつかの記号はふだんは単体では表示できなくなっています。それらを表示するためのコマンドを表 10.3 に示します。

表 10.3: 特殊記号一覧

記号	コマンド	記号	コマンド	記号	コマンド
\$	\\$	^	\^	_	_
\	\backslash	{	\{	}	\}

使用例を以下に示します。

```
set terminal postscript eps plus [Enter]
```



```

set output 'gnuplot-plus-special.eps' [Enter]
set label '\size=150 \$ \^ \_ \{ \}' at 0.1,0.5 [Enter]
plot [0:2] [0:1.0] sin(x) [Enter]

```

結果は図 10.4 のようになります。

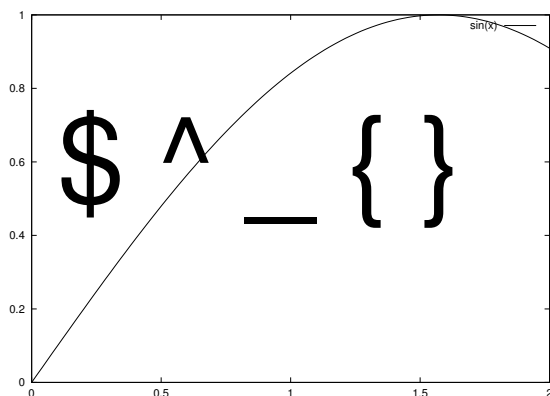


図 10.4: 特殊記号の使用例

10.4.2 各種記号, 外国語の文字

著作権表示の記号, 英語以外の外国語の記号などのいくつかの記号が用意されています。これらを表 10.4 に示します。

表 10.4: 各種記号と外国語記号一覧

記号	コマンド	記号	コマンド	記号	コマンド	記号	コマンド
†	\dag	‡	\ddag	§	\S	¶	\P
©	\copyright	£	\pounds	œ	\oe	Œ	\OE
æ	\ae	Æ	\AE	å	\aa	Å	\AA
ø	\o	Ø	\O	ł	\l	Ł	\L
ß	\ss						

使用例を以下に示します。

```

set output 'gnuplot-plus-misc-symbols.eps' [Enter]
\verb?set title '\size=24 \copyright 2001 ' [Enter]
\verb?set label '\size=48 \dag \ddag \S \P \pounds \AE \ae \OE \oe' at 0.1,0.5 [Enter]
plot [0:2] [0:1] sin(x) [Enter]

```

結果は図 10.5 のようになります。

10.5 数式の表現

plus モードにおける数式の表現は \TeX の数式組版機能を簡略化したものです。 \TeX を知らない人は,

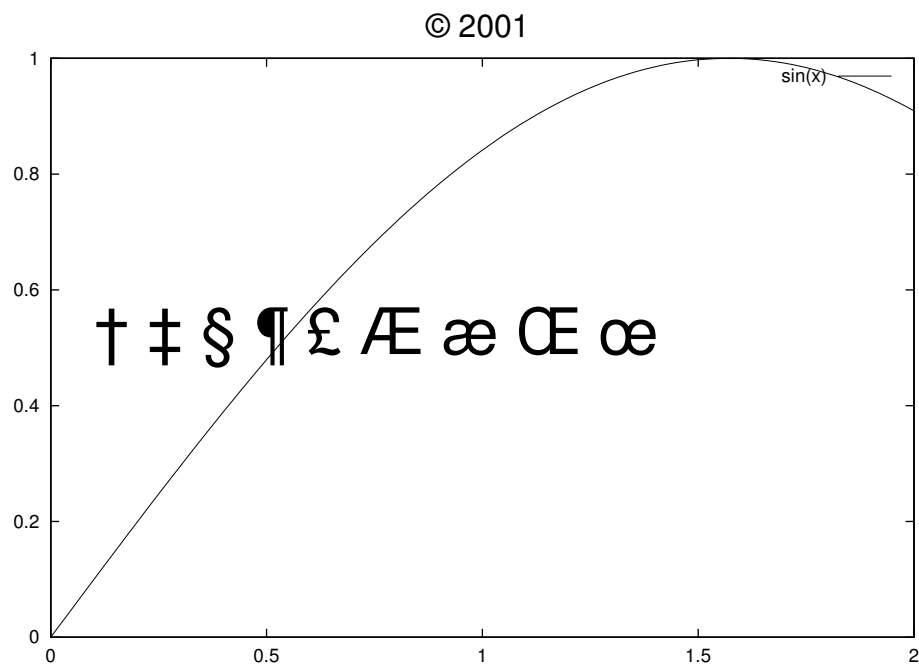


図 10.5: いろいろな記号の使用例

- 記号\$ではさまれている領域が数式として認識される
- 下付き添字を使いたいときには数式環境中（記号\$ で挟まれた領域）に a_1 などのように書く
- 上付き添字を使いたいときには数式環境中に a^1 などのように書く

ということだけ覚えておいて下さい。

10.6 グラフの標題に簡単な数式を入れる

まず簡単な例題を見ておくことにします。

以下に、グラフの横軸に「下付き添字の例: a_1 」、縦軸に「上付き添字の例: a^2 」というラベルを付け、グラフ全体に「平方根の例: $\sqrt{2}$ 」という題目を付けている例を示します。

```
set terminal postscript eps plus [Enter]
set output 'gnuplot-plus-1.eps' [Enter]
set xlabel '下付き添字の例: $a_1$' [Enter]
set ylabel '上付き添字の例: $a^2$' [Enter]
set title '平方根の例: $\sqrt{2}$' [Enter]
plot sin(x) [Enter]
```

結果は図 10.6 のようになります。

この例からもわかるように、数式を入れるときには、文字列を

‘ 数式でない部分\$数式\$’

のように書きます。数式が 2 個含まれる場合には、

‘ 数式でない部分\$数式 1\$数式でない部分 2\$数式 2\$’

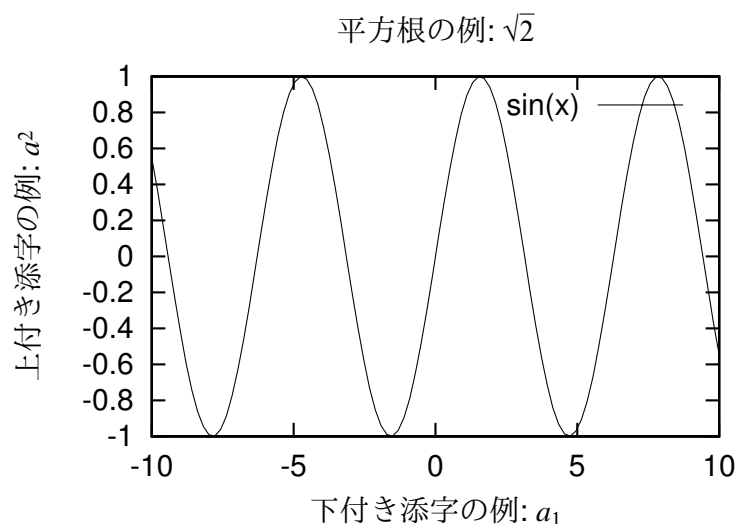


図 10.6: グラフに簡単な数式を入れた例

のようにします。数式が 3 個以上含まれる場合も同様です。数式が 2 個以上ある場合には、数式と認識される部分は

- 1 個目の記号\$と 2 個目の記号\$のあいだ
- 3 個目の記号\$と 4 個目の記号\$のあいだ
-

となります。記号\$の数が奇数個の場合はエラーになります。

10.7 数式を組むためのいくつかのコマンド

plus モードには数式を組むときに必要となる、分数、平方根を表示するためのコマンドや、数式の上や下に線を引くコマンドが用意されています。これらの一覧を表 10.5 に示します。

表 10.5: 数式を組むためのコマンド一覧

機能	コマンド	使用例	結果
分数	<code>\frac</code>	<code>\frac{2}{3}</code>	$\frac{2}{3}$
分数の平方根	<code>\fracsqrt</code>	<code>\fracsqrt{\frac{2}{3}}</code>	$\sqrt{\frac{2}{3}}$
分数の上に線を引く	<code>\fracoverline</code>	<code>\fracoverline{\frac{2}{3}}</code>	$\overline{\frac{2}{3}}$
平方根	<code>\sqrt</code>	<code>\sqrt{2}</code>	$\sqrt{2}$
上に線を引く	<code>\overline</code>	<code>\overline{テキスト, 数式}</code>	テキスト, 数式
下に線を引く	<code>\underline</code>	<code>\underline{テキスト, 数式}</code>	テキスト, 数式
ボックスに入れる	<code>\mbox</code>	<code>\mbox{テキスト, 数式}</code>	テキスト, 数式

表 10.5 の最後の 3 個のコマンドは数式以外でも使えます。

10.8 利用可能な数学記号

plus モードで利用可能な数学記号は \TeX で用意されている数学記号に若干の変更を加えたものです。以下にこれらを列挙してゆきます。

10.8.1 矢印

上下左右の向きの矢印と両側に矢のついた矢印が利用可能です。表 10.6 に一覧を示します。「記号」という見出しがついた部分に書かれているのが表示される記号、「コマンド」という見出しがついた部分に書かれているのが対応するコマンドです。

表 10.6: 数式モードにおける矢印記号一覧

記号	コマンド	記号	コマンド	記号	コマンド	記号	コマンド
\Downarrow	<code>\Downarrow</code>	\Leftarrow	<code>\Leftarrow</code>	\Leftrightarrow	<code>\Leftrightarrow</code>	\Rightarrow	<code>\Rightarrow</code>
\Uparrow	<code>\Uparrow</code>	\downarrow	<code>\downarrow</code>	\leftarrow	<code>\leftarrow</code>	\leftrightarrow	<code>\leftrightarrow</code>
\rightarrow	<code>\rightarrow</code>	\uparrow	<code>\uparrow</code>				

10.8.2 ギリシャ文字

ギリシャ文字については、いくつかの大文字と小文字すべて、さらに小文字の書体を変えたものが利用可能です。ギリシャ文字の大文字に省略されているものがあるのは、それらの字形が英語の大文字と同じだからです。表 10.7 に大文字一覧を、表 10.8 に小文字一覧を示します。表の見方は先と同様です。

表 10.7: ギリシャ文字 (大文字) 一覧

記号	コマンド	記号	コマンド	記号	コマンド	記号	コマンド
Γ	<code>\Gamma</code>	Δ	<code>\Delta</code>	Θ	<code>\Theta</code>	Λ	<code>\Lambda</code>
Ξ	<code>\Xi</code>	Π	<code>\Pi</code>	Σ	<code>\Sigma</code>	Υ	<code>\Upsilon</code>
Φ	<code>\Phi</code>	Ψ	<code>\Psi</code>	Ω	<code>\Omega</code>		

表 10.8: ギリシャ文字 (小文字) 一覧

記号	コマンド	記号	コマンド	記号	コマンド	記号	コマンド
α	<code>\alpha</code>	β	<code>\beta</code>	γ	<code>\gamma</code>	δ	<code>\delta</code>
ϵ	<code>\epsilon</code>	ε	<code>\varepsilon</code>	ζ	<code>\zeta</code>	η	<code>\eta</code>
θ	<code>\theta</code>	ϑ	<code>\vartheta</code>	ι	<code>\iota</code>	κ	<code>\kappa</code>
λ	<code>\lambda</code>	μ	<code>\mu</code>	ν	<code>\nu</code>	ξ	<code>\xi</code>
π	<code>\pi</code>	ϖ	<code>\varpi</code>	ρ	<code>\rho</code>	σ	<code>\sigma</code>
ς	<code>\varsigma</code>	τ	<code>\tau</code>	υ	<code>\upsilon</code>	ϕ	<code>\phi</code>
φ	<code>\varphi</code>	χ	<code>\chi</code>	ψ	<code>\psi</code>	ω	<code>\omega</code>

10.8.3 数式中のアクセント記号

数式を組んでいると、文字の上に微分をあらわすドットなどのいろいろな印をつける必要が生じることがあります。このようなときのためのコマンドを表 10.9 にまとめておきます。なお、表 10.9 ではアクセントが付けられる文字の例を a にしてありますが、この部分には任意の文字を指定できます。

表 10.9: 数式中のアクセント記号一覧

記号	コマンド	記号	コマンド	記号	コマンド	記号	コマンド
\hat{a}	<code>\hat{a}</code>	\check{a}	<code>\check{a}</code>	\breve{a}	<code>\breve{a}</code>	\acute{a}	<code>\acute{a}</code>
\grave{a}	<code>\grave{a}</code>	\tilde{a}	<code>\tilde{a}</code>	\bar{a}	<code>\bar{a}</code>	\vec{a}	<code>\vec{a}</code>
\dot{a}	<code>\dot{a}</code>	\ddot{a}	<code>\ddot{a}</code>				

10.8.4 関数記号

いろいろな数学関数のために対応する記号が用意されています。関数記号を表示するためのコマンドの大部分は、その関数名の先頭に記号「\」をつけたものです。関数記号の一覧を表 10.10 に示します。

表 10.10: 関数記号一覧

記号	コマンド	記号	コマンド	記号	コマンド	記号	コマンド
\arccos	<code>\arccos</code>	\arcsin	<code>\arcsin</code>	\arctan	<code>\arctan</code>	\arg	<code>\arg</code>
\cos	<code>\cos</code>	\cosh	<code>\cosh</code>	\cot	<code>\cot</code>	\coth	<code>\coth</code>
\csc	<code>\csc</code>	\deg	<code>\deg</code>	\det	<code>\det</code>	\dim	<code>\dim</code>
\exp	<code>\exp</code>	\gcd	<code>\gcd</code>	\hom	<code>\hom</code>	\inf	<code>\inf</code>
\ker	<code>\ker</code>	\lg	<code>\lg</code>	\lim	<code>\lim</code>	\liminf	<code>\liminf</code>
\limsup	<code>\limsup</code>	\ln	<code>\ln</code>	\log	<code>\log</code>	\max	<code>\max</code>
\min	<code>\min</code>	\Pr	<code>\Pr</code>	\sec	<code>\sec</code>	\sin	<code>\sin</code>
\sinh	<code>\sinh</code>	\sup	<code>\sup</code>	\tan	<code>\tan</code>	\tanh	<code>\tanh</code>

10.8.5 2項演算子と関係記号

数式を組むときには、加減乗除の記号「+」、「-」、「*」、「/」以外にもいろいろな記号がありますし、そもそも数学でよく使われる乗算記号 \times は英文字 x とまぎらわしいためにコンピュータではキーボードから直接入力できるようにはなっていません。また、不等号の記号「<」「>」はキーボードにありますが、記号「 \leq 」「 \geq 」はキーボードにはありません。

plus モードには、このような記号を表示する能力があります。利用可能な記号の一覧を表 10.11

10.8.6 その他の数学記号

これまで述べてきた以外にも、数式を組んでいるときには、和の記号 \sum 、積の記号 \prod 、偏微分記号 ∂ などのいろいろな記号が必要になります。

これまで述べてきたものの以外で用意されている数学記号の一覧を表 10.12 に示します。

表 10.11: 2 項演算子と関係記号一覧

記号	コマンド	記号	コマンド	記号	コマンド	記号	コマンド
\pm	<code>\pm</code>	\times	<code>\times</code>	\cdot	<code>\cdot</code>	\div	<code>\div</code>
\dagger	<code>\dagger</code>	\equiv	<code>\equiv</code>	\sim	<code>\sim</code>	\approx	<code>\approx</code>
\cong	<code>\cong</code>	\propto	<code>\propto</code>				
\leq	<code>\leq</code>	\geq	<code>\geq</code>	\neq	<code>\neq</code>		
\leq	<code>\le</code>	\geq	<code>\ge</code>	\neq	<code>\ne</code>		

表 10.12: その他の数学記号一覧

記号	コマンド	記号	コマンド	記号	コマンド	記号	コマンド
\aleph	<code>\aleph</code>	\hbar	<code>\hbar</code>	\imath	<code>\imath</code>	\wp	<code>\wp</code>
\Re	<code>\Re</code>	\Im	<code>\Im</code>	\angle	<code>\angle</code>	∇	<code>\nabla</code>
$\sqrt{\quad}$	<code>\surd</code>	∂	<code>\partial</code>	∞	<code>\infty</code>	\sum	<code>\sum</code>
\int	<code>\smallint</code>	$\langle \quad \rangle$	<code>\langle \quad \rangle</code>	$\langle \quad \rangle$	<code>\langle \quad \rangle</code>		

10.8.7 大きい括弧

複雑に入り組んだ数式を組んでいるときには、いろいろな大きさの括弧を使って数式をみやすくする、ということがよく行われます。このような目的のために、通常より大きい括弧が何種類か用意されています。

表 10.13: 大きい括弧一覧

記号	コマンド	記号	コマンド	記号	コマンド	記号	コマンド
(\quad)	<code>\bigl(\quad)</code>	(\quad)	<code>\bigr(\quad)</code>	$\{\quad\}$	<code>\bigl\{\quad\}</code>	$\{\quad\}$	<code>\bigr\}\quad\}</code>

10.8.8 いくつかの例

参考のために、数式の組版の例をいくつか表 10.14 に示します。これらは数式モード（記号「\$」で囲まれた領域）でしか使えないことに注意して下さい。また、実際の組版の内容は表 10.14 に示されているものに比べてかなり汚くなります。

10.8.9 機能の制限

これまで紹介してきたコマンドには、いくつか機能制限があります。どのような制限があるかというと、

- 添字に分数を使うことはできない
- 分数の中にさらに分数を指定することはできない

というものです。このうち、最初の方の制限は若干問題になることがあります。たとえば、 $a^{\frac{1}{10}}$ のような数式を組むことはできません。

また、10.8.8 節でも述べたように、組まれる数式は一般にあまり美しくありません。より美しい数式を組みたい場合は、terminal を latex にしてグラフを保存するか、terminal を tgif にしてグラフを保存して tgif を使って図を入れるとよいでしょう。

表 10.14: 数式組版の例

数式	コマンド
$\sin t \leq t$	<code>\sin t \leq t</code>
$\{((a_1^2 + a_2^2) + a_3^2) + a_4^2\}$	<code>\bigl\{ \bigl((a_1^2+a_2^2)+a_3^2 \bigr)+a_4^2 \bigr\}</code>
$\sum_{i=1}^{\infty} a_i \times b_i$	<code>\sum_{i=1}^{\infty} a_i \times b_i</code>
$\int_{p_1}^{p_2} \exp \tau d\tau$	<code>\smallint_{p_1}^{p_2} \exp [\tau] d \tau</code>
$\frac{\partial \Psi}{\partial x}$	<code>\frac{\partial \Psi}{\partial x}</code>
$\sqrt{\frac{b_0 s + b_1}{s^2 + a_1 s + a_2}}$	<code>\frac{\sqrt{b_0 s + b_1}}{s^2 + a_1 s + a_2}</code>
$\lim_{x \rightarrow 0} \frac{\sin x}{x}$	<code>\lim_{x \rightarrow 0} \frac{\sin x}{x}</code>
$\ddot{x} = -\lambda \dot{x} - \mu x + \cos(t)$	<code>\ddot{x} = -\lambda \dot{x} - \mu x + \cos(t)</code>