

目次

第5章	グラフ作成：Gnuplot	75
5.1	対話的な使い方	76
5.1.1	終了：quit	77
5.1.2	ヘルプ：help <i>topics</i>	77
5.1.3	デモ：/usr/lib/demos	77
5.2	二次元プロット：plot	77
5.3	式	78
5.3.1	演算子	79
5.3.2	組込み関数	79
5.3.3	虚数単位：i={0.0,0.1}	79
5.4	オプションの設定：set <i>option</i>	83
5.4.1	サンプル数：set samples	83
5.4.2	範囲指定：set *range	84
5.4.3	軸名：set *label	85
5.4.4	対数軸の指定：set {no}logscale	85
5.4.5	目盛数値の書式：set format	86
5.4.6	スタイル：with <i>styles</i>	86
5.4.7	その他に set できる機能	87
5.5	データ（点列）の表示	92
5.5.1	列指定子：using	93
5.6	多彩な出力形式の利用	94
5.6.1	EPS ファイルの出力	94
5.6.2	tgif の obj 形式で出力する	95
5.6.3	table 形式の利用	96
5.7	三次元プロット：splot	98
5.7.1	三次元プロットの等高線図：set contour	100

5.7.2	三次元データ点列のプロット	101
5.7.3	格子データへのマッピング: dgrid3d	102
5.8	非対話的な使い方: スクリプトで制御	103
5.9	媒介変数表示: set parametric	104
5.10	極座標表示: set polar	105
5.11	バージョン 3.6 の注目すべき新機能	106
5.11.1	データと理論曲線の一致: fit	106
5.11.2	重ね合わせ: set multiplot	109
5.12	繰返し: reread	109
5.13	Gnuplot の大技小技	111
5.14	gnuplot+	114
5.15	Emacs との連携	116
5.16	他のプロットツール	116
5.16.1	Xgraph	116
5.16.2	GNU plotutils	117
5.16.3	Yorick: An Interpreted Language	118

5

グラフ作成 : Gnuplot

Thomas Williams 氏, Colin Kelley 氏, その他多くの人々によって開発された, グラフ作成ツール Gnuplot は, 他のツールの入力形式に合わせた出力が可能です. UNIX では標準といってよい PostScript, \LaTeX はむろんのこと, 6章で紹介するドローイングツール Tgif (日本語が入力可能) の obj 形式もサポートしています. 変わり種としては, \TeX の METAFONT のソースも作成しますから, 複雑なロゴを作ってみるのも面白いかもしれません. また, 基本とも言える生の数表 (table) 形式を利用して, 三次元レンダリングの美しい画像も作成できます. 基本的には, 関数のプロットが得意ですが, もちろん実験データをファイルから読み込んで, データを加工 (四則演算や関数演算を施す) しグラフ化する機能も十分備わっています. パイプから読み込む機能もありますから, 複雑なデータ処理は awk や perl をフィルターとして用いるという, きわめて UNIX 的な使い方できます.

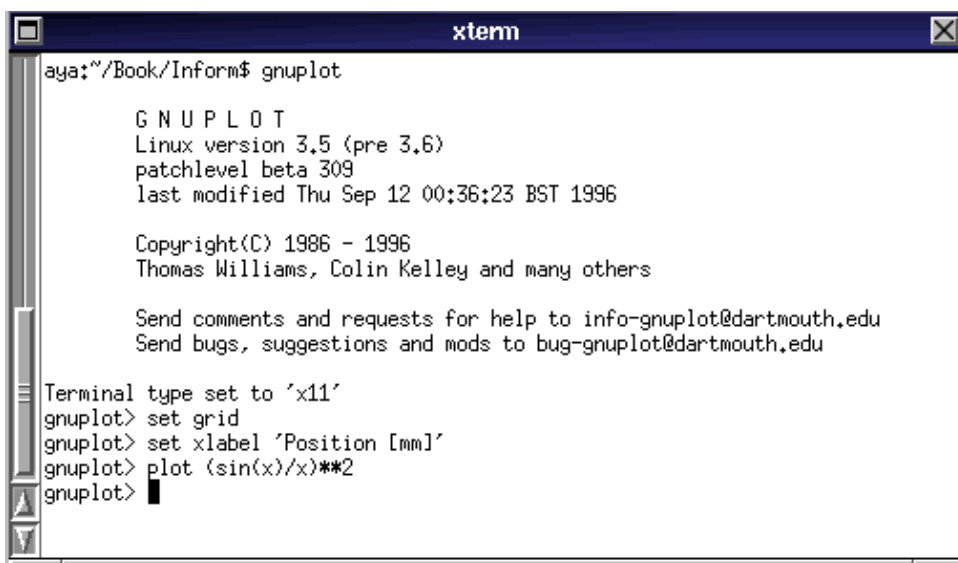
現在, 正式配布のバージョンは 3.5 ですが, 3.6 ではとても便利な機能が追加されています. 本章では基本的に 3.6 に基づいて紹介しますし, β バージョンはかなり安定しているのでぜひ入手してください.

5.1 対話的な使い方

xterm や kterm などの上で、ただ単に

```
gnuplot
```

と入力します。すると、図 5.1 のようにプロンプト “gnuplot>” を表示してコマンド入力待ちになります。このようにコマンドをキー入力して実行させ、



```
aya:~/Book/Inform$ gnuplot

  GNU PLOT
  Linux version 3.5 (pre 3.6)
  patchlevel beta 309
  last modified Thu Sep 12 00:36:23 BST 1996

  Copyright(C) 1986 - 1996
  Thomas Williams, Colin Kelley and many others

  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> set grid
gnuplot> set xlabel 'Position [mm]'
gnuplot> plot (sin(x)/x)**2
gnuplot>
```

図 5.1: gnuplot の起動画面

オプションの適否などを確認していく使い方を、一般に対話的と呼びます。ところで、ある程度オプションの設定値も決まり、同じ図面を何枚も描くだけという状況になったら、いちいちコマンドをキー入力するのは面倒です。そこで、コマンドを記述したスクリプトファイルを用意しておき、起動時にファイル名を与えて次のように実行させることもできます。

```
gnuplot scriptfile
```

この非対話的な使い方についての応用については後で述べます。

5.1.1 終了 : quit

終了したい場合には, `quit` あるいは `exit` と入力します. C-d でも終了します.

5.1.2 ヘルプ : `help topics`

Gnuplot にはとても詳しいオンラインヘルプを備えていますから (L^AT_EX のマニュアルや, `mule` の `info` ファイルと同じ内容), これを大いに活用しましょう. 疑問が漠然としている場合は,

```
gnuplot> help
```

でメニュー形式のヘルプが現れますから, 指示に従い一通り読んでみるのもよいかもしれません. また使いこなしていった, 内容を知りたいトピックがはっきりしている場合には,

```
gnuplot> help トピック
```

のように直接に指定します.

5.1.3 デモ : `/usr/lib/demos`

Gnuplot の全機能を本書で紹介することは不可能です. 詳しいデモが多分 `/usr/lib/gnuplot/demos` にありますから, 覗いてみるとよいでしょう. ルートの権限があるならば, 上記のディレクトリに移動して,

```
gnuplot all.dem
```

としてみましよう. ルートの権限がない場合には途中でデモが中断します (記録を書き込もうとするデモがあります) ので, 自分のホーム以下にディレクトリを作って, デモにあるファイルをすべてコピーして実行してください.

5.2 二次元プロット : plot

対話的に実行していきましょう. `gnuplot` を起動してプロンプトが現れたら,

```
gnuplot> plot (sin(x)/x)**2
```

と入力してみてください . 図 5.2 のように別の窓が現れてグラフが表示されます . x 軸の範囲はデフォルトでは -10 から 10 です . 軸の目盛 (tics と呼

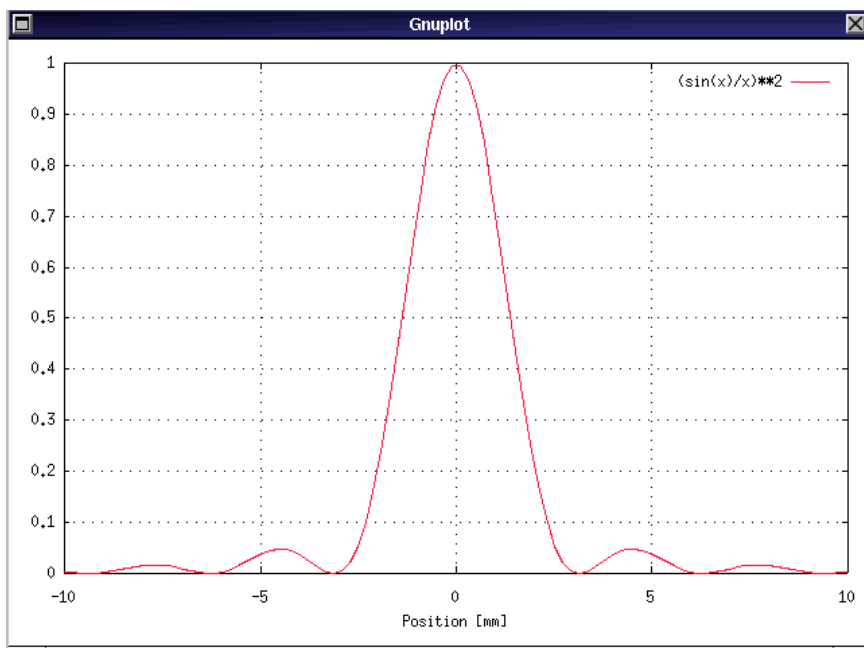


図 5.2: gnuplot は X では別窓にグラフを表示する

ばれる)も適切につけられます . もちろん範囲や目盛や軸の名前を任意に設定することもできます (後述) . y 軸の範囲は既定ではすべてが図面内に収まるように自動調整されます (autoscale) . このように , 関数 $y = f(x)$ の二次元プロットが簡単に描けます .

5.3 式

Gnuplot で扱える演算子や組み込み関数をまとめておきましょう . もちろん , これらの演算子や関数を組み合わせてユーザー関数を定義することができます .

5.3.1 演算子

演算子は被演算子 (オペランド) の数によって, 表 5.1 のように単項, 二項, 三項演算子と分類されます.

5.3.2 組込み関数

表 5.2 に Gnuplot の組込み関数を示します.



C 言語でもよくある, 初心者を戸惑わせる演算の規則に注意しましょう. それは gnuplot 上で `'print 5/2'` と `'print 5.0/2'` の結果の違いに現れます. 前者は整数同士の演算なので, 結果が整数になってしまいます『2で割る』の意味が, 有効数字無限の『2.0000...』で割る意味ならば, 実数を期待していることを明示的に示すために, 後者の表現を用いなければなりません.

ついでに Gnuplot から離れますが, 『 π の値を 50 桁まで見たい』ときにはどうしたらよいでしょうか? 答えは, 任意精度計算言語 bc です. 実行結果を次に示します.

```
gnuplont>!bc -lq
scale=50
4.0*a(1)
3.14159265358979323846264338327950288419716939937508
quit
gnuplot>
```

スクリプトの例にも出てきますが, '!' に続けてコマンドを入力すると, シェルにそのコマンドを実行させることができます.

5.3.3 虚数単位: $i=\{0.0,0.1\}$

複素定数は { 実部 (定数), 虚部 (定数) } の形式で表されます. 括弧の中に変数を入れることができません. 複素変数は, まず虚数単位を $i=\{0,1\}$ と定義してから, $x + i * y$ のように表記します. 複素数に関する重要な公式, Euler の公式を表現できるかどうか確かめましょう.

$$e^{i\theta} = \cos \theta + i \sin \theta$$

表 5.1: 演算子

演算子	例	説明
単項演算子 (unary)		
-	-a	負
+	+a	正 (何もしない)
~	~a	*ビットの反転 (例) ~ (2**32-1) = 0
!	!a	*論理否定
!	a!	*階乗
\$	\$3	*'using' 使用時にデータ列番号を指定
二項演算子 (binary)		
**	a**b	冪乗
*	a*b	乗算
/	a/b	除算
%	a%b	*剰算 (例) 5%2=1, 10%7=3
+	a+b	加算
-	a-b	減算
==	a==b	等値
!=	a!=b	非等値
<	a<b	小さい
<=	a<=b	小さいか等しい
>	a>b	大きい
>=	a>=b	大きいか等しい
&	a&b	*ビットごとの論理積 (例) 16&15=0
^	a^b	*ビットごとの排他的論理和 (例) 16^16=0, 16^15=31
	a b	*ビットごとの論理和 (例) 16 16=16, 16 15=31
&&	a&&b	*論理積
	a b	*論理和
三項演算子 (ternary)		
? :	a?b:c	a が真なら b 偽なら c を評価して結果を返す

*の付いている演算の引数は整数に限る .

このように扱えるかどうかは , 以下の関数が正常に表示できるかどうかで

表 5.2: Gnuplot の組み込み関数

関数	名前・内容	関数	名前・内容
abs(z)	絶対値	acos(z)	逆余弦
acosh(z)	逆双曲線余弦	asin(z)	逆正弦
asinh(z)	逆双曲線正弦	atan(z)	逆正接
atan2(z,z)	逆正接	atanh(z)	逆双曲線正接
besj0(x)	0 次ベッセル	besj1(x)	1 次ベッセル
besy0(x)	0 次ノイマン	besy1(x)	1 次ノイマン
ceil(rz)	z 以上の最小の整数	cos(z)	余弦
cosh(z)	双曲線余弦	erf(rz)*	(正規化) 誤差関数
erfc(rz)	1-erf(z)	exp(z)	指数
floor(rz)	z 以下の最大整数	gamma(rz)	ガンマ
ibeta(p,q,rz)	不完全ベータ	igamma(a,rz)	不完全ガンマ
imag(z)	z の虚部	int(rz)	rz の整数部
inverf(rz)**	erf(rz) の逆関数	invnorm(rz)	norm(rz) の逆関数
lgamma(rz)	対数ガンマ	log(z)	自然対数
log10(z)	常用対数	norm(rz)***	正規分布関数の累積
rand(rz)	疑似乱数発生	real(z)	z の実部
sgn(rz)	rz の符号	sin(z)	正弦
sinh(z)	双曲線正弦	sqrt(z)	平方根
tan(z)	正接	tanh(z)	双曲線正接

(注) 引数に z とある場合は複素数に対して定義されており, x とある場合は実数にのみ定義されていて z を与えるとエラーとなる. rz とある場合には複素数 z を与えてもエラーにはならないが, 実数に対して定義されており, その虚部が無視される.

* 定義は $\text{erf}(x) = \frac{\sqrt{\pi}}{2} \int_{-\infty}^x e^{-t^2} dt$

** すなわち $y = \text{erf}(x)$ とすると $x = \text{inverf}(y)$

*** 誤差関数との関係は $\text{norm}(x) = \frac{1}{2} + \frac{1}{2} \text{erf}\left(\frac{x}{\sqrt{2}}\right)$

検証できます.

$$\Re\{e^{i\theta}\} = \cos \theta, \quad \Im\{e^{i\theta}\} = \sin \theta$$

以下のように対話的に実行して, \cos , \sin 曲線が現れること確認してください.

Gnuplot での複素数の扱い : Euler 公式の表示

```
gnuplot> i={0,1.0}
gnuplot> plot [0:2*pi] real(exp(i*x)), imag(exp(i*x))
```

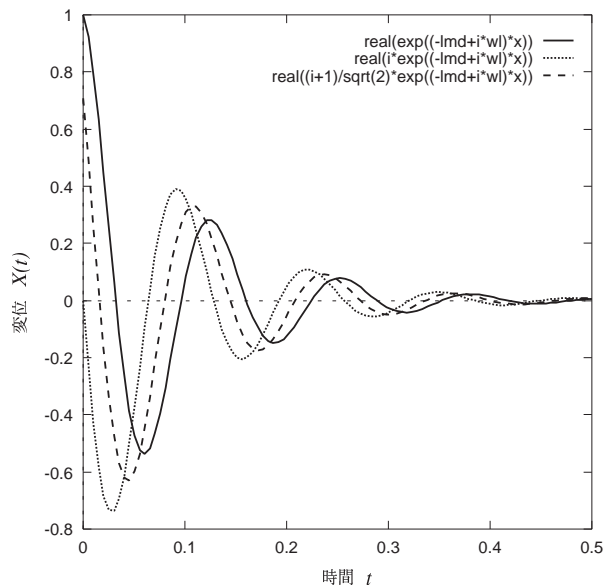


図 5.3: 減衰振動問題の複素解表示

もう少し凝った問題を考えましょう. 振動理論では解を複素数のまま扱い, 実部もしくは虚部を解として採用すると決めておくと, 計算が非常に楽になります.

質点の減衰振動の一般解は

$$X(t) = Ce^{(-\lambda + i\sqrt{\omega^2 - \lambda^2})t}$$

と書き表されますが, これをそのままグラフに描いてみましょう.

```
i={0.0,1.0}
w=50.0
```

```

lmd=10.0
w1=sqrt(w**2-lmd**2)
plot [0:0.5] real(exp((-lmd+i*w1)*x)), \
          real(i*exp((-lmd+i*w1)*x)), \
          real((i+1)/sqrt(2)*exp((-lmd+i*w1)*x))
pause -1

```

図 5.3 を見てください。どうです，減衰振動を表していますね。しかも，複素定数を変えることで様子が変わっていくことがわかります。もちろん実際には初期条件に合致するように，複素定数を決めます。

5.4 オプションの設定 : set option

自分の要求に合ったグラフを描けるように，色々なオプションを設定するコマンド `set option` が Gnuplot にはあります。頻繁に使うものについて以下に説明していきます。

5.4.1 サンプル数 : set samples *samples_1* {, *samples_2*}

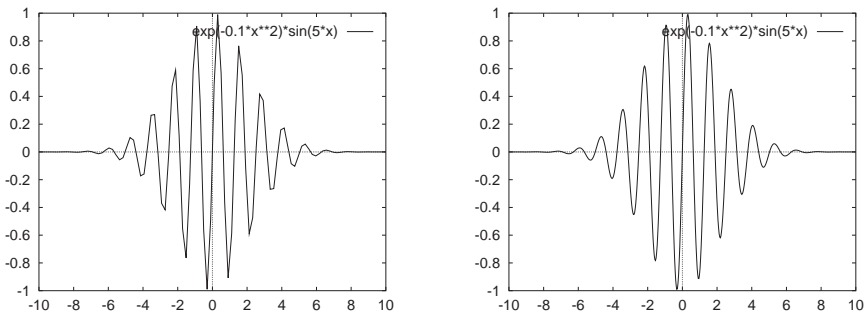


図 5.4: samples を多くすると曲線が滑らくなる

次のコマンドで描かれた図形を見てください。

```
gnuplot> exp(-0.1*x**2)*sin(5*x)
```

ギザギザです，汚らしいですね (図 5.4 左)。これは描く点の数が不足しているために，折れ線が目だっているのです。もっと点を増やしてみましょう。

```
gnuplot> set samples 600
gnuplot> replot
```

だいぶ綺麗になりました (図 5.4 右) . `replot` はもう一度同じプロットを行うコマンドで, 入力の手間が省けます . なお, `samples_2` は三次元プロット $z = f(x, y)$ に対して有効で, y 方向のサンプル数を設定します .

5.4.2 範囲指定 : set *range [min:max]

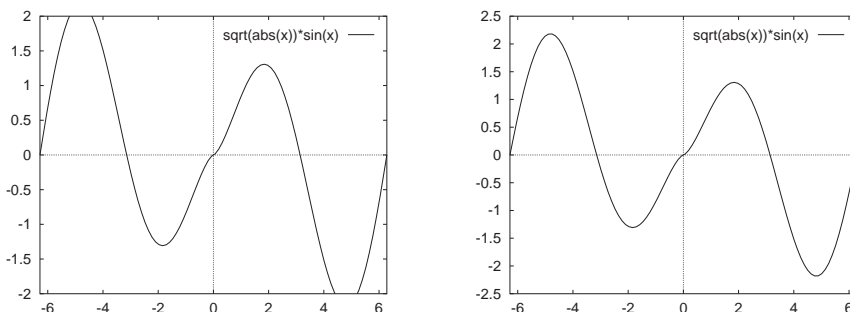


図 5.5: range 設定の適否 : `set autoscale` で適切な range が設定される .

例えば, 表示範囲は

```
set xrange [-1:1]
set yrange [-pi:2*pi]; set zrange[:0]
```

と設定できます . 以後はこの範囲で図を描きます . が, `plot` 時に範囲指定も可能です .

```
gnuplot> plot [-2*pi:2*pi][-2:2] sqrt(abs(x))*sin(2*x)
```

これだと 図 5.5 左のように上下が収まりません . 自動設定に戻したいですね .

```
set autoscale y
```

これで y 軸は 図 5.5 右のように上下が収まるように自動設定されます . `replot` して確かめてください .

5.4.3 軸名 : `set *label {"label"}{{xoff},{yoff}}{"font"}`

ラベルと位置を指定して軸に名前をつけます。標準位置は、 x 軸（横軸）では『軸の下中央そろえ』です。 y 軸は、出力形式により異なっていて、例えば X11 では軸の上側左隅となり、ちょっと見苦しいですが、`tgif` 形式では、文字が 90 度回転して軸の左側に位置し、理工学系論文の図の体裁に合致します。図 5.1 で

```
gnuplot> set xlabel 'Position [mm]'
```

を行った結果が、図 5.2 に現れています。オフセットなどの感触は、以下のよう実行して確かめてみてください。

```
gnuplot> set xlabel 'Position [mm]'  
gnuplot> plot besj0(x)  
gnuplot> set xlabel 'Position [mm]' 10,-0.5  
gnuplot> replot
```

フォントは X11 では変化しません。`tgif` や EPS 形式の場合に有効ですから、そこで確かめてください。バージョン 3.6 では 2 番目の軸 `x2label` と `y2label` が使えます。

5.4.4 対数軸の指定 : `set {no}logscale x,y,z,xy`

数桁にわたる広い変数範囲上でプロットする場合には対数軸が用いられます。そこで、 x, y, z 軸を対数にするかどうか指定します。理工学系では必須の機能でしょう。線形軸が既定ですが、`nologscale` と明示するほうが間違いがないかもしれません。

```
gnuplot> plot [0.01:100] exp(-0.05*x)  
gnuplot> set logscale y  
gnuplot> replot  
gnuplot> set logscale xy  
gnuplot> replot  
gnuplot> set nologscale xy  
gnuplot> replot
```

5.4.5 目盛数値の書式: `set format {axes}{"format-string"}`

軸 ($axes=x, y, z, xy, x2, y2$) の目盛には数値が振られます。目盛自身は '`set *tics ...`' で指定しますが、数値の書式は、C 言語に準じて指定できます。既定は `%g` です。表 5.3 に載せた以外にも、数値に関しては `%x %o %t %l %s %T %L %S %c %P` が使えます。日時を示すモードでは、日時に多くのフォーマットが用意されています。詳しくは `gnuplot` 上で "`gnuplot> help format`" としてオンラインヘルプを見てください。

表 5.3: 目盛の数値の書式指定

書式指定	意味	表示例
<code>"%e"</code>	浮動小数点形式	1.0E-12, 3.45E+6
<code>"%.3e"</code>	小数点以下 3 桁表示	1.002E-12, 3.458E+6
<code>"%f"</code>	固定小数点形式	1.12345, 0.00004
<code>"%.2f"</code>	小数点以下 2 桁表示	1.12, 0.00
<code>"%g"</code>	<code>%f</code> と <code>%e</code> で短く表記可能なほうで表示する	

5.4.6 スタイル : `with styles`

グラフの形式・記号・線種等を指定します。バージョン 3.6 でオプションが増えて、線の太さやマークの大きさを一本ごとに指定できるようになりました。バージョン 3.5 では線の太さやマークの大きさは一括指定しかできなかったのです。例を示しますが、引数の順番を守って入力してください。

```
gnuplot> plot sin(x) with linespoints lt 2 lw 1 pt 3 ps 2
```

`with` の直後でプロットの種類を指定します (図 5.6)。理工学系で使われる主なものは、`lines`, `points`, `linespoints`, `impulses`, `boxes`, `yerrorbars`, `vector` です。続いて線種 (`lt`)、線の太さ (`lw`)、記号の種類 (`pt`)、記号の大きさ (`ps`) の順に指定できます。

`lt` や `pt` で指定できる種類は、出力形式によって異なりますが、バージョン 3.6 では `Tgif` と `PS` では統一されて 64 種類になりました (図 5.7)。ただし、残念ですが、X11 上では 8 種類 (しかも番号が異なる) しか確認できません。X11 での種類を確認したければ、

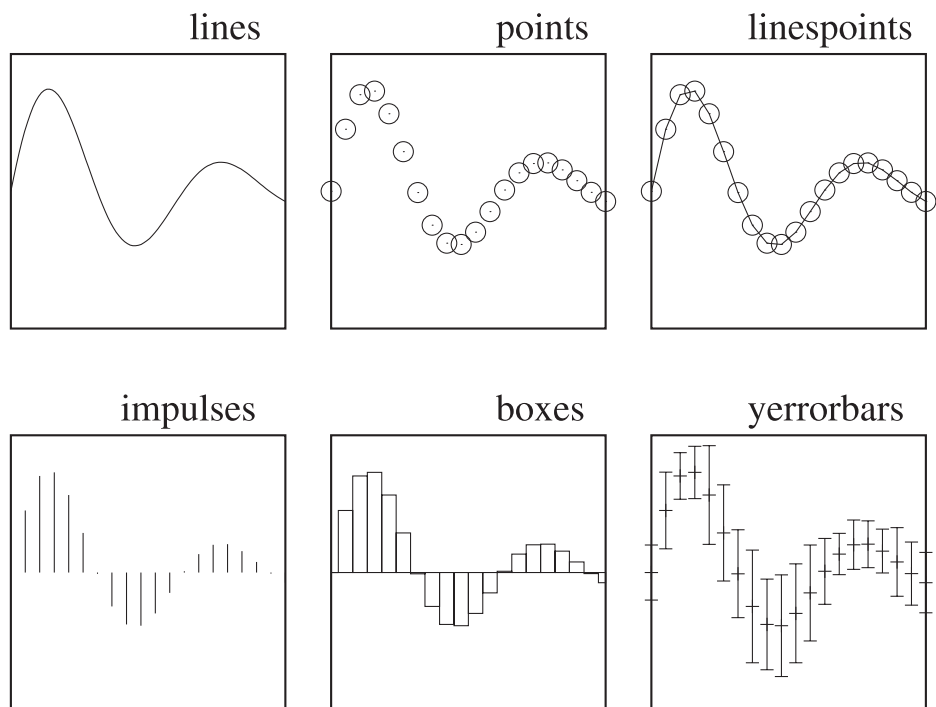


図 5.6: プロットの種類 : with の直後で指定する .

```
gnuplot> test
```

を実行してみてください .

バージョン 3.5 では線の太さや記号の大きさを 1 本ごとに指定はできません . 色番号と記号 (と線) 番号のみを以下のように指定します .

```
gnuplot> plot sin(x) with linespoints 2 2
```

5.4.7 その他に set できる機能

まず set 可能な機能や変数をすべて見てみましょう (表 5.4) .

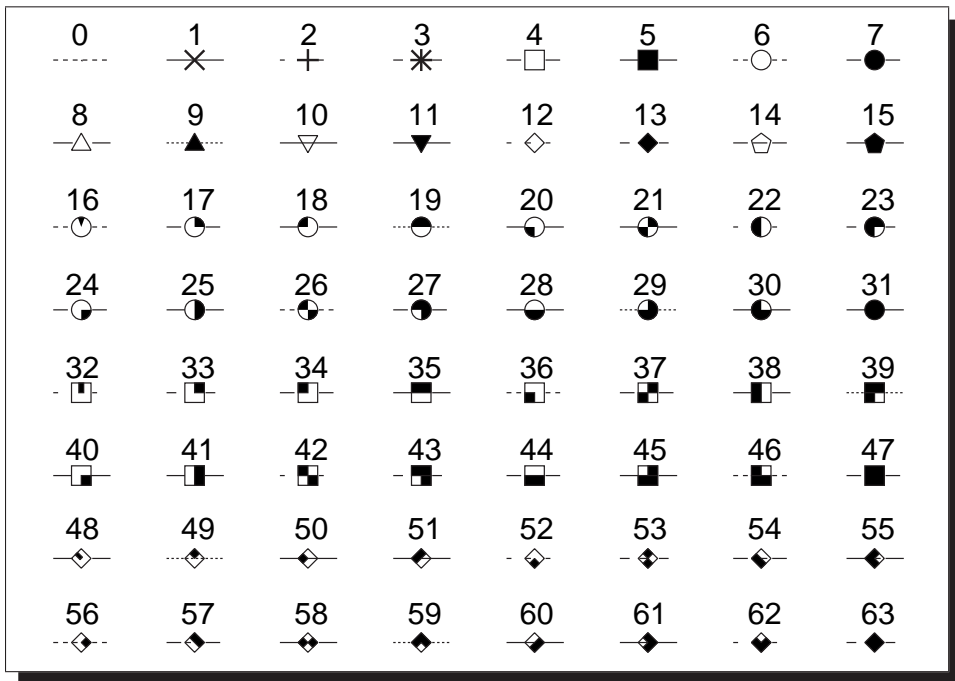


図 5.7: 64 個の線種と記号 : Tgif と PS で共通 , X11 では 8 種類のみ

```
gnuplot> show all
```

と入力しますと、いやー沢山ありますね。

表 5.4: set 可能なオプション一覧表示

```

G N U P L O T
Linux version 3.5 (pre 3.6)
patchlevel beta 338 (+1.1.3 1997/11/21)
last modified Thu Jul 24 00:12:49 BST 1997

Copyright(C) 1986 - 1993, 1997
Thomas Williams, Colin Kelley and many others

Send comments and requests for help to info-gnuplot@dartmouth.edu
Send bugs, suggestions and mods to bug-gnuplot@dartmouth.edu
autoscaling is x: ON, y: ON, z: ON
errorbars are plotted with bars of size 1.000000
border is drawn 31
boxwidth is auto
point clip is OFF
drawing and clipping lines between inrange and outrange points
not drawing lines between two outrange points
contour for surfaces are not drawn
data grid3d is disabled
mapping for 3-d data is cartesian
dummy variables are "x" and "y"
tic format is x-axis: "%g", y-axis: "%g", z-axis: "%g", x2-axis: "%g", \
y2-axis: "%g"

data are plotted with points
functions are plotted with lines
grid is OFF
xzeroaxis is OFF
x2zeroaxis is OFF
yzeroaxis is OFF
y2zeroaxis is OFF
keytitle is ""
key is ON, position: top right corner
key is right justified, not reversed and not boxed
sample length is 4 characters
vertical spacing is 1 characters
width adjustment is 0 characters
key title is ""
no logscaling
offsets are 0, 0, 0, 0
lmargin is computed automatically
bmargin is computed automatically
rmargin is computed automatically
tmargin is computed automatically
output is sent to STDOUT
parametric is OFF

```

```
pointsize is 1
encoding is default
polar is OFF
Angles are in radians
sampling rate is 100, 100
iso sampling rate is 10, 10
view is 60 rot_x, 30 rot_z, 1 scale, 1 scale_z
surface is drawn
hidden surface is drawn
size is scaled by 1,1
No attempt to control aspect ratio
origin is set to 0,0
terminal type is x11
tics are IN, ticslevel is 0.5
ticscale is 1 and miniticscale is 0.5
x-axis tic labelling is on border, mirrored on opposite border,
labels are not rotated
computed automatically
second x-axis tic labelling is OFF
y-axis tic labelling is on border, mirrored on opposite border,
labels are not rotated
computed automatically
second y-axis tic labelling is OFF
z-axis tic labelling is on border, labels are not rotated
computed automatically
minor xtics are computed automatically for log scales
minor ytics are computed automatically for log scales
minor ztics are computed automatically for log scales
minor x2tics are computed automatically for log scales
minor y2tics are computed automatically for log scales
time is "", offset at 0.000000, 0.000000
set xrange [* : *] noreverse nowriteback # (currently [-10:10] )
set yrange [* : *] noreverse nowriteback # (currently [-10:10] )
set x2range [* : *] noreverse nowriteback # (currently [-10:10] )
set y2range [* : *] noreverse nowriteback # (currently [-10:10] )
set zrange [* : *] noreverse nowriteback # (currently [-10:10] )
title is "", offset at 0.000000, 0.000000
xlabel is "", offset at 0.000000, 0.000000
ylabel is "", offset at 0.000000, 0.000000
zlabel is "", offset at 0.000000, 0.000000
x2label is "", offset at 0.000000, 0.000000
y2label is "", offset at 0.000000, 0.000000
xdata is set to numerical
ydata is set to numerical
x2data is set to numerical
y2data is set to numerical
zdata is set to numerical
```

```
read format for time is "%d/%m/%y\n%H:%M"
locale is "C"
zero is 1e-08
No string is interpreted as missing data
last plot command was:
```

```
Variables:
pi = 3.14159265358979
```

```
User-Defined Functions:
```

前にも言いましたが項目を `help` で調べることができます。いくつかについて補足します。

`(no)border` グラフの周りの枠を描くかどうか。

`(no)grid tic` の位置に、格子線を描くかどうか。極座標系を選択したり、線の種の指定が可能です。

`(no)key` 各プロットごとの名前を表示するかどうか。バージョン 3.6 では位置指定も可能です。

`angles degrees | radians` 極座標系での角度の単位の設定。既定は `degree` (度) なので、`radian` に変えたい場合は `set angles radian` とします。

`view` 三次元プロットの場合の視点方向の設定。四つの値； $\langle x$ 軸回りの回転角度 \rangle 、 $\langle z$ 軸回りの回転角度 \rangle 、 \langle 全体の拡大率 \rangle 、 $\langle z$ 軸方向のみの拡大率 \rangle を指定します。例えば `set view 0,0,` とすると、 xy 平面を z 軸から見る (二次元プロットの視点と同じ) ことになります。また `set view 90,0,` とすると、 xz 平面を y 軸の負の側から見ることになります。

`size` 水平方向と垂直方向の拡大率を設定します。ただし、軸名を含んだ大きさなので、枠の大きさを厳密に設定することは困難です。正方形にした場合には、`set square` を指定しましょう。ほぼ正方形になります。

`*tics` 目盛の付け方を設定します。第 1 の形式では `set xtics <初期値>,<間隔>,<終値>` ” と記述します。例えば `set xtics 0,0.2,1.0` では

0, 0.2, 0.4, 0.6, 0.8, 1.0 に目盛がふられます．第 2 の形式では “ytics (“名前”<位置>,”名前”<位置>, …)” と記述します．例えば “set ytics (“low” 0, “medium” 3.5, “high” 5.0)” や位置のみの “set xtics (1, 2, 4, 7, 10, 20, 40, 70, 100)” などです．

5.5 データ（点列）の表示

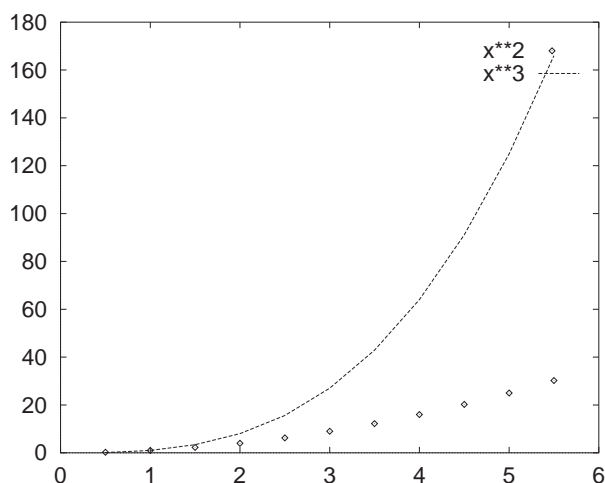


図 5.8: データ（点列）ファイルのグラフ化の例

次のように三つの値が 1 行ごとに並んだデータファイル 'sqx.dat' があつたとします．

```
0 0 0
0.5 0.25 0.125
1.0 1.00 1.000
...
5.0 25.00 125.000
5.5 30.25 166.375
```

これを Gnuplot で表示してみましょう．対話モードで起動して，plot を実行します．

```
gnuplot> plot 'sqx.dat'
```

何もオプション指定しなければデータファイルの第1列を x の値, 第2列を y の値として, マーク (point) で描きます。では, 第1列と第3列を x, y の値とした折れ線 (line) を描くにはどうしたらよいでしょうか。答えは, 列指定子 “*using i:j*” を用いればよいのです。折れ線はスタイルで指定します (図 5.8)。

```
gnuplot> plot 'sqx.dat' title 'x**2', 'sqx.dat' using 1:3
        title 'x**3' with line
```



データファイル 'sqx.dat' は x, x^2, x^3 の並びです。これはどのように作りますか? mule で電卓片手に打ち込むのは悲しいことです。

```
for(x=0;x<6;x+=0.5)print x,"\t",x*x,"\t",x*x*x,"\n"
quit
```

という内容のスクリプト 'sqx.bc' を作り, 任意精度計算言語 bc で “*bc -q -l sqx.bc > sqx.dat*” と実行すればできますね。まだ大袈裟だ, もっと手軽にしたいというならば, awk や perl あるいは ruby などのスクリプト系言語の出番です。最古参の awk を使う例を示します。

```
awk 'BEGIN{OFS="\t"; for(x=0;x<6;x+=0.5) print x,x^2,x^3}'
    > sqx.dat
```

5.5.1 列指定子 : using

前項で説明した通り, 列指定子を用いてファイルのどの列のデータをプロットに使うのか設定できます。さらに, それらの演算結果を使うことも可能です。その場合にはファイルの第 n 列のデータは ' $\$n$ ' で表し, 演算式全体を括弧で囲みます。'sqx.dat' を使って, 次のスクリプトを実行してみてください。

```
plot 'sqx.dat' using 1:3 w l, 'sqx.dat' using 1:($2**1.5)
```

また Gnuplot では, 区切り文字として空白またはタブしか認識しませんから, ファイルのデータがそれら以外の文字で区切られて詰まっている場合, つまり

```
1.0,2.0e-3;2400,...
2.0,5.6e-3;5700,...
....
```

のようなデータは使えません．そこで，`using` のオプションでデータの中の区切り文字を明示した書式を指定して認識させることが可能です．上記のデータに対しては，

```
plot '***' using 1:3 "%lf,%lf;%lf"
```

とすれば，1列目と3列目を x, y 座標にした二次元プロットが描かれます．

5.6 多彩な出力形式の利用

Gnuplot は非常に多くのプリンター言語やアプリケーション独自の記録形式に対応してファイルの出力を行います．形式の指定は “`set terminal ***`” で，出力先の指定は “`set output ***`” で行います．ここでは， \LaTeX で取り込める EPS 形式，Tgif の `obj` 形式，汎用性の高い `table` 形式を説明します．

5.6.1 EPS ファイルの出力

EPS 形式でファイルに出力しましょう．

— EPS 形式で出力する方法

```
gnuplot> set term postscript eps 22
gnuplot> set output 'sinxx.eps'
gnuplot> replot
```

“`set term postscript eps ...`” により出力ターミナル形式を EPS に変えます．“`set output 'filename.eps'`” で出力先をファイル `filename.eps` にします．そして，同じプロットを再度出力するのは “`replot`” で行います．この EPS ファイルを `epsbox.sty` で取り込んだものが 図 5.9 です．

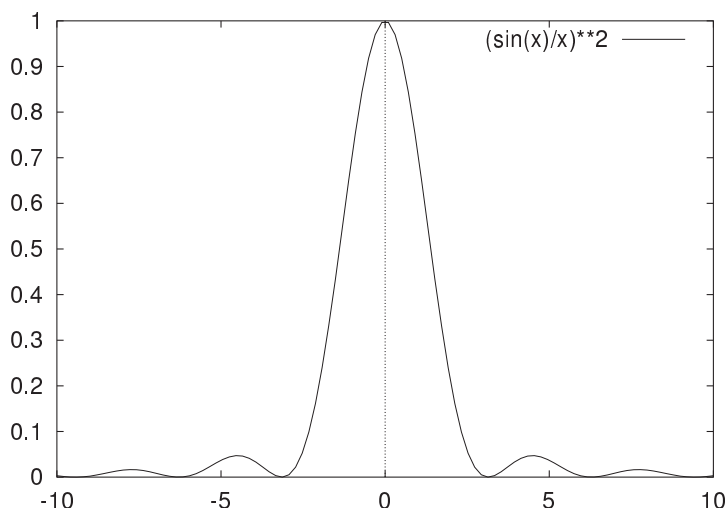


図 5.9: EPS 形式で保存したものを取り込んだ場合

5.6.2 tgif の obj 形式で出力する

tgif で読み込んで編集できる形式を指定して出力しましょう。EPS の場合と同様に `term` を `tgif` に `set` して、`output` のファイル名前を指定します。

————— tgif 形式で出力する方法

```
gnuplot> set term tgif
gnuplot> set output 'filename.obj'
```

強制振動における、振動数と定常振幅の関係曲線を例に取りましょう。自由な場合の固有振動数が ω 、粘性抵抗による減衰率が λ である振動系に振動数 p の外力を加えて強制振動を起こさせた場合、 $\omega^2 - \lambda^2 > 0$ の場合には定常状態における振幅 $C(p)$ は

$$C(p) = \frac{l}{\sqrt{(\omega^2 - p^2)^2 + 4\lambda^2 p^2}}$$

と表されます (l は定数)。 ω で正規化して改めて、 $x = p/\omega$ とおいて、 x 依

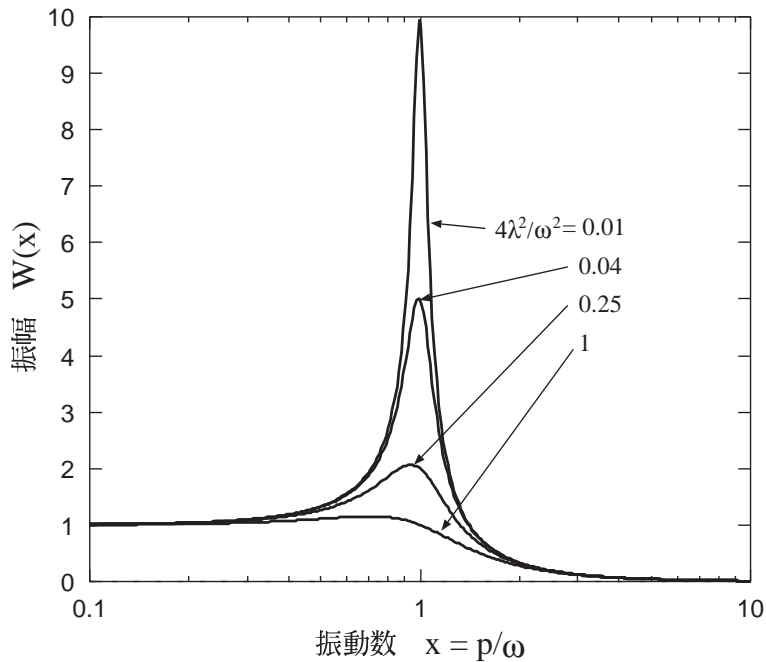


図 5.10: Gnuplot Tgif の組み合わせで作成した例:日本語が可能

存性部分だけを取り出すと

$$W(x) = \frac{1}{\sqrt{(1-x^2)^2 + \frac{4\lambda^2}{\omega^2}x^2}}$$

と書かれますから, $W(x)$ を $-1 < \log_{10} x < 1$ の範囲で, パラメータ $4\lambda^2/\omega^2$ が $1, (0.5)^2, (0.2)^2, (0.1)^2$ の場合について Gnuplot でグラフを描き, tgif 上で手を加えて 図 5.10 のような仕上がりが得られます.

5.6.3 table 形式の利用

Gnuplot の table 形式出力はとても素直な数値の並びです. $y = f(x)$ の二次元プロットのデータは,


```
#Curve 0, 100 points
#x y type
x0 y0 i
x1 y1 i
...
xk yk i
...
```

のように各値が空白で区切られた並びで出力されます (これはバージョン 3.6 の形式です . 3.5 ではこれと異なる並びとなりますので , ご注意ください) . したがって , awk などのスクリプト言語で簡単に加工できます . 例えば

table 形式の利用例

```
gnuplot> set term table
gnuplot> set output "|awk '{print $2,$1}' "
gnuplot> plot x**2
```

と出力先をパイプで awk に接続すれば , awk がデータの出力順を逆転して表示します .

立体画像の作成 : gnuconv

table 形式のデータを基に陰影をつけた立体的な画像を作成するツール , gnuconv を Carsten Hammer 氏 (chammer@POST.uni-bielefeld.de) が作成しました . 氏は README に “とても完璧とはほど遠い” と記していますが , 謙遜でしょう . Jonas Yngvesson (jonas-y@isy.liu.se) , Inge Wallin (ingwa@isy.liu.se) 両氏の画像ライブラリ SIPP (SImple Polygon Processor) を使っていますが , これも十分実用になるものです .

では , 図 5.11 を見てください . これは Gnuplot のデモにある媒介変数表示で描いた『貝』を , 次のような手続きで , gnuconv に少しだけ筆者が手を入れた改造版 gnuconvf で立体画像化したものです .

```
set parametric
set samples 39
set isosamples 39,78
set size 0.7,1
```

```

set view 15,170,1,2
set title "Parametric Shell"
set view 50,30,1.0
set urange [0:2*pi]
set vrange [0:2*pi]
plot cos(u)*u*(1+cos(v)/2),sin(v)*u/2,sin(u)*u*(1+cos(v)/2)
set output 'sphere.table'
set term table
replot
!gnuconvf -i sphere.table -s 256
!cjpeg -quality 30 -outfile paramet1.jpg /tmp/out.ppm
!xli paramet1.jpg

```

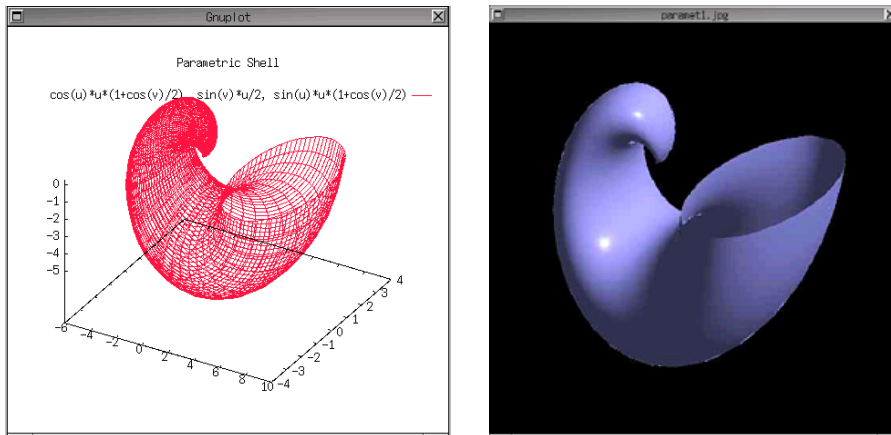


図 5.11: gnuconv による『貝』の立体画像

5.7 三次元プロット : splot

関数 $z = f(x, y)$ の三次元プロットは `splot` コマンドで描きます。

```
gnuplot> set isosample 29
gnuplot> set hidden3d
gnuplot> plot [-pi:pi][-2:2] sin(x)**2*exp(-y**2)
gnuplot> set view ,0,, ;replot
gnuplot> set view ,15,, ;replot
gnuplot> set view ,30,, ;replot
gnuplot> set view ,45,, ;replot
gnuplot> set view ,60,, ;replot
```

を実行してみてください。網目状に点を結んだ図が描かれます(図 5.12)。“set isosample n ” は xy 面での格子の細かさを指定します(すなわち, $n \times n$ の点で計算することになります)。“set hidden3d” は隠線処理を行う意味です。後半では set view コマンドで視点を変えて replot で再描画しています。 z 軸回りに図が回転していくことを確認してください。なお, 視点の既定値は 60,30,1,1 です。

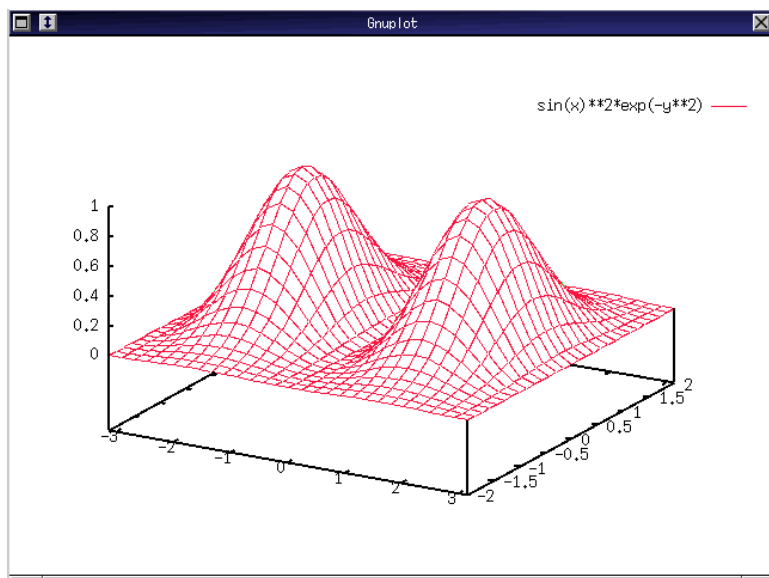


図 5.12: split による $\sin^2 x \exp(-y^2)$ の三次元プロット

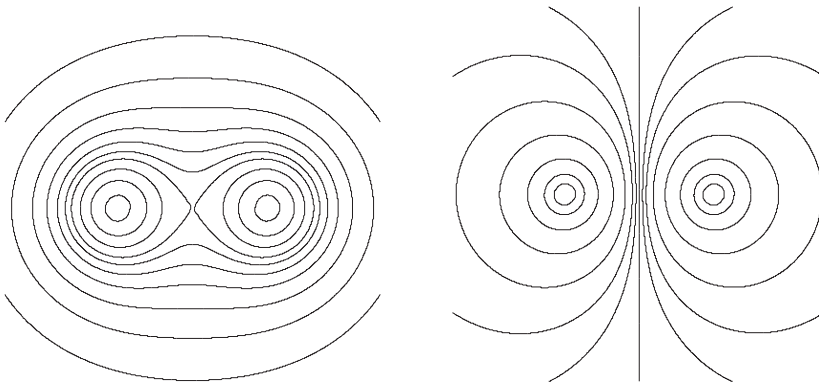


図 5.13: 大きさが等しい二つの点電荷による電場の等電位線図

5.7.1 三次元プロットの等高線図：set contour

splot では、地図のように等高線を描くことができます。ここでは、電磁気学で習う二つの点電荷による電場の等電位線図を描いてみましょう(図 5.13)。

```

set nokey
set view 0, 0, 1, 1
set samples 100, 100
set isosamples 60, 60
set nosurface
set contour base
set cntrparam bspline
set cntrparam order 5
set cntrparam points 7
set size 1,1.05
set xrange [-5 : 5]
set yrange [-5 : 5]
set output 'potent1.obj'
set cntrparam levels discrete 0.4,0.5,0.6,0.7,0.8,0.9,1.0,\
                            1.2,1.6,3.2
splot 1.0/sqrt((x-2.0)**2+y**2)+1.0/sqrt((x+2.0)**2+y**2)
set output 'potent2.obj'
set cntrparam levels discrete -3.2,-1.6,-0.8,-0.4,-0.2,-0.1,\
                            -0.05,0,0.05,0.1,0.2,0.4,0.8,1.6,3.2
splot 1.0/sqrt((x-2.0)**2+y**2)-1.0/sqrt((x+2.0)**2+y**2)

```

必要な設定は“set contour”です。この例題では表面形状は描かないので“set nosurface”とします。また、等高線を引く高さは“set cntrparam ...

” で設定します . 滑らかな曲線を得るために , `bspline` 関数を使用しています .

5.7.2 三次元データ点列のプロット

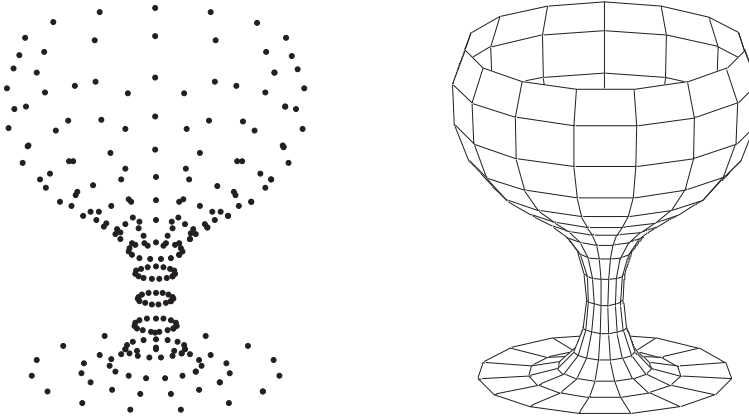


図 5.14: 三次元点データ `glass.dat` の点と曲線プロット

点 (x, y, z) の集まりを `splot` で三次元プロットすることができます . しかし , 点のみでは三次元形状が把握できないでしょう . 網目状に点を結んだ図にしたいものです . それには “`splot 'datafile' w lines`” と `style` を指定すればよいのですが , 三次元の場合とは異なって , 網目で描くためには , 一つの曲線に属する点の数をそろえないといけません . 例えば , デモに含まれるワイングラスの形状データを “`less /usr/lib/gnuplot/demos/glass.dat`” で見てください . 空行で区切られた 1 塊のデータには 16 点が属する , 16×16 のデータになっています . もちろん 16×32 でもよいのですが , とにかく等しい数の点データを含んでいる必要があります . 図 5.14 は

```
splot "/usr/lib/gnuplot/demos/glass.dat"  
splot "/usr/lib/gnuplot/demos/glass.dat" with lines
```

により点・網目曲線でワイングラスを描いたものです .

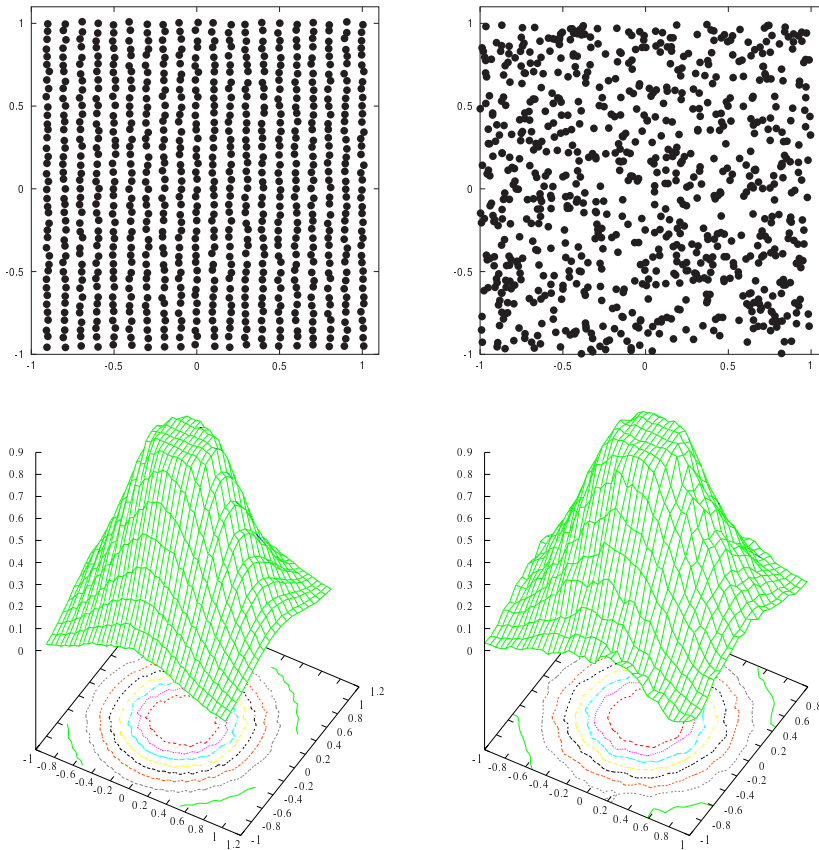


図 5.15: dgrid3d による三次元プロットの再現性．左上：少し揺らいだ規則格子に並んだデータ，右上：乱雑に並んだデータ，左下：規則格子に並んだデータを用いた山と等高線，右下：乱雑に並んだデータを基に dgrid3d で再現した山と等高線．

5.7.3 格子データへのマッピング：dgrid3d

splot ではデータは格子点上にあるものとして図を描きますが，実際には必ずしも格子点上のデータばかりではありません．dgrid3d を設定すると，データとして与えられた任意の点 (x, y, z) すべてが一つの曲面上にあるものと解釈してその曲面を描きます（図 5.15）．その際，補間により規則格子上のデータを作成して，そのデータに基づき splot を実行します．もちろんこの格子

データは “set table” により数表として出力することもできますから , 別のツール (gnuconv) などの入力データとして利用可能となります .

5.8 非対話的な使い方 : スクリプトで制御

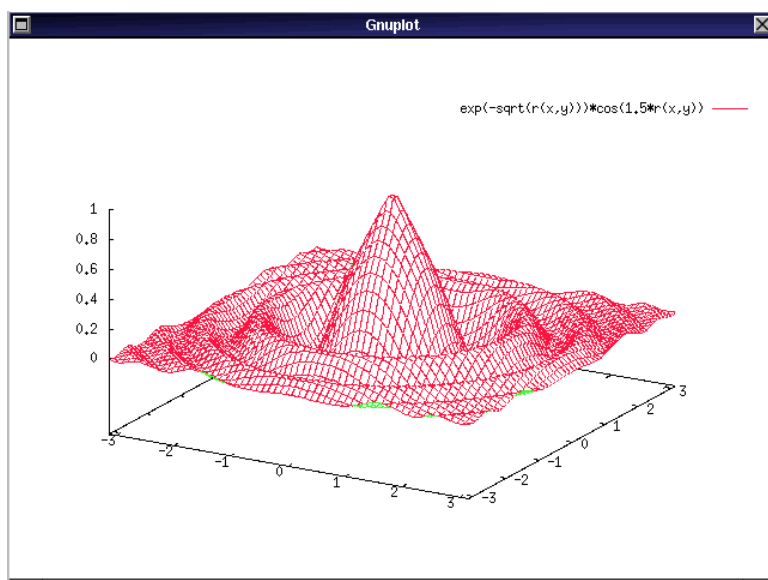


図 5.16: 非対話モードで行った 3 次元プロットの例

Gnuplot のコマンドを並べたファイルを読み込ませて , 仕事をさせることができます . 次の内容で expcos.gp を作成してください .

```
set isosample 60
set hidden3d
r(x,y)=(x**2+y**2)
set zrange[0:1]
splot [-pi:pi] [-pi:pi] exp(-sqrt(r(x,y))) * cos(1.5*r(x,y))
pause -1
```

このスクリプトを Gnuplot に実行させます .

```
gnuplot expcos.gp
```

すると少し時間がかかるかもしれませんが、図 5.16 のように別窓が開かれ、`splot` により関数 $z = f(x, y)$ の俯瞰図が表示されます。“`pause s`” は s 秒表示して終了するという意味です。 $s = -1$ では時間が無限になります、この場合 `Enter` が押されると終了となります。

5.9 媒介変数表示 : set parametric

関数関係を $y = f(x), z = g(x, y)$ のように表すことができない場合でも、媒介変数を用いて、二次元では $y = y(t), x = x(t)$ の形で、三次元では $x = x(u, v), y = y(u, v), z = z(u, v)$ の形で表すことが可能となる場合があります。最も簡単な例は円や球の表示でしょう。

媒介変数の例 : 円や球の表示

```
gnuplot> set parametric
gnuplot> set size 0.721,1.0
gnuplot> plot [-pi:pi] cos(t), sin(t)
gnuplot> set view ,,,2
gnuplot> set urange[0:pi]
gnuplot> set vrange[0:2*pi]
gnuplot> set isosample 40
gnuplot> splot sin(u)*cos(v), sin(u)*sin(v), cos(u)
```

物理の教科書にあるリサージュ (Lissajous) 図形を描いてみましょう (図 5.17)。一般形は $x = A \cos(\omega_1 t + \alpha)$, $y = B \sin(\omega_2 t + \beta)$ です。以下に示すのは物理の教科書にある $A = B$, $\omega_1 : \omega_2 = 2 : 3$, $\alpha = 0$ の場合で、位相差 $\beta - \alpha$ を変化させています。

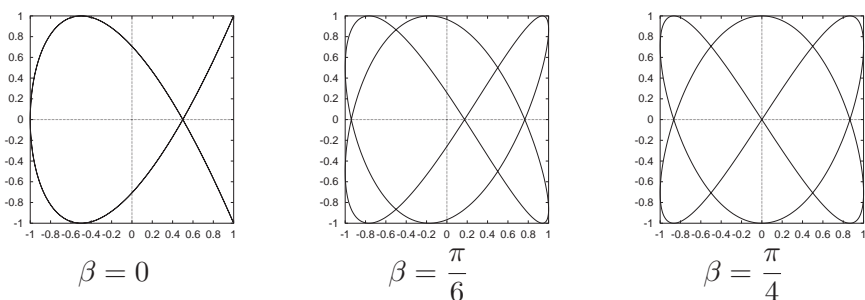
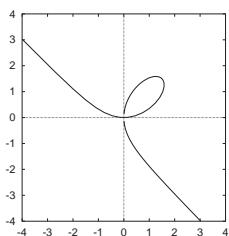


図 5.17: 媒介変数表示 : リサージュ図形の作図例

描かれた形のもっと面白い例を挙げましょう (図 5.18) . 岩波全書の “数学公式 I” に載っているものです .

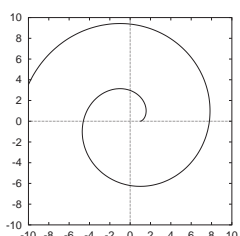
デカルトの葉形
(Folium of Descartes)

$$x = \frac{3at}{1+t^3} \quad y = \frac{3at^2}{1+t^3}$$



円周の伸開線
(involute)

$$x = a(\cos \theta + \theta \sin \theta) \\ y = a(\sin \theta - \theta \cos \theta)$$



サイクロイド
(cycloid)

$$x = a(\theta - \sin \theta) \\ y = a(1 - \cos \theta)$$

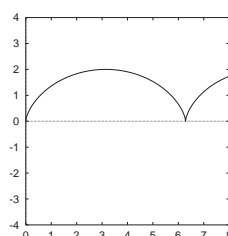


図 5.18: 媒介変数で表される著名な曲線

5.10 極座標表示 : set polar

二次元極座標では原点からの距離 r と偏角 θ の間の関数関係式 : $r = f(\theta)$ で図形を表示します (図 5.19) . 半径 R の円は非常に簡単に $r = R$ と表されます . ただし , Gnuplot では偏角の変数名は x なので間違えないください .

“数学公式 I” に載っているものを描いてみましょう (図 5.20) . まずは , 図 5.19 の “四葉” のように一般形式 $r = a \sin n\theta$ ($a > 0$) と書かれる “正葉線” です .

```

set term tgif
set output 'sin2x.obj'
set polar
set size 0.721, 0.615
set noborder
set noxtics
set noytics
plot sin(2*x)

```

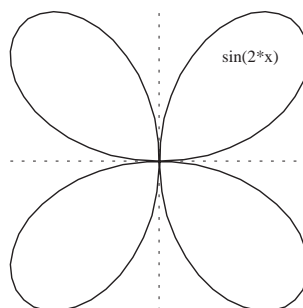
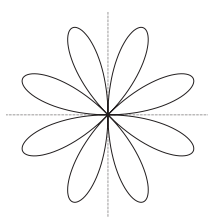
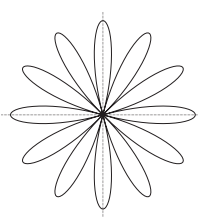


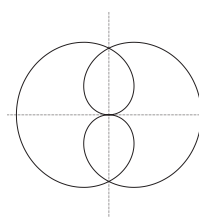
図 5.19: 極座標表示 : $r(\theta) = \sin 2\theta$ のスクリプトとプロット



$n = 4$



$n = 6$



$n = \frac{1}{2}$

図 5.20: 正葉線 $r = a \sin n\theta$ ($a > 0$)

5.11 バージョン 3.6 の注目すべき新機能

バージョン 3.5 以降に次々と改良が施されてバージョン 3.6 はまだ β ですが、マイナー番号が現在 (1998 年 5 月) 443 にもなっています。最近では gif や png の画像フォーマットでの出力に手を染めたりして、正式リリースがいつになるやら、またどのような新機能が盛り込まれるのか予測できません。本書ではすでに 3.6 の新機能を使って説明を行っていますが、ここであらためて、バージョン 3.6 の特に注目したい素晴らしい機能をまとめて紹介します。

5.11.1 データと理論曲線の一致 : fit

パラメータを含む関数 $f(x; c_1, c_2, c_3, \dots)$ として理論式が与えられている場合を考えます。実験データが理論式に従うときには、適当なパラメータ c_i が

決定されなければなりません。このパラメータ推定には一般に最小二乗法が用いられますが、特別な場合（線形多項式）を除き、自分で解析プログラムを組むのは困難です。Gnuplot では非線形なものまで含んで、このパラメータ推定を行って、理論式とデータの一致を測ってくれるコマンドがバージョン 3.6 から取り入れられました。書式は

— fit の書式 —

```
fit function "datafile" via parameter list
```

です。論より証拠，例題で理解してください。物質の平衡蒸気圧 p [Pa] と温度 T [K] の関係は，

$$\log_{10} p = -\frac{A}{T} + B + C \log_{10} T + DT$$

と理論的に判明しています（実は， C, D は調整パラメータで理論式とはいいいがたいのですが）。理科年表で水銀の平衡蒸気圧を調べて，このパラメータを決定しましょう。水銀の絶対温度 T と蒸気圧の常用対数 $\log_{10} p$ のデータ

```
231 -3.8761
248 -2.8761
267 -1.8761
290 -0.8761
319 0.1239
355 1.1239
398 2.1239
```

をファイル名 "Hg.dat" で準備します。そこで，次のスクリプトを実行しましょう。

— 理論式とデータの一致 : fit の使用例 —

```
f(x)=-A/x+B+C*log10(x)+D*x
A=3000
B=20
C=-1
D=0.001
fit f(x) "Hg.dat" via A,B,C,D
plot "Hg.dat", f(x)
pause -1
```

fit コマンドの後、しばらくパラメータ行列の計算経過が表示され、収束結果が表示されます。なお、全計算経過は fit.log に残りますので、途中経過が確認できます。図 5.21 は tgif への出力をスクリプトに指示して加工したものです。

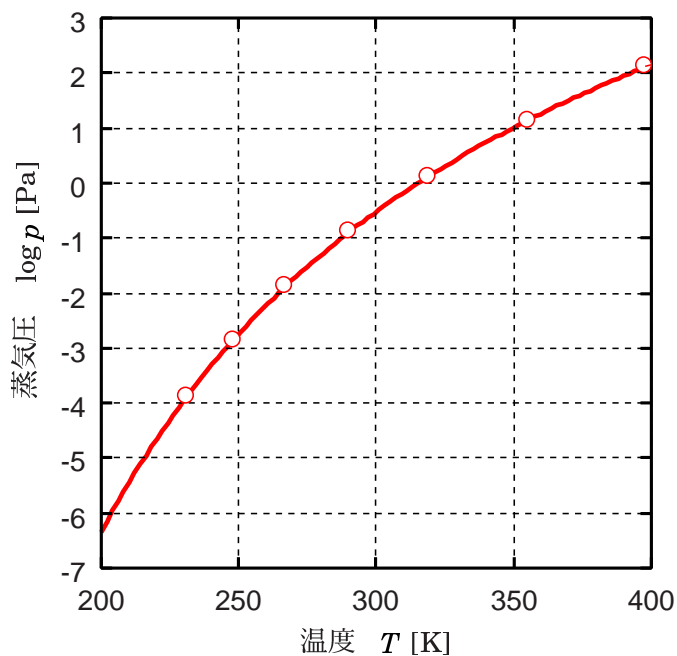


図 5.21: fitting で求めた水銀の蒸気圧曲線

5.11.2 重ね合わせ : set multiplot

バージョン 3.6 では複数のグラフを同時に描く機能 `multiplot` も取り入れられました . 次のスクリプトは z 軸回りに視点を回転させて眺めた図形を同じ画面上に描いたものです (図 5.22) .

```

set nokey; set grid; set isosamples 19,19; set hidden3d
set nograd; set xtics -6, 2, 2; set noytics; set noztics
set ticslevel 0.0
k=0.2; c=pi; r(x,y)=sqrt(x**2+y**2)
set size 1.0,1.0; set origin 0.0, 0.0
set multiplot
set size 0.5,0.5; set origin 0.0,0.5
splot [-2*c:c][-c:c] cos(r(x,y))*exp(-k*r(x,y))
set size 0.5,0.5; set origin 0.5,0.5
set view 60,75,,
splot [-2*c:c][-c:c] cos(r(x,y))*exp(-k*r(x,y))
set size 0.5,0.5; set origin 0.0,0.0
set view 60,120,,
splot [-2*c:c][-c:c] cos(r(x,y))*exp(-k*r(x,y))
set nomultiplot
pause -1

```

5.12 繰返し : reread

Gnuplot はプログラミング言語ではないので `while(条件文)` などの分岐制御ができません . まったくできないのではなく , 繰返しのための `reread` コマンドがあります . その働きは , あるスクリプト中に `reread` コマンドがあったとすると , そこまでスクリプトを最初から実行するというものです . これと三項演算子 `'a?b:c'` および `'load'` コマンドを組み合わせれば , 条件付きのループがある程度可能です . 図 5.7 の線種と記号の一覧は次の二つのスクリプトで作りました . `'mark.gp'` で `'marksub.gp'` を load していますが , 変数 `i, j, jmax` が `'marksub.gp'` と共有されています . `marksub.gp` では , `i, j` をループ変数にして , 記号が表示される `key` の位置を決定しています .

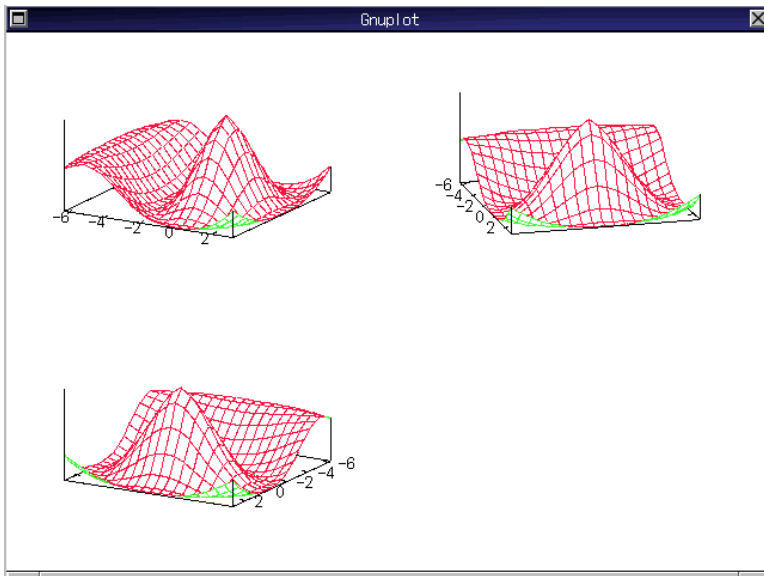


図 5.22: multiplot の使用例

— 'mark.gp' の list —

```

set term tgif
set output 'marks.obj'
set sample 2
set pointsize 1
set xrange[0:100]
set yrange[0:100]
i=0; j=0
jmax = 7
set multiplot
load "marksub.gp"
set nomultiplot

```

— 'marksub.gp' の list —

```

set size 1,1
set origin 0,0
set noborder
set noxtics; set noytics
set format x ""; set format y ""
set key i, 95-j*5
plot [0:10][0:100] -1 t " " \
                w lp 8*j+i 8*j+i
i=i+1
if ( i >= 8 ) i = 0; j = j+1
if ( j <= jmax ) reread

```

5.13 Gnuplot の大技小技

記号を大きくしたい : `set pointsize <大きさ>`

記号の大きさは X11 画面上で見える場合には、邪魔にならない大きさでよいのですが、印刷するために Tgif や PS 形式にした場合は小さすぎるようです。一括して大きさを変更するコマンドは “`set pointsize <大きさ>`” です。バージョン 3.6 では、スタイル指定時に `ps <大きさ>` オプションで記号の大きさを各線ごとに変更することができます。

強制的な左そろえ : `set lmargin <マージン>`

`multiplot` を用いて、 x 軸を共通にしていくつかのグラフを並べる際に、 y 軸のラベルの違いにより、枠が不ぞろいになってしまう場合があります。これを回避するには、枠のマージンを強制的に指定する “`set lmargin <マージン>`” コマンドを使うとよいです。

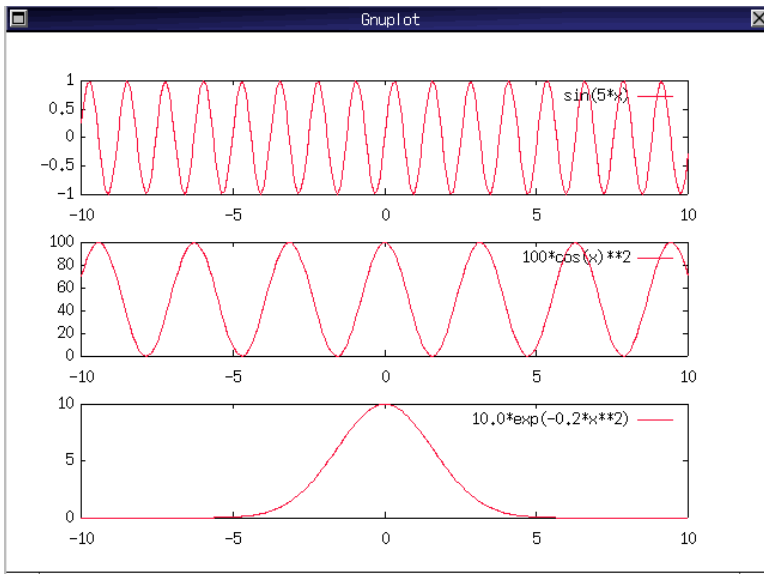


図 5.23: `multiplot` で枠の位置を揃える

見出しなしの目盛 : `set format {<軸>} ""`

前項の例では共通の x 軸の目盛数字が重複していて見苦しいです。一番下の目盛数字を残して、上の二つのプロットから目盛数字を消すにはどうしたらよいでしょうか。“`set noxtics`”では目盛自身が消えてしまいます。目盛数字の書式を『何も書かない=空白』と指定すれば解決します。次のスクリーンを実行して 図 5.23 と比較してみてください。

```
set multiplot
set samples 200
set origin 0.0,0.05; set size 0.9,0.35; set ytics 0,5,10
set lmargin 10; set bmargin 1
plot 10.0*exp(-0.2*x**2)
set origin 0.0,0.35; set size 0.9,0.35; set ytics 0,20,100
set lmargin 10; set bmargin 1
set format x ""
plot 100*cos(x)**2
set origin 0.0,0.65; set size 0.9,0.35; set ytics -1,0.5,1
set lmargin 10; set bmargin 1
set format x ""
plot sin(5*x)
set nomultiplot
pause -1
```

正方形にしたい : `set size square`

枠を厳密に正方形にすることは難しいです。Gnuplot で `size` とは、軸のラベル(文字)までを含んだ大きさを指しているからです。しかしながら、おおよそでよいならば“`set size square`”を使ってみましょう(図 5.24)。

プロット名(key)の位置を変えたい

プロットの名前(key)は通常、図の右上に表示されます。バージョン 3.5 ではこの表示の有無のみを“`set (no)key`”で指定できただけでしたが、パー

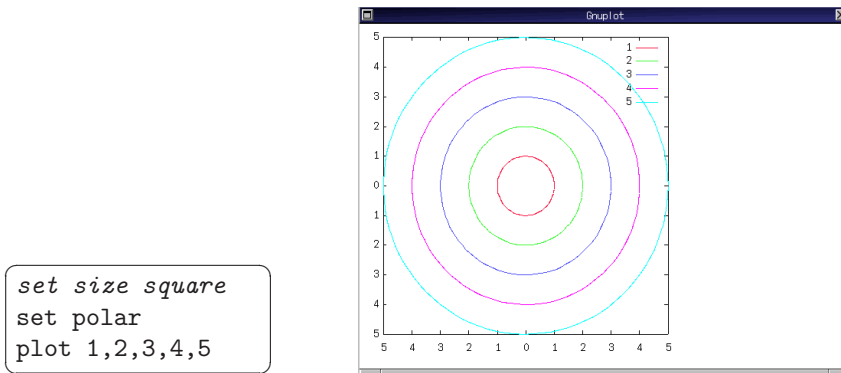


図 5.24: グラフの枠を正方形に指定したスクリプトとプロット

ジョン 3.6 では指定可能な内容が非常に充実しました . 詳しくは , Gnuplot のヘルプを見てください . 位置は

```
set key [left | right | top | bottom | outside | below
        | <position>]
```

で変更できます . ただし , <position> はグラフ上の x,y 座標で指定します . その他に , 右そろえや左そろえで出力する Left | Right 指定や , 枠で囲う box 指定も可能となっています .

z 軸 base 面の位置を変えたい : set ticslevel

三次元プロットの等高線図では描かれる面の既定は base 面で , 斜めから見ても等高線図が見えるように z 表示範囲外のかなり下に設定されています (図 5.25) . このため , z 方向の描画に使われる領域が狭まっています . ぎりぎりいっぱい z 方向を使いたいなら , ticslevel を 0.5 以下に設定して base 面を変更しましょう . “set ticslevel 0” で base 面は表示範囲の下端に位置します . ticslevel の既定値は 0.5 です .

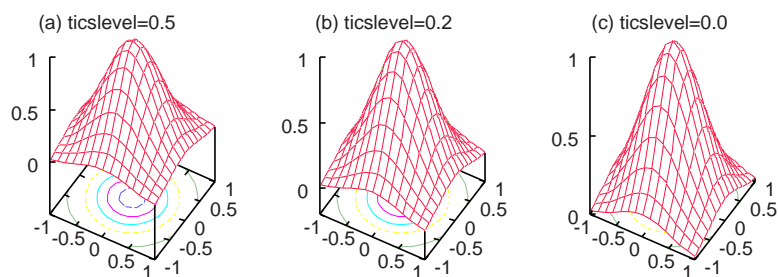


図 5.25: ticslevel の効果

5.14 gnuplot+

図 5.26 の図面が Gnuplot 上の指示のみで描けることに感心した読者も多いのではないのでしょうか。gnuplot+ は、PostScript 形式の出力に L^AT_EX like な処理 (数式モード) 機能や日本語表示機能を追加するためのパッチパッケージで、千葉大学情報センターの山賀氏が開発・保守をしています。詳しい情報は以下の URL をご覧ください。

<http://www.ipc.chiba-u.ac.jp/~yamaga/gnuplot+>

パッチを当てて gnuplot をコンパイルし直す必要がありますが、その手間をかけただけの成果があがります。PS の機能を十分使って非常に美しい仕上りのグラフが作成できるようになります。文字の回転なども直接 Gnuplot 上で指示できますから、レイアウトがほぼ固まったものについては tgif などのドローイングツールを経由せずに図が仕上がるでしょう。

以下のような gnuplot スクリプトを実行して得られた結果が図 5.26 です。

二次電子放出係数の入射エネルギー依存性

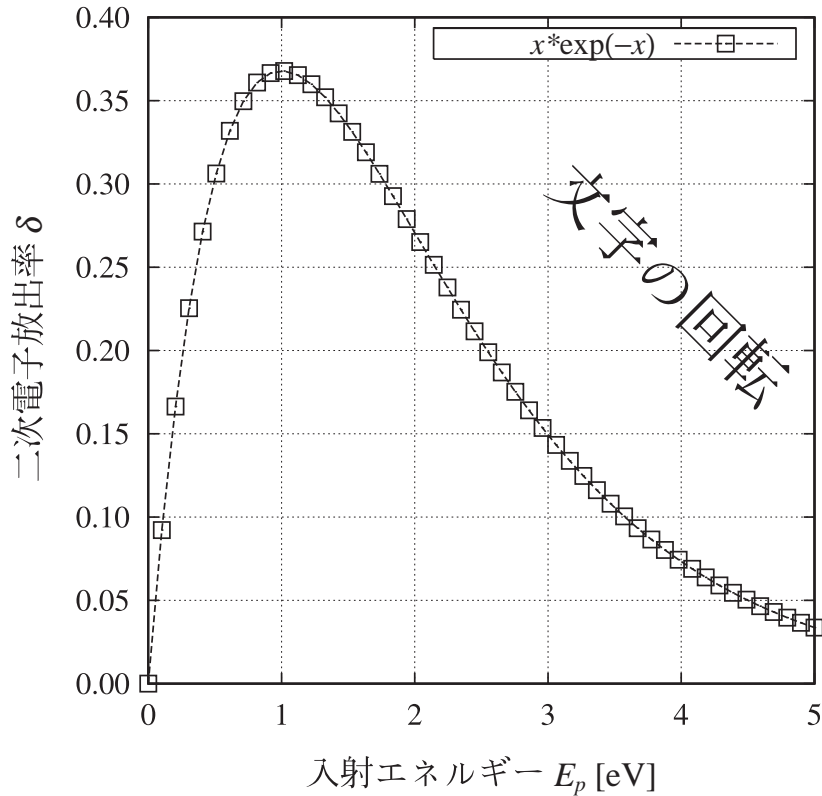


図 5.26: gnuplot+ によるギリシャ文字や添字の出力例

gnuplot+ の使用例

```

set term postscript portrait plus "Times-Roman-Gothic" 18
set output 'gp+.ps'
set size square
set sample 50
set grid
set key samplen 5 box
set format y '%.2f'
set title "\size=22 \bf 二次電子放出係数の入射エネルギー依存性" 0,2
set xlabel "\size=20 入射エネルギー $E_p$ [eV]" 0,-0.5
set ylabel "\size=20 二次電子放出率 $\delta$" 1,0
set label 1 "\rotate=-45 \size=36 文字の回転" at 3.0,0.30
plot [0:5] x*exp(-x) t '$x*\exp(-x)$' w lp lt 2 lw 2 pt 4 ps 1.5
    
```

5.15 Emacs との連携

対話的なモードでは、一度に1行しか表示されませんから、それまでコマンドの履歴全体を確認する場合や変更を行うには不便です。そこで、Gershon Elber 氏が提供した Emacs 上の Gnuplot 環境 'gnuplot.el' を使ってみるのもよいでしょう。キーバインドは

ESC r send-region-to-gnu-plot 指定領域のコマンド実行

ESC e send-line-to-gnu-plot カーソル行のみのコマンド実行

の二つしかありませんが、これだけでも随分と楽にはなります。'gnuplo.el' を /usr/local/share/emacs/site-lisp などに加えておき、各自の設定ファイル '.emacs' に次の行を追加すれば、拡張子 '.gp' の付いてるファイルを Gnuplot のソースとして認識するようになります。

.emacs への設定

```
(setq auto-mode-alist (append '(("\\.gp$" . gnu-plot-mode))
                              auto-mode-alist))
```

5.16 他のプロットツール

5.16.1 Xgraph

Gnuplot はもともとが関数のプロッターなので、スクリプトファイルを読み込む機能はありますが、標準入力からデータを pipe でもらってというような使い方はできません。それに比べて Xgraph は標準入力から x, y の座標データを読み込んで二次元プロットを作成してくれます。すなわち、

```
awk 'BEGIN{for (i = 0; i < 5000; i++) print rand(), rand()}'
      | xgraph -nl -P -bg snow
```

のような使い方ができます。既定では折れ線を描くので、この場合には一点一点を描くモード (scatter plot) '-nl' と、大きめの点記号 '-P' をオプション指定しています。図 5.27 に実行画面を示します。印刷ダイアログでは、当然ながら PS ファイルの保存も指定できます。"xgraph -h" とでもして、オプショ

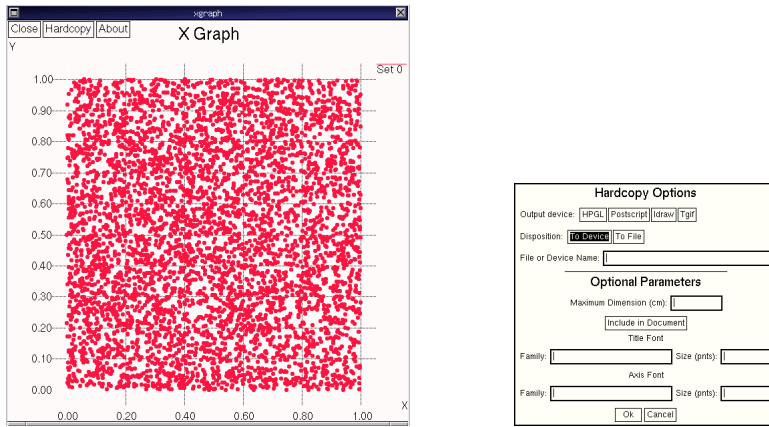


図 5.27: xgraph とその印刷ダイアログ画面

ンを見てください。一通りの図が描けるようになっています。ただし、ソースが古いこともあって、Linux ではコンパイル時に大量の warning がでますし、実際にはいくつかのオプションは正常に機能しないようです。

5.16.2 GNU plotutils

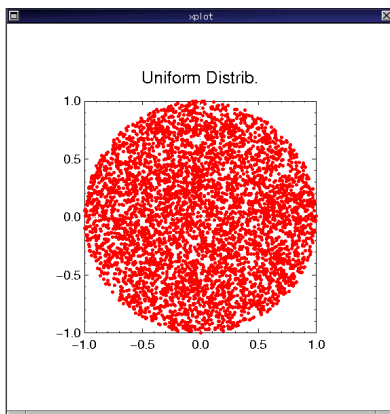


図 5.28: graph による一様乱数分布のプロット

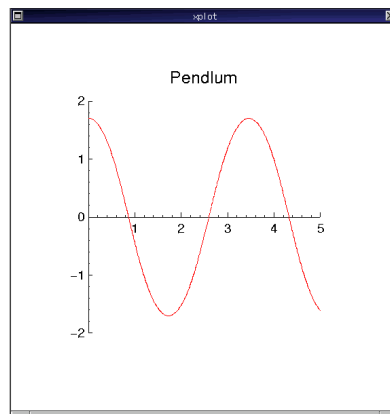


図 5.29: ode による単振り子の常微分方程式の数値解析

Xgraph は保守されていないし、コンパイルも make 一発という風でもないので、ちょっと不安という方に朗報です。GNU が開発しているプロットのためのユーティリティ群があります。プロットライブラリやデータの記述書式を標準化しています。もちろん、実用的なツールも含まれていて Xgraph と同じような機能を持つ graph は優れたものです (図 5.28)。awk からパイプでデータ入力する例を示します。

```
awk 'BEGIN{for (i = 0; i < 5000; i++){ r=sqrt(rand());
t=6.283*rand(); print r*cos(t), r*sin(t)}}'
| graph -TX -x -1 1 -y -1 1 -S 16 0.02 -m -1 -L'Uniform Distrib.'
```

-T は表示デバイス指定のオプションで、X, ps, fig (xfig), hpgl (HP-GL), tek (Tektronics 4014) を選択できます。-T 指定にしないと GNU のグラフィックメタファイル形式で出力します。-S n は記号の指定。-m n は線種指定で、scatter plot にするには負の値を指定します。また、-L "ラベル"、-X "x 軸ラベル" など使えます。その他のオプションは "graph -help" で眺めてください。GNU plotutils には常微分方程式を解くためのツール ode が含まれています。使い方が簡単なので、とても重宝します (図 5.29)。

5.16.3 Yorick: An Interpreted Language

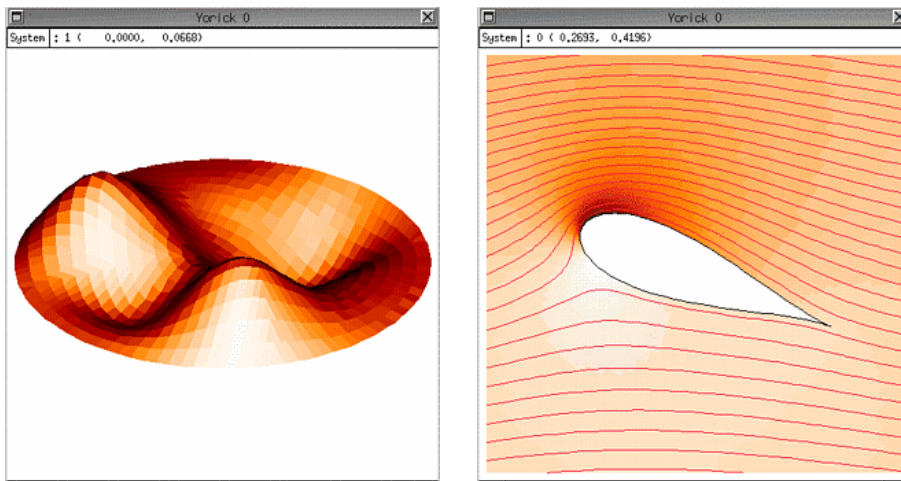


図 5.30: yorick のデモより : 円鼓膜の振動と翼の周囲の流体解析

Gnuplot 単独ではカラーアニメーションは無理です。gnuconv と組み合わせる gif 形式のセル画面を一枚一枚作成するスクリプトで処理するといった方法が思い浮かびますが、かなり労力がいりそうです。Yorick は汎用のインタプリタ言語ですが、標準でアニメーション関数が用意されている点が便利そうなので紹介します。図 5.30 にデモの一部を示します。ただし、アニメーション関数を呼ぶプログラムを、C 言語に似た様式で書く必要があります。



関連サイト情報

- http://www.cs.dartmouth.edu/gnuplot_info.html , ダートマス大学にある gnuplot の公式サイト .
- <http://science.nas.nasa.gov/~woo/gnuplot/beta> , Alex Woo 氏の gnuplot3.6 β 作業ページ , 配布のミラー .
- <ftp://cmpr1.phys.soton.ac.uk/pub/> , gnuplot3.6 β の公式配布サイト .
- <http://www.ipc.chiba-u.ac.jp/~yamaga/gnuplot+> , 本文と重複しますが , gnuplot+ の作者山賀氏の Web サイト .

索引

- angles, 91
- autoscale, 84
- awk, 93, 97, 116, 118

- bc, 79
- besj0(), 79
- besy0(), 79
- border, 91

- contour, 100

- dgrid3d, 102

- epsbox.sty, 94
- erf(), 79
- exit, 77

- fit, 106
- floor(), 79
- format, 86

- gnuconv, 97, 103
- gnuplot.el, 116
- gnuplot+, 114
- graph, 118
- grid, 91

- help, 77

- isosample, 99

- key, 91

- label, 85
- lgamma(), 79
- logscale, 85

- margin, 111
- Metafont, 75
- multiplot, 109

- ode, 118
- output, 94

- parametric, 104
- plot, 77
- plotutils, 117
- pointsize, 111
- polar, 105
- postscript, 75

- quit, 77

- rand(), 79
- range, 84
- replot, 99
- reread, 109

- sample, 83
- sgn(), 79

size, 91
splot, 98
square, 112

table, 96
teminal, 94
tics, 92
ticslevel, 113

using, 93

view, 91

Xgraph, 116

Yorick, 118

演算子

三項演算子, 79

単項演算子, 79

二項演算子, 79

虚数単位, 79

区切り文字, 94

組込み関数, 79

式, 78

スタイル, 86

対話的, 76

点列の表示, 92

パイプ, 97

立体画像, 97