7 グラフック・ユーティリティ–gnuplot–

gnuplot は Thomas Williams、Colin Kelley らによって作られたグラフィック・ユーティリティであり、現在では Unix のみならず、Windows や MacOS 上でも利用できる。gnuplot は「数式をグラフにする」「データ (表) をグラフにする」道具として、数値計算や実験データの表示 (視覚化) や、論文作成用のグラフ作成などに非常に有用である。

そこで本章では、数値計算の講義・演習を一旦休止して、gnuplot の使い方について簡単に解説する。gnuplot を利用することにより、学生実験のレポート作成や、今後の数値計算結果の表示などに積極的に役立てて欲しい。

7.1 gnuplot の入手方法

gnuplot は**フリーソフトウエア**(使用料、配布などがフリー) であるので、各人の PC(Windows や MacOS など) に自由に導入して利用することができる。

gnuplot のオフィシャル・サイトは、

```
http://www.gnuplot.info
```

で、こちらから gnuplot をダウンロードすることができる。実際の導入方法などについては、付属の説明や、インターネット上のドキュメントを参照のこと。

7.2 さあ始めよう

7.2.1 起動と終了

gnuplot は、kterm から、

```
% gnuplot↵
```

とすると起動する。gnuplot が起動している間は、kterm は gnuplot の端末 (gnuplot に対する命令を入力する) になるので、**最後に&を付けてはいけない**。

起動すると、

```
% gnuplot

        G N U P L O T
        Version 4.0 patchlevel 0
        ...(途中省略)...
Terminal type set to 'x11'
gnuplot>
```

というオープニングメッセージが表示された後、gnuplot のプロンプト gnuplot>が表示され、gnuplot にコマンドを入力できる状態になる。

終了は、プロンプト gnuplot>から以下のいずれかを入力すればよい。

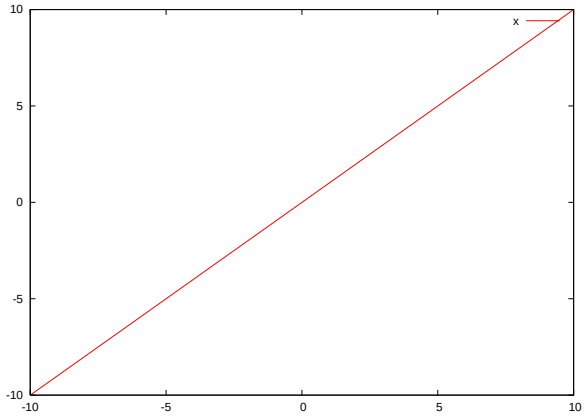

図 7.1: $f(x) = x$ のグラフの例。

```
gnuplot> exit↵
gnuplot> quit↵
gnuplot> q↵
gnuplot> Ctrl-d
```

7.2.2 式のグラフ

gnuplot のプロンプトから

```
gnuplot> plot x↵
```

と入力してみよう。すると、新たなウインドウが開き、図 7.1 に示すグラフが表示される。入力が間違っていると、「invalid command(コマンドが間違っています)」や「undefined variable(変数が間違っています)」などの**エラーメッセージ**が表示され、新たなウインドウは開かない。入力を確かめ、もう一度正確に入力すること。

plot は 2 次元グラフを表示するコマンドで、書式は、

```
plot 式
```

である。式は x を変数として記述すればよく、例えば

```
gnuplot> plot sin(x)↵
```

とすれば、$f(x) = \sin x$ のグラフが描かれる。このとき、新しいウインドウは現れず、前に描いた $f(x) = x$ のグラフが消えて、そのウインドウに $f(x) = \sin x$ のグラフが描かれる。グラフを重ねて表示する方法は 7.2.3 節を参照のこと。

図 7.1 を詳しく見ると、横軸・縦軸共に範囲が $[-10, 10]$ となっている。plot コマンドでは標準では、横軸は $[-10, 10]$ にされ、縦軸はグラフがうまく収まるように自動的に設定される。範囲をユーザーが設定する方法は 7.3.2 節を参照のこと。またグラフの右上には描いた式が表示される。

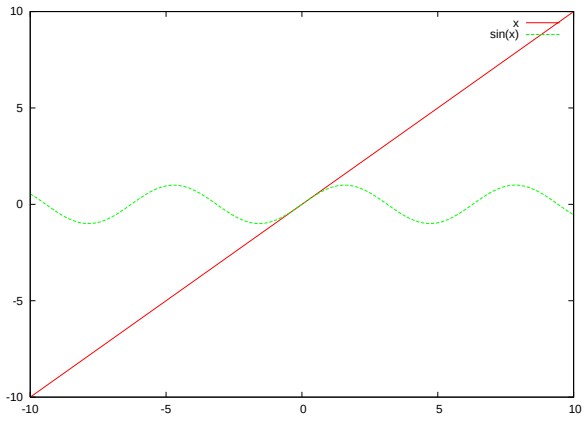

図 7.2: $f(x) = x$ と $\sin x$ の 2 つのグラフを表示する例。

7.2.3 複数のグラフの表示

複数のグラフを表示する場合も plot コマンドを用いればよく、書式は、

> plot **式 1**, **式 2**, ···

である。例えば、$f(x) = x$ と $f(x) = \sin x$ の 2 つのグラフを重ねて表示したい場合は

```
gnuplot> plot x, sin(x)↵
```

とすればよく、図 7.2 のようなグラフが表示される。

もう一つの方法は、replot コマンドを用いる方法である。replot は、直前に実行された plot コマンドを再実行すると共に、指定された式を描く。従って、

```
gnuplot> plot x↵
gnuplot> replot sin(x)↵
```

とすると、最初に実行された plot x を、replot コマンドが再実行すると共に、sin(x) のグラフも描かれる。

7.2.4 データのプロット

以下のようなデータの入ったファイルが、ファイル名 test1.dat として gnuplot を起動したディレクトリにあるとする。

```
# test1.dat
# x     y
-10.0 -0.839
 -9.0 -0.911
 -8.0 -0.146
 -7.0  0.754
```

```
-6.0  0.960
-5.0  0.284
-4.0 -0.654
-3.0 -0.990
-2.0 -0.416
-1.0  0.540
 0.0  1.000
 1.0  0.540
 2.0 -0.416
 3.0 -0.990
 4.0 -0.654
 5.0  0.284
 6.0  0.960
 7.0  0.754
 8.0 -0.146
 9.0 -0.911
10.0 -0.839
```

1 行目と 2 行目が#で始まっているが、これはコメントを表す。gnuplot は#以下の 1 行をコメントとみなして無視する。3 行目以降は、1 行毎に x と y の値のペア (データ) が記入されている。gnuplot では、1 行に 1 つのデータを書く決まりになってる。

データファイルの内容をプロットする書式は、

plot **’ファイル名’** (または**” ファイル名”**)

とファイル名をシングルクオテーション (’) またはダブルクオテーション (”) で囲んで指定する。標準では gnuplot を起動したディレクトリからファイルが読み込まれる。ディレクトリが異なる場合は、

```
plot '../SAMPLE/test1.dat'↵
```

などとディレクトリを含む形でファイル名を指定すればよい。ただし、

```
plot '~/SAMPLE/test1.dat'↵
```

などと~ (ホームディレクトリを表す) を使った表現は使えない。

例題の場合は

```
gnuplot> plot 'test1.dat'↵
```

とすればデータ ($y = f(x) = \cos x$ である) が表示される。

7.2.5 線でつなぐ

データを表示する際に、データ間を結んだ線を同時に描きたい場合がある。そのような場合は、plot コマンドの with オプションを用いる。書式は、

```
plot 'ファイル名'(または式) with 線種
```

である。線種は例えば、

linespoints	データ点とデータを結んだ線を表示
lines	データを結んだ線のみを表示

である。

例題の場合は

```
gnuplot> plot 'test1.dat' with linespoints↵
```

とすればデータ点とそれらを結んだ線が表示され、データが $f(x) = \cos x$ から作られたものであることが良く分かる。

7.2.6 3 次元グラフの表示

splot は 3 次元 (2 変数) グラフを表示するコマンドで、書式は、

```
splot 式
```

である。式は x と y を変数として記述すればよく、例えば

```
gnuplot> splot x**2+y**2↵
```

とすれば、$f(x, y) = x^2 + y^2$ のグラフが描かれる。

7.2.7 3 次元のデータプロット

データを 3 次元で表示する場合も splot コマンドを用いる。データファイルの読み込みや、データ点・データ点を結ぶ線の描画の指定の仕方は plot コマンドの場合と同じで、書式は

```
splot 'ファイル名' with 線種
```

である。

一番簡単な使い方は、(x, y) 平面を $m \times n$ のメッシュ(方眼) に分割し、各点での値 z を表として与えるというものである。$x = x_1, x_2, \cdots, x_m$、$y = y_1, y_2, \cdots, y_n$ と分割されている場合、データファイルは以下のように用意すればよい。1 行に x, y, z の 3 つの値を記入し、ある $y_i (i = 1 \sim n)$ に対する m 個のデータ毎に空行で区切られている事に注意せよ。

$$
\begin{array}{ccc}
x_1 & y_1 & z_{11} \\
x_2 & y_1 & z_{21} \\
\vdots & \vdots & \vdots \\
x_m & y_1 & z_{m1} \\
\multicolumn{3}{c}{\text{(空行)}} \\
x_1 & y_2 & z_{12} \\
x_2 & y_2 & z_{22} \\
\vdots & \vdots & \vdots \\
x_m & y_2 & z_{m2} \\
\multicolumn{3}{c}{\text{(空行)}} \\
\multicolumn{3}{c}{\vdots} \\
x_1 & y_n & z_{1n} \\
x_2 & y_n & z_{2n} \\
\vdots & \vdots & \vdots \\
x_m & y_n & z_{mn}
\end{array}
$$

このようなデータの例を

```
/home/teacher/z6wt01in/SAMPLE/laplace.dat
```

に置くので、各人コピーすること。このデータをプロットするには以下のようにすればよい。

```
gnuplot> splot ‘laplace.dat‘ with line↵
```

表示されるグラフは、中央に電荷を置いた場合の電位分布である。

7.3 グラフのカスタマイズ

前節で説明したコマンドを使えば、数値データの視覚化などの最低限の処理は行える。しかしながら、レポートに載せるグラフにするには、グラフの範囲を変更したり、各軸の説明を入れたりする必要がある。そこで本節では、このようなグラフのカスタマイズについて解説する。

7.3.1 set コマンドと show コマンド

gnuplot でグラフをカスタマイズするには set コマンドを用いる。書式は

set **オプション**

である。現在のオプションの状態を見るには show コマンドを使えばよく、書式は

show **オプション**

である。

gnuplot が現在どのようなグラフを描く設定になっているかは

```
show all↵
```

として、全てのオプションの現在値を表示させることにより確認することができる。各人、試してみて欲しい。

7.3.2 軸の範囲を指定する

軸の範囲を指定するには、plot/splot コマンドのオプションで指定する方法と、set コマンドで指定する方法の 2 つがある。

plot/splot のオプションで指定

plot/splot のオプションで軸の範囲を指定する書式は

> plot［ x **軸の左端値**:x **軸の右端値**］［ y **軸の下端値**:y **軸の上端値**］ **式**
> splot［ x **軸の左端値**:x **軸の右端値**］［ y **軸の前端値**:y **軸の奥端値**］［ z **軸の下端値**:z **軸の上端値**］ **式**

である。データを表示する場合は、**式**の部分を’**ファイル名**’にすればよい。

各軸の範囲を指定する値は、例えば x 軸の場合、x **軸の左端値**$<x$ **軸の右端値**である必要はない。あくまで、左右端の値の指定であるので、軸の向きを変えることもできる。

具体的な例としては

```
gnuplot> plot [-5:5] sin(x)↵
gnuplot> plot [-5:5] [-1.2:1.2] sin(x)↵
gnuplot> splot [] [] [:2] sin(x)*cos(y)↵
```

等が挙げられる。

1 番目の例では、y 軸の範囲は指定されていないので、標準の値 (グラフが全て収まるような値) が使われる。2 番目の例では、x 軸も y 軸も、ユーザーが指定した範囲に限られる。3 番目の例では、x と y 軸の範囲は指定されていない ([]) ので、標準の値 ([-10:10]) が使われる。z 軸の範囲も下限値は指定されていないので、標準値 (グラフがうまく収まるような値。この場合は -1) が使われる。

set コマンドで指定

set コマンドで軸の範囲を指定する書式は

> set xrange［ x **軸の左端値**:x **軸の右端値**］
> set yrange［ y **軸の下端 (または前端) 値**:y **軸の上端 (または奥端) 値**］
> set zrange［ z **軸の下端値**:z **軸の上端値**］

である。

具体的な例としては

```
gnuplot> splot sin(x)*cos(y)↵
gnuplot> set xrange [-3:3]↵
gnuplot> set yrange [-2:2]↵
gnuplot> set zrange [-2:2]↵
gnuplot> replot↵
```

が挙げられる。1 行目では各軸の範囲は指定されていないので、標準値 (x, y 軸は $[-10:10]$、z 軸はグラフが全て収まるように $[-1:1]$) が使われる。2–4 行目で各軸の範囲を指定しており、5 行目の replot(直前の plot/splot コマンドを再実行) では、指定した範囲でグラフが描かれる。

各軸の範囲設定を自動設定に戻すには、

```
set autoscale 座標軸
```

とすればよく、**座標軸**には x,y,z,xy などが指定できる。省略すると、全ての軸が自動設定になる。

7.3.3 軸の目盛りを変更する

set コマンドで目盛りの刻み位置を変える書式は

```
set xtics  x 軸の目盛りの開始位置，目盛りの間隔，目盛りの終了位置
set ytics  y 軸の目盛りの開始位置，目盛りの間隔，目盛りの終了位置
set ztics  z 軸の目盛りの開始位置，目盛りの間隔，目盛りの終了位置
```

である。

例えば

```
gnuplot> plot sin(x)↵
gnuplot> set xtics -10,2,10↵
gnuplot> replot↵
```

とすると、x 軸は $[-10, 10]$ の領域に対して、1 行目の場合は 5 刻みで目盛りが付き、3 行目の場合は 2 刻みで目盛りが付く。

目盛りの刻み位置を標準 (自動設定) に戻すには

```
set xtics auto
set ytics auto
set ztics auto
```

とすればよい。

7.3.4 小目盛りを付ける

set コマンドで軸に小目盛り (xtics などで指定した目盛りの間に表示される細かい目盛り) を表示・指定する書式は

```
set mxtics  x 軸の小目盛りの刻み数
set mytics  y 軸の小目盛りの刻み数
set mztics  z 軸の小目盛りの刻み数
```

である。ここで、**小目盛りの刻み数**は、xtics 等で指定された目盛りの間を更にいくつに刻むかを指定する。

例えば

```
gnuplot> set autoscale↵
gnuplot> set mxtics 5↵
gnuplot> plot sin(x)↵
```

とすれば、$\sin x$ が x 軸の通常の目盛りは 5 刻み、小目盛りは 1 刻みのグラフ上に描かれる。

標準では小目盛りは表示されない。小目盛りを表示しない標準の状態に戻すには、unset コマンドを使えばよい。書式は、

```
unset mxtics
unset mytics
unset mztics
```

である。

7.3.5 タイトルを付ける

set コマンドでグラフにタイトルを付ける書式は

```
set title ’タイトル’ 横へのオフセット文字数, 縦へのオフセット文字数
```

である。オフセット文字数を省略すると、タイトルはグラフの上部中央に表示される。タイトルの位置を変更したい場合は、上部中央からのオフセット (移動量) を文字数で指定する。

例えば

```
gnuplot> set title ’Gauss Function’↵
gnuplot> plot exp(-x**2)↵
```

とすると、ガウス関数 e^{-x^2} が描かれ、グラフ上部中央に Gauss Function と表示される。

7.3.6 軸にラベルを付ける

set コマンドで各軸にラベル (説明) を付ける書式は

```
set xlabel ’x 軸のラベル’ 横へのオフセット文字数，縦へのオフセット文字数
set ylabel ’y 軸のラベル’ 横へのオフセット文字数，縦へのオフセット文字数
set zlabel ’z 軸のラベル’ 横へのオフセット文字数，縦へのオフセット文字数
```

である。オフセット文字数を省略すると、ラベルは各軸の中央に表示される。ラベルの位置を変更したい場合は、中央からのオフセット (移動量) を文字数で指定する。

レポート提出時などは、必ず各軸のラベルを付けるように心がける事。

7.3.7 曲線のラベルの変更

これまで作成したグラフを見ると、式やデータファイル名がグラフの右上に表示されている。gnuplot ではこのグラフの説明を**キー**という。書式は plot/splot のオプションで軸の範囲を指定する書式は

```
plot  式 title  ’キー’
splot  式 title  ’キー’
```

である。データを表示する場合は、**式**の部分を’**ファイル名**’にすればよい。

例えば

```
gnuplot> plot sin(x) title ’Key 1’,cos(x) title ’Key 2’↵
```

とすれば、標準で sin(x)(cos(x)) と表示されるものが Key 1(Key 2) と表示される。

7.3.8 対数グラフ

set コマンドで各軸を対数スケールにする書式は

```
set logscale 軸 底
```

である。**軸**は、x,y,z,xy,yz,xz,xyz のいずれかであり、省略すると全ての軸が対数スケールになる。**底**は対数の底を指定し、省略すると 10 となる。

例えば

```
gnuplot> plot 10**x↵
gnuplot> set logscale y↵
gnuplot> replot↵
```

とすると、1 行目では通常のスケールで 10^x が描かれるが、3 行目では縦軸 (y 軸) が対数スケールで描かれる。

対数スケールから、通常のスケールに戻すには

```
unset logscale 軸
```

である。**軸**は、x,y,z,xy,yz,xz,xyz のいずれかであり、省略すると全ての軸が通常のスケールになる。

7.3.9 データの並びを指定する

これまではデータファイル内のデータの並びは x y(z) としてきたが、これは gnuplot の標準というだけであり、オプション using を使うことにより並びを指定する事ができる。書式は

```
plot ’ファイル名’ using x の列番号:y の列番号
splot ’ファイル名’ using x の列番号:y の列番号:z の列番号
```

である。列番号は左から $1, 2, \cdots$ である。

例えば、以前使用したデータファイル test1.dat で

```
gnuplot> plot ’test1.dat’ using 2:1 with linespoints↵
```

とすると、x 軸と y 軸を入れ換えたグラフを描く事ができる。

7.3.10 データを演算する

前出の using オプションでは、データの各列に対応した値を、列番号に$を付けて表すことができる。例えば、1 列目のデータは$1、2 列目のデータは$2 などである。

従って、以前使用したデータファイル test1.dat で

```
gnuplot> plot ’test1.dat’ using 1:($1*$2) with linespoints↵
```

とすると、1 列目のデータを x、1 列目と 2 列目のデータの積を y としたグラフを描く事ができる。

7.3.11 データを補間する

gnuplot には、データファイル内の点を補間したり近似したりして、曲線でグラフを表示するオプション smooth がある。書式は

```
plot ’ファイル名’ smooth 補間方法
splot ’ファイル名’ smooth 補間方法
```

である。補間方法に csplines を指定すると**スプライン補間**が、bezier を指定すると**ベジエ曲線**が適用される。各補間方法等は各人で調べる事。

以前使用したデータファイル test1.dat で

```
gnuplot> plot 'test1.dat' using 1:($1*$2) smooth csplines↵
gnuplot> plot 'test1.dat' using 1:($1*$2) smooth bezier↵
```

として、補間方法の違いを各人確認すること。スプライン補間では必ず全ての点を通るように曲線が描かれているのに対して、ベジエ曲線を用いる場合は必ずしも通らないことに注意せよ。

7.4 グラフの表示型式

これまでは主に線グラフを扱ってきたが、この節ではヒストグラム (棒グラフ) などの他の表示型式や、グラフに用いる線や点の種類を変更する方法を解説する。

7.4.1 ライン表示

関数やデータをラインのみで表示するには、オプション lines を使用する。書式は

```
plot  式(または'ファイル名') with  lines
splot  式(または'ファイル名') with  lines
```

である。なお、**式**の場合は、with lines が標準であるので省略することができる。

7.4.2 データポイント表示

関数やデータを点 (マーク) で表示するには、オプション points を使用する。書式は

```
plot  式(または'ファイル名') with  points
splot  式(または'ファイル名') with  points
```

である。

式の場合は、標準で等間隔にサンプル点が 100 点選ばれて、点で表示される。サンプル点の数を変更するには、

```
set samples サンプル点の数
```

とすればよい。

7.4.3 ラインとデータポイント表示

オプション linespoints を使うと、前述の lines と points 両方のオプションを使ってグラフが描かれる。書式は

```
plot  式(または'ファイル名') with  linespoints
splot  式(または'ファイル名') with  linespoints
```

である。
例えば

```
gnuplot> set samples 50↵
gnuplot> plot cos(x) with linespoints↵
```

とすると、50 のサンプル点を使って、関数 $\cos x$ が linespoints スタイルで表示される。ここで、linespoints オプションでは、サンプル点間が直線で結ばれることに注意せよ。

7.4.4 ヒストグラム表示

データをヒストグラムで表示するには、

```
plot  'ファイル名' with  steps
plot  'ファイル名' with  fsteps
plot  'ファイル名' with  histeps
```

とすればよい。
ヒストグラムはオプションに応じて以下のように描かれる。

steps	(x_1, y_1) から始まるヒストグラム
fsteps	(x_1, y_1) で終わるヒストグラム
histeps	(x_1, y_1) を中心とするヒストグラム

サンプルデータを、

```
/home/teacher/z6wt01in/SAMPLE/test2.dat
```

におくので、各人オプションを変えて表示してみて、オプションの働きを理解すること。

7.4.5 誤差グラフ

実験データに誤差はつきものである。従って実験データを整理して表示する際、誤差グラフを使うことになる。
(x, y^{+a}_{-b}) というデータを誤差棒付きで表示するには、データファイルを、

$$x \quad y \quad y-b \quad y+a$$

という順で用意し

```
plot 'ファイル名' with yerrorbars
```

とすればよい。
$(x, y \pm c)$ というデータを誤差棒付きで表示するには、データファイルを、

$$x \quad y \quad c$$

という順で用意し

```
plot 'ファイル名' with yerrorbars
```

としてもよい。

$(x^{\alpha}_{-\beta}, y)$ というデータを誤差棒付きで表示するには、データファイルを、

$x \;\; y \;\; x-\beta \;\; x+\alpha$

という順で用意し

```
plot 'ファイル名' with xerrorbars
```

とすればよい。

$(x \pm \gamma, y)$ というデータを誤差棒付きで表示するには、データファイルを、

$x \;\; y \;\; \gamma$

という順で用意し

```
plot 'ファイル名' with xerrorbars
```

としてもよい。

$(x^{\alpha}_{-\beta}, y^{a}_{-b})$ というデータを誤差棒付きで表示するには、データファイルを、

$x \;\; y \;\; x-\beta \;\; x+\alpha \;\; y-b \;\; y+a$

という順で用意し

```
plot 'ファイル名' with xyerrorbars
```

とすればよい。

$(x \pm \gamma, y \pm c)$ というデータを誤差棒付きで表示するには、データファイルを、

$x \;\; y \;\; \gamma \;\; c$

という順で用意し

```
plot 'ファイル名' with xyerrorbars
```

としてもよい。

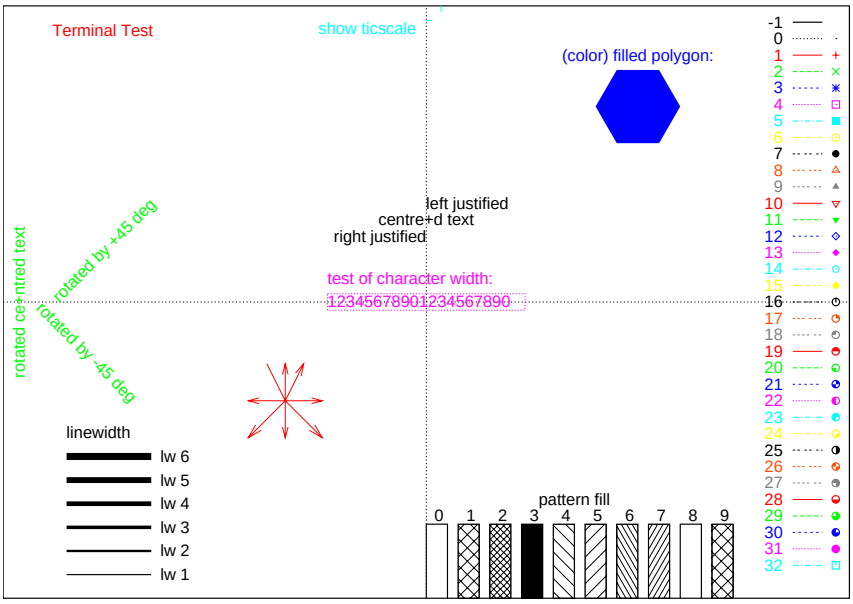

図 7.3: test コマンドの表示例。

7.4.6 棒グラフ

棒グラフを表示するには

```
plot 'ファイル名' with boxes
```

とする。

標準では、各棒が接するように横幅は自動的に決められる。横幅を指定するには

```
set boxwidth 棒の幅
```

とすればよい。**棒の幅**を省略すると、標準 (各棒が接するように自動設定) に戻る。

7.4.7 線や点の種類を変える

gnuplot には複数の線種や点種が用意されている。どのようなものが用意されているかは

```
gnuplot> test↵
```

とすることにより、図 7.3 に示すグラフが表示され確認することができる。各人、確認してみること。

線や点の種類を変えるには、with オプションの引数で

```
with スタイル 線種 点種
```

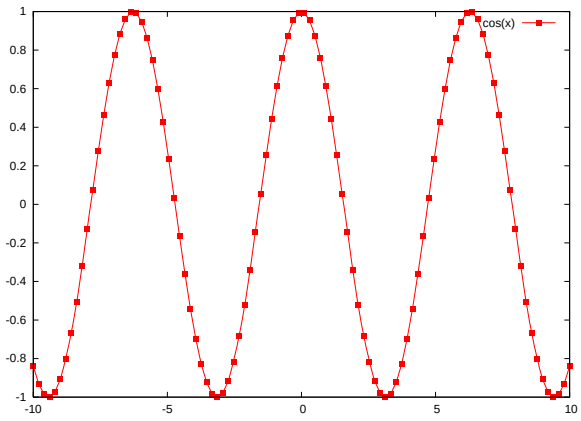

図 7.4: with オプションで線種・点種を指定して $f(x) = \cos x$ を表示した例。

とすればよい。**スタイル**は lines や points や linespoints である。**線種**および**点種**は test コマンドで表示される番号を指定する。

注意は、

- **スタイル**が points の場合も**線種**を指定する必要がある
- カラー表示の場合、**点種**の色は、**線種**の色と同じになる

である。

例えば

```
gnuplot> plot cos(x) with linespoints 1 5↵
```

とすると、図 7.4 に示すグラフが得られる。

7.5 最小二乗法

gnuplot には、非線型関数の最小二乗フィットを行う機能が組み込まれている。最小二乗フィットを行うコマンドは fit で、書式は

```
fit 関数 'ファイル名' via 変数
```

である。**関数**は事前に定義しておく必要があり、**変数**は各々をカンマ (,) で区切って指定する。

ここでは μ^+ 粒子の寿命測定を例に解説する。表 7.1 に 0.5 μs 毎の崩解数の測定例を載せる。

このデータのバックグラウンドが無視できるほど小さいならば、崩解数 $f(x)$ は時間 x の関数として

$$f(x) = a\exp(-bx) \tag{7.1}$$

で与えられる。μ^+ の寿命 τ は $1/b$ から求める事ができる。

崩解数のデータは Poisson 分布に従うとすると、統計的なばらつきとして $\pm\sqrt{\text{崩解数}}$ が伴う。そこで、

表 7.1: μ^+ 粒子の寿命測定のデータ。

時間 (μs)	崩解数	時間 (μs)	崩解数
0.25	999	5.25	98
0.75	818	5.75	86
1.25	662	6.25	76
1.75	501	6.75	53
2.25	419	7.25	43
2.75	307	7.75	24
3.25	262	8.25	24
3.75	208	8.75	21
4.25	171	9.25	16
4.75	131	9.75	22

時間　崩解数　$\sqrt{\text{崩解数}}$

の順にデータを記入したデータファイルを

```
/home/teacher/z6wt01in/SAMPLE/mu_decay.dat
```

に置く。

gnuplot では、まずフィットする関数を定義する。例題の場合は

```
gnuplot> f(x)=a*exp(-b*x)↵
```

とすればよい。

次に変数 a, b の初期値を与える。標準では初期値に 1 が与えられ、フィットが行われる。例題の場合は、初期値では指数関数の中が負で大きくなりすぎて計算できないので、適切な初期値を与える必要がある (標準の 1 で問題なければ、この部分は省略できる)。具体的には

```
gnuplot> a=1000↵
gnuplot> b=0.5↵
```

とすればよい。

実際にフィットするには

```
gnuplot> fit f(x) 'mu_decay.dat' using 1:2:3 via a,b↵
```

とすればよい。ここで `using 1:2:3` の部分は、データの並びが $x, y, \Delta y$ であることを表している。実行すると、計算の途中結果が表示され、非線型の最小二乗アルゴリズムにより、最適なパラメータの値が算出される[1]。画面表示の最後の方は、

```
Final set of parameters            Asymptotic Standard Error
=======================            ==========================
```

[1]計算の途中経過は、`fit.log` というファイル名で保存されている

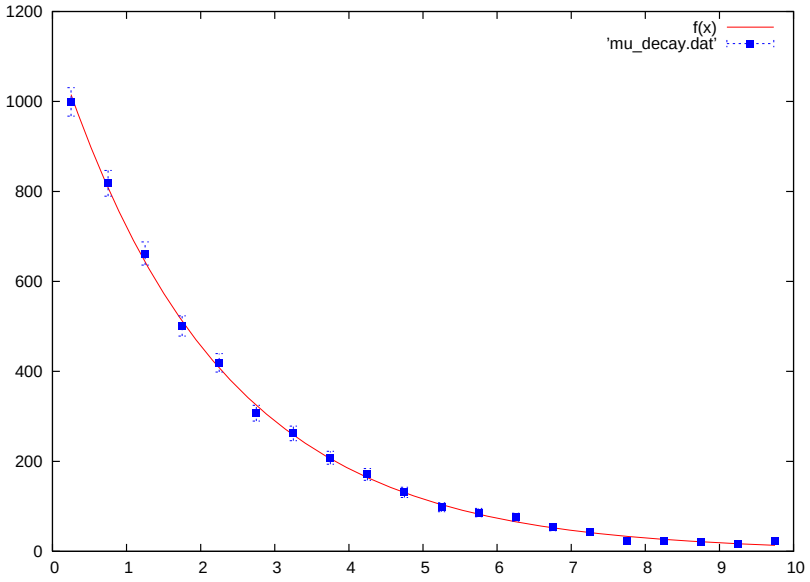

図 7.5: 表 7.1 に与えられた μ^+ 粒子の寿命測定データへの指数関数のあてはめの結果。

```
a               = 1137.76          +/- 19.49        (1.713%)
b               = 0.455834         +/- 0.006014     (1.319%)


correlation matrix of the fit parameters:

                a      b
a                1.000
b                0.736  1.000
```

となっており、b の値として、

$$b = 0.456 \pm 0.006\,(\mu\mathrm{s}^{-1}) \tag{7.2}$$

が得られている。これから、寿命は

$$\tau = \frac{1}{b} = 2.19 \pm 0.03\,(\mu\mathrm{s}) \tag{7.3}$$

と求まる。各人、文献値と比較してみよ。

元のデータファイルと、$f(x)$ によりフィットした曲線を重ねて描くには

```
gnuplot> plot f(x), 'mu_decay.dat' with yerrorbars 3 5↵
```

とすれば、図 7.5 に示すようなグラフが得られる。各人、オプションを適宜指定して、分かり易い (見やすい) グラフを作成してみよ。

7.6 グラフの出力

これまではグラフを画面上で見てきたが、プリンタに出力するには PostScript 型式のファイルに出力する必要があるし、後に説明する TEX や LATEX で取りこむには EPS 型式のファイルに出力する必要がある。

本節ではその方法について説明する。

7.7 PostScript 型式での出力

PostScript 型式で出力するには

```
set terminal postscript
```

により出力型式を PostScript にし、

```
set output 'ファイル名'
```

で出力ファイル名を指定した後、グラフを描けばよい。グラフは画面には表示されず、指定したファイルに書き込まれる。PostScript 型式のファイル名の拡張子は.ps が標準である。

PostScript 型式のファイルは gv コマンドにより画面で内容を確認することができる (プレビューという)。例えば、ファイル名が graph.ps なら

```
% gv graph.ps↵
```

とすればよい。gv を終了するには、File メニューを左クリックしてメニューをプルダウンし、Quit を選択すれば良い。

gv で確認して問題無ければ、プリンタに出力すればよい (紙を無駄にしない為にも、gv で確認した方がよい)。

画面出力に戻すには

```
set terminal x11
```

とする。

7.8 EPS 型式での出力

EPS 型式で出力するには

```
set terminal postscript eps
```

により出力型式を EPS にし、

```
set output 'ファイル名'
```

で出力ファイル名を指定した後、グラフを描けばよい。グラフは画面には表示されず、指定したファイルに書き込まれる。EPS 型式のファイル名の拡張子は.eps が標準である。

EPS 型式のファイルも gv コマンドにより画面で内容を確認することができる。確認して問題なければ、TEX や LATEX で取り込めばよい。