第１章　基本的な使い方
1.1 gnuplotを使ってみよう
1.2 関数を使ってみよう
1.3 関数を定義してみよう

第２章　任意のグラフをプロットする
2.1 gnuplotのプロット方法
2.2 任意のグラフをプロットする
2.3 グラフのプロット形式を変更する
2.4 軸のカスタマイズ
2.5 凡例文字列の変更と非表示

第３章　グラフを重ねてプロットする方法と幾つかの調整
3.1 データファイルを２つ用意する
3.2 ２つのグラフを重ねて表示する
3.3 軸の目盛りの幅を調整する
3.4 凡例の位置を変更する
3.5 グリッドを表示する

第４章　ファイルの出力
4.1 gnuplotが出力できる形式
4.2 ファイルへの保存方法
4.3 plt形式での保存と再利用

第５章　ファイルへの出力(EPS編)
5.1 PostScript形式で出力するメリット
5.2 PostScript形式で指定できるオプション
5.3 PostScript形式で出力する方法

第６章　線種とマークのカスタマイズ
6.1 testコマンド（含、マークの種類の変更）
6.2 マークの大きさを変更する
6.3 線種を変更する

第 7 章　一つのデータファイルに複数のデータをまとめる

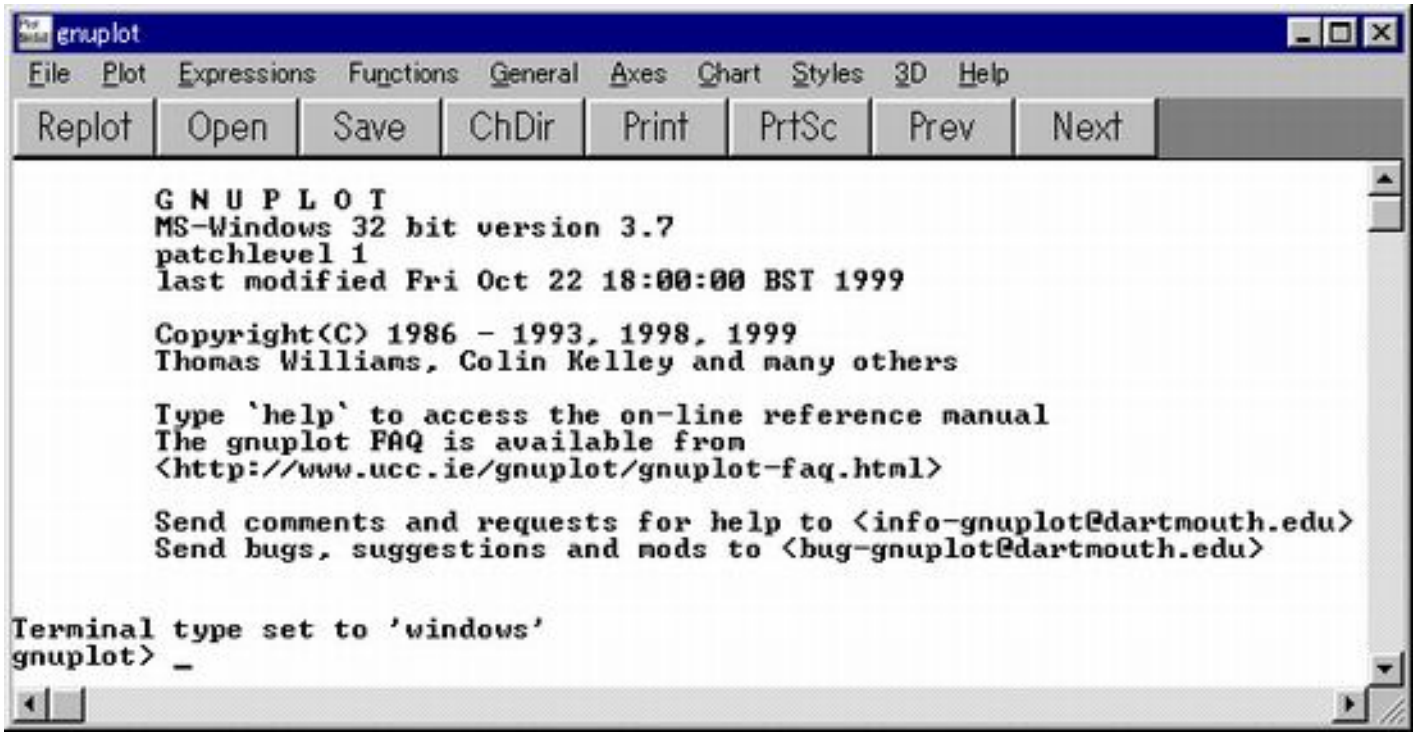
7.1 これからやること
7.2 データファイルを用意する
7.3 indexコマンド

第１章
基本的な使い方

indexへ

1.1　gnuplotを使ってみよう

ここでは、難しいことは抜きにして、とりあえずはgnuplotを使ってみましょう。
まず、gnuplotを起動してください。次のような画面が現れるはずです。



おそらく、最下行のgnuplot>の後にカーソルが点滅していますね。ユーザはここに命令を打ち込むことで、gnuplotに命令を送ります。

それでは、早速３次関数をプロットさせてみましょう。
グラフを描画させる命令は、plotとsplotの２つがあります。plotは２次元関数、splotは３次元関数をプロットさせるための命令です。今回は、二次元関数ですので、plotを使用します。プロットさせる関数は、

y=x^3

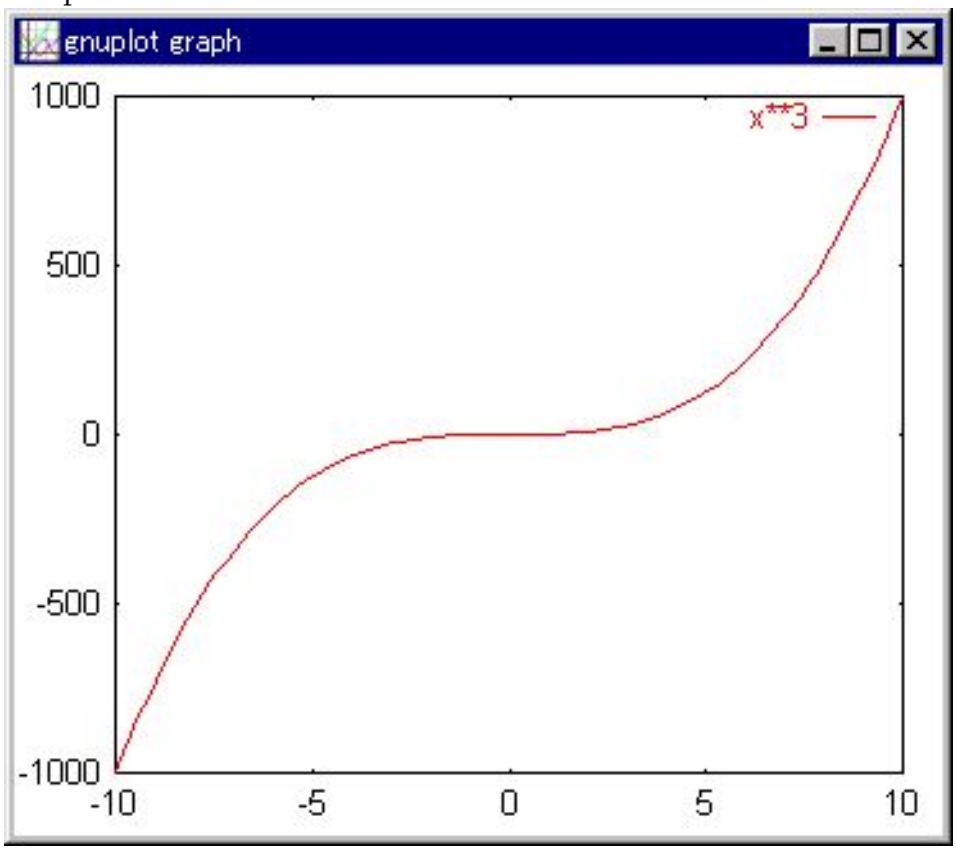
にしましょう（キャップ（　^　）は累乗をあらわします。）。もっと複雑な関数を書いてみ

てもいいのですが、出力が正しく得られることを確認したので、あえて簡単な関数にしました。
gnuplot>の後に続いて次のように記述してEnterキーを押してください。

```
plot x**3
```

そうすると、新しいウインドウが開いて、見事３次関数がプロットされるはずです。このように累乗を表すときは**を使用します。

sample1.1の出力



ついでですから、３次元の関数もプロットしてみましょう。関数は、例によってなんでもいいのですが、ここでは次の関数をプロットさせることにします。

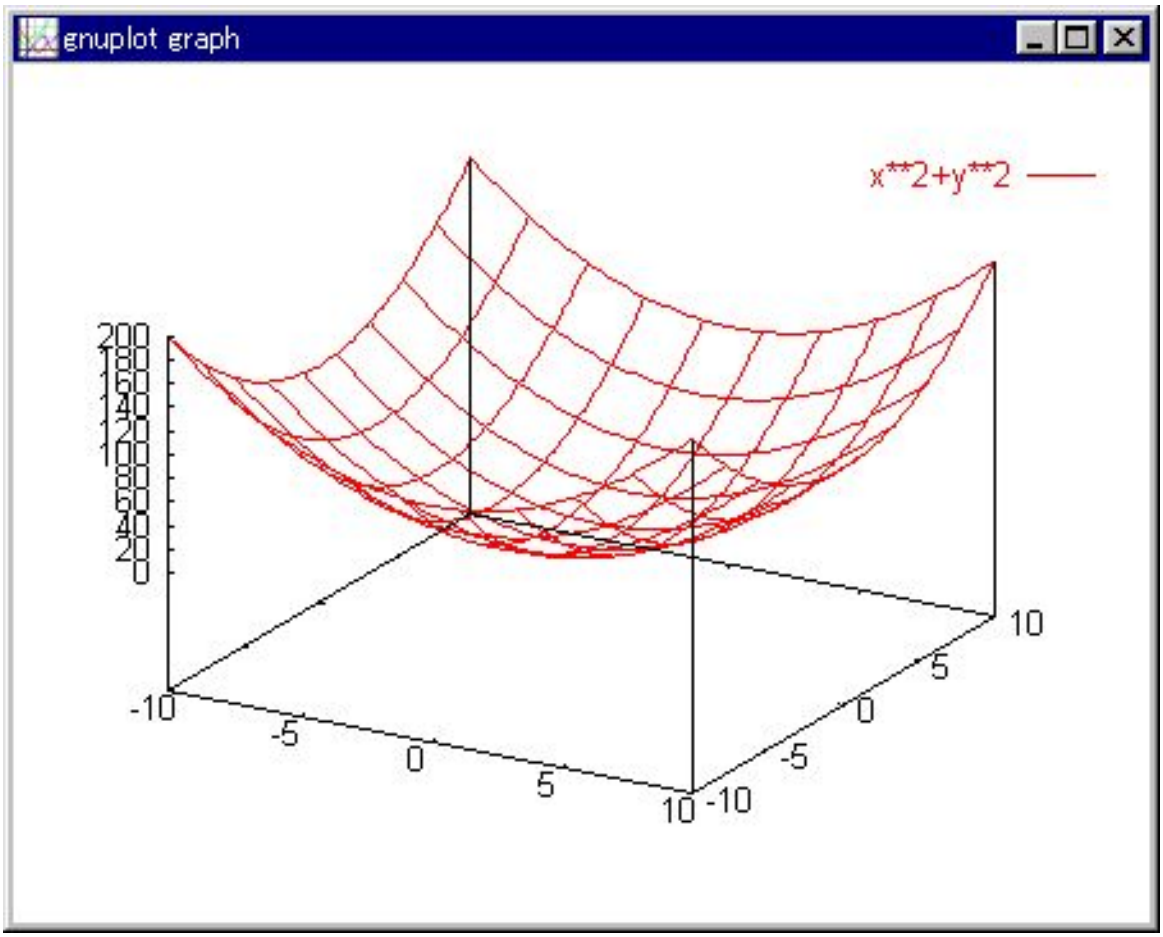
$$z=x^2+y^2$$

今度は３次元のプロットですので、plotではなくsplotを使用します。使用の仕方はplotと同様です。

```
splot x**2+y**2
```

出力は、次のようになりましたか？

sample1.2の出力

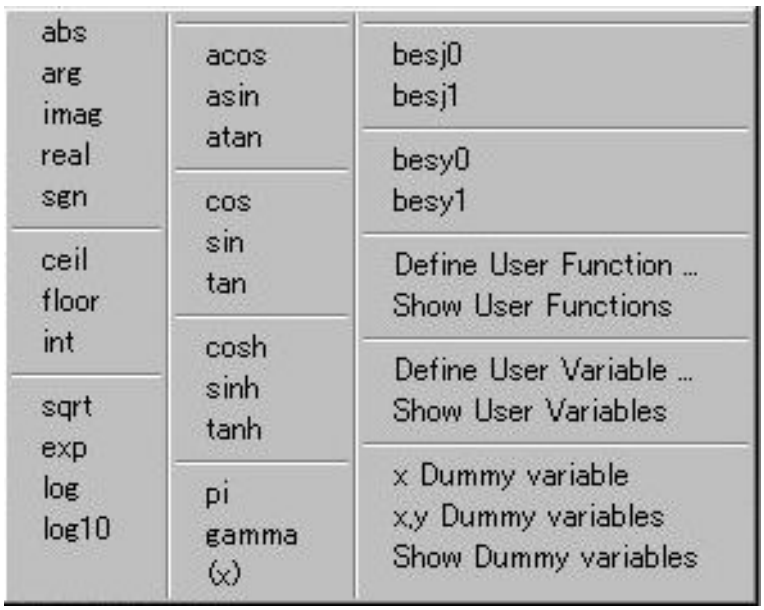


indexへ

1.2　関数を使ってみよう

　ここでは、定義されている関数をプロットしてみましょう。関数とは高校の数学の教科書にあったsin（サイン）とかcos（コサイン）とかlog（ログ）とかというヤツです。gnuplotでどのような関数が使えるかは、メニューバーのFunctionsを開くと、一覧が得られます（下図参照）。

Functionメニューで開かれるメニュー

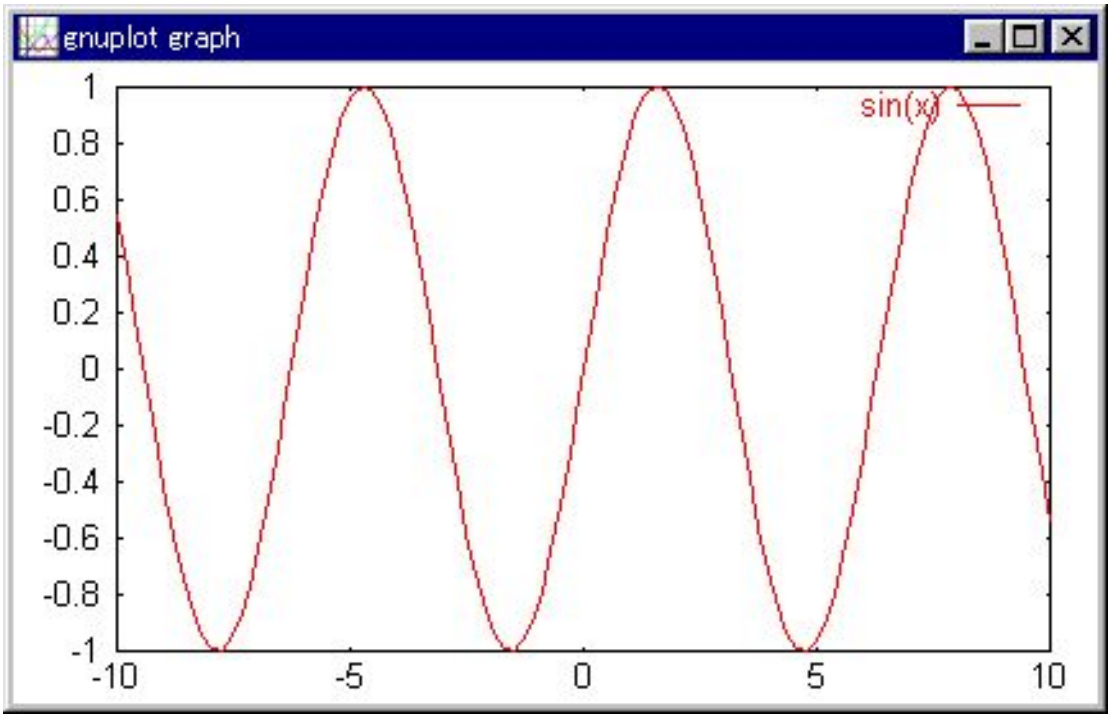


それぞれの、関数の意味については、マニュアルを参照してください。このFunctionメニューから関数を選ぶと、その関数名がgnuplotのコマンドラインに追加されます。

ここでは、試しにsinのグラフをプロットしてみましょう。次に示すコマンドを記述してEnterキーを押してください。

```
plot sin(x)
```

そうすると、次のようにsin関数（正弦波）が描画されるはずです。



ここでは、sin(x)と書きましたが、２次元プロットの場合、断りなく使える変数はxだけと覚えて置いてください。例えば、sin(y)というグラフをプロットさせようとすると、"yが定義されていません"という意味のメッセージが表示されるはずです。ただし、変数を書くべきところに値を書く場合は、全く問題はありません。たとえば、sin(2)というグラフをプロットさせると、直線のグラフが出力されます。
同様の意味で、xを使った関数なら次のようなグラフもプロット可能です。

```
plot sin(1/(x**2+1))
```

出力は自分で確かめてみてくださいね。

indexへ

1.3　関数を定義してみよう

1.2では既に用意されている関数をプロットしてみました。ここでは、自分のつくった関数を定義してみましょう。
関数を定義しておくと、その関数名を入力するだけでsin(x)をプロットしたときのように、そのグラフが即座に描画させることができます。この機能は、関数が長く、何度も書かなくてはならないときに便利ですね。また、関数を定義しておけば、sin(2)やsin(1/(x++2+1))のように変数にさまざまな値を指定してプロットさせることが可能です。この機能は、定型の関数に様々な値を代入してプロットさせるときに便利ですよね。

さっそく、関数を定義してみましょう。例ですから、簡単な関数をあげておきました。次のようにコマンドラインに書き込んでください。

```
f(x)=sin(2*x)
```

これで、f(x)という関数名にsin(2x)という関数が定義されました。次に、この定義した関数をプロットしてみましょう。

```
plot f(x)
```

きちんとf(x)=sin(2x)のグラフがプロットされましたか？試しにf(1/x)やf(x**2)なども試してみると面白いでしょう。

目次へ
第２章　任意のグラフをプロットする　へ

第2章
任意のグラフをプロットする

indexへ

2.1　gnuplotのプロット方法

gnuplotでは次にあげる幾つかのプロット方法があります。

- コマンドラインに直接関数を入力する（参：1.2「関数を使ってみよう」）
  - 数式で表せる関数しかプロットできない
  - 教科書や参考書などのグラフ作成には向いている
- データファイルを別途用意し、そのデータに基づくグラフをプロットする
  - データファイル次第で、任意のグラフをプロットできる
  - 実験データのプロットなどはこれが便利
  - sinやcosなど関数であらわせるものをプロットするのには向かない

どちらの方法も各々に長所と短所があります。コマンドラインに直接関数を入力する方法は、前章で既に紹介しましたので、本章ではデータファイルをつくって任意のグラフをプロットする方法を紹介しましょう。

indexへ

2.2　任意のグラフをプロットする

なによりもまず、やってみるのが一番です。テキストエディタを開き次のように書き込んで、適当な名前で保存してください。ただし、拡張子は.datを指定してください。

```
1 10
2 15
4 7
10 12
13 9
```

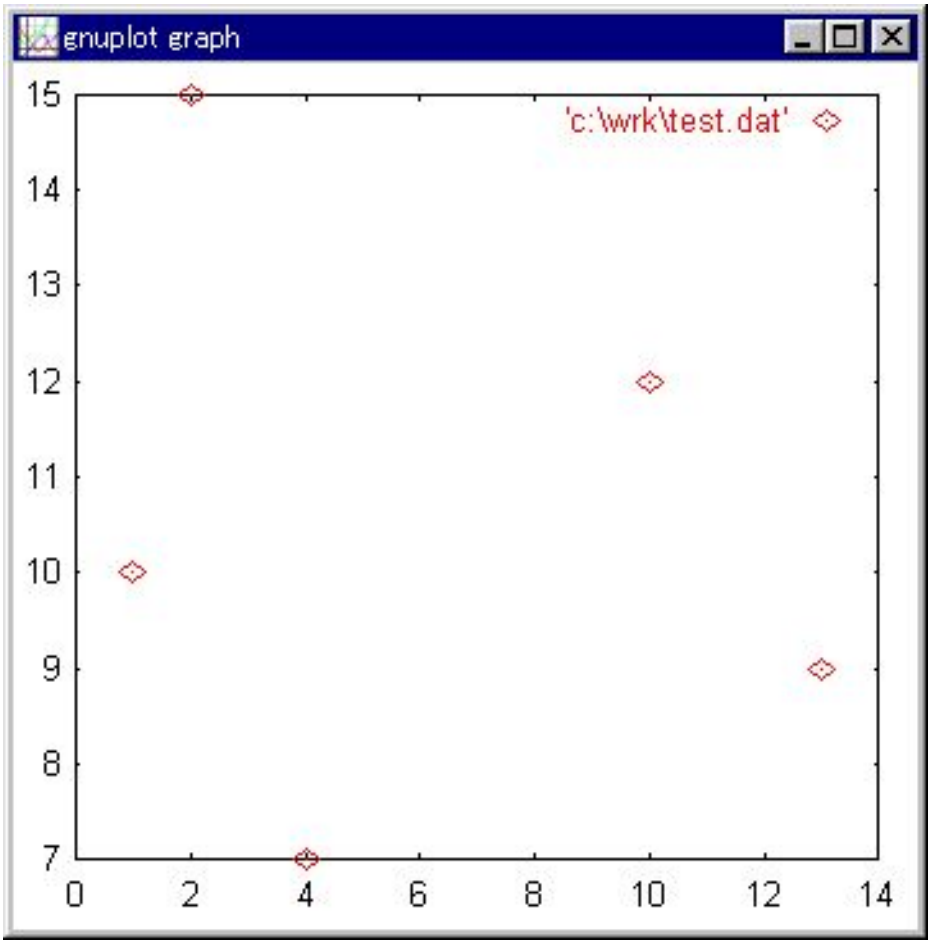
別に数値はなんでも（少数でも）構いません、行数も何行合っても構いませんが、列数は必ず２列にしてください。また、各行の１つ目と２つ目の数値の間には半角空白（Tabでもよいようなのですが）を挿入してください。今作成してもらったデータファイルの内容は、１列目がx軸（横軸）、２列目がy軸（縦軸）のあたいです。すなわち、このデータファイルによって、

(1,10)、(2,15)、(4,7)、(10,12)、(13,9)

の点が順にプロットされます。
　これで、データファイルは完成しました。次に、これをgnuplotに読み込ませて、実際にプロットさせてみましょう。gnuplotを起動し、次のようにコマンドを打ち込みます。なお、*file*の部分は先に保存したデータファイルの所在をフルパスで入力するか、cdコマンド（或いはメニューバーの"ChDir"）によってディレクトリを移動させて置いてください。

plot '*file*.dat'

そうすると、次のような出力が得られるはずです。



とりあえず、データファイルの座標がプロットされていることを確認できました。しかしながら、このグラフにはいろいろと不満があるはずです。例えば、y軸が0から始めたいとか、プロットした点を線（点線）で結びたいとか、各軸にラベルをつけたいとか...、いいだせばきりがないですね。こういった要求は全てかなえることができます。以下の節で、これらを順に説明してゆきましょう。

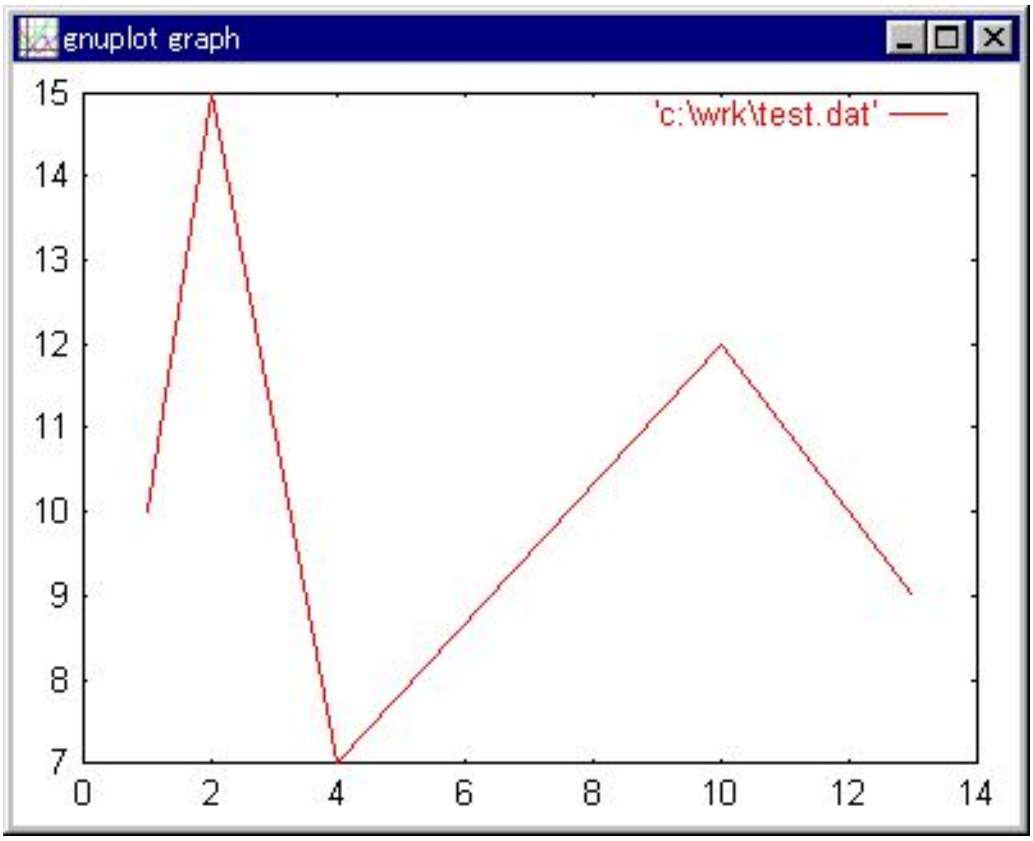
indexへ

2.3　グラフのプロット形式を変更する

さて、2.2でプロットした座標を線で結んで折れ線グラフにして見ましょう。
プロットした座標を線で結ぶにはgnuplotに次のように入力します。

plot '*file*.dat' with lines

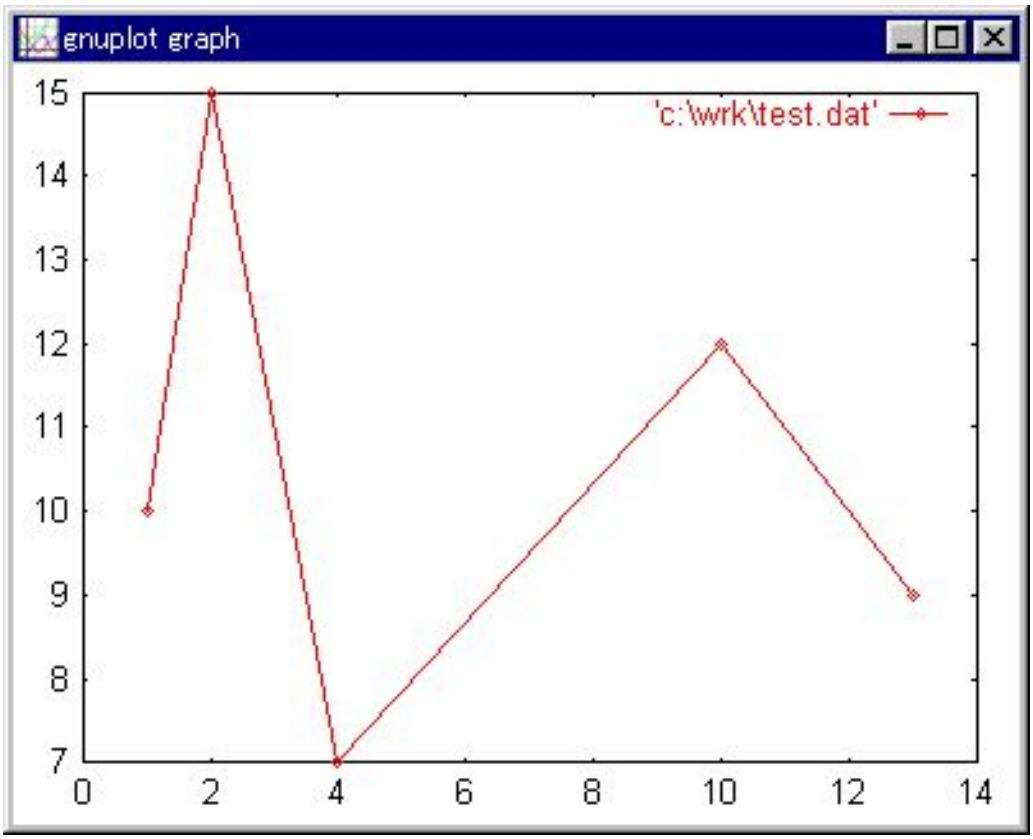
そうすると、次の出力が得られるはずです。



ところがこうすると、座標のマークが消えてしまいましたね。座標のマークがないほうがよい場合もありますが、そうでない場合は、次のようにします。

plot '*file*.dat' with linespoints

出力は次のように、座標のプロットマークとそれを結ぶ直線が描画されるはずです。



このようにwithのあとに様々なオプションを指定することによって、グラフのプロットスタイルを色々と変更することができます。次の表を参考にして、いろいろとやってみることをお勧めします。

withの後に続くキーワード	効果
boxerrorbars	boxesとyerrorbarsの合わさったもの
boxes	矩形（棒）グラフ
boxxyerrorbars	boxとxyerrorbarsの合わさったもの
candlesticks	金融（ボックス）
dots	点
errorbars	エラーバー
financebar	金融（マーク）
fsteps	ステップ関数状（起点のY値保持）
histeps	ステップ関数状（点がステップの中心）
impulses	インパルス関数状
lines	折れ線
linespoints	linesとpointsの合わさったもの
points	座標にマークをつける
step	ステップ関数状（起点のX値保持）
xerrorbars	x軸方向のエラーバー
xyerrorbars	XY方向のエラーバー
yerrorbars	Y方向のエラーバー

上記の命令の中には、デフォルトのままでは使いづらいものがあります（boxesなど）。上記の

各々のwithオプションはさらに細かな設定が可能です。それについては、追々説明したいと考えています。

indexへ

2.4　軸のカスタマイズ

次に、軸のカスタマイズをしましょう。目標は、

- y軸を0からはじめる
- y軸の上端を20にする
- xy両軸にラベルをつける

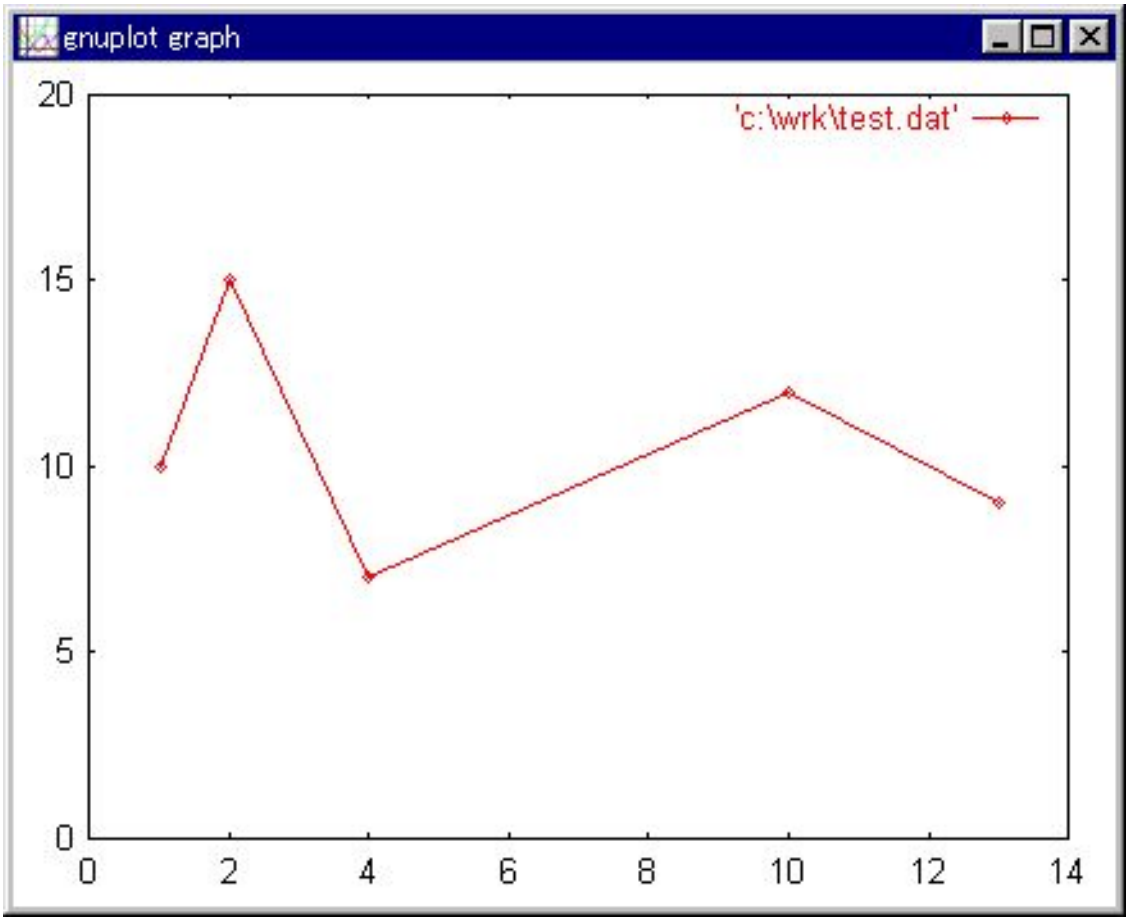
軸の開始点と終了点を変更するには、y軸の場合は

```
set yrange[0:20]
```

x軸の場合は、yrangeの変わりにxrangeとします。上記のコマンドの0は開始値、20は終了値です。したがって、このコマンドによって、「y軸が0からはじまって20で終わる」という設定がなされたわけです。それを確かめるために、再び次のコマンドでプロットしてみましょう。

```
plot 'file.dat' with linespoints
```

出力は次のようになり、確かにy軸が変更されていますね。



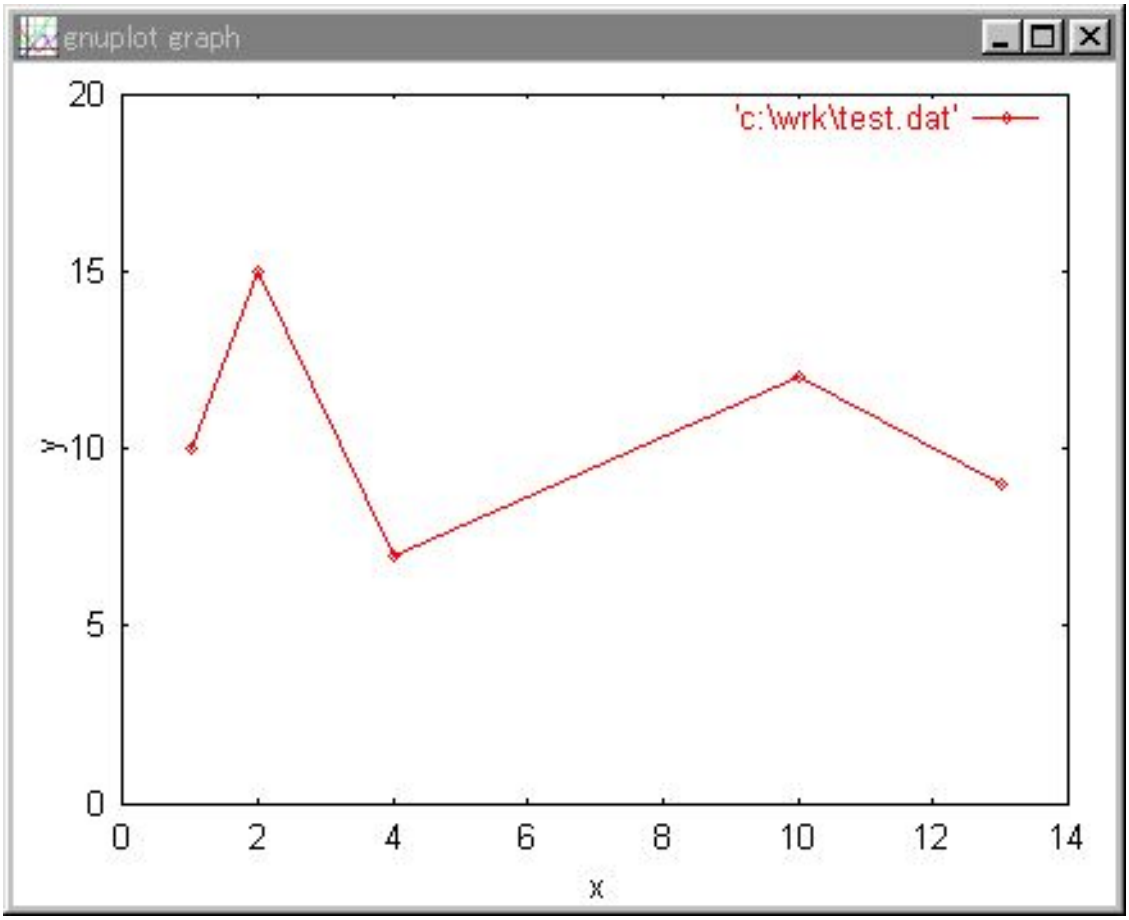
次に、x軸とy軸にラベルをつけましょう。芸がないですが、x軸には「x」、y軸には「y」とつけることにします。
軸にラベルをつける場合は、次のようにします。

```
set xlabel 'x'
set ylabel 'y'
```

その後、再びプロットさせてみてください。

```
plot 'file.dat' with linespoints
```

出力は、次のように軸にラベルがつけられたものが得られるはずです。



これを読んでおられる方のなかには、y軸のラベルがy軸の上端に出力されている方がおられるかもしれません。これは、gnuplotによるものではなく、お使いのパソコンによるものです。文字の垂直方向への出力が可能なパソコンでは上図のような出力が得られますが、そうでない場合はy軸のラベルはy軸上端に出力されます。

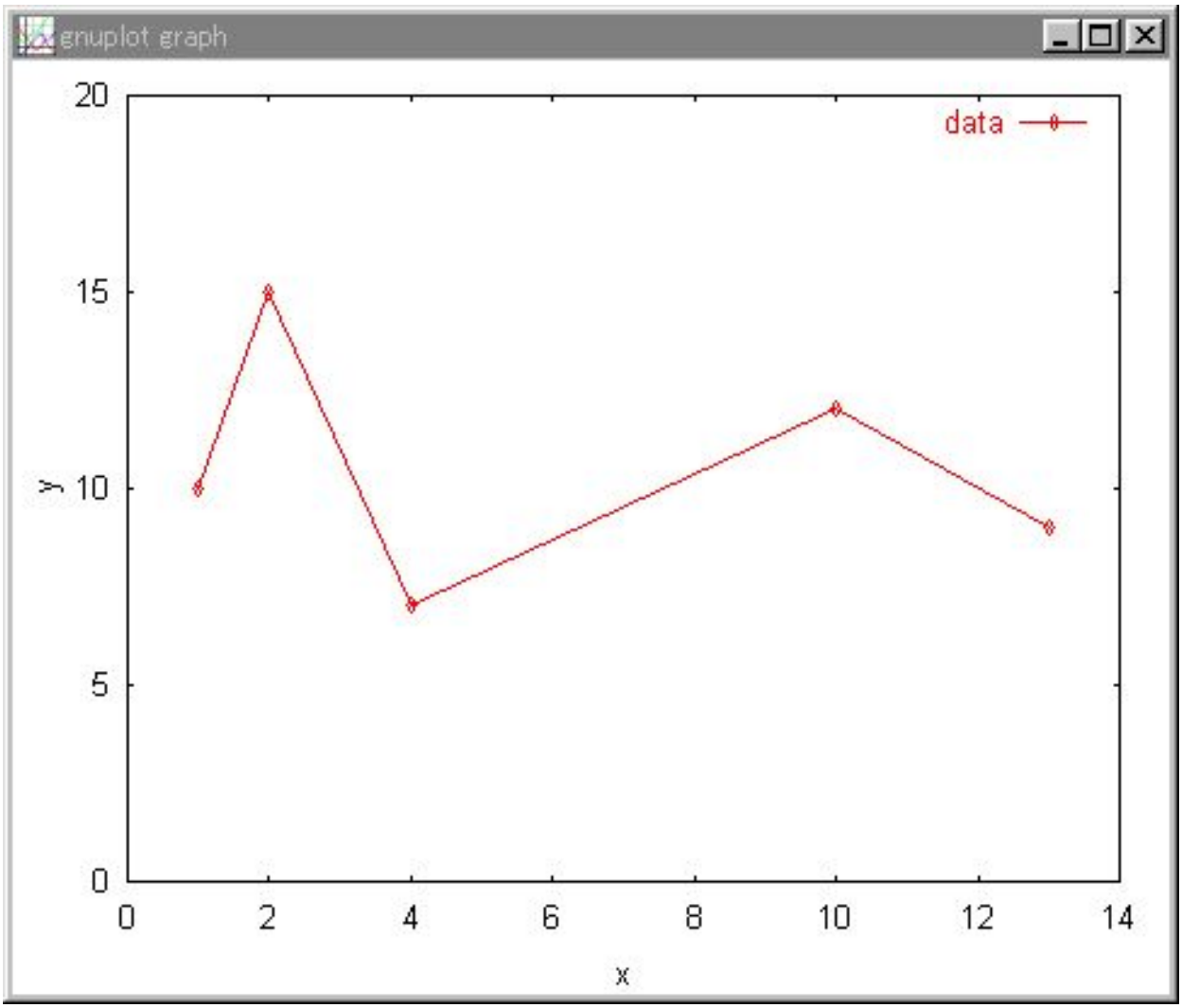
indexへ

2.5　凡例文字列の変更と非表示

本節の最後として、凡例文字列（グラフの右上の'c:\wrk\test.dat'）の変更を行いましょう。
現在、例に使用しているような一本のグラフしかないような場合、凡例は普通付けませんが、練習ですからそこは目をつむりましょう。凡例の文字列を変更する場合には、plotコマンドを入力する際に同時に行います。ここでは、「data」という凡例名をつけることにします。

```
plot 'file.dat' title 'data' with linespoints
```

上記のようにデータファイル名の後にtitle '凡例文字列'とつづることによって、次のように凡例の文字列を変更することが可能です。



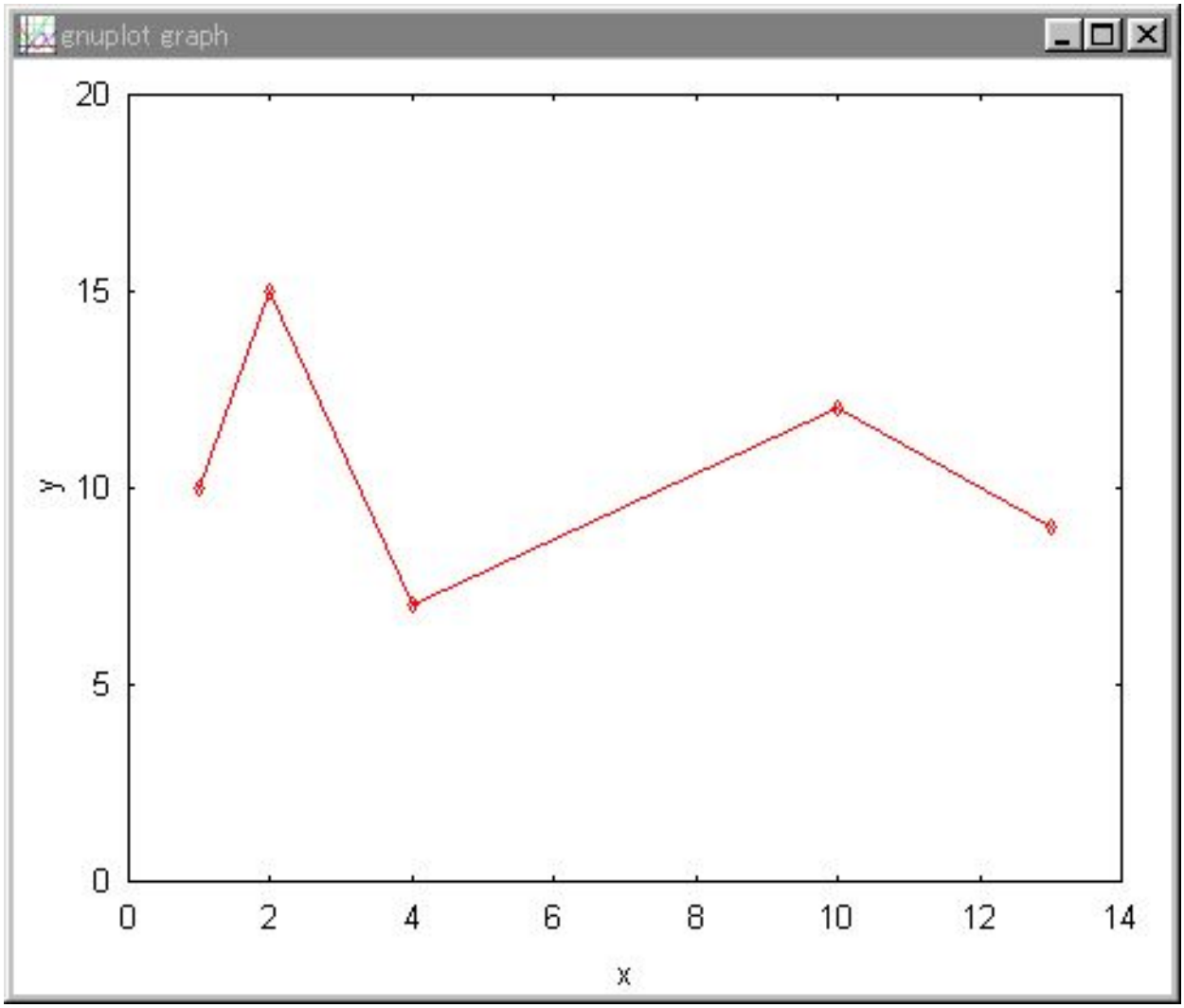
本節の冒頭にも述べましたが、このように一本のグラフの場合、通常は凡例を付けません。こういう場合は、次のようにして凡例の表示をOFFにできます。

```
set nokey
```

逆に凡例の表示を再びONにしたい場合には、

```
set key
```

を入力します。



これで、だいぶグラフらしいグラフが作成できるようになったはずです。

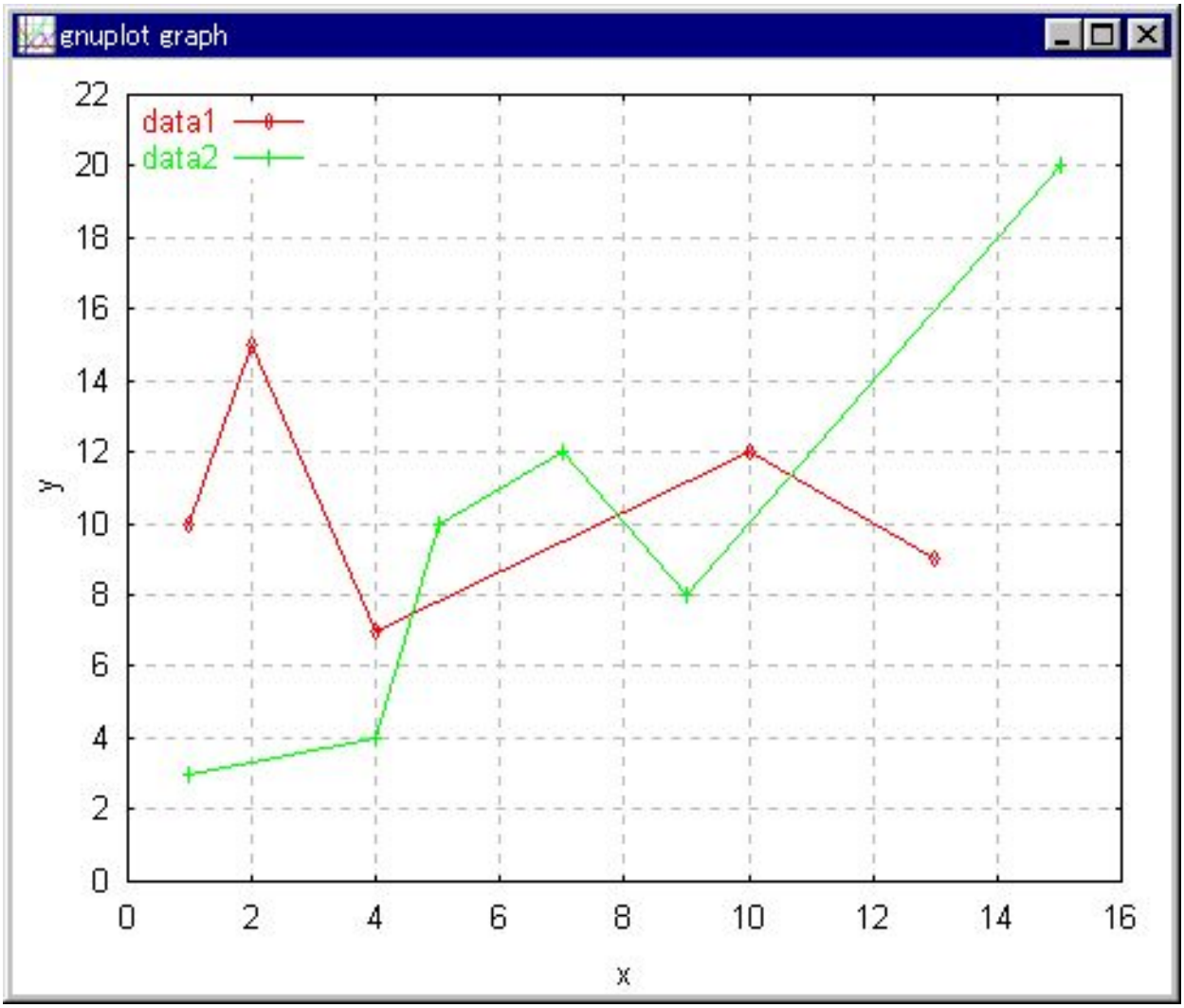
目次へ
第１章　「基本的な使い方」　へ
第３章　「グラフを重ねて表示する方法と幾つかの調整」　へ

第3章
「グラフを重ねてプロットする」方法と幾つかの調整

indexへ

3.1　データファイルを二つ用意する

２章までの内容で、一本のグラフについてはほぼプロットできるようになりました。次に、２本のグラフを重ねてプロットしてみましょう。具体的には、次のようなグラフを作ることが目標です。



データファイルには次の２つを使用しました。

test1.dat	test2.dat

1 10 2 15 4 7 10 12 13 9	1 3 4 4 5 10 7 12 9 8 15 10

上記のようなグラフを得るためにどのような設定が必要か考えてみましょう。

1. 軸ラベルは、x軸が「x」でy軸が「y」
2. x軸は0〜16（自動的に設定されるかもしれない）
3. y軸は0〜22（自動的に設定されるかもしれない）
4. 凡例は表示する（デフォルト）
5. y軸の目盛りの刻み幅を２にする
6. デフォルトのままの凡例の出力位置では、グラフとかぶってしまうので、位置をずらす
7. グリッドを表示する

ここで示した1から4については、第２章ですでに説明済みですので、ここでは省きます。

indexへ

3.2　２つのグラフを重ねて表示する

２つのグラフを表示するのは、それほど難しくありません。方法は、plotコマンドのあとに必要なだけファイルを並べるだけです。

```
plot 'file.dat' title 'data1' with linespoints, \
'file.dat' title 'data2' with linespoints
```

上のコマンドには見かけないものがありますね。そう、\マークです。gnuplot上では\マークはバックスラッシュで表示されますが、同じ意味です。コマンドが長くなる場合などは、スペルミスなどが起きやすいので、上記のようにして２行にわけて書くことができます。２行に分ける場合は、その行の最後に\（バックスラッシュ）を入れます。このとき、次の行頭は、gnuplotの文字が消えプロンプト（>）だけになりますが、びっくりしなくても大丈夫です。

もちろん、下のように、一行で入力をしても全く問題はありません。

```
plot 'file.dat' title 'data1' with linespoints,'file.dat' title 'data2' with linespoints
```

indexへ

3.3　軸の目盛りの幅を変更する

まず、y軸の目盛りの刻み幅を変更してみましょう。これは、次のようにyticsのの値をsetし直すことによって実現できます。

set ytics 2

このことから容易に想像できると思いますが、

set xtics 5

などとやれば、x軸の目盛りの刻み幅を５に設定できます。

indexへ

3.4　凡例の位置を変更する

凡例の位置を変更するには、

set key left

とします。このようにすることで、凡例の位置がグラフの内側左に移動させることができます。
このset keyの後に続けることのできるオプションには次のようなものがあります。

set key	場所
top	グラフの内側上
bottom	グラフの内側下
left	グラフの内側左
right	グラフの内側右
outside	グラフの外側右
below	グラフの外側下

これらのオプションは矛盾がない範囲で組み合わせて使用できます。例えば、

set key left　bottom

と指定すると、凡例はグラフの内側左下になります。当然、bottomとtopを併用したり、rigthとleftを併用することはできません。

indexへ

3.5　グリッドを表示する

グリッドを表示するのは簡単です。

```
set grid
```

とするだけです。また、グリッドを表示した状態から非表示の状態にするには、

```
set nogrid
```

とします。

目次へ
第２章　「任意のグラフをプロットする」　へ
第４章　「ファイルへ出力する」　へ

第4章
ファイルへ出力する

indexへ

4.1　gnuplotが出力できる形式

gnuplotは多くの出力形式をサポートしています。３章までに示した例では、全て「gnuplot graph」というウインドウが開かれ、そこにプロットされていました。これは、いわば出力先が画面だったということです。しかしながら、多くの場合は出力先は画像ファイルやプリンタでしょう。

gnuplotが出力先としてサポートしているものは、次のようなものがあります。

指定	形式
windows	PLT（独自形式、再編集が可能）
postscript	PoscScript
gif	GIF
latex	LaTeX
fig	FIG
mf	METAFONT
pbm	PBM
x11	X

他にもまだありますが、ここでは取り上げません。ヘルプなどを参考にしてみてください。

indexへ

4.2　ファイルへの保存方法

ファイルへ保存するためには、幾つかの手順を踏まなくてはなりません。

1. 出力形式を指定する
2. 保存先ファイルを用意する
3. 保存先ファイルに書き込む

まず、第２章や第３章の方法に従って、適当なグラフを「gnuplot graph」ウインドウに表示させておいてください。次にgnuplotに戻り、次のコマンドを入力してください。このコマンドは出力をGIF形式で保存するという指定です。

```
set terminal gif
```

すると、次のような２行が自動的に流れるはずです。

```
Terminal type set to 'gif'
Options are small size 640,480
```

内容についてはあまり気にする必要はないでしょう。要は、「きちんと入力されました」みたいな事です。もし、「set terminal gif」の入力時にスペルミスなどがあれば、エラーが指摘されます。

さて、GIF形式で出力する指定ができたら、次に保存先ファイルを指定します。

```
set output 'file.gif'
```

*file*の部分には、保存するファイル名を指定してください。このとき、cdコマンドで保存先ディレクトリに移動していない場合は、ディレクトリを移動するか*file*をフルパスで指定してください。

最後に、plotコマンドによって用意した出力ファイルに書き込みます。plot命令の使用の仕方については、第２章、第３章をご覧ください。

```
plot 'file.dat' with linespoints
```

以上で保存作業は終わりです。指定したディレクトリに指定した保存ファイルができているはずです。出力ファイルを開いて出力結果が正しく書き込まれているか確認してください。

ここでは、全て手作業による保存方法を紹介しましたが、メニューバーのSaveボタンを押すことによって、２、３の手順を自動化してくれます。このとき、Saveボタンを押す前に、出力形式を選択しておくのを忘れないでください。

ここでは、GIF形式を例に出力の保存方法を紹介しましたが、4.1にあげた各保存形式には、いくつかのオプションを指定することができます。それに関しましては今後別の章にまとめるつもりです。

indexへ

4.3　plt形式での保存と再利用

　plt形式での保存は再びこのpltをgnuplotで読み込むことによって再編集が可能です。plt形式での保存方法は、terminalをwindows（gnuplot起動時のデフォルト）にセットして、保存をします。具体的には、

```
set terminal windows　←起動してからterminalを変更した場合は必要
Terminal type set to 'windows'
Options are 'color "Arial" 10'
set output 'file.plt'
plot 'file.dat' with linespoints
```

そして、再びこの*.pltファイルを読み込む場合には、

```
load 'file.plt'
```

と入力します。

目次へ
第３章 「グラフを重ねて出力する方法と幾つかの調整」 へ
第５章 「ファイルへ出力する(EPS編)」へ

第5章
ファイルへ出力する(EPS編)

indexへ

5.1　PostScript形式で出力するメリット

　gnuplotで出力したグラフをpLaTeXに貼り付けたい場合には、EPS形式で出力することを特にお勧めします。理由は、pLaTeXがEPSと非常に愛称がよいといからです。pLaTeXで作る書類にgifやpbmの画像を貼り付ける場合には、BoundingBoxファイルを指定するか、\includegraphics命令のオプション引数に画像のサイズを指定しないと、縦横比が保持されません。一方、EPS形式で保存された画像ファイルは画像のサイズをpLaTeXが読める形式で保持していますので、こういった操作は不要になります。EPS形式は拡大縮小につよく、またAdobe Illustratorなどで再編集できるという点も大きなメリットです。
　本章では、こういった背景からEPS出力について特別に取り上げて説明をします。

indexへ

5.2　PostScript形式で指定できるオプション

　PoscScriptを出力形式に選んだ場合、他の出力形式と異なり、多くのオプションを指定できます。それらを下の表に示します。

オプション	機能
landscape	ランドスケープ(デフォルト)
portrate	ポートレイト
EPS	Encapusulate PostScript
enhanced	PostScriptの拡張テキストコントロール
noenhanced	enhancedの無効化(デフォルト)
color	カラー
monoclome	モノクロ(デフォルト)
solid	実線(デフォルト)
dashed	波線
フォント名	デフォルトはHelvetica
フォントサイズ	デフォルトは14pt
defaultplex	デフォルト印刷
simplex	片面印刷
duplex	両面印刷

それぞれ色分けしたパラメータは、同時には指定できないものです。
これらのオプションの指定の仕方は、gnuplotのヘルプによると、

```
set terminal postscript {<mode>} {enhanced|noenhanced}
                        {color|monoclome} {solid|dashed}
                        {<duplexing>}
                        {"<fontname>"} {<fontsize>}
```

とかかれています。<mode>には、表中のlandscapeかportrateかepsを指定します。次の{enhanced | noenhanced}のように"|"で仕切られているパラメータは、そのうちの一つを選べということです。そして、５番目の{<duplexing>}にはdefaultplexかsimplexかduplexを選びます。{"<fontneme>"}と{<fontsize>}には文字通りフォント名とフォントサイズを指定します。これらの、オプション引数は全てデフォルト値が設定されているので、省略が可能です。しかしながら、Adobe Illustratorで編集したり、pLaTeXに貼り付けるということを考えている場合、次のことに注意してください。

- <mode>にはepsを指定する
- enhancedは指定しない

<mode>にepsを指定しなくても特に困ったこと(表示できないなど)は起きませんが、epsの指定をした出力が一番使いやすい出力だと思います。landscapeやportraiteの指定での出力と比べてみてください。
また、enhancedを指定すると、例えばx軸の見出しにx^2と指定すると、指数（xの肩に2が乗る）表示が可能になりますが、残念ながらこの機能はgnuplot独自のものらしく(?)Adobe IllustratorやpLaTeXではエラーが出て表示させることすらできません（GViewでは表示可能なので、本当はできるのかもしれません）。

indexへ

5.3　PostScript形式で出力する方法

それでは、実際に5.2を参考にしてPoscScript形式での出力をさせてみましょう。
gnuplotのコマンドラインで、次のように入力します。

```
set terminal postscript eps
```

そうすると、

```
Terminal type set to 'postscript'
Options are 'eps noenhanced monochrome dashed defaultplex "Helvetica" 14'
```

という文字が流れます。よく見ると、２行目には4.2であげたデフォルト値が並んでいますね。
次に、第４章で説明したように出力ファイルを指定し、plotコマンドによって出力をファイルに書き込みます。具体的には、次のコマンドです。

```
set output 'file.eps'
plot 'file.dat' with linespoints
```

また、次のようにすると、カラーで文字サイズが12ptの出力が得られます。

```
set terminal postscript eps color 12
```

今回は、出力ファイルを一度も画面に出さずにファイルへ出力させましたが、もちろん第４章で紹介した方法のように、一度gnuplot graphのウインドにグラフを出力させてからでもファイルへの保存は可能です。

なお、本節の出力では軸のカスタマイズや凡例の出力などは省略してあります。詳細は第３章、第４章を参考にしてください。

目次へ
第４章 「ファイルへの出力」 へ
第６章 「線種とマークのカスタマイズ」 へ

第6章
線種とマークのカスタマイズ

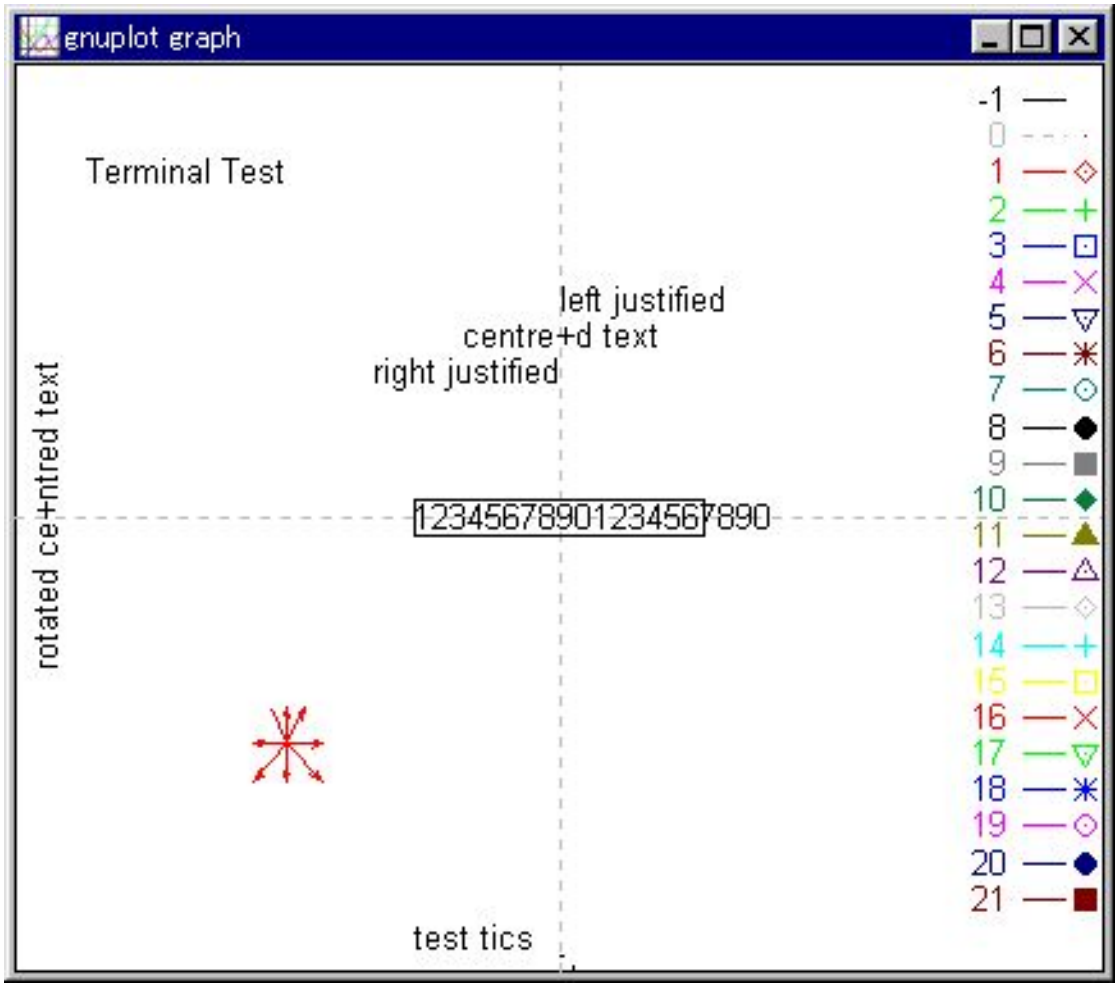
indexへ

6.1　testコマンド

　testコマンドにより各terminalごとに使用可能な線種とマークの一覧を得ることができます。試しに、次のコマンドを入力してみてください。ただし現在のterminal（出力形式）がwindowsであることを確認しておいてください。

```
test
```

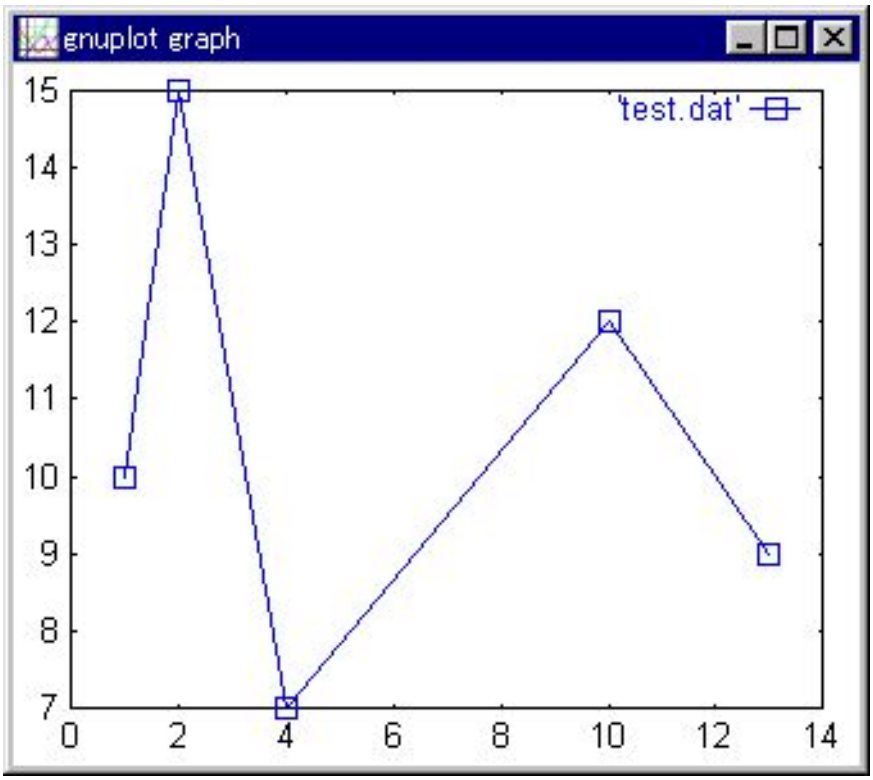
　そうすると、例によってgnuplot graphのウインドが開き、次のようなものが現れるはずです。



　この右側に出力されている線とマークの組み合わせを番号によって指定することができます。具体的には、次のようにwithオプションでプロットスタイルを指定した後に番号を指定します。

plot '*file*.dat' with linespoints 3

上の例では、linespoints（線とマーク）でプロットさせ、そのプロットスタイルを3と指定しています。testコマンドによって得られた番号と線種･マークの対応が出力に反映されていることを確認してください。



このような、線種とマークはterminalごとに別々に設定されています。terminalがwindowsの場合には、gnuplot graphが起動してくれて、testコマンドの出力を即確認できるのですが、他のterminalは一度ファイルに保存しなくては確認できません。例えば、次のコマンドはEPSの場合の例です（尚、黒字はgnuplotが自動的に表示する部分ですので、入力する必要はありません）。

```
set terminal postscript eps
Terminal type set to 'postscript'
Options are 'eps noenhanced monochrome dashed defaultplex "Helvetica" 14'
set output 'file.eps'
test
```

testコマンドの出力結果は、*file*.epsというファイルに保存されています。この内容を見るにはEPSに対応したビューアーまたはドローツールが必要です。代表的なものは、Adobe Illustratorがありますが、残念ながらこのtestコマンドの出力に限っては、Illustratorで開けませんでした（原因は不明）。そこで、こちらにAlladin GhostViewで開いた出力を紹介しましょう。Alladin GhostViewはPS形式、EPS形式に対応したビューアーです。画像がかなり荒く出力されてしまいますが、簡易ビューアーとしては重宝するものです。

ついでですので、gif形式での出力とpbm形式での出力とLaTeX形式での出力を示しますので、参考にしてください。

indexへ

6.2　マークの大きさを変更する

マークの大きさを変更するには、次のようにします。

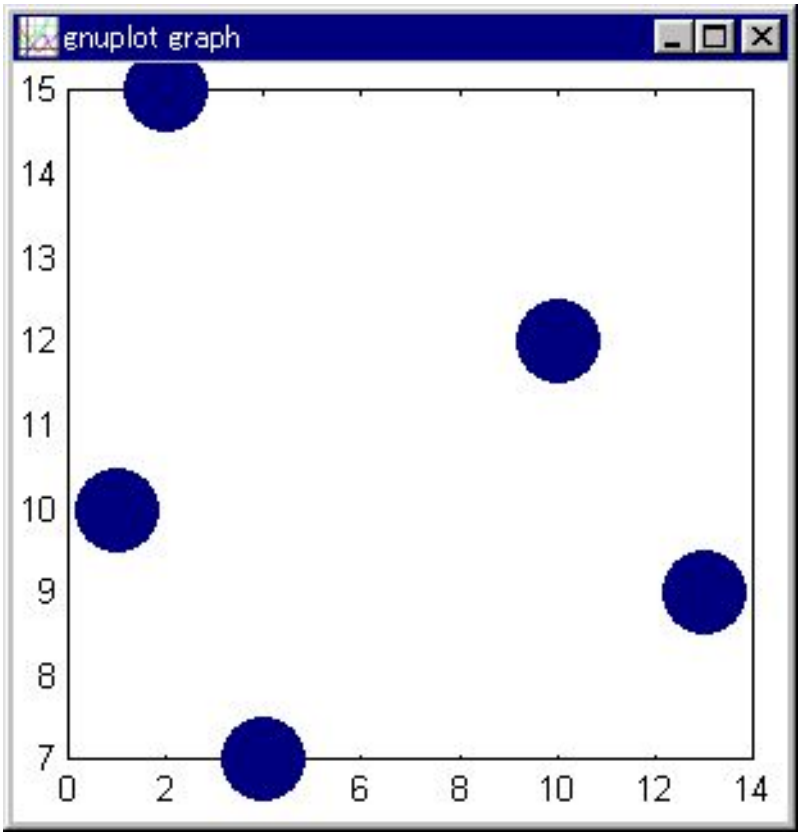
set pointsize <multiplier>

<multiplier>にはマークのサイズを指定します。デフォルトでは1.0です。例えば、次の例ではマークの大きさを5に設定しています。

set pointsize 5

そして、グラフをプロットしてみましょう。

plot '*file*.dat' with points 20



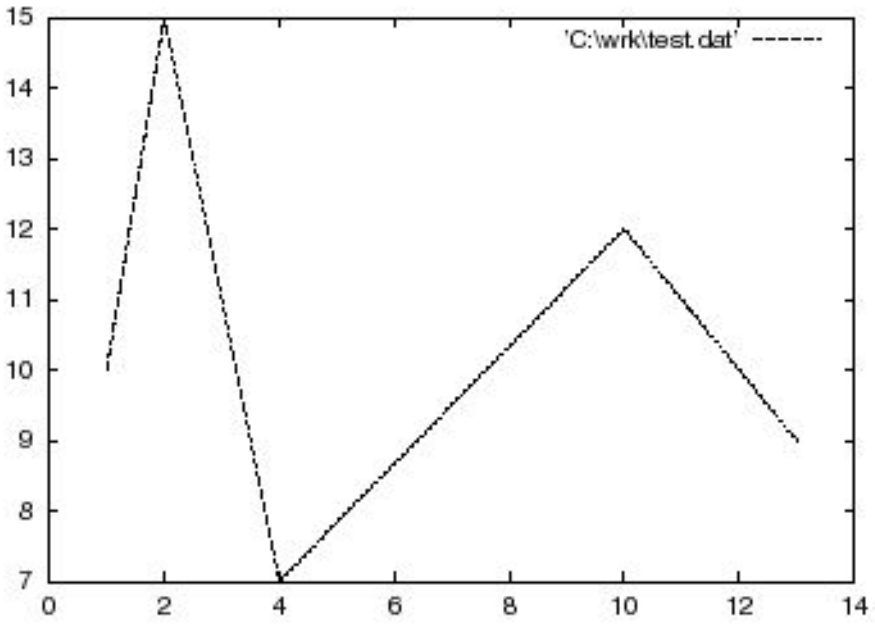
indexへ

6.3　線種を変更する

withオプションでlinesやlinespointsを指定したときには、線種を変更できます。
linesの場合には、

plot '*file*.dat' with lines 2

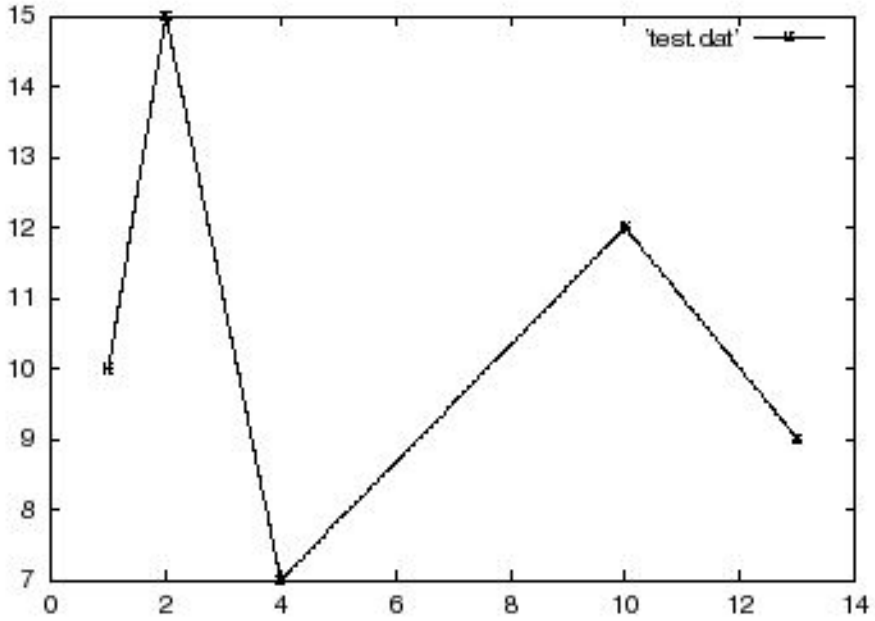
などとすると、testコマンドの出力例からもわかるように、波線でプロットすることが可能です（EPSの場合）。



また、linespointsの場合、線種とマークを別々に指定することができます。

plot '*file*.dat' with linespoints 2 3

この場合、最初の引数2が線種、次の引数3がマークの種類になります。出力は次のようになります（EPSの場合）。



目次へ
第５章 「ファイルへの出力(EPS編)」 へ
第７章　「一つのデータファイルに複数のデータをまとめる」　へ

第7章
一つのデータファイルに複数のデータをまとめる

indexへ

7.1　これからやること

gnuplotのデータファイルを利用するときには、今までは一つのデータ(グラフ)に対して一つのファイルを用意しなくてはなりませんでした。たとえば、3.2でグラフを重ねて表示する場合に、.datファイルを2つ用意しました。本章では、一つのデータファイルに複数のデータを格納し、3.2と同様のことを実現する方法を紹介しましょう。こういった要求は、一つの試料に対する複数の実験結果を一つのデータファイルにまとめておきたいという場合に有用です。

indexへ

7.2 データファイルの準備

plotコマンドとともにindexコマンドを使用すると、一つのデータファイルに複数の種類のデータを格納しておき、必要に応じて必要なデータだけをプロットすることができます。

まずはデータファイルの作成です。次のようなデータファイルを作成してください。

```
0  0
1  1
2  2
3  3
4  4
5  5
                ←空白行
                ←空白行
0  0
1  1
2  4
3  9
4  16
5  25
                ←空白行
                ←空白行
0  0
1  1
2  8
3  27
4  64
5  125
```

```
                ←空白行
                ←空白行
0  0
1  1
2  16
3  81
4  256
5  625
```

1行目～6行目を「第0セット」、9行目～14行目を「第1セット」、・・・と呼ぶことにします。各セットの間はかならず２行以上の空白行を挿入してください。ここで一つ注意しておきたいのは、セットを数えるときにはかならず第0セットから数えるということです。

indexへ

7.3　indexコマンド

それでは早速indexコマンドを使用してみましょう。

```
plot 'file.dat' index 0 with linespoints
```

このプロットの結果は第0セットだけが表示されたはずです。出力結果はここでは示しませんので、各自でお確かめになってください。
次に、第1セットから第3セットまでを表示させて見ましょう。すなわち、データファイル中の第0セットを表示させないようにします。これは、次に示すように「index 開始セット:終了セット」のようにすると実現できます。

```
plot 'file.dat' index 1:3 with linespoints
```

出力結果はグラフが3本だけ表示され、第0セットのデータが表示されていないことがわかりますね。こちらも出力結果は示しませんので、各自で確かめてください。
実は、indexにはもう一つパラメータを指定することができます。例えば、複数あるセットのうち２つおきにプロットさせたいという場合を想定したものです。このようなことを実現するためには、「index 開始セット:終了セット:ステップ」という書式でコマンドを入力します。

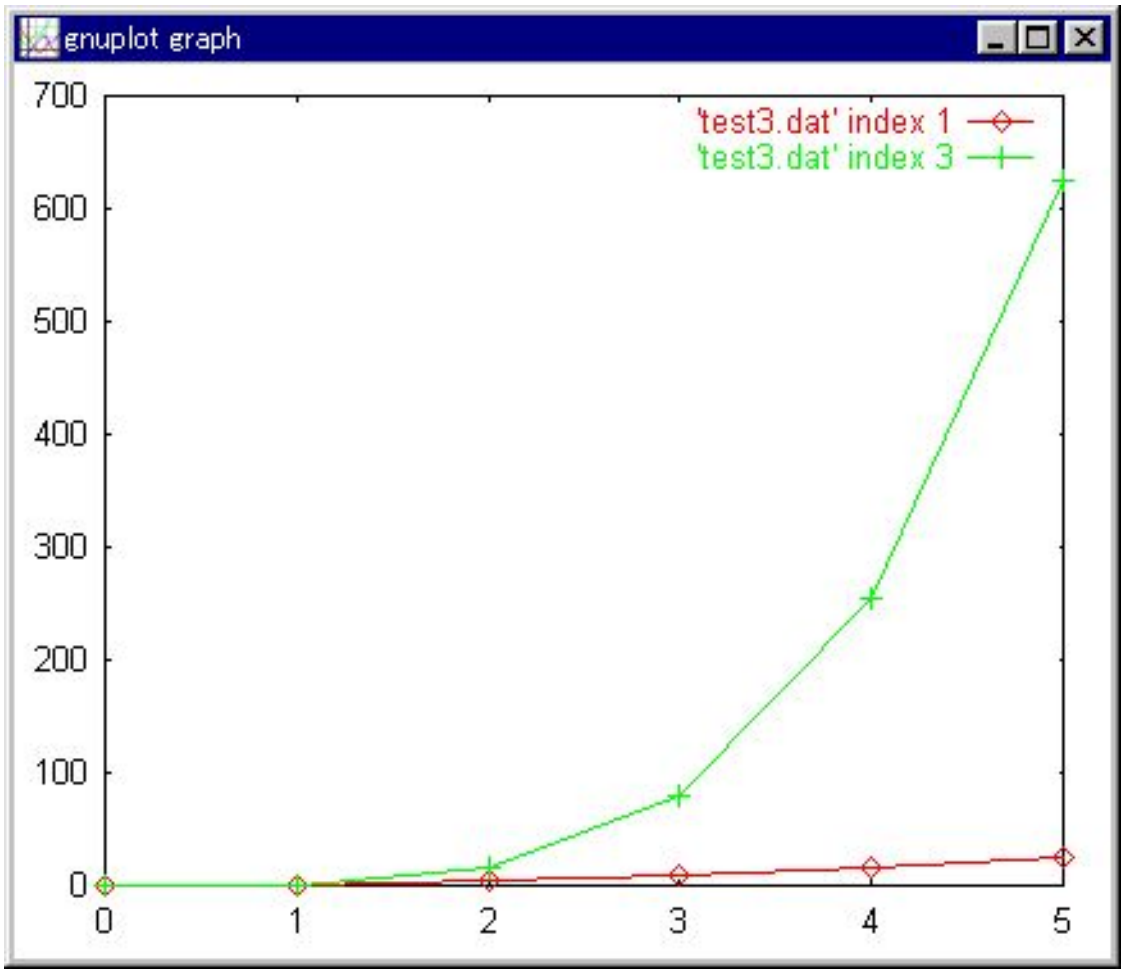
```
plot 'file.dat' index 0:100:10 with linespoints
```

このようにすると、第0セットから第100セットのうち、10個飛ばしにセットがプロットされます。すなわち、プロットされるセット番号は、「0、10、20、…、100」となります。

それでは、最後にこれらの応用例として、一つのデータファイルから２本のグラフを重ねてプロットさせて見ましょう。ここまでくればもう簡単ですね。

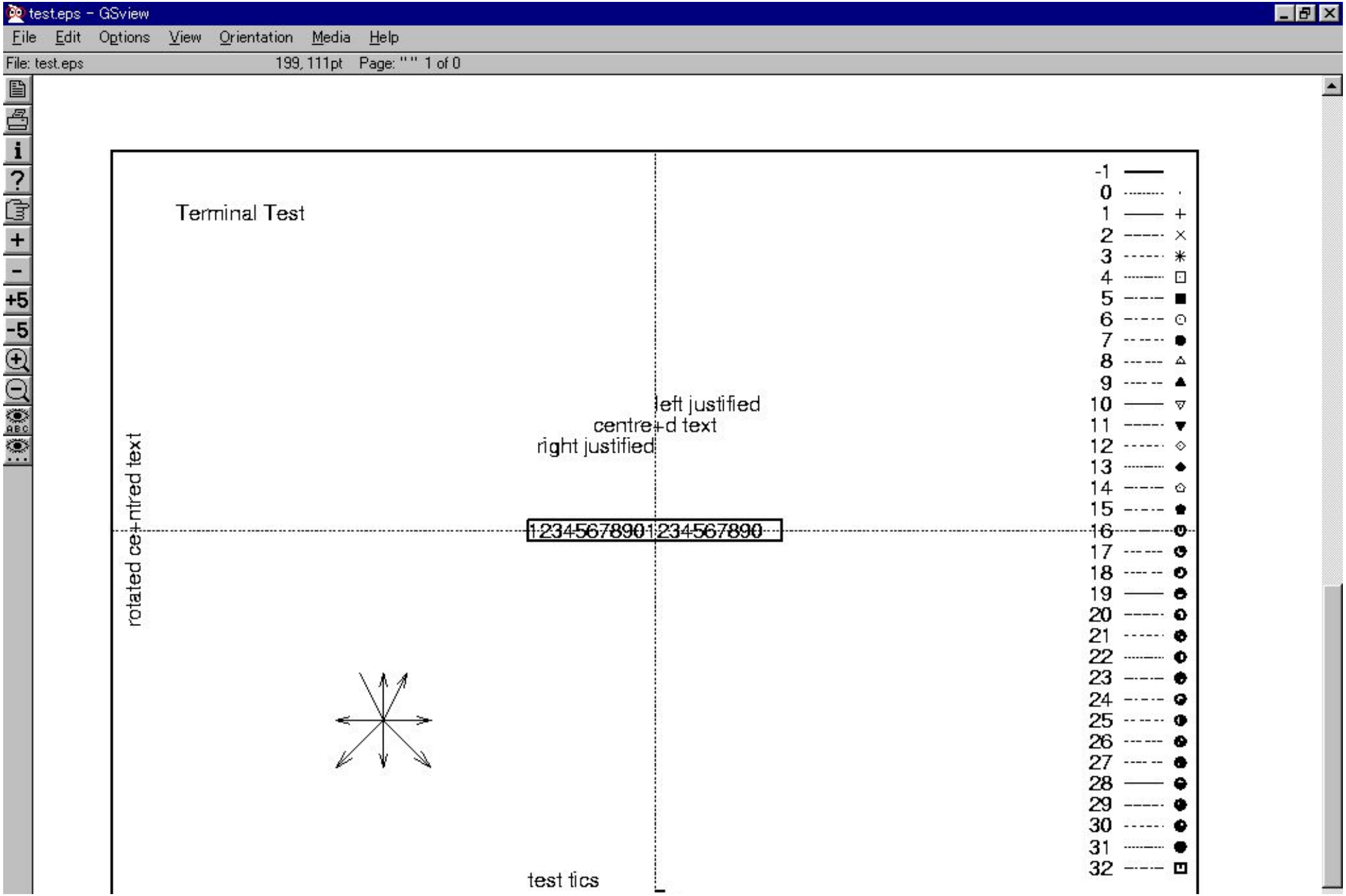
```
plo t  'file.dat' index 1 with lines,'dile.dat' index 3 with lines
```

出力は次のようになります。



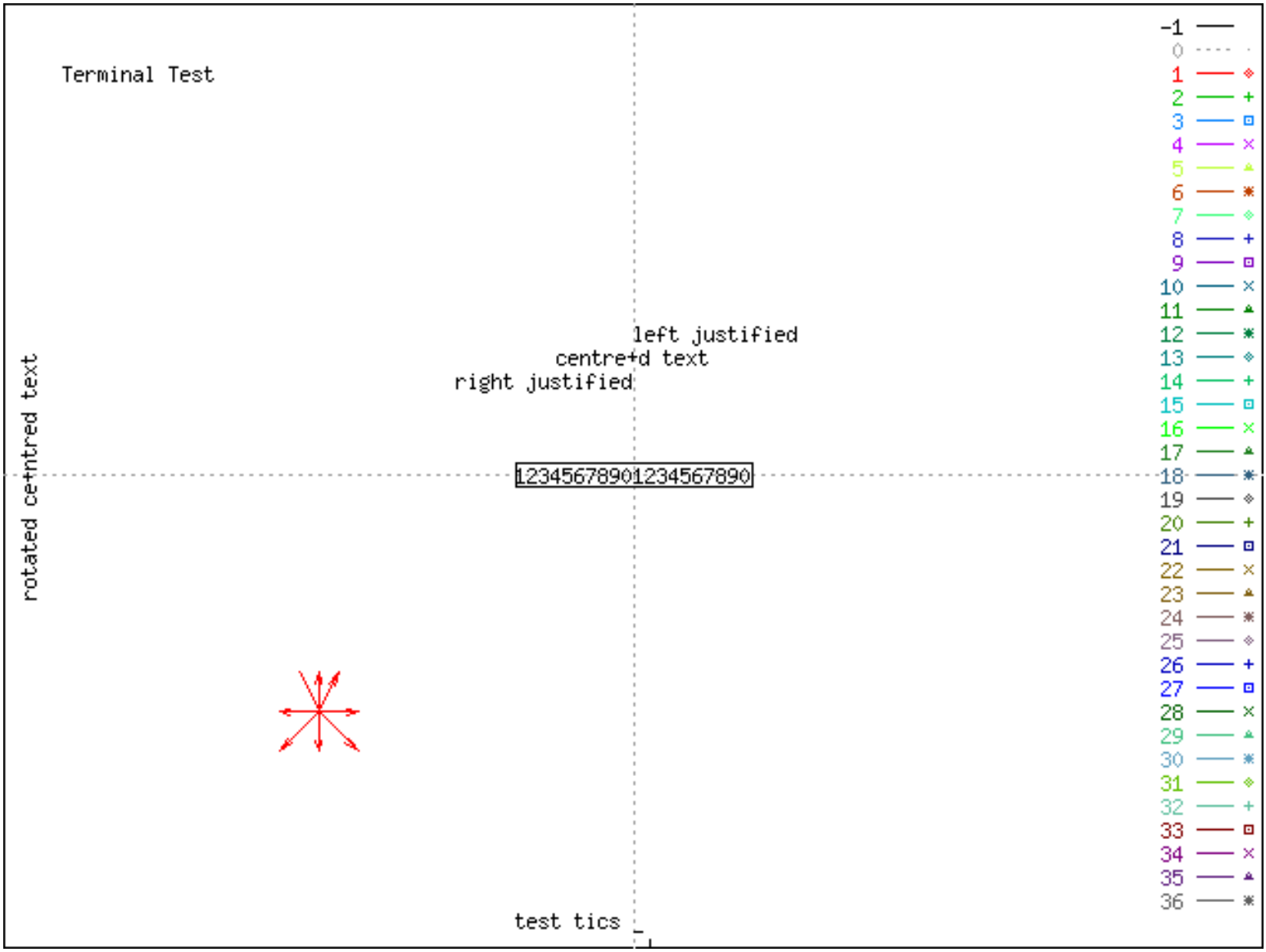
目次へ
第６章「線種とマークのカスタマイズ」へ
第８章　　へ

Alladin GhostView　でみた　testコマンドの出力(EPSの場合)。

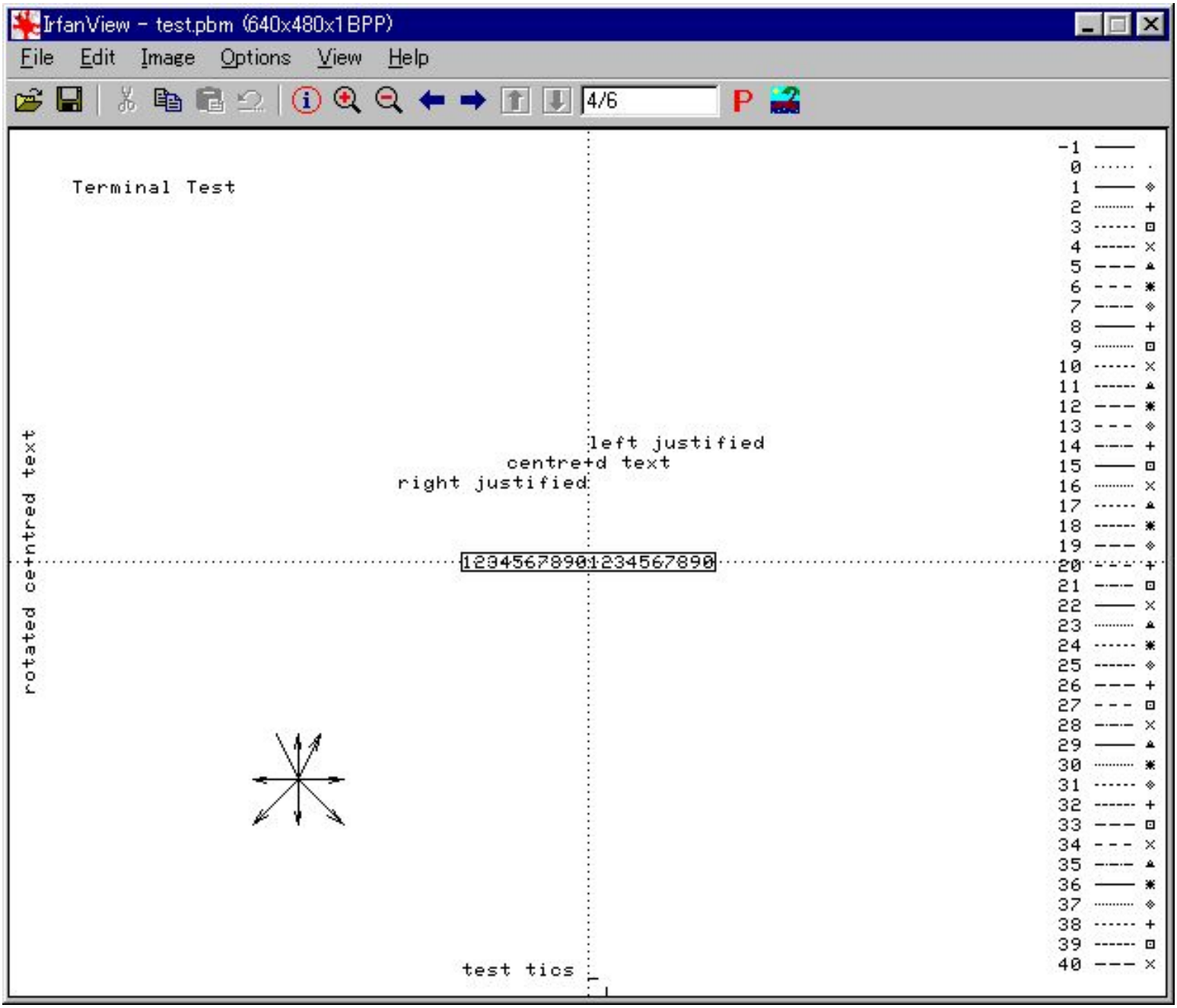


Alladin GhostViewでみたtestコマンドの出力(EPS)。

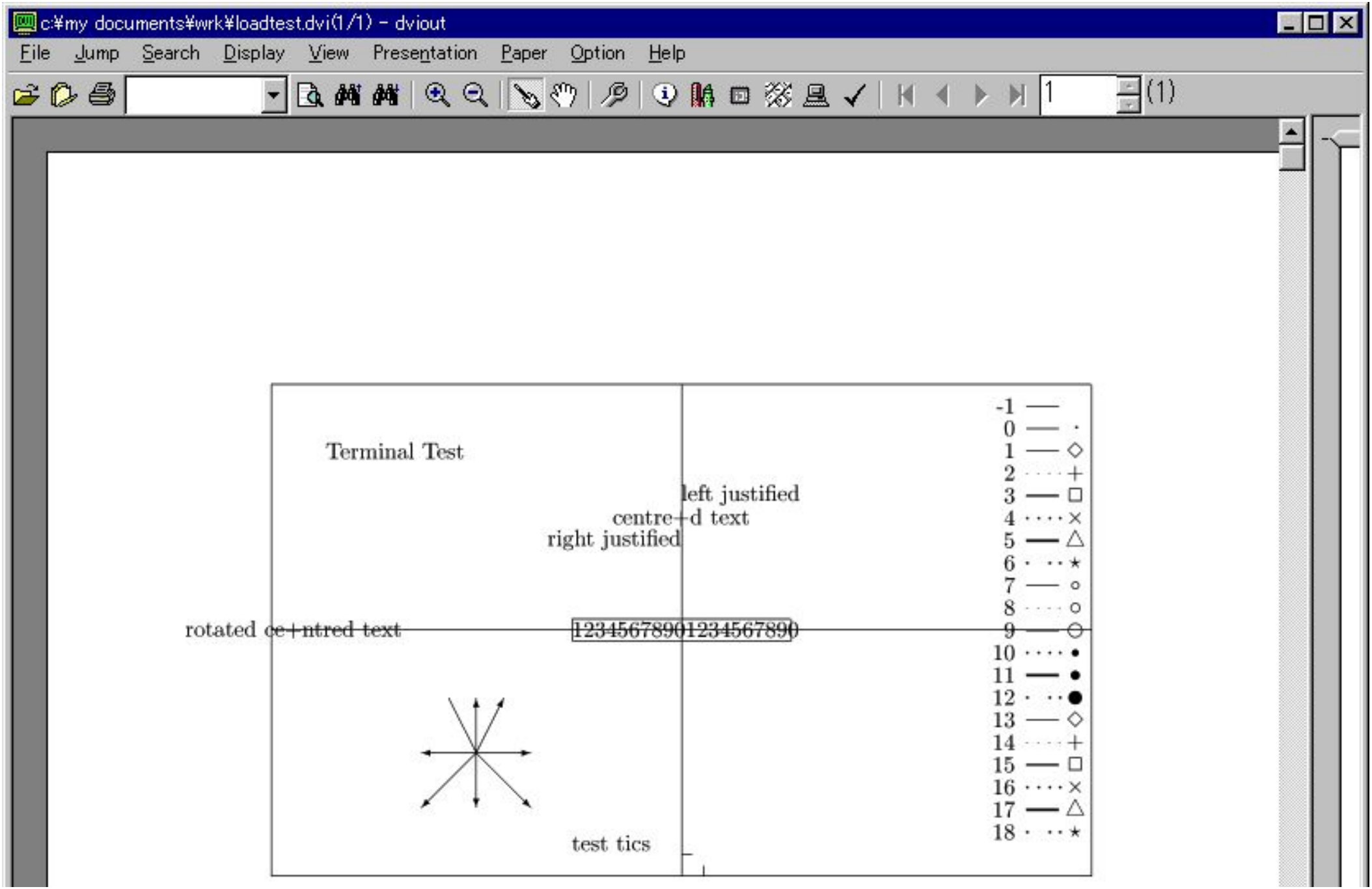
testコマンドの出力(GIFの場合）。

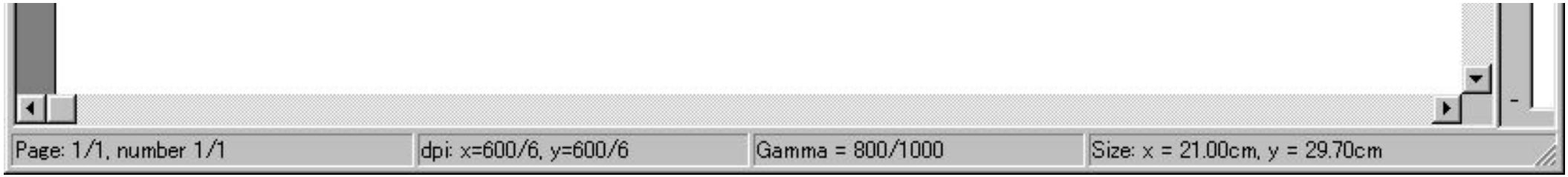


testコマンドの出力(PBMの場合）。



testコマンドの出力(LaTeXの場合）。





LaTeX形式で出力した場合、そのままでは画像を見ることができません。
次のようなtex文書を作って、dviビューアで見てください。

```
/documentclass[a4j]{jreport}
/usepackage{latexsym}
/begin{document}
/input{test}
/end{document}
```

なお、\input命令で読み込んでいるファイルはtest.texというファイルで、gnuplot
が出力した結果が保存されています。
また、この出力を得るためには必ずlatexsym.styが必要になりますので、お持ちで
ない方はCTANより入手してください。