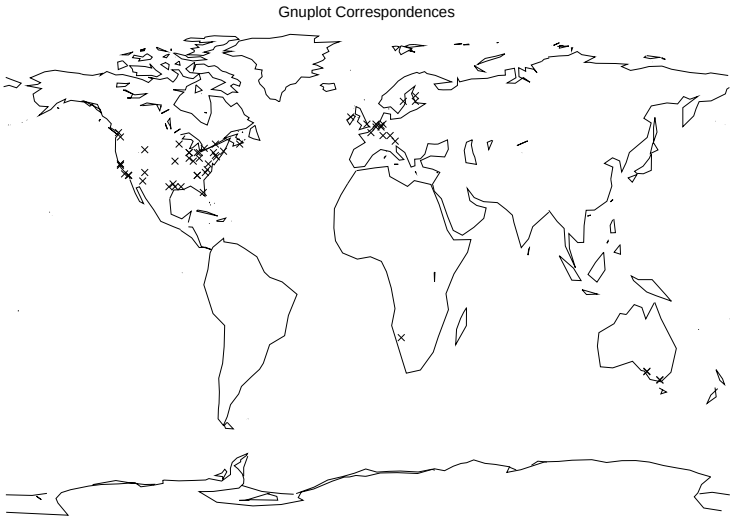
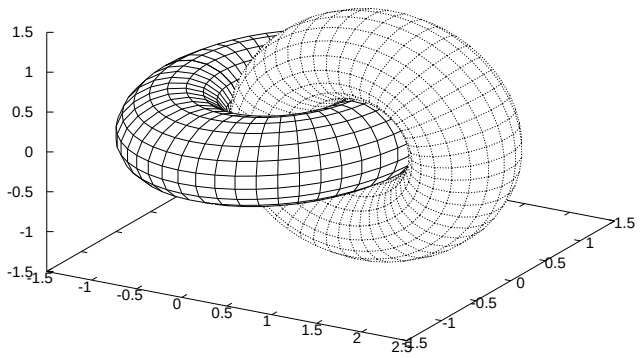
2021年度地球惑星物理学演習テキスト
gnuplot 入門*

改訂:平田 佳織†

2021 年 4 月 14 日





*2020 年度に増田さんが作成したものを改訂

†東京大学理学系研究科地球惑星科学専攻/宇宙科学研究所 太陽系科学研究系、臼井研究室、修士課程 2 年 (email: hirata-kaori444@g.ecc.u-tokyo.ac.jp)

0　はじめに

gnuplot は Unix や Windows、Macintosh などの様々なプラットフォーム上で動く、簡単にデータや関数のプロットができる大変便利なグラフツールです。線や点、等高線、ベクトル場などを、2D および 3D で描くことができます。

使い始めると奥が深く様々なことができるのですが、そのためにはいくつかのコマンドを覚える必要があります。ここでは、その中でも最も基本的なものについて説明をしていきます。ここに書かれている機能が全てではありませんので、マニュアルやインターネット等を参考にしつつ是非 gnuplot を使いこなせるようになって下さい。

なお、gnuplot tips (not so Frequently Asked Questions) というホームページがとても有用な使い方を詳しく教えてくれるので、ぜひ訪問してみてください。

http://lowrank.net/gnuplot/

1　gnuplot の起動と終了の仕方 (gnuplot, quit,...)

まず始めに、gnuplot の起動と終了の仕方について説明します。

gnuplot を起動するには QTerminal 等のウィンドウで **gnuplot** と入力します。すると、画面には次のように表示されます。

```
$ gnuplot


    G N U P L O T
         Version 5.2 patchlevel 8    last modified 2019-12-01

         Copyright (C) 1986-1993, 1998, 2004, 2007-2019
         Thomas Williams, Colin Kelley and many others

         gnuplot home:     http://www.gnuplot.info
         faq, bugs, etc:   type "help FAQ"
         immediate help:   type "help"  (plot window: hit 'h')

Terminal type set to 'qt'
```

gnuplot が起動するとプロンプトが

```
gnuplot>
```

に変わります。これが gnuplot のコマンドラインで、ここにコマンドを対話的に入力しながら作業を進めます。

'gnuplot> ' のプロンプトで使えるコマンドには大きく分けて

- 終了、ファイルの読み込み、保存のためのコマンド (**quit, load, save** 等)
- プロット実行のためのコマンド (**plot, replot, splot** 等)
- プロットでのパラメータを変更するためのコマンド (**set xrange** 等)

- 関数の定義、変数への代入、変数内容の表示、計算のためのコマンド (**f(x)=sin(x)** 等)
- shell に関するコマンド (**pwd, !ls** 等)

等があります。
quit か **q** と入力すると gnuplot は終了します。

2 コマンドが分からなくなったら (help)

これから gnuplot で使うコマンドのいくつかについて説明をしていきます。コマンドの数は決して少なくありません。もしもコマンドの使い方を忘れてしまった場合には

```
gnuplot> help <コマンド名>
```

と入力して下さい。**オンラインマニュアル**が (英語で) 表示されます。**help** は **?** でも代用できます。

マニュアルがさらにいくつかのサブトピックについてわかれている場合には、大まかな説明が表示された後で、どのサブトピックについて調べるのかを聞いてきます。

```
gnuplot> ?  plot
(大まかな説明)
Subtopic of plot:
```

調べたいサブトピックがあればそれを、これ以上細分化されたマニュアルは必要ないのであれば単にリターンを入力します。

なお、**help** の後のコマンド名を省略した場合には gnuplot に関する全体的なマニュアルが表示されます。

3 関数を描いてみる (plot,replot)

すでに形がわかっている関数を作図してみましょう。作図するためのコマンドは **plot** です。例えば、

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

と入力してみましょう。Gnuplot というタイトルのウィンドウが立ち上がりグラフが表示されましたね (図 1)。[1]

関数形がわかっている曲線をグラフにするには、このように

```
gnuplot> plot <x を変数とする関数>
```

と入力します[2]。

いくつかのグラフを一度に作図したい場合には関数をカンマで区切って横に並べます (図 2)。

```
gnuplot> plot sin(x), cos(x)
```

既に作図したグラフに更にグラフを重ねる場合には **replot** を用います (図 3)。

```
gnuplot> replot cos(x*2)
```

単に **plot cos(x*2)** と入力すると先のグラフが消えてしまうので注意して下さい。

[1] この時、画面上の制御がプロットウインドウに移ってしまいます。もちろんマウスでターミナルでクリックしてターミナルに戻ることも出来ますが、「Alt」+「Tab」の画面の切り替えを利用すればマウスに手を移動せずにターミナルに戻ることができます。

[2] 媒介変数による関数指定もできます (**parametric** のオンラインマニュアルを参照)

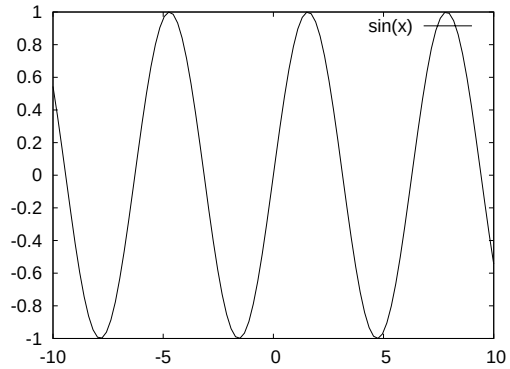


図 1: 正弦関数

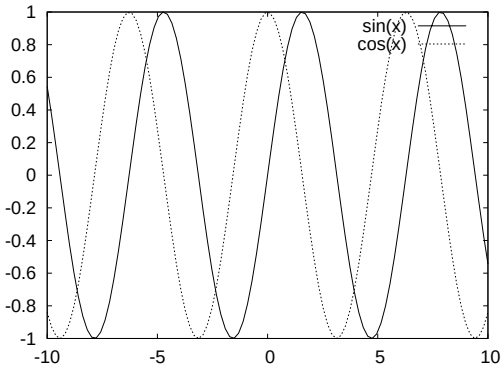


図 2: 正弦関数と余弦関数

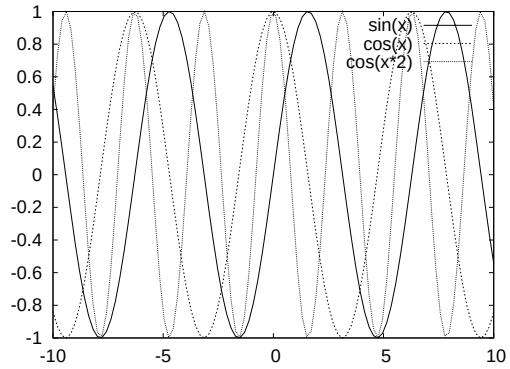


図 3: 図 2 に $\cos(x*2)$ を重ねた

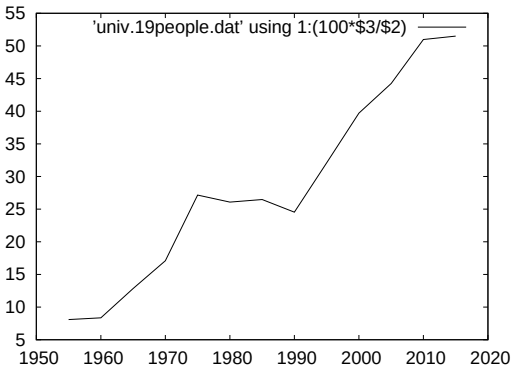


図 4: 大学入学者数の 18 歳人口に対する比率

4　ファイルからデータを読み込む (using, every)

グラフを表示したいのは形がわかっている関数ばかりではありません。ファイルに保存してあるデータを読み込んでそれをグラフにする時も当然あるでしょう。これにもやはり **plot** コマンドを用います。

```
gnuplot> plot ’データファイルの名前’
```

のように、データファイルの名前（絶対パス or 相対パス）を ’ または ” でくくってやります。データファイルは一つ以上のスペースで区切られた $(x, y(, z...))$ の組が複数行から成るものとします。何か適当なデータファイルを作ってそれをグラフにしてみましょう。この時、# を含む行の# 以降はグラフには反映されません。後で見た時にそれが何のデータだったのかわかるような解説を書いておくと良いでしょう。例えばこんな感じです。

```
#   大学入学人数
#      (千人)
#        年          18 歳人口    入学者人数
         1955          1,682          136
         1960          1,998          167
         1965          1.948          250
         1970          1,947          333
         1975          1,561          424
         1980          1,580          412
         1985          1,556          412
         1990          2,005          492
         1995          1,773          569
         2000          1,511          600
         2005          1,365          604
         2010          1,214          619
         2015          1,200          618
```

このデータは dover の /home2/hirata2021/enshu2021/data/univ.19people.dat に書いてあります。ファイルから読み込んだデータをプロットする場合、スタイルは `points` になります。これを変更する場合には

```
gnuplot> set style data lines
```

のように設定しておくか、もしくは

```
gnuplot> plot 'univ.19people.dat' with line
```

のように、毎回スタイルを指定します[3]。

データファイルに書かれているデータが複数列ある時には、**plot** コマンドの際に **using** という引数を付けることによって何列目のデータをグラフにするのかを指定できます。例えば 3 列目のデータを x 軸、1 列目のデータを y 軸にとってグラフを描く場合には

```
gnuplot> plot 'univ.19people.dat' using 3:1
```

のようにします。データが 3 列以上ある場合に **using** を省略すると、1 列目が x 軸、2 列目が y 軸とみなされます。

また、 **using** はそれぞれのデータの各列に対応した値を列番号に **$**をつけて表すことができます。例えば 1 列目のデータは **$1**、2 列目のデータは **$2** で表されます。これを利用することによって、各列のデータを `gnuplot` 上で演算してグラフに表示させることができます。

また、データファイルの一部だけをプロットしたいときには **every** という引数を用います。例えば、1 行おきにプロットする場合には

```
gnuplot> plot 'univ.19people.dat' every 2
```

のようにします。

例として、上の「18 歳人口と大学入学人数」のデータにおいて 2 行目 (18 歳人口) と 3 行目 (入学者数) の演算で「大学入学者数の 18 歳人口に対する比率」という値をグラフ表示させることができます (図 4)。この場合の書式は次のようになります (**p** は **plot** の、**w l** は **with lines** の省略形)。

```
gnuplot> p 'univ.19people.dat' using 1:(100*$3/$2) w l
```

[3] スタイルとして `lines(points)` を設定した場合でも、データファイルが空行 (何も書かれていない行) を含んでいると、空行の直前のデータと直後のデータの間には線は引かれません。

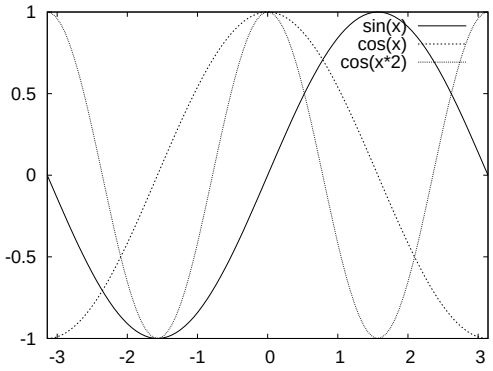


図 5: 描図範囲を $-\pi$ から π までとした

5 細かい図の指定

plot コマンドを用いてグラフを書けるようになりましたが、必ずしも自動で設定されるプロット領域と自分の見たい変域が一致しているとは限りません。また、水平軸・鉛直軸が何なのかを明示していない点で情報がまだ不十分です。これらの指定方法について学習しましょう。

5.1 作図範囲を指定する (set x(y)range, set auto scale)

作図範囲を指定するには **set xrange** (または **yrange**) コマンドを用います。例えば、x の作図範囲を $-\pi$ から π までにしたい時には

```
gnuplot> set xrange [-pi:pi]
```

と入力します。ただし **set** コマンドは設定を変えるだけですから、画面に表示されているグラフは変化しません。画面に表示されているグラフを更新するには

```
gnuplot> replot
```

と入力してもう一度作図し直す必要があります (図 5)。y に関する作図範囲も同様に **set yrange** コマンドによって変えられます。

作図範囲の指定を取り消したい場合には

```
gnuplot> set autoscale
```

と入力します。もちろんこの後にも **replot** と入力する必要があります。以降このテキストでは **replot** の入力は省略しますが、設定を変えた (**set** コマンドを使った) 後は必ず **replot** と入力して図に反映させる必要があります。

5.2 曲線に名前を付ける (title, set key)

plot コマンドによってグラフを描くと、右上にそれがどういう曲線か表示されます。これを変えるには **plot** コマンドの際に **title** という引数を加えます (図 6)。

```
gnuplot> plot sin(x) title 'hogehoge'
```

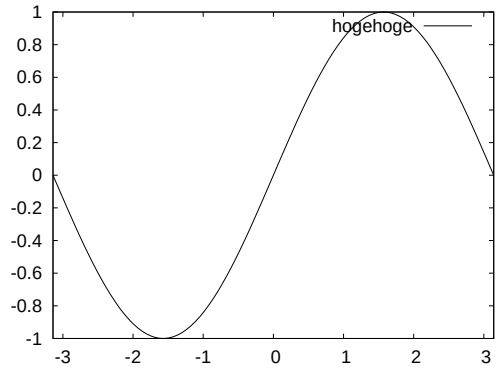


図 6: 曲線に名前をつける

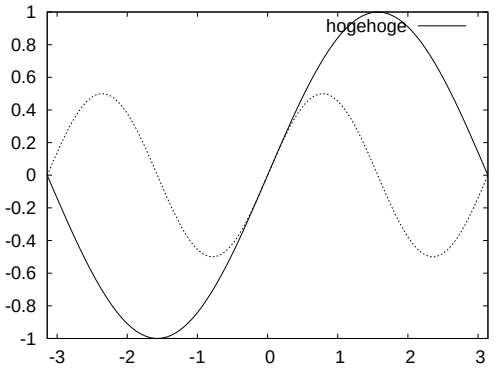


図 7: 図 6 に無名の曲線を重ねた

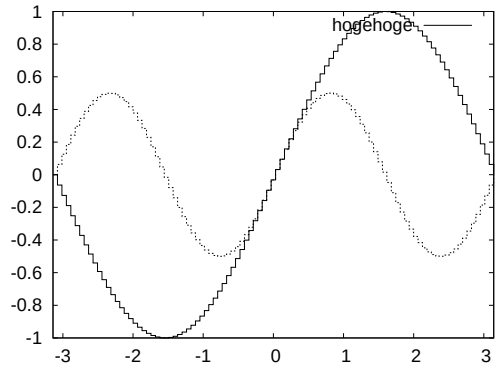


図 8: 図 7 の表示スタイルを **steps** にした

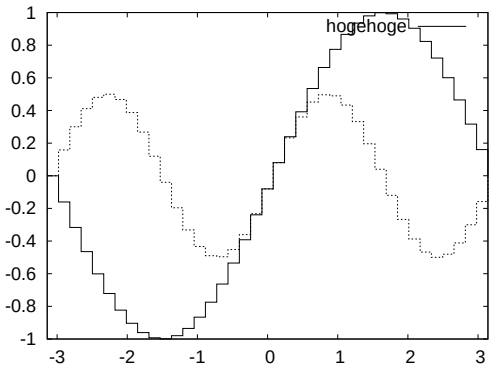


図 9: 図 8 の段の数を 40 段にした場合

曲線の名前の部分を ’ または ” でくくることに注意して下さい。

曲線の名前を表示する必要がない時には

```
gnuplot> plot sin(x) title 'hogehoge', sin(x)*cos(x) notitle
```

のようにします (図 7)。

曲線の名前の表示場所を設定するには **set key** というコマンドを使います。ここでは詳しい説明は省略しますので、オンラインマニュアルでその使い方を確かめてみて下さい。

5.3 描写スタイルを変える (set style function, with, set grid)

今まで表示したグラフは連続した曲線で描かれていました。この表示方法を変えるコマンドとして **set style function** があります。例えば

```
gnuplot> set style function steps
```

としてみましょう。グラフが階段状になりましたね (図 8)。

段の数 (一般には gnuplot が描画のためにサンプルする点数) を変えるには **set samples** というコマンドを用います。

```
gnuplot> set samples 40
```

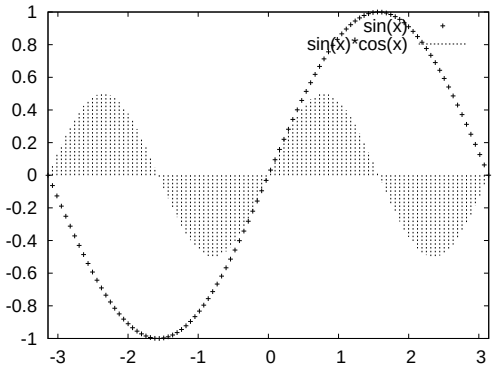


図 10: **points** と **impulses** のグラフの例

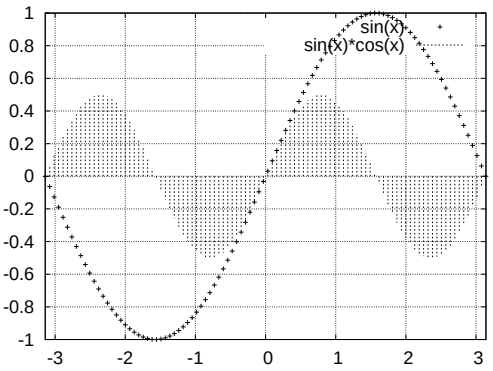


図 11: 図 10 に格子をいれた

とすれば、40 段のグラフが描かれます (図 9)。

グラフの表示はこの他にも、点線 (**dots**) や棒グラフ (**boxes**)、パルス状のグラフ (**impulses**) などが使えます。詳しくは、

```
gnuplot> help set style
```

を見てください。

複数のグラフをそれぞれ異なったスタイルで描きたい場合には、**plot** コマンドの際に **with** をつけて指定します。例えば

```
gnuplot> plot sin(x) with points, sin(x)*cos(x) with impulses
```

のようにします (図 10)。

set grid コマンドを使うと、x 軸、y 軸それぞれの目盛が刻まれている値の格子が入ります (図 11)。格子を消すには **unset grid** とします。

5.4 対数プロット (set logscale)

gnuplot では対数軸の作図も可能です。例えば

```
gnuplot> set logscale y
gnuplot> plot exp(x+1)
```

としてみましょう。y 軸が対数表示になりましたね (図 12)。**logscale** の後を省略した場合には全ての軸が対数軸になります。

対数プロットをやめるには **unset logscale y** とします。

5.5 軸に名前をつける (set x(y)label, title)

大学入学者数のデータを例として、x 軸と y 軸が何を意味しているのか表示させてみましょう。これには **set xlabel** (または **ylabel**) を用います (図 13)。

```
gnuplot> set xlabel 'year'
gnuplot> set ylabel 'new students / 18-year-old (%)'
```

グラフ全体の表題を付けるには **set title** です。

```
gnuplot> set title 'New University Students in Japan'
```

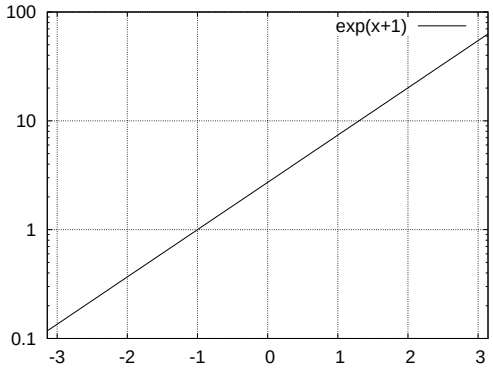


図 12: 片対数グラフの例

5.6 軸の目盛をかえる (set x(y)tics)

データファイルで使っている単位と、描きたいグラフの単位が異なっていることはよくあることです。このような場合には **set xtics** (または **ytics**) コマンドを使って目盛の振り方を変更します。'univ.19people.dat' の 1 列目は西暦表示となっていますが、 これを元号に直すには以下のようにします。

```
gnuplot> set xtics('S25' 1950, ' ' 1955, 'S35' 1960, ' ' 1965, 'S45' 1970,
' ' 1975, 'S55' 1980, ' ' 1985, 'H02' 1990, ' ' 1995, 'H12' 2000, ' ' 2005,
'H22' 2010, ' ' 2015, 'R2' 2020)
```

これで、1950, 1960, 1970, 1980, 1990, 2000, 2010, 2020 の位置にはそれぞれ S25, S35, S45, S55, H02, H12, H22, R2 という目盛が振られ、1955, 1965, 1975, 1985, 1995, 2005, 2015 の位置には何も書かれていない目盛が刻まれます (図 14)。y 軸に目盛を入れる場合も同様です。

また、目盛りの値には元ファイルの値を用いつつ目盛りの間隔を変更したい場合には、**set xtics** ＜刻み幅＞とします。 引数なしに単に **set xtics** とした場合には、目盛は標準 (自動指定) のものに戻ります。目盛が不要の場合には **unset tics** と します。また、**xtics font "フォント名, フォントサイズ"**とすればフォントサイズや種類を変えることができます。デフォルトのフォントサイズは小さめなので、スライド用のグラフを作成するときなどはこれを使用して目盛を読み取りやすくしましょう。普段は表示されていませんが小目盛の設定もでき、**mxtics** ＜分割数＞とすると小目盛の数を変更できます。

6 エラーバーを付ける (errorbars)

観測値や実験値をグラフにする場合の多くは、そこにエラーバーを表示する必要があります。このような場合には **errorbars** というスタイルを指定します。

```
gnuplot> set style data errorbars
```

または

```
gnuplot> plot 'observation.dat' with errorbars
```

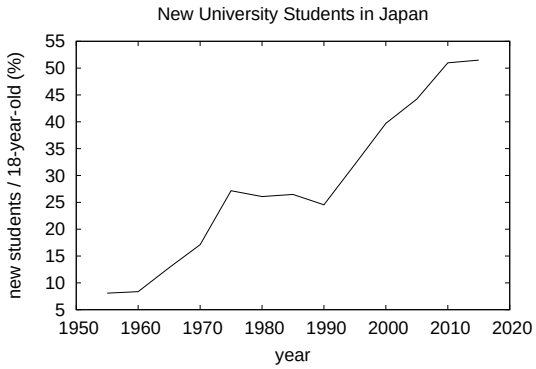


図 13: 各タイトルを入れた例

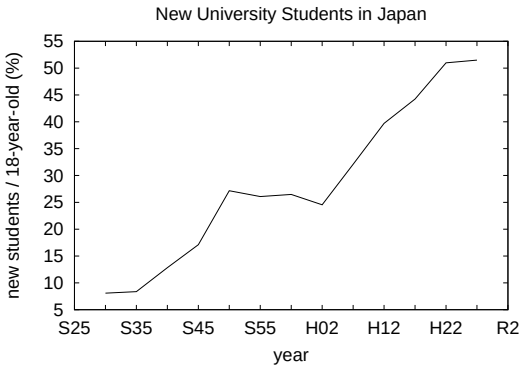


図 14: 図 13 の目盛を和暦にした例

データファイルには、3 列または 4 列のデータが必要です。データが 3 列の場合にはそれぞれの行は $(x, y, \Delta y)$ の組として解釈され、$(x, y\text{-}\Delta y)$ から $(x, y{+}\Delta y)$ までの線が引かれます。データが 4 列の場合には、それぞれの列は $(x, y, y\mathrm{low}, y\mathrm{high})$ として解釈され、$(x, y\mathrm{low})$ から $(x, y\mathrm{high})$ までの線が引かれます。複数列のデータの中から使用する列を指定する場合には、次のように **using** を使います。

```
gnuplot> plot 'observation.dat' using 1:4:3:5 with errorbars
```

7 練習問題 1

先に進む前に、ここまで学習したことの確認をしましょう。

1. dover の /home2/hirata2021/enshu2021/data/univ.19people.dat を scp で手元に持ってくる。
2. 横軸に西暦、縦軸に大学新入生数をプロットする。
3. 横軸に西暦、縦軸に 18 歳人口をとって棒グラフでプロットする。(ヒント:**with boxes**)
4. 横軸に和暦、縦軸に 18 歳人口に対する大学新入生数の割合をプロットする。
5. 前問のグラフに縦軸と横軸にラベルをつけ、タイトルもつける。

8 plt ファイルと load(load '***.plt')

ここまではすべて対話方式でコマンドを入力してきましたが、複数のコマンドをひとつのファイルに書いてまとめて一気に実行することができます。コマンドファイルには、例えば以下の 'univ.19people.plt' のように、一般に plt という拡張子を用いる場合が多いです。Emacs 等のエディタで以下の内容のファイルを書いてみてください。

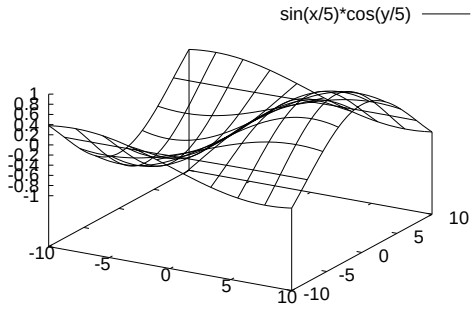


図 15: 3 次元プロットの例

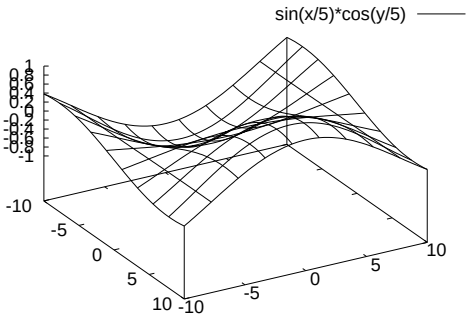


図 16: 図 15 の視点の位置を変更

```
set xlabel 'year'
set ylabel 'new students / 18-year-old (%)'
set title 'New University Students in Japan'
set xtics('S25' 1950,' ' 1955, 'S35' 1960,' ' 1965,'S45' 1970,' '
1975,'S55' 1980,' ' 1985, 'H02' 1990,' ' 1995,'H12' 2000,' ' 2005,
'H22' 2010, ' ' 2015, 'R2' 2020)
set xrange [1950:2020]
set yrange [0:60]
set ytics 10
p 'univ.19people.dat' using 1:(100*$3/$2) w l
```

これらのコマンド群を一気に実行するには **load** というコマンドを使います。

```
gnuplot> load 'univ.19people.plt'
```

としてみると練習問題 1 の図が作れます。

9　3 次元プロット

gnuplot では 3 次元のグラフも作成することができます。

9.1　3 次元プロット (splot, set view)

3 次元プロットする際には **splot** というコマンドを使います。

```
gnuplot> splot sin(x/5)*cos(y/5)
```

デフォルトでは視点の位置は x 軸から 60°、 z 軸から 30° の地点になります (図 15)。これを変更するには **set view** コマンドを用います (図 16)。

```
gnuplot> set view 50,60
```

とすれば、画面水平右向きが x 軸、鉛直上向きが y 軸、画面手前向きが z 軸であった座標系を、x 軸の周りに 50° z 軸の周りに 60° それぞれ回転させたグラフの画面への投射像が得られます。また、最近の gnuplot ではマウスで図を回転させることも可能です。

データファイルから値を読み込んで 3 次元プロットする際には **plot** コマンド同様

```
gnuplot> splot 'data3d.dat' with line
```

のようにします[4]。この時、データファイルには 3 列以上のデータが書かれていることが必要です[5]。3 次元データは (X, Y, Z) の組にして与えます。出力形式は以下のようにします。(**注意!!空行が一行必要です。**)

```
#   x   y   z
    0   0   0
    0   1   1
    0   2   4
    0   3   9

    1   0   1
    1   1   2
    1   2   5
    1   3  10

    2   0   2
    2   1   3
    2   2   6
    2   3  11

    3   0   3
    3   1   4
    3   2   7
    3   3  12
```

ファイルに空行で区切られた同数のデータが複数書かれている場合、スタイルとして **lines** (**points**) を指定するとデータを格子上に結んだグラフが描かれます。
(上記データは dover の /home2/hirata2021/enshu2021/data/data3d.dat にあるのでデータ形式を確認した上で **splot** してみてください。)

9.2 コンタープロット (set contour(surface), set contrparam)

コンタープロットとは等高線地図を描くことです。等高線地図を描くには 3 次元データをプロットする時に

```
gnuplot> set contour
```

と設定します。すると、x-y 平面上に等高線地図が描かれます (図 17)。

等高線を x-y 平面ではなくグラフの曲面上に重ねるには

```
gnuplot> set contour surface
```

と設定します。

等高線を描く際の様々なパラメタ (等高線を描く際の近似の仕方、等高線の開始値、幅、終了値等) を変更するには **cntrparam** の値を設定します。例えば

```
gnuplot> set cntrparam levels incremental 0,0.5
```

とすると、0 から開始して 0.5 刻みで等高線が描かれます。詳しくは **cntrparam** のオンラインマニュアルを見て下さい。

等高線地図は多くの場合 x-y 平面に投影した形で表されます。そのためには次のような設定をすると良いでしょう。

[4] バージョンによっては **splot** の前に **set parametric** を設定しなければならないかも知れません
[5] 1 列だけの場合にも 3 次元プロットができます。**parametric** オンラインマニュアルを参照のこと

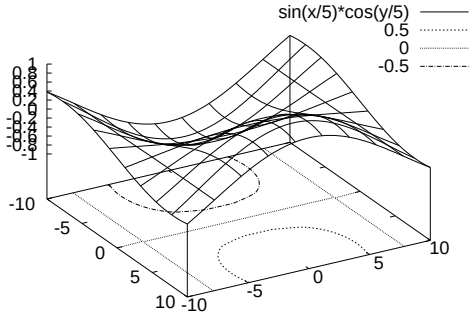


図 17: 図 16 に等高線を描いたもの

```
gnuplot> unset surface
gnuplot> set view 0,0
gnuplot> unset ztics
```

それぞれのコマンドの意味はオンラインマニュアルで確かめてみて下さい。

ファイルからデータを読みとって等高線地図を描く場合には、データファイルに欠損値があってはならないことに気をつけて下さい。

9.3 カラーのグラデーション表示にする (set pm3d)

三次元の図をカラーのグラデーションにする方法として **pm3d** というものがあります。

```
gnuplot> set pm3d
```

とすると、このあとの三次元プロットはグラデーションを用いたものとなります。ためしに

```
gnuplot> splot sin(x/5)*cos(y/5)
```

としてみてください。さきほどの図 15 の時と違ってグラデーションになったでしょうか。
ちなみに、このモードをやめるときは

```
gnuplot> unset pm3d
```

とすればもとのモードになります。

同様に $x-y$ 平面に投影された等高線をグラデーションにするには

```
gnuplot> set pm3d at b
```

としてください。図 17 の場合と比較するために、

```
gnuplot> unset surface
gnuplot> set view 0,0
gnuplot> unset ztics
gnuplot> splot sin(x/5)*cos(y/5)
```

としてみましょう。

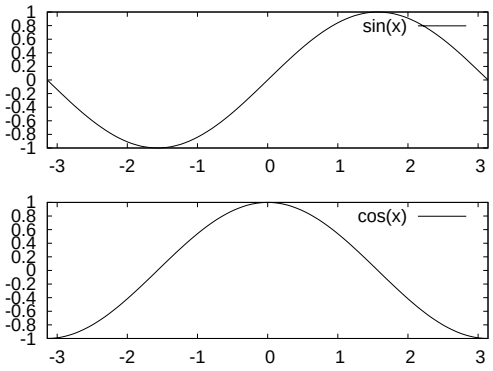


図 18: 2 つのグラフを一度に描写

10 複数のグラフを描写する (set multiplot)

一度に複数のグラフを描写する場合には **set multiplot** コマンドを使います。このコマンドにより gnuplot は **multiplot** のモードに入ります。例えば

```
gnuplot> set multiplot
multiplot> set size 1,0.5
multiplot> set origin 0,0.5
multiplot> plot sin(x)
multiplot> set origin 0,0
multiplot> plot cos(x)
```

とすると、上に $\sin x$ 下に $\cos x$ のグラフが描かれます (図 18)。**set origin** コマンドはグラフの左下の位置を決め、**set size** コマンドはグラフの縦横のサイズを変更します。

`multiplot` モードを終了させるには

```
multiplot> unset multiplot
```

と入力して下さい。

11 グラフを出力する (set terminal postscript, set output)

グラフを印刷するためには、画面の情報をポストスクリプト形式でファイルに保存する必要があります。これには次のようにします。

```
gnuplot> set terminal postscript
gnuplot> set output 'output.ps'
gnuplot> replot
```

出力先を決めた後に **replot** をすることを忘れないでください。terminal をファイル出力に変えてからグラフを書き出さなければ何も書かれません。**replot** とすると直前のプロットと同じものを書き出すことができます。

出力ファイルを閉じるには、

```
gnuplot> set output
```

とします。(**注意！！ set output をしないで他へ出力に切り替える (set terminal qt 等) とファイルが壊れるので、ファイルを閉じるか、gnuplot を一度終了する (quit) のを忘れないようにしてください。**)

また、このままの状態だとプロット出力が postscript に書かれるので、出力を再び画面に表示するために

```
gnuplot> set terminal qt
```

とします。

出来上がった図は **gv** というコマンド (PostScript and PDF view) で見ることができます。

```
gv output.ps
```

また、デフォルトではグラフは横描き (*landscape*) に出力されるのですが、これを縦描き (*portrait*) にしたい場合には

```
gnuplot> set terminal postscript portrait
```

のように設定します。また

```
gnuplot> set size 0.5,0.25
```

と設定すると、グラフが横に半分、縦に 1/4 の大きさになります。図の縦横比を変更したいときは

```
gnuplot> set size ratio 2.0
```

とすると、縦:横=2:1 の図を作ることができます。

set terminal は postscript 形式に限らず様々な形式で出力できます。例えば、描画ソフトである Tgif のファイル形式で出力するときには、以下のようにします。

```
gnuplot> set terminal tgif
gnuplot> set output 'output.obj'
gnuplot> replot
```

要は terminal の種類を tgif と指定するだけです。

同様にして、作成した図を Latex に貼りつける際には eps (Encapsulated Postscript) 形式で出力する必要があります。その場合には

```
gnuplot> set terminal postscript eps
```

のようにします。カラーで出力したい場合には

```
gnuplot> set terminal postscript eps color
```

などとしてください。eps や color などのオプションは、

```
gnuplot> help set terminal postscript
```

とすることで調べることができます。

tgif や eps ファイル以外にも、png、jpg、ps、pdf 等、様々なファイル形式に書き出すことができます。詳しくは、

```
gnuplot> help set terminal
```

で調べてみて下さい。また、これらの画像ファイルを表示するには **display**(ImageMagick) というコマンドが便利です。

12 load と save

先ほど **load** というコマンドを学びましたが、この **load** と兄弟関係にあるコマンドが **save** です。**save** コマンドを用いることで、これまでに対話形式で実行してきたコマンドを先程自分で書いた plt ファイルと同様な出力として得られます。
例えば、

```
gnuplot> p cos(x)
gnuplot> set xlabel 'x'
gnuplot> set ylabel 'y'
gnuplot> set terminal postscript
gnuplot> set output 'cos.ps'
gnuplot> rep
gnuplot> save 'cos.plt'
```

としてみましょう (**rep** は **replot** の省略形)。
これを emacs などで開いてみると、
── ここから ── ここから ── ここから ── ここから ──

```
#!/usr/bin/gnuplot -persist
#
#
#     G N U P L O T
#     Version 4.6 patchlevel 6    last modified September 2014
#     Build System: Linux i686
#
#     Copyright (C) 1986-1993, 1998, 2004, 2007-2014
#     Thomas Williams, Colin Kelley and many others
#
#     gnuplot home:     http://www.gnuplot.info
#     faq, bugs, etc:   type "help FAQ"
#     immediate help:   type "help"  (plot window: hit 'h')
# set terminal postscript landscape noenhanced defaultplex \
   leveldefault monochrome colortext \
   dashed dashlength 1.0 linewidth 1.0 butt noclip \
   nobackground \
   palfuncparam 2000,0.003 \
   "Helvetica" 14  fontscale 1.0
# set output 'cos.ps'
           ⋮
set loadpath
set fontpath
set psdir
set fit noerrorvariables noprescale
GNUTERM = "x11"
p cos(x)
#    EOF
----- ここまで ----- ここまで ----- ここまで ----- ここまで -----
```

のように書かれていると思います。
一瞬意味不明に見えるかも知れませんが、これは実際に gnuplot> において入力することができるコマンドを並べただけなのです。 自分で打ち込んでない部分 (例えばここでは、**set loadpath** など) は、gnuplot が勝手に解釈してデフォルトの環境を出力してくれます。このファイルを良く見ると、自分で入力した **set title** や **set xlabel** が書き出されていることが分かると思います。**p cos(x)** は設定された環境が全て反映されるように最後に書かれるようになっ ています。
このファイルを先ほど学んだ **load** コマンドで読み込めば同じ状況が再現されます。ただし、**set terminal postscript** や **set output 'cos.ps'** のように、terminal のタイプを変更したり何かを出力したりするコマンドの部分は、**save** で出力したコマンドファイルではコメントアウトされます。皆さんのファイルでもきっとそうなっていることと思います (コマンドファイルのコメントアウトの最後の部分)。もし load することで postscript ファイルを作りたければ、この部分のコメントアウトを消したうえで、先ほど学んだ **load** コマンドを用いるかあるいは kterm など普通のターミナル上で

```
$ gnuplot cos.plt
```

と入力することで postscript ファイルを作ってやることができます。
plt ファイルはこれから研究をする上でとても強力な手段となります。例えば $\sin x, \sin 2x, \sin 3x$ のグラフを出力したいとします。この時、いちいち gnuplot を立ち上げて何度も **plot sin(x)** などと打つのは面倒ですよね (まあ、3 つぐらいなら面倒ではないかも知れませんが、$\sin 100x$ までだとしたらどうでしょう)。そんな時は、前もって emacs などで

```
----- ここから ----- ここから ----- ここから ----- ここから -----
#! /usr/bin/gnuplot
set xrange [0:2*pi]
set yrange [-1:1]
set terminal postscript
set output "sin1x.ps"
p sin(x)
set output "sin2x.ps"
p sin(2*x)
set output "sin3x.ps"
p sin(3*x)
----- ここまで ----- ここまで ----- ここまで ----- ここまで -----
```

と書いておきます。コピー&ペースト等を使えば簡単ですね。たとえば sintest.plt という名前で保存したとします。その上で、

```
$ gnuplot sintest.plt
```

とすると…、一発で 3 つの postscript ファイルができてしまいます!!
後から学ぶと思いますが、unix のコマンドを用いれば、$\sin x$ から $\sin 100x$ まで出力するようなコマンドファイルだって簡単に作れてしまいます。unix(シェル) とコマンドファイルのコンビネーションによって、gnuplot は非常に強力なアイテムになります。
「そんなことしないよ。」と思っている人がいるかもしれません。しかし、研究をしていく上でこういったことは意外と良くあります。例えば、「数値計算を行って 10 秒ごとにデータを取り、それを全てグラフにして時間発展を見たい!」と思った時にこの技が大変役に立ちます。

13　練習問題 2

1. $f(x,y) = x^2 \exp\left(-x^2\right) y^2 \exp\left(-y^2\right)$ を $-3 < x < 3$, $-3 < y < 3$ の範囲で 3 次元プロット (**splot**) する。
2. メッシュの数を増やす。(ヒント：**set isosample**)
3. 等高線を底面に描く。
4. pm3d を用いてカラーのグラデーションを用いてプロットしてみる。
5. load すると一発で $f(x,y)$ を $-3 < x < 3$, $-3 < y < 3$ の範囲で底面だけを 2 次元表示する plt ファイルをゼロから自分で書く。(ヒント：**set pm3d map**)
6. eps 形式かつカラー表示で書きだし、gv で図ができていることを確認する。
   →課題 2 へ。

Appendix 1: アニメーション

先ほどのコマンドファイルをうまく使うと、アニメーションを作ってやることができます。
たとえば…、

```
----- ここから ----- ここから ----- ここから ----- ここから -----
#! /usr/bin/gnuplot
set xrange [-20:20]
set yrange [0:2]
i=0
p exp(-(x+i)**2)
pause -1
i=i+0.5
rep
pause -1
i=i+0.5
rep

 ⋮

rep
----- ここまで ----- ここまで ----- ここまで ----- ここまで -----
```

リターンを押すごとに画像が動いて見えますよね。**pause** コマンドは **pause 秒数**でその秒数だけ停止する、というコマンドです。秒数を-1 にしたときにはリターンが押されるまでその状態で停止します。この技は、微分方程式の時間発展の結果のデータファイルを順番に読み込ませて時間発展の雰囲気を掴んだりする時にかなり有用です。ぜひ試してみてください。

Appendix 2: 関数の利用 (f(x))

数式中の変数の値を少し変えただけの複数のグラフを描画したいとき、複雑な数式だと入力する作業が面倒なだけでなく、入力ミスから全然違うグラフを描画してしまうということがおきやすくなります。そんな時に便利なのが関数定義という機能です。これは数学関数の $\sin, \cos$ などのように、gnuplot 中で使える一般的な関数 $f(x)$ を自分で定義することができるというものです。
例えば次のような関数を表示したいとき

$$y = a\sin(x)\cos(x) \quad (a = 1, 2) \tag{1}$$

```
gnuplot> f(a,x) = a*sin(x)*cos(x)
gnuplot> plot f(1,x)
gnuplot> replot f(2,x)
```

とすれば、いちいちにたような数式を入力することなく、2 つの場合についてグラフを描くことができます。

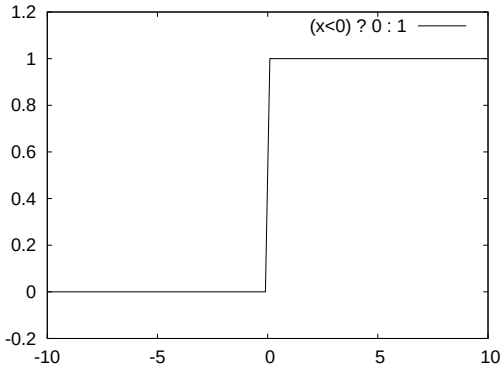


図 19: 三項演算子の利用

Appendix 3: 三項演算子の利用

例えば Heaviside 関数のように x の値によって関数の定義がかわってしまうようなグラフを描画するときに三項演算子というものを利用すると、簡単に描くことができます。
三項演算子は次のような書き方をします。

条件文 ? 条件が真の時の処理 : 条件が偽の時の処理

具体例として、Heaviside 関数を見てみましょう。Heaviside 関数の定義は次のようになっています。

$$f(x) = \begin{cases} 1 & x > 0 \\ 0 & x < 0 \end{cases} \tag{2}$$

これを描くためには次のように入力します (図 19)。

```
gnuplot> set yrange [-0.2:1.2]
gnuplot> plot (x < 0) ? 0 : 1
```

Appendix 4: 媒介変数の利用 (set parametric)

これまで紹介したものはすべて $y = f(x), z = f(x, y)$ と表せるものでした。しかしこれだけでは表現できないものもあります。そんなとき、gnuplot では媒介変数を利用することもできます。

```
gnuplot> set parametric
```

と入力することで媒介変数モードとなります。このとき媒介変数として認識される変数は、二次元プロットなら「t」、三次元プロットなら「u,v」となります。次に円・球を描く例をのせます。

円を描く (図 20)

```
gnuplot> set parametric
gnuplot> plot cos(t),sin(t)
```

球を描く (図 21)

```
gnuplot> set parametric
gnuplot> splot sin(u)*cos(v),sin(u)*sin(v),cos(u)
```

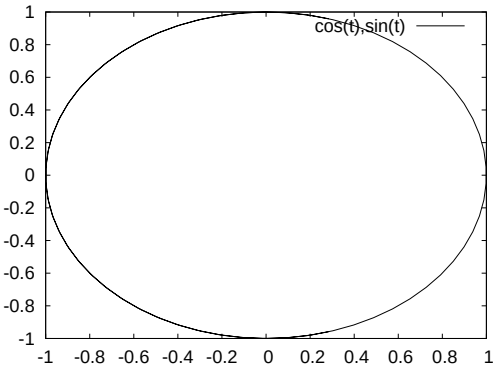


図 20: 媒介変数の使用 (円)

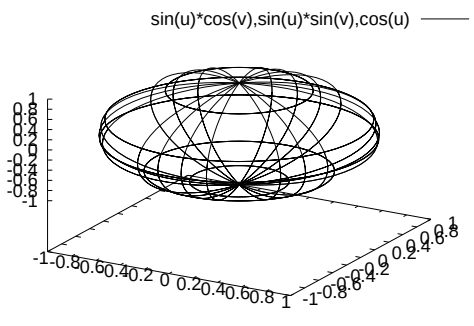


図 21: 媒介変数の使用 (球)

Appendix 5: gnuplot 上での shell コマンドの利用 (shell, !)

gnuplot から一時的に shell に抜けることができます。そのためには、

```
gnuplot> shell
```

と打ちます。そうするとプロンプトが gnuplot 起動前のものに戻るはずです。gnuplot に戻りたいときには **exit** で shell を終了して gnuplot のプロンプトに戻ってきます。

1 行だけの shell コマンドなら **! command** を使って gnuplot 内から実行することもできます。例えば

```
gnuplot> !ls
```

などとやってみてください。

また、**pwd** と **cd** は gnuplot のコマンドラインからも実行できます (**cd** は **!**をつけると使えません)。以下の例のように、ディレクトリ名は必ず引用符で括る必要があることに注意してください。

```
gnuplot> cd '../'
```

Appendix 6: 最小二乗フィッティング

gnuplot の強力な機能の一つとして関数のフィッティング (当てはめ) があります。パラメータを含んだ関数を用意すると、データファイルの数値に対して適当なパラメータの数値を自動的に検索してくれます。関数は線形、非線形を問いません。

例として、あるデータファイル 'linear.dat' を直線にフィッティングするときを考えます。まず、パラメータを含む関数 $f(x)$ を定義します。関数フィッティングには **fit** コマンドを使います。オプションとして検索したいパラメータ名を **via** に続けて与えます。

```
gnuplot> f(x) = a*x + b
gnuplot> fit f(x) 'linear.dat' via a, b
gnuplot> plot 'linear.dat'
gnuplot> replot f(x)
```

収束しない場合は、大雑把に検討をつけた初期値を手で与えてやれば上手くいく場合が多いです。例えば、関数 $f(x)$ を定義したあとに $a = -3$ などとして初期値を与えてから **fit** します。

Appendix 7: 画像の処理について (gv, Image Magick, GIMP)

画像を処理する方法はいくつかありますが，ここでは簡単に 4 つほど紹介します.

gv　**gv** はターミナル上で,

```
$ gv ***.ps
```

のように打つと画像がババっと表示されるコマンドです。簡単に図を確認する事ができるので今後よく使用すると思います。
（というか既に使っていますね）

Image Magick　Image Magick は ps, eps, jpg, png, gif など多くの形式に対応した画像処理ソフトです。主に **display** というコマンドと **convert** というコマンドを使います。
display はターミナル上で,

```
$ display ***.ps
```

のように打つと gv と同様に画像が表示されます。
convert は画像処理のためのコマンドです。ターミナル上で,

```
$ convert ***.ps ***.pdf
```

のように打つと ps ファイルが pdf ファイルに変換されます。
他にも

```
$ convert -resize 80 file1 file2
```

と打つと file1 を 80 %に縮小した画像が file2 に書き込まれたり、

```
$ convert -delay 50 hoge1.gif hoge2.gif hoge3.gif hogehoge.gif
```

と打つと hoge1∼3 が 0.5 秒ずつ流れる gif アニメーションを作ることができたりします。この convert はかなり有用だと思いますので調べて使ってみてください。

gimp　**gimp** はターミナル上で,

```
$ gimp ***.ps
```

のように打つと同様に画像が表示され，色々と手を加える事もできるコマンドです。

14 課題

1. 媒介変数表示
   半径 1 の円と、$x = 0$、$y = 0$ の直線を $-1 < x < 1$, $-1 < y < 1$ の範囲で描いてください。グラフの縦横比は 1:1 にしてください (Hint: **set size square**)。

   # 提出：スクリプトファイル（kadai1.plt）とグラフ（kadai1.eps）

2. **3 次元プロット**
   # 提出：練習問題 2-6 で作成したスクリプトファイル（kadai2.plt）とグラフ（kadai2.eps）

3. **2 軸プロット**
   dover の /home2/masuda2021/enshu2021/data/kadai3.dat に、本国におけるバナナの輸入量についてのデータがあります。横軸に年を取り、左の y 軸（第一軸）を用いて総輸入量の棒グラフをプロットし、さらに右の y 軸（第二軸）を用いて各国からの輸入量の総輸入量に対する割合（%）をプロットしてください。

   Hint

   ```
   gnuplot> set y2tics
   gnuplot> p ' データ 1 '
   gnuplot> rep ' データ 2' axes x1y2
   ```

   # 提出：スクリプトファイル（kadai3.plt）とグラフ（kadai3.eps）

4. 最小二乗フィッティング
   大気中に 2 つの等圧面があり、圧力差 Δp, 高度差 Δz であるとすると

   $$\Delta p = \rho g \Delta z$$

   という関係が成り立ちます (ρ:密度,g:重力加速度)。状態方程式より

   $$\rho = \frac{p}{RT}$$

   なので (R:空気の気体定数,T:絶対温度)、温度がほぼ一定という近似のもとで積分すると

   $$p = p_0 \exp\left(-\frac{z}{H}\right)$$
   $$\Leftrightarrow \log p = \log p_0 - \frac{z}{H}$$

   となります。ここで $H = RT/g$ をスケールハイトといい、大気の厚さの目安として知られています。

   dover の /home2/hirata2021/enshu2021/data/kadai4.dat にはラジオゾンデ観測によって得られた気圧の鉛直プロファイルが書かれています（1 行目：高度 (m)、2 行目：気圧 (hPa)、3 行目：気圧の自然対数)。最小二乗法により傾きを求め、そこからスケールハイトを求めてみましょう。

Hint

```
gnuplot> f(x) = a*x + b
gnuplot> plot 'kadai4.dat' using 1:3
gnuplot> fit f(x) 'kadai4.dat' using 1:3 via a, b
```

提出：求まった**スケールハイト**（kadai4.txt）+ 横軸を高度、縦軸を気圧の自然対数として、観測値とフィッティングした結果の直線の両方をプロットしたグラフ（kadai4.eps）

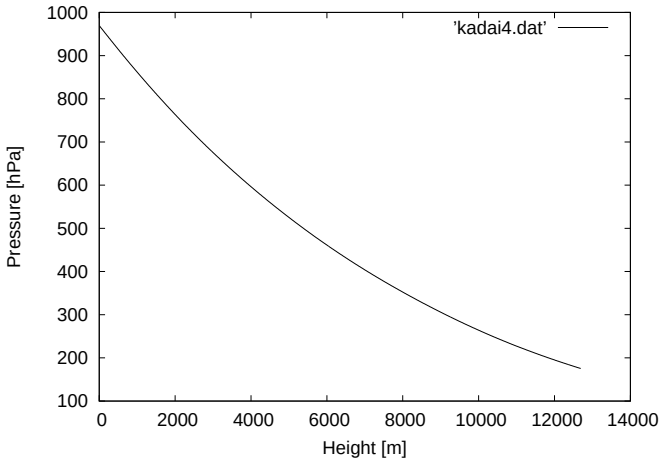


図 22: 高度と気圧

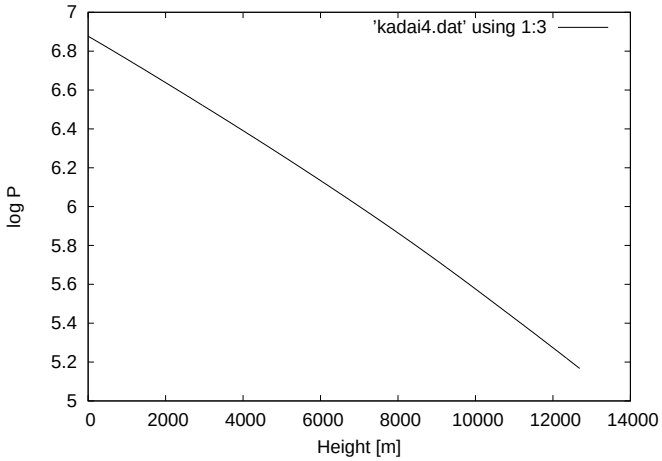


図 23: 高度と気圧の自然対数

dover の /home2/hirata2021/enshu2021/teishutsu に s2126?? （??は自分の学生証番号の下 2 桁）というディレクトリを作成し、そのなかに全課題で指定されたファイル 8 つを置いてください。すべてのグラフには単位を明記した軸ラベル、タイトルをつけ、見やすいグラフにしてください。スクリプトファイルは load すると一発でグラフが表示されるものとし、読みやすいように記述してください。また、提出したファイルは TA は読めるが他の学生は読めないように適切にパーミッションを設定してください。

提出期限は **4 月 21 日 (水)12 時 59 分**とします。