第 05 回：可視化

陰山 聡

計算科学演習 I（2015 年前期）

2015.05.14

準備

サンプルプログラム

`cd` （ホームディレクトリに移動）

`mkdir vis01` （ディレクトリ作成。名前は何でも OK）

`cd vis01` （そのディレクトリに移動）

`cp /tmp/150514/leibniz.* .` （サンプルコード（二つ）をコピー）

可視化とは

可視化の一般論

- 情報可視化 Information visualization
- データ可視化 Data visualization, Scientific visualization

1 次元可視化

x の関数 $f(x)$ を直感的に理解するためには、グラフを書くのが一番である。例えば、

$$y = f(x)$$

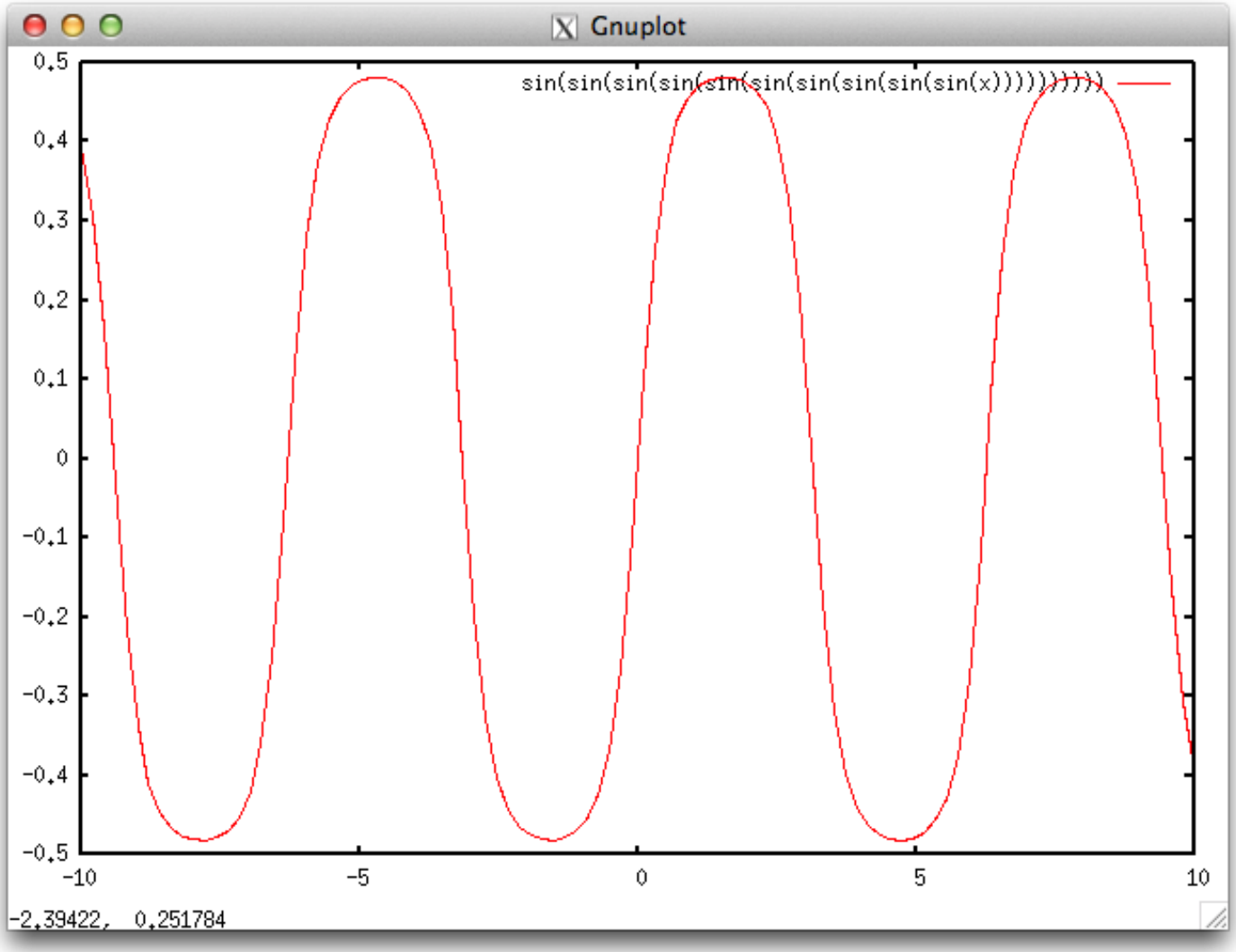
に対して、f の値域が定義域に含まれるとき、

$$f^{\times 2} := f((f(x)), \quad f^{\times 3} := f(f(f(x))), \quad \cdots$$

等と定義する。

$f(x) = \sin x$ に対する $f^{\times 10}(x)$ はどのような関数であろうか？

$$y = \sin(\sin(\sin(\sin(\sin(\sin(\sin(\sin(\sin(\sin(x)))))))))))$$



クイズ

x を 0 以上の実数として、x の x 乗、つまり

$$f(x) = x^x \quad (x \geq 0)$$

はどんな関数であろうか？

- 最大値／最小値をとる x は？
- $x = 0$ の時の値 $f(0) = 0^0$ は何だろう？

【後で演習】

2次元データの可視化：等高線

x と y の関数、つまり 2 次元の関数 $f(x, y)$ の形を理解するには、等高線を描くのがよい。地表面での大気の圧力 p の分布 $p(x, y)$ の等高線は天気図でおなじみである。

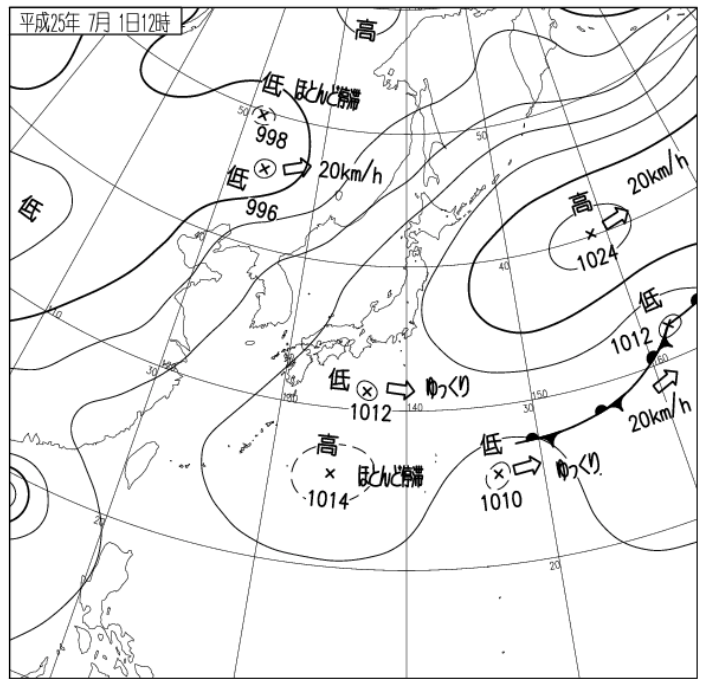
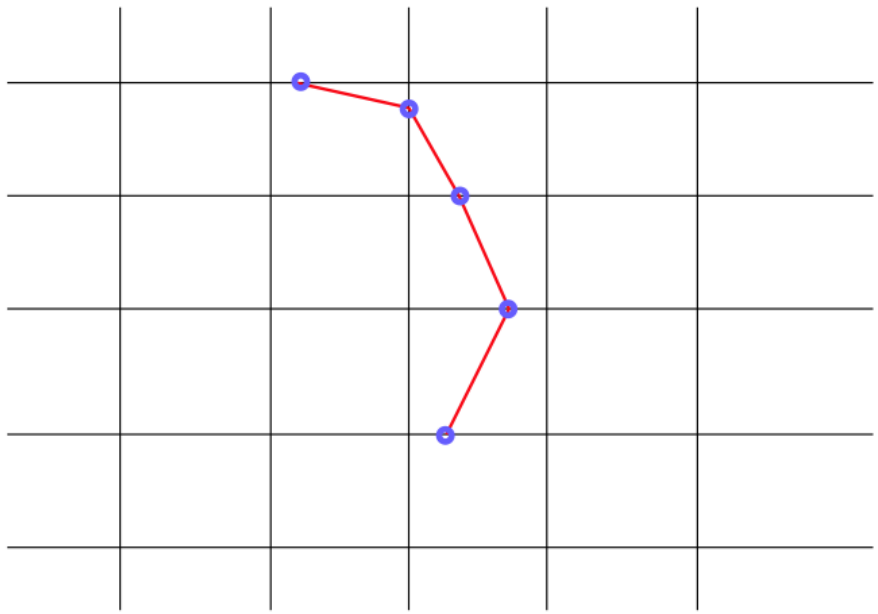


Figure: 気象庁のウェブページ: http://www.jma.go.jp/jp/g3/より

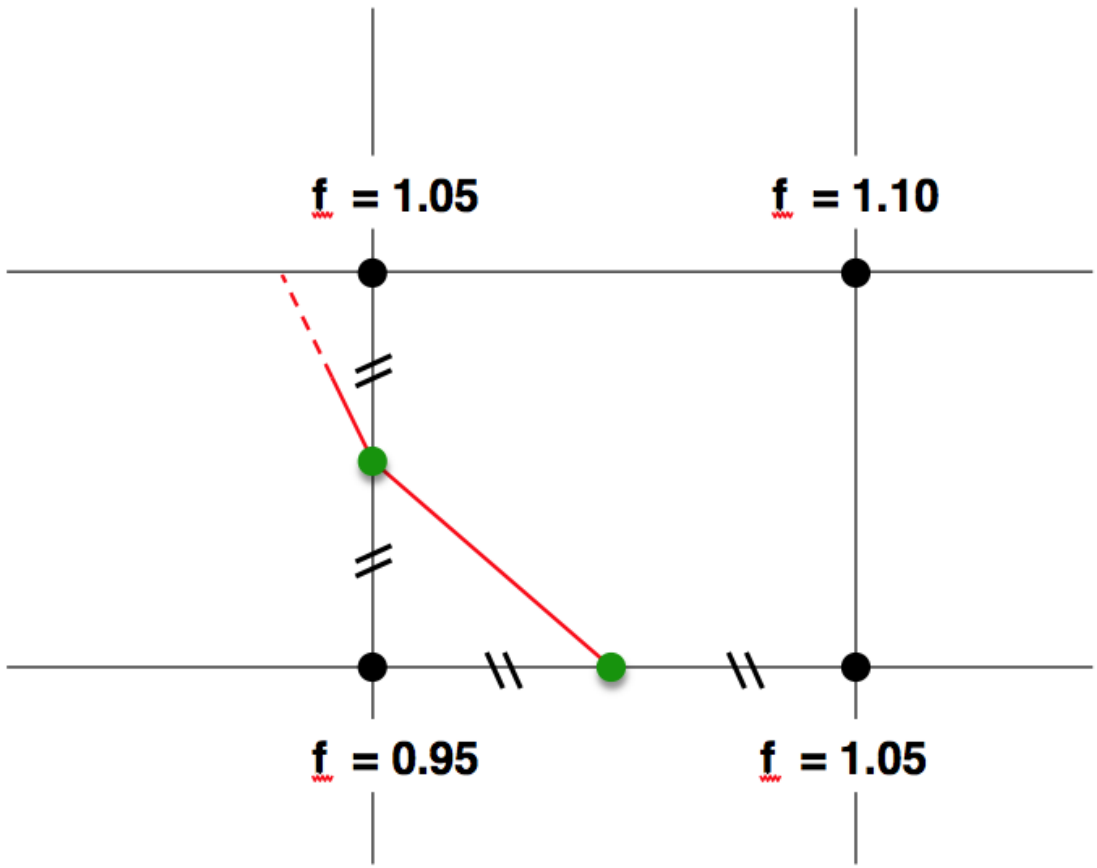
等高線の描画アルゴリズム

ここで等高線の描画アルゴリズムを紹介しよう。計算格子点上に定義されたデータから一本の等高線を描くには、下の図のように短い線分をつなげていけばよい。

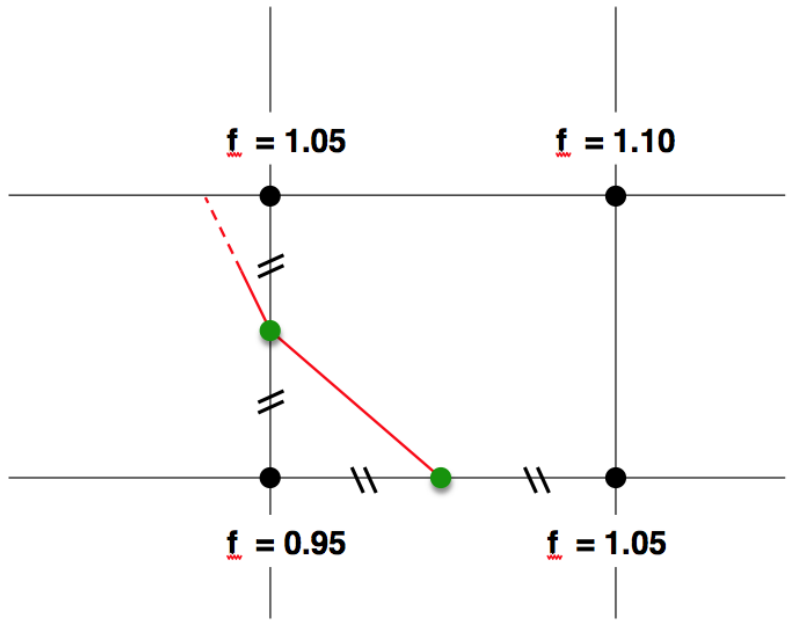


一つの線分は 4 つの計算格子点で定義された長方形領域（セル）の中で直線を描く。例えば f(x,y)=1.0 の値の等高線を描く場合を考えよう。

あるセルの 4 つの頂点における f の値が全て 1.0 よりも大きいか、あるいは 1.0 未満であれば、f=1.0 の等高線はこのセルを通らない。



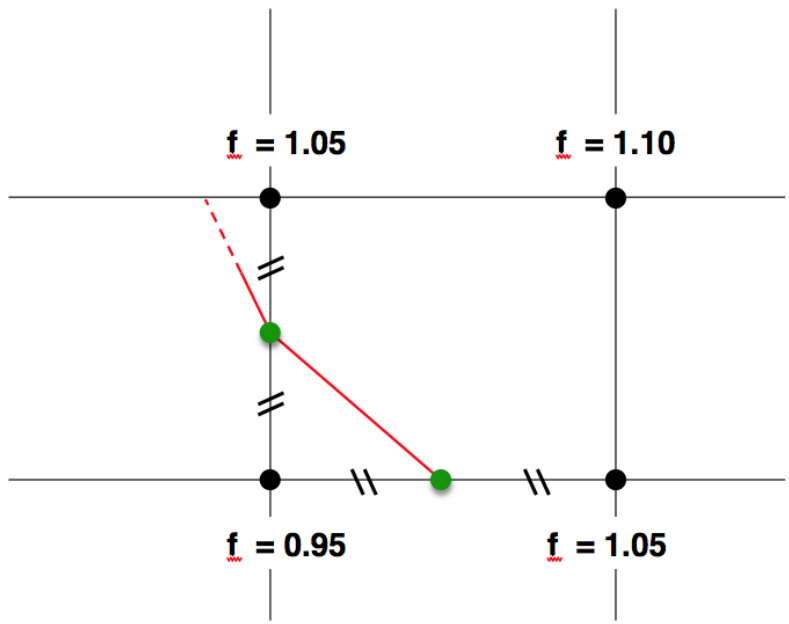
セルを周囲の 4 つの辺、それぞれの両端の頂点での f の値が 1.0 を「挟めば」その辺を等高線が通る。辺上のどの位置を等高線が横切るかは、線形補間をすればよい。下の図はちょうど中点を通る例である。



このように一つのセルに対して行う処理を次々と順番に、全てのセル対して行えば、等高線ができあがる。

このアルゴリズムは marching squares と呼ばれる。

等高線とは、2 次平面上に分布するスカラー場 $f(x, y)$ を曲線の分布図に変換して可視化する手法と言える。

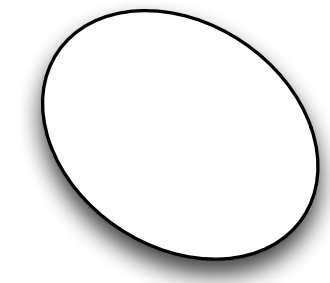


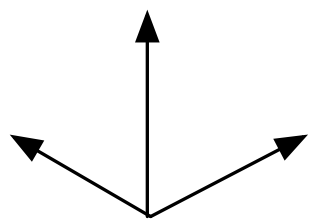
3 次元データの可視化：等値面

等高線の「3 次元版」を考えよう。

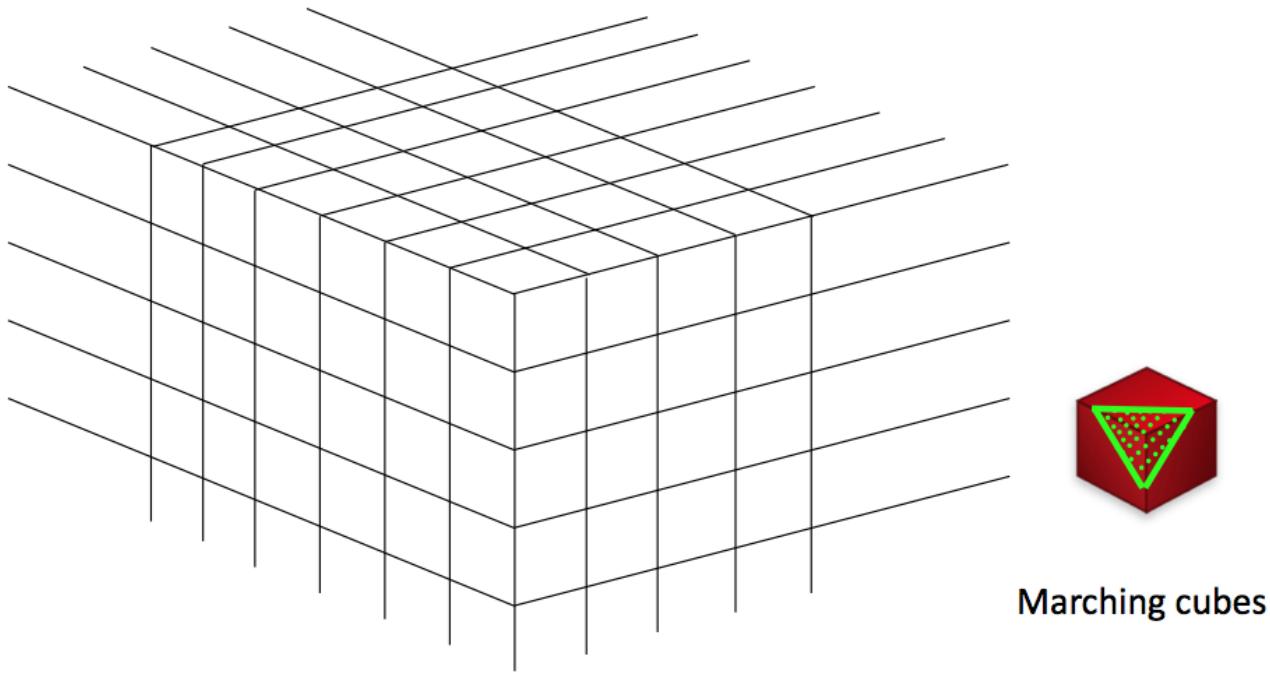
3 次元空間中の関数 $f(x, y, z)$ がある値（例えば 1.0）をとる点の集合は、方程式 $f(x, y, z) = 1.0$ で決まる曲面である。

これを等値面という。





等値面を描くためのアルゴリズムとして Marching Cubes（米国特許（US4710876A, 1985）であったが、既に失効している）がある。



gnuplot 入門

可視化ソフトウェア

様々な可視化アルゴリズムを実装した便利なソフトウェアが多数開発されている。

- 市販可視化ソフト
  - IDL, AVS/Express, Tecplot, ...
- 無料可視化ソフト
  - ParaView, VisIt, Amira, Vapor, ...
- 数式処理ソフトの可視化機能を使う
  - Mathematica, MATLAB, ...
- 基本ライブラリ
  - VTK, Visualization Library, ...

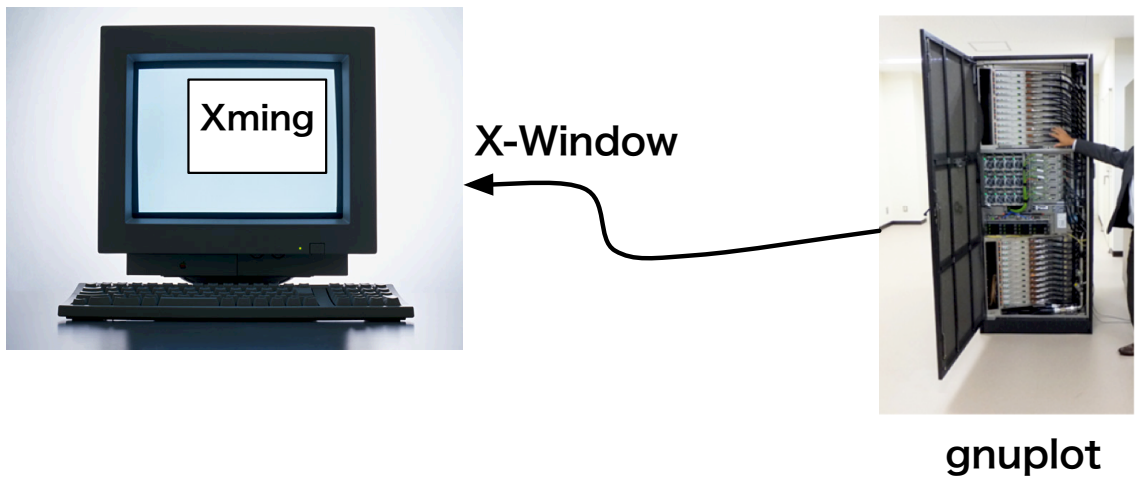
この演習では、gnuplot（http://www.gnuplot.info/）を利用する。

gnuplot とは

http://www.gnuplot.info/faq/faq.html

演習室の環境設定

- π-computer （のログインノード）にインストールされている gnuplot を使う
- グラフは（Unix の）X-Window システム （X11）
- 端末の（マイクロソフトの）Windows システムで、X11 のクライアントを立ち上げる
- デフォルトでは外部の X11 アプリケーションは拒否する設定なので、それを変更する必要がある。



演習室での設定手順

各自の端末で：

1. 全てのプログラム → Xming→Xming（特になにも起きない）
2. Tera term を立ち上げる
   - 2.1 →「キャンセル」
   - 2.2 →「設定」
   - 2.3 →「SSH 転送」
   - 2.4 → リモートの (X) アプリケーションを ··· にチェックが入っていなければチェック
   - 2.5 → ファイル → 「新しい接続」→ ログイン

（参考）UNIX 系システムからの設定手順

1. X11 が使えるようにする（普通は何もする必要はない）
2. Mac では OS のバージョンによってはオプションインストールが必要かも（X11.app または XQuartz.app）。
3. `ssh -X my_id@pi.ircpi.kobeu.ac.jp`

gnuplot の立ち上げ

- 上記の手順で X11 アプリケーションの「貼り付け」を許可した上で、
- （π-computer 上で）gnuplot と打つ。

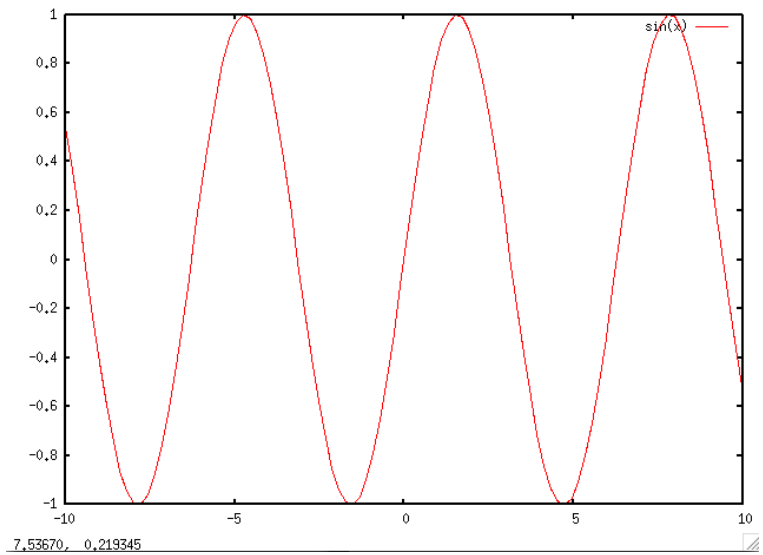
確認

以下のコマンドプロンプトが出れば gnuplot の立ち上げ成功。

gnuplot>

ここで gnuplot> plot sin(x)

と入れてみよう。以下のようなグラフが表示されれば成功。



gnuplot のヘルプと終了方法

ヘルプは gnuplot のプロンプトで help と打つ。

gnuplot の終了はプロンプトで quit と打つ。

gnuplot の単項演算子

gnuplot 4.4 のマニュアルより引用

単項演算子		
記号	例	説明
-	-a	マイナス符号
+	+a	プラス符号 (何もしない)
~	~a	* 1 の補数 (ビット反転)
!	!a	* 論理的否定
!	a!	* 階乗
$	$3	* ‘using‘ 内での引数/列指定

gnuplot の二項演算子

gnuplot 4.4 のマニュアルより引用

二項演算子		
記号	例	説明
**	a**b	累乗
*	a*b	積
/	a/b	商
%	a%b	* 余り
+	a+b	和
-	a-b	差
==	a==b	等しい
!=	a!=b	等しくない
<	a<b	より小さい
<=	a<=b	以下
>	a>b	より大きい
>=	a>=b	以上

&	a&b	* ビット積 (AND)
^	a^b	* ビット排他的論理和 (XOR)
|	a|b	* ビット和 (OR)
&&	a&&b	* 論理的 AND
||	a||b	* 論理的 OR
=	a = b	代入
,	(a,b)	累次評価
.	A.B	文字列の連結
eq	A eq B	文字列が等しい
ne	A ne B	文字列が等しくない

gnuplot の組み込み関数

gnuplot 4.4 のマニュアルより引用

数学ライブラリ関数		
関数	引数	戻り値
abs(x)	任意	x の絶対値, $\|x\|$; 同じ型
abs(x)	複素数	x の長さ, $\sqrt{\mathrm{real}(x)^2 + \mathrm{imag}(x)^2}$
acos(x)	任意	$\cos^{-1} x$ (アークコサイン)
acosh(x)	任意	ラジアンでの $\cosh^{-1} x$ (逆双曲余弦)
arg(x)	複素数	x の偏角
asin(x)	任意	$\sin^{-1} x$ (アークサイン)
asinh(x)	任意	ラジアンでの $\sinh^{-1} x$ (逆双曲正弦)
atan(x)	任意	$\tan^{-1} x$ (アークタンジェント)
atan2(y,x)	整数または実数	$\tan^{-1}(y/x)$ (アークタンジェント)
atanh(x)	任意	ラジアンでの $\tanh^{-1} x$ (逆双曲正接)
EllipticK(k)	実数 k $\in$ (-1:1)	$K(k)$ 第 1 種完全楕円積分
EllipticE(k)	実数 k $\in$ [-1:1]	$E(k)$ 第 2 種完全楕円積分
EllipticPi(n,k)	実数 n<1, 実数 k $\in$ (-1:1)	$\Pi(n,k)$ 第 3 種完全楕円積分

besj0(x)	整数または実数	j_0 ベッセル関数 (0 次ベッセル関数)
besj1(x)	整数または実数	j_1 ベッセル関数 (1 次ベッセル関数)
besy0(x)	整数または実数	y_0 ベッセル関数 (0 次ノイマン関数)
besy1(x)	整数または実数	y_1 ベッセル関数 (1 次ノイマン関数)
ceil(x)	任意	$\lceil x \rceil$, x (の実部) 以上の最小の整数
cos(x)	任意	x のコサイン $\cos x$
cosh(x)	任意	$\cosh x$, x のハイパボリックコサイン
erf(x)	任意	erf(real(x)), x の 実部の誤差関数
erfc(x)	任意	erfc(real(x)), 1.0 - (x の実部の誤差関数)
exp(x)	任意	e^x, x の指数関数
floor(x)	任意	$\lfloor x \rfloor$, x (の実部) 以下の最大の整数
gamma(x)	任意	gamma(real(x)), x の実部のガンマ関数
ibeta(p,q,x)	任意	ibeta(real(p, q, x)), p,q,x の実部の不完全ベータ関数
inverf(x)	任意	x の実部の逆誤差関数
igamma(a,x)	任意	igamma(real(a, x)), a,x の実部の不完全ガンマ関数
imag(x)	複素数	x の虚数部分 (実数)
invnorm(x)	任意	x の実部の逆正規分布関数

int(x)	実数	x の整数部分 (0 に向かって丸め)
lambertw(x)	実数	Lambert W 関数
lgamma(x)	任意	lgamma(real(x)), x の実部のガンマ対数関数
log(x)	任意	$\log_e x$, x の自然対数 (底 e)
log10(x)	任意	$\log_{10} x$, x の対数 (底 10)
norm(x)	任意	x の実部の正規分布 (ガウス分布) 関数
rand(x)	任意	rand(real(x)), 疑似乱数生成器
real(x)	任意	x の実部
sgn(x)	任意	$x > 0$ なら 1, $x < 0$ なら -1, $x = 0$ なら 0. x の虚部は無視
sin(x)	任意	$\sin x$, x のサイン
sinh(x)	任意	$\sinh x$, x のハイパボリックサイン
sqrt(x)	任意	$\sqrt{x}$, x の平方根
tan(x)	任意	$\tan x$, x のタンジェント
tanh(x)	任意	$\tanh x$, x のハイパボリックタンジェント

演習

$$f(x) = x^x$$

のグラフを描け。

解答

```
plot x**x title "x\^x"
```

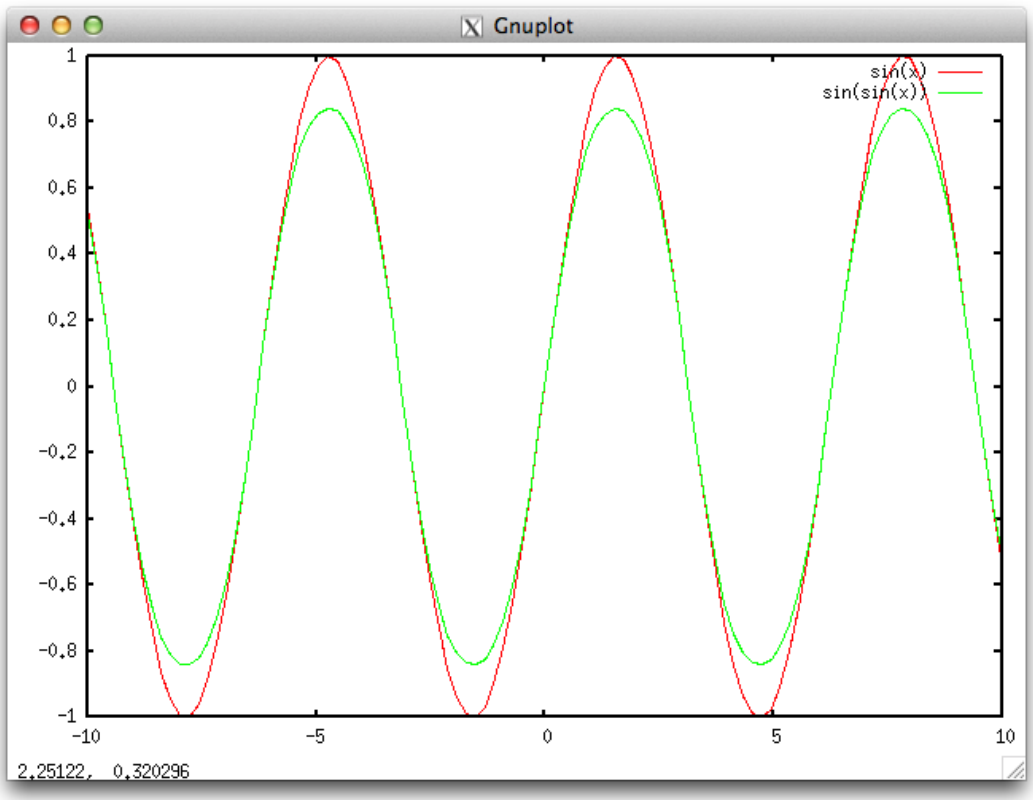
複数のグラフ

カンマで区切る

```
plot sin(x), sin(sin(x))
```

グラフを区別するには

```
plot sin(x) title "sin(x)", sin(sin(x)) title "sin(sin(x))"
```



様々なパラメータ（set コマンド）

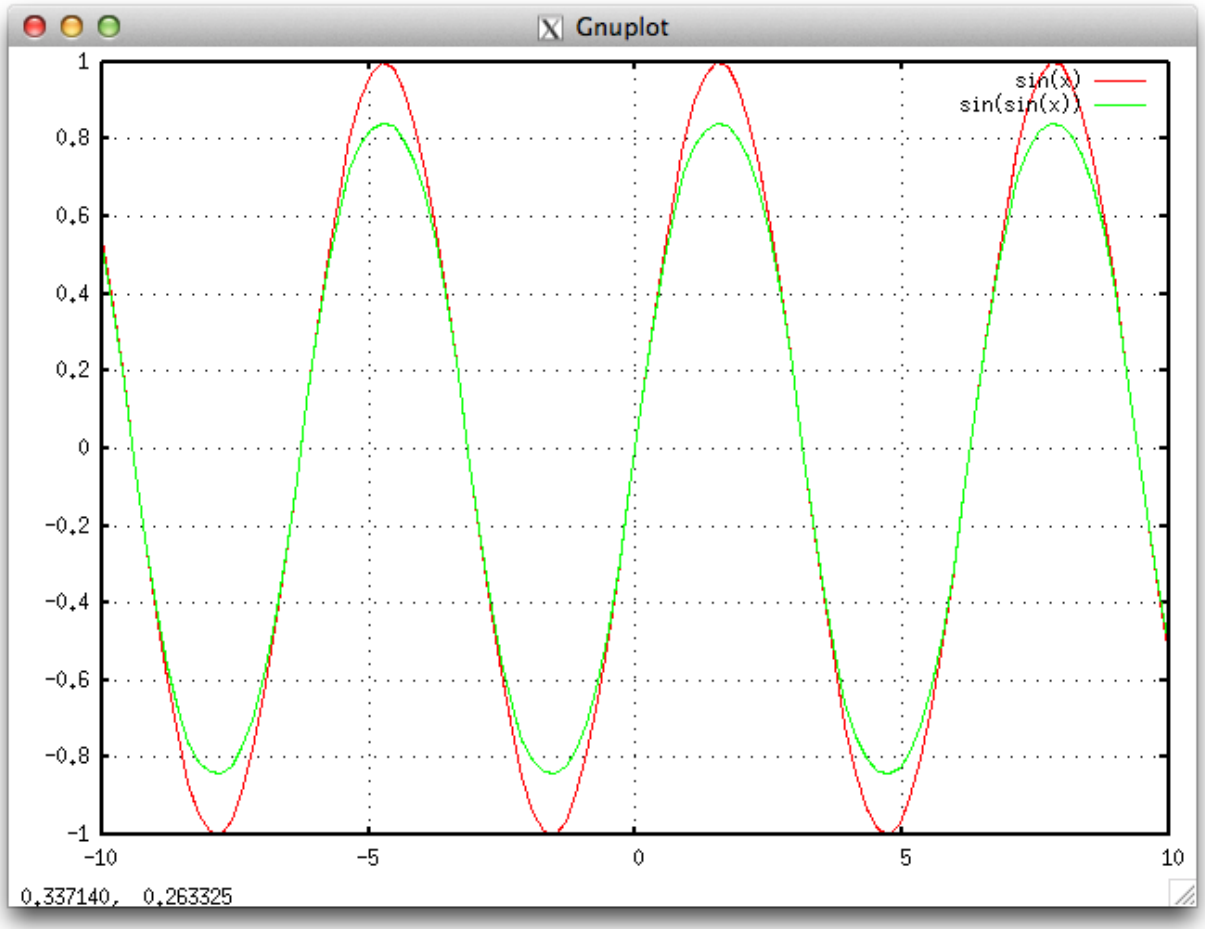
```
set title "y=x^x"
set xlabel "x (no units)"
set ylabel "y (no units)"
plot x**x
```

定義域と値域、replot

```
set xrange [0:5]
replot
```

グリッド表示

```
set grid
replot
```



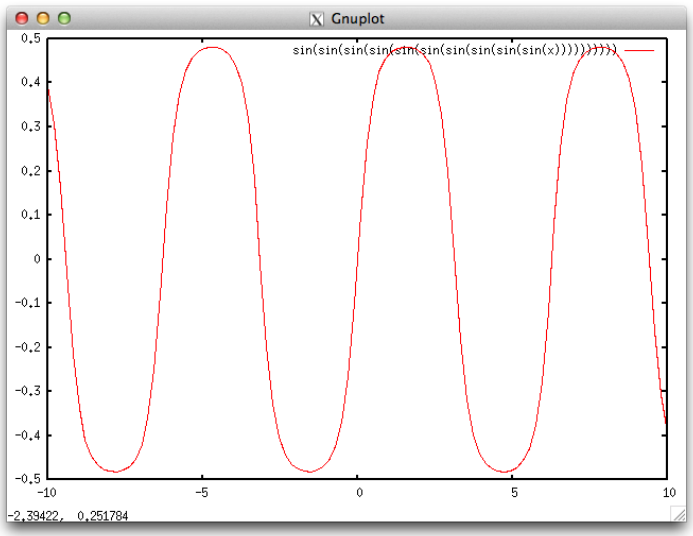
関数の定義

```
s2(x) = sin(sin(x))

s4(x) = s2(s2(x))

s10(x) = s4(s4(s2(x)))

plot s10(x)
```



データのファイルからの読み込み

gnuplot には、ファイルに書き込まれた離散データを読み込み、それをグラフにする機能がある。

グレゴリー・ライプニッツ級数

$$\pi = 4\left(\frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \cdots\right)$$

第 n 項までの級数がどの程度 π に近いかみるプログラム

leibniz.f95

```
1    4.0000000000000000
2    2.6666666666666670
3    3.4666666666666668
4    2.8952380952380956
5    3.3396825396825403
.              .
.              .
.              .
```

【演習】データ作成

(1) leibniz.f95 を gfortan コンパイルし、実行せよ。

```
gfortran leibniz.f95
./a.out （100 行の長い出力）
./a.out | head （あるいは more / less / tail コマンド）
./a.out > test.data
```

(2) ファイル test.data の中身を確認せよ

（エディタで開くよりも more / less / head / tail コマンドで見る方が早い。

less test.data （スペースで次、b で前、q で終了）

```
# sample data generated by leibniz.f95
#       term    sum
#
           1    4.0000000000000000
           2    2.6666666666666670
           3    3.4666666666666668
           4    2.8952380952380956
           5    3.3396825396825403
           6    2.9760461760461765
           7    3.2837384837384844
```

【演習】1次元グラフ

gnuplot を立ち上げ、コマンドプロンプトに

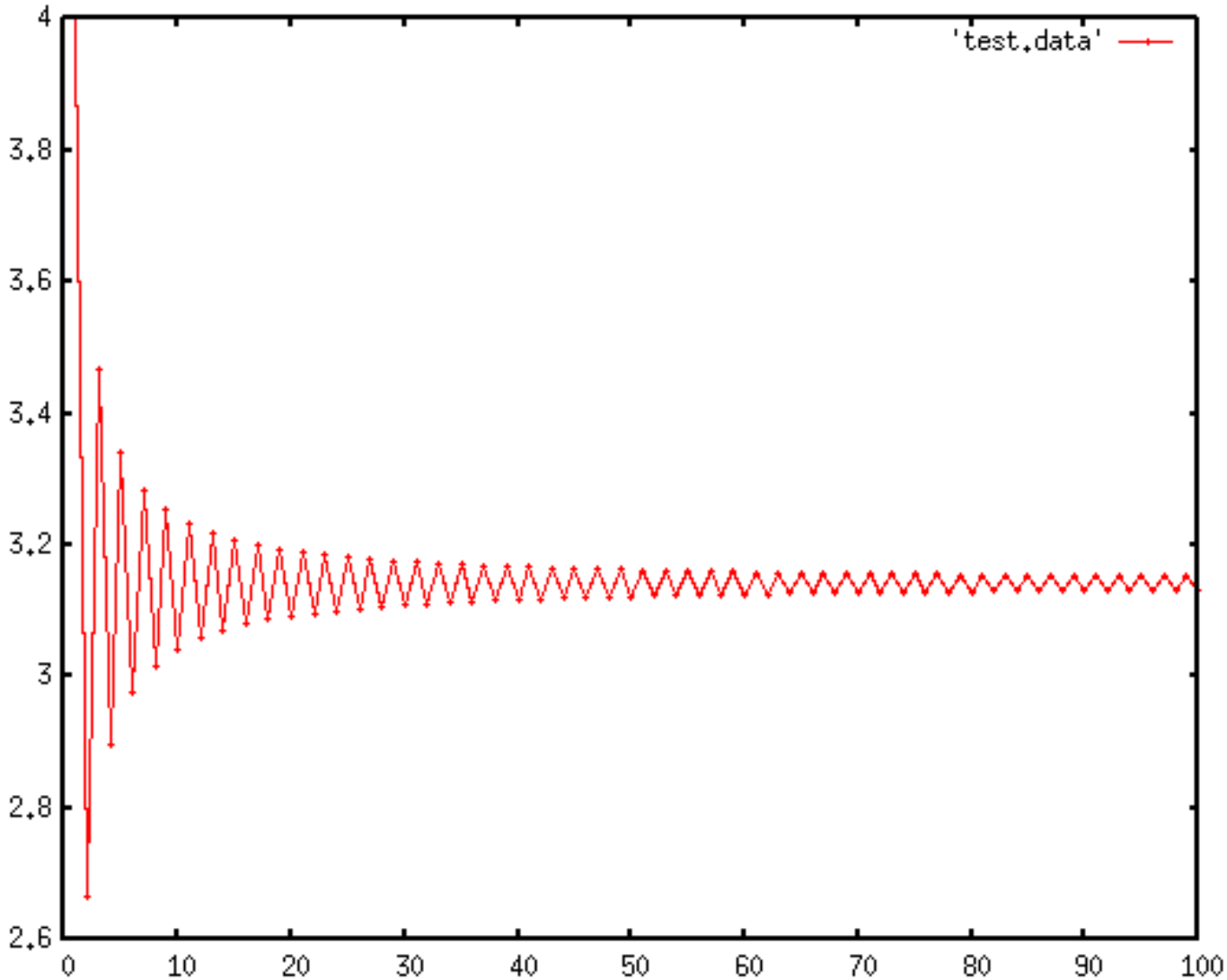
plot 'test.data' w lp

と入れよ。

---

lp は linespoints の略で、線（line）と点（point）を表示することを意味する。（w linespoints と書いてもよい。）

出力例



gnuplot の入力ファイル

- # はコメント開始
- 1 行に x, y 値のペア
- デフォルトでは第 1 列が plot の x 座標、第 2 列が y 座標（変更可能）

【演習】オプションの変更

(1) ラベルの文字を消す。

gnuplot のコマンドプロンプトで 2 行入れる：

```
unset key
replot
```

(2) 縦軸の表示範囲を調整をする。

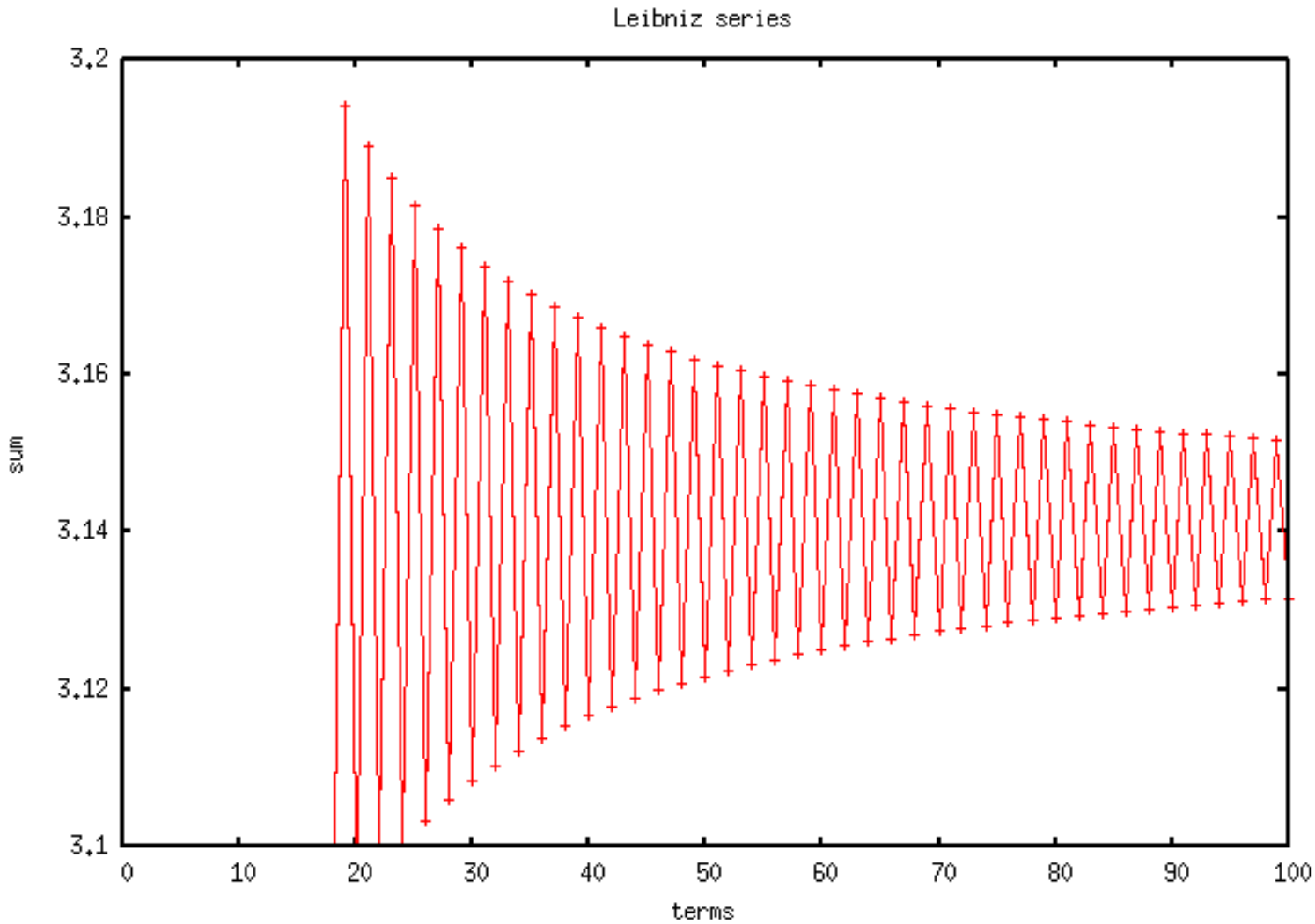
gnuplot のコマンドプロンプトで 2 行入れる：

```
set yrange [3.1:3.2]
replot
```

(3) 図全体のタイトルと、x 軸、y 軸の説明を入れる。

```
set title "Leibniz series"
set xlabel "terms"
set ylabel "sum"
replot
```

出力例


Leibniz series
3.2
3.18
3.16
3.14
3.12
3.1
sum
0
10
20
30
40
50
60
70
80
90
100
terms

演習

- leibniz.f95 を改訂して 200 項までの和をとるプログラムにせよ。
- そのプログラムを実行し、どの程度 π に近づくか gnuplot で見よ。

gnuplot スクリプト

- gnuplot ではコマンドプロンプトに手で入力する内容をファイルから読み込ませることが出来る。⇒ gnuplot script

leibniz.gp （拡張子は任意）

```
#
# leibniz.gp
#
set yrange [3.1:3.2]
set xlabel "terms"
set ylabel "sum"
plot "test.data" w lp
pause -1
```

最後の pause -1 は（一瞬だけ）表示してすぐに終了してしまうのを防ぐため。

スクリプトの実行

gnuplot がまだ立ち上がっていたら quit コマンドで終了し、改めて shell から

```
gnuplot leibniz.gp
```

と打て。

レポート課題（演習）

test.data のデータに重ねて、$y = \pi$ の直線も描くような gnuplot スクリプトファイルを作り、leibniz2.gp とせよ。

（ヒント： gnuplot では pi という変数に π が入っている。定数グラフは plot pi で描ける。）

【提出するもの】leibniz.gp と leibniz2.gp の差分をメールで。

【提出方法】`diff leibniz.gp leibniz2.gp | mail kage`

【提出期限】5/21（木）12:00

アンケート

1. 自分の学籍番号の桁に現れる数字を足せ。その和を n とする。(例：135X204X ならば $n = 1+3+5+2+0+4 = 15$)
2. n^n を 4 で割った余りに 3 を足して、それを m とせよ。
   ちなみに Unix では `echo "15^15 % 4 + 3" | bc` で計算できる。
3. その m を使い、Emacs で Ctr-u m Esc-x hanoi と打て。
4. 「それ」が終わった人から、アンケートに回答。