

# 可視化（第2回）

## 計算科学演習 I

神戸大学 システム情報学研究科 計算科学専攻 陰山

2015.07.14

# 事務連絡

# 講師変更

- 坂本先生 ⇒ 陰山

## 今後のスケジュール

- 7/14 可視化（第2回）（陰山）
- 7/21 実践編Ⅰ（陰山）
- 7/28 実践編Ⅱ（坂本）

# 環境設定

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 → 最初のダイアログで「キャンセル」
  - 2.2 → Tera term の「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いければチェック
  - 2.5 → ファイル → 「新しい接続」 → ログイン

## (参考) Mac からの設定

1. X11 を立ち上げる (X11.app または XQuartz.app)。
2. `ssh -X my_id@pi.ircp.kobe-u.ac.jp`
3. ( $\pi$ -computer 上で) `gnuplot` と入力

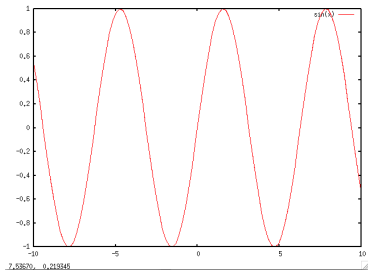
## 確認

以下のコマンドプロンプトが出るはず。

```
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できれば成功。



## 今日の演習の準備

```
cp -r /tmp/160714/ your_directory
```

ls -l で4つ (data, sierpinski, src01d, src02d) のディレクトリがあることを確認。

```
cd your_directory/src01d
```



# サンプル 1D データ作成

ソースコードの確認。

`data1d_generator.f95`

ポイント

- `module` の活用
- 単精度・倍精度浮動小数点数の指定方法

## const\_m モジュール

単精度・倍精度浮動小数点数用種別選択子の（これまでとは別の）指定方法を紹介します：

```
module const_m
  implicit none
  !---<fortran constants>---
  integer, parameter :: SP = selected_real_kind(6)
  integer, parameter :: DP = selected_real_kind(15)

  !---<simulation constants>---
  integer, parameter :: NN = 101
end module const_m
```

## 【演習】

```
f95 data1d_generator.f95 && ./a.out > data.1d
```

代わりに `make data.1d` と打ってもよい。(make コマンドの使い方は後述。)

## data.1d

```
kage@pi data_sample_1d]$ head data.1d
# sample data 1d
-10.000000000000000      0.54402111088936977
-9.8000000000000007     0.36647912925192838
-9.5999999999999996     0.17432678122297965
-9.4000000000000004     -2.47754254533577647E-002
-9.1999999999999993     -0.22288991410024764
-9.0000000000000000     -0.41211848524175659
-8.8000000000000007     -0.58491719289176169
-8.5999999999999996     -0.73439709787411334
-8.4000000000000004     -0.85459890808828043
[kage@pi data_sample_1d]$
```

## 復習：gnuplotの入力ファイルフォーマット

- # 以降はコメント
- 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

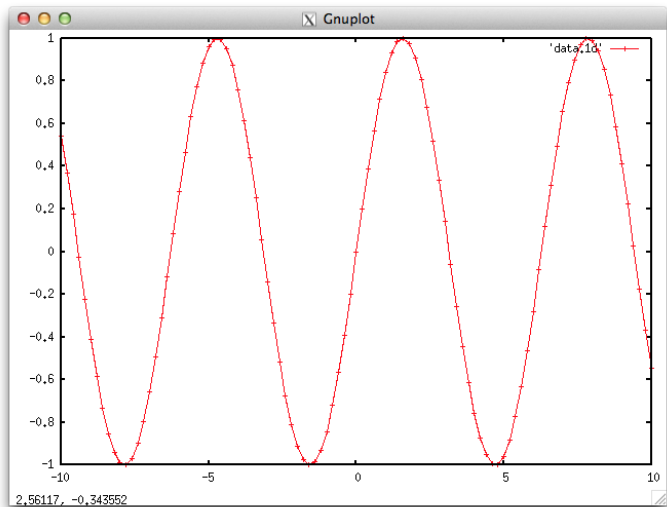
gnuplot を立ち上げて

```
gnuplot> plot 'data.1d' w lp
```

としてみよう。(PDF からコピーするとシングルクォートが変わってしまうかも。)

w lp は with 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



## 復習：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

ディレクトリ src01d の中に graph1d.gp がある。

```
#
# A simple sample gnuplot script: graph1d.gp
#
#
# set size square                # same side lengths for x and y
# set size 0.65, 1              # other way to set regular rectan
set xlabel 'x'                  # x-axis
set ylabel 'func'              # y-axis
set xrange[-10:10]            # x coords range
set yrange[-1:1]              # y coords range
set title 'graph 1d'
plot 'data.1d' w lp
pause -1
```



## 【演習】

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 で

```
gnuplot graph1d.gp
```

と入力。

```
make graph
```

としてもよい。

## Makefile (make コマンド) の復習

target を source から作る方法を記述。

```
target: source  
[__タブ__] command_to_make_target_from_source
```

target と source のタイムスタンプを比較して、source の方が新しい時のみ target を新たに作る。

# Makefile (make コマンド) の復習

## 再帰的記述

```
target: source01
```

```
[_ _ タブ _ _] command_to_make_target_from_source01
```

```
source01: source02
```

```
[_ _ タブ _ _] command_to_make_source01_from_source02
```

```
source02: source03
```

```
[_ _ タブ _ _] ...
```

# アニメーション

## 【コード解説】

$\text{recsin}(x, n) = \sin(\sin(\sin(\dots(\sin(x)))) \dots)$  を計算するプログラム

`anim_data_generator.f95`

ポイント

- 再帰関数
- 連番ファイルの作成

## 【準備】

- ソースコード： `anim_data_generator.f95`
- 実行： `make anim.data`

連番ファイルが 50 個生成されたことを確認。

## gnuplot によるアニメーション

- アニメーション=静止画の連続。
- gnuplot のスクリプトを使えばアニメーションも簡単にできる。
- `pause t` とすると  $t$  秒待ってから表示。
- `pause t` 小数の  $t$  がサポートされるかどうかはプラットフォーム依存。

## アニメーション用スクリプトの例

```
set xlabel 'x'           # x-axis
set ylabel 'func'       # y-axis
set xrange[-10:10]     # x coords range
set yrange[-1:1]       # y coords range
set title 'anim 1d'
plot 'data.1d.000' w lp
pause 5
plot 'data.1d.001' w lp
pause 0.5
plot 'data.1d.002' w lp
pause 0.5
plot 'data.1d.003' w lp
.
.
pause -1    # Enter キーが押されるまで待つ
```



## アニメーション用スクリプト

- 100 フレームのアニメーション → plot コマンドを 100 行
- 手で書くのは大変！

スクリプト作成プログラム: `anim_plotscript_generator.f95`

```
make anim.gp
```

`anim.gp` が生成される。中身を確認。

# アニメーションの実行

make anim

# 2次元可視化

## 2次元可視化

これまでは1次元データの可視化であった。

これから `gnuplot` を利用して2次元データの可視化を行う。

`plot` コマンドではなく `splot` コマンドを使う

```
gnuplot > splot [関数 f(x,y)]
```

例：`gnuplot > splot x**2 + y**2`

# 演習

plot コマンドで様々な  $f(x,y)$  を描いてみよう。

## plot のパラメータ

gnuplot のプロンプトに以下を入力せよ。

```
plot exp(-x**2-y**2)
```

```
set isosamples 50 # 描画に使われる線の数
```

```
replot
```

```
set xrange [-2:2] # x の定義域
```

```
replot
```

```
set yrange [-2:2] # y の定義域
```

```
replot
```

## 視点の設定

```
show view # 現在の視点の表示
```

```
set view 60,0 # x軸の周りに60度回転
```

```
replot
```

```
set view 60,30 # x軸の周りに60度回転し、その後にz軸の周りに  
30度回転
```

## 対話的視点移動

グラフをマウスでドラッグしてみよう。

自分の描いたグラフを「下」から見てみよう。



# 隠面処理

```
set hidden3d    #隠面処理  
replot
```

## 等高線の表示

```
set contour base    # 等高線の表示
replot
set cntrparam levels 30    # 等高線の数の変更
replot
```

## 等高線だけの表示

```
unset surface # 等高線だけの表示  
replot
```

## 色分布による表示

```
set pm3d at b      # draw with colored contour  
replot
```

# データの2D表示

## 2D データフォーマット

x00 y00 関数值

x01 y00 関数值

x02 y00 関数值

•

•

•

x09 y00 関数值

(空行)

x00 y01 関数值

x01 y01 関数值

•

•

x09 y01 関数值

(空行)

•

•

x09 y09 関数值

## サンプルデータの作成

```
cd [your directory]/src02d
```

# ソースコード解説

heat4.f95



## 2D データ出力ルーチン (前半)

正方形上 (x,y 平面上) に分布する温度をすべて書き出す。

```

subroutine print__profile_2d(p,jj,f)
  type(ranks_t), intent(in) :: p
  type(span_t),  intent(in) :: jj
  real(DP), dimension(0:NGRID+1, &
                      jj%stt-1:jj%end+1), intent(in) :: f
  real(DP), dimension(0:NGRID+1,0:NGRID+1) &
                      :: f_global ! 2d prof to be saved
  integer  :: counter = 0 ! has save attrib.
  type(span_t)  :: jj2 ! used for f_global
  character(len=4)  :: serial_num ! put on file name
  character(len=*) , parameter :: base = "../data/temp.2d."
  integer  :: i, j

```

## 2D データ出力ルーチン (後半)

```
jj2 = adjust_jstart_and_jend(p, jj)
write(serial_num, '(i4.4)') counter
f_global(:, :) = set_prof_2d(jj, jj2, f)
if ( p%myrank==0 ) then
  open(10, file=base//serial_num)
  do j = 0 , NGRID+1
    do i = 0 , NGRID+1
      write(10,*) i, j, f_global(i, j)
    end do
    write(10,*) ' ' ! gnuplot requires a blank line here.
  end do
  close(10)
end if
counter = counter + 1
end subroutine print__profile_2d
```

## データ作成

1. `mpifrtpx heat4.f95`
2. `pjsub heat4.sh`

`../data` ディレクトリにデータ (`temp.2d.0000`) が出力されるはず。

上の二つのステップは

`make data`

だけでも OK

## 出力データの確認

../data ディレクトリ中の連番つきファイル temp.2d.0000 の中身は以下のようになっているはず。確認せよ。

55	35	0.1165598588705999
56	35	9.9624877672293416E-002
57	35	8.1734108631726782E-002
58	35	6.2857224006520482E-002
59	35	4.2963409431420671E-002
60	35	2.2021479795155254E-002
61	35	0.0000000000000000
0	36	0.0000000000000000
1	36	2.1867122785152873E-002
2	36	4.2655284590767971E-002
3	36	6.2396502500601705E-002
4	36	8.1122529495226178E-002

•

## 2次元表示 gnuplot スクリプト

heat4\_lines.gp

```

#
# A sample gnuplot script: heat4_lines.gp
#
#   [ line contours ]
#
# set size square           # same side lengths for x and y
set size 0.65, 1          # same side lengths for x and y
set xlabel "i"            # x-axis
set ylabel "j"            # y-axis
set xrange[0:50]          # i-grid min & max
set yrange[0:50]          # j-grid min & max
set nosurface              # do not show surface plot
unset ztics                # do not show z-tics
set contour base           # enables contour lines
set cntrparam levels 10    # draw 10 contours
set view 0,0               # view from the due north
set title "Temperature"
plot "data/temp_2d_0000" using 1:2:3 w l # with lines

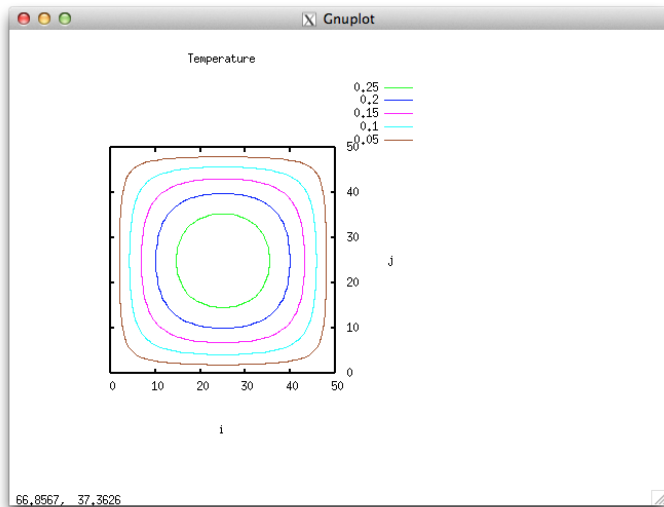
```

## 描画

もしもまだ `gnuplot` の対話モード（プロンプトが出ている状態）ならそこから抜けて、UNIX シェルから以下を入力。

```
gnuplot 'heat4_lines.gp'
```

## 結果の例



## 色分布による可視化（静止画）

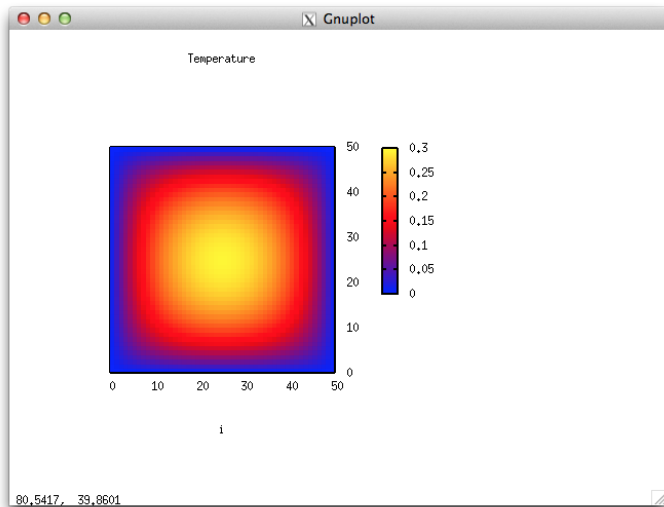
```
gnuplot 'heat4_colors.gp'
```



## heat4\_colors.gp

```
#  
# A sample gnuplot script: heat4_plot_contour_colors.gp  
#  
# [ color contours ]  
#  
# set size square           # same side lengths for x and y  
set size 0.65, 1           # same side lengths for x and y  
set xlabel "i"             # x-axis  
set ylabel "j"             # y-axis  
set xrange[0:50]          # i-grid min & max  
set yrange[0:50]          # j-grid min & max  
set palette defined (0 "blue", 0.15 "red", 0.3 "yellow")  
set nosurface              # do not show surface plot  
unset ztics                # do not show z-tics  
set pm3d at b              # draw with colored contour  
set view 0,0               # view from the due north  
set title "Temperature "  
plot "../data/temp.2d.0000" using 1:2:3  
pause -1
```

## 結果の例



## 鳥瞰図

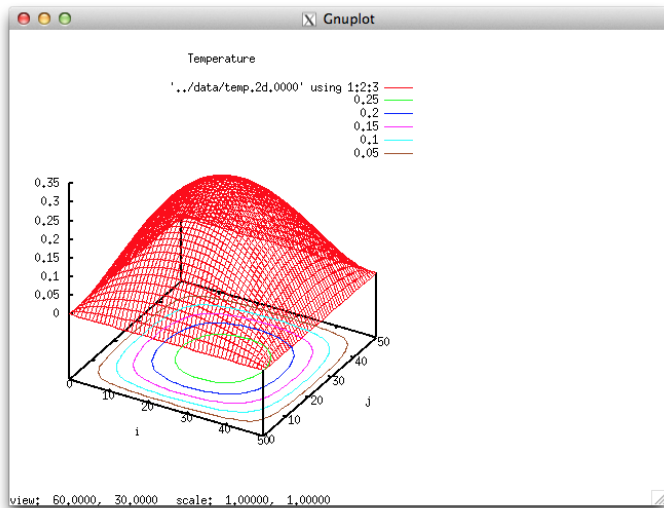
空をとぶ鳥から見下ろしたような図は一般に鳥瞰図 (bird's eye view) と呼ぶ。

## gnuplot スクリプト

ファイル名: plot4\_plot\_birdseyeview.gp

```
#
# a sample gnuplot script: plot4_plot_birdseyeview.gp
#
# [ Bird"s Eye View ]
#
# set size square          # same side lengths for x and y
set size 0.65, 1
set xlabel "i"            # x-axis
set ylabel "j"           # y-axis
set xrange[0:50]         # i-grid min & max
set yrange[0:50]        # j-grid min & max
set contour base         # enables contour lines
set cntrparam levels 10  # draw 10 contours
# set palette defined (0 "blue", 0.15 "red", 0.3 "yellow")
# set pm3d                # draw with colored contour
set title "Temperature  "
splot "../data/temp.2d.0000" using 1:2:3 w l
pause
```

## 結果の例



## 鳥瞰図のアニメーション

plot の view パラメータを調整してアニメーションを作る。

- スクリプト生成プログラム  
heat4\_plot\_rotating\_birdseyeview\_generator.f95 を確認せよ。
- gfortran heat4\_plot\_rotating\_birdseyeview\_generator.f95
- ./a.out > anim.gp
- anim.gp の中身を確認
- gnuplot anim.gp と入力（最初に 5 秒間静止）

上のステップは

```
make anim
```

だけで OK（make clean してから make data とし、../data にデータができたのを確認してから make anim と打ってみよ）。

# レポート課題

## シェルピンスキーギャスケット

```
cd [your_directory]/sierpinski
```

ここに「OpenMP 第2回」(三宅先生)で見たプログラム `sierp.f95` がある。  
復習しよう。



## アニメーション用にコードを改訂

sierp.f95  $\Rightarrow$  sierp2.f95

ポイント

- omp section による並列計算
- ギャスケット生成過程全体を関数化（引数：何回乱数を振るか）
- 連番ファイルの生成（整数から3桁の文字列）
- module の利用（private と public）

# 実行

ソースコード：sierp2.f95 ジョブスクリプト：enshu.sh（谷口先生作成）

- コンパイル：frtpx -Kopenmp sierp2.f95
- ジョブ投入：pjsub enshu.sh

上の一連の操作は `make data` で実行してもよい。（Makefile を見よ。）

ジョブ実行結果: 連番ファイル（data.000 ... data.099）

data.??? の中身を見てみよう。

`make graph` を実行しよう。

# 課題

## 課題

sierp2.f95 では

- 確率  $1/3$  で  $(x^{n+1}, y^{n+1}) = (x^n/2 + 1, y^n/2)$
- 確率  $1/3$  で  $(x^{n+1}, y^{n+1}) = (x^n/2 - 1, y^n/2)$
- 確率  $1/3$  で  $(x^{n+1}, y^{n+1}) = (x^n/2, y^n + 1)$

としている。これを

- 確率  $1/10$  で  $(x^{n+1}, y^{n+1}) = (x^n/2 + 1, y^n/2)$
- 確率  $1/10$  で  $(x^{n+1}, y^{n+1}) = (x^n/2 - 1, y^n/2)$
- 確率  $8/10$  で  $(x^{n+1}, y^{n+1}) = (x^n/2, y^n + 1)$

と変更したプログラム **sierp3.f95** を作成し、その結果を **gnuplot** で確認せよ。

## 【提出方法】

**【提出先メールアドレスがいつもと違うことに注意！】**

提出方法： `diff sierp2.f95 sierp3.f95 | mail kage`

締め切り： 7/20（水曜日） 23:59