

# 計算科学演習 A

## 実践編 1

陰山

計算科学専攻

2018.07.19

# 内容

「床暖房問題」を例にとり、計算科学の実践的演習を行う。

- 問題の定式化
- 離散化
- コーディング
- 時間計測
- 可視化
- 並列化 (MPI + OpenMP (ハイブリッド並列化))
- 大規模並列 (最大 84 ノード = 1344 コア) 計算

準備

# サンプルコードのコピー

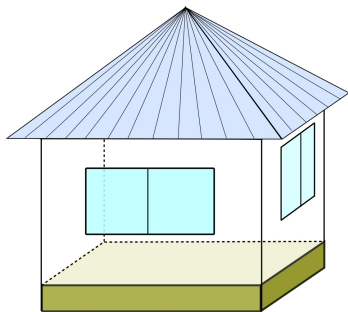
```
cp -r /tmp/180719 自分のディレクトリ
```

注意：書き込み権限がないファイルを編集する場合は `chmod` を忘れずに。

# 問題設定

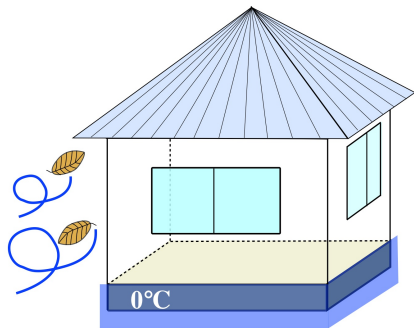
## 問題設定： 床暖房システム

一間の家がある。床は長方形。外は冬。床暖房システムがあるので、家の中は暖かい。



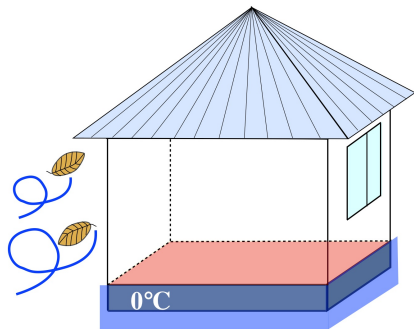
# 問題設定：床暖房システム

外気温は0度。壁も0度。



## 問題設定：床暖房システム

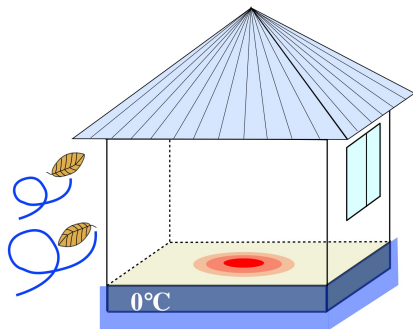
この床暖房システムは床全体を一様に加熱する。



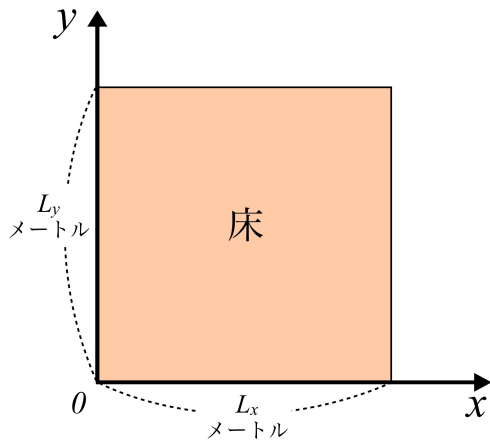


## 問題設定： 暖房問題

床暖房がオフだと、床面は全体が0度。床暖房をオンにすると床の温度は上がるが、壁に接している部分（床の周囲の長方形の辺上）は0度。問題： 最終的な温度分布は？



# 座標系



# 問題設定

- $L_x \times L_y$  平方メートルの長方形領域
- 辺上の温度は常に  $0^\circ$  (固定)
- 面内に熱源が分布
- 面の熱拡散率  $k$
- 面内の温度分布は？

# 熱伝導（熱拡散）方程式

温度  $T(x, t)$  に対する基本方程式

$$\frac{\partial T}{\partial t} = k \frac{\partial^2 T}{\partial x^2}$$

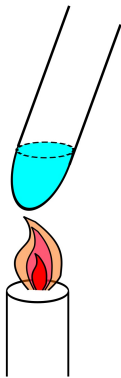
$k$ : 熱拡散係数

## 熱源があるとき

$$\frac{\partial T}{\partial t} = s$$

$s$  : 熱源 (heat source)

ろうそくで温度計を熱している図



# 熱擴散方程式

1D

$$\frac{\partial T(x, t)}{\partial t} = k \frac{\partial^2 T(x, t)}{\partial x^2} + s(x)$$

2D

$$\frac{\partial T(x, y, t)}{\partial t} = k \left\{ \frac{\partial^2 T(x, y, t)}{\partial x^2} + \frac{\partial^2 T(x, y, t)}{\partial y^2} \right\} + s(x, y)$$

## 熱拡散方程式の表現

$$\frac{\partial T(x, y, t)}{\partial t} = k \left( \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) T(x, y, t) + s(x, y)$$

あるいは

$$\frac{\partial T(x, y, t)}{\partial t} = k \nabla^2 T(x, y, t) + s(x, y)$$

$\nabla^2$ : ラプラシアン

# 問題の数学的定式化

$(0, 0) \leq (x, y) \leq (L_x, L_y)$  の長方形領域で

$T(0, y) = T(L_x, y) = T(x, 0) = T(x, L_y) = 0$  という境界条件のもと

$$\frac{\partial T(x, y, t)}{\partial t} = k \nabla^2 T(x, y, t) + s(x, y)$$

という熱拡散方程式を解き、


定常状態 ( $\frac{\partial T}{\partial t} = 0$ ) の温度分布


$T(x, y)$  を求めよ。



# 離散化

# 時間・空間の離散化

連続点 

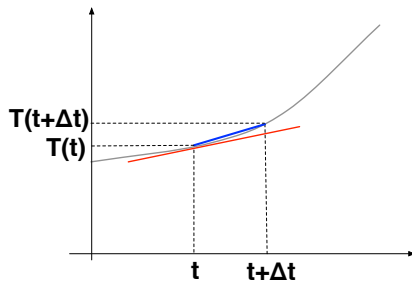
離散点 

# 熱拡散方程式の離散化

$$\frac{\partial T(x, y, t)}{\partial t} = k \nabla^2 T(x, y, t) + s(x, y)$$

を差分法で離散化する。

# 時間微分



$$\frac{\partial T}{\partial t} \sim \frac{T(t + \Delta t) - T(t)}{\Delta t}$$

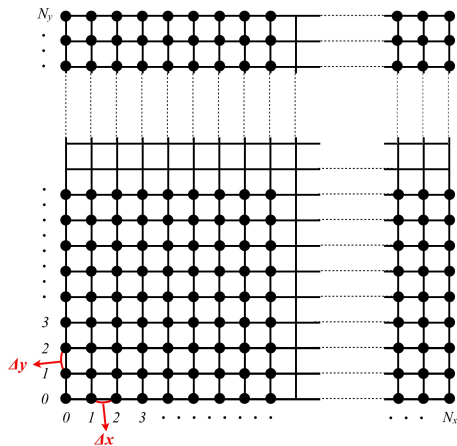
引き算 1 回 + 割り算 1 回で微分を近似 (1/Δt をあらかじめ計算しておけば、割り算の代わりに掛け算)

# 空間微分

$$\frac{\partial^2 T(x_i, y_j)}{\partial x^2} \sim \frac{T(x_{i+1}, y_j) - 2T(x_i, y_j) + T(x_{i-1}, y_j)}{\Delta x^2}$$

$$\frac{\partial^2 T(x_i, y_j)}{\partial y^2} \sim \frac{T(x_i, y_{j+1}) - 2T(x_i, y_j) + T(x_i, y_{j-1}))}{\Delta y^2}$$

# 空間の離散化と格子点の番号付け



## 熱拡散方程式の差分表現

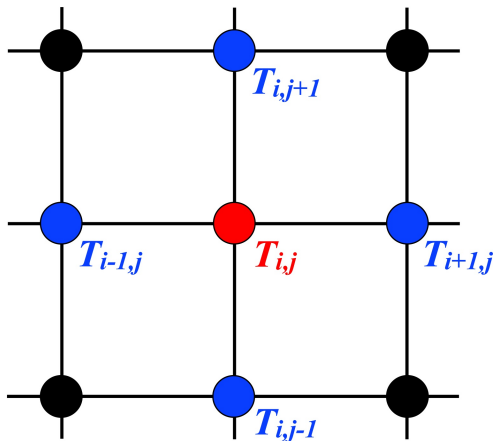
$$\frac{\partial T(x, y, t)}{\partial t} = k \left( \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) T(x, y, t) + s(x, y)$$

$T(x_i, y_j, t) \Rightarrow T_{i,j}(t)$  と略記。

$$\begin{aligned} \frac{T_{i,j}(t + \Delta t) - T_{i,j}(t)}{\Delta t} &= k \left( \frac{T_{i+1,j}(t) - 2T_{i,j}(t) + T_{i-1,j}(t)}{\Delta x^2} \right. \\ &\quad \left. + \frac{T_{i,j+1}(t) - 2T_{i,j}(t) + T_{i,j-1}(t)}{\Delta y^2} \right) \\ &\quad + s_{i,j} \end{aligned}$$

# 差分ステンスル

$$\frac{T(x_{i+1},y_j)-2T(x_i,y_j)+T(x_{i-1},y_j)}{\Delta x^2}, \quad \frac{T(x_i,y_{j+1})-2T(x_i,y_j)+T(x_i,y_{j-1}))}{\Delta y^2}$$





## 時間発展の式

$$\begin{aligned} T_{i,j}(t + \Delta t) &= T_{i,j}(t) \\ &+ \frac{k \Delta t}{\Delta x^2} \{T_{i+1,j}(t) - 2T_{i,j}(t) + T_{i-1,j}(t)\} \\ &+ \frac{k \Delta t}{\Delta y^2} \{T_{i,j+1}(t) - 2T_{i,j}(t) + T_{i,j-1}(t)\} \\ &+ \Delta t s_{i,j} \end{aligned}$$

この式は

$\Delta t$  だけ未来の温度 = 現在の温度分布の四則演算

という形をしている。

## 式変形

$$\begin{aligned} T_{i,j}(t + \Delta t) = & \Delta t \left\{ \frac{k}{\Delta x^2} (T_{i+1,j} + T_{i-1,j}) \right. \\ & \left. + \frac{k}{\Delta y^2} (T_{i,j+1} + T_{i,j-1}) + s_{i,j} \right\} \\ & + \left\{ 1 - \Delta t \cdot 2 \left( \frac{k}{\Delta x^2} + \frac{k}{\Delta y^2} \right) \right\} T_{i,j} \end{aligned}$$

最後の項 ( $T_{i,j}$  に比例する項) の係数がゼロになるように  $\Delta t$  を決める。

つまり

$$\alpha_x = \frac{k}{\Delta x^2}$$

$$\alpha_y = \frac{k}{\Delta y^2}$$

を定義した上で、

$$\Delta t = \frac{1}{2(\alpha_x + \alpha_y)}$$

とする。

# 熱拡散方程式の離散化

$$T_{i,j}(t + \Delta t) = \Delta t \{ \alpha_x (T_{i+1,j}(t) + T_{i-1,j}(t)) \\ + \alpha_y (T_{i,j+1}(t) + T_{i,j-1}(t)) \\ + s_{i,j} \}$$

現在の時刻の温度分布から、 $\Delta t$  だけ未来の温度分布を計算。これを繰り返して定常状態になるまで計算すればよい。

⇒ ヤコビ (Jacobi) 法

## 最も簡単な場合

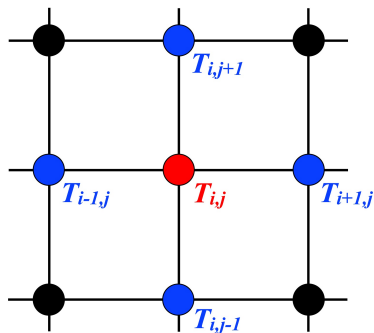
熱拡散係数  $k = 1$  で、熱源  $s(x, y) = 4$ 、そして  $\Delta x = \Delta y \equiv h$  の時  
 $\alpha_x = \alpha_y = 1/h^2$ ,  $\Delta t = h^2/4$

$$T_{i,j}(t + \Delta t) = \frac{T_{i+1,j}(t) + T_{i-1,j}(t) + T_{i,j+1}(t) + T_{i,j-1}(t)}{4} + h^2$$

と簡単な式になる。

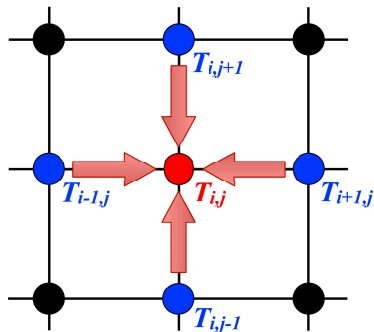
## ヤコビ法のイメージ

$$T_{i,j}(t + \Delta t) = \frac{T_{i+1,j}(t) + T_{i-1,j}(t) + T_{i,j+1}(t) + T_{i,j-1}(t)}{4} + h^2$$



# ヤコビ法のイメージ

$$T_{i,j}(t + \Delta t) = \frac{T_{i+1,j}(t) + T_{i-1,j}(t) + T_{i,j+1}(t) + T_{i,j-1}(t)}{4} + h^2$$



実装



## サンプルコード: heat2.f90

- ヤコビ法で正方形領域の定常状態の温度分布を求める
- 100 ステップに一度、中心の温度を書き出す

## コード解説: heat2.f90

```
do n=1, LOOP_MAX ! 時間更新
  do j=1, NGRID
    do i=1, NGRID
      un(i,j)=(u(i-1,j)+u(i+1,j))+... ! 作業配列
    end do
  end do
  u ...= un ... ! un を u にコピー
end do
```

# 演習

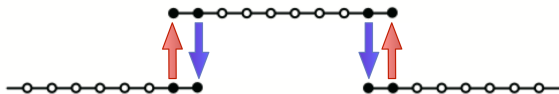
- heat2.f90 を理解しよう
- $\pi$ -computer 上でシリアルで実行してみよう:
- まずは `gfortran heat2.f90 && ./a.out > heat2.data`
- そして `gnuplot heat2.gp`

ちなみに中心点の温度の収束値の解析解は 0.294...

# 並列化

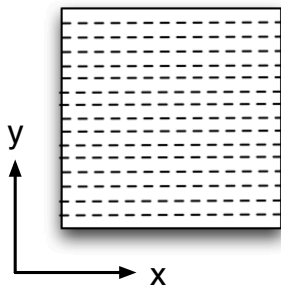
# 1次元並列化

$x$  軸方向または  $y$  軸方向の 1 次元空間を複数の領域に分割する (領域分割)。

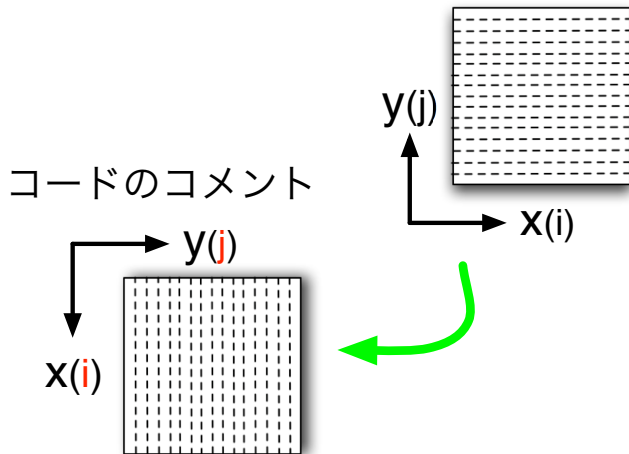


MPI の `sendrecv` で通信。  
以下では  $y$  軸方向に分割する。

# 1次元領域分割



# 1次元領域分割



# サンプルコード: heat3.f90

```
!  
!   When NGRID=11, nprocs=3  
!  
!   j=0  1  2  3  4  5  6  7  8  9 10 11 12  
! i=0 +---+---+---+---+---+---+---+---+---+---+---+---+  
!  1 |   |   |   |   |   |   |   |   |   |   |   |   |  
!  2 |   |   |   |   |   |   |   |   |   |   |   |   |  
!  3 |   |   |   |   |   |   |   |   |   |   |   |   |  
!  
!  .  
!  
!  .  
!  
!  .  
!  
!  .  
!  
! 11 |   |   |   |   |   |   |   |   |   |   |   |   |  
! 12 +---+---+---+---+---+---+---+---+---+---+---+---+  
!           | rank0 |   |           |   | rank2 |   |  
!       jstt-----jend |           | jstt-----jend  
!  
!                   |   rank1   |  
!  
!                   jstt-----jend
```



# 演習

- heat2.f90 を MPI 並列化した。heat3.f90
- heat3.f90 にはバグ（足りない部分）がある。それを修正せよ。【ヒント：MPI 通信部分】
- 実行方法は冒頭のコメント行にある通り：  
mpifrtpx heat3.f90 でコンパイルしてから、  
pjsub -o heat3.data heat3.sh

## ジョブスクリプト heat3.sh

```
#!/bin/bash
#PJM -N "heat3"
#PJM -L "rscgrp=small"
#PJM -L "node=4"
#PJM -L "elapse=02:00"
#PJM -j

drawLine()
{
    echo "#{1..50} | sed 's/[ 0-9]//g'"
}

drawLine
mpiexec ./a.out
drawLine
```

## heat3.data の確認

```
#####  
# myrank= 1  jstart & jend = 13 24  
# myrank= 2  jstart & jend = 25 36  
# myrank= 3  jstart & jend = 37 49  
# myrank= 0  jstart & jend = 1 12  
100 3.999301081967088E-02  
200 7.924913771637861E-02  
300 0.1152443143128371  
400 0.1463953538713233  
500 0.1725844298183796  
600 0.1943135320532408
```

## 時間発展のグラフ

(標準) 出力ファイル `heat3.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。)

`gnuplot` を立ち上げ、コマンドプロンプトに

```
gnuplot> plot 'heat3.data' w lp
```

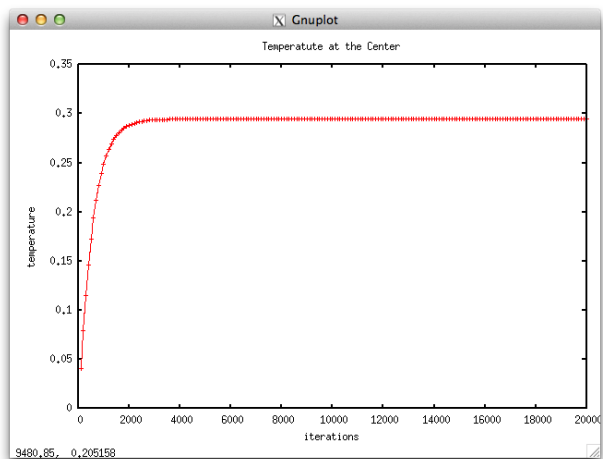
と入れよ。

あるいは

```
gnuplot heat3.gp
```

でもよい。

# 出力例



# 最終温度分布の可視化

heat3.f90 で計算された最終的な平衡温度分布を gnuplot で見てみよう。

正方形を真ん中で横に切る  $y=0.5$  の線上での温度の  $x$  分布をグラフにする。

グリッド番号  $i$  ではなく、 $x$  座標の値を書き出す。 $i$  番目の格子点の  $x$  座標は

$$x_i = h \times (i - n_{\text{mid}})$$

という関係にある。

## data ディレクトリ

これからの演習でデータファイルを多数生成するので、データファイル出力専用のディレクトリを用意しよう。

最初にコピーした `../data`  
というディレクトリを使う。

## heat3.f90 にデータ書き出し機能をつける

heat3\_print\_final\_x\_prof.f90

【演習】 このコードにも heat3.f90 と同じバグがあるのでまずはそれを修正せよ。

diff heat3.f90 heat3\_print\_final\_x\_prof.f90

```
! heat3_print_final_x_prof.f90
!   + open/close file 10
!   + print out cross section 1D data.
!   + integer jcut   (cross section for output)
!   + function this_process_has()
!   usage (on pi-computer)
!     1) mkdir ../data (unless there is already.)
!     2) mpifrtpx heat3_print_final_x_prof.f90
!     3) pjsub heat3.sh (share the jobscript with heat3.f90)
```



# 演習

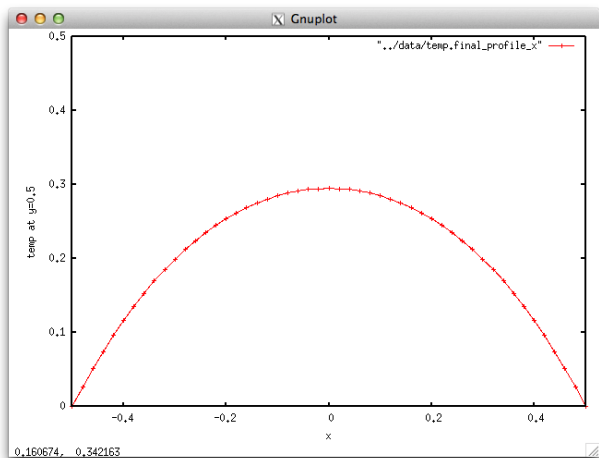
- (1) (バグを修正した) `heat3_print_final_x_prof.f90` をコンパイルし、実行せよ。
  - コンパイル `mpifrtpx heat3_print_final_x_prof.f90`
  - ジョブ投入 `pjsub heat3.sh` (同じジョブスクリプトを使う)
- (2) うまくいけば ディレクトリ `../data/` に `temp.final_profile_x` というファイルができています。
- (3) `more / less / head / tail` コマンドで確認。
- (4) `gnuplot` を立ち上げる
- (5) `plot "../data/temp.final_profile_x" w lp`

—  
最後の2ステップは `gnuplot heat3_print_final_x_prof.gp` でも OK.

## gnuplot スクリプト src/heat3\_print\_final\_x\_prof.gp

```
# heat3_print_final_x_prof.gp
#
# final temperature profile at y=0.5 as a function of x
#
set xrange [-0.5:0.5]
set yrange [0:0.5]
set xlabel "x"
set ylabel "temp at y=0.5"
plot "../data/temp.final_profile_x" w lp
pause -1
```

# 結果例



# アニメーション

アニメーションによって収束の様子を確認しよう。そのためのデータ（連番つきファイル群）を書き出すためのプログラム

`heat3_print_x_prof_for_animation.f90`

このプログラムをコンパイル+実行せよ。

ジョブスクリプト（これまで同様）`heat3.sh`

うまくいけば `data` ディレクトリに連番ファイルが出力されるはず。ls -l 等で確認せよ。

# アニメーション用スクリプトサンプル

```
#
# gnuplot script generated by heat3_animation_x_prof_gp_gg
#

set xlabel "x"           # x-axis
set ylabel "temperature" # y-axis
set xrange [-0.5:0.5]    # x-coordinate
set yrange [0.0:0.5]     # temperature min & max
plot "../data/temp.j=middle.0000" w lp
pause 5
plot "../data/temp.j=middle.0001" w lp
pause 1
plot "../data/temp.j=middle.0002" w lp
pause 1
.
.
```

## 【演習】 1次元グラフ アニメーション

- `heat3_print_x_prof_for_animation_plotscript_generator.f90` を確認せよ。
- 変数 `NGRID` と `counter_end` をチェックせよ。
- `gfortran`  
`heat3_print_x_prof_for_animation_plotscript_generator.f90`
- `./a.out > automatically_generated.gp`
- ファイル `automatically_generated.gp` の中身を確認する
- `gnuplot automatically_generated.gp` で実行

# 2次元可視化

(先週の復習)

diff heat3\_print\_x\_prof\_for\_animation.f90

heat4\_print\_final\_2d\_prof.f90

```
! heat4_print_final_2d_prof.f90
!   + subroutine print__profile_2d
!   c module constants --> module common
!   + type ranks_t :: p
!   + type span_t :: jj
!   - myrank, nprocs, left, right (combined into "p")
!   - jstart, jend (combined into "jj")
!   + function adjust_jstart_and_jend
!   + function set_prof_2d
```



## 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 sav
  type(span_t)    :: jj2                ! used fo
  character(len=4) :: serial_num         ! put on
  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
```

# 演習

heat4\_print\_final\_2d\_prof.f90 をコンパイル&実行してみよう。  
うまくいけば ../data/temp.2d.0000 ができているはず。

## 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
.		

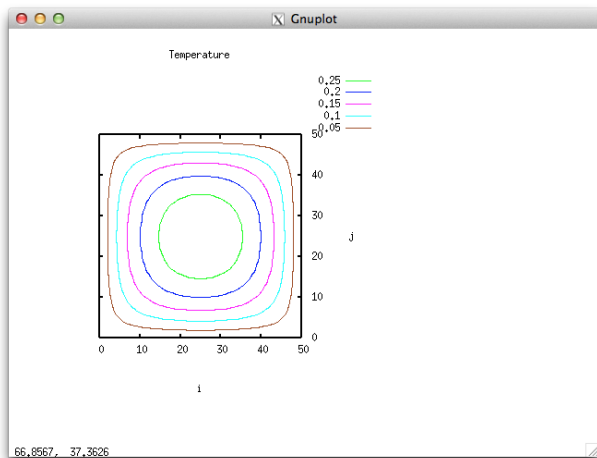
# gnuplot スクリプト生成

```
heat4_plot_contour_lines
#
# A sample gnuplot script: heat4_plot_contour_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"
```

## 【演習】 2次元等高線の表示

- data/temp.2d.0000 のファイルに記された温度の分布を gnuplot の等高線で可視化してみよう。
- ファイル名 : heat4\_plot\_contour\_lines.gp
- 実行方法 : gnuplot heat4\_plot\_contour\_lines.gp
- ファイル名やパラメータ等を自由に変更してその効果を試せ。

# 結果の例



## 色分布による可視化（静止画）

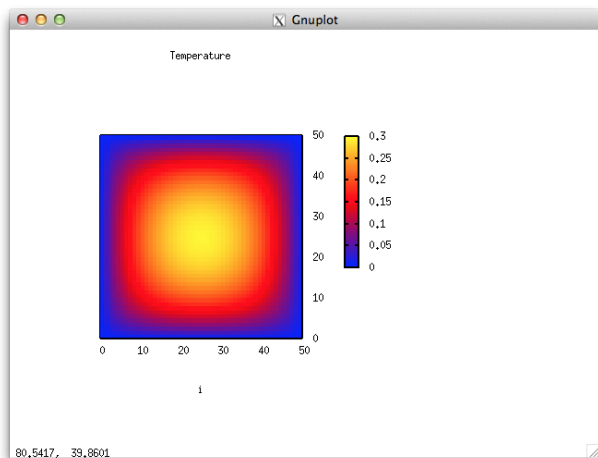
- 等高線を描く代わりに正方形領域内部各点の温度を色で表現することも可能である。
- 実際に描いてみよう。
- gnuplot のサンプルスクリプトは次のとおり。



## heat4\_plot\_contour\_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
```

# 結果の例



## gnuplot による鳥瞰図（静止画）

- 2次元温度分布  $T(x,y)$  を高さ ( $z$ ) で表す (height plot)
- 鳥瞰図 (bird's eye view)

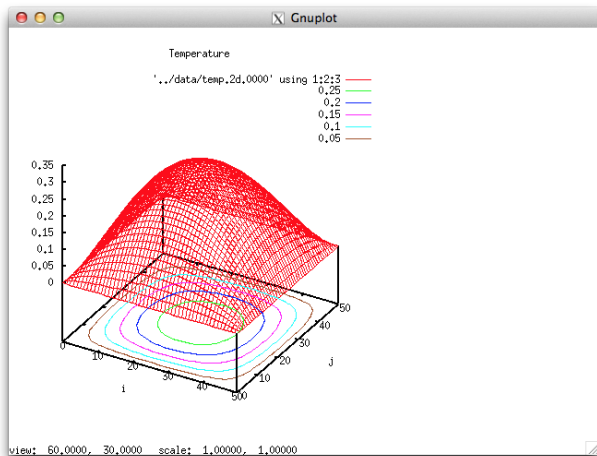
# gnuplot スクリプト

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
```

# 結果の例

マウスで回転できることに注意。



## gnuplot による鳥瞰図（回転のアニメーション）

gnuplot の view パラメータで視線の方向を変更したアニメーションを作ってみよう。

# gnuplot スクリプト生成プログラム

heat4\_plot\_rotating\_birdseyeview\_generator.f90

# レポート

heat4\_print\_final\_2d\_prof.f90 の計算を、もっと荒い格子（つまり低い解像度）で計算するプログラム heat4b\_print\_final\_2d\_prof.f90 を作り、その結果を確認せよ。

提出方法：修正したコードの差分を以下のコマンドで提出。

```
diff heat4_print_final_2d_prof.f90  
    heat4b_print_final_2d_prof.f90 | mail kage
```

✂切：7/26（水）23:59