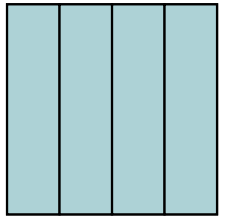
# 計算科学演習A2
# MPIを用いた並列計算（IV）

神戸大学大学院システム情報学研究科
横川 三津夫
yokokawa@port.kobe-u.ac.jp

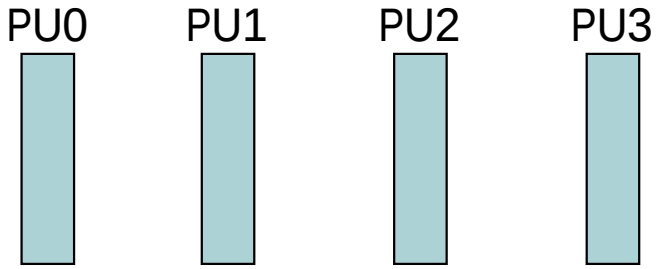
# MPI_sendrecvの応用

- 2次元配列がブロック列分割されている.

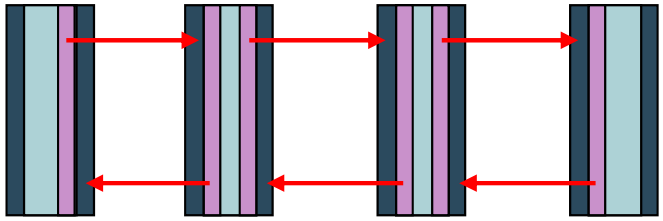
- このとき，自分の両端の１列の要素を，隣接するプロセスの持つ領域の外側に受信用の領域を確保し，その領域にそれぞれ転送するプログラムを作る.



2次元配列



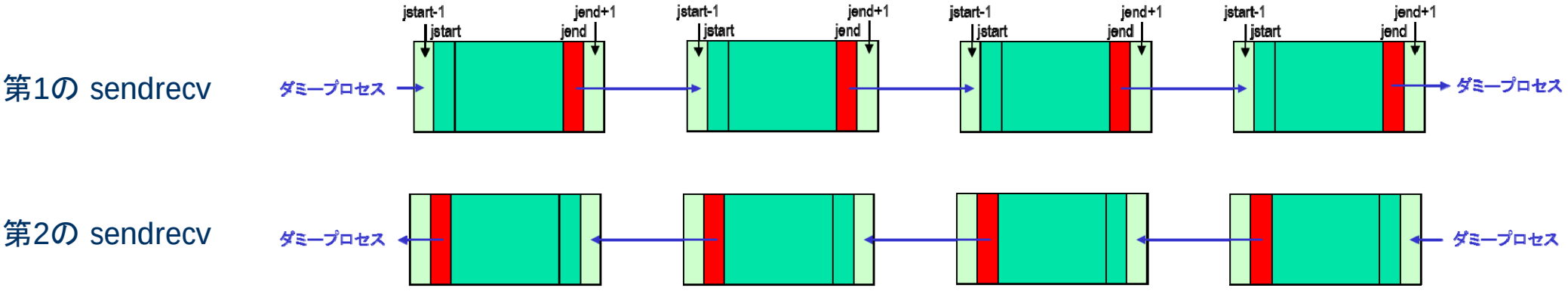

ブロック列分割



両隣のプロセスから1列を受信
（受信用の領域を確保しておく）

# MPI_sendrecvの応用（続き）

- 配列の確保
  - 自プロセスの担当範囲は jstart ～ jend 列
  - 受信領域を考慮し，jstart-1 ～ jend+1 列の領域を確保
- mpi_sendrecv による送受信
  - まず，右隣に jend 列を送り，左隣から jstart-1 列に受信
  - 次に，左隣に jstart 列を送り，右隣から jend+1 列に受信
  - 両端のプロセスは，ダミープロセス（MPI_PROC_NULL) と送受信するようにする.
    - MPI_sendrecvで同じように記述できる.



# mpi_sendrecv関数

```
mpi_sendrecv( sendbuf, sendcount, sendtype, dest,  sendtag, &
              recvbuf, recvcount, recvtype, source, recvtag, &
              comm, status, ierr )
```

- sendbuf:　送信するデータのための変数名（先頭アドレス）
- sendcount:　送信するデータの数（整数型）
- sendtype:　送信するデータの型
  - MPI_INTEGER, MPI_REAL8, MPI_CHARACTER など
- dest:　送信する相手のプロセス番号
- sendtag:　メッセージ識別番号．送られて来たデータを区別するための番号
- recvbuf:　受信するデータのための変数名（先頭アドレス）
- recvcount:　受信するデータの数（整数型）
- recvtype:　受信するデータの型
- source:　送信してくる相手のプロセス番号
- recvtag:　メッセージ識別番号．送られて来たデータを区別するための番号
- comm:　コミュニケータ（例えば，MPI_COMM_WORLD）
- status:　受信の状態を格納するサイズMPI_STATUS_SIZEの配列（整数型）
- ierr:　戻りコード（整数型）

# プログラム M-8（続く）

```
program sendrecv
  use mpi
  implicit none
  integer, parameter :: m=100
  integer            :: i,j,jstart,jend
  real(kind=8), dimension(:,:), allocatable :: u
  real(kind=8)       :: err

  integer            ::  nprocs,myrank,ierr,left,right
  integer, dimension(MPI_STATUS_SIZE) :: istat

  call mpi_init(ierr)
  call mpi_comm_size(MPI_COMM_WORLD,nprocs,ierr)
  call mpi_comm_rank(MPI_COMM_WORLD,myrank,ierr)

  jstart=m*myrank/nprocs+1
  jend=m*(myrank+1)/nprocs
  allocate(u(m,jstart-1:jend+1))


  do i=1, m
    do j=jstart, jend
      u(i,j)  =  dble(i+j)
    end do
  end do
```

Recvで必要な配列変数の宣言

各プロセスの担当する列の範囲を計算

jstart-1列～jend+1列の領域を確保

自分の担当する列に値を設定

# プログラム M-8（続き）

```
  left = myrank-1
  if (myrank==0) left=MPI_PROC_NULL
  right = myrank+1
  if (myrank==nprocs-1) right=MPI_PROC_NULL

! from the right process to the left
  call mpi_sendrecv( )

! from the left process to the right
  call mpi_sendrecv( )


  err = 0.0
  if ( myrank > 0 ) then
    do I = 1, m
      err = err + abs(u(i,jstart-1)-dble(i+jstart-1))
    end do
  endif
  if ( myrank < nprocs-1 ) then
    do I = 1, m
      err = err + abs(u(i,jend+1)-dble(i+jend+1))
    end do
  endif
  print *, 'myrank =', myrank, 'error =', err

  deallocate( u )
  call mpi_finalize(ierr)

end program sendrecv
```

左右のプロセスのプロセス番号を計算
（存在しない場合は MPI_PROC_NULL とする）

mpi_sendrecv による送受信

正しく受信できたことを確認

# 演習M4-1 ：M-8プログラムの実行

- sendrecv.f90 は未完成である.
- MPI_sendrecvを完成させ，コンパイルして，４，８プロセスで実行し，データの送受信が正しくできていることを確かめよ.

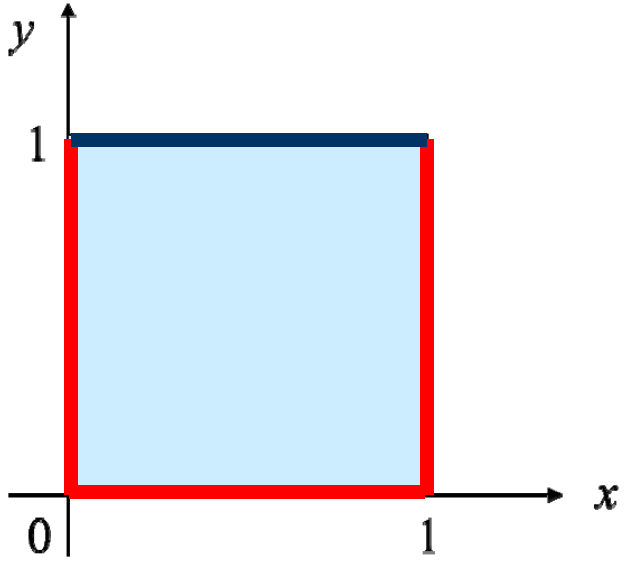
```
$  cp  /tmp/mpi4/sendrecv.f90  ./
```

  - すべてのプロセスが error = 0.0 を出力すればよい.

# 2次元定常熱伝導問題

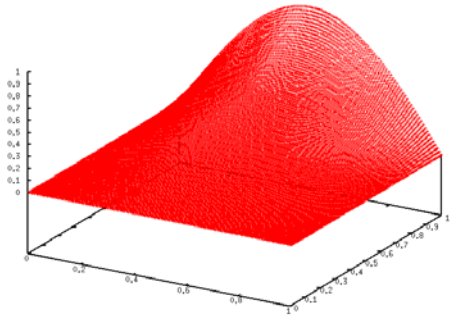
- 2次元正方形領域 [0,1]×[0,1] での熱伝導問題

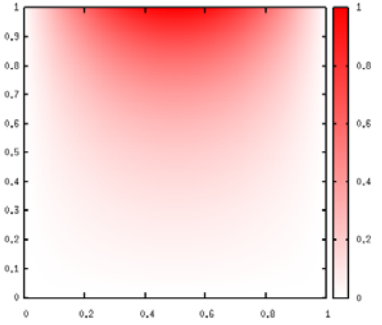
$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=0$$




- 境界条件

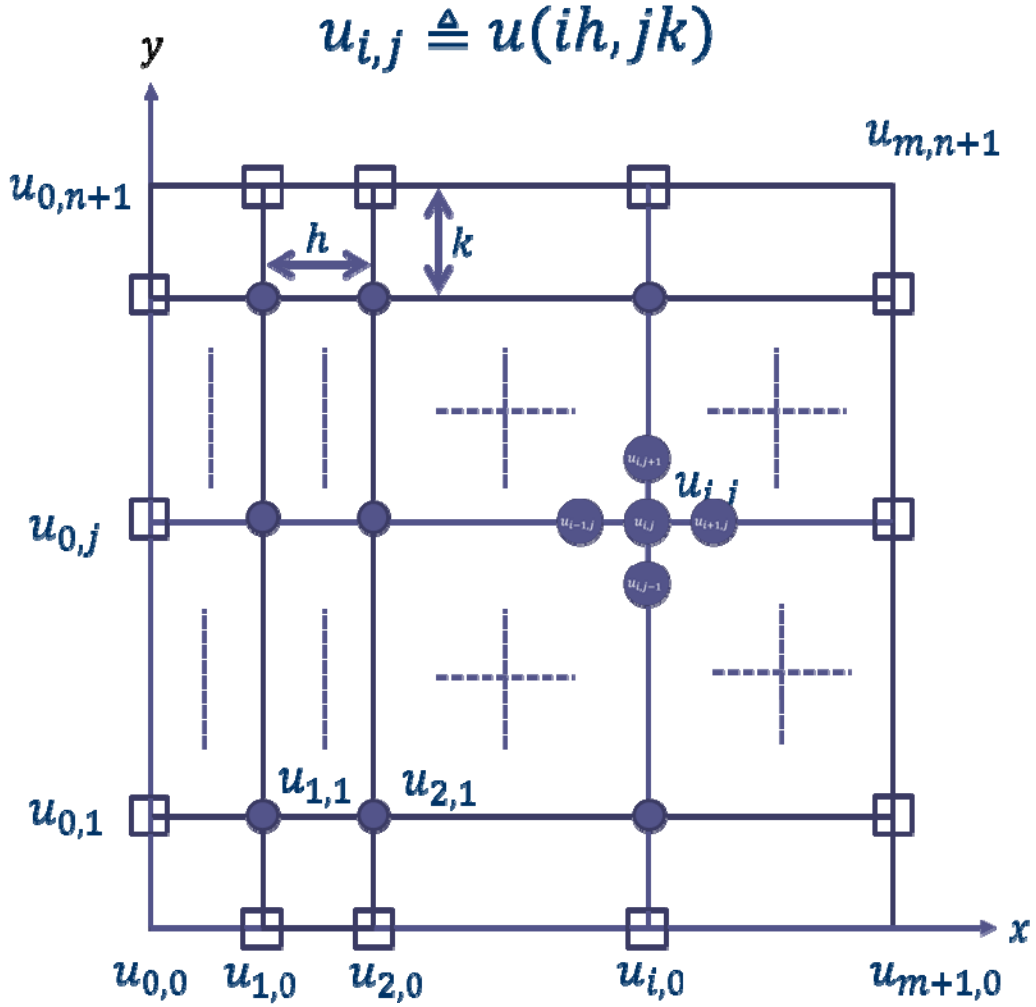
$$u(0,y)=0 \quad (0\le y\le 1)$$
$$u(1,y)=0 \quad (0\le y\le 1)$$
$$u(x,0)=0 \quad (0\le x\le 1)$$
$$u(x,1)=\sin(\pi x) \quad (0\le x\le 1)$$





# 2次元定常熱伝導問題の離散化

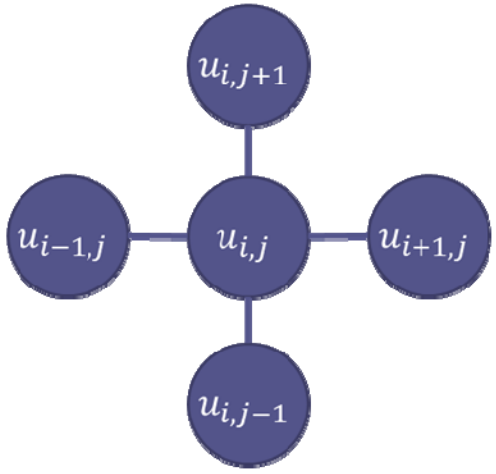
$x$方向を$(m+1)$等分：$h = 1/(m+1)$，　$y$方向を$(n+1)$等分：$k = 1/(n+1)$




□ 境界条件として既知の値

● 内部の格子点（未知数）

内部の格子点での関係式を作る．



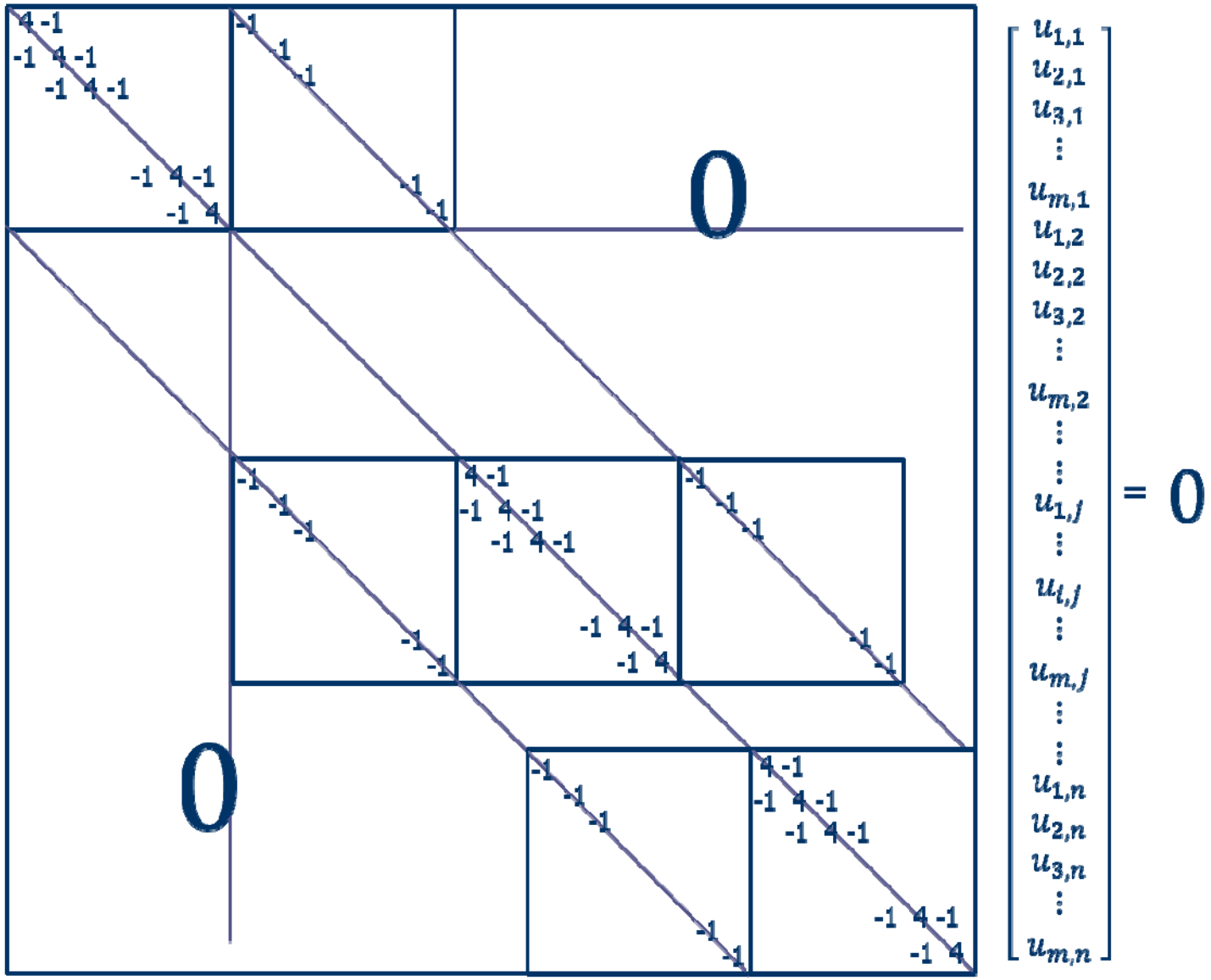
$u_{i,j}$を辞書式順序で並べたベクトルを$\boldsymbol{u}$とすると，点$(i,j)$に関する離散式は

$$\boldsymbol{u} = \begin{bmatrix} u_{1,1} \\ u_{2,1} \\ u_{3,1} \\ \vdots \\ u_{m,1} \\ u_{1,2} \\ u_{2,2} \\ u_{3,2} \\ \vdots \\ u_{m,2} \\ \vdots \\ \vdots \\ u_{1,j} \\ \vdots \\ u_{i,j} \\ \vdots \\ u_{m,j} \\ \vdots \\ \vdots \\ u_{1,n} \\ u_{2,n} \\ u_{3,n} \\ \vdots \\ u_{m,n} \end{bmatrix}$$

$$u_{i,j-1} + u_{i-1,j} - 4u_{i,j} + u_{i+1,j} + u_{i,j+1} = 0$$

$$[0 \quad \cdots \quad \underbrace{1 \quad \cdots \quad 1}_{m\text{離れている}} \quad -4 \quad \underbrace{1 \quad \cdots \quad 1}_{m\text{離れている}} \quad 0 \quad \cdots \quad 0] \begin{bmatrix} u_{1,1} \\ u_{2,1} \\ u_{3,1} \\ \vdots \\ u_{m,1} \\ u_{1,2} \\ u_{2,2} \\ u_{3,2} \\ \vdots \\ u_{m,2} \\ \vdots \\ \vdots \\ u_{1,j} \\ \vdots \\ u_{i,j} \\ \vdots \\ u_{m,j} \\ \vdots \\ \vdots \\ u_{1,n} \\ u_{2,n} \\ u_{3,n} \\ \vdots \\ u_{m,n} \end{bmatrix} = 0$$

# 行列を用いた表現


4 -1
-1 4 -1
-1 4 -1
-1 4 -1
-1 4
-1
-1
-1
-1
0
-1
-1
-1
4 -1
-1 4 -1
-1 4 -1
-1 4 -1
-1 4
-1
-1
-1
-1
0
-1
-1
-1
4 -1
-1 4 -1
-1 4 -1
-1 4 -1
-1 4
-1
-1
$u_{1,1}$ $u_{2,1}$ $u_{3,1}$ $u_{m,1}$ $u_{1,2}$ $u_{2,2}$ $u_{3,2}$ $u_{m,2}$ $u_{1,j}$ $u_{i,j}$ $u_{m,j}$ $u_{1,n}$ $u_{2,n}$ $u_{3,n}$ $u_{m,n}$
= 0

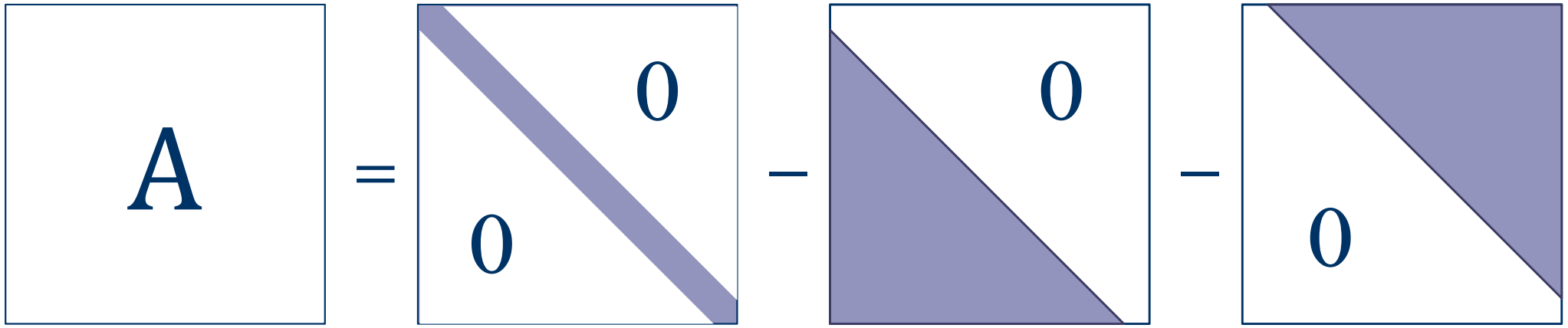
# 連立一次方程式の反復解法

- 反復解法
  - $x^{(0)}$を真の解$x^*$の近似値とし，反復ベクトルの系列 $x^{(0)} \rightarrow x^{(1)} \rightarrow x^{(2)} \rightarrow \cdots \rightarrow x^{(m)}$により，真の解に近づける方法
  - 反復ベクトルの系列の作り方によりいろいろな解法がある．

    - 定常反復法
      - ヤコビ法（Jacobi法）
      - ガウス・ザイデル法（Gauss-Seidel法）
      - 逐次過大緩和法（Successive Over-Relaxation：SOR法）
      - 交互方向法（Alternating-Direction Implicit Iterative Method: ADI法）など

    - 非定常反復法
      - 共役勾配法（Conjugate Gradient Method，CG法）
      - 双共役勾配法（Bi-CG法）
      - 一般化最小残差法（GMRES法）など

# 行列分離




行列分離： $\mathrm{A} = \mathrm{D} - \mathrm{E} - \mathrm{F}$

$\mathrm{D} = \mathrm{diag}(a_{11}, a_{22}, \cdots, a_{nn})$：対角行列

E：狭義下三角行列

F：狭義上三角行列

$$\mathrm{A}\boldsymbol{x} = \boldsymbol{b} \quad \Rightarrow \quad (\mathrm{D} - \mathrm{E} - \mathrm{F})\boldsymbol{x} = \boldsymbol{b}$$

# ヤコビ法（その1）

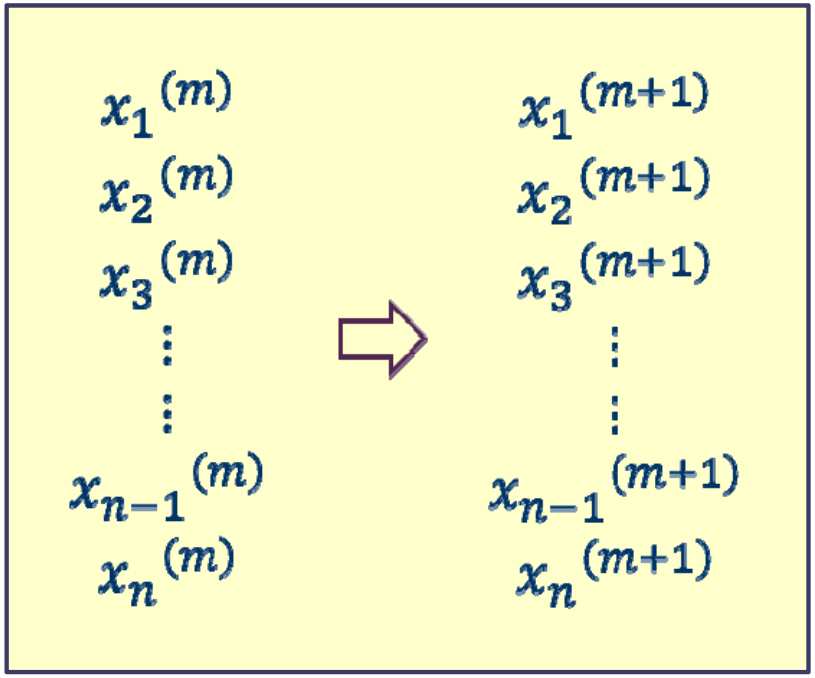
$$(\mathrm{D}-\mathrm{E}-\mathrm{F})\boldsymbol{x}=\boldsymbol{b} \quad \Rightarrow \quad \mathrm{D}\boldsymbol{x}=(\mathrm{E}+\mathrm{F})\boldsymbol{x}+\boldsymbol{b}$$

$$\boldsymbol{x}^{(m+1)} = \underbrace{\mathrm{D}^{-1}(\mathrm{E}+\mathrm{F})}_{\text{ヤコビ行列}}\boldsymbol{x}^{(m)} + \mathrm{D}^{-1}\boldsymbol{b} \quad (m \geq 0)$$

で，反復ベクトル系列を生成する.

$$x_i^{(m+1)} = \left( -\sum_{j=1, j\neq i}^{n} a_{ij} x_j^{(m)} + b_i \right) / a_{ii}$$

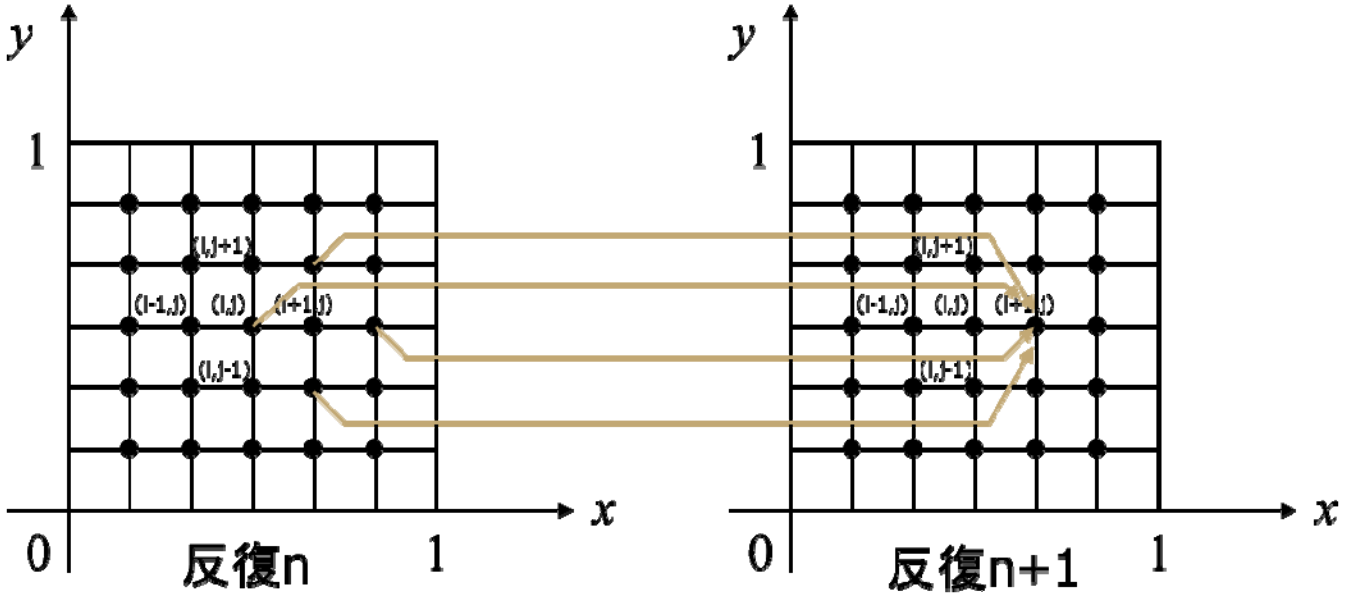
$$(1 \leq i \leq n, m \geq 0)$$



# 熱伝導問題へのヤコビ法の適用

- 反復ベクトル生成のアルゴリズム

```
do j=1, m
    do i=1, m
        u_ij^(n+1) = (u_i-1,j^(n) + u_i+1,j^(n) + u_i,j-1^(n) + u_i,j+1^(n) ) / 4
    end do
end do
```



# 演習M4-2 ：プログラムM-9の実行

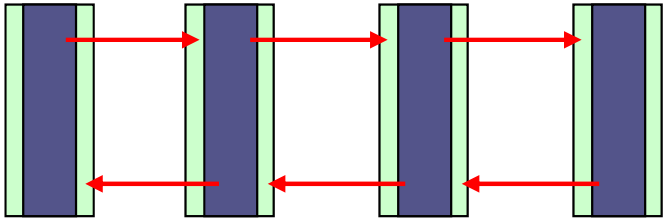
- 2次元定常熱伝導問題の逐次プログラムM-9をコンパイルして実行せよ.

```
$ cp  /tmp/mpi4/heat2d.f90  ./
$ f95 heat2d.f90
$ ./a.out
```

  - $\|\boldsymbol{b} - A\boldsymbol{x}\|/\|\boldsymbol{b}\| < 10^{-5}$ となっていることを確認せよ.

# heat2d.f90 の並列化

- 考え方
  - 2次元配列 u，uo をブロック列分割
    - 配列 u は，jstart ～ jend 列の領域を確保
    - 配列 uo は，受信領域を考慮し，jstart-1 ～ jend+1 列の領域を確保
  - u の計算前に，uをuoにコピーし，左のプロセスから uo の jstart-1 列，右のプロセスから uo の jend+1 列を送ってもらう.
    - sendrecv.f90 と同様にして，mpi_sendrecv を用いて送受信



両隣のプロセスから1列を受信
（受信用の領域を確保しておく）

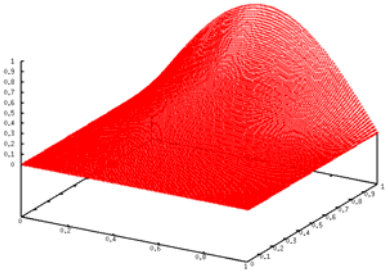
  - u の jstart ～ jend 列の計算を行う.
  - プロセス0で収束の判定を行うようにする.
    - ローカルで残差の部分和を計算し，プロセス0で総和を取る.

# 演習M4-3 heat2d.f90の並列化

- heat2d.f90をMPIを用いて並列化せよ.
- nを64に変更し，2, 4, 8 プロセスで実行せよ．プログラムが正しく動作することを確認せよ.

# 課題：ヤコビ法の並列化と性能評価

① （必須）計算結果をgnuplotで図示せよ.
- 計算結果の出力は，プロセス0で行えば良い.



② （必須）MPIによる並列プログラムにおいてn=128とし，プロセス数1, 2, 4, 8 として，反復開始から終了までの計算時間を計測し，速度向上率を求めよ．また，それをグラフにせよ.

③ （任意課題）MPIに加え，OpenMPのディレクティブを挿入し，MPI+OpenMPのハイブリッド並列の実行性能を評価せよ．スレッド並列は1，2，4，8，16と変化させるものとする.
- ハイブリッド並列では，u(i,j) などの変数について， jについてMPI並列（プロセス並列），iについてOpenMPによるスレッド並列とすれば良い.

# 課題の提出方法と提出期限

- 課題をレポートとしてまとめ，pdf形式にして，メールで yokokawa@port.kobe-u.ac.jp まで送ること.
  - 課題①，②は必須（プログラムもレポートに含める）
  - 課題③は任意
  - メールのSubjectは，”4:アカウント名” とすること.

- 期限：7月19日（水） 午後5時