課題S3解説

2007年6月20日

中島 研吾

並列計算プログラミング（616-2057）・先端計算機演習I（616-4009）

課題S3：実施内容

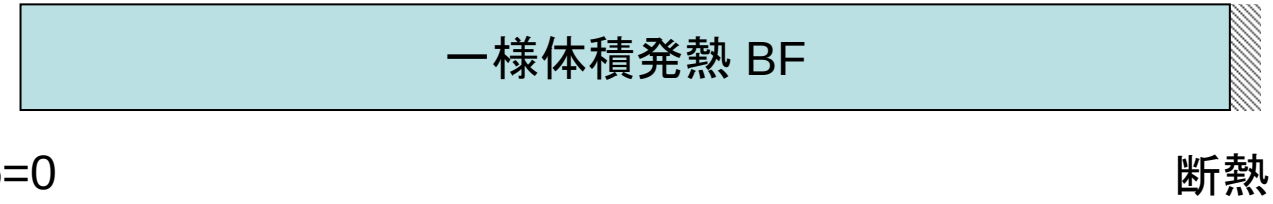
- 「<$S3>/heat_cg.f」，「<$S3>/heat_cg.c」を参考にして，一次元熱伝導方程式をCG法により解くプログラムを並列化せよ。
  - 一般化された通信テーブルを導入する
- 実施課題
  - N=100およびN=103の場合について，1，2，4，8CPUを使用して計算してみよ。
    - 実は，N=100程度では並列化の効果は余り・・・全く得られない
  - プロセッサ数を変えた場合，反復回数はどうなるか？
    - その理由についても検討せよ
  - Nを大きくして（例えば10^4，10^5），反復回数を100程度に固定して，プロセッサ数を変更した場合について計算を実施してみよ。

一次元熱伝導方程式(1/4)

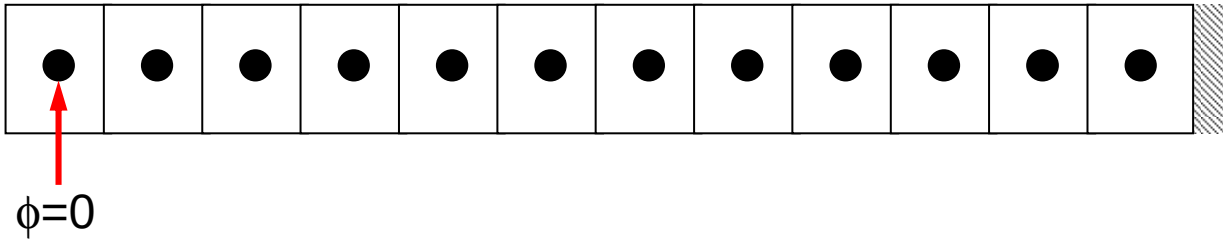
支配方程式:熱伝導率＝1(一様)

$$\frac{d^2\phi}{dx^2} + BF = 0, \quad \phi = 0 @ x = 0, \frac{d\phi}{dx} = 0 @ x = x_{\max}$$

$$\phi = -\frac{1}{2} BF\, x^2 + BF\, x_{\max}\, x$$



以下のような離散化(要素中心で従属変数を定義)をしているので注意が必要



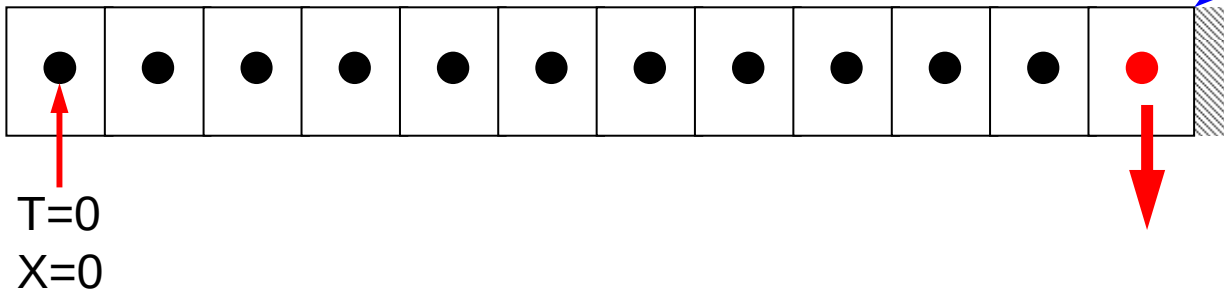
断熱となっているのはこの面,しかし温度は計算されない

一次元熱伝導方程式(2/4)

解析解

$$\phi = -\frac{1}{2} BF\, x^2 + BF\, x_{\max}\, x$$

断熱となって
いるのはこの面,
しかし温度は計算
されない(X=X_{max})。



Δx=1.d0, メッシュ数=50, とすると, X_{max}=49.5,
●の点のX座標は49.0となる。BF=1.0d0とすると●での温度は:

$$\phi = -\frac{1}{2} 49^2 + 49.5 \times 49 = -1200.5 + 9850.5 = 1225$$

一次元熱伝導方程式(3/4)

連立一次方程式

- 差分法による離散化

$$\left(\frac{d^2\phi}{dx^2}\right)_i \approx \frac{\left(\frac{d\phi}{dx}\right)_{i+1/2} - \left(\frac{d\phi}{dx}\right)_{i-1/2}}{\Delta x} = \frac{\frac{\phi_{i+1}-\phi_i}{\Delta x} - \frac{\phi_i-\phi_{i-1}}{\Delta x}}{\Delta x} = \frac{\phi_{i+1}-2\phi_i+\phi_{i-1}}{\Delta x^2}$$

- 各要素における線形方程式は以下のような形になる
  - これを共役勾配法(Conjugate Gradient法)で解く

$$\frac{d^2\phi}{dx^2} + BF = 0 \quad \rightarrow$$

$$\frac{\phi_{i+1}-2\phi_i+\phi_{i-1}}{\Delta x^2} + BF(i) = 0 \quad (1 \le i \le N)$$

$$\frac{1}{\Delta x^2}\phi_{i+1} - \frac{2}{\Delta x^2}\phi_i + \frac{1}{\Delta x^2}\phi_{i-1} + BF(i) = 0 \quad (1 \le i \le N)$$

$$A_L(i)\times\phi_{i-1} + A_D(i)\times\phi_i + A_R(i)\times\phi_{i+1} = BF(i) \quad (1 \le i \le N)$$

$$A_L(i) = \frac{1}{\Delta x^2},\ A_D(i) = -\frac{2}{\Delta x^2},\ A_R(i) = \frac{1}{\Delta x^2}$$

一次元熱伝導方程式(4/4)

係数行列の格納形式

各要素における線形方程式

$$A_L(i)\times\phi_{i-1}+A_D(i)\times\phi_i+A_R(i)\times\phi_{i+1}=BF(i)\quad(1\le i\le N)$$

$$A_L(i)=\frac{1}{\Delta x^2},\ A_D(i)=-\frac{2}{\Delta x^2},\ A_R(i)=\frac{1}{\Delta x^2}$$

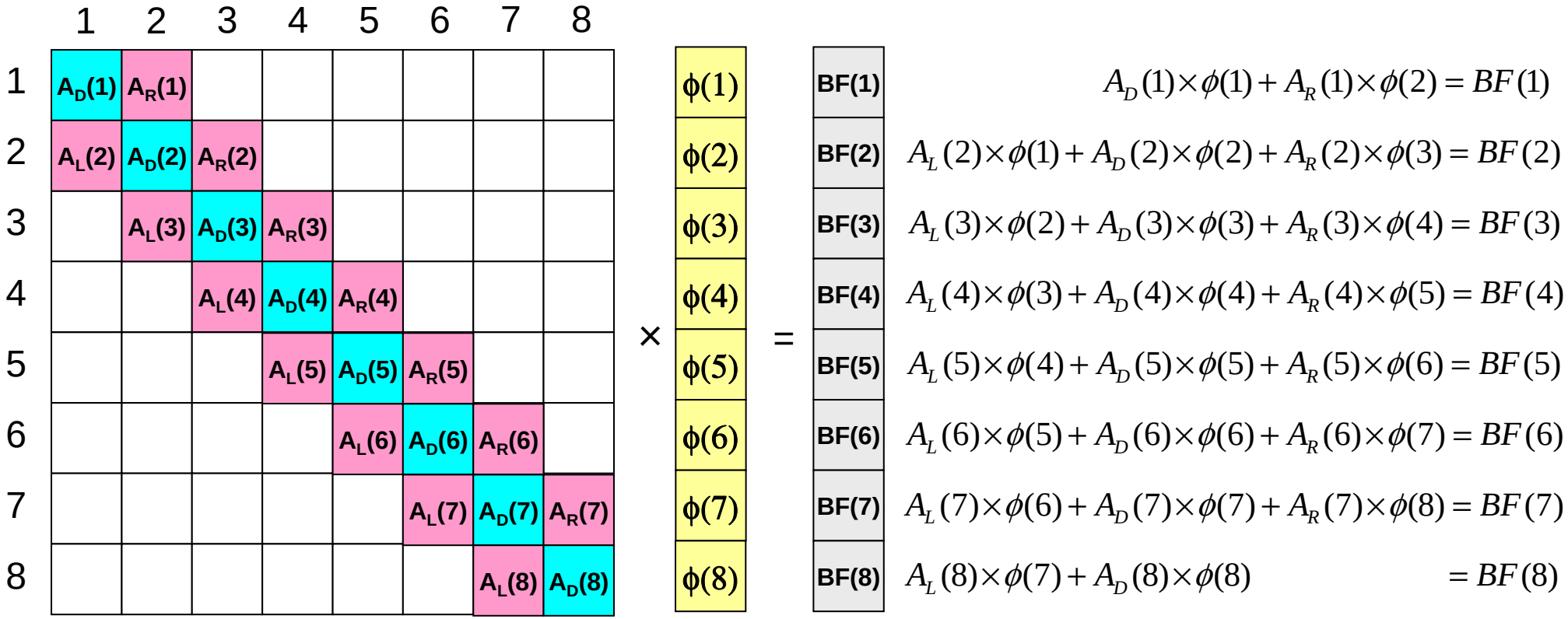
より一般的な形で格納すると・・・

$$DIAG(i)\times PHI(i)+\sum_{k=INDEX(i-1)+1}^{INDEX(i)}\left[AMAT(k)\times PHI(ITEM(k))\right]=RHS(i),\quad(i=1,\ldots,N)$$

対角成分　　非対角成分

N=8の場合：連立一次方程式

自分とその周囲のみに非ゼロ成分：疎行列



$$DIAG(i)\times PHI(i)+\sum_{k=INDEX(i-1)+1}^{INDEX(i)}\left[AMAT(k)\times PHI(ITEM(k))\right]=RHS(i),\quad (i=1,\ldots,N)$$

対角成分　　非対角成分

解いている方程式自体は課題S2と同じ ！

係数行列の格納形式

非ゼロ成分のみを格納，疎行列向け方法

$$DIAG(i)\times PHI(i)+$$

$$\sum_{k=INDEX(i-1)+1}^{INDEX(i)}\left[AMAT(k)\times PHI(ITEM(k))\right]=$$

$$RHS(i),\quad (i=1,\ldots,N)$$

DIAG(i)	対角成分（実数，i=1,N）
INDEX(i)	非対角成分数に関する一次元配列（整数，i=0,N）
ITEM(k)	非対角成分の要素番号（整数，k=1, INDEX(N)）
AMAT(k)	非対角成分（実数，k=1, INDEX(N)）

	1	2	3	4	5	6	7	8
1	$A_D(1)$	$A_R(1)$						
2	$A_L(2)$	$A_D(2)$	$A_R(2)$					
3		$A_L(3)$	$A_D(3)$	$A_R(3)$				
4			$A_L(4)$	$A_D(4)$	$A_R(4)$			
5				$A_L(5)$	$A_D(5)$	$A_R(5)$		
6					$A_L(6)$	$A_D(6)$	$A_R(6)$	
7						$A_L(7)$	$A_D(7)$	$A_R(7)$
8							$A_L(8)$	$A_D(8)$

行列ベクトル積への適用

非ゼロ成分のみを格納，疎行列向け方法

DIAG(i)	対角成分(実数，i=1,N)
INDEX(i)	非対角成分数に関する一次元配列 (整数，i=0,N)
ITEM(k)	非対角成分の要素番号 (整数，k=1， INDEX(N))
AMAT(k)	非対角成分 (実数，k=1， INDEX(N))

$$DIAG(i)\times PHI(i)+\sum_{k=INDEX(i-1)+1}^{INDEX(i)}\left[AMAT(k)\times PHI(ITEM(k))\right]=RHS(i),\quad (i=1,\ldots,N)$$

```
{Y}= [A]{X}

do i= 1, N
  Y(i)= D(i)*X(i)
  do k= INDEX(i-1)+1, INDEX(i)
    Y(i)= Y(i) + AMAT(k)*X(ITEM(k))
  enddo
enddo
```

	1	2	3	4	5	6	7	8
1	$A_D(1)$	$A_R(1)$						
2	$A_L(2)$	$A_D(2)$	$A_R(2)$					
3		$A_L(3)$	$A_D(3)$	$A_R(3)$				
4			$A_L(4)$	$A_D(4)$	$A_R(4)$			
5				$A_L(5)$	$A_D(5)$	$A_R(5)$		
6					$A_L(6)$	$A_D(6)$	$A_R(6)$	
7						$A_L(7)$	$A_D(7)$	$A_R(7)$
8							$A_L(8)$	$A_D(8)$

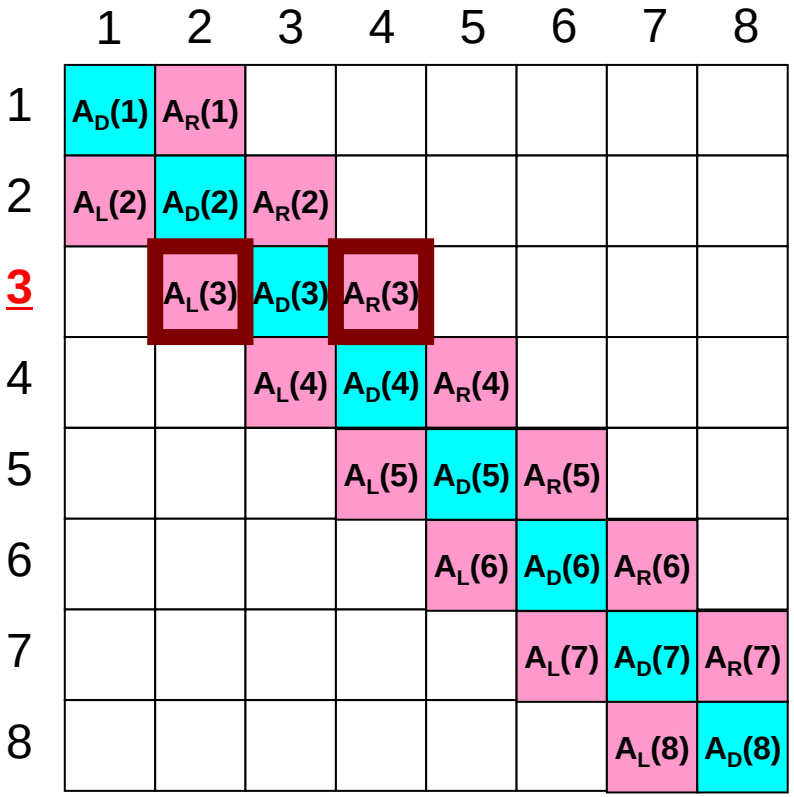
係数行列の格納形式(3/8)
INDEX, AMATの関係

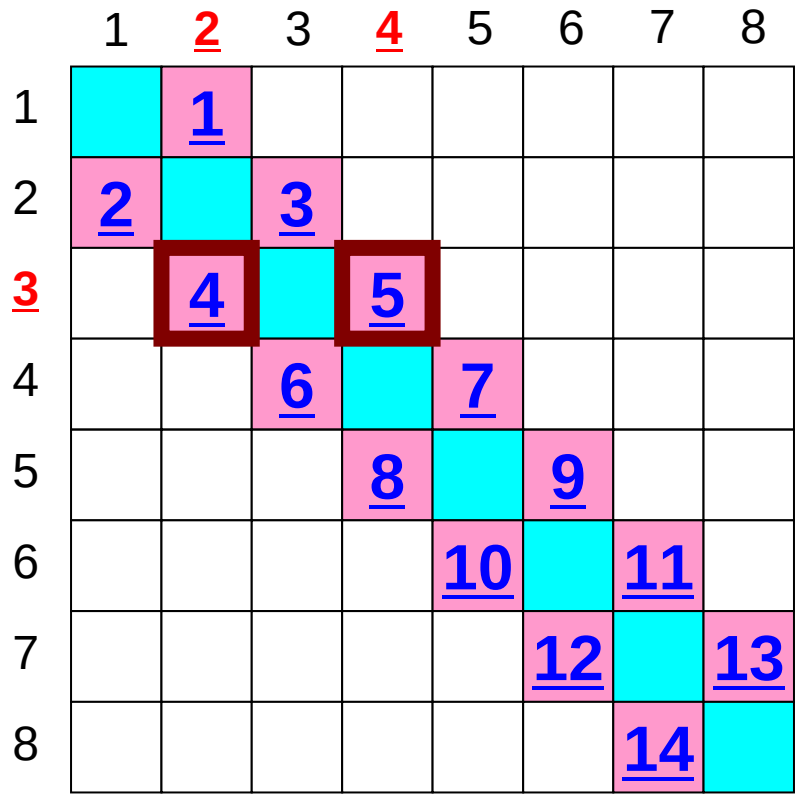
	1	2	3	4	5	6	7	8
1	$A_D(1)$	$A_R(1)$						
2	$A_L(2)$	$A_D(2)$	$A_R(2)$					
3		$A_L(3)$	$A_D(3)$	$A_R(3)$				
4			$A_L(4)$	$A_D(4)$	$A_R(4)$			
5				$A_L(5)$	$A_D(5)$	$A_R(5)$		
6					$A_L(6)$	$A_D(6)$	$A_R(6)$	
7						$A_L(7)$	$A_D(7)$	$A_R(7)$
8							$A_L(8)$	$A_D(8)$

	1	2	3	4	5	6	7	8
1		1						
2	2		3					
3		4		5				
4			6		7			
5				8		9		
6					10		11	
7						12		13
8							14	

```
INDEX(2)=3, INDEX(3)=5
ITEM(4)=2, AMAT(4)= A_L(3)
ITEM(5)=4, AMAT(5)= A_R(3)
```

係数行列の格納形式(3/8)
INDEX, AMATの関係






```
INDEX(2)=3, INDEX(3)=5
ITEM(4)=2, AMAT(4)= A_L(3)
ITEM(5)=4, AMAT(5)= A_R(3)
```

課題S3：プログラム実行法

- 実行法

```
$ cd <$S3>
$ pgf90 -O3 heat_cg.f
$ ./a.out
```

```
$ cd <$S3>
$ pgcc -O3 heat_cg.c
$ ./a.out
```

input.dat

```
100              N              メッシュ数
1.d0 1.d0        dx, BF         メッシュ幅，体積発熱量
5000             ITERmax        最大反復回数
1.d-7  1.95d0    EPS, OMEGA     打切誤差，SORのω
```

CG法による一次元熱伝導方程式計算プログラム

- 前処理付き共役勾配法
  - Preconditioned Conjugate Gradient Method
  - 対称正定行列用
- 対角スケーリング（Diagonal Scaling）
  - [A]の対角成分のみからなる行列を前処理行列[M]とする。
  - 簡単（な問題にしか適用できない）。
  - 点ヤコビ（Point Jacobi）前処理とも呼ばれる。

```
Compute r(0)= b-[A]x(0)
for i= 1, 2, …
    solve [M]z(i-1)= r(i-1)
    ρi-1= r(i-1) z(i-1)
    if i=1
      p(1)= z(0)
     else
      βi-1= ρi-1/ρi-2
      p(i)= z(i-1) + βi-1 p(i)
    endif
    q(i)= [A]p(i)
    αi  = ρi-1/p(i)q(i)
    x(i)= x(i-1) + αip(i)
    r(i)= r(i-1) - αiq(i)
    check convergence |r|
end
```

対角スケーリング，点ヤコビ前処理

- 前処理行列として，もとの行列の対角成分のみを取り出した行列を前処理行列 [M] とする。
  - 対角スケーリング，点ヤコビ（point-Jacobi）前処理

$$[M]=\begin{bmatrix} D_1 & 0 & \cdots & 0 & 0 \\ 0 & D_2 & & 0 & 0 \\ \cdots & & \cdots & & \cdots \\ 0 & 0 & & D_{N-1} & 0 \\ 0 & 0 & \cdots & 0 & D_N \end{bmatrix}$$

- **solve [M]z$^{(i-1)}$= r$^{(i-1)}$**という場合に逆行列を簡単に求めることができる。

CG法による一次元熱伝導方程式 プログラム概要(1/7)

```
!C
!C   1D Poisson Equation Solver by
!C   CG (Conjugate Gradient) Method
!C
!C   d/dx(dPHI/dx) + BF = 0
!C   PHI=0@x=0
!C
      program CG_poi
      implicit REAL*8 (A-H,O-Z)

      integer :: N, ITERmax
      integer :: R, Z, P, Q, DD

      real(kind=8) :: dx, OMEGA, RESID, dPHI, dPHImax, BF, EPS
      real(kind=8), dimension(:), allocatable :: PHI, RHS
      real(kind=8), dimension(:  ), allocatable :: DIAG, AMAT
      real(kind=8), dimension(:,:), allocatable :: W

      integer, dimension(:  ), allocatable :: INDEX, ITEM

!C
!C-- INIT.
      open  (11, file='input.dat', status='unknown')
       read (11,*) N
       read (11,*) dX, BF
       read (11,*) ITERmax
       read (11,*) EPS
      close (11)
```

CG法による一次元熱伝導方程式 プログラム概要(2/7)

```
allocate (PHI(N), DIAG(N), AMAT(2*N-2), RHS(N))
allocate (INDEX(0:N), ITEM(2*N-2), W(N,4))
 PHI= 0.d0
AMAT=  1.d0/dX
DIAG= -2.d0/dX
 RHS=  -BF * dX
```

$$\frac{d^2\phi}{dx^2} + BF = 0, \quad \phi = 0 @ x = 0, \frac{d\phi}{dx} = 0 @ x = x_{\max}$$

$$\left(\frac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{\Delta x^2} \right) \times V + BF \times V = 0$$

非対角成分の数は「2」
ただし，i=1，i=Nのときは 「1」

したがって，非対角成分の総数は「2*N-2」

CG法による一次元熱伝導方程式 プログラム概要(2/7)

```
allocate (PHI(N), DIAG(N), AMAT(2*N-2), RHS(N))
allocate (INDEX(0:N), ITEM(2*N-2), W(N,4))
 PHI= 0.d0
AMAT=  1.d0/dX
DIAG= -2.d0/dX
 RHS=  -BF * dX
```

$$\left(\frac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{\Delta x^2} \right) \times V + BF \times V = 0$$

$$\left(\frac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{\Delta x^2} \right) \times \Delta x + BF \times \Delta x = 0$$

非対角成分の数は「2」
ただし，i=1，i=Nのときは 「1」

したがって，非対角成分の総数は「2*N-2」

CG法による一次元熱伝導方程式 プログラム概要(2/7)

```
allocate (PHI(N), DIAG(N), AMAT(2*N-2), RHS(N))
allocate (INDEX(0:N), ITEM(2*N-2), W(N,4))
 PHI= 0.d0
AMAT=  1.d0/dX
DIAG= -2.d0/dX
 RHS=  -BF * dX
```

	1	2	3	4	5	6	7	8
1	$A_D(1)$	$A_R(1)$						
2	$A_L(2)$	$A_D(2)$	$A_R(2)$					
3		$A_L(3)$	$A_D(3)$	$A_R(3)$				
4			$A_L(4)$	$A_D(4)$	$A_R(4)$			
5				$A_L(5)$	$A_D(5)$	$A_R(5)$		
6					$A_L(6)$	$A_D(6)$	$A_R(6)$	
7						$A_L(7)$	$A_D(7)$	$A_R(7)$
8							$A_L(8)$	$A_D(8)$

$$\left(\frac{\phi_{i+1}-2\phi_i+\phi_{i-1}}{\Delta x^2}\right)\times\Delta x+BF\times\Delta x=0$$

$$\frac{\phi_{i+1}-2\phi_i+\phi_{i-1}}{\Delta x}=\left(\frac{1}{\Delta x}\right)\phi_{i-1}-\left(\frac{2}{\Delta x}\right)\phi_i+\left(\frac{1}{\Delta x}\right)\phi_{i+1}=-BF\times\Delta x$$

$A_L(i)$ $A_D(i)$ $A_R(i)$ RHS(i)

非対角成分の数は「2」
ただし，i=1，i=Nのときは 「1」

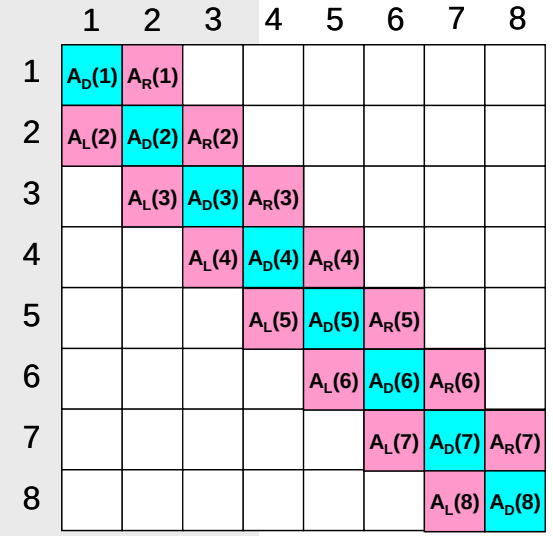
したがって，非対角成分の総数は「2*N-2」

CG法による一次元熱伝導方程式 プログラム概要(3/7)

```
      INDEX= 2
!C
!C-- CONNECTIVITY
      INDEX(0)= 0
      INDEX(1)= 1
      INDEX(N)= 1

      do i= 1, N
        INDEX(i)= INDEX(i) + INDEX(i-1)
      enddo

      do i= 1, N
        jS= INDEX(i-1)
        if (i.eq.1) then
          ITEM(jS+1)= i+1
          AMAT(jS+1)= 0.d0
          DIAG(i   )= 1.d0
           RHS(i   )= 0.d0
         else if
     &     (i.eq.N) then
          ITEM(jS+1)= i-1
          DIAG(i   )= -1.d0/dX
         else
          ITEM(jS+1)= i-1
          ITEM(jS+2)= i+1
          if (i-1.eq.1) then
            AMAT(jS+1)= 0.d0
          endif
        endif
      enddo
```

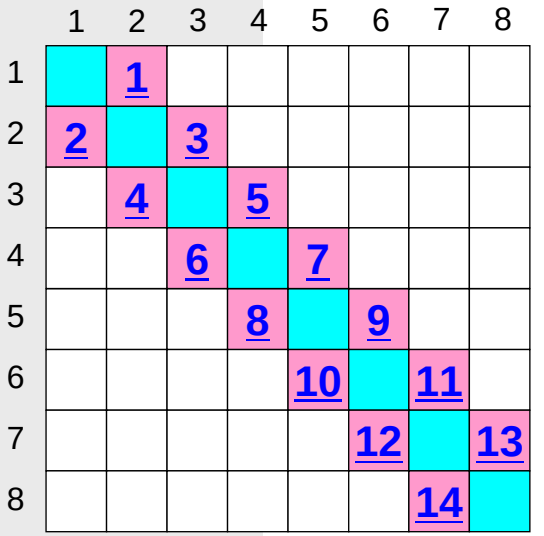
非対角成分の数は「2」
ただし，i=1，i=Nのときは 「1」

このルールに従って「INDEX」生成



CG法による一次元熱伝導方程式 プログラム概要(3/7)

```
      INDEX= 2
!C
!C-- CONNECTIVITY
      INDEX(0)= 0
      INDEX(1)= 1
      INDEX(N)= 1

      do i= 1, N
        INDEX(i)= INDEX(i) + INDEX(i-1)
      enddo

      do i= 1, N
        jS= INDEX(i-1)
        if (i.eq.1) then
          ITEM(jS+1)= i+1
          AMAT(jS+1)= 0.d0
          DIAG(i   )= 1.d0
           RHS(i   )= 0.d0
         else if
     &     (i.eq.N) then
          ITEM(jS+1)= i-1
          DIAG(i   )= -1.d0/dX
         else
          ITEM(jS+1)= i-1
          ITEM(jS+2)= i+1
          if (i-1.eq.1) then
            AMAT(jS+1)= 0.d0
          endif
        endif
      enddo
```




```
INDEX(0)=0
INDEX(1)=1
                ITEM(1)=2, AMAT(1)= AR1
INDEX(2)=3
                ITEM(2)=1, AMAT(2)= AL2
                ITEM(3)=3, AMAT(3)= AR3
INDEX(3)=5
                ITEM(4)=2, AMAT(4)= AL3
                ITEM(5)=4, AMAT(5)= AR3
…
INDEX(N-1)= 2*N-3
                ITEM(2*N-4)= N-2, AMAT(2*N-4)= AL(N-1)
                ITEM(2*N-3)= N  , AMAT(2*N-3)= AR(N-1)
INDEX(N)  = 2*N-2
                ITEM(2*N-2)= N-1, AMAT(N*N-2)= AL(N)
```

CG法による一次元熱伝導方程式 プログラム概要(3/7)

```
      INDEX= 2
!C
!C-- CONNECTIVITY
       INDEX(0)= 0
       INDEX(1)= 1
       INDEX(N)= 1

       do i= 1, N
         INDEX(i)= INDEX(i) + INDEX(i-1)
       enddo

       do i= 1, N
         jS= INDEX(i-1)
         if (i.eq.1) then
           ITEM(jS+1)= i+1
           AMAT(jS+1)= 0.d0
           DIAG(i   )= 1.d0
            RHS(i   )= 0.d0
          else if                                          &
     &      (i.eq.N) then
           ITEM(jS+1)= i-1
           DIAG(i   )= -1.d0/dX
          else
           ITEM(jS+1)= i-1
           ITEM(jS+2)= i+1
           if (i-1.eq.1) then
             AMAT(jS+1)= 0.d0
           endif
         endif
       enddo
```

境界条件(i=1):後述

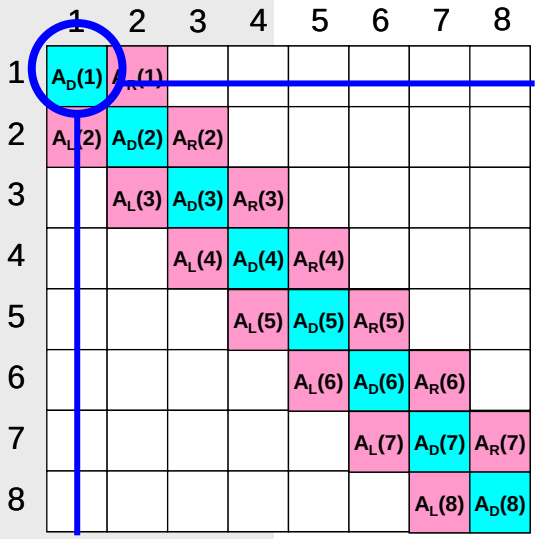
境界条件(i=N):S2のときと同じ

それ以外

	1	2	3	4	5	6	7	8
1	$A_D(1)$	$A_R(1)$						
2	$A_L(2)$	$A_D(2)$	$A_R(2)$					
3		$A_L(3)$	$A_D(3)$	$A_R(3)$				
4			$A_L(4)$	$A_D(4)$	$A_R(4)$			
5				$A_L(5)$	$A_D(5)$	$A_R(5)$		
6					$A_L(6)$	$A_D(6)$	$A_R(6)$	
7						$A_L(7)$	$A_D(7)$	$A_R(7)$
8							$A_L(8)$	$A_D(8)$

CG法による一次元熱伝導方程式 プログラム概要(3/7)

```
      INDEX= 2
!C
!C-- CONNECTIVITY
       INDEX(0)= 0
       INDEX(1)= 1
       INDEX(N)= 1

       do i= 1, N
         INDEX(i)= INDEX(i) + INDEX(i-1)
       enddo

       do i= 1, N
         jS= INDEX(i-1)
         if (i.eq.1) then
           ITEM(jS+1)= i+1
           AMAT(jS+1)= 0.d0
           DIAG(i   )= 1.d0
            RHS(i   )= 0.d0
          else if
     &      (i.eq.N) then
           ITEM(jS+1)= i-1
           DIAG(i   )= -1.d0/dX
          else
           ITEM(jS+1)= i-1
           ITEM(jS+2)= i+1
           if (i-1.eq.1) then
             AMAT(jS+1)= 0.d0
           endif
         endif
       enddo
```



境界条件(i=1):後述

境界条件(i=N):SORのときと同じ

それ以外

固定境界条件が指定されている点を
非対角成分として持っている場合
非対角成分をゼロクリアする。

境界条件の処理：i=N

```
i  = N
jS= INDEX(i-1)

ITEM(jS+1)= i-1
AMAT(jS+1)= +1.d0/dx  デフォルト値
DIAG(i   )= -1.d0/dX
```

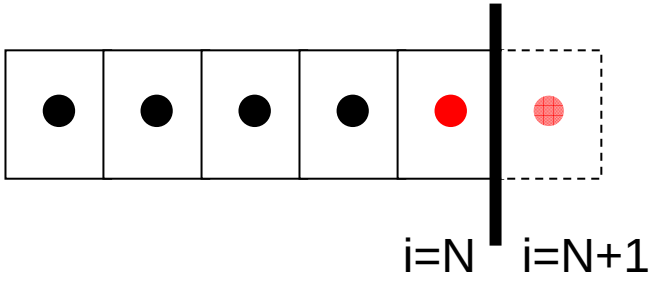
$$\frac{d\phi}{dx}=0 @ x=x_{\max} \Rightarrow \frac{\phi_{N+1}-\phi_N}{\Delta x}=0$$

$$\left(\frac{\phi_{N+1}-2\phi_N+\phi_{N-1}}{\Delta x^2}\right)\times\Delta x+BF\times\Delta x=0$$

$$\Rightarrow\left(\frac{-\phi_N+\phi_{N-1}}{\Delta x^2}\right)\times\Delta x+BF\times\Delta x=0$$

$$\Rightarrow (0)\phi_{N+1}+\left(\frac{-1}{\Delta x}\right)\phi_N+\left(\frac{1}{\Delta x}\right)\phi_{N-1}=-BF\times\Delta x$$

DIAG(i)　AMAT(jS+1)　RHS



境界面で断熱条件が成立するためには，$\phi_{N+1}=\phi_N$を満たすような仮想的な要素があると都合が良い

境界条件の処理：i=1（1/4）

```
ITEM(jS+1)= i+1
AMAT(jS+1)= 0.d0
DIAG(i   )= 1.d0
RHS (i   )= 0.d0
```

$$\phi = 0 @ x = 0 \Rightarrow \phi_1 = 0$$

$$\Rightarrow (0)\phi_2 + (1)\phi_1 + (0)\phi_0 = 0$$

DIAG **RHS**

	1	2	3	4	5	6	7	8
1	1	0						
2	a	-2a	a					
3		a	-2a	a				
4			a	-2a	a			
5				a	-2a	a		
6					a	-2a	a	
7						a	-2a	a
8							a	-a

$$a = \frac{1}{\Delta x}$$

- 課題S2ではこのようにしていた
- これでは係数行列が対称とならないため共役勾配法が使えない。

境界条件の処理：i=1（2/4）

```
ITEM(jS+1)= i+1
AMAT(jS+1)= 0.d0
DIAG(i   )= 1.d0
RHS (i   )= 0.d0
```

$$\phi = 0 @ x = 0 \Rightarrow \phi_1 = 0$$

$$\Rightarrow (0)\phi_2 + (1)\phi_1 + (0)\phi_0 = 0$$

DIAG **RHS**

	1	2	3	4	5	6	7	8
1	1	0						
2	a	-2a	a					
3		a	-2a	a				
4			a	-2a	a			
5				a	-2a	a		
6					a	-2a	a	
7						a	-2a	a
8							a	-a

$$a = \frac{1}{\Delta x}$$

- 課題S2ではこのようにしていた
- これでは係数行列が対称とならないため共役勾配法が使えない。

境界条件の処理：i=1（3/4）

- i=2における方程式

$$(a)\phi_3 + (-2a)\phi_2 + (a)\phi_1 = RHS_2$$

- ここでϕ_1は既知の値 $\bar{\phi}_1$

	1	2	3	4	5	6	7	8
1	1	0						
2	a	-2a	a					
3		a	-2a	a				
4			a	-2a	a			
5				a	-2a	a		
6					a	-2a	a	
7						a	-2a	a
8							a	-a

$$a = \frac{1}{\Delta x}$$

境界条件の処理：i=1（4/4）

- i=2における方程式

$$(a)\phi_3 + (-2a)\phi_2 + (a)\phi_1 = RHS_2$$

- ここでϕ_1は既知の値 $\bar{\phi}_1$

$$(a)\phi_3 + (-2a)\phi_2 + (0)\phi_1 = RHS_2 - (a)\bar{\phi}_1$$

- このケースの場合は，$\bar{\phi}_1 = 0$であるため，右辺（RHS）の修正は不要である。

	1	2	3	4	5	6	7	8
1	1	0						
2	0	-2a	a					
3		a	-2a	a				
4			a	-2a	a			
5				a	-2a	a		
6					a	-2a	a	
7						a	-2a	a
8							a	-a

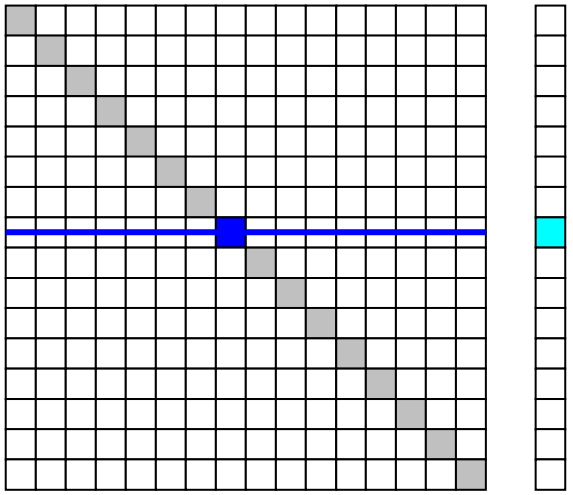
$$a = \frac{1}{\Delta x}$$

固定境界条件の処理：消去

i=kkの点でPHIxの値に固定されている場合

```
do i= 1, N
  if (i.eq.kk) then
    DIAG(i)= 1.d0
    RHS (i)= PHIx
    do j= INDEX(i-1)+1, INDEX(i)
      AMAT(j)= 0.d0
    enddo
  endif
endif

do i= 1, N
  if (i.ne.kk) then
    do j= INDEX(i-1)+1, INDEX(i)
      jj= ITEM(j)
      if (jj.eq.kk) then
        RHS (i)= RHS(i) - AMAT(j)*PHIx
        AMAT(j)= 0.d0
      endif
    enddo
  enddo
enddo
```

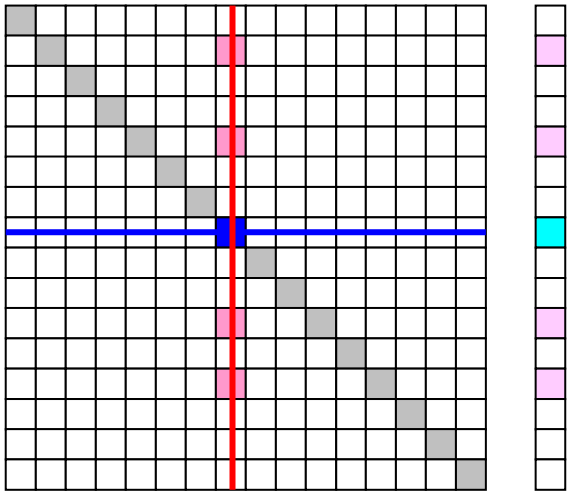
ゼロクリア



固定境界条件の処理：消去
i=kkの点でPHIxの値に固定されている場合

```
      do i= 1, N
        if (i.eq.kk) then
          DIAG(i)= 1.d0
          RHS (i)= PHIx
          do j= INDEX(i-1)+1, INDEX(i)
            AMAT(j)= 0.d0
          enddo
        endif
      endif

      do i= 1, N
        if (i.ne.kk) then
          do j= INDEX(i-1)+1, INDEX(i)
            jj= ITEM(j)
            if (jj.eq.kk) then
              RHS (i)= RHS(i) - AMAT(j)*PHIx
              AMAT(j)= 0.d0
            endif
          enddo
        enddo
      enddo
```

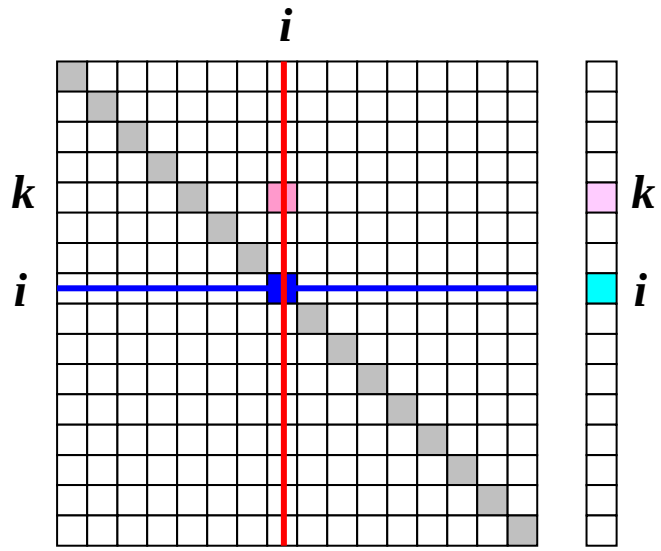
ゼロクリア

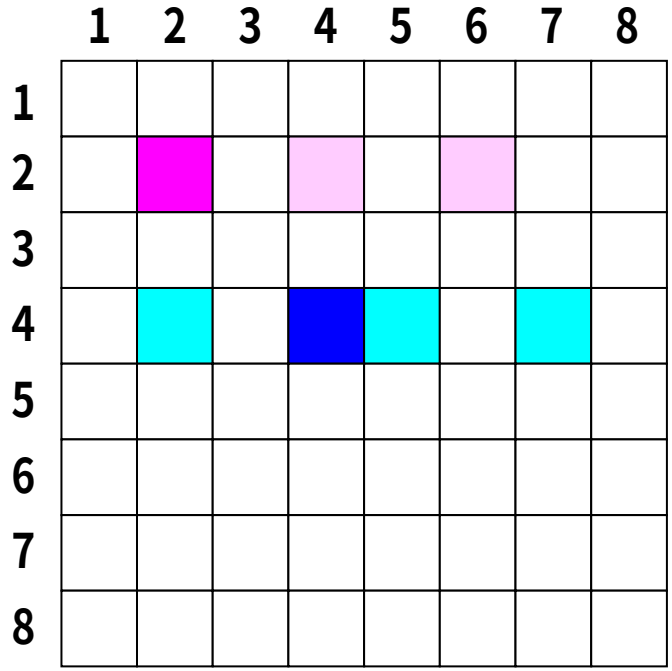


今回の問題ではPHIX(jj)に相当するものが0だったのでこの右辺の処理は不要であった。

固定境界条件の処理:具体例

点「i」で固定境界条件を適用しているものとする $\phi = \tilde{\phi}_i$





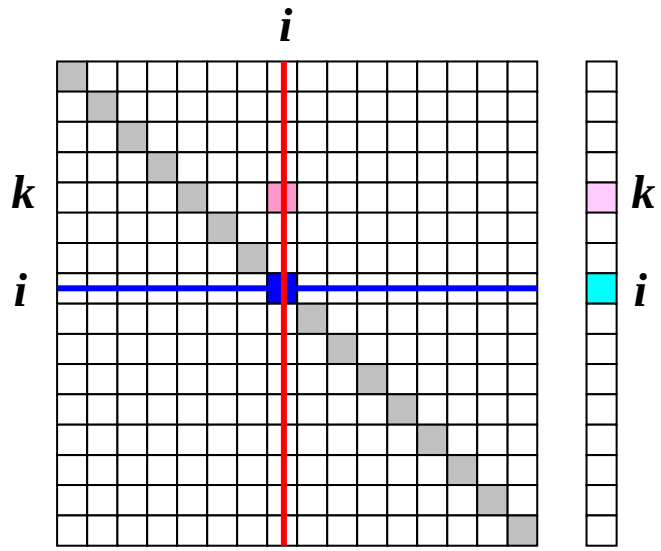
$i=4, k=2$ とすると

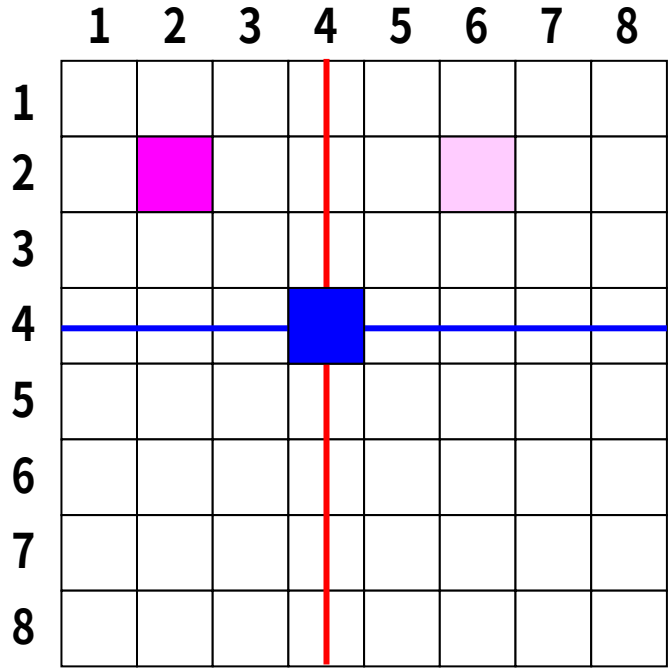
$$DIAG_2\phi_2 + AMAT_{24}\phi_4 + AMAT_{26}\phi_6 = RHS_2$$

$$DIAG_4\phi_4 + AMAT_{42}\phi_2 + AMAT_{45}\phi_5 + AMAT_{47}\phi_7 = RHS_4$$

固定境界条件の処理:具体例

点「i」で固定境界条件を適用しているものとする $\phi = \tilde{\phi}_i$





$i=4, k=2$ とすると

$$DIAG_2\phi_2 + AMAT_{24}\phi_4 + AMAT_{26}\phi_6 = RHS_2$$

$$DIAG_4\phi_4 + AMAT_{42}\phi_2 + AMAT_{45}\phi_5 + AMAT_{47}\phi_7 = RHS_4$$

$$DIAG_2\phi_2 + AMA_{26}\phi_6 = RHS_2 - AMAT_{24}\tilde{\phi}_4$$

$$\phi_4 = \tilde{\phi}_4$$

CG法による一次元熱伝導方程式 プログラム概要(4/7)

```
!C
!C +---------------+
!C | CG iterations |
!C +---------------+
!C===
      R = 1
      Z = 2
      Q = 2
      P = 3
      DD= 4

      do i= 1, N
        W(i,DD)= 1.0D0 / DIAG(i)
      enddo
!C
!C-- {r0}= {b} - [A]{xini} |

      do i= 1, N
        W(i,R) = DIAG(i)*PHI(i)
        do j= INDEX(i-1)+1, INDEX(i)
          W(i,R) = W(i,R) + AMAT(j)*PHI(ITEM(j))
        enddo
      enddo

      BNRM2= 0.0D0
      do i= 1, N
        BNRM2 = BNRM2  + RHS(i)  **2
        W(i,R)= RHS(i) - W(i,R)
      enddo
!C*****************************************************************
```

対角成分の逆数(前処理用)
その都度，除算をすると効率が悪いため，予め配列に格納

CG法による一次元熱伝導方程式 プログラム概要(4/7)

```
!C
!C +---------------+
!C | CG iterations |
!C +---------------+
!C===
      R = 1
      Z = 2
      Q = 2
      P = 3
      DD= 4

      do i= 1, N
        W(i,DD)= 1.0D0 / DIAG(i)
      enddo
```

W(i,1)= W(i,R) ⇒ {r}
W(i,2)= W(i,Z) ⇒ {z}
W(i,2)= W(i,Q) ⇒ {q}
W(i,3)= W(i,P) ⇒ {p}

W(i,4)= W(i,DD) ⇒ 1/DIAG

Compute $r^{(0)}= b-[A]x^{(0)}$
for i= 1, 2, …
  solve $[M]z^{(i-1)}= r^{(i-1)}$
  $\rho_{i-1}= r^{(i-1)}\ z^{(i-1)}$
  if i=1
    $p^{(1)}= z^{(0)}$
  else
    $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
    $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i)}$
  endif
  $q^{(i)}= [A]p^{(i)}$
  $\alpha_i = \rho_{i-1}/p^{(i)}q^{(i)}$
  $x^{(i)}= x^{(i-1)} + \alpha_i p^{(i)}$
  $r^{(i)}= r^{(i-1)} - \alpha_i q^{(i)}$
  check convergence |r|
end

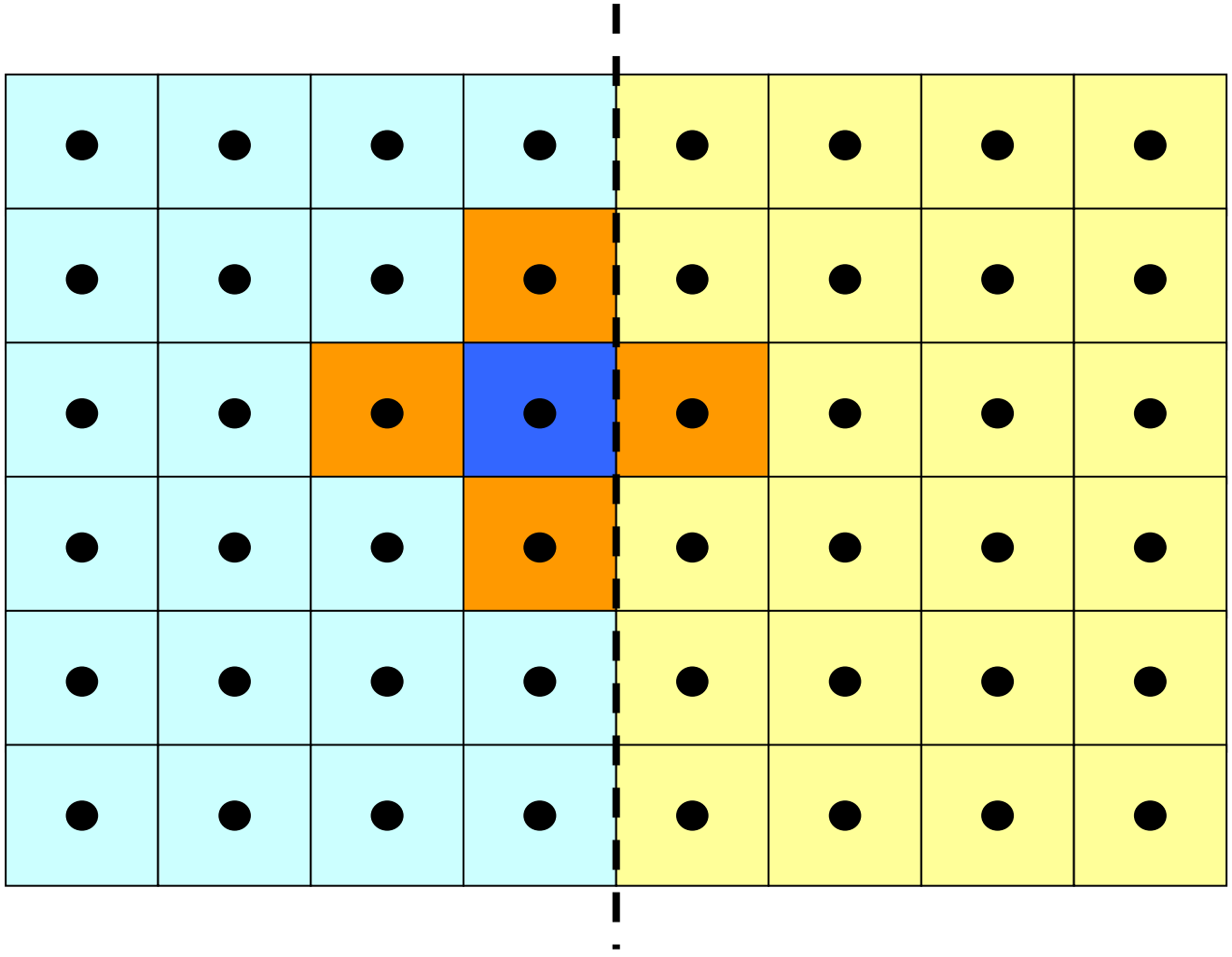
前処理付き共役勾配法
Preconditioned Conjugate Gradient Method（CG）

Compute $\mathbf{r}^{(0)}= \mathbf{b}-[A]\mathbf{x}^{(0)}$
for i= 1, 2, …
  solve $[M]z^{(i-1)}= r^{(i-1)}$
  $\rho_{i-1}= \mathbf{r}^{(i-1)}\ \mathbf{z}^{(i-1)}$
  if i=1
    $p^{(1)}= z^{(0)}$
  else
    $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
    $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i)}$
  endif
  $\mathbf{q}^{(i)}= [A]\mathbf{p}^{(i)}$
  $\alpha_i = \rho_{i-1}/\mathbf{p}^{(i)}\mathbf{q}^{(i)}$
  $x^{(i)}= x^{(i-1)} + \alpha_i p^{(i)}$
  $r^{(i)}= r^{(i-1)} - \alpha_i q^{(i)}$
  check convergence $|\mathbf{r}|$
end

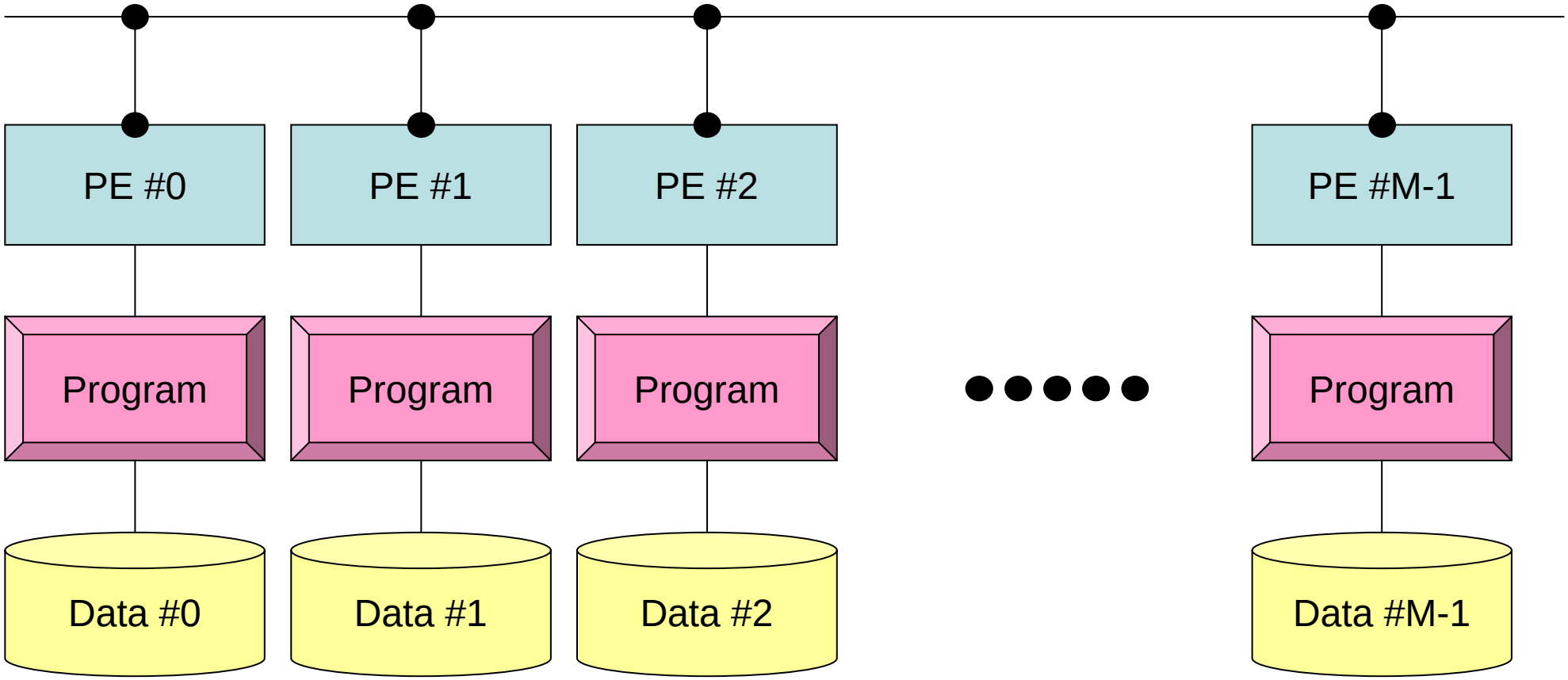
並列計算，領域間通信が必要な部分

- 行列ベクトル積
- 内積

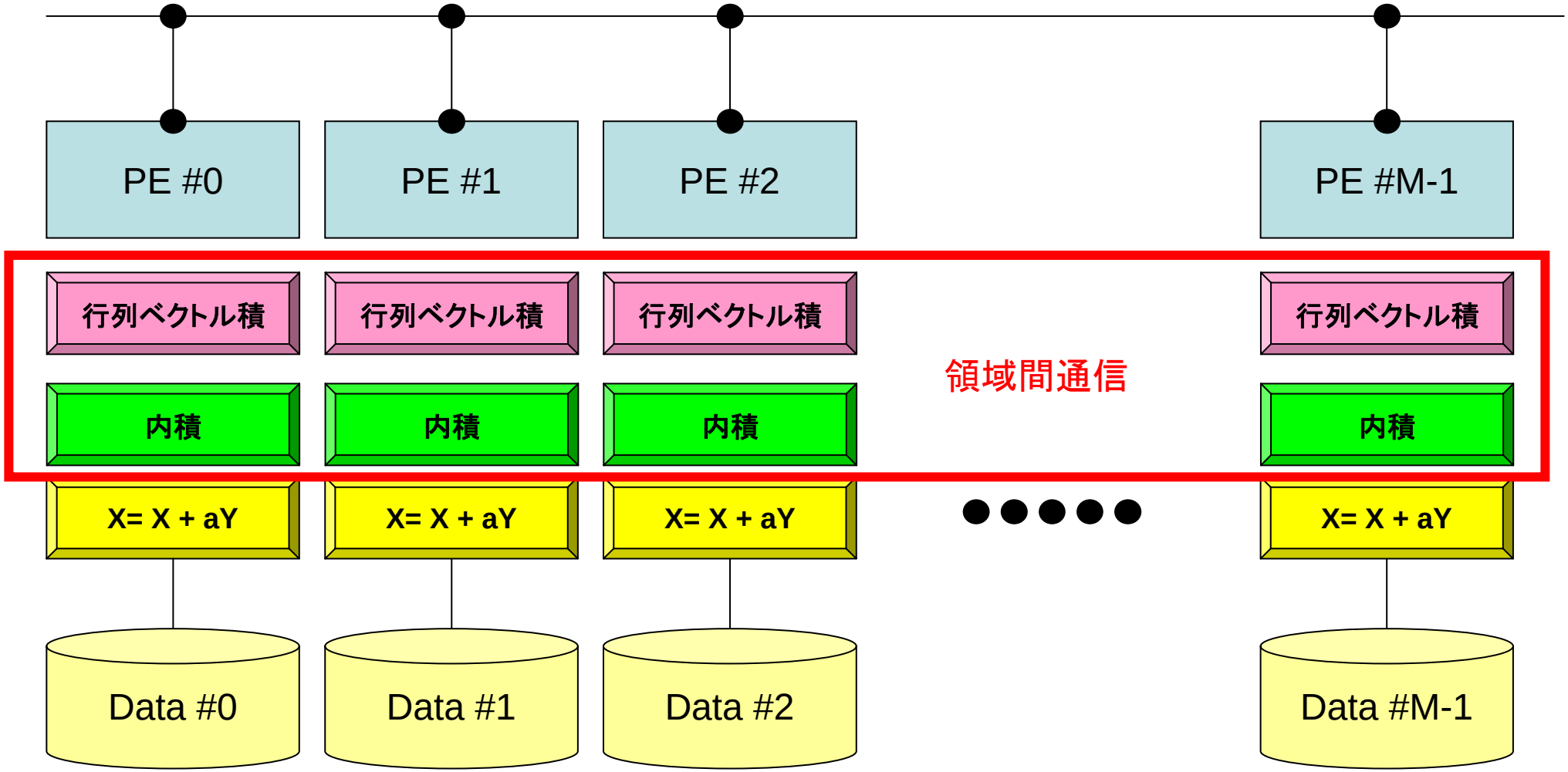
行列ベクトル積：他領域の値が必要
オーバーラップ＋通信テーブル



再びSPMD


PE #0
PE #1
PE #2
PE #M-1
Program
Program
Program
Program
Data #0
Data #1
Data #2
Data #M-1

CG法でははこうなる


PE #0
PE #1
PE #2
PE #M-1
行列ベクトル積
行列ベクトル積
行列ベクトル積
行列ベクトル積
内積
内積
内積
内積
X= X + aY
X= X + aY
X= X + aY
X= X + aY
領域間通信
Data #0
Data #1
Data #2
Data #M-1

共役勾配法の「並列化」(1/2)

- 行列ベクトル積
  - 課題**S2**で作成したような，領域境界データの交換を実施して，外点のベクトル値の最新値を得ておく。

```
!C
!C-- {q}= [A]{p}

      exchange W(i,P)

      do i= 1, N
        W(i,Q) = DIAG(i)*W(i,P)
        do j= INDEX(i-1)+1, INDEX(i)
          W(i,Q) = W(i,Q) + AMAT(j)*W(ITEM(j),P)
        enddo
      enddo
```

共役勾配法の「並列化」(2/2)

- 内積
  - MPI_ALLREDUCE

```
!C
!C +--------------+
!C | RHO= {r}{z} |
!C +--------------+
!C===
      RHO= 0.d0

      do i= 1, N
        RHO= RHO + W(i,R)*W(i,Z)
      enddo

      allreduce RHO

!C===
```

課題S3の方針，ファイル

- 基本的に一次元例題，課題S2のアプローチを踏襲

```
>$ cd <$07S>    各自作成したディレクトリ

FORTRAN
>$ cp /home/nakajima/class/2007summer/F/s3r-f.tar .
>$ tar xvf s3r-f.tar

C
>$ cp /home/nakajima/class/2007summer/C/s3r-c.tar .
>$ tar xvf s3r-c.tar
```

直下に「/S3-ref」というディレクトリができている。
<$07S>/S3-refを<$S3R>と呼ぶ。

ファイル

s3a.f

s3a.c：従来型（PHI(-1)を許す）
s3b.c：厳密型（PHI(-1)をPHI(N+1)とする）

input.dat
cinput.dat
go.sh

```
mpif90 -O3 XXX.f
mpicc  -O3 XXX.c
<modify “go.sh”>
qsub go.sh
```

S3：一次元熱伝導方程式

並列版（s3a.f，SEND/RECV使用）（1/7）

```
      program CG_poi
      implicit REAL*8 (A-H,O-Z)
      include 'mpif.h'

      integer :: PETOT, my_rank, ierr
      integer :: N, ITERmax
      integer :: R, Z, P, Q, DD

      real(kind=8) :: dx, RESID, dPHI, dPHImax, BF, EPS
      real(kind=8), dimension(:), allocatable :: PHI, RHS
      real(kind=8), dimension(:  ), allocatable :: DIAG, AMAT
      real(kind=8), dimension(:,:), allocatable :: W

      integer, dimension(:  ), allocatable :: INDEX, ITEM

      integer(kind=4) :: NEIBPETOT, BUFlength
      integer(kind=4), dimension(2) :: NEIBPE

      integer(kind=4), dimension(0:2) :: import_index, export_index
      integer(kind=4), dimension(  2) :: import_item , export_item

      real(kind=8), dimension(2) :: SENDbuf, RECVbuf

      integer(kind=4), dimension(:), allocatable :: stat1

!C
!C +-------+
!C | INIT. |
!C +-------+
!C===

!C
!C-- MPI init.
      call MPI_INIT      (ierr)
      call MPI_COMM_SIZE (MPI_COMM_WORLD, PETOT, ierr )
      call MPI_COMM_RANK (MPI_COMM_WORLD, my_rank, ierr )
```

S3：一次元熱伝導方程式

並列版（s3a.f，SEND/RECV使用）（2/7）

```
!C
!C-- CTRL data
      if (my_rank.eq.0) then
        open  (11, file='input.dat', status='unknown')
          read (11,*) Ng
          read (11,*) dX, BF
          read (11,*) ITERmax
          read (11,*) EPS
        close (11)
      endif

      call MPI_BCAST (Ng     , 1, MPI_INTEGER, 0, MPI_COMM_WORLD, ierr)
      call MPI_BCAST (ITERmax, 1, MPI_INTEGER, 0, MPI_COMM_WORLD, ierr)
      call MPI_BCAST (dx,    1, MPI_DOUBLE_PRECISION, 0,                &
     &                                               MPI_COMM_WORLD, ierr)
      call MPI_BCAST (BF,    1, MPI_DOUBLE_PRECISION, 0,                &
     &                                               MPI_COMM_WORLD, ierr)
      call MPI_BCAST (EPS,   1, MPI_DOUBLE_PRECISION, 0,                &
     &                                               MPI_COMM_WORLD, ierr)

!C
!C-- LOCAL MESH size
      N= Ng / PETOT

      if (N.le.1) goto 900

      nr = Ng - nnn*PETOT
      if (my_rank+1.le.nr) N= N + 1
```

S3：一次元熱伝導方程式
並列版（s3a.f，SEND/RECV使用）（2/7）

```
!C
!C-- CTRL data
      if (my_rank.eq.0) then
        open  (11, file='input.dat', status='unknown')
          read (11,*) N
          read (11,*) dX, BF
          read (11,*) ITERmax
          read (11,*) EPS
        close (11)
      endif

      call MPI_BCAST (Ng     , 1, MPI_INTEGER, 0, MPI_COMM_WORLD, ierr)
      call MPI_BCAST (ITERmax, 1, MPI_INTEGER, 0, MPI_COMM_WORLD, ierr)
      call MPI_BCAST (dx,    1, MPI_DOUBLE_PRECISION, 0,                &
     &                                               MPI_COMM_WORLD, ierr)
      call MPI_BCAST (BF,    1, MPI_DOUBLE_PRECISION, 0,                &
     &                                               MPI_COMM_WORLD, ierr)
      call MPI_BCAST (EPS,   1, MPI_DOUBLE_PRECISION, 0,                &
     &                                               MPI_COMM_WORLD, ierr)

!C
!C-- LOCAL MESH size
      N= Ng / PETOT

      if (N.le.1) goto 900

      nr = Ng - nnn*PETOT
      if (my_rank+1.le.nr) N= N + 1
```

課題S2と同じ

S3：一次元熱伝導方程式

並列版（s3a.f，SEND/RECV使用）（3/7）

```
!C
!C-- MATRIX
      allocate (PHI(0:N+1), DIAG(N), AMAT(2*N),RHS(N))
      allocate (INDEX(0:N), ITEM(2*N), W(0:N+1,4))
       PHI= 0.d0
      AMAT=  1.d0/dX
      DIAG= -2.d0/dX
       RHS=  -BF * dX

      INDEX= 2
!C
!C-- CONNECTIVITY
      INDEX(0)= 0

      if (my_rank.eq.0)        INDEX(1   )= 1
      if (my_rank.eq.PETOT-1) INDEX(nnn)= 1

      do i= 1, nnn
        INDEX(i)= INDEX(i) + INDEX(i-1)
      enddo
```

	1	2	3	4	5	6	7	8
1	$A_D(1)$	$A_R(1)$						
2	$A_L(2)$	$A_D(2)$	$A_R(2)$					
3		$A_L(3)$	$A_D(3)$	$A_R(3)$				
4			$A_L(4)$	$A_D(4)$	$A_R(4)$			
5				$A_L(5)$	$A_D(5)$	$A_R(5)$		
6					$A_L(6)$	$A_D(6)$	$A_R(6)$	
7						$A_L(7)$	$A_D(7)$	$A_R(7)$
8							$A_L(8)$	$A_D(8)$

$$\left(\frac{\phi_{i+1}-2\phi_i+\phi_{i-1}}{\Delta x^2}\right)\times\Delta x+BF\times\Delta x=0$$

$$\frac{\phi_{i+1}-\phi T_i+\phi_{i-1}}{\Delta x}=\left(\frac{1}{\Delta x}\right)\phi_{i-1}-\left(\frac{2}{\Delta x}\right)\phi_i+\left(\frac{1}{\Delta x}\right)\phi_{i+1}=-BF\times\Delta x$$

A_L A_D A_R RHS

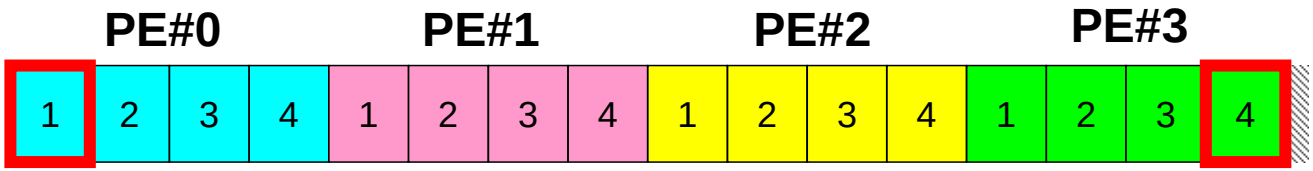
S3-1：一次元熱伝導方程式

並列版（s3-1.f，SEND/RECV使用）（3/7）

```
!C
!C-- MATRIX
      allocate (PHI(0:N+1), DIAG(N), AMAT(2*N),RHS(N))
      allocate (INDEX(0:N), ITEM(2*N), W(0:N+1,4))
       PHI= 0.d0
      AMAT=  1.d0/dX
      DIAG= -2.d0/dX
       RHS=  -BF * dX

      INDEX= 2
!C
!C-- CONNECTIVITY
      INDEX(0)= 0

      if (my_rank.eq.0)        INDEX(1)= 1
      if (my_rank.eq.PETOT-1)  INDEX(N)= 1

      do i= 1, N
        INDEX(i)= INDEX(i) + INDEX(i-1)
      enddo
```

	1	2	3	4	5	6	7	8
1	AD1	AR1						
2	AL2	AD2	AR2					
3		AL3	AD3	AR3				
4			AL4	AD4	AR4			
5				AL5	AD5	AR5		
6					AL6	AD6	AR6	
7						AL7	AD7	AR7
8							AL8	AD8

非対角成分の数は「2」
ただし，両端では「1」



S3：一次元熱伝導方程式

並列版（s3a.f，SEND/RECV使用）（4/7）

```
      do i= 1, N
        jS= INDEX(i-1)
        if (my_rank.eq.0 .and. i.eq.1) then
          ITEM(jS+1)= i+1
          AMAT(jS+1)= 0.d0
          DIAG(i   )= 1.d0
           RHS(i   )= 0.d0
         else if                                                    &
     &     (my_rank.eq.PETOT-1 .and. i.eq.nnn) then
          ITEM(jS+1)= i-1
          DIAG(i   )= -1.d0/dX
         else
          ITEM(jS+1)= i-1
          ITEM(jS+2)= i+1
          if (my_rank.eq.0.and.i-1.eq.1) then
            AMAT(jS+1)= 0.d0
          endif
        endif
      enddo
```

境界条件（Xmin）

境界条件（Xmax）

それ以外

S3：一次元熱伝導方程式

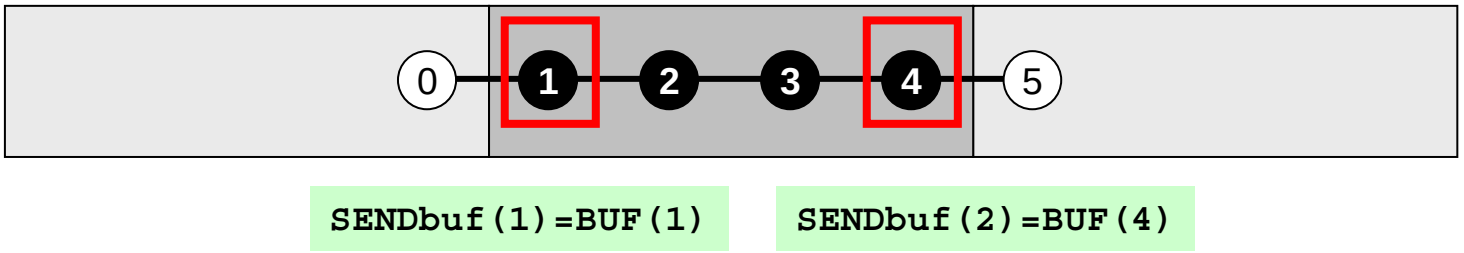
並列版（s3a.f，SEND/RECV使用）（5/7）

一般化された通信テーブル

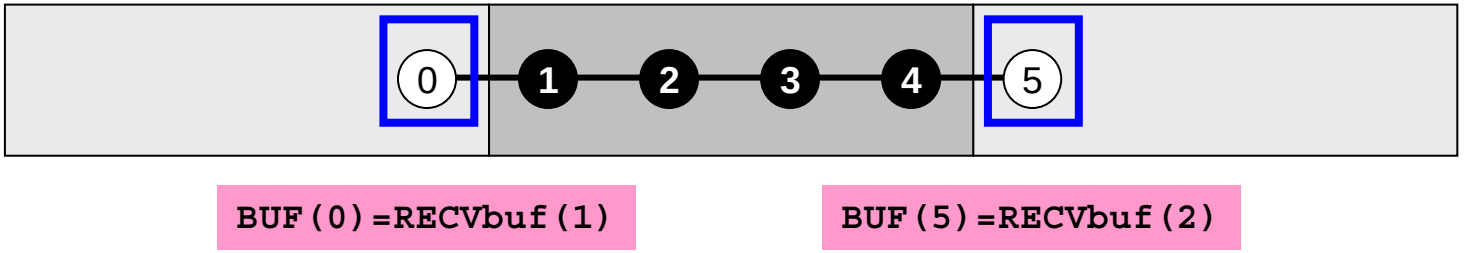
```
!C
!C-- COMMUNICATION
      NEIBPETOT= 2
      if (my_rank.eq.0       ) NEIBPETOT= 1
      if (my_rank.eq.PETOT-1) NEIBPETOT= 1
      if (PETOT.eq.1)          NEIBPETOT= 0

      NEIBPE(1)= my_rank - 1
      NEIBPE(2)= my_rank + 1

      if (my_rank.eq.0       ) NEIBPE(1)= my_rank + 1
      if (my_rank.eq.PETOT-1) NEIBPE(1)= my_rank - 1

      BUFlength= 1

      import_index= 0
      export_index= 0
      import_item = 0
      export_item = 0

      import_index(1)= 1
      import_index(2)= 2
      import_item (1)= 0
      import_item (2)= N+1

      export_index(1)= 1
      export_index(2)= 2
      export_item (1)= 1
      export_item (2)= N

      if (my_rank.eq.0) then
        import_item (1)= N+1
        export_item (1)= N
      endif
```

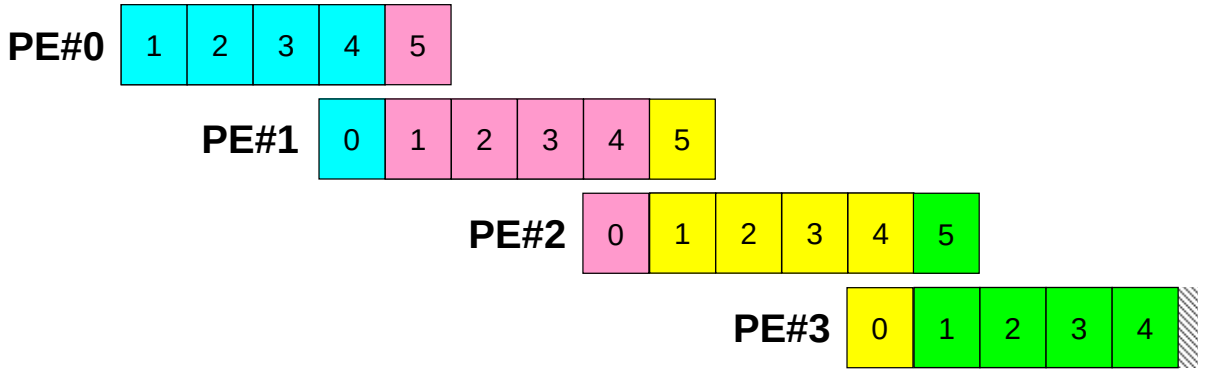
1D差分法の並列計算に必要な情報
通信テーブル

隣接PEへの
送信(境界点)

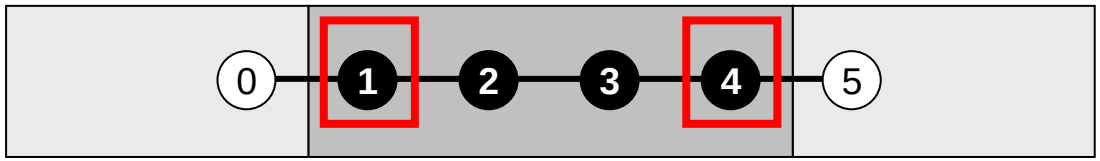


隣接PEからの
受信(外点)

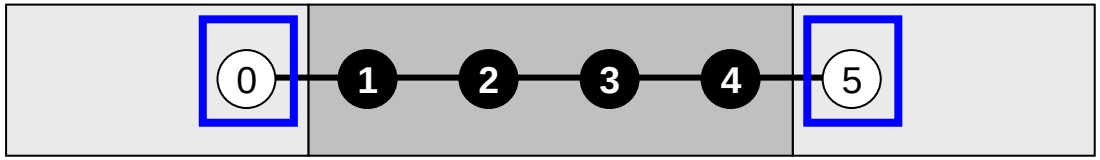




一般化された通信テーブル



SENDbuf(1)=BUF(1)　　SENDbuf(2)=BUF(4)



BUF(0)=RECVbuf(1)　　BUF(5)=RECVbuf(2)

```
NEIBPETOT= 2
NEIBPE(1)= my_rank - 1
NEIBPE(2)= my_rank + 1

import_index(1)= 1
import_index(2)= 2
import_item  (1)= 0
import_item  (2)= N+1

export_index(1)= 1
export_index(2)= 2
export_item  (1)= 1
export_item  (2)= N

if (my_rank.eq.0) then
  import_item  (1)= N+1
  export_item  (1)= N
  NEIBPE(1)= my_rank+1
endif
```

S3：一次元熱伝導方程式

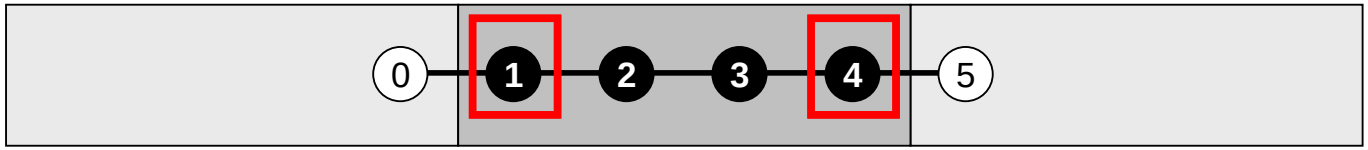
並列版（s3a.f，SEND/RECV使用）（6/7）

一般化された通信テーブルによる通信：行列ベクトル積

```
!C
!C-- {q}= [A]{p}
!C-    init
        do neib= 1, NEIBPETOT
          do k= export_index(neib-1)+1, export_index(neib)
            kk= export_item(k)
            SENDbuf(k)= W(kk,P)
          enddo
        enddo
!C
!C-- SEND & RECV.
        do neib= 1,  NEIBPETOT
          is= export_index(neib-1) + 1
          ir= import_index(neib-1) + 1
          len_s= export_index(neib) - export_index(neib-1)
          len_r= import_index(neib) - import_index(neib-1)
          call MPI_SENDRECV
     &         (SENDbuf(is), len_s, MPI_DOUBLE_PRECISION,NEIBPE(neib),0,&
     &          RECVbuf(ir), len_r, MPI_DOUBLE_PRECISION,NEIBPE(neib),0,&
     &          MPI_COMM_WORLD, stat1, ierr)
        enddo
!C-    update
        do neib= 1, NEIBPETOT
          do k= import_index(neib-1)+1, import_index(neib)
            kk= import_item(k)
            W(kk,P)= RECVbuf(k)
          enddo
        enddo
        do i= 1, N
          W(i,Q) = DIAG(i)*W(i,P)
          do j= INDEX(i-1)+1, INDEX(i)
            W(i,Q) = W(i,Q) + AMAT(j)*W(ITEM(j),P)
          enddo
        enddo
```

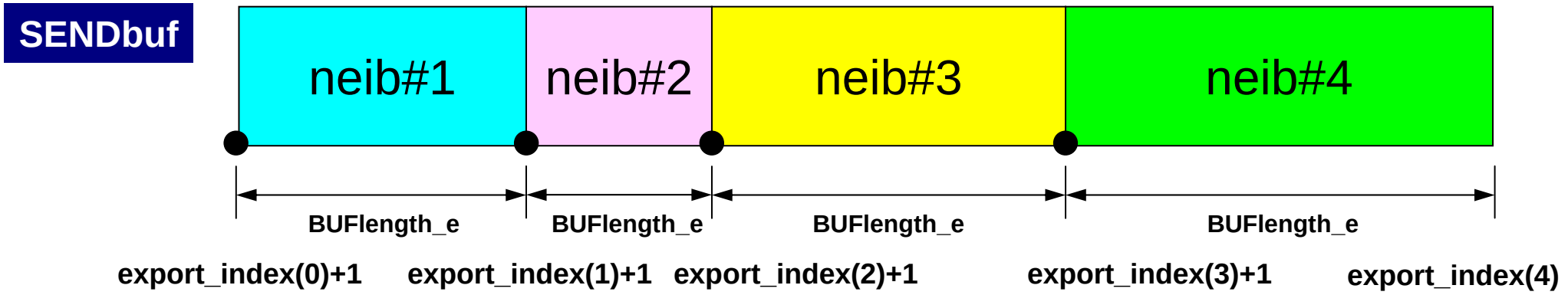
送信バッファに
W(i,p) の値を入れる

送信:一次元問題



- 送信相手
  - NEIBPETOT, NEIB(neib)
    - NEIBPETOT=2, NEIB(1)= my_rank-1, NEIB(2)= my_rank+1
- それぞれの送信相手に送るメッセージサイズ
  - export_index(neib), neib= 1, NEIBPETOT
    - export_index(0)=0, export_index(1)= 1, export_index(2)= 2
- 「境界点」番号
  - export_item(k), k= 1, export_index(NEIBPETOT)
    - export_item(1)= 1, export_item(2)= N
- それぞれの送信相手に送るメッセージ
  - SENDbuf(k), k= 1, export_index(NEIBPETOT)
    - SENDbuf(1)= BUF(1), SENDbuf(2)= BUF(N)

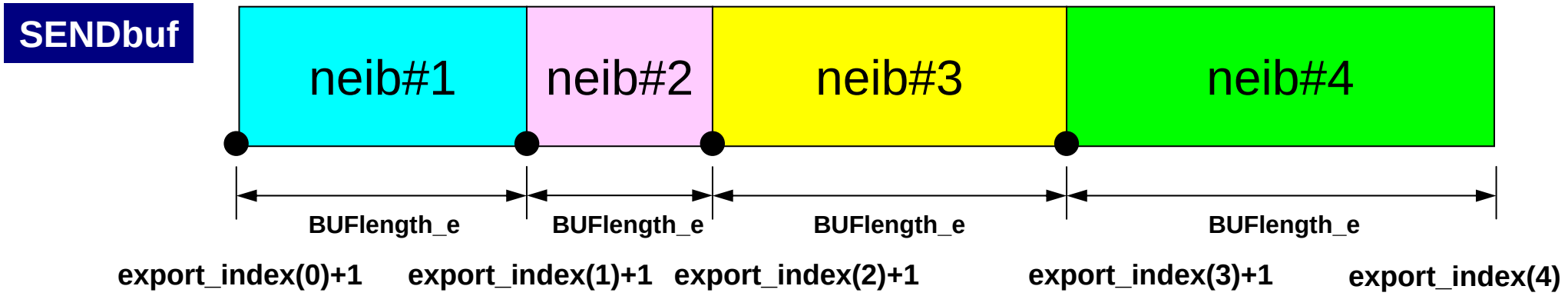
送信（MPI_Isend/Irecv/Waitall）




```
do neib= 1, NEIBPETOT
  do k= export_index(neib-1)+1, export_index(neib)
    kk= export_item(k)
    SENDbuf(k)= VAL(kk)
  enddo
enddo

do neib= 1, NEIBPETOT
  iS_e= export_index(neib-1) + 1
  iE_e= export_index(neib  )
  BUFlength_e= iE_e + 1 - iS_e

  call MPI_ISEND                                                    &
&         (SENDbuf(iS_e), BUFlength_e, MPI_INTEGER, NEIBPE(neib), 0,&
&          MPI_COMM_WORLD, request_send(neib), ierr)
enddo

call MPI_WAITALL (NEIBPETOT, request_send, stat_recv, ierr)
```

送信バッファへの代入
温度などの変数を直接送信，受信に使うのではなく，このようなバッファへ一回代入して計算することを勧める。

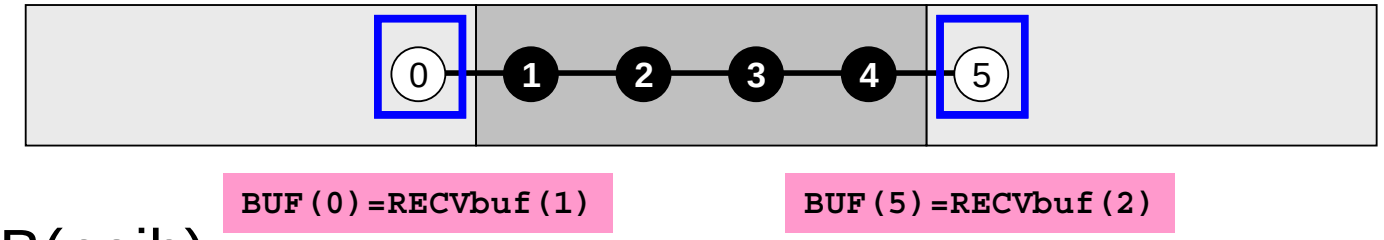
送信（MPI_Sendrecv）




```
do neib= 1, NEIBPETOT
  do k= export_index(neib-1)+1, export_index(neib)
    kk= export_item(k)
    SENDbuf(k)= VAL(kk)
  enddo
enddo

do neib= 1, NEIBPETOT
  iS_e= export_index(neib-1) + 1
  iE_e= export_index(neib  )
  BUFlength_e= iE_e + 1 - iS_e

  call MPI_SENDRECV                                                  &
&         (SENDbuf(iS_e), BUFlength_e, MPI_INTEGER, NEIBPE(neib), 0,&
&          RECVbuf(iS_i), BUFlength_i, MPI_INTEGER, NEIBPE(neib), 0,&
&          MPI_COMM_WORLD, stat_sr, ierr)
  enddo
```

送信バッファへの代入

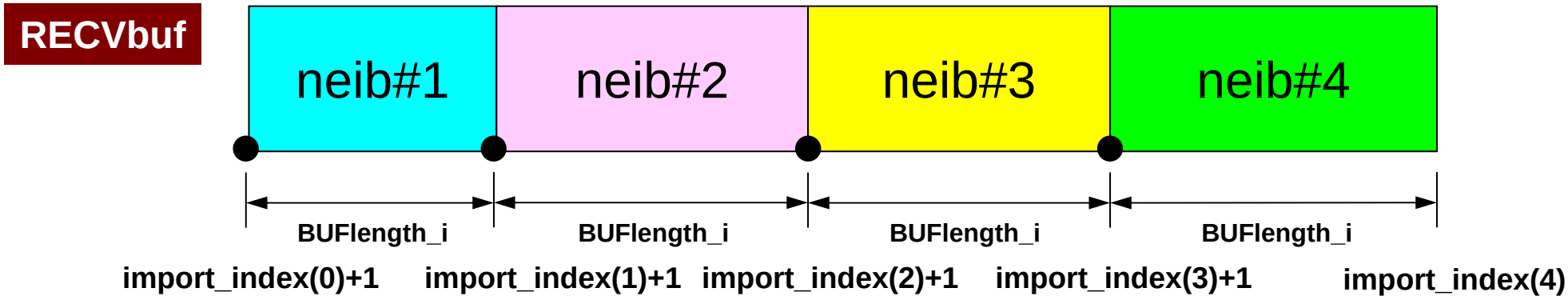
受信:一次元問題



- 受信相手
  - NEIBPETOT, NEIB(neib)
    - NEIBPETOT=2, NEIB(1)= my_rank-1, NEIB(2)= my_rank+1
- それぞれの受信相手から受け取るメッセージサイズ
  - import_index(neib), neib= 1, NEIBPETOT
    - import_index(0)=0, import_index(1)= 1, import_index(2)= 2
- 「外点」番号
  - import_item(k), k= 1, import_index(NEIBPETOT)
    - import_item(1)= 0, import_item(2)= N+1
- それぞれの受信相手から受け取るメッセージ
  - RECVbuf(k), k= 1, import_index(NEIBPETOT)
    - BUF(0)=RECVbuf(1), BUF(N+1)=RECVbuf(2)

受信（MPI_Isend/Irecv/Waitall）

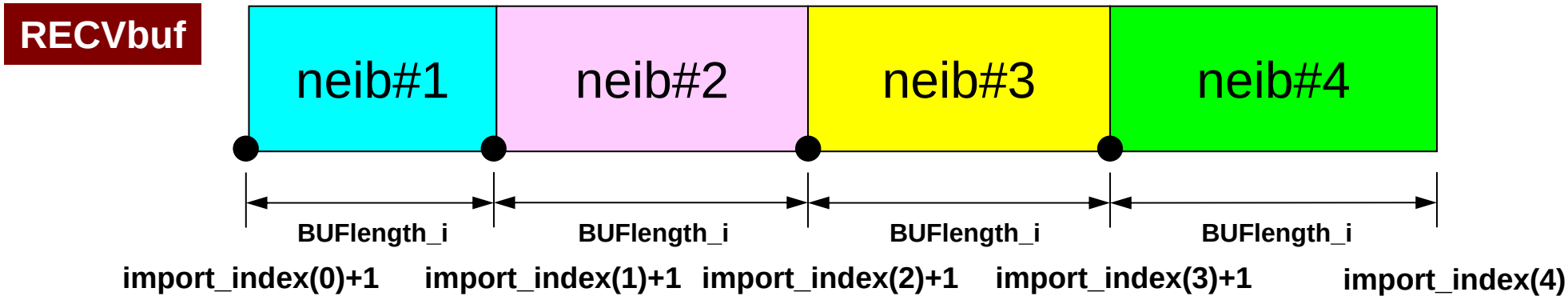
```
 do neib= 1, NEIBPETOT
   iS_i= import_index(neib-1) + 1
   iE_i= import_index(neib  )
   BUFlength_i= iE_i + 1 - iS_i

   call MPI_IRECV                                                  &
&          (RECVbuf(iS_i), BUFlength_i, MPI_INTEGER, NEIBPE(neib), 0,&
&           MPI_COMM_WORLD, request_recv(neib), ierr)
  enddo

  call MPI_WAITALL (NEIBPETOT, request_recv, stat_recv, ierr)

  do neib= 1, NEIBPETOT
     do k= import_index(neib-1)+1, import_index(neib)
       kk= import_item(k)
       VAL(kk)= RECVbuf(k)                                  受信バッファから代入
     enddo
  enddo
```



受信（MPI_Sendrecv）

```
 do neib= 1, NEIBPETOT
   iS_i= import_index(neib-1) + 1
   iE_i= import_index(neib  )
   BUFlength_i= iE_i + 1 - iS_i

   call MPI_SENDRECV                                                  &
&         (SENDbuf(iS_e), BUFlength_e, MPI_INTEGER, NEIBPE(neib), 0,&
&          RECVbuf(iS_i), BUFlength_i, MPI_INTEGER, NEIBPE(neib), 0,&
&          MPI_COMM_WORLD, stat_sr, ierr)
  enddo

  do neib= 1, NEIBPETOT
    do k= import_index(neib-1)+1, import_index(neib)
      kk= import_item(k)
      VAL(kk)= RECVbuf(k)
    enddo
  enddo
```

受信バッファからの代入



S3：一次元熱伝導方程式

並列版（s3a.f，SEND/RECV使用）（6/7）

一般化された通信テーブルによる通信：行列ベクトル積

```
!C
!C-- {q}= [A]{p}
!C-    init
        do neib= 1, NEIBPETOT
          do k= export_index(neib-1)+1, export_index(neib)
            kk= export_item(k)
            SENDbuf(k)= W(kk,P)
          enddo
        enddo
!C
!C-- SEND & RECV.
        do neib= 1,  NEIBPETOT
          is= export_index(neib-1) + 1
          ir= import_index(neib-1) + 1
          len_s= export_index(neib) - export_index(neib-1)
          len_r= import_index(neib) - import_index(neib-1)
          call MPI_SENDRECV
     &         (SENDbuf(is), len_s, MPI_DOUBLE_PRECISION,NEIBPE(neib),0,&
     &          RECVbuf(ir), len_r, MPI_DOUBLE_PRECISION,NEIBPE(neib),0,&
     &          MPI_COMM_WORLD, stat1, ierr)
        enddo
!C-    update
        do neib= 1, NEIBPETOT
          do k= import_index(neib-1)+1, import_index(neib)
            kk= import_item(k)
            W(kk,P)= RECVbuf(k)
          enddo
        enddo
        do i= 1, N
          W(i,Q) = DIAG(i)*W(i,P)
          do j= INDEX(i-1)+1, INDEX(i)
            W(i,Q) = W(i,Q) + AMAT(j)*W(ITEM(j),P)
          enddo
        enddo
```

送受信

S3：一次元熱伝導方程式

並列版（s3a.f，SEND/RECV使用）（6/7）

一般化された通信テーブルによる通信：行列ベクトル積

```
!C
!C-- {q}= [A]{p}
!C-    init
        do neib= 1, NEIBPETOT
          do k= export_index(neib-1)+1, export_index(neib)
            kk= export_item(k)
            SENDbuf(k)= W(kk,P)
          enddo
        enddo
!C
!C-- SEND & RECV.
        do neib= 1,  NEIBPETOT
          is= export_index(neib-1) + 1
          ir= import_index(neib-1) + 1
          len_s= export_index(neib) - export_index(neib-1)
          len_r= import_index(neib) - import_index(neib-1)
          call MPI_SENDRECV
     &         (SENDbuf(is), len_s, MPI_DOUBLE_PRECISION,NEIBPE(neib),0,&
     &          RECVbuf(ir), len_r, MPI_DOUBLE_PRECISION,NEIBPE(neib),0,&
     &          MPI_COMM_WORLD, stat1, ierr)
        enddo
!C-    update
        do neib= 1, NEIBPETOT
          do k= import_index(neib-1)+1, import_index(neib)
            kk= import_item(k)
            W(kk,P)= RECVbuf(k)
          enddo
        enddo
        do i= 1, N
          W(i,Q) = DIAG(i)*W(i,P)
          do j= INDEX(i-1)+1, INDEX(i)
            W(i,Q) = W(i,Q) + AMAT(j)*W(ITEM(j),P)
          enddo
        enddo
```

隣接PEから受信した受信バッファの値を`W(i,p)`に入れる

行列ベクトル積
`{q}=[A]{p}`

S3：一次元熱伝導方程式

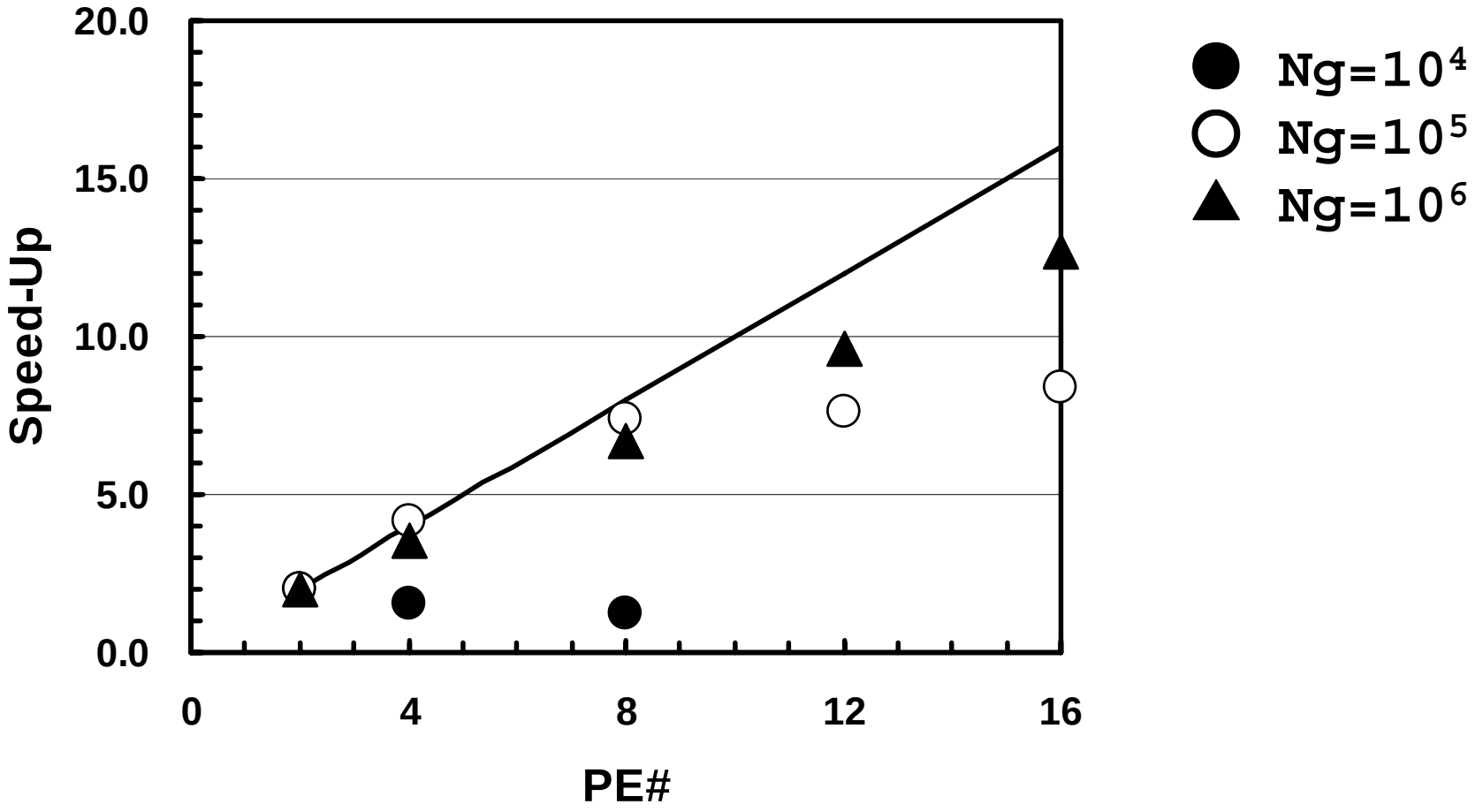
並列版（s3a.f，SEND/RECV使用）：内積（7/7）

```
!C
!C-- ALPHA= RHO / {p}{q}
      C10= 0.d0
      do i= 1, N
        C10= C10 + W(i,P)*W(i,Q)
      enddo
      call MPI_allREDUCE (C10, C1, 1, MPI_DOUBLE_PRECISION,              &
     &                    MPI_SUM, MPI_COMM_WORLD, ierr)
      ALPHA= RHO / C1

!C
!C-- {x}= {x} + ALPHA*{p}
!C   {r}= {r} - ALPHA*{q}
      do i= 1, N
        PHI(i)  = PHI(i) + ALPHA * W(i,P)
        W  (i,R)= W(i,R) - ALPHA * W(i,Q)
      enddo

      DNRM20 = 0.0
      do i= 1, N
        DNRM20= DNRM20 + W(i,R)**2
      enddo
      call MPI_allREDUCE (DNRM20, DNRM2, 1, MPI_DOUBLE_PRECISION,        &
     &                    MPI_SUM, MPI_COMM_WORLD, ierr)

        RESID= dsqrt(DNRM2/BNRM2)

        if (my_rank.eq.0.and.mod(iter,1000).eq.0) then
          write (*, '(i5,1pe16.6)') iter, RESID
        endif

        if ( RESID.le.EPS) goto 900
        RHO1 = RHO

      enddo
```

計算結果

- PE数を変えても反復回数は変化しないことを確認せよ。
- N=100,000として，1,000回反復させると
  - 2PE　　4.77 sec. (2PE/node 以下同様：ppn =2）
  - 8PE　　1.29 sec.
  - 16PE　1.14 sec.

Speed-up：1000回反復時 PE#=2の場合を「2.00」とする



前処理付き共役勾配法
Preconditioned Conjugate Gradient Method（CG）

Compute $\mathbf{r}^{(0)}= \mathbf{b}-[A]\mathbf{x}^{(0)}$
for i= 1, 2, …
  solve $[M]z^{(i-1)}= r^{(i-1)}$
  $\rho_{i-1}= \mathbf{r}^{(i-1)}\ \mathbf{z}^{(i-1)}$
  if i=1
    $p^{(1)}= z^{(0)}$
  else
    $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
    $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i)}$
  endif
  $\mathbf{q}^{(i)}= [A]\mathbf{p}^{(i)}$
  $\alpha_i = \rho_{i-1}/\mathbf{p}^{(i)}\mathbf{q}^{(i)}$
  $x^{(i)}= x^{(i-1)} + \alpha_i p^{(i)}$
  $r^{(i)}= r^{(i-1)} - \alpha_i q^{(i)}$
  check convergence $|\mathbf{r}|$
end

並列計算，領域間通信が必要な部分

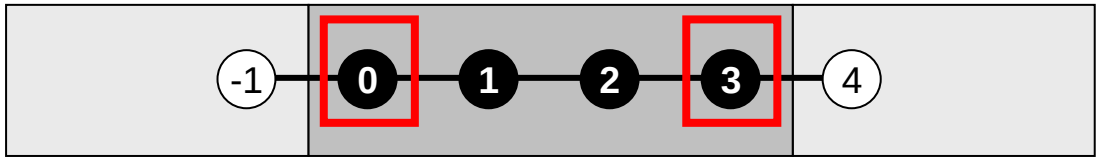
- 行列ベクトル積
- 内積

行列ベクトル積 $\mathbf{q}=[A]\mathbf{p}$ において，右辺の値は既知（計算中更新されない）ため，領域数が増加しても計算内容は変化せず，反復回数も不変である。

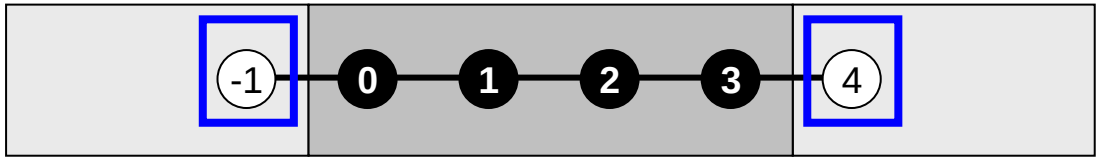
課題S2，S3からわかること

- Jacobi法，GS法，CG法を使用している差分法の並列化
  - シリアル（単独CPU）の場合とほとんど変わらない。
  - 行列ベクトル積（Jacobi法，GS法も広い意味では行列ベクトル積と同様の計算），内積に注意する程度。
  - 通信テーブルなど並列データ構造が重要
- eps_fvmの並列化も並列データ構造がカギとなるであろう。

一般化された通信テーブル：C言語
s3a.c



SENDbuf(1)=BUF(1)　　SENDbuf(2)=BUF(4)

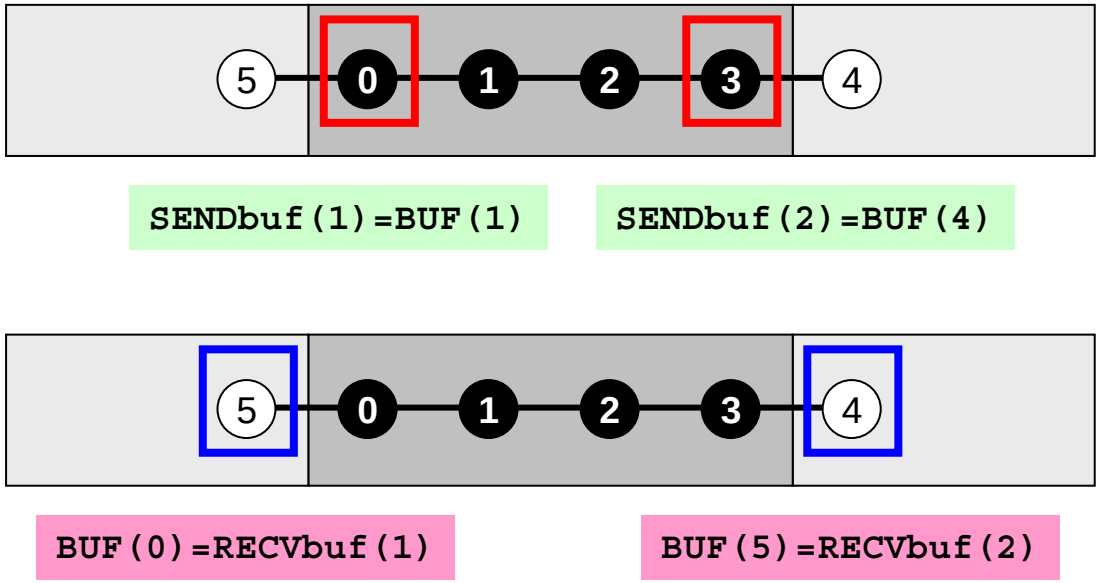


BUF(0)=RECVbuf(1)　　BUF(5)=RECVbuf(2)

C言語の場合BUF[-1]という配列は本来存在しないが，
コンパイラによっては通ってしまう場合がある。

```
NEIBPETOT= 2
NEIBPE(1)= my_rank - 1
NEIBPE(2)= my_rank + 1

import_index(1)= 1
import_index(2)= 2
import_item (1)= -1
import_item (2)= N

export_index(1)= 1
export_index(2)= 2
export_item (1)= 0
export_item (2)= N-1

if (my_rank.eq.0) then
  import_item (1)= N
  export_item (1)= N-1
  NEIBPE(1)= my_rank+1
endif
```

一般化された通信テーブル：C言語 s3b.c




```
NEIBPETOT= 2
NEIBPE(1)= my_rank - 1
NEIBPE(2)= my_rank + 1

import_index(1)= 1
import_index(2)= 2
import_item (1)= N+1
import_item (2)= N

export_index(1)= 1
export_index(2)= 2
export_item (1)= 0
export_item (2)= N-1

if (my_rank.eq.0) then
  import_item (1)= N
  export_item (1)= N-1
  NEIBPE(1)= my_rank+1
endif
```

[-1]の代わりに[N+1]を使用すれば問題は無くなる（JacobiやSORではこれがやりにくい）。