

課題S2解説

2011年夏季集中講義

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S2 : 実施内容

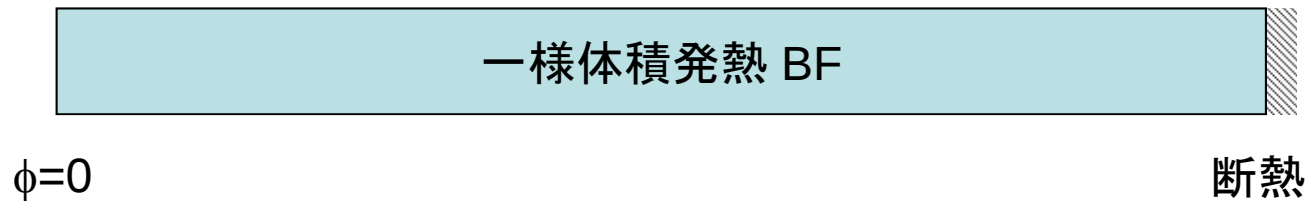
- 「<\$T-S2>/heat_cg.f」, 「<\$T-S2>/heat_cg.c」を参考にして、一次元熱伝導方程式をCG法により解くプログラムを並列化せよ。
 - 一般化された通信テーブルを導入する
- 実施課題
 - N=100およびN=103の場合について、1, 2, 4, 8CPUを使用して計算してみよ。
 - 実は、N=100程度では並列化の効果は余り・・・全く得られない
 - プロセッサ数を変えた場合、反復回数はどうなるか？
 - その理由についても検討せよ
 - Nを大きくして(例えば 10^4 , 10^5), 反復回数を100程度に固定して、プロセッサ数を変更した場合について計算を実施してみよ。

一次元熱伝導方程式(1/4)

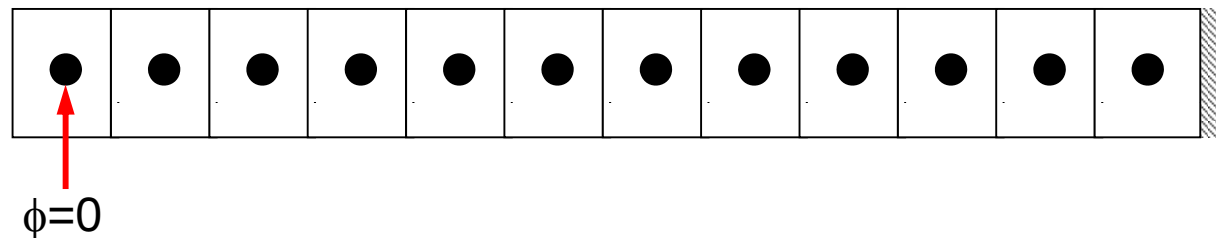
支配方程式: 熱伝導率=1(一様)

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

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



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



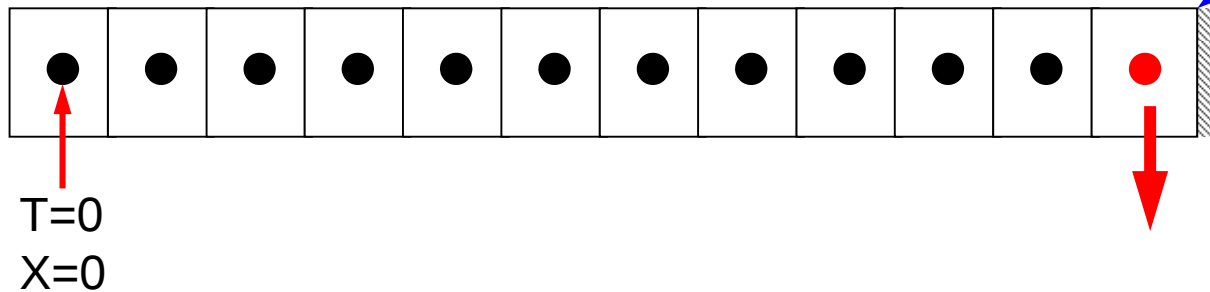
断熱となっ
ているのはこの面,
しかし温度は計算
されない

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($X=X_{\max}$)。



$\Delta x=1.d0$, 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

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



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

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

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

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

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

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

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

より一般的な形で格納すると・・・

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

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

	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							$A_L(8)$	$A_D(8)$

 $\times$

$\phi(1)$
$\phi(2)$
$\phi(3)$
$\phi(4)$
$\phi(5)$
$\phi(6)$
$\phi(7)$
$\phi(8)$

 $=$

$BF(1)$
$BF(2)$
$BF(3)$
$BF(4)$
$BF(5)$
$BF(6)$
$BF(7)$
$BF(8)$

$A_D(1) \times \phi(1) + A_R(1) \times \phi(2) = BF(1)$
 $A_L(2) \times \phi(1) + A_D(2) \times \phi(2) + A_R(2) \times \phi(3) = BF(2)$
 $A_L(3) \times \phi(2) + A_D(3) \times \phi(3) + A_R(3) \times \phi(4) = BF(3)$
 $A_L(4) \times \phi(3) + A_D(4) \times \phi(4) + A_R(4) \times \phi(5) = BF(4)$
 $A_L(5) \times \phi(4) + A_D(5) \times \phi(5) + A_R(5) \times \phi(6) = BF(5)$
 $A_L(6) \times \phi(5) + A_D(6) \times \phi(6) + A_R(6) \times \phi(7) = BF(6)$
 $A_L(7) \times \phi(6) + A_D(7) \times \phi(7) + A_R(7) \times \phi(8) = BF(7)$
 $A_L(8) \times \phi(7) + A_D(8) \times \phi(8) = BF(8)$

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

対角成分

非対角成分

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

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

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

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

DIAG(i) 対角成分(実数, $i=1, N$)
 INDEX(i) 非対角成分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*))

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

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

$\{Y\} = [A] \{X\}$

```

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		<u>1</u>						
2	<u>2</u>		<u>3</u>					
<u>3</u>		<u>4</u>		<u>5</u>				
4			<u>6</u>		<u>7</u>			
5				<u>8</u>		<u>9</u>		
6					<u>10</u>		<u>11</u>	
7						<u>12</u>		<u>13</u>
8							<u>14</u>	

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

	1	2	3	4	5	6	7	8
1	$A_D(1)$	$A_R(1)$						
2	$A_L(2)$	$A_D(2)$	$A_R(2)$					
<u>3</u>		$A_L(3)$	$A_D(3)$	$A_R(3)$				
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	<u>2</u>	3	<u>4</u>	5	6	7	8
1		<u>1</u>						
2	<u>2</u>		<u>3</u>					
<u>3</u>		<u>4</u>		<u>5</u>				
4			<u>6</u>		<u>7</u>			
5				<u>8</u>		<u>9</u>		
6					<u>10</u>		<u>11</u>	
7						<u>12</u>		<u>13</u>
8							<u>14</u>	

INDEX(2)=3, INDEX(3)=5

ITEM(4)=2, AMAT(4)= $A_L(3)$

ITEM(5)=4, AMAT(5)= $A_R(3)$

課題S2 : プログラム実行法

- 実行法

```
$ cd <$T-S2>  
$ f90 heat_cg.f  
$ ./a.out
```

```
$ cd <$T-S2>  
$ cc 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.e-7	EPS	打切誤差

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  $i = 1, 2, \dots$ 
  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
     $\beta_{i-1} = \rho_{i-1} / \rho_{i-2}$ 
     $p^{(i)} = z^{(i-1)} + \beta_{i-1} p^{(i-1)}$ 
  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

```

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

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

- **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   $d/dx(d\text{PHI}/dx) + \text{BF} = 0$ 
!C   $\text{PHI}=0 @ x=0$ 
!C
!C  program CG_poi
!C  implicit REAL*8 (A-H,O-Z)
!C
!C  integer :: N, ITERmax
!C  integer :: R, Z, P, Q, DD
!C
!C  real(kind=8) :: dx, OMEGA, RESID, dPHI, dPHImax, BF, EPS
!C  real(kind=8), dimension(:), allocatable :: PHI, RHS
!C  real(kind=8), dimension(: ), allocatable :: DIAG, AMAT
!C  real(kind=8), dimension(:, :), allocatable :: W
!C
!C  integer, dimension(: ), allocatable :: INDEX, ITEM
!C
!C-- INIT.
!C  open  (11, file='input.dat', status='unknown')
!C  read  (11, *) N
!C  read  (11, *) dX, BF
!C  read  (11, *) ITERmax
!C  read  (11, *) EPS
!C  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
```

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

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

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							$A_L(8)$	$A_D(8)$

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

$$\frac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{\Delta x} = \underbrace{\left(\frac{1}{\Delta x} \right)}_{A_L(i)} \phi_{i-1} - \underbrace{\left(\frac{2}{\Delta x} \right)}_{A_D(i)} \phi_i + \underbrace{\left(\frac{1}{\Delta x} \right)}_{A_R(i)} \phi_{i+1} = \underbrace{-BF \times \Delta x}_{RHS(i)}$$

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

	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
endif
enddo

```

	1	2	3	4	5	6	7	8
1		<u>1</u>						
2	<u>2</u>		<u>3</u>					
3		<u>4</u>		<u>5</u>				
4			<u>6</u>		<u>7</u>			
5				<u>8</u>		<u>9</u>		
6					<u>10</u>		<u>11</u>	
7						<u>12</u>		<u>13</u>
8							<u>14</u>	

```

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):後述

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

```

	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							$A_L(8)$	$A_D(8)$

境界条件($i=1$):後述

&

境界条件($i=N$):後述

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

```

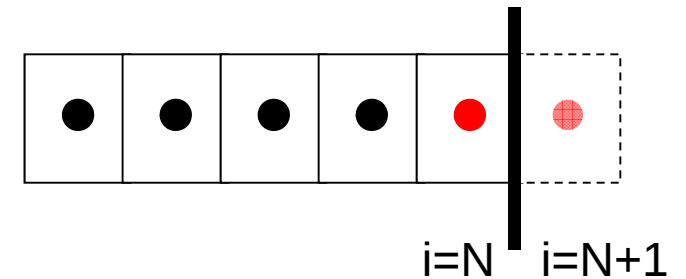
$$\frac{d\phi}{dx} = 0 @ x = x_{\max} \Rightarrow \frac{\phi_{N+1} - \phi_N}{\Delta x} = 0$$

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

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

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

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

- 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。

	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(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

- 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。

	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: $i=1$ (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

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

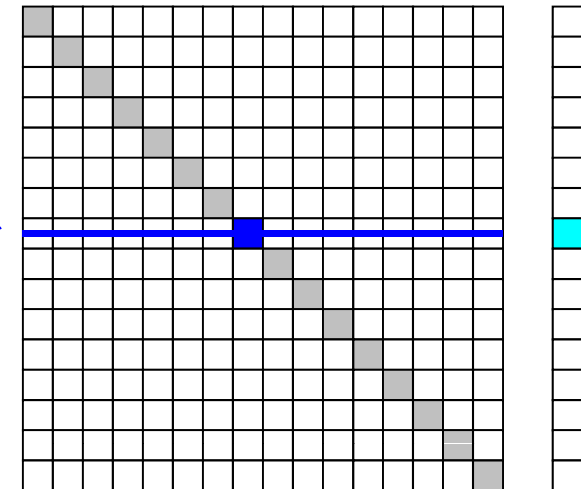
固定境界条件の処理: 消去

$i=kk$ 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

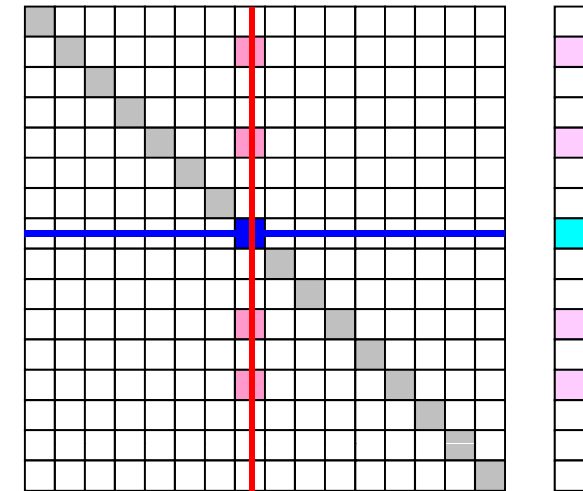
```

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

```

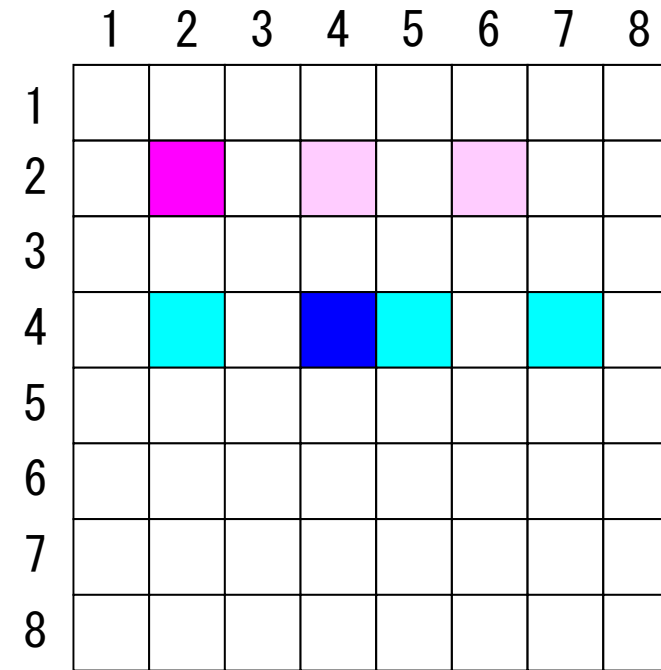
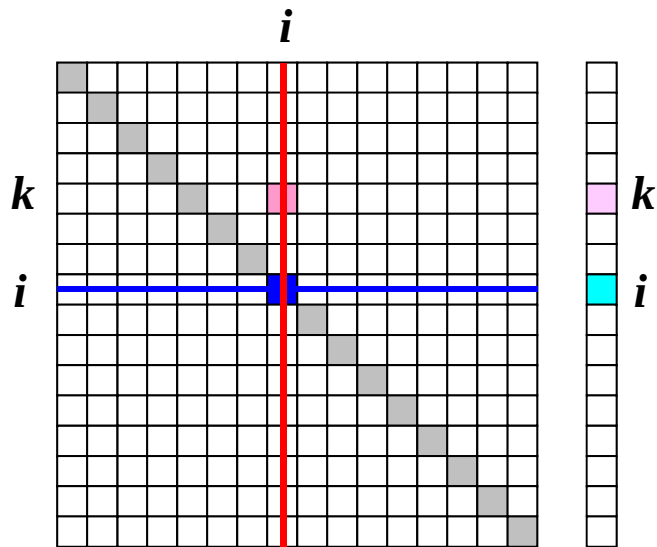
ゼロクリア



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 $\phi = \tilde{\phi}_i$



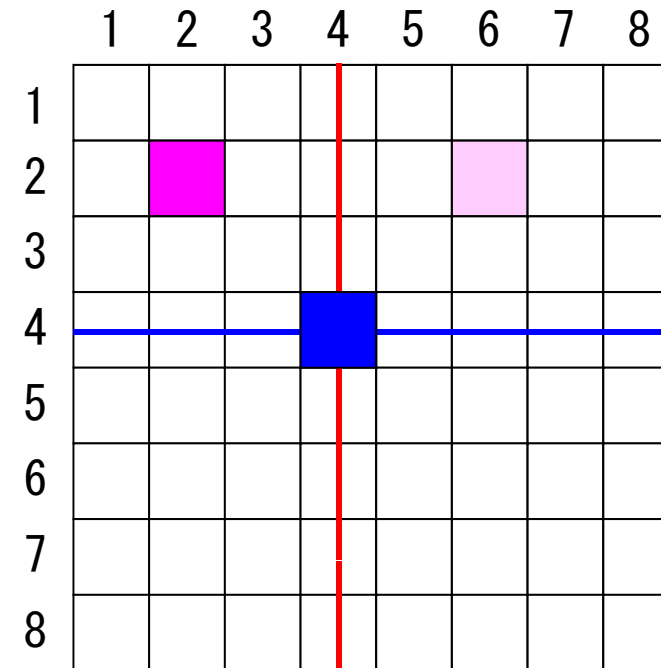
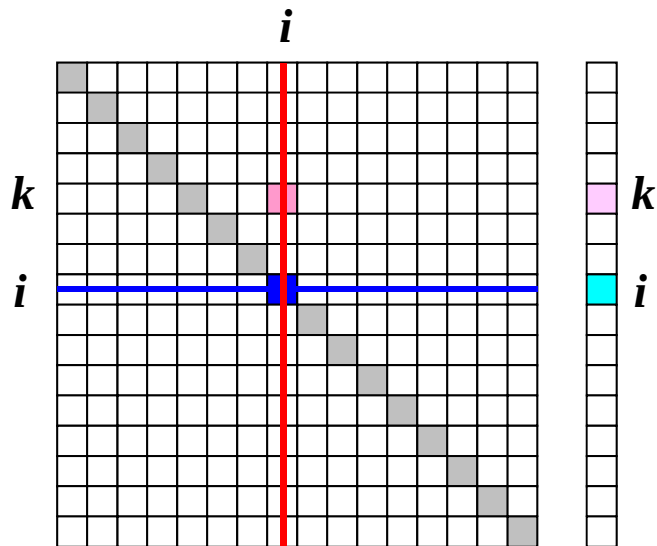
$i=4, k=2$ とすると

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

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

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

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

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

$$\begin{aligned} & \xrightarrow{\text{Red Arrow}} DIAG_2 \phi_2 + AMA_{26} \phi_6 = RHS_2 - AMAT_{24} \tilde{\phi}_4 \\ & \phi_4 = \tilde{\phi}_4 \end{aligned}$$

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
```

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

```
!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*****
```

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
```

$W(i, 1) = W(i, R) \Rightarrow \{r\}$

$W(i, 2) = W(i, Z) \Rightarrow \{z\}$

$W(i, 2) = W(i, Q) \Rightarrow \{q\}$

$W(i, 3) = W(i, P) \Rightarrow \{p\}$

$W(i, 4) = W(i, DD) \Rightarrow 1/DIAG$

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-1)
  endif
  q(i) = [A]p(i)
  αi = ρi-1 / p(i) q(i)
  x(i) = x(i-1) + αi p(i)
  r(i) = r(i-1) - αi q(i)
  check convergence |r|
end
```

前処理付き共役勾配法

Preconditioned Conjugate Gradient Method (CG)

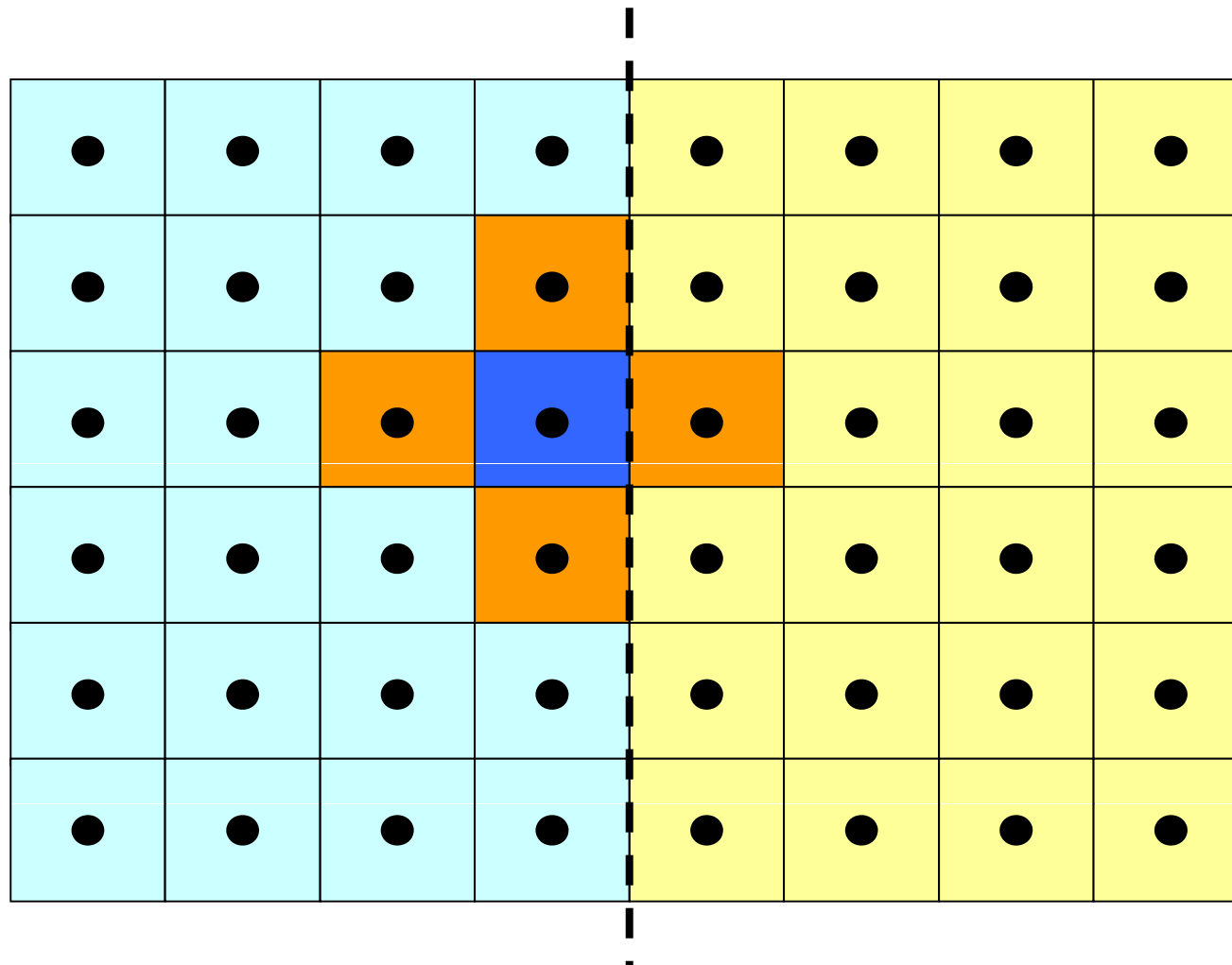
```

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

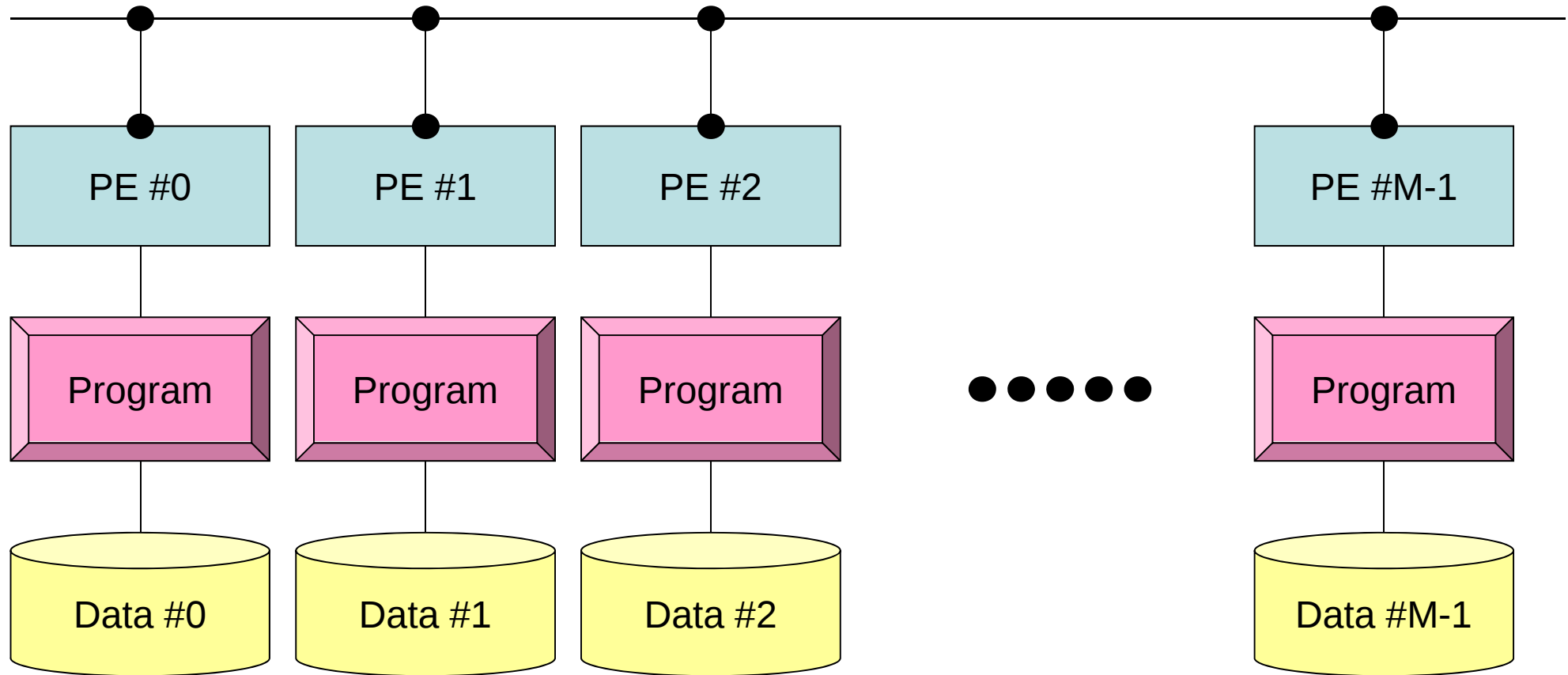
並列計算, 領域間通信が必要な部分

- 行列ベクトル積
- 内積

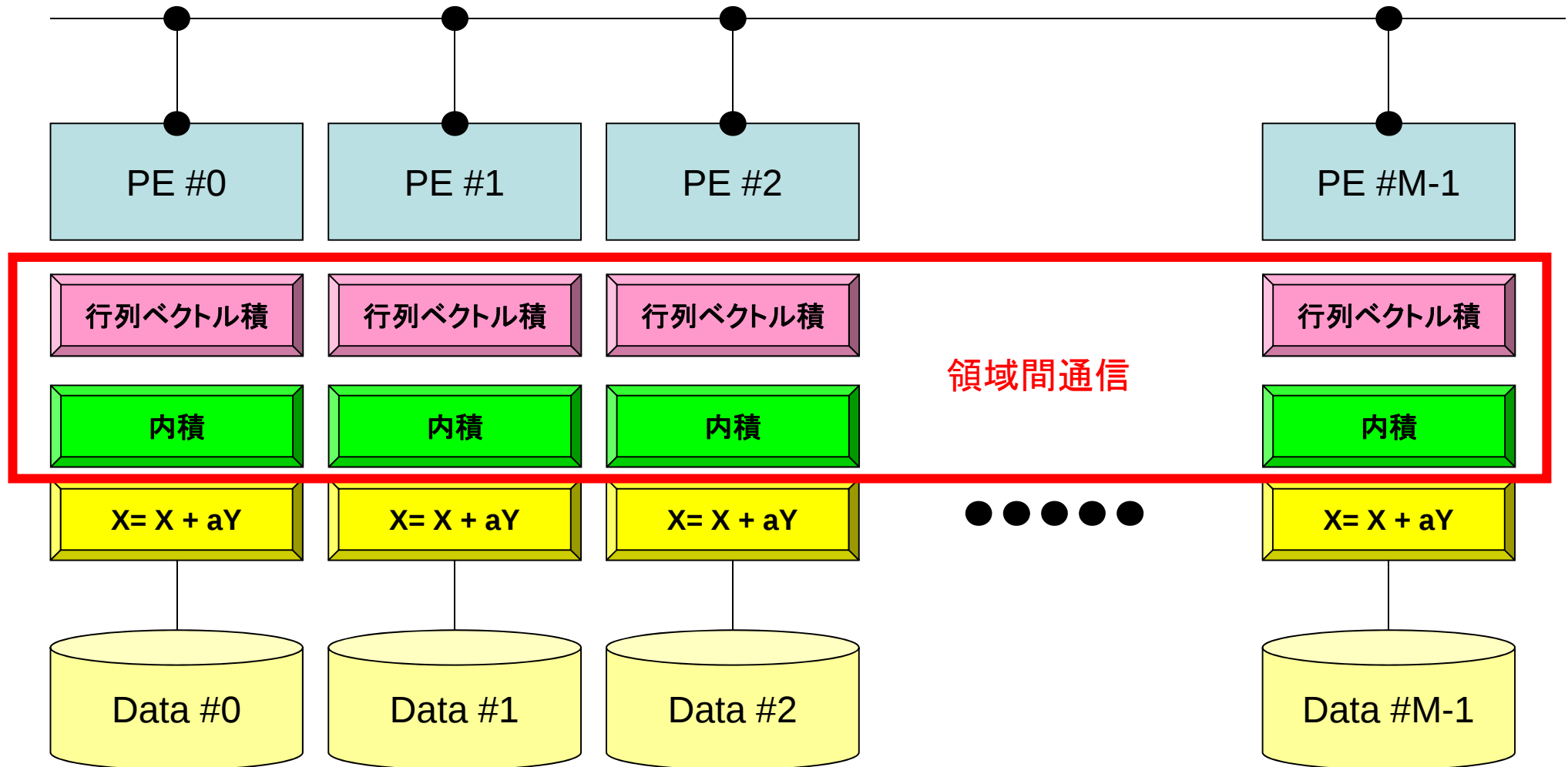
行列ベクトル積：他領域の値が必要 オーバーラップ＋通信テーブル



再びSPMD



CG法ではではこうなる



共役勾配法の「並列化」(1/2)

- 行列ベクトル積
 - 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S2の方針, ファイル

- 基本的にCE06~CE08のアプローチを踏襲

```
>$ cd <$T-EPS> 各自作成したディレクトリ
```

FORTRAN

```
>$ cp /home/t000000/EPSSummer/F/s2r-f.tar .  
>$ tar xvf s2r-f.tar
```

C

```
>$ cp /home/t000000/EPSSummer/C/s2r-c.tar .  
>$ tar xvf s2r-c.tar
```

直下に「/S2-ref」というディレクトリができている。
<\$T-EPS>/S2-refを<\$T-S2R>と呼ぶ。

ファイル

s2-a.f, heat_cg.f
s2-a.c, heat_cg.c

input.dat
go.sh

mpif90 -Oss -noparallel s2-a.f
mpicc -Os -noparallel s2-a.c
<modify "go.sh", "go2.sh">
qsub go.sh, go2.sh

go.sh

```
#@$-r test
#@$-q lecture
#@$-N 1
#@$-J T4
#@$-e err
#@$-o test.lst
#@$-lM 28GB
#@$-lE 00:05:00
#@$-s /bin/sh
#@$
```

```
cd $PBS_O_WORKDIR
```

```
mpirun numactl -localalloc ./a.out
exit
```

go2.sh

```
#@$-r test
#@$-q lecture
#@$-N 1
#@$-J T4
#@$-e err
#@$-o test.lst
#@$-lM 28GB
#@$-lE 00:05:00
#@$-s /bin/sh
#@$
```

```
cd $PBS_O_WORKDIR
```

```
mpirun ./numarun.sh ./a.out
exit
```

numarun.sh : Policy-4相当

```
#!/bin/bash
MYRANK=$MXMPI_ID
MYVAL=$(expr $MYRANK / 4)
SOCKET=$(expr $MYVAL % 4)
numactl --cpunodebind=$SOCKET --localalloc $@
```

go2.sh: 4 cores

```
#@$-r test  
#@$-q lecture  
#@$-N 1  
#@$-J T4  
...
```

go2.sh: 16 cores

```
#@$-r test  
#@$-q lecture  
#@$-N 1  
#@$-J T16  
...
```

go2.sh: 32 cores

```
#@$-r test  
#@$-q lecture  
#@$-N 2  
#@$-J T16  
...
```

go2.sh: 16 cores

```
#@$-r test  
#@$-q lecture  
#@$-N 2  
#@$-J T8  
...
```

go2.sh: 32 cores

```
#@$-r test  
#@$-q lecture  
#@$-N 4  
#@$-J T8  
...
```

go2.sh: 64 cores

```
#@$-r test  
#@$-q lecture  
#@$-N 4  
#@$-J T16  
...
```

S2: 一次元熱伝導方程式並列版(1/8)

```

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 :: stat_send
integer(kind=4), dimension(:,:), allocatable :: stat_recv
integer(kind=4), dimension(:), allocatable :: request_send
integer(kind=4), dimension(:), allocatable :: request_recv

!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 )

```

S2: 一次元熱伝導方程式並列版(2/8)

```

!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

```

S2: 一次元熱伝導方程式並列版(2/8)

```
!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
```

S2: 一次元熱伝導方程式並列版(3/8)

```

!C
!C-- MATRIX
      allocate (PHI(N+2), DIAG(N), AMAT(2*N), RHS(N))
      allocate (INDEX(0:N), ITEM(2*N), W(N+2,4))
      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							$A_L(8)$	$A_D(8)$

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

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

 A_L
 A_D
 A_R
 RHS

S2: 一次元熱伝導方程式並列版 (3/8)

```
!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

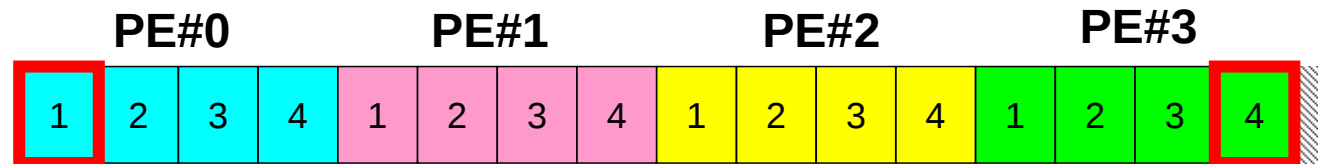
!C
!C-- CONNECTIVITY
  INDEX(0)= 0

  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」



S2: 一次元熱伝導方程式並列版(4/8)

```

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.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.eq.1) ITEM(jS+1)= N+1
    if (i.eq.N) ITEM(jS+2)= N+2
    if (my_rank.eq.0.and.i.eq.N) ITEM(jS+2)= N+1

    if (my_rank.eq.0.and.i-1.eq.1) then
      AMAT(jS+1)= 0.d0
    endif
  endif
endif
enddo

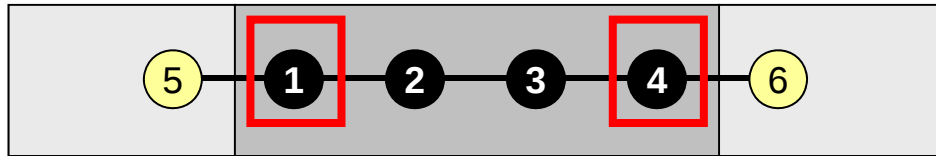
```

境界条件(Xmin)

境界条件(Xmax)

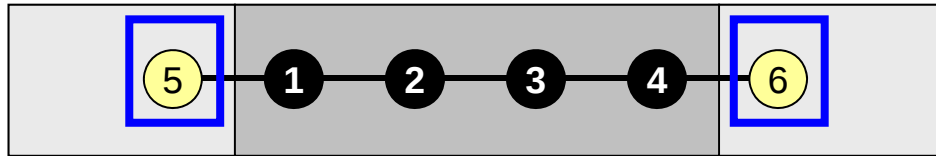
それ以外

一般化された通信テーブル



SENDbuf(1)=BUF(1)

SENDbuf(2)=BUF(4)



BUF(5)=RECVbuf(1)

BUF(6)=RECVbuf(2)

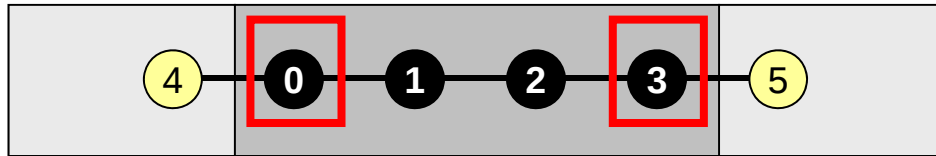
```
NEIBPETOT= 2
NEIBPE(1)= my_rank - 1
NEIBPE(2)= my_rank + 1
```

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

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

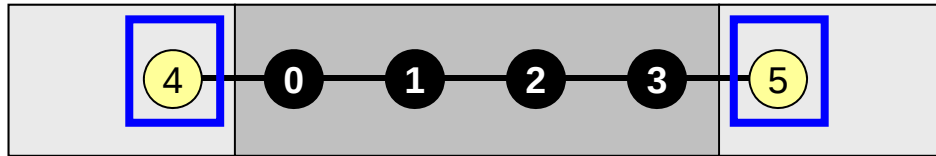
```
if (my_rank.eq.0) then
  import_item (1)= N+1
  export_item (1)= N
  NEIBPE(1)= my_rank+1
endif
```

一般化された通信テーブル:C言語



SENDbuf[0]=BUF[0]

SENDbuf[1]=BUF[3]



BUF[4]=RECVbuf[0]

BUF[5]=RECVbuf[1]

```
NEIBPETOT= 2
NEIBPE[0]= my_rank - 1
NEIBPE[1]= my_rank + 1
```

```
import_index[1]= 0
import_index[2]= 1
import_item [0]= N
import_item [1]= N+1
```

```
export_index[1]= 0
export_index[2]= 1
export_item [0]= 0
export_item [1]= N-1
```

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

S2: 一次元熱伝導方程式並列版(4/8)

```

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.N) then
    ITEM(jS+1)= i-1
    DIAG(i)   = -1.d0/dX
  else
    ITEM(jS+1)= i-1
    ITEM(jS+2)= i+1

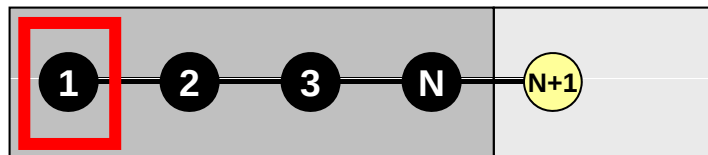
    if (i.eq.1) ITEM(jS+1)= N+1
    if (i.eq.N) ITEM(jS+2)= N+2
    if (my_rank.eq.0.and.i.eq.N) ITEM(jS+2)= N+1

    if (my_rank.eq.0.and.i-1.eq.1) then
      AMAT(jS+1)= 0.d0
    endif
  endif
endif
enddo

```

境界条件(Xmin)

&



S2: 一次元熱伝導方程式並列版(4/8)

```

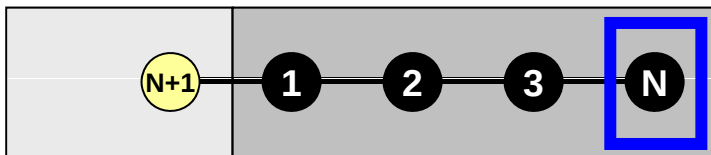
do i= 1, N
  jS= INDEX(i-1)
  if (my_rank.eq.0 .and. i.eq.1) then
    ITEM(jS+1)= i+1
    AMAT(jS+1)= 0.d0
    DIAG(i    )= 1.d0
    RHS(i     )= 0.d0
  else if
    & (my_rank.eq.PETOT-1 .and. i.eq.N) then
    ITEM(jS+1)= i-1
    DIAG(i    )= -1.d0/dx
  else
    ITEM(jS+1)= i-1
    ITEM(jS+2)= i+1

    if (i.eq.1) ITEM(jS+1)= N+1
    if (i.eq.N) ITEM(jS+2)= N+2
    if (my_rank.eq.0.and.i.eq.N) ITEM(jS+2)= N+1

    if (my_rank.eq.0.and.i-1.eq.1) then
      AMAT(jS+1)= 0.d0
    endif
  endif
endif
enddo

```

境界条件(Xmax)



S2: 一次元熱伝導方程式並列版(4/8)

```

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.N) then
    ITEM(jS+1)= i-1
    DIAG(i    )= -1.d0/dX
  else
    ITEM(jS+1)= i-1
    ITEM(jS+2)= i+1

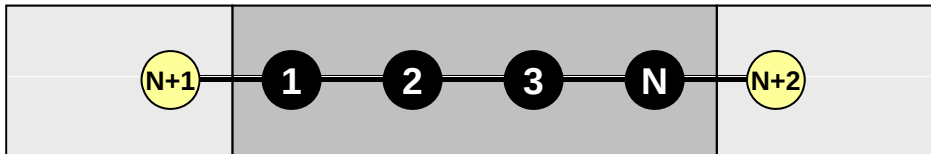
    if (i.eq.1) ITEM(jS+1)= N+1
    if (i.eq.N) ITEM(jS+2)= N+2
    if (my_rank.eq.0.and.i.eq.N) ITEM(jS+2)= N+1

    if (my_rank.eq.0.and.i-1.eq.1) then
      AMAT(jS+1)= 0.d0
    endif
  endif
endif
enddo

```

&

それ以外



S2: 一次元熱伝導方程式並列版(4/8)

```

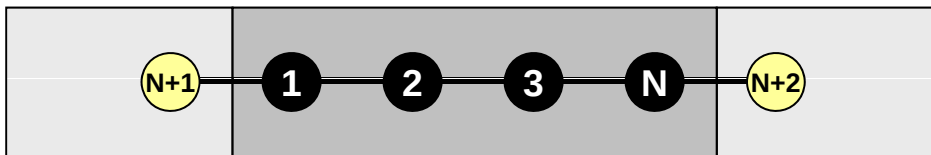
do i= 1, N
  jS= INDEX(i-1)
  if (my_rank.eq.0 .and. i.eq.1) then
    ITEM(jS+1)= i+1
    AMAT(jS+1)= 0.d0
    DIAG(i    )= 1.d0
    RHS(i    )= 0.d0
  else if
&    (my_rank.eq.PETOT-1 .and. i.eq.N) then
    ITEM(jS+1)= i-1
    DIAG(i    )= -1.d0/dX
  else
    ITEM(jS+1)= i-1
    ITEM(jS+2)= i+1

    if (i.eq.1) ITEM(jS+1)= N+1
    if (i.eq.N) ITEM(jS+2)= N+2
    if (my_rank.eq.0.and.i.eq.N) ITEM(jS+2)= N+1

    if (my_rank.eq.0.and.i-1.eq.1) then
      AMAT(jS+1)= 0.d0
    endif
  endif
endif
enddo

```

それ以外



S2: 一次元熱伝導方程式並列版(4/8)

```

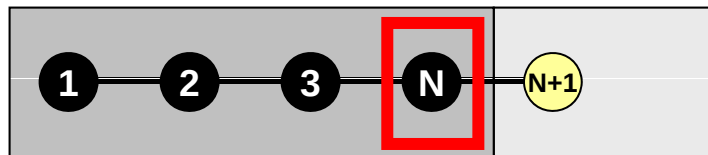
do i= 1, N
  jS= INDEX(i-1)
  if (my_rank.eq.0 .and. i.eq.1) then
    ITEM(jS+1)= i+1
    AMAT(jS+1)= 0.d0
    DIAG(i    )= 1.d0
    RHS(i    )= 0.d0
  else if
&    (my_rank.eq.PETOT-1 .and. i.eq.N) then
    ITEM(jS+1)= i-1
    DIAG(i    )= -1.d0/dX
  else
    ITEM(jS+1)= i-1
    ITEM(jS+2)= i+1

    if (i.eq.1) ITEM(jS+1)= N+1
    if (i.eq.N) ITEM(jS+2)= N+2
    if (my_rank.eq.0.and.i.eq.N) ITEM(jS+2)= N+1

    if (my_rank.eq.0.and.i-1.eq.1) then
      AMAT(jS+1)= 0.d0
    endif
  endif
endif
enddo

```

それ以外



S2: 一次元熱伝導方程式並列版(5/8)

```

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

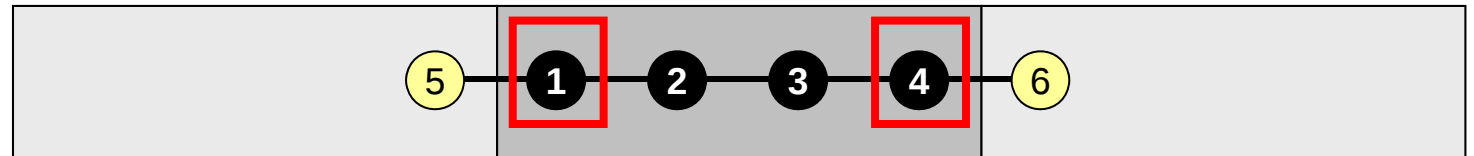
  export_index(1)= 1
  export_index(2)= 2
  export_item (1)= 1
  export_item (2)= N

  if (my_rank.eq.0) then
    import_item (1)= N+1
    export_item (1)= N
  endif
!C
!C-- INIT. arrays for MPI_WAITALL
  allocate (stat_send(MPI_STATUS_SIZE,NEIBPETOT))
  allocate (stat_recv(MPI_STATUS_SIZE,NEIBPETOT))
  allocate (request_send(NEIBPETOT))
  allocate (request_recv(NEIBPETOT))

```

1D差分法の並列計算に必要な情報 新しい局所番号付け＋通信テーブル

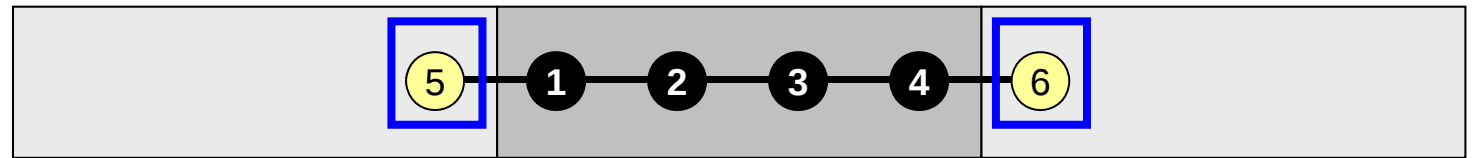
隣接PEへの
送信(境界点)



SENDbuf(1)=BUF(1)

SENDbuf(2)=BUF(4)

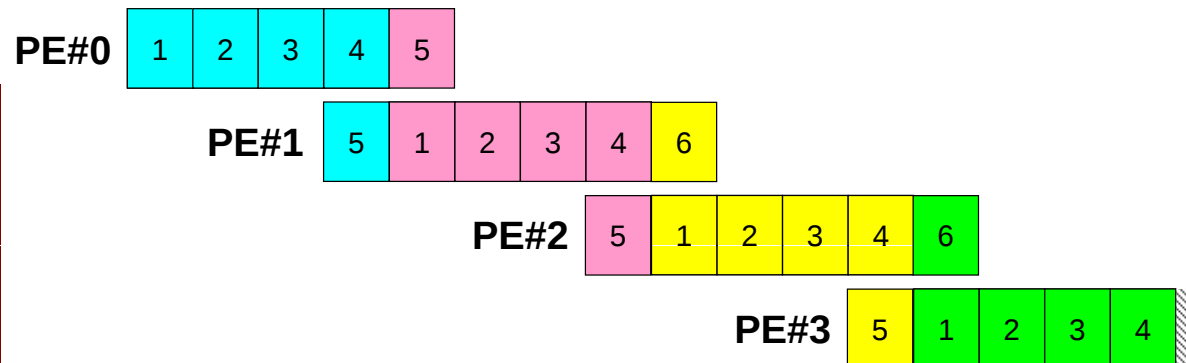
隣接PEからの
受信(外点)



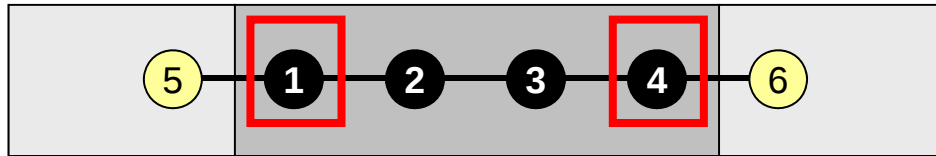
BUF(0)=RECVbuf(1)

BUF(5)=RECVbuf(2)

多次元への拡張を
考えるとこのような
局所番号付けが自然
(0~N+1)⇒(1~N+2)

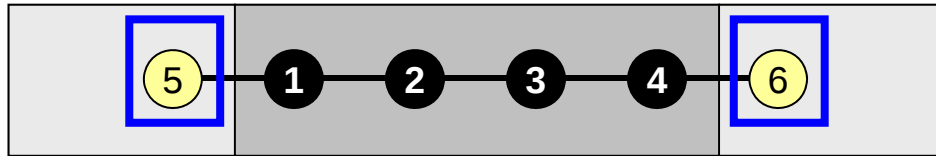


一般化された通信テーブル



SENDbuf(1)=BUF(1)

SENDbuf(2)=BUF(4)



BUF(5)=RECVbuf(1)

BUF(6)=RECVbuf(2)

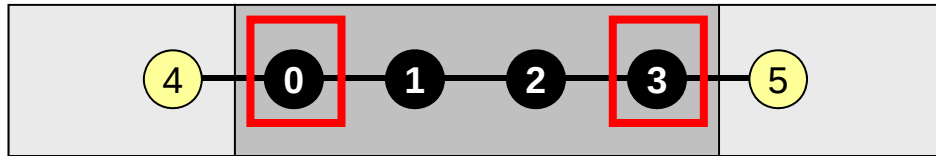
```
NEIBPETOT= 2
NEIBPE(1)= my_rank - 1
NEIBPE(2)= my_rank + 1
```

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

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

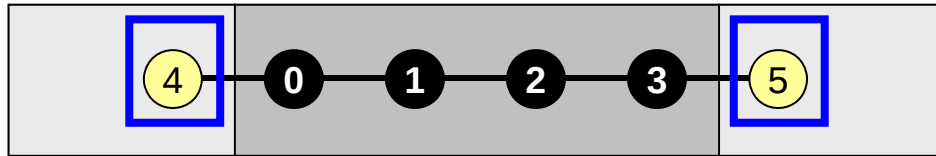
```
if (my_rank.eq.0) then
  import_item (1)= N+1
  export_item (1)= N
  NEIBPE(1)= my_rank+1
endif
```

一般化された通信テーブル:C言語



SENDbuf[0]=BUF[0]

SENDbuf[1]=BUF[3]



BUF[4]=RECVbuf[0]

BUF[5]=RECVbuf[1]

```
NEIBPETOT= 2
NEIBPE[0]= my_rank - 1
NEIBPE[1]= my_rank + 1
```

```
import_index[1]= 0
import_index[2]= 1
import_item [0]= N
import_item [1]= N+1
```

```
export_index[1]= 0
export_index[2]= 1
export_item [0]= 0
export_item [1]= N-1
```

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

S2: 一次元熱伝導方程式並列版(6/8)

一般化された通信テーブルによる行列ベクトル積

```

!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
        len_s= export_index(neib) - export_index(neib-1)
        call MPI_ISEND (SENDbuf(is), len_s, MPI_DOUBLE_PRECISION,
&                                NEIBPE(neib), 0, MPI_COMM_WORLD,
&                                request_send(neib), ierr)
&
&      enddo

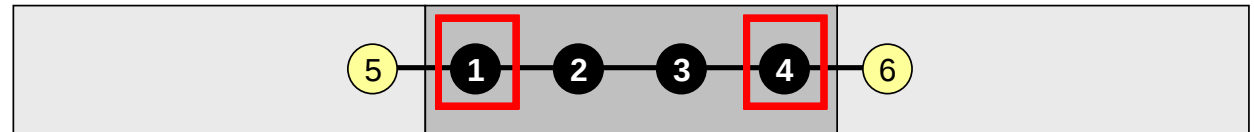
      do neib= 1, NEIBPETOT
        ir = import_index(neib-1) + 1
        len_r= import_index(neib) - import_index(neib-1)
        call MPI_Irecv (RECVbuf(ir), len_r, MPI_DOUBLE_PRECISION,
&                                NEIBPE(neib), 0, MPI_COMM_WORLD,
&                                request_recv(neib), ierr)
&
&      enddo
      call MPI_WAITALL (NEIBPETOT, request_recv, stat_recv, ierr)

!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
      call MPI_WAITALL (NEIBPETOT, request_send, stat_send, ierr)

```

送信バッファに
W(i,p) の値を入れる

送信:一次元問題



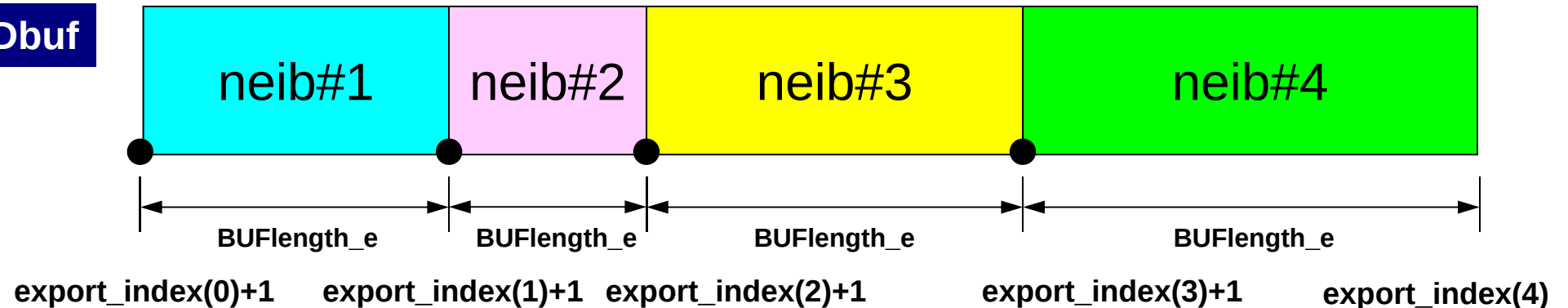
SENDbuf(1)=BUF(1)

SENDbuf(2)=BUF(4)

- 送信相手
 - NEIBPETOT, NEIBPE(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= 0, 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)

SENDbuf



```
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。

S2: 一次元熱伝導方程式並列版(6/8)

一般化された通信テーブルによる行列ベクトル積

```

!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

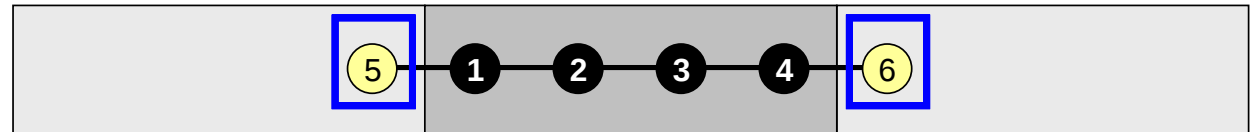
!C
!C-- SEND & RECV.
      do neib= 1, NEIBPETOT
        is  = export_index(neib-1) + 1
        len_s= export_index(neib) - export_index(neib-1)
        call MPI_ISEND (SENDbuf(is), len_s, MPI_DOUBLE_PRECISION,      &
&                      NEIBPE(neib), 0, MPI_COMM_WORLD,                &
&                      request_send(neib), ierr)
      enddo

      do neib= 1, NEIBPETOT
        ir  = import_index(neib-1) + 1
        len_r= import_index(neib) - import_index(neib-1)
        call MPI_IRECV (RECVbuf(ir), len_r, MPI_DOUBLE_PRECISION,      &
&                      NEIBPE(neib), 0, MPI_COMM_WORLD,                &
&                      request_recv(neib), ierr)
      enddo
      call MPI_WAITALL (NEIBPETOT, request_recv, stat_recv, ierr)

!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
      call MPI_WAITALL (NEIBPETOT, request_send, stat_send, ierr)

```

受信:一次元問題



BUF(0)=RECVbuf(1)

BUF(5)=RECVbuf(2)

- 受信相手
 - NEIBPETOT, NEIBPE(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= 0, 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)= N+1, import_item(2)= N+2
- 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(N+1)=RECVbuf(1), BUF(N+2)=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

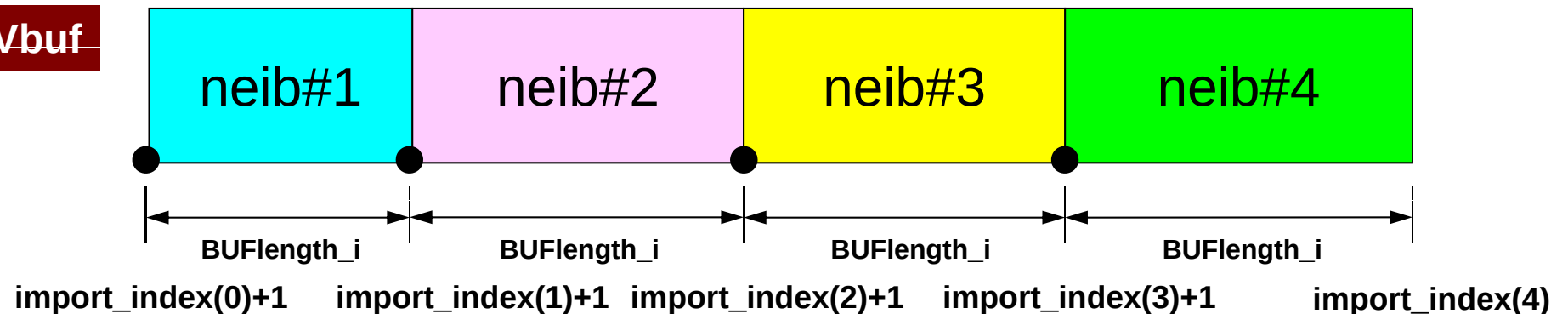
call MPI_WAITALL (NEIBPETOT, request_recv, stat_recv, ierr)

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

```

受信バッファから代入

RECVbuf



S2: 一次元熱伝導方程式並列版(6/8)

一般化された通信テーブルによる行列ベクトル積

```

!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
        len_s= export_index(neib) - export_index(neib-1)
        call MPI_ISEND (SENDbuf(is), len_s, MPI_DOUBLE_PRECISION,      &
&                                NEIBPE(neib), 0, MPI_COMM_WORLD,      &
&                                request_send(neib), ierr)
      enddo

      do neib= 1, NEIBPETOT
        ir = import_index(neib-1) + 1
        len_r= import_index(neib) - import_index(neib-1)
        call MPI_IRECV (RECVbuf(ir), len_r, MPI_DOUBLE_PRECISION,      &
&                                NEIBPE(neib), 0, MPI_COMM_WORLD,      &
&                                request_recv(neib), ierr)
      enddo
      call MPI_WAITALL (NEIBPETOT, request_recv, stat_recv, ierr)

!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
      call MPI_WAITALL (NEIBPETOT, request_send, stat_send, ierr)

```

S2: 一次元熱伝導方程式並列版(6/8)

一般化された通信テーブルによる行列ベクトル積

```

!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
        len_s= export_index(neib) - export_index(neib-1)
        call MPI_ISEND (SENDbuf(is), len_s, MPI_DOUBLE_PRECISION,
&                      NEIBPE(neib), 0, MPI_COMM_WORLD,
&                      request_send(neib), ierr)
&
&
      enddo

      do neib= 1, NEIBPETOT
        ir = import_index(neib-1) + 1
        len_r= import_index(neib) - import_index(neib-1)
        call MPI_IRECV (RECVbuf(ir), len_r, MPI_DOUBLE_PRECISION,
&                      NEIBPE(neib), 0, MPI_COMM_WORLD,
&                      request_recv(neib), ierr)
&
&
      enddo
      call MPI_WAITALL (NEIBPETOT, request_recv, stat_recv, ierr)

!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
      call MPI_WAITALL (NEIBPETOT, request_send, stat_send, ierr)

```

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

S2: 一次元熱伝導方程式並列版(7/8)

一般化された通信テーブルによる行列ベクトル積

```
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
```

行列ベクトル積
 $\{q\}=[A]\{p\}$

S2: 一次元熱伝導方程式並列版(8/8)

内積

```

!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
      RH01 = 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。

前処理付き共役勾配法

Preconditioned Conjugate Gradient Method (CG)

```

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

```

並列計算, 領域間通信が必要な部分

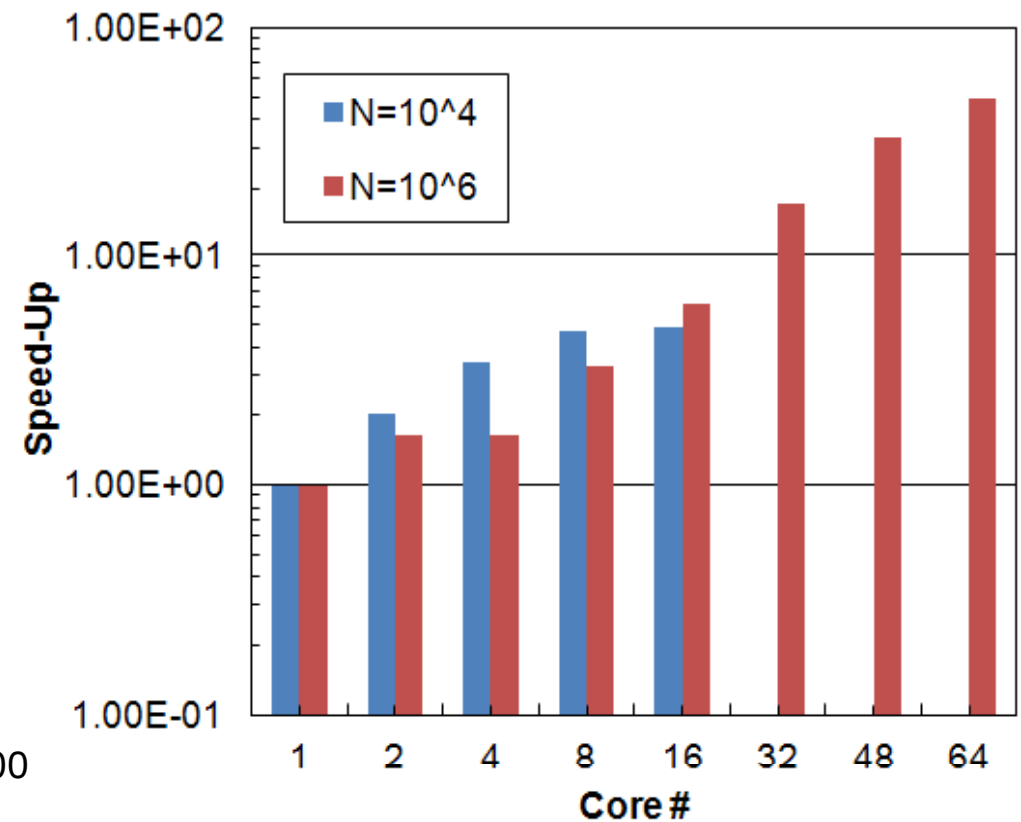
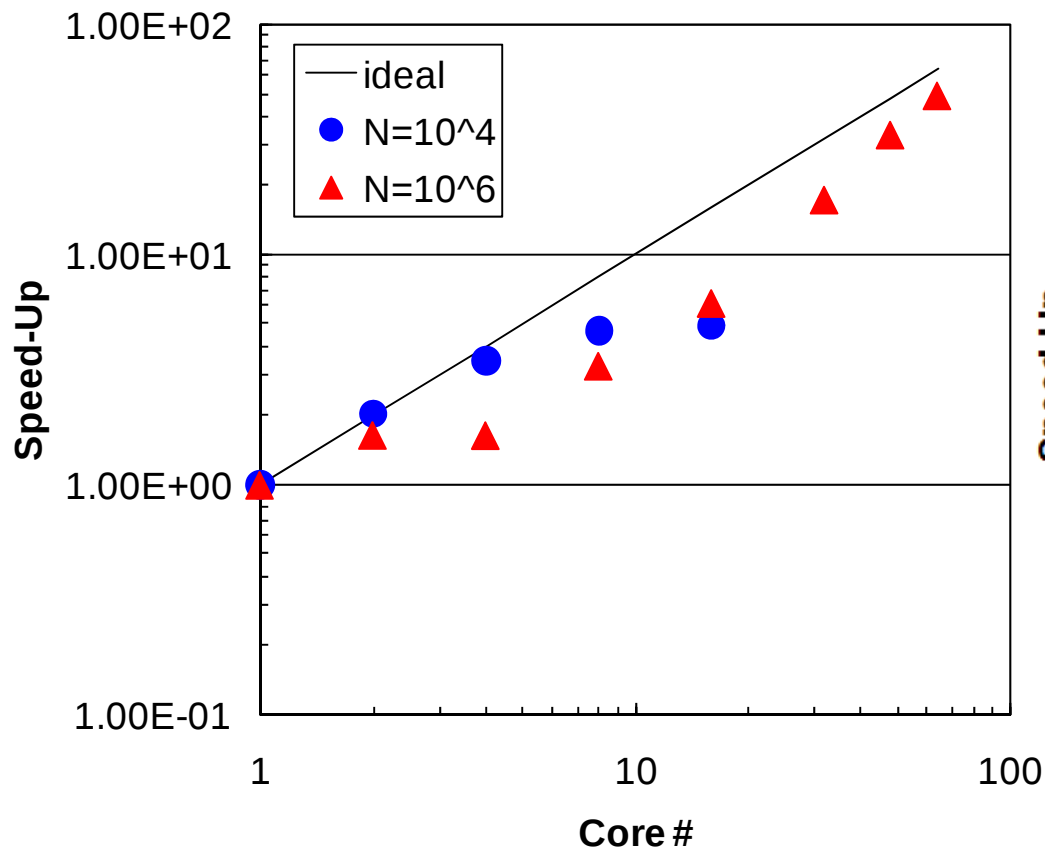
- 行列ベクトル積
- 内積

行列ベクトル積 $\mathbf{q} = [\mathbf{A}]\mathbf{p}$ 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。

並列性能: T2K(東大) go2.sh (Policy-4)

Strong Scaling: 1コア性能を基準

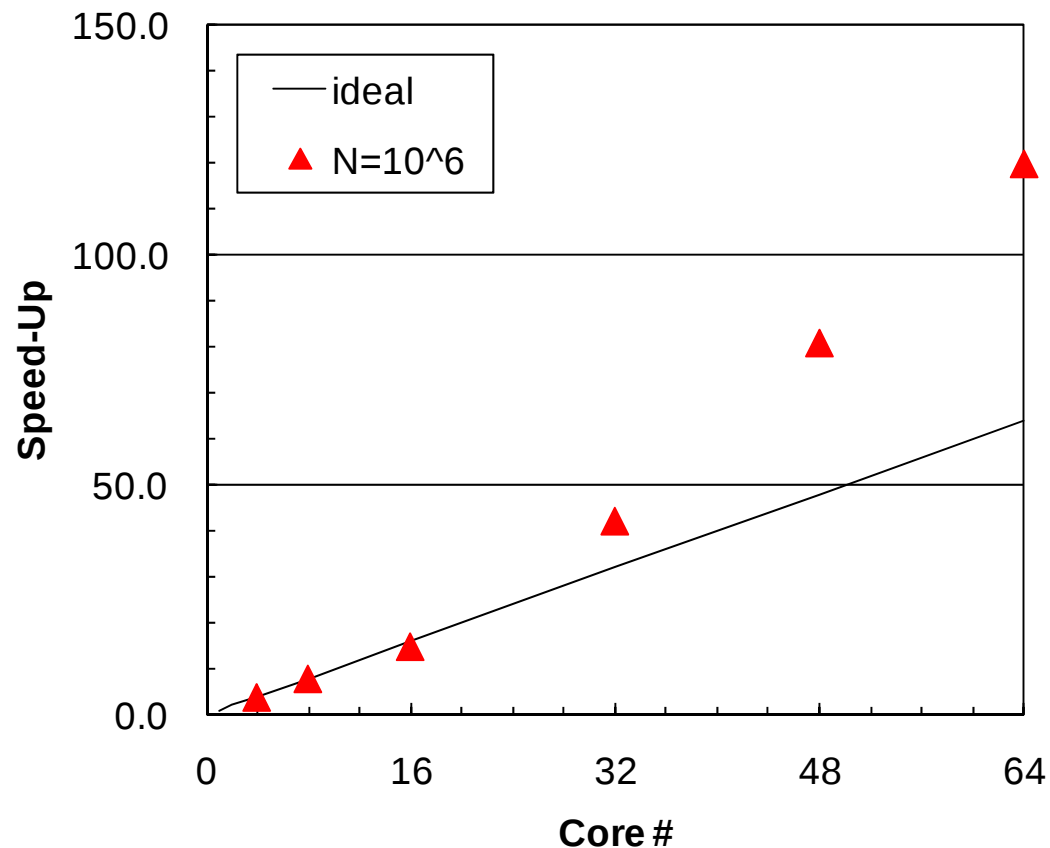
$N=10^6$: 1,000回の反復時間



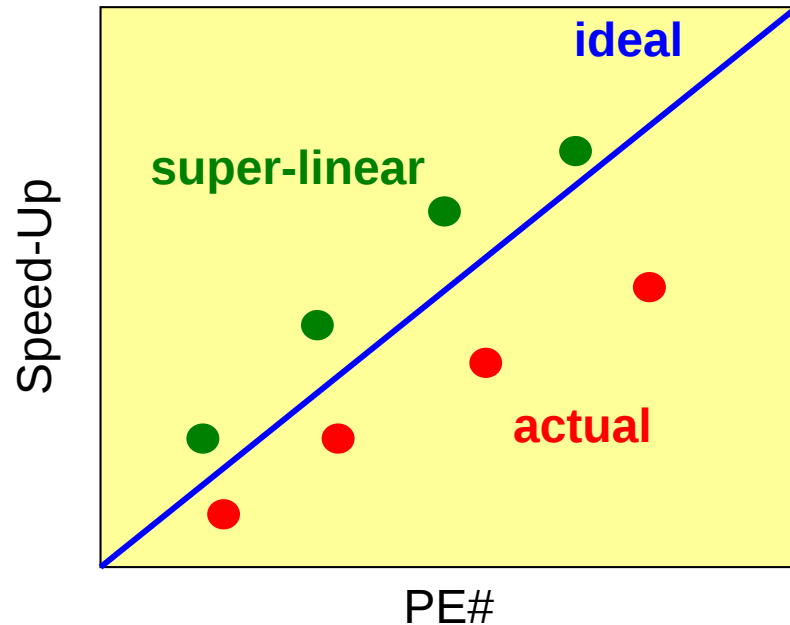
並列性能: T2K(東大) go2.sh (Policy-4)

Strong Scaling: 4コア(1ソケット)性能を基準

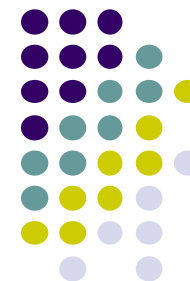
$N=10^6$: 1,000回の反復時間



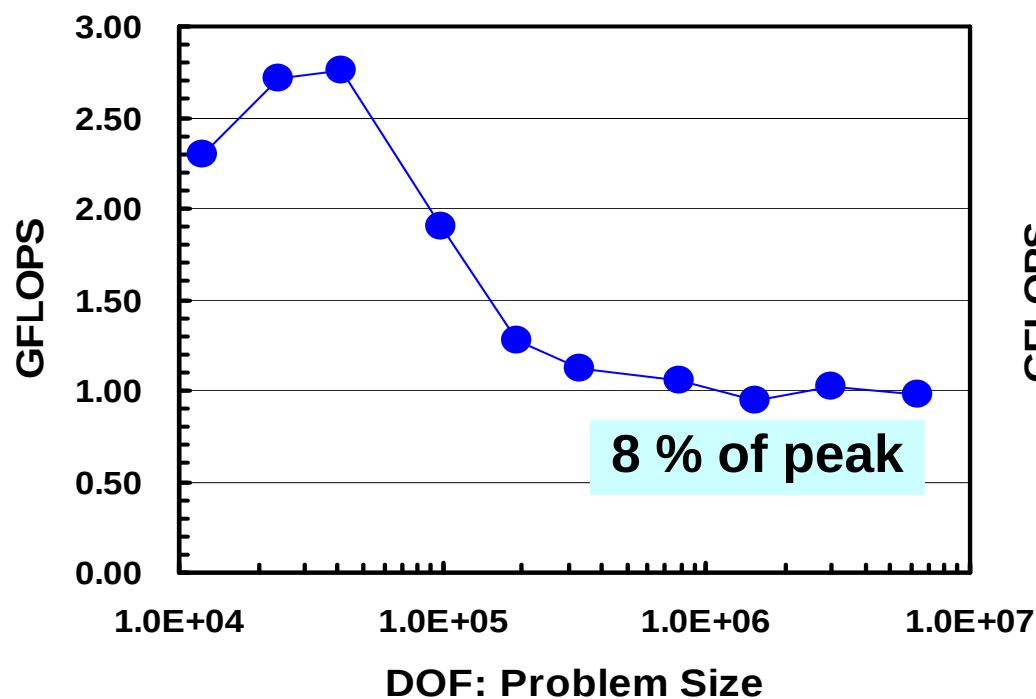
Strong-Scalingにおける「Super-Linear」



- 問題規模を固定して、使用PE数を増加させて行った場合、通常は通信の影響のために、効率は理想値(m 個のPEを使用した場合、理想的には m 倍の性能になる)よりも低くなるのが普通である。
- しかし、スカラプロセッサ(PC等)の場合、逆に理想値よりも、高い性能が出る場合がある。このような現象を「Super-Linear」と呼ぶ。
 - ベクトル計算機では起こらない。

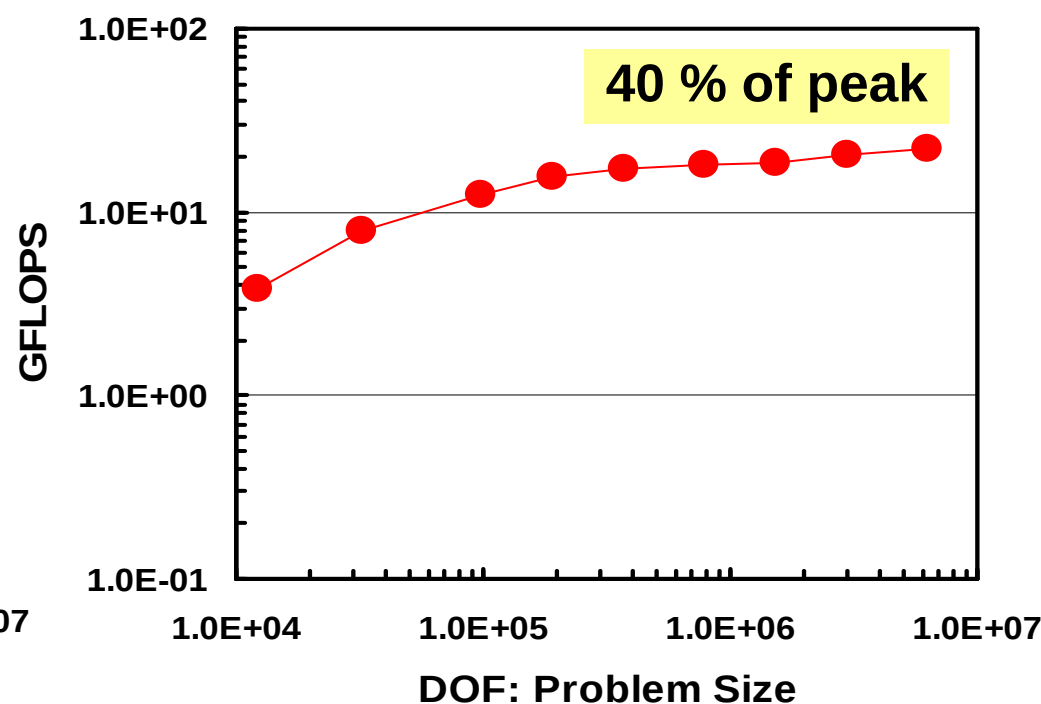


典型的な挙動



IBM-SP3:

問題サイズが小さい場合はキャッシュの影響のため性能が良い

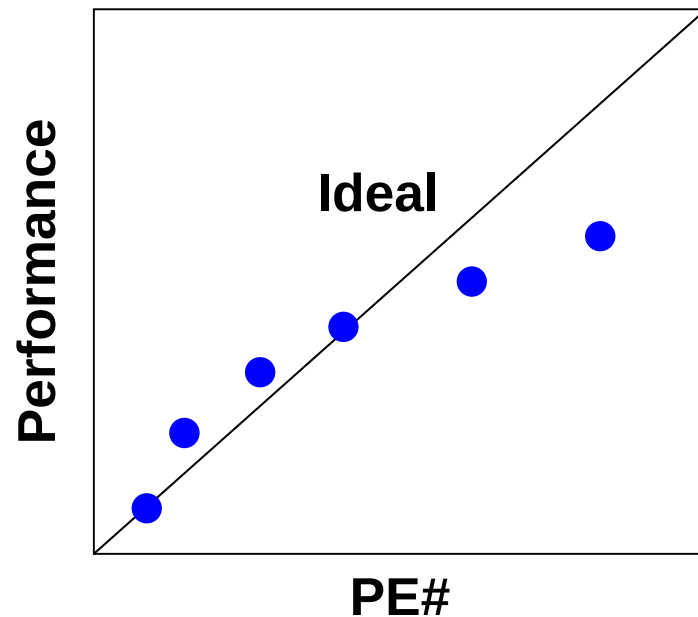


Earth Simulator:

大規模な問題ほどベクトル長が長くなり、性能が高い

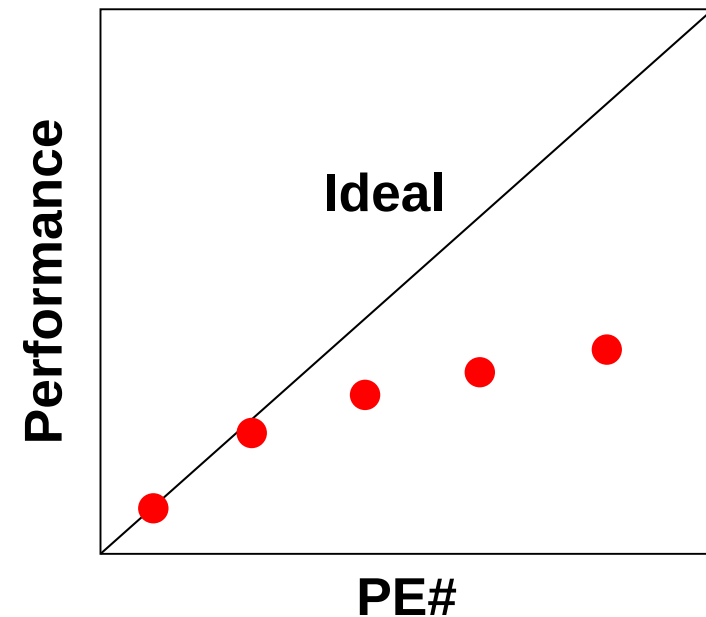
並列計算

Strong Scaling (全体問題規模固定)



IBM-SP3:

PE (Processing Element) 数が少ない場合はいわゆるスーパースカラー。
PE数が増加すると通信オーバーヘッドのため性能低下。

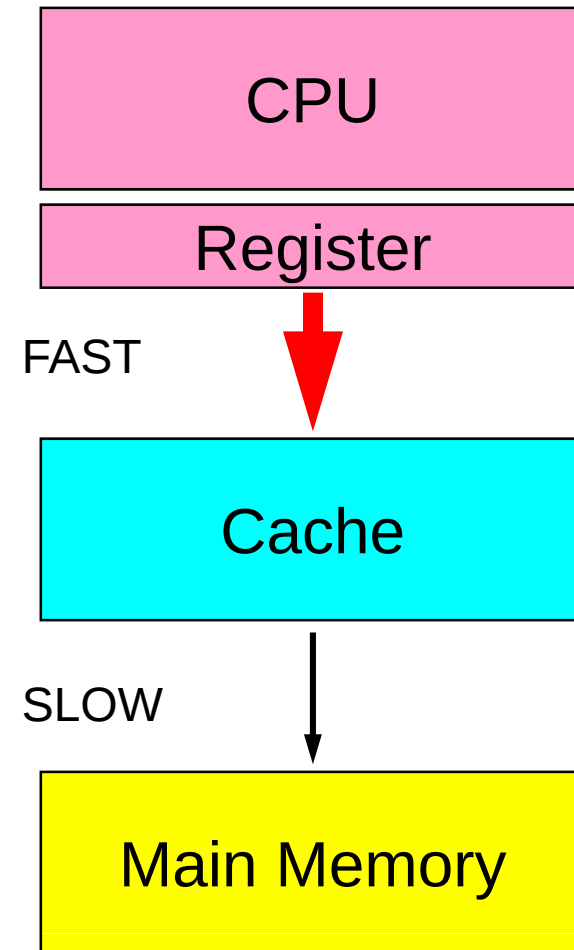


Earth Simulator:

PE数が増加すると、通信オーバーヘッドに加え、PEあたりの問題規模が小さくなるため性能低下。

Super-Linearの生じる理由

- キャッシュの影響
- スカラープロセッサでは、全般に問題規模が小さいほど性能が高い。
 - キャッシュの有効利用



疎行列ソルバーの性能：三次元弾性問題

ICCG法, T2K・SR11000 1ノード: メモリバンド幅が効く

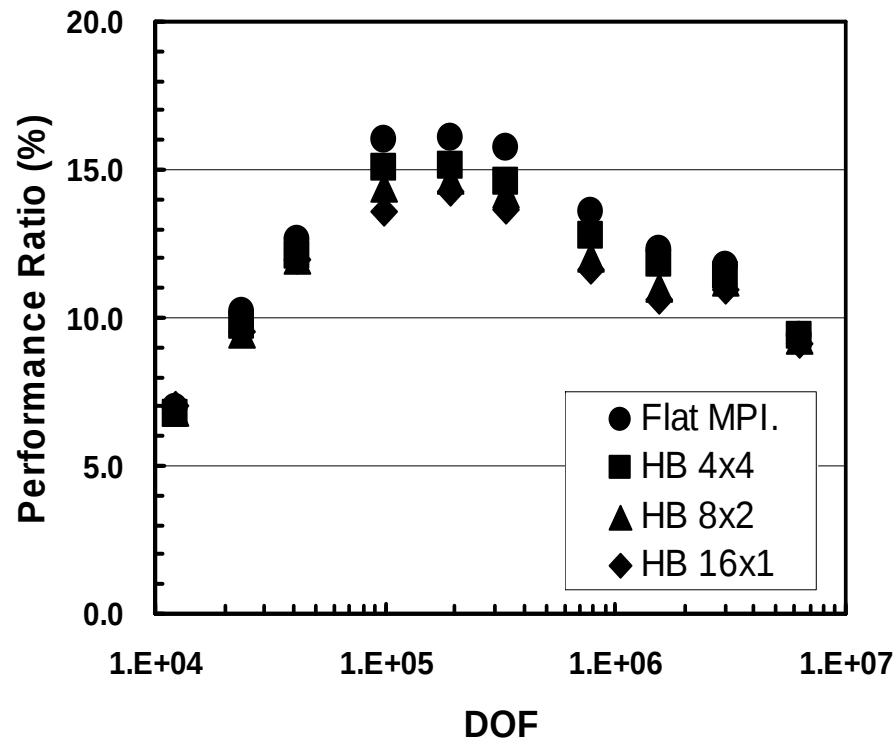
Hitachi SR11000/J2

Power 5+ 2.3GHz x 16

147.2 GFLOPS/node

100 GB/s for STREAM/Triad

L3 cache: 18MB/core



T2K/Tokyo

Opteron 2.3GHz x 16

147.2 GFLOPS/node

20 GB/s for STREAM/Triad

L3 cache: 0.5MB/core

