筑波大学計算科学研究センター
CCS HPCサマーセミナー
「並列数値アルゴリズムI」

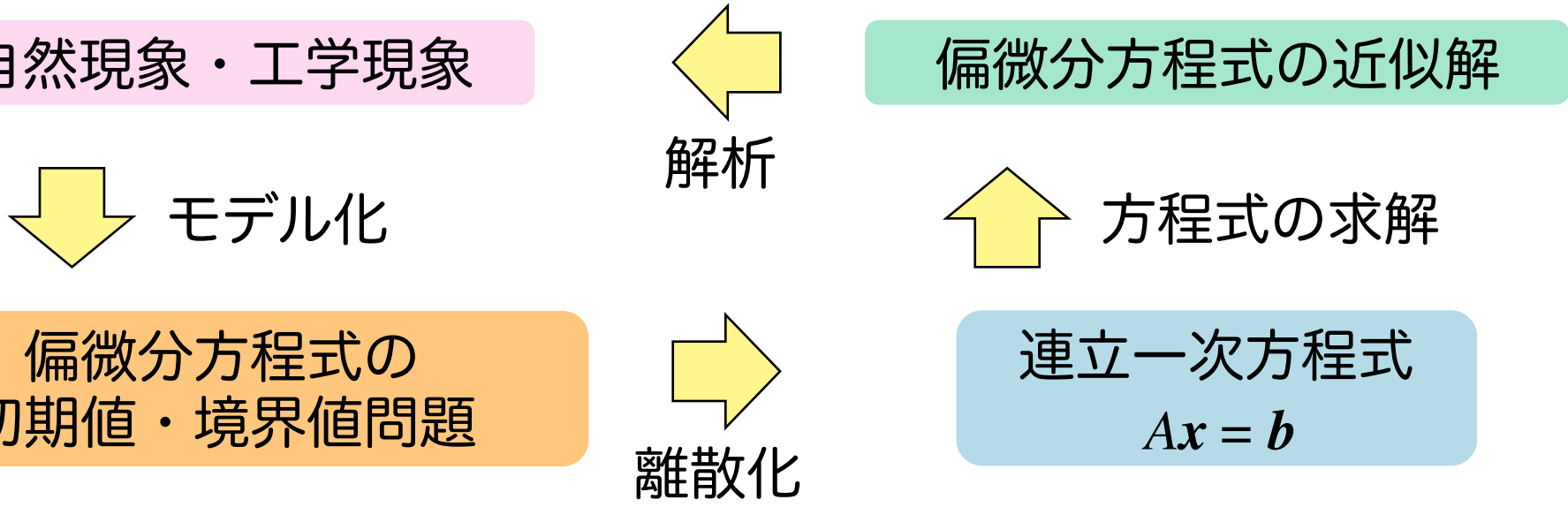
多田野 寛人
tadano@cs.tsukuba.ac.jp
筑波大学システム情報系
計算科学研究センター

講義内容

- 連立一次方程式の求解法
  - 様々な行列に対する解法
  - 疎行列の扱い方
  - 基本的な線形演算の MPI での並列化
- 複数本の右辺ベクトルをもつ連立一次方程式の解法
  - 解法の特色
  - 効率的な計算手法と OpenMP での並列化
- レポート課題について

連立一次方程式の求解法

自然現象・工学現象の解析



連立一次方程式は様々な分野における
数値シミュレーションで現れ,
計算時間の大部分が求解に費やされている

連立一次方程式

連立一次方程式：$A\boldsymbol{x} = \boldsymbol{b}$

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

連立一次方程式は様々な分野における
数値シミュレーションで現れ，
計算時間の大部分が求解に費やされている

直接解法と反復解法

直接解法

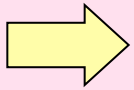
Gauss消去法，LU分解など

1) 有限回の演算で必ず解くことが出来る
2) 係数行列 A を変形するため，非零要素数が増大

反復解法

Krylov（クリロフ）部分空間反復法

1) 必要な演算は係数行列 A とベクトルの積，内積など



係数行列の疎性がそのまま使える

2) 問題によっては多くの反復回数を要することがある

代表的な直接解法

● Gauss の消去法

$$A\boldsymbol{x} = \boldsymbol{b}$$

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

変形

$$U\boldsymbol{x} = \boldsymbol{b}'$$

$$\begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & \cdots & u_{2n} \\ & & \ddots & \vdots \\ 0 & & & u_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b'_1 \\ b'_2 \\ \vdots \\ b'_n \end{bmatrix}$$

● LU 分解法　係数行列だけを変形

$$LU\boldsymbol{x} = \boldsymbol{b}$$

$$\begin{bmatrix} 1 & & 0 \\ l_{21} & 1 & \\ \vdots & \vdots & \ddots \\ l_{n1} & l_{n2} & \cdots \quad 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & \cdots & u_{2n} \\ & & \ddots & \vdots \\ 0 & & & u_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

1) $\boldsymbol{y} = U\boldsymbol{x}$ とおいて，$L\boldsymbol{y} = \boldsymbol{b}$ を解く．

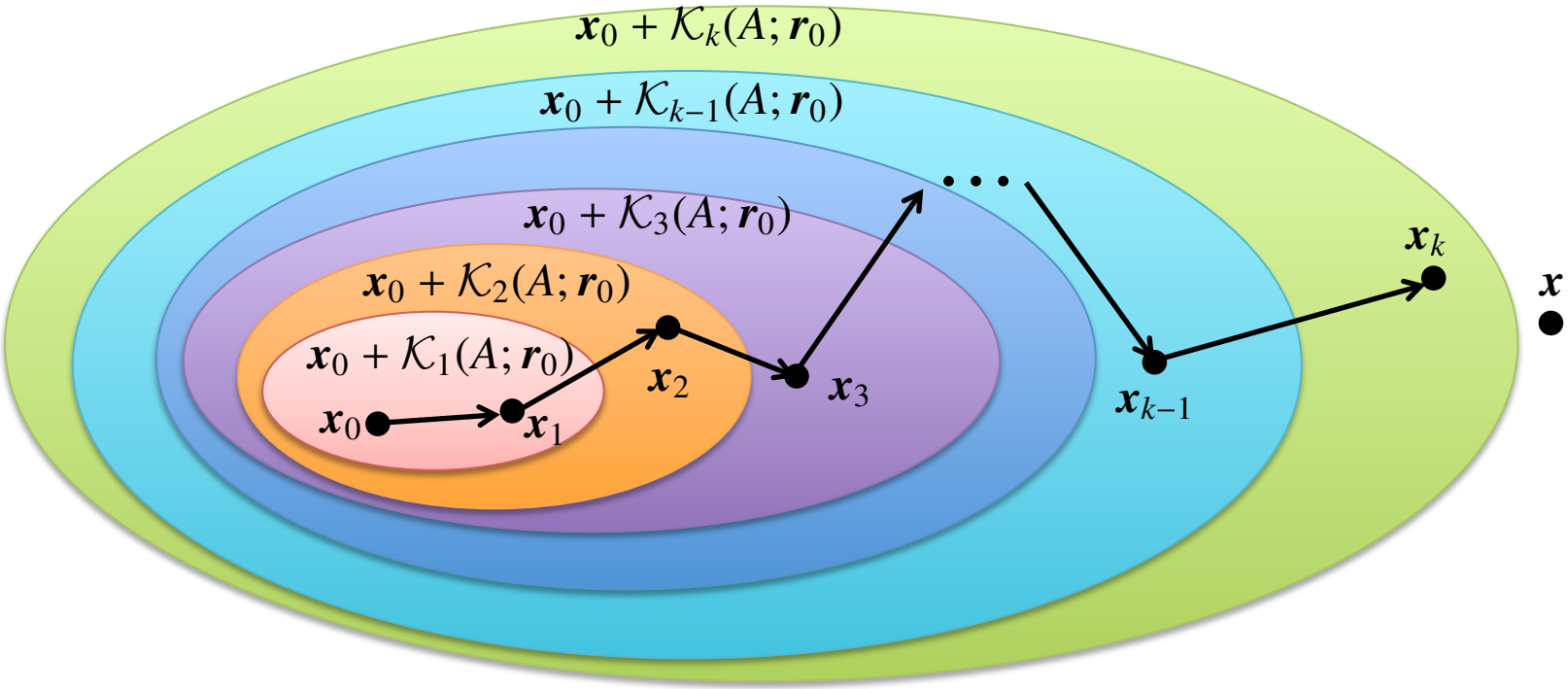
2) $U\boldsymbol{x} = \boldsymbol{y}$ を解く．

疎行列の直接解法ソフトウェア

- MUMPS（MPI並列も可）
  - http://graal.ens-lyon.fr/MUMPS/
- SuperLU（SuperLU_DIST でMPI並列も可）
  - http://crd-legacy.lbl.gov/~xiaoye/SuperLU/
- Pardiso（MPI, OpenMP並列も可）
  - Intel Math Kernel Library (MKL) にも入っている
  - http://www.pardiso-project.org/
- WSMP（MPI, スレッド並列も可）
  - http://researcher.watson.ibm.com/researcher/view_project.php?id=1426

Krylov部分空間反復法

- 初期解 $\boldsymbol{x}_0$ から出発し，反復によって更新して真の解 $\boldsymbol{x}$ を探索.
- $\mathcal{K}_j(A;\boldsymbol{r}_0)$ はクリロフ部分空間と呼ばれる. $\mathcal{K}_j(A;\boldsymbol{r}_0)$ はベクトル $\boldsymbol{r}_0, A\boldsymbol{r}_0, \ldots, A^{j-1}\boldsymbol{r}_0$ で張られる部分空間を表す.
- $\boldsymbol{r}_0$ は初期残差と呼ばれ, $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$ で計算される.



Krylov部分空間反復法の概念図.

エルミート行列に対する解法

1. 係数行列がエルミート行列（$A = A^{\mathrm{H}}$）の場合

- 共役勾配法（**C**onjugate **G**radient method: **CG法**）
- 共役残差法（**C**onjugate **R**esidual method: **CR法**）
- 最小残差法（**Min**imal **Res**idual Method: **MINRES法**）

係数行列のエルミート性を使うことで
短い漸化式（計算量が少ない）の
アルゴリズムが導出できる

補足：エルミート行列

$$A = A^{\mathrm{H}} = \bar{A}^{\mathrm{T}}$$

$$(a_{ij} = \bar{a}_{ji})$$

エルミート行列に対する解法

$\boldsymbol{x}_0$ is an initial guess,

Compute $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$,

Set $\boldsymbol{p}_0 = \boldsymbol{r}_0$,

For $k = 0, 1, \ldots,$ **until** $\|\boldsymbol{r}_k\|_2 \leq \varepsilon_{\mathrm{TOL}} \|\boldsymbol{b}\|_2$ **do** :

$$\boldsymbol{q}_k = A\boldsymbol{p}_k,$$ ← 行列ベクトル積

$$\alpha_k = \frac{(\boldsymbol{r}_k, \boldsymbol{r}_k)}{(\boldsymbol{p}_k, \boldsymbol{q}_k)},$$ ← 内積

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k,$$

$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k \boldsymbol{q}_k,$$ ← ベクトルの定数倍と加算

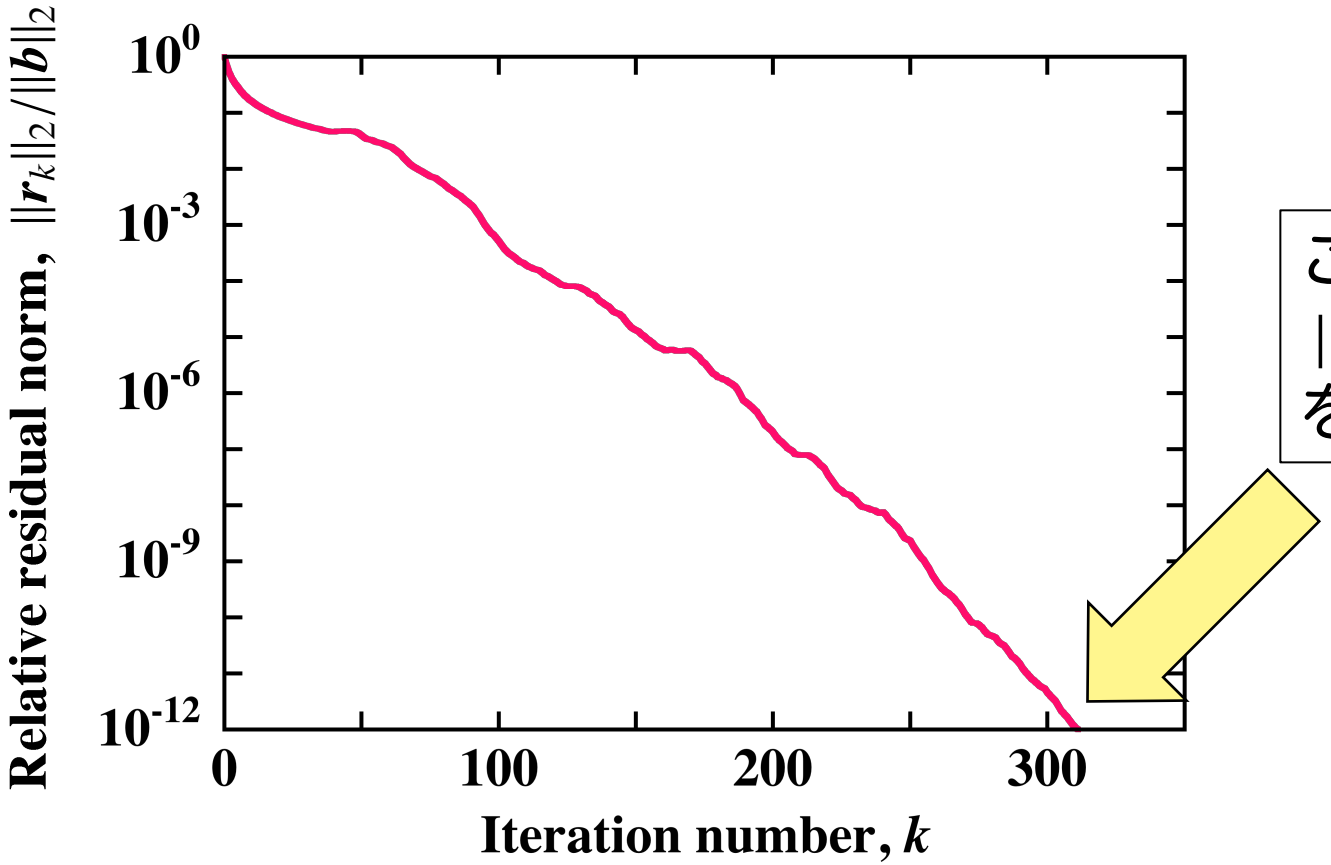
$$\beta_k = \frac{(\boldsymbol{r}_{k+1}, \boldsymbol{r}_{k+1})}{(\boldsymbol{r}_k, \boldsymbol{r}_k)},$$ ← 内積

$$\boldsymbol{p}_{k+1} = \boldsymbol{r}_{k+1} + \beta_k \boldsymbol{p}_k,$$ ← ベクトルの定数倍と加算

End For

共役勾配法（CG法）

CG法の相対残差履歴のイメージ



このイメージでは
$\|r_k\|_2/\|b\|_2 \leq 10^{-12}$
を満たしたら停止

・反復の過程で，相対残差（$\|r_k\|_2/\|b\|_2$）をチェックする.
・自分で定めた条件を満たしたら反復をやめ，x_k を解として採用.

非エルミート行列に対する解法

2. 係数行列が 非エルミート行列（$A \neq A^{\mathrm{H}}$）の場合

残差の双直交条件から導出される解法

- 双共役勾配法（**Bi-C**onjugate **G**radient: **BiCG法**）
- 自乗共役勾配法（**C**onjugate **G**radient **S**quared: **CGS法**）
- 双共役勾配安定化法（**BiCG Stab**ilization: **BiCGSTAB法**）

⇒ 計算量は少ないが，残差は単調減少しない

残差の最小条件から導出される解法

- 一般化共役残差法（**G**eneralized **C**onjugate **R**esidual: **GCR法**）
- 一般化最小残差法（**G**eneralized **M**inimal **Res**idual: **GMRES法**）

⇒ 残差は単調減少するが，長い漸化式が必要

非エルミート行列に対する解法

$\boldsymbol{x}_0$ is an initial guess,
Compute $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$,
Choose $\boldsymbol{r}_0^*$ such that $(\boldsymbol{r}_0^*, \boldsymbol{r}_0) \neq 0$,
Set $\boldsymbol{p}_0 = \boldsymbol{r}_0$ and $\boldsymbol{p}_0^* = \boldsymbol{r}_0^*$,
For $k = 0, 1, \ldots,$ **until** $\|\boldsymbol{r}_k\|_2 \leq \varepsilon_{\mathrm{TOL}} \|\boldsymbol{b}\|_2$ **do**:

$$\boldsymbol{q}_k = A\boldsymbol{p}_k, \qquad \boldsymbol{q}_k^* = A^{\mathrm{H}}\boldsymbol{p}_k^*,$$

$$\alpha_k = \frac{(\boldsymbol{r}_k^*, \boldsymbol{r}_k)}{(\boldsymbol{p}_k^*, \boldsymbol{q}_k)},$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k,$$

$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k \boldsymbol{q}_k, \qquad \boldsymbol{r}_{k+1}^* = \boldsymbol{r}_k^* - \bar{\alpha}_k \boldsymbol{q}_k^*,$$

$$\beta_k = \frac{(\boldsymbol{r}_{k+1}^*, \boldsymbol{r}_{k+1})}{(\boldsymbol{r}_k^*, \boldsymbol{r}_k)},$$

$$\boldsymbol{p}_{k+1} = \boldsymbol{r}_{k+1} + \beta_k \boldsymbol{p}_k, \qquad \boldsymbol{p}_{k+1}^* = \boldsymbol{r}_{k+1}^* + \bar{\beta}_k \boldsymbol{p}_k^*,$$

End For

行列ベクトル積

内積

ベクトルの定数倍と加算

双共役勾配法（BiCG法）

非エルミート行列に対する解法

$\boldsymbol{x}_0$ is an initial guess,
Compute $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$,
Set $\boldsymbol{p}_0 = \boldsymbol{r}_0$ and $\boldsymbol{q}_0 = \boldsymbol{s}_0 = A\boldsymbol{r}_0$,
For $k = 0, 1, \ldots,$ **until** $\|\boldsymbol{r}_k\|_2 \leq \varepsilon_{\mathrm{TOL}}\|\boldsymbol{b}\|_2$ **do** :

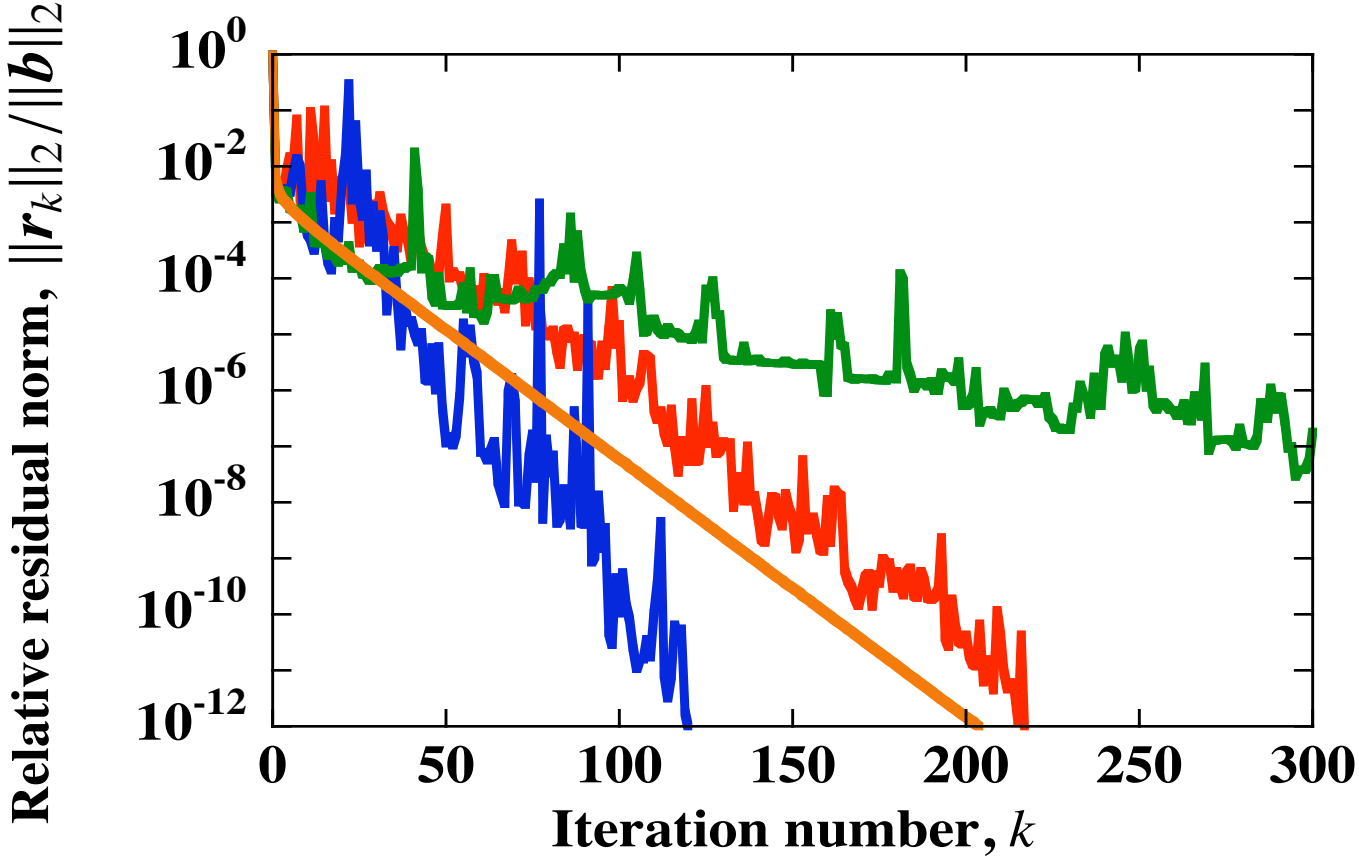
$$\alpha_k = \frac{(\boldsymbol{q}_k, \boldsymbol{r}_k)}{(\boldsymbol{q}_k, \boldsymbol{q}_k)},$$
$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k,$$
$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k \boldsymbol{q}_k,$$
$$\boldsymbol{s}_{k+1} = A\boldsymbol{r}_{k+1},$$
$$\beta_{k,i} = -\frac{(\boldsymbol{q}_i, \boldsymbol{s}_{k+1})}{(\boldsymbol{q}_i, \boldsymbol{q}_i)}, \quad (i = 0, 1, \ldots, k)$$
$$\boldsymbol{p}_{k+1} = \boldsymbol{r}_{k+1} + \sum_{i=0}^{k} \beta_{k,i} \boldsymbol{p}_i,$$
$$\boldsymbol{q}_{k+1} = \boldsymbol{s}_{k+1} + \sum_{i=0}^{k} \beta_{k,i} \boldsymbol{q}_i,$$

End For

- 行列ベクトル積は**1**反復あたり**1**回
- 長い漸化式を使うため多くのメモリが必要
- リスタートにより演算量とメモリ量を削減

一般化共役残差法（**GCR**法）

反復法の収束特性



反復法の相対残差履歴

－：BiCG法，－：CGS法，－：BiCGSTAB法，－：GCR法

複素対称行列に対する解法

3. 係数行列が複素対称行列（$A = A^{\mathrm{T}} \neq A^{\mathrm{H}}$）の場合

・共役直交共役勾配法

（**C**onjugate **O**rthogonal **C**onjugate **G**radient: **COCG法**）

係数行列が複素対称行列の場合は，
1反復あたり**1**回の行列ベクトル積で，
かつ短い漸化式で計算ができる

補足：複素対称行列

$$A = A^{\mathrm{T}} \neq A^{\mathrm{H}}$$

$$(a_{ij} = a_{ji} \neq \bar{a}_{ji})$$

複素対称行列に対する解法

x_0 is an initial guess,

Compute $r_0 = b - Ax_0$,

Set $p_0 = r_0$,

For $k = 0, 1, \ldots,$ **until** $\|r_k\|_2 \leq \varepsilon_{\mathrm{TOL}} \|b\|_2$ **do** :

$$q_k = Ap_k,$$ ← 行列ベクトル積

$$\alpha_k = \frac{(\bar{r}_k, r_k)}{(\bar{p}_k, q_k)},$$ ← 内積

$$x_{k+1} = x_k + \alpha_k p_k,$$

$$r_{k+1} = r_k - \alpha_k q_k,$$ ← ベクトルの定数倍と加算

$$\beta_k = \frac{(\bar{r}_{k+1}, r_{k+1})}{(\bar{r}_k, r_k)},$$ ← 内積

$$p_{k+1} = r_{k+1} + \beta_k p_k,$$ ← ベクトルの定数倍と加算

End For

共役直交共役勾配法（COCG法）

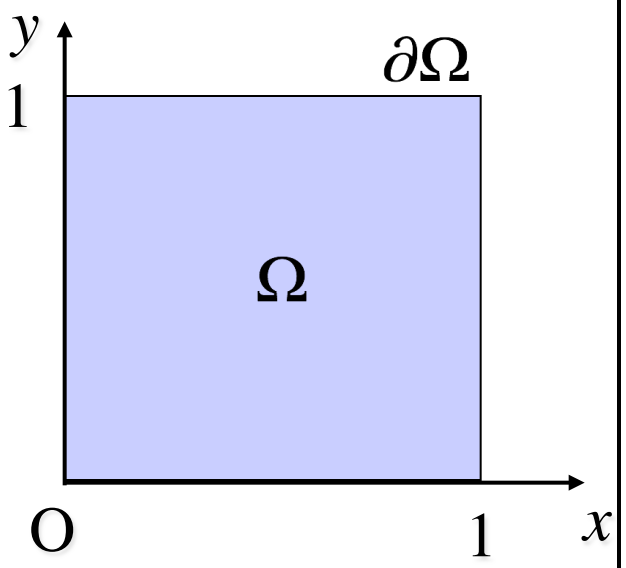
疎行列が現れる例

2次元 Poisson 問題

$$\begin{cases} \dfrac{\partial^2 u}{\partial x^2} + \dfrac{\partial^2 u}{\partial y^2} = f, & \text{in } \Omega \\ u = \bar{u}, & \text{on } \partial\Omega \end{cases}$$

f, $\bar{u}$ は既知の関数.

Ω をx, y 方向に $(M+1)$等分し
5点中心差分で離散化



$M \times M$次行列をもつ連立一次方程式に帰着

行列の要素数：M^4 個
非零要素の数：$5M^2 - 4M$ 個

疎行列の格納形式（CRS形式）

Compressed Row Storage (CRS) 形式
【補足】Compressed Sparse Row (CSR) と呼ぶこともあります.

$$A = \begin{bmatrix} a_{11} & 0 & a_{13} & 0 & a_{15} \\ 0 & a_{22} & 0 & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & 0 & 0 \\ 0 & 0 & a_{43} & a_{44} & 0 \\ 0 & a_{52} & 0 & a_{54} & a_{55} \end{bmatrix}$$

`val`: 非零要素を格納する配列

`col_ind`: 非零要素の列番号を格納

`row_ptr`: 各行の先頭の非零要素が格納されている場所を指す配列

`val`:

a_{11}	a_{13}	a_{15}	a_{22}	a_{24}	a_{25}	a_{31}	a_{32}	a_{33}	a_{43}	a_{44}	a_{52}	a_{54}	a_{55}

`col_ind`:

1	3	5	2	4	5	1	2	3	3	4	2	4	5

`row_ptr`:

1	4	7	10	12	15

最後は非零要素数**+1** の値

疎行列の格納形式（CCS形式）

Compressed Column Storage (CCS) 形式
【補足】Compressed Sparse Column (CSC) と呼ぶこともあります.

$$A = \begin{bmatrix} a_{11} & 0 & a_{13} & 0 & a_{15} \\ 0 & a_{22} & 0 & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & 0 & 0 \\ 0 & 0 & a_{43} & a_{44} & 0 \\ 0 & a_{52} & 0 & a_{54} & a_{55} \end{bmatrix}$$

`val`: 非零要素を格納する配列

`row_ind`: 非零要素の行番号を格納

`col_ptr`: 各列の先頭の非零要素が格納されている場所を指す配列

`val`:

a_{11}	a_{31}	a_{22}	a_{32}	a_{52}	a_{13}	a_{33}	a_{43}	a_{24}	a_{44}	a_{54}	a_{15}	a_{25}	a_{55}

`row_ind`:

1	3	2	3	5	1	3	4	2	4	5	1	2	5

`col_ptr`:

1	3	6	9	12	15

最後は非零要素数+1 の値

CRS形式の行列ベクトル積

行列 A とベクトル x の積 $y = Ax$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Fortran Code

```
do i=1,n
  y(i) = 0.0D0
  do j=row_ptr(i), row_ptr(i+1)-1
    y(i) = y(i)+val(j)*x(col_ind(j))
  end do
end do
```

CCS形式の行列ベクトル積

行列 A とベクトル $\boldsymbol{x}$ の積 $\boldsymbol{y} = A\boldsymbol{x}$

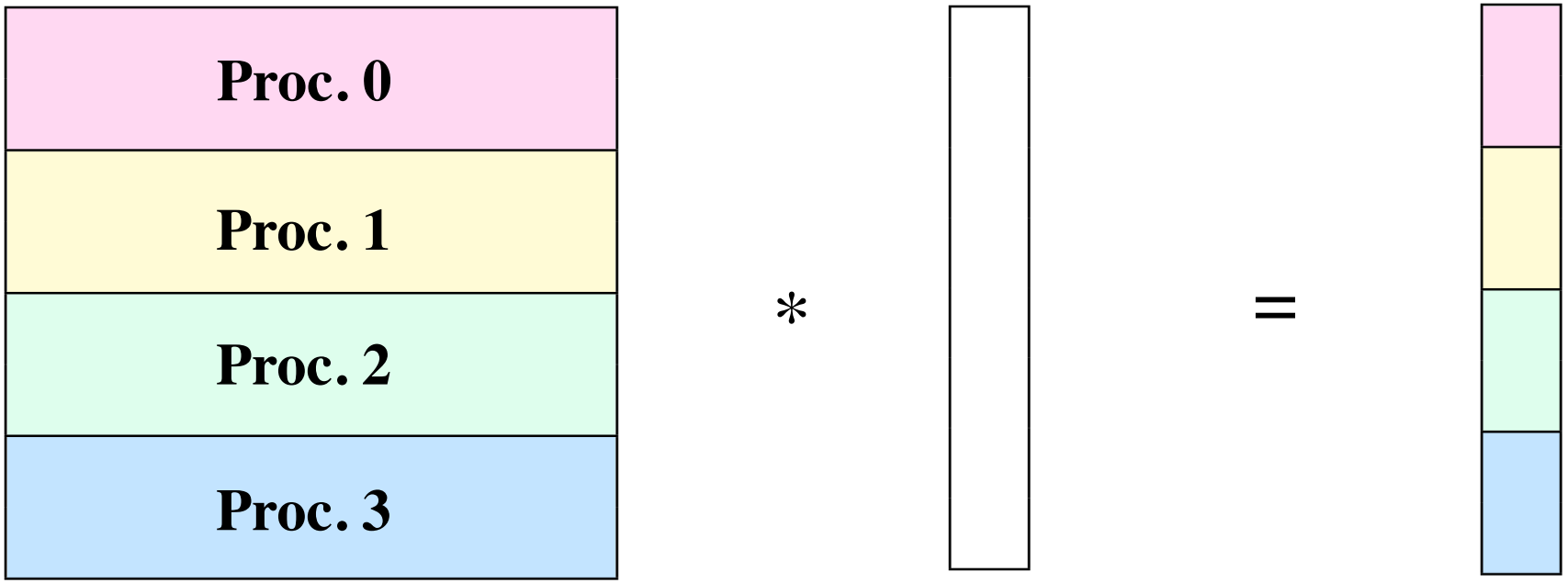
$$\boldsymbol{y} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_n] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \sum_{i=1}^{n} \boldsymbol{a}_i x_i$$

Fortran Code

```
do i=1,n
  y(i) = 0.0D0
end do
do j=1,n
  do i=col_ptr(j),col_ptr(j+1)-1
    y(row_ind(i)) = y(row_ind(i))+val(i)*x(j)
  end do
end do
```

行列ベクトル積の並列化

・CRS形式の場合の $y = Ax$ の計算



A x y

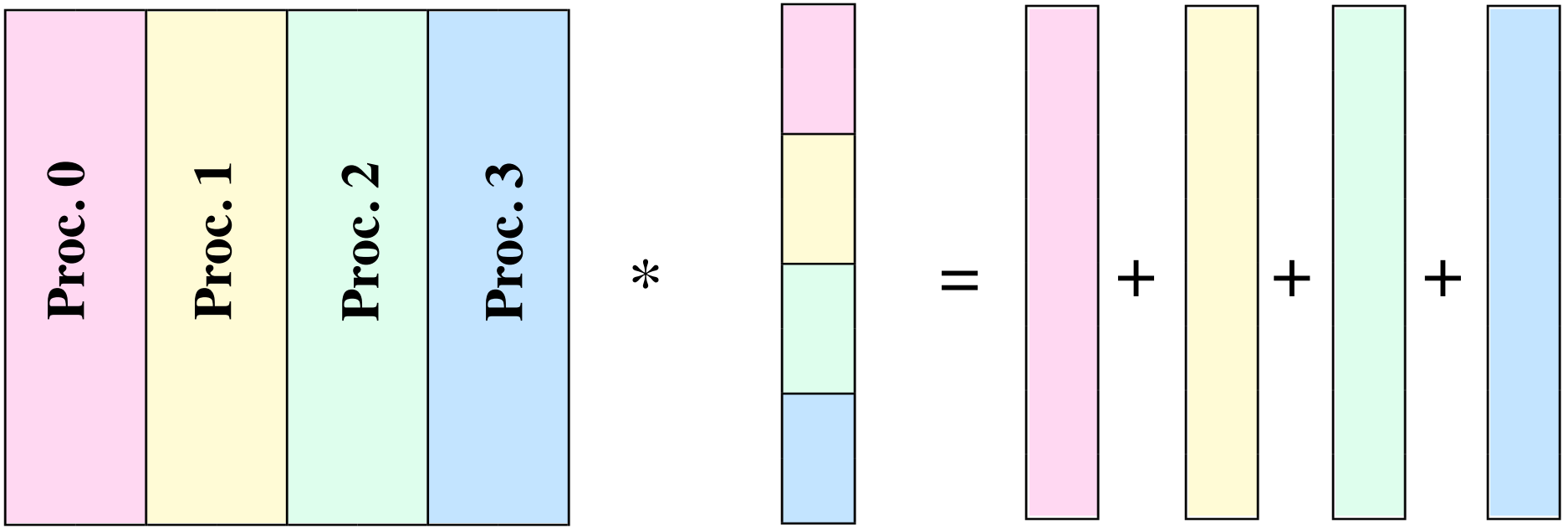
各プロセスで分散してデータを保持する

全てのプロセスで保持する

MPI_Gather で Proc. 0 に集める

行列ベクトル積の並列化

・CCS形式の場合の $y = Ax$ の計算



A x y

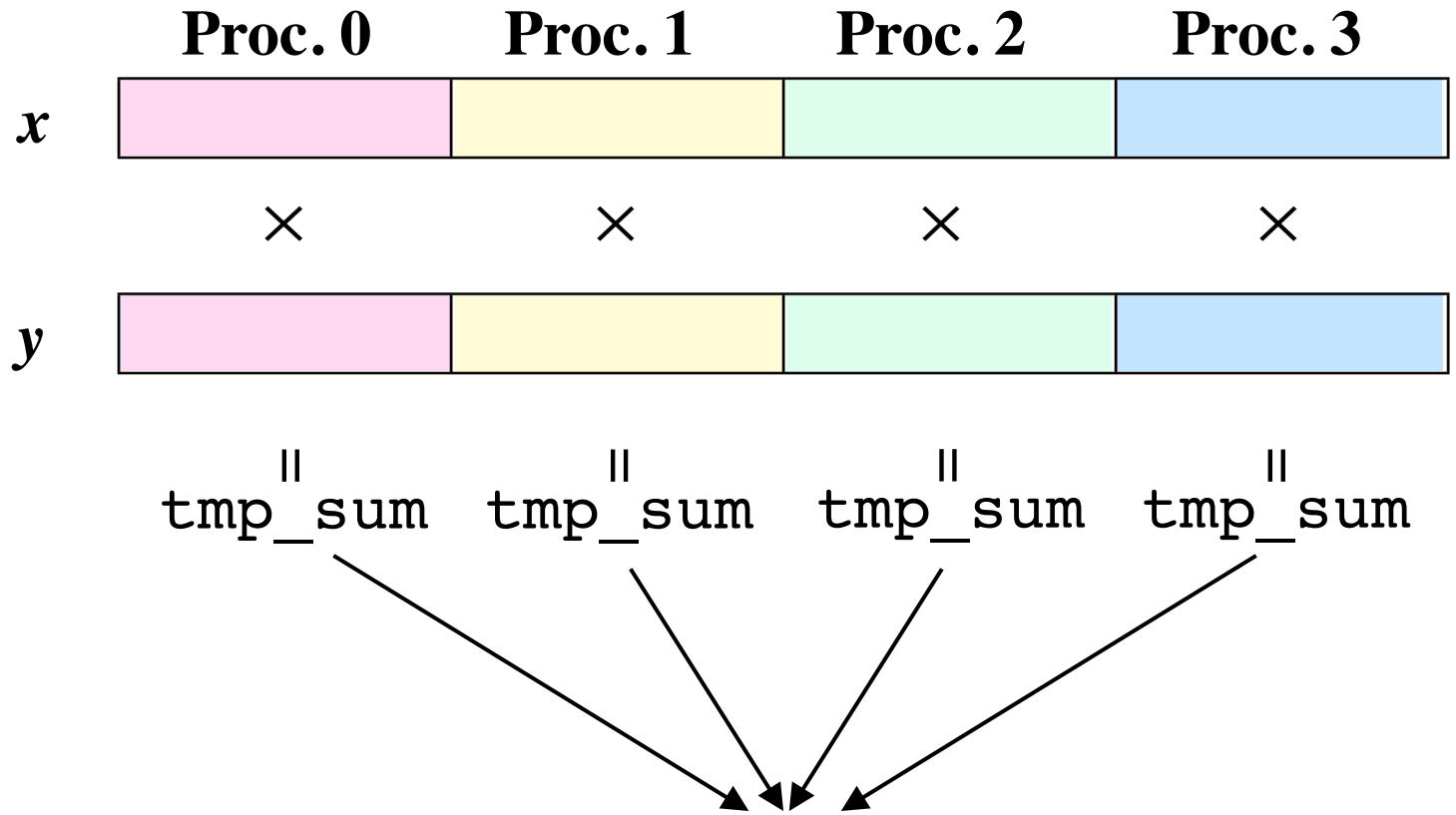
各プロセスで分散してデータを保持する

分散して保持する

MPI_Reduce でProc. 0に結果を足し合わせて送る

内積の並列化

$$(x, y) = \sum_{j=1}^{n} \bar{x}_j y_j$$



MPI_Reduce で Proc.0 に集める

内積計算の MPI コード例

$$(\boldsymbol{x}, \boldsymbol{y}) = \sum_{j=1}^{n} \bar{x}_j y_j$$

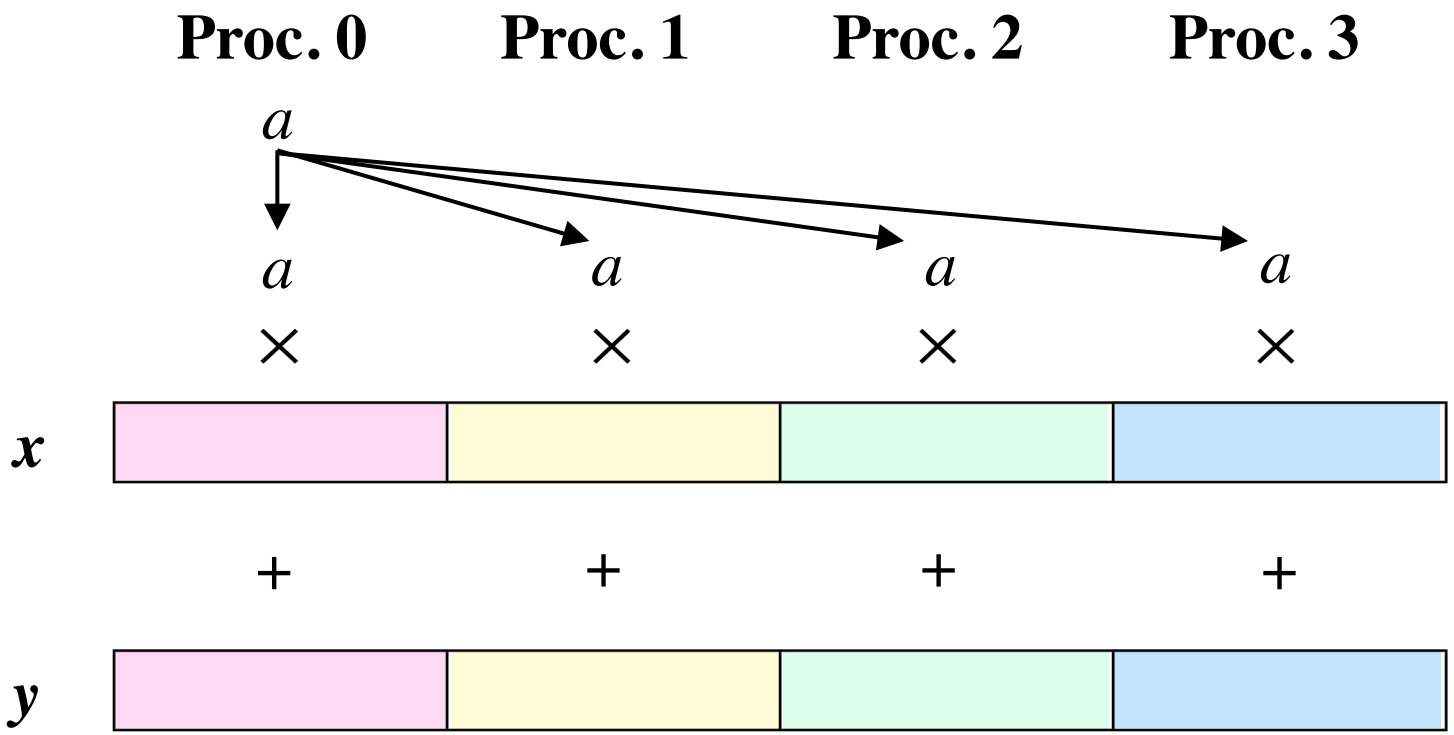
```
program main
include 'mpif.h'
...
call mpi_init(ierr)
call mpi_comm_size(mpi_comm_world, nprocs, ierr)
call mpi_comm_rank(mpi_comm_world, myrank, ierr)
...
tmp_sum = (0.0D0, 0.0D0)
do i=istart(myrank+1), iend(myrank+1)
  tmp_sum = tmp_sum + conj(x(i)) * y(i)
end do

call mpi_reduce(tmp_sum, sum, 1, mpi_double_complex,
                mpi_sum, 0, mpi_comm_world, ierr)
...
call mpi_finalize(ierr)
```

ベクトルの定数倍と加算の並列化

$y = y + ax$ （x, y：ベクトル，a：スカラー）

a を MPI_Bcast で全プロセスに送る



連立一次方程式の前処理法

Krylov部分空間反復法で，残差が収束しないことがある

Krylov 部分空間法の特徴

係数行列が単位行列に近い場合，少ない反復回数で残差が収束

連立一次方程式

$Ax = b$

前処理

前処理後の連立一次方程式

$A'x' = b'$

係数行列 A' が単位行列に近くなるように変形！

連立一次方程式の前処理法

係数行列 A を近似する前処理法

$$A \approx K_1 K_2 \Longleftrightarrow K_1^{-1} A K_2^{-1} \approx I$$

これを用いて

$$A\boldsymbol{x} = \boldsymbol{b} \Longleftrightarrow \underbrace{(K_1^{-1} A K_2^{-1})}_{A'} \underbrace{(K_2 \boldsymbol{x})}_{\boldsymbol{x}'} = \underbrace{K_1^{-1} \boldsymbol{b}}_{\boldsymbol{b}'}$$

【注】行列 K_1, K_2 は逆行列が簡単に計算できるように構成.
例えば，上三角行列や下三角行列にする.

- エルミート行列の場合　：不完全コレスキー分解（$A \approx LL^{\mathrm{H}}$）
- 非エルミート行列の場合：不完全 LU 分解（$A \approx LU$）

ここで，L : 下三角行列，U : 上三角行列.

連立一次方程式の前処理法

逆行列 A^{-1} を近似する前処理法

$$A \approx M^{-1} \Longleftrightarrow AM \approx I,\ MA \approx I$$

これを用いて

$$Ax = b \Longleftrightarrow \underbrace{MA}_{A'}\, x = \underbrace{Mb}_{b'}$$

または $$Ax = b \Longleftrightarrow \underbrace{(AM)}_{A'}\,\underbrace{(M^{-1}x)}_{x'} = b$$

このカテゴリに属する前処理法として「近似逆行列前処理」や「多項式前処理」などがある.

近似逆行列前処理

逆行列 A^{-1} を近似する行列 M を生成する前処理

$F(M) = \|I - AM\|_{\mathrm{F}}^2$ を最小にするように M を決定

$$F(M) = \|I - AM\|_{\mathrm{F}}^2 = \sum_{j=1}^{n} \|\boldsymbol{e}_j - A\boldsymbol{m}_j\|_2^2$$

- 行列 M の非零構造は，任意に選択できる
- M を密行列とすれば，$M = A^{-1}$ となる

互いに独立な n 個の最小自乗問題のため，並列処理が可能！

補足：フロベニウスノルム

$$\|A\|_{\mathrm{F}} = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} |a_{ij}|^2}$$

多項式前処理

逆行列 A^{-1} の近似を A の多項式で生成する

行列のNeumann展開による多項式生成

$\|I - A\| < 1$ の場合

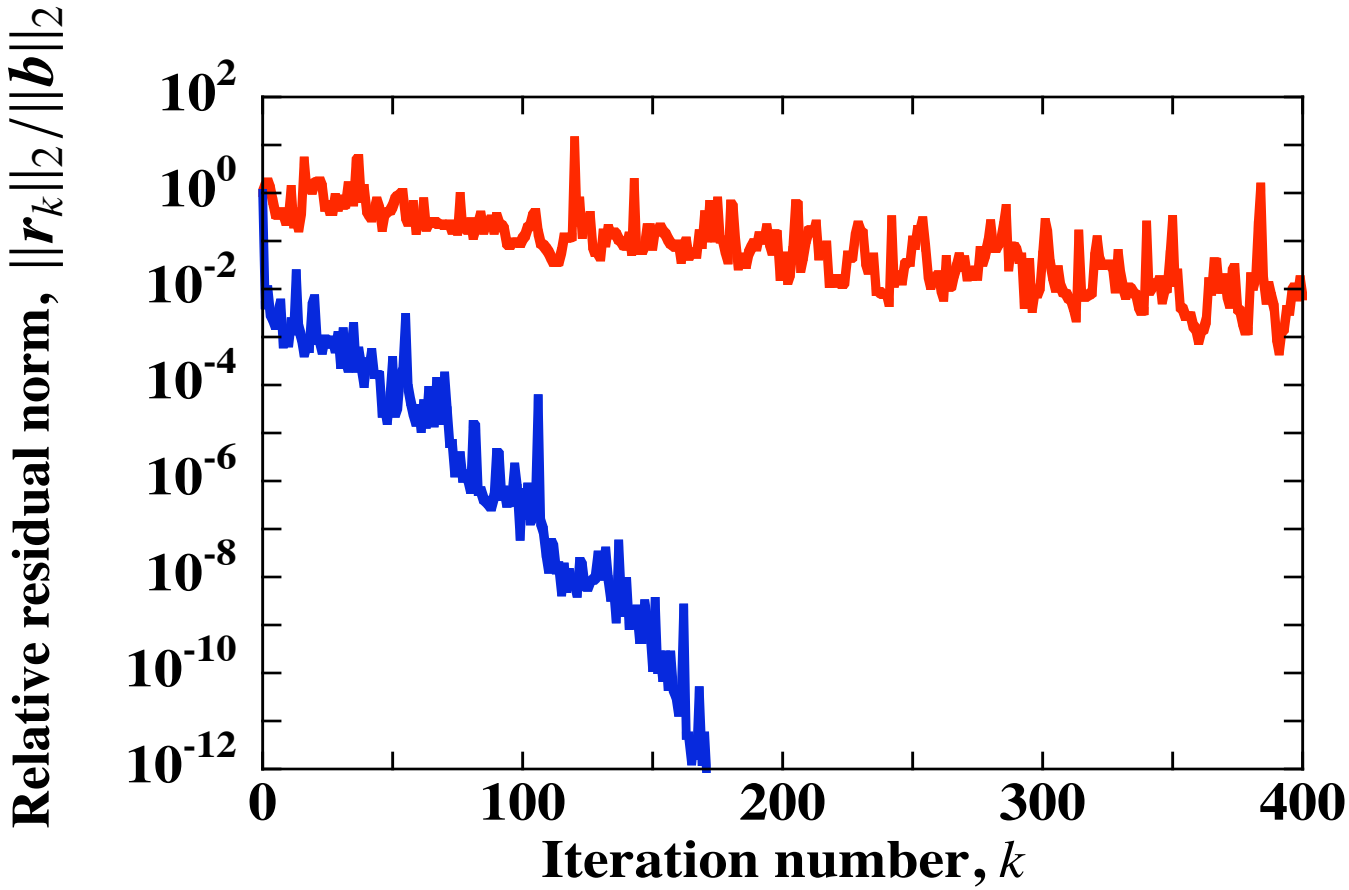
$$A^{-1} = [I - (I - A)]^{-1} = \sum_{j=0}^{\infty} (I - A)^j$$

より，m 次までで打ち切った行列を M とすると

$$A^{-1} \approx M = \sum_{j=0}^{m} (I - A)^j$$

・実際に行列 M は生成せず，反復法内で行列ベクトル積で計算
・行列ベクトル積の並列化により，同前処理を並列化できる

前処理付き反復法の収束特性



反復法の相対残差履歴

－：BiCG法，　－：近似逆行列前処理付きBiCG法

複数本の右辺ベクトルをもつ連立一次方程式の解法

複数右辺ベクトルをもつ方程式

右辺が L 本の連立一次方程式

$$AX = B$$

ここで，A : n次行列,

$$X = \left[\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(L)}\right],\ B = \left[\boldsymbol{b}^{(1)}, \boldsymbol{b}^{(2)}, \ldots, \boldsymbol{b}^{(L)}\right]$$

直接法による解法

- 係数行列の完全分解（$A = LU$ など）が必要
- 完全分解できれば，L 回の前進・後退代入で OK
- 分解には多くの計算量，メモリ量が必要

反復法で L 本の方程式を効率よく解けないか？

Block Krylov 部分空間反復法

Block Krylov 部分空間反復法の種類

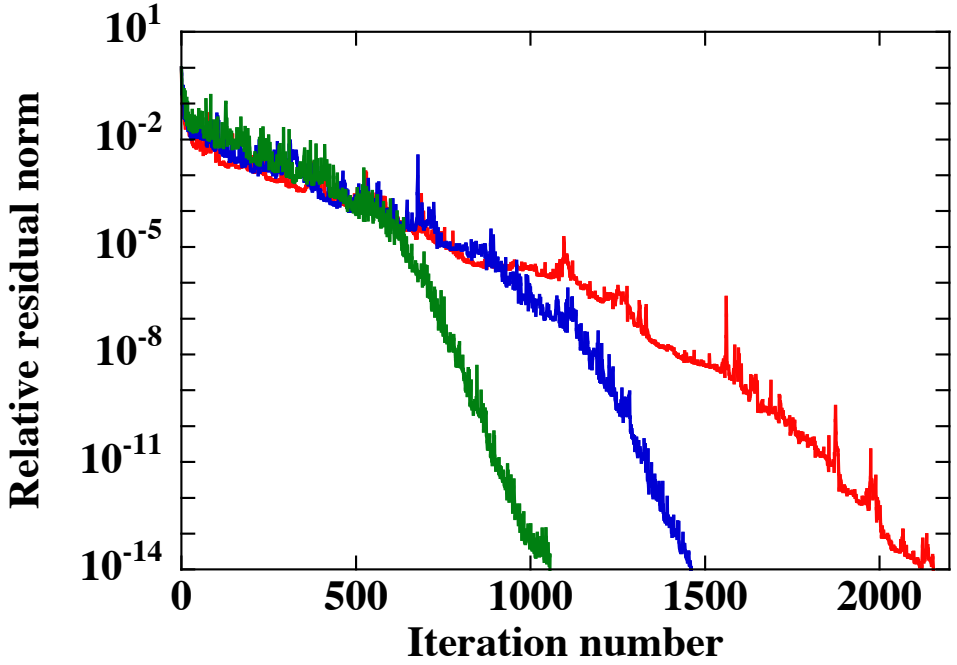
・Block BiCG法	O'Leary (1980)
・Block GMRES法	Vital (1990)
・Block QMR法	Freund (1997)
・Block BiCGSTAB法	Guennouni (2003)
・Block BiCGGR法	Tadano (2009)

複数本の右辺ベクトルをまとめて扱うことで
効率よく解を求めることができる

Block Krylov 部分空間反復法

「効率よく」とはどういうことか？

1本ずつ解いた場合より少ない反復回数で残差が収束する（ことがある）



Block BiCGSTAB法の相対残差履歴.

ここで, ■：$L=1$, ■：$L=2$, ■：$L=4$.

Block BiCGSTAB法

$X_0 \in \mathbb{C}^{n\times L}$ is an initial guess,
Compute $R_0 = B - AX_0$,
Set $P_0 = R_0$,
Choose $\tilde{R}_0 \in \mathbb{C}^{n\times L}$,
For $k = 0, 1, \ldots,$ **until** $\|R_k\|_{\mathrm{F}} \le \varepsilon \|B\|_{\mathrm{F}}$ **do:**

$$
\begin{aligned}
V_k &= AP_k, \\
\textbf{Solve } (\tilde{R}_0^{\mathrm{H}} V_k)\alpha_k &= \tilde{R}_0^{\mathrm{H}} R_k \text{ for } \alpha_k, \\
T_k &= R_k - V_k\alpha_k, \\
Z_k &= AT_k, \\
\zeta_k &= \mathrm{Tr}\left[Z_k^{\mathrm{H}} T_k\right] / \mathrm{Tr}\left[Z_k^{\mathrm{H}} Z_k\right], \\
X_{k+1} &= X_k + P_k\alpha_k + \zeta_k T_k, \\
R_{k+1} &= T_k - \zeta_k Z_k, \\
\textbf{Solve } (\tilde{R}_0^{\mathrm{H}} V_k)\beta_k &= -\tilde{R}_0^{\mathrm{H}} Z_k \text{ for } \beta_k, \\
P_{k+1} &= R_{k+1} + (P_k - \zeta_k V_k)\beta_k,
\end{aligned}
$$

End

BiCGSTAB 法と異なるところ

1. 行列・ベクトル積の本数が1本からL本に増加.
2. α_k, β_kが L 次行列になった.
3. ベクトルの定数倍の計算が行列・行列積になった.
4. ζ_k の計算に行列のトレース $\mathrm{Tr}\,[\,\cdot\,]$ が必要になった.
   トレース：対角成分の和.

行列・ベクトル積の効率化

・行列は CRS 形式で格納されているとする.
・$Y = AX$ を計算. Y, X は n行L列の配列.

```
do k=1,L
 do i=1,n
  do j=row_ptr(i), row_ptr(i+1)-1
   Y(i,k)=Y(i,k)+val(j)*X(col_ind(j),k)
  end do
 end do
end do
```

[問題点]
・X についてメモリの連続アクセスができていない
（Fortran は行方向に連続にデータが並んでいる）
・行列データを L 回読まなければならない.

行列・ベクトル積の効率化

[解決策]

・X, Y を転置した形で保持させる

```
do i=1,n
 do j=row_ptr(i), row_ptr(i+1)-1
  do k=1,L
   Y(k,i)=Y(k,i)+val(j)*X(k,col_ind(j))
  end do
 end do
end do
```

・Xについて（少なくとも L 回は）連続アクセスができる.

・行列データは1回しか読み込まない.

・Yについても連続アクセスが保たれている.

$n \times L$行列・$L \times L$行列積の計算

・行列・ベクトル積の効率化のためにベクトルを転置した.

・$n \times L$ 行列と $L \times L$ 行列の積の計算も工夫が必要.

転置

$$T_k = R_k - V_k \alpha_k \quad \Rightarrow \quad T_k^{\mathrm{T}} = R_k^{\mathrm{T}} - \alpha_k^{\mathrm{T}} V_k^{\mathrm{T}}$$

```
do j=1,n
 do i=1,L
  T(i,j)=R(i,j)
 end do
end do
do j=1,n
 do i=1,L
  do k=1,L
   T(k,j)=T(k,j)- Alpha(k,i)*V(i,j)
  end do
 end do
end do
```

Alpha は転置済みとする.

ベクトルと α_k を全て
転置して保持することで
連続アクセスが可能に.

$L \times n$行列・$n \times L$行列積の計算

- α_k, β_k を求めるために必要.
- $C_k = \tilde{R}_0^{\mathrm{H}} V_k$ を計算することを考える.

```
do j=1,n
 do i=1,L
  do k=1,L
   C(k,i)=C(k,i)+R0(k,j)*V(i,j)
  end do
 end do
end do
```

- R0 にはあらかじめ共役をとったものを代入.
- C_k の計算も，連続メモリアクセスを保持できる.

OpenMP による並列化

・共有メモリ向けの並列化インターフェース.
・既存のプログラムに数行加えるだけで並列化ができる.

```
!$OMP PARALLEL
  [  プログラム  ]
!$OMP END PARALLEL
```

と書くと，スレッドが立ち上がり，スレッド毎に別々の処理ができるようになる.

（以下のコードは，!$OMP PARALLEL と
!$OMP END PARALLEL で囲まれているとします．）

OpenMP による並列化

1. 行列・ベクトル積の並列化

```
!$OMP DO PRIVATE(j,k)
do i=1,n
 do j=row_ptr(i), row_ptr(i+1)-1
  do k=1,L
   Y(k,i)=Y(k,i)+val(j)*X(k,col_ind(j))
  end do
 end do
end do
```

最初の do ループの前に !$OMP DO ... を加えるだけ.

OpenMP による並列化

2. $n \times L$行列・$L \times L$行列の積の並列化

```
!$OMP DO PRIVATE(i)
do j=1,n
 do i=1,L
  T(i,j)=R(i,j)
 end do
end do
!$OMP DO PRIVATE(i,k)
do j=1,n
 do i=1,L
  do k=1,L
   T(k,j)=T(k,j)- Alpha(k,i)*V(i,j)
  end do
 end do
end do
```

2行加えるだけで，簡単に並列化.

OpenMP による並列化

3. *L*×*n*行列・*n*×*L*行列の積の並列化

配列に対する Reduction 処理が必要となる。

コードの冒頭で以下を実行

```
NTH  = OMP_GET_NUM_THREADS()
MYID = OMP_GET_THREAD_NUM()+1
!$OMP SINGLE
allocate( TMP(L,L,NTH) )
!$OMP END SINGLE
```

NTH　：スレッド数

MYID ：自分のスレッド番号

TMP　：並列計算を行うための配列

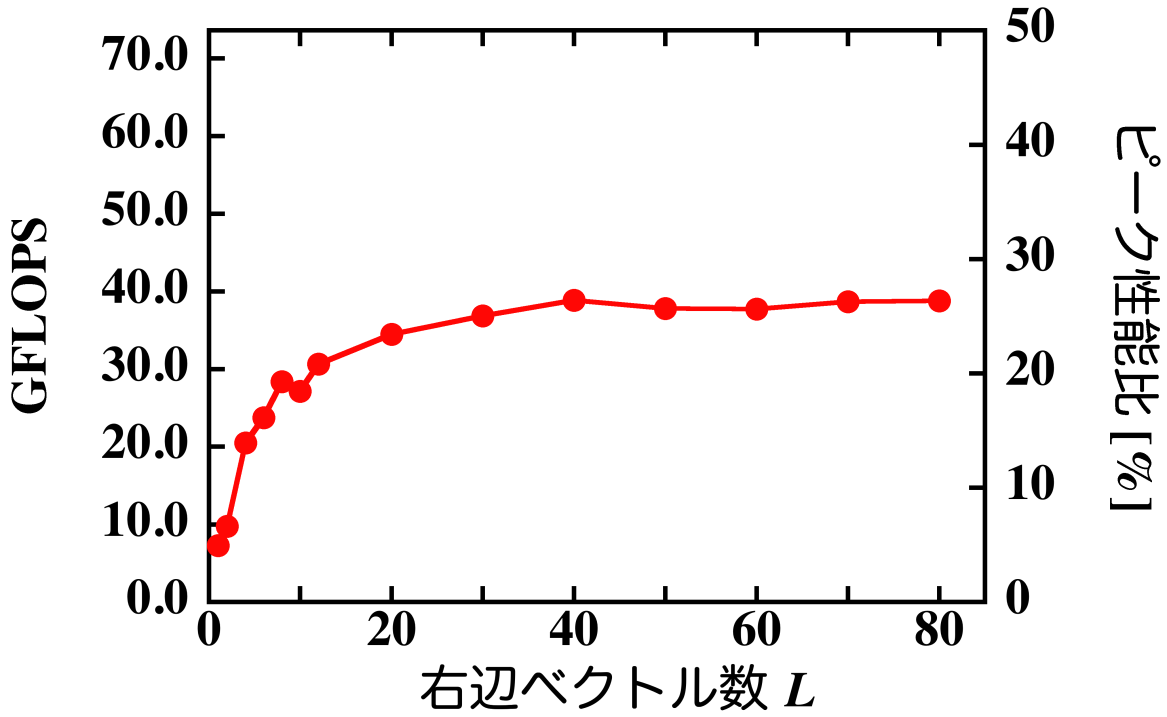
OpenMP による並列化

次に以下を実行して $C_k = \tilde{R}_0^{\mathrm{H}} V_k$ を計算する.

```
!$OMP DO PRIVATE(i,k)
do j=1,n
 do i=1,L
  do k=1,L
   TMP(k,i,MYID) = TMP(k,i,MYID)+R0(k,j)*V(i,j)
  end do
 end do
end do
!$OMP SINGLE
do k=1,NTH
 do j=1,L
  do i=1,L
   C(i,j) = C(i,j) + TMP(i,j,k)
  end do
 end do
end do
!$OMP END SINGLE
```

このコードで配列に対する
Reduction 処理を実現.

行列ベクトル積の性能



右辺ベクトル数の変化に対する GFLOPS値，ピーク性能比変化.

行列サイズ：1,572,864，非ゼロ要素数：80,216,064.

実験環境：T2K-Tsukuba の 1ノード（ピーク性能：147.2 GFLOPS）

CPU : AMD Opteron 2.3GHz × 4，OpenMP で16並列で計算.

OpenMP による並列化

[解いた方程式]
サイズ：1, 572, 864
非零要素数：80, 216, 064
右辺ベクトル数：4

[計算環境]
CPU: Intel Xeon X5550 2.67GHz × 2
Mem: 48GBytes
OS: Cent OS 5.3
コンパイラ：Intel Fortran ver. 11.1
オプション：-fast -openmp

スレッド数	時間（反復回数）	時間 / 反復回数	並列化効率
1	303.49 (179)	1.6955	1.00
2	183.07 (179)	1.0227	1.66
3	138.07 (179)	0.7713	2.20
4	104.61 (181)	0.5749	2.95
5	80 57 (181)	0.4451	3.81
6	78.56 (181)	0.4340	3.91
7	74.96 (181)	0.4141	4.09
8	68.18 (181)	0.3767	4.50

まとめ

- 連立一次方程式の解法である Krylov 部分空間反復法を取り上げた.
- 疎行列に対する行列・ベクトル積の実装方法とその並列化について述べた.
- Block Krylov 部分空間反復法とコードの最適化，及び OpenMP での並列化について述べた.

レポート課題

1. 行列

$$A = \begin{bmatrix} 2 & 1 & & & \\ \gamma & 2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & \gamma & 2 & 1 \\ & & & \gamma & 2 \end{bmatrix}$$

をCRS形式で保持するプログラムを作りなさい.

2. CRS形式の行列ベクトル積，転置行列 (A^{T}) ベクトル積を行うプログラムを作りなさい.

3. 双共役勾配法で連立一次方程式 $A\boldsymbol{x} = \boldsymbol{b}$ を解くプログラムを作りなさい．但し，$\boldsymbol{b} = A[1, 1, \ldots, 1]^{\mathrm{T}}$ とする．行列のパラメータは，$0 < \gamma < 1$ とし，$\|\boldsymbol{r}_k\|_2 / \|\boldsymbol{b}\|_2 \leq 1.0 \times 10^{-10}$ となったら停止すること.

4. 複数のパラメータ γ について実験し，反復過程における相対残差 $\|\boldsymbol{r}_k\|_2 / \|\boldsymbol{b}\|_2$ をグラフにプロットしなさい.

プログラムリストも提出すること．プログラミング言語は何を用いてもよい.