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」

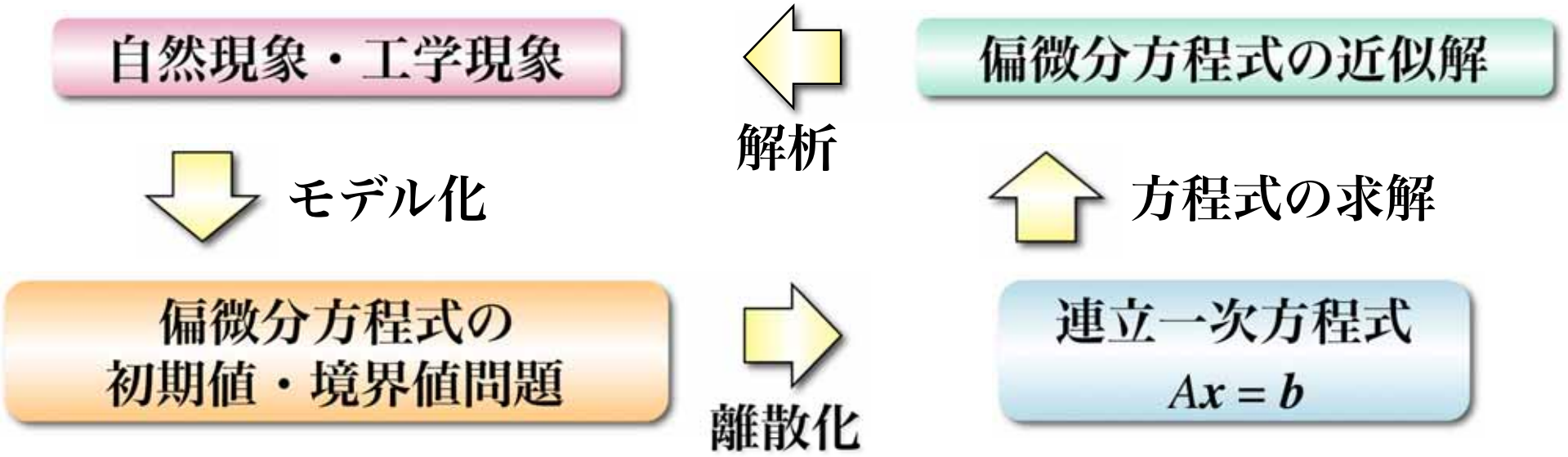
多田野寛人
tadano@cs.tsukuba.ac.jp
筑波大学大学院システム情報工学研究科
計算科学研究センター

講義内容

- 連立一次方程式の求解法
- 精度混合型 Krylov 部分空間反復法
- 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：$Ax = b$

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

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) 係数行列 A を変形するため，非零要素数が増大

⇒ 係数行列の疎性が利用出来ない

2) 有限回の演算で必ず解くことが出来る

反復解法

Krylov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

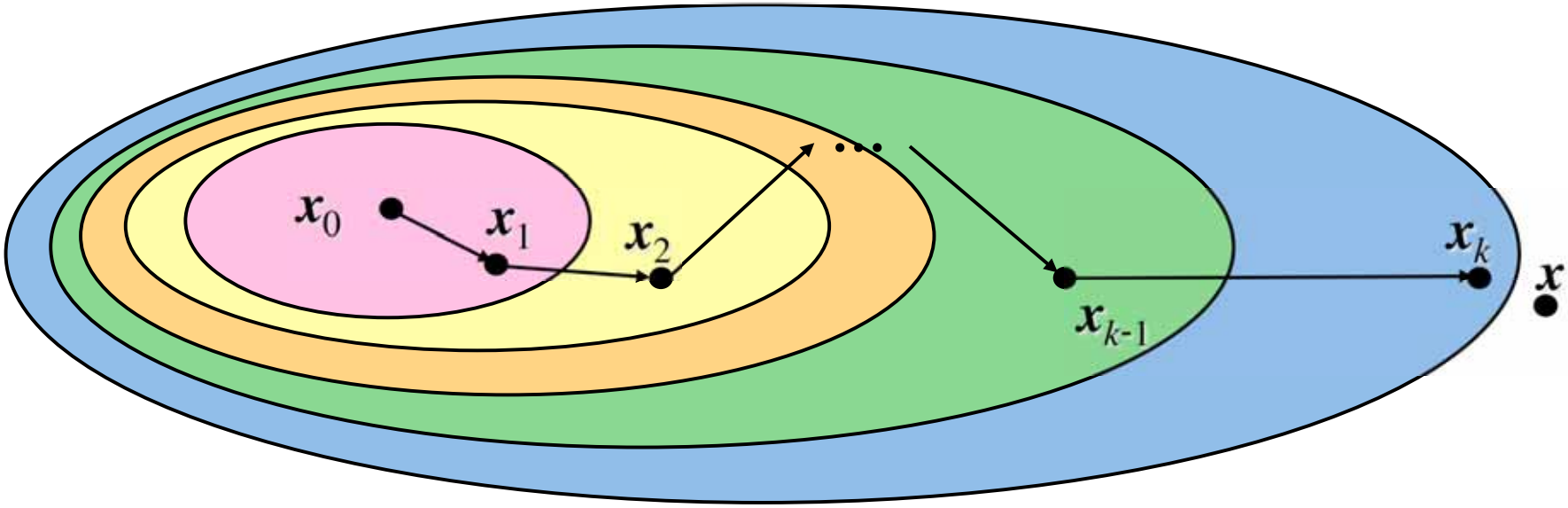
Krylov部分空間反復法

連立一次方程式 $Ax = b$ の近似解生成

近似解の生成条件　$\boldsymbol{x}_k = \boldsymbol{x}_0 + \boldsymbol{z}_k,\ \ \boldsymbol{z}_k \in \mathcal{K}_k(A; \boldsymbol{r}_0)$

Krylov部分空間：　$\mathcal{K}_k(A; \boldsymbol{r}_0) = \mathrm{span}(\boldsymbol{r}_0, A\boldsymbol{r}_0, \ldots, A^{k-1}\boldsymbol{r}_0)$

残差ベクトル　$\boldsymbol{r}_k = \boldsymbol{b} - A\boldsymbol{x}_k = \boldsymbol{r}_0 - A\boldsymbol{z}_k$



Krylov部分空間法の概念図.

Hermite行列に対する解法

1. 係数行列が Hermite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**inal **Res**idual Method: MINRES法）

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

補足：Hermite行列

$$A = A^{\mathrm{H}} = \bar{A}^{\mathrm{T}}$$
$$(a_{ij} = \bar{a}_{ji})$$

Hermite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** $\|r_k\|_2 \le \varepsilon_{\mathrm{TOL}}\|b\|_2$ **do** :

$$q_k = Ap_k,$$ ← 行列ベクトル積

$$\alpha_k = \frac{(r_k, r_k)}{(p_k, q_k)},$$ ← 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

$$\beta_k = \frac{(r_{k+1}, r_{k+1})}{(r_k, r_k)},$$ ← 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

共役勾配法（CG法）

非Hermite行列に対する解法

2. 係数行列が 非Hermite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**-Conjugate **G**radient: BiCG法）
- 自乗共役勾配法（Conjugate **R**esidual **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（Generalized Conjugate **R**esidual: GCR法）
- 一般化最小残差法（Generalized Minimal **Res**idual: GMRES法）

⇒ 残差は単調減少するが，長い漸化式が必要

非Hermite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Hermite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 :**

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

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

$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k \boldsymbol{q}_k,$$

$$\boldsymbol{s}_{k+1} = A\boldsymbol{r}_{k+1},$$

$$\beta_{k,i} = -\frac{(\boldsymbol{q}_i, \boldsymbol{s}_{k+1})}{(\boldsymbol{q}_i, A\boldsymbol{q}_i)}, \quad (i = 0, 1, \ldots, k)$$

$$\boldsymbol{p}_{k+1} = \boldsymbol{r}_{k+1} + \sum_{i=0}^{k} \beta_{k,i} \boldsymbol{p}_i,$$

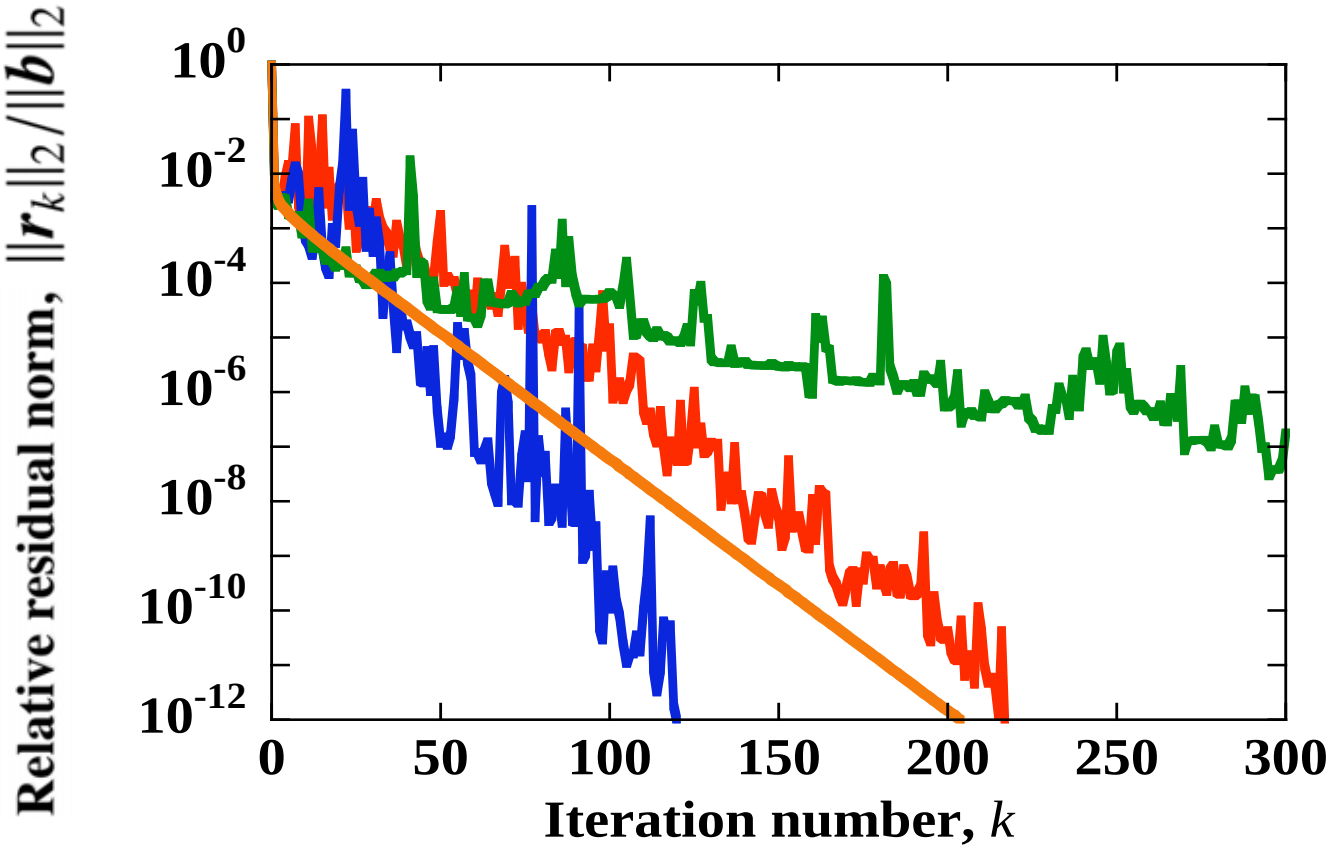
$$\boldsymbol{q}_{k+1} = \boldsymbol{s}_{k+1} + \sum_{i=0}^{k} \beta_{k,i} \boldsymbol{q}_i,$$

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

$\boldsymbol{x}_0$ 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

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

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

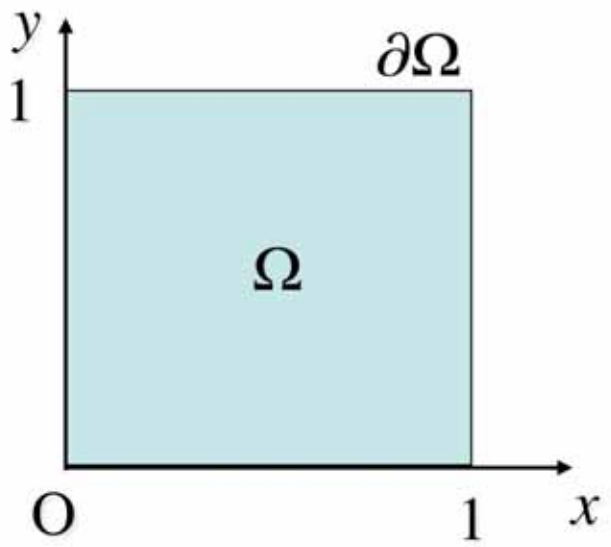
共役直交共役勾配法（COCG法）

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

$M \times M$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

Compressed Row Storage (CRS) 形式
非零要素を行方向に探索

$$A = \begin{bmatrix} 1 & 0 & 3 & 0 & 2 \\ 0 & 1 & 0 & 4 & 6 \\ 2 & 5 & 3 & 0 & 0 \\ 0 & 0 & 7 & 8 & 0 \\ 0 & 1 & 0 & 3 & 9 \end{bmatrix}$$

`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し示す配列

val :	1	3	2	1	4	6	2	5	3	7	8	1	3	9
col_ind :	1	3	5	2	4	5	1	2	3	3	4	2	4	5
row_ptr :	1	4	7	10	12	15								

非零要素数+1 の値を格納

疎行列の格納形式

Compressed Column Storage (CCS) 形式
非零要素を列方向に探索

$$A = \begin{bmatrix} 1 & 0 & 3 & 0 & 2 \\ 0 & 1 & 0 & 4 & 6 \\ 2 & 5 & 3 & 0 & 0 \\ 0 & 0 & 7 & 8 & 0 \\ 0 & 1 & 0 & 3 & 9 \end{bmatrix}$$

`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し示す配列

`val` :	1	2	1	5	1	3	3	7	4	8	3	2	6	9
`row_ind` :	1	3	2	3	5	1	3	4	2	4	5	1	2	5
`col_ptr` :	1	3	6	9	12	15								

非零要素数+1 の値を格納

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 x の積 $y = Ax$

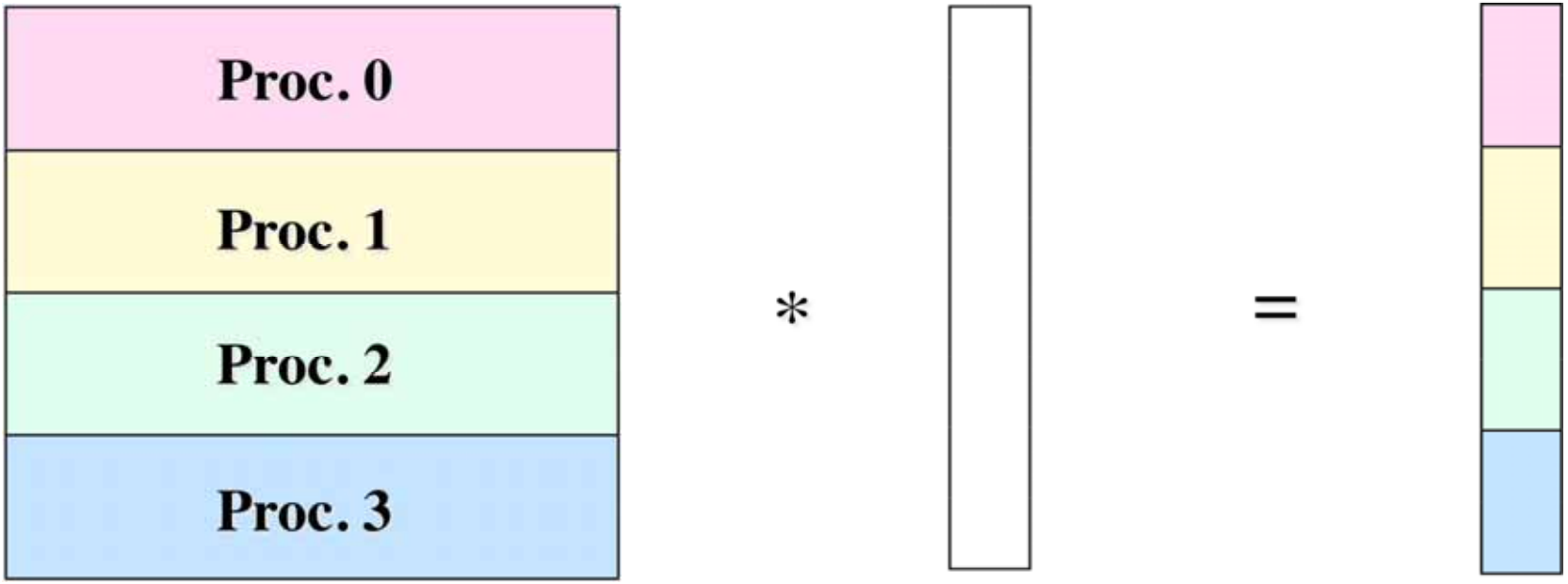
$$y = [a_1, a_2, \ldots, a_n] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \sum_{i=1}^{n} a_i x_i$$

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

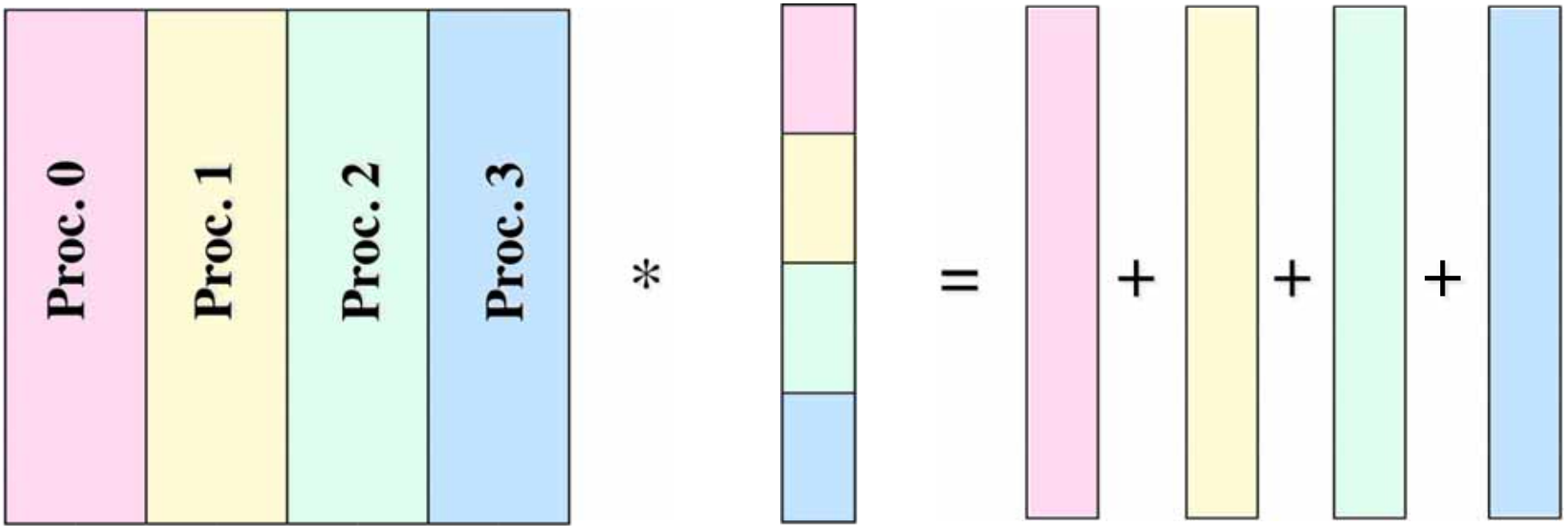
各プロセスで分散してデータを保持する

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

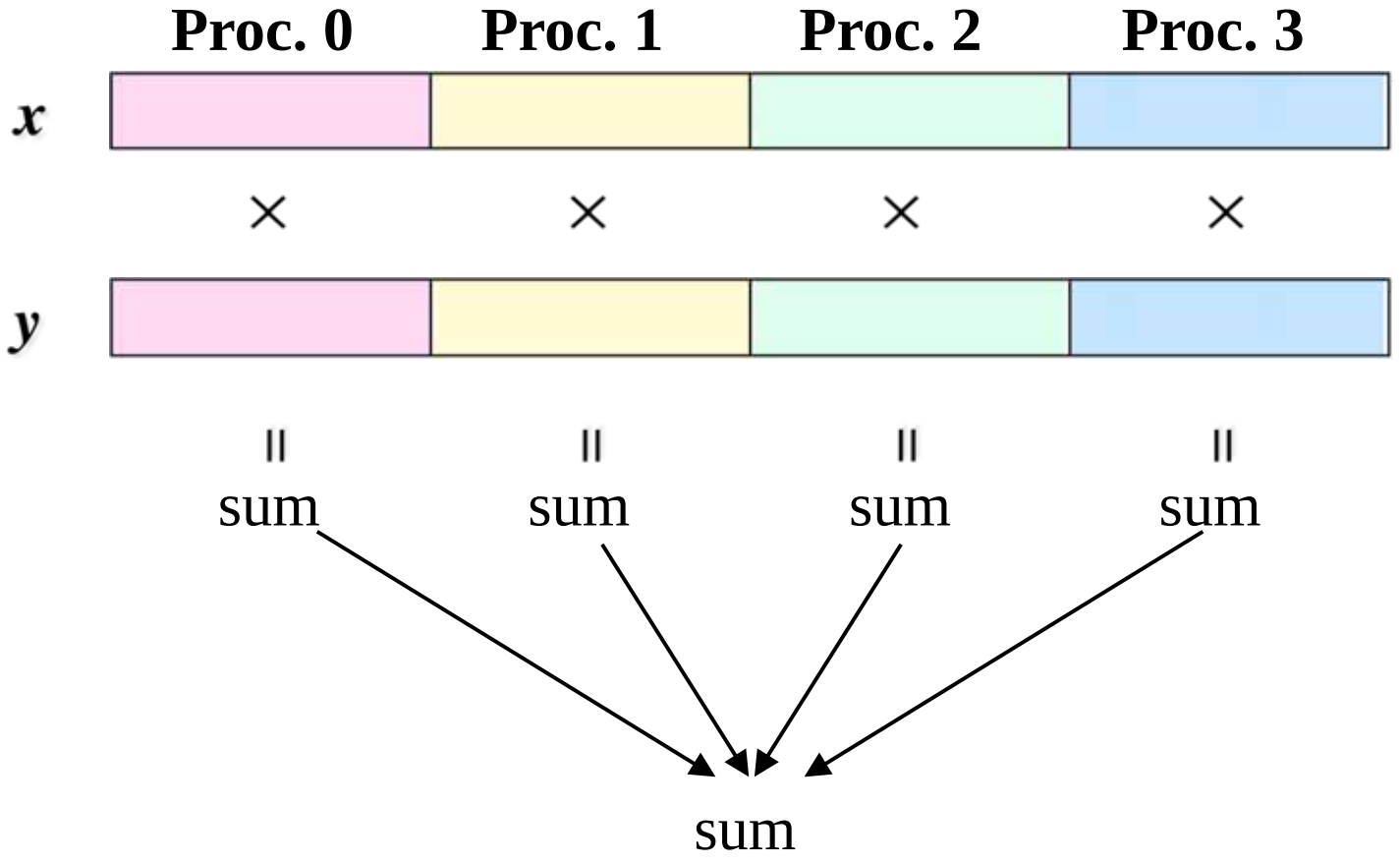
各プロセスで分散してデータを保持する

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

MPIコード例

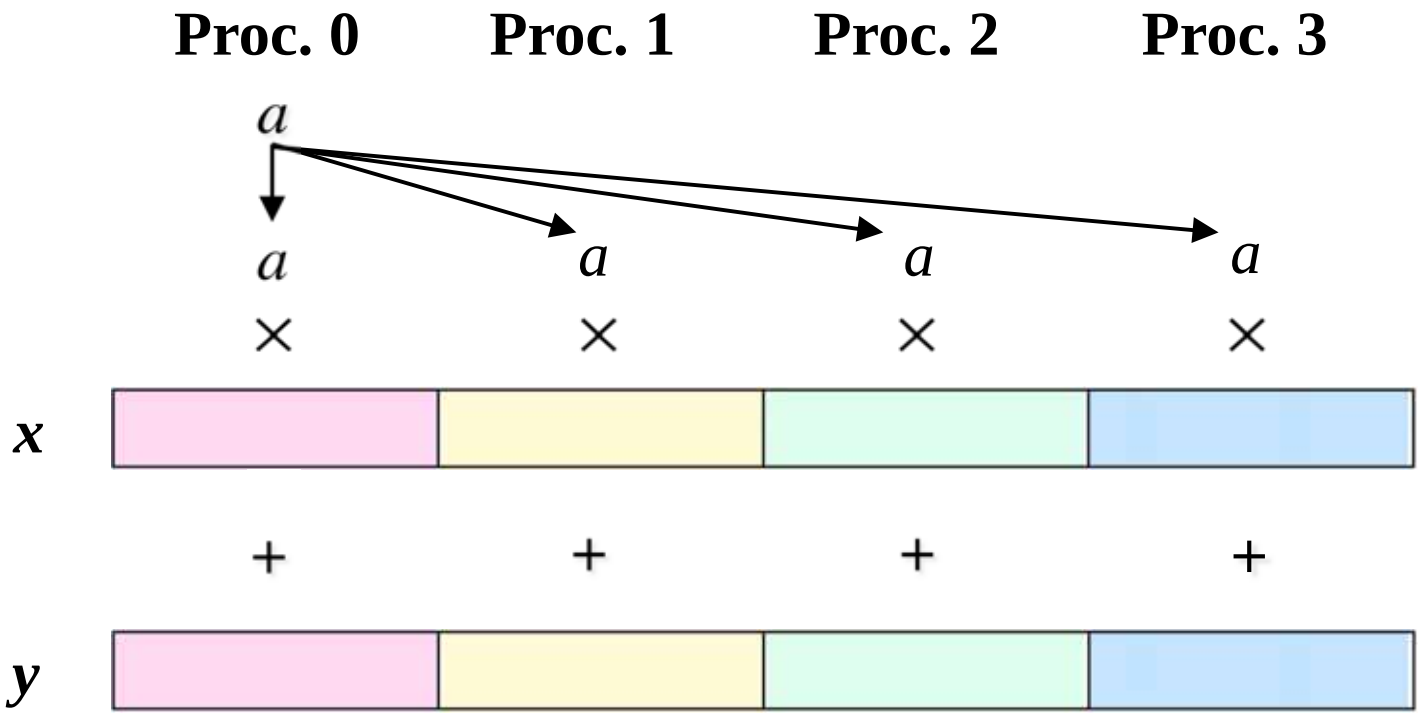
$$(x, y) = \sum_{j=1}^{n} \bar{x}_j y_j$$

```
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

$$y = y + ax$$

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

$$Ax = b \Longleftrightarrow (K_1^{-1} A K_2^{-1})(K_2 x) = K_1^{-1} b$$

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

$$Ax = b \Longleftrightarrow MAx = Mb$$

または $\quad Ax = b \Longleftrightarrow (AM)(M^{-1}x) = b$

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

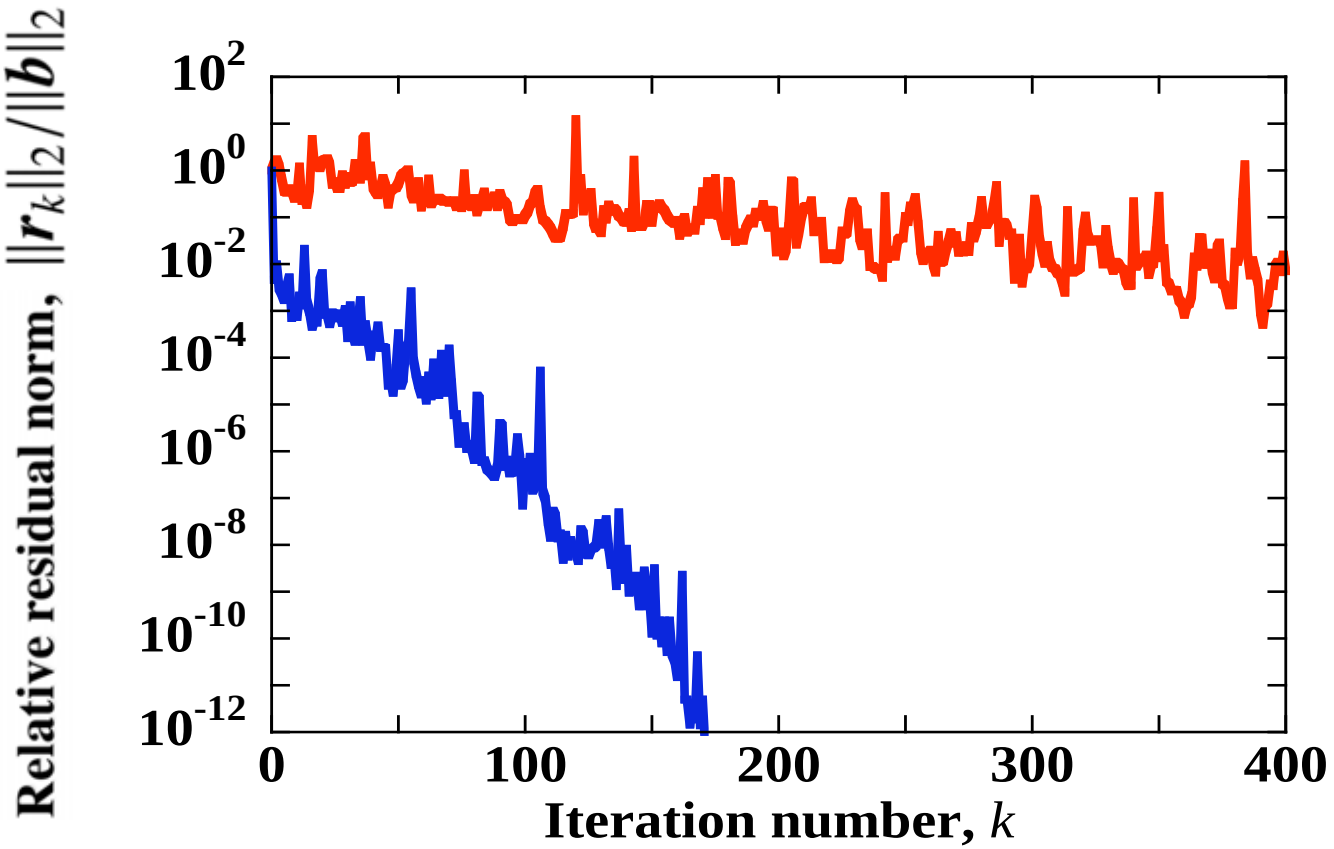
$$A^{-1} = [I - (I - A)]^{-1} = \sum_{j=0}^{\infty} (I - A)^j$$

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

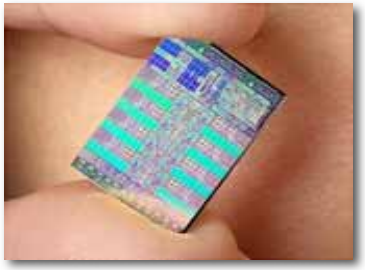


反復法の相対残差履歴

—：BiCG法，—：近似逆行列前処理付きBiCG法

精度混合型 Krylov 部分空間反復法

Cell B/E や GPGPUによる計算



○ Cell Broadband Engine

・1つの Power Processing Element (PPE) と
　8つの Synersistic Processing Element (SPE) から構成
・SPE は単精度計算を高速に行うことができる

○ GPU を用いた計算 (GPGPU)

・多くのストリーミングプロセッサ (SP) を持つ
・Tesla C1060 では 単精度演算で 933 GFLOPS

これらの性能を生かせるアルゴリズムが必要

Krylov部分空間法の前処理

Krylov部分空間反復法の特徴

係数行列 A が単位行列に近い場合，少ない反復回数で収束

連立一次方程式

$Ax = b$

左前処理

$MAx = Mb$

右前処理

$AMy = b,\quad y = M^{-1}x$

M: $MA \approx I$ 及び $AM \approx I$ を満たす前処理行列.

前処理付きBiCGSTAB法

左前処理

x_0 is an initial guess,
compute $r_0 = p_0 = M(b - Ax_0)$,
choose r_0^* such that $(r_0^*, r_0) \neq 0$,
for $k = 0, 1, \ldots,$ **until** $\|r_k\|_2 \leq \varepsilon\|b\|_2$ **do**:
begin

$$
\begin{aligned}
u_k &= Ap_k, \\
v_k &= Mu_k, \\
\alpha_k &= (r_0^*, r_k)/(r_0^*, v_k), \\
t_k &= r_k - \alpha_k v_k, \\
s_k &= At_k, \\
q_k &= Ms_k, \\
\zeta_k &= (q_k, t_k)/(q_k, q_k), \\
x_{k+1} &= x_k + \alpha_k p_k + \zeta_k t_k, \\
r_{k+1} &= t_k - \zeta_k q_k, \\
\beta_k &= (\alpha_k/\zeta_k) \cdot (r_0^*, r_{k+1})/(r_0^*, r_k), \\
p_{k+1} &= r_{k+1} + \beta_k(p_k - \zeta_k v_k),
\end{aligned}
$$

end

前処理部分

右前処理

x_0 is an initial guess,
compute $r_0 = p_0 = b - Ax_0$,
choose r_0^* such that $(r_0^*, r_0) \neq 0$,
for $k = 0, 1, \ldots,$ **until** $\|r_k\|_2 \leq \varepsilon\|b\|_2$ **do**:
begin

$$
\begin{aligned}
u_k &= Mp_k, \\
v_k &= Au_k, \\
\alpha_k &= (r_0^*, r_k)/(r_0^*, v_k), \\
t_k &= r_k - \alpha_k v_k, \\
s_k &= Mt_k, \\
q_k &= As_k, \\
\zeta_k &= (q_k, t_k)/(q_k, q_k), \\
x_{k+1} &= x_k + \alpha_k u_k + \zeta_k s_k, \\
r_{k+1} &= t_k - \zeta_k q_k, \\
\beta_k &= (\alpha_k/\zeta_k) \cdot (r_0^*, r_{k+1})/(r_0^*, r_k), \\
p_{k+1} &= r_{k+1} + \beta_k(p_k - \zeta_k v_k),
\end{aligned}
$$

end

精度混合型アルゴリズム

x_0 is an initial guess,
compute $r_0 = p_0 = b - Ax_0$,
choose r_0^* such that $(r_0^*, r_0) \neq 0$,
for $k = 0, 1, \ldots,$ **until** $\|r_k\|_2 \leq \varepsilon \|b\|_2$ **do**:
begin

$u_k = Mp_k$, 単精度計算
$v_k = Au_k$,
$\alpha_k = (r_0^*, r_k)/(r_0^*, v_k)$,
$t_k = r_k - \alpha_k v_k$,
$s_k = Mt_k$, 単精度計算
$q_k = As_k$,
$\zeta_k = (q_k, t_k)/(q_k, q_k)$,
$x_{k+1} = x_k + \alpha_k u_k + \zeta_k s_k$,
$r_{k+1} = t_k - \zeta_k q_k$,
$\beta_k = (\alpha_k/\zeta_k) \cdot (r_0^*, r_{k+1})/(r_0^*, r_k)$,
$p_{k+1} = r_{k+1} + \beta_k (p_k - \zeta_k v_k)$,

end

倍精度計算

前処理計算部分

$u_k = Mp_k$, $s_k = Mt_k$

多くの演算量が必要！

精度混合アプローチ

- 前処理計算部分
  - → 単精度計算で高速化
- その他の部分
  - → 倍精度計算を行う

誤差の影響（左前処理）

第 k ステップの漸化式

$$
\begin{aligned}
& u_k = Ap_k, \\
[1]\quad & v_k = Mu_k, \\
& \alpha_k = (r_0^*, r_k)/(r_0^*, v_k), \\
[2]\quad & t_k = r_k - \alpha_k v_k, \\
& s_k = At_k, \\
[3]\quad & q_k = Ms_k, \\
& \zeta_k = (q_k, t_k)/(q_k, q_k), \\
[4]\quad & x_{k+1} = x_k + \alpha_k p_k + \zeta_k t_k, \\
& r_{k+1} = t_k - \zeta_k q_k, \\
& \beta_k = (\alpha_k/\zeta_k)\cdot(r_0^*, r_{k+1})/(r_0^*, r_k), \\
& p_{k+1} = r_{k+1} + \beta_k(p_k - \zeta_k v_k),
\end{aligned}
$$

[1] v_k には前処理部分で誤差が混入

$\tilde{v}_k = v_k + \delta v_k$, δv_k : v_k の誤差

[2] α_k, t_k, s_k にも誤差が混入

$\alpha_k \Rightarrow \tilde{\alpha}_k$, $t_k \Rightarrow \tilde{t}_k$, $s_k \Rightarrow \tilde{s}_k$

[3] q_k には前処理部分で誤差が混入

$\tilde{q}_k = q_k + \delta q_k$, δq_k : q_k の誤差

[4] $\zeta_k, x_{k+1}, r_{k+1}$ にも誤差が混入

$\zeta_k \Rightarrow \tilde{\zeta}_k$, $x_{k+1} \Rightarrow \tilde{x}_{k+1}$, $r_{k+1} \Rightarrow \tilde{r}_{k+1}$

記号 ~ は誤差が混入した値を表す

誤差の影響（左前処理）

第 $(k+1)$ 番目の残差ベクトル

$$
\begin{aligned}
\tilde{\boldsymbol{r}}_{k+1} &= \tilde{\boldsymbol{t}}_k - \tilde{\zeta}_k \tilde{\boldsymbol{q}}_k \\
&= \boldsymbol{r}_k - \tilde{\alpha}_k \tilde{\boldsymbol{v}}_k - \tilde{\zeta}_k \tilde{\boldsymbol{q}}_k \\
&= \boldsymbol{r}_k - \tilde{\alpha}_k (\boldsymbol{v}_k + \delta \boldsymbol{v}_k) - \tilde{\zeta}_k (\boldsymbol{q}_k + \delta \boldsymbol{q}_k) \\
&= \boldsymbol{r}_k - \tilde{\alpha}_k (M A \boldsymbol{p}_k + \delta \boldsymbol{v}_k) - \tilde{\zeta}_k (M A \tilde{\boldsymbol{t}}_k + \delta \boldsymbol{q}_k) \\
&= M \left[\boldsymbol{b} - A(\boldsymbol{x}_k + \tilde{\alpha}_k + \tilde{\zeta}_k \tilde{\boldsymbol{t}}_k) \right] - \tilde{\alpha}_k \delta \boldsymbol{v}_k - \tilde{\zeta}_k \delta \boldsymbol{q}_k \\
&= M(\boldsymbol{b} - A \tilde{\boldsymbol{x}}_{k+1}) - \tilde{\alpha}_k \delta \boldsymbol{v}_k - \tilde{\zeta}_k \delta \boldsymbol{q}_k
\end{aligned}
$$

残差と近似解の関係式　$\tilde{\boldsymbol{r}}_{k+1} = M(\boldsymbol{b} - A\tilde{\boldsymbol{x}}_{k+1})$　が満たされない！

誤差の影響（右前処理）

第 k ステップの漸化式

[1] $u_k = Mp_k,$

$v_k = Au_k,$

[2] $\alpha_k = (r_0^*, r_k)/(r_0^*, v_k),$

$t_k = r_k - \alpha_k v_k,$

[3] $s_k = Mt_k,$

$q_k = As_k,$

$\zeta_k = (q_k, t_k)/(q_k, q_k),$

[4] $x_{k+1} = x_k + \alpha_k u_k + \zeta_k s_k,$

$r_{k+1} = t_k - \zeta_k q_k,$

$\beta_k = (\alpha_k/\zeta_k) \cdot (r_0^*, r_{k+1})/(r_0^*, r_k),$

$p_{k+1} = r_{k+1} + \beta_k(p_k - \zeta_k v_k),$

[1] u_k には前処理部分で誤差が混入

$\tilde{u}_k = u_k + \delta u_k$, δu_k : u_k の誤差

[2] v_k, α_k, t_k にも誤差が混入

$v_k \Rightarrow \tilde{v}_k$, $\alpha_k \Rightarrow \tilde{\alpha}_k$, $t_k \Rightarrow \tilde{t}_k$

[3] s_k には前処理部分で誤差が混入

$\tilde{s}_k = s_k + \delta s_k$, δs_k : s_k の誤差

[4] $q_k, \zeta_k, x_{k+1}, r_{k+1}$ にも誤差が混入

$q_k \Rightarrow \tilde{q}_k$, $\zeta_k \Rightarrow \tilde{\zeta}_k$,

$x_{k+1} \Rightarrow \tilde{x}_{k+1}$, $r_{k+1} \Rightarrow \tilde{r}_{k+1}$

誤差の影響（右前処理）

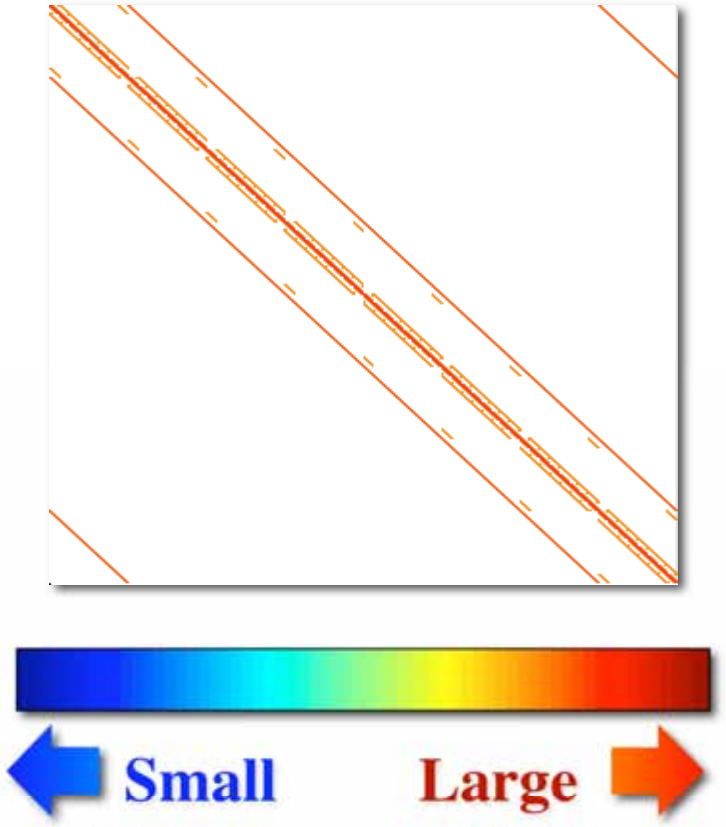
第 $(k+1)$ 番目の残差ベクトル

$$
\begin{aligned}
\tilde{\boldsymbol{r}}_{k+1} &= \tilde{\boldsymbol{t}}_k - \tilde{\zeta}_k \tilde{\boldsymbol{q}}_k \\
&= \boldsymbol{r}_k - \tilde{\alpha}_k \tilde{\boldsymbol{v}}_k - \tilde{\zeta}_k \tilde{\boldsymbol{q}}_k \\
&= \boldsymbol{r}_k - \tilde{\alpha}_k A\tilde{\boldsymbol{u}}_k - \tilde{\zeta}_k A\tilde{\boldsymbol{s}}_k \\
&= \boldsymbol{b} - A(\boldsymbol{x}_k + \tilde{\alpha}_k \tilde{\boldsymbol{u}}_k + \tilde{\zeta}_k \tilde{\boldsymbol{s}}_k) \\
&= \boldsymbol{b} - A\tilde{\boldsymbol{x}}_{k+1}
\end{aligned}
$$

残差と近似解の関係式　$\tilde{\boldsymbol{r}}_{k+1} = \boldsymbol{b} - A\tilde{\boldsymbol{x}}_{k+1}$　が満たされる

数値実験

量子色力学（QCD）で現れる連立一次方程式



連立一次方程式 : $Ax = b$

係数行列 A

$$A = I_n - \kappa \begin{bmatrix} D_{00} & D_{01} \\ D_{10} & D_{11} \end{bmatrix}$$

I_n : $n \times n$ 単位行列,

κ : スカラーパラメータ.

右辺ベクトル b

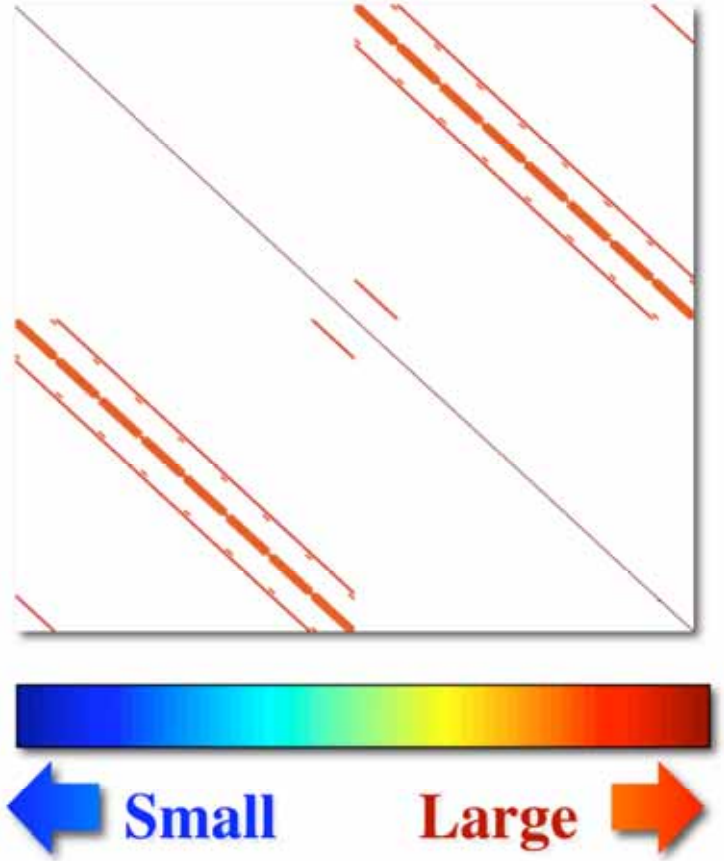
$$b = [5.4, 5.4, \ldots, 5.4]^{\mathrm{T}}$$

行列 A の非零構造　(conf5.4-00l8x8-1000).

A のサイズ : 49, 152, nnz(A) : 196, 6080.

数値実験

Red-Black オーダリング



係数行列 A

$$A = I_n - \kappa \begin{bmatrix} D_{00} & D_{01} \\ D_{10} & D_{11} \end{bmatrix}$$

RB オーダリング

並べ替え後の行列

$$A' = I_n - \kappa \begin{bmatrix} O & D'_{01} \\ D'_{10} & O \end{bmatrix}$$

並べ替え後の非零構造　(conf5.4-00l8x8-1000).

A のサイズ : 49, 152, nnz(A) : 196, 6080.

数値実験

SSOR 前処理

$$A' = I_n - \kappa \begin{bmatrix} O & D'_{01} \\ D'_{10} & O \end{bmatrix}$$

SSOR 前処理

$$\hat{A} = (I_n + L)^{-1} A' (I_n + U)^{-1} = I_n - \kappa^2 \begin{bmatrix} O & O \\ O & D'_{10} D'_{01} \end{bmatrix}$$

前処理後の連立一次方程式

$$\begin{bmatrix} I_{n/2} & O \\ O & I_{n/2} - \kappa^2 D'_{10} D'_{01} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}}^{(0)} \\ \hat{\boldsymbol{x}}^{(1)} \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{b}}^{(0)} \\ \hat{\boldsymbol{b}}^{(1)} \end{bmatrix}$$

解くべき連立一次方程式は

$$(I_{n/2} - \kappa^2 D'_{10} D'_{01}) \hat{\boldsymbol{x}}^{(1)} = \hat{\boldsymbol{b}}^{(1)}$$

となる.

多項式前処理

行列の Neumann 展開

行列 A が $\|I - A\| < 1$

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

A^{-1} は A の多項式で表される.

多項式による前処理行列

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

m : 前処理行列 M の多項式の次数.

数値実験

実験環境

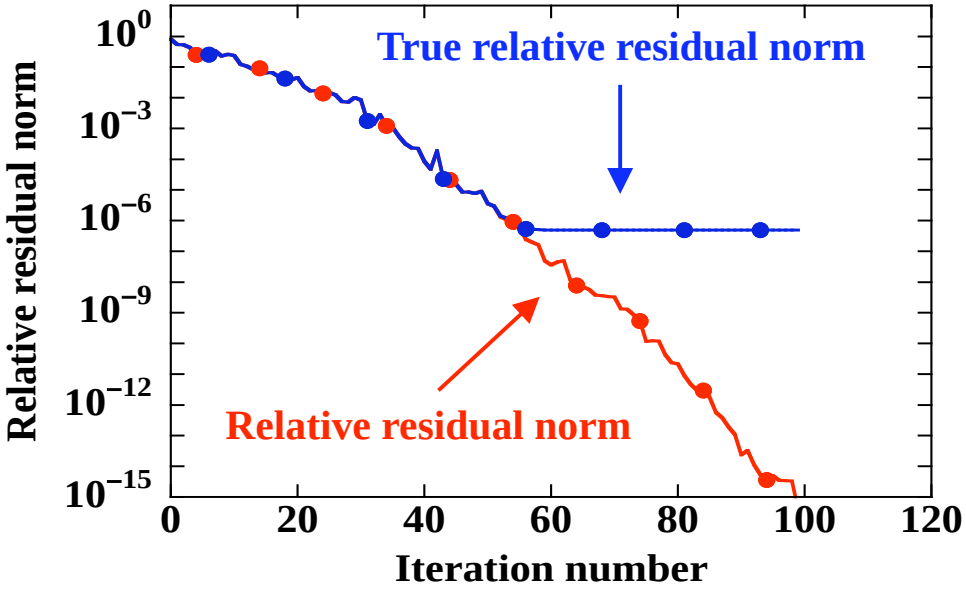
CPU	Intel Core 2 Duo 2.4GHz
Memory	4.0GBytes
OS	Mac OS X ver. 10.5.5
Compiler	Intel Fortran ver. 10.1
Compile option	`-O3 -xT`

実験条件

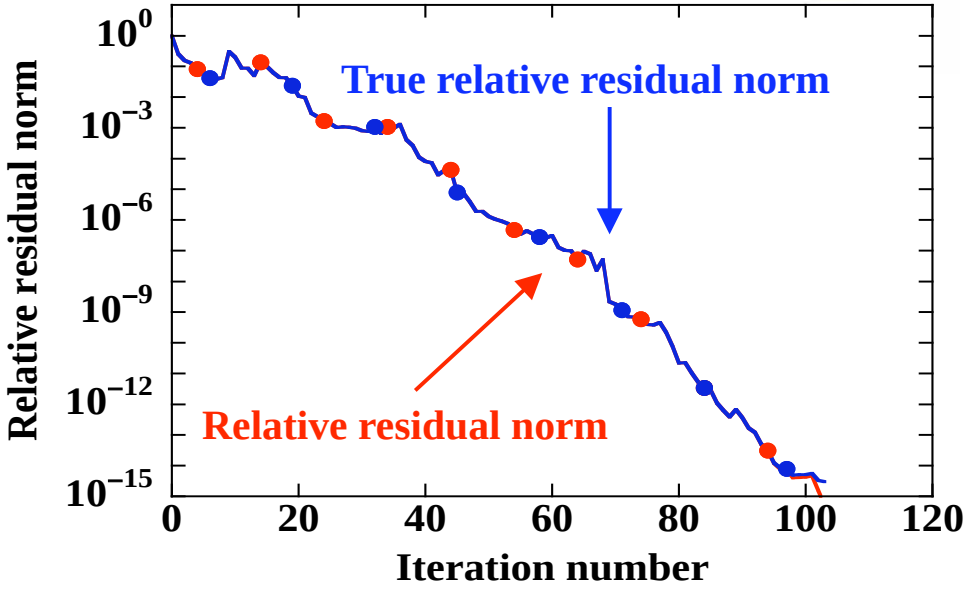
Solver	BiCGSTAB
Preconditioner	Polynomial preconditioner
Initial solution x_0	$\mathbf{0}$
Shadow residual r_0^*	r_0
Stopping criterion	$\|r_k\|_2 / \|\hat{b}^{(1)}\|_2 \leq 1.0 \times 10^{-15}$

前処理部分は全て単精度によって計算を行った.

実験結果



(a) 左前処理の場合.



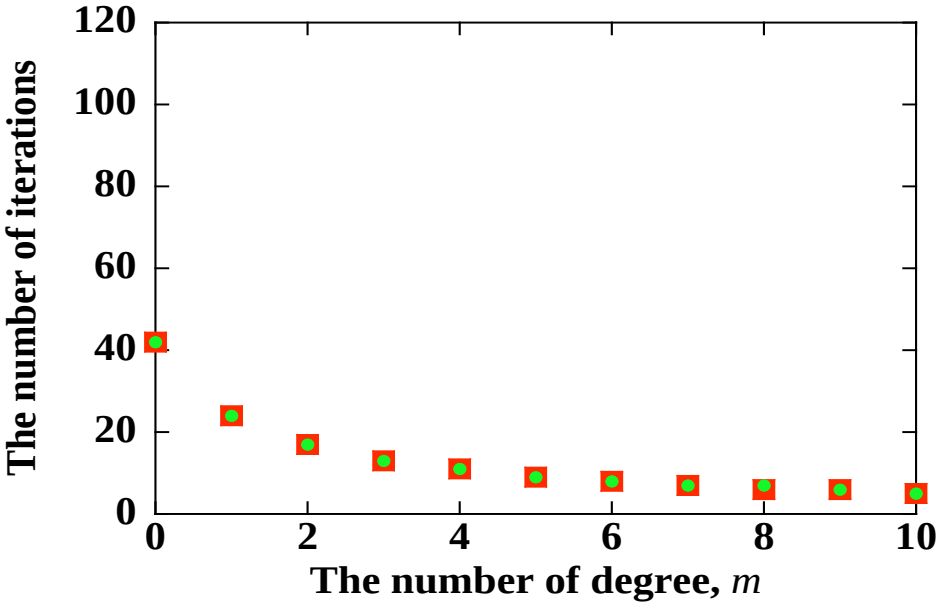
(b) 右前処理の場合.

相対残差履歴（$\kappa = 0.182$, 次数 $m = 5$）.

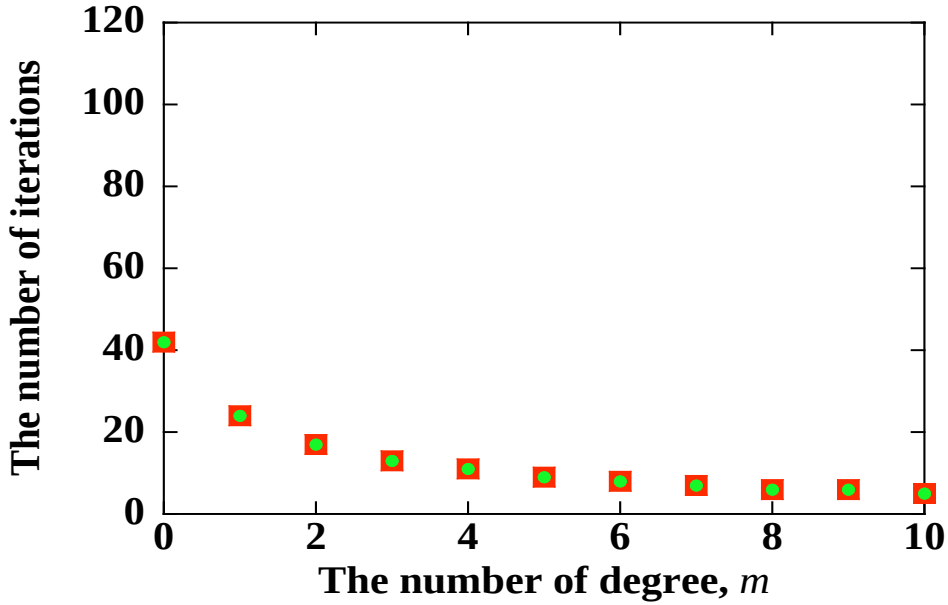
● : 相対残差ノルム（反復中の残差ベクトルから計算）.

● : 真の相対残差ノルム（近似解から$\|\boldsymbol{b} - A\boldsymbol{x}_k\|_2 / \|\boldsymbol{b}\|_2$を計算）.

実験結果



(a)　左前処理.

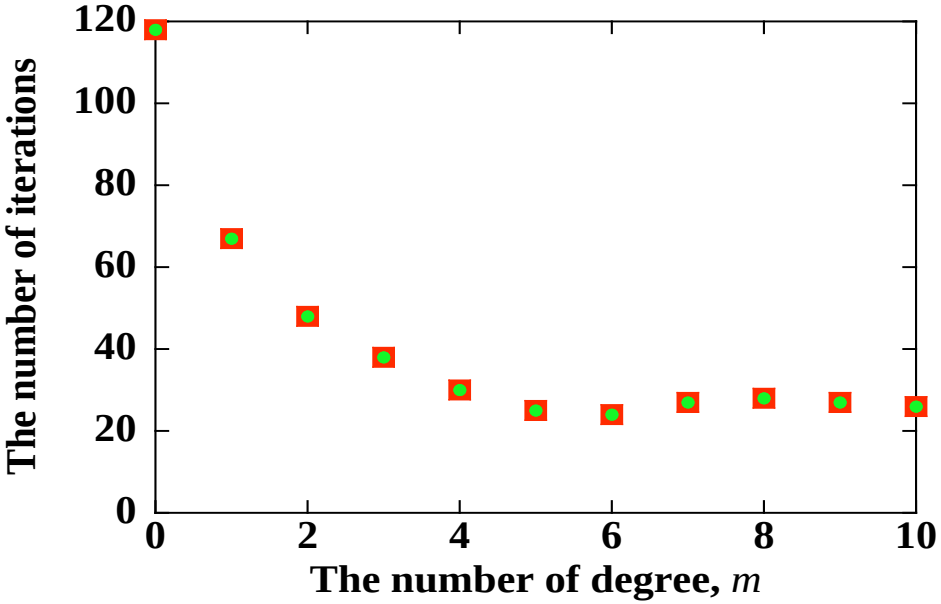


(b)　右前処理.

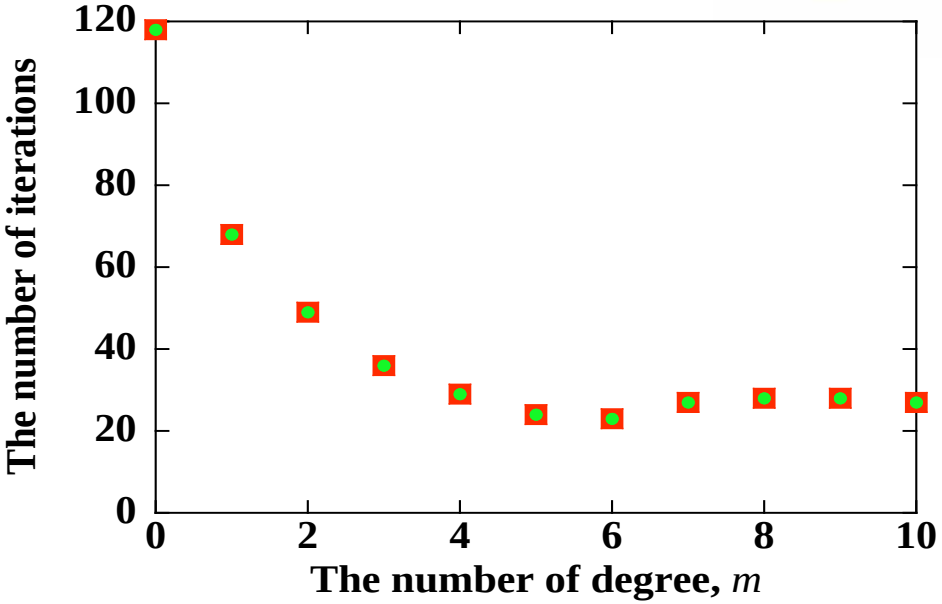
単精度前処理と倍精度前処理の反復回数比較（$\kappa = 0.15$）.

■：単精度前処理,　●：倍精度前処理.

実験結果



(a)　左前処理.

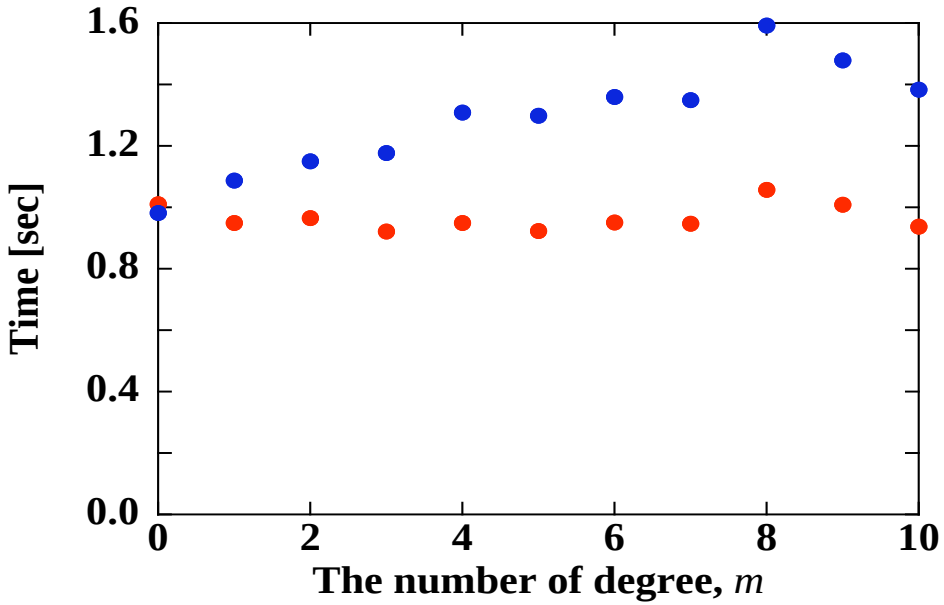


(b)　右前処理.

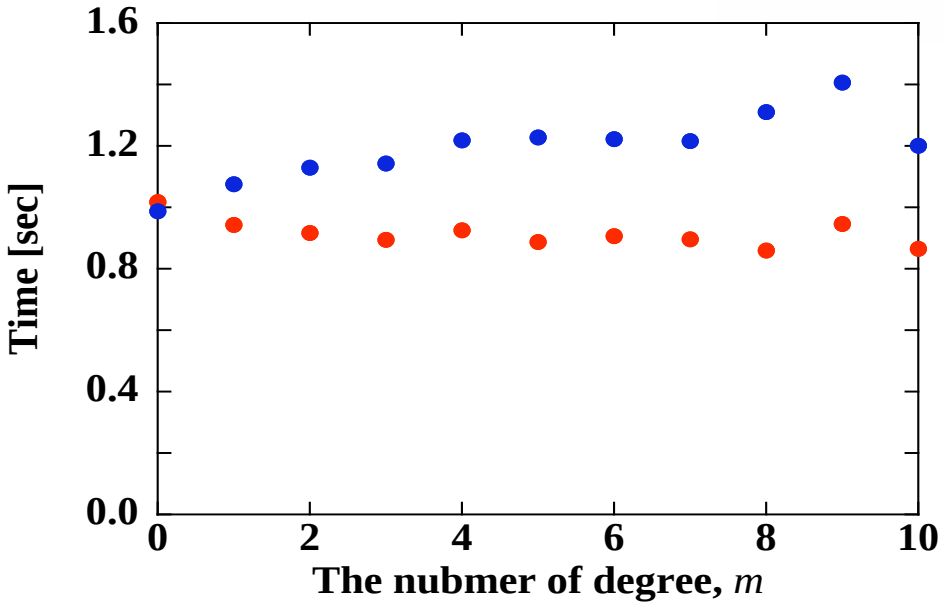
単精度前処理と倍精度前処理の反復回数比較（$\kappa = 0.17$）.

■：単精度前処理,　●：倍精度前処理.

実験結果



(a)　左前処理.

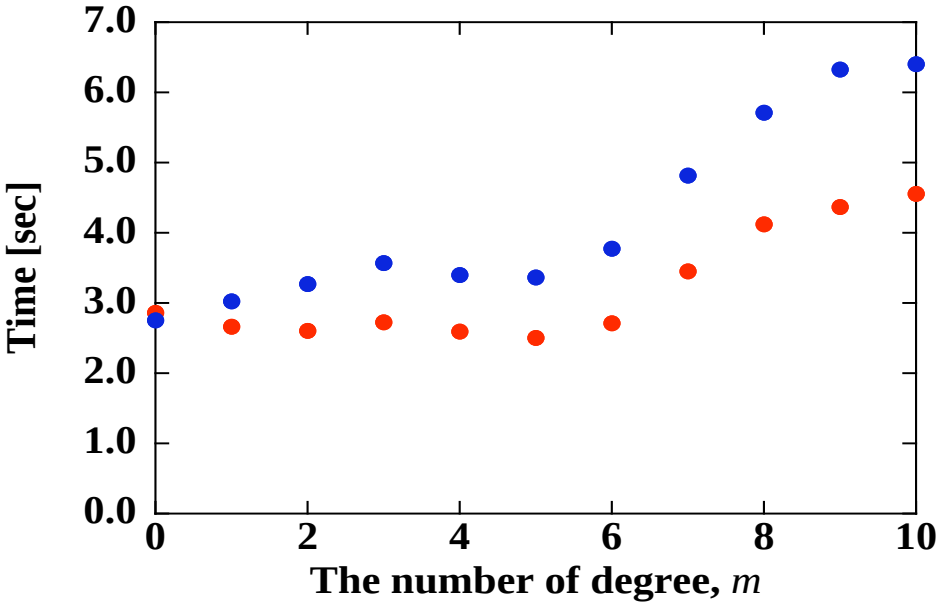


(b)　右前処理.

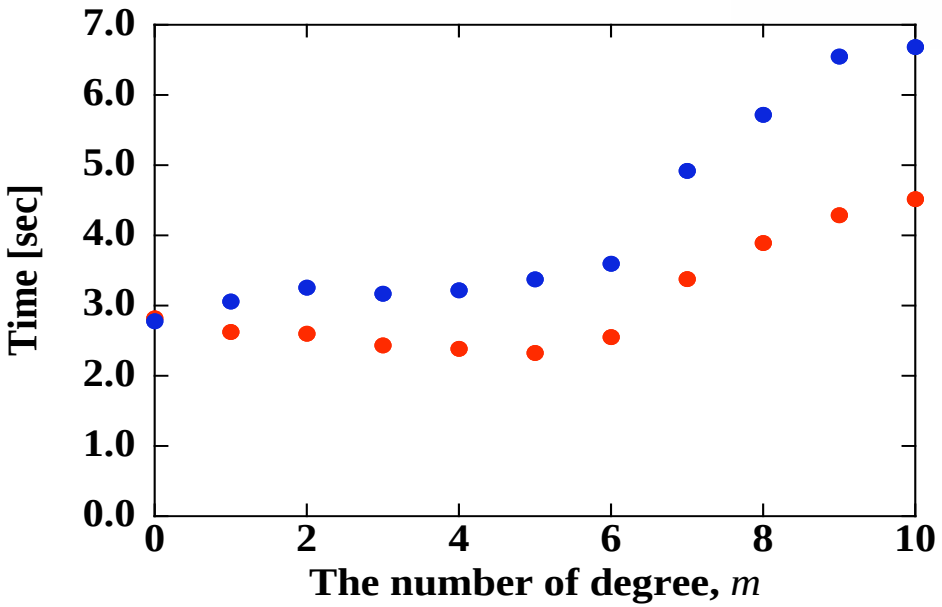
単精度前処理と倍精度前処理の計算時間比較（$\kappa = 0.15$）.

●：単精度前処理,　●：倍精度前処理.

実験結果



(a)　左前処理.

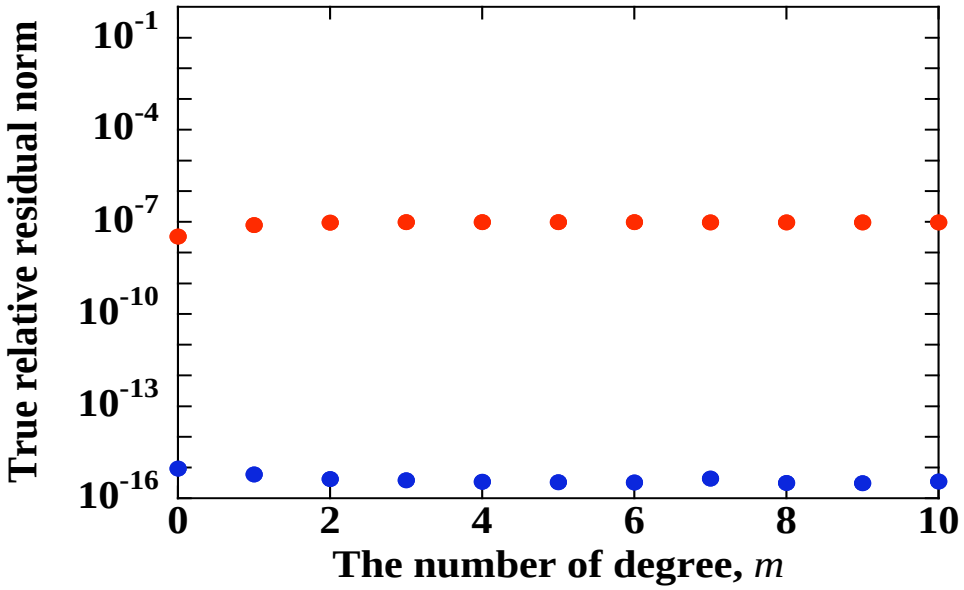


(b)　右前処理.

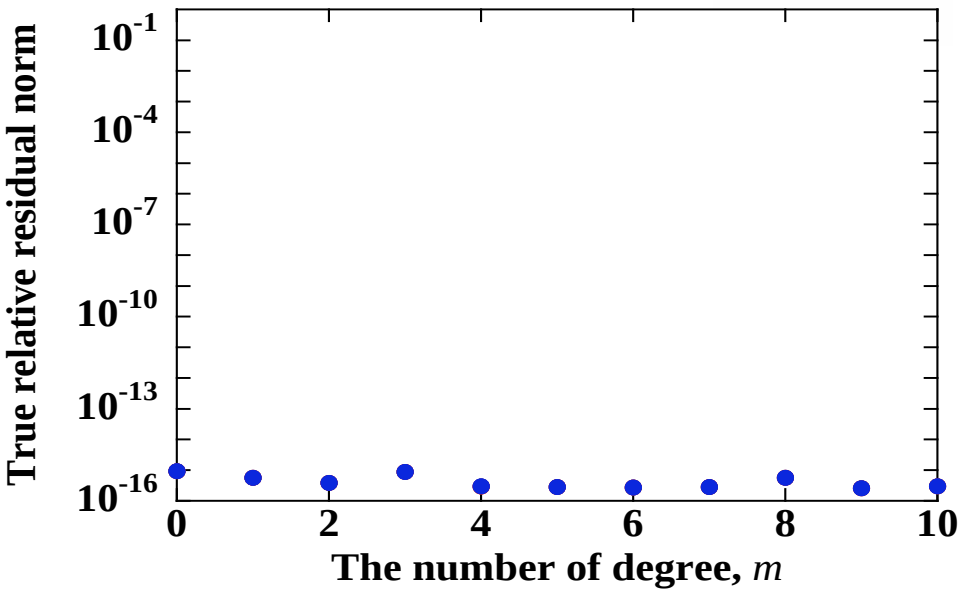
単精度前処理と倍精度前処理の計算時間比較（$\kappa = 0.17$）.

●：単精度前処理,　●：倍精度前処理.

実験結果



(a)　左前処理.

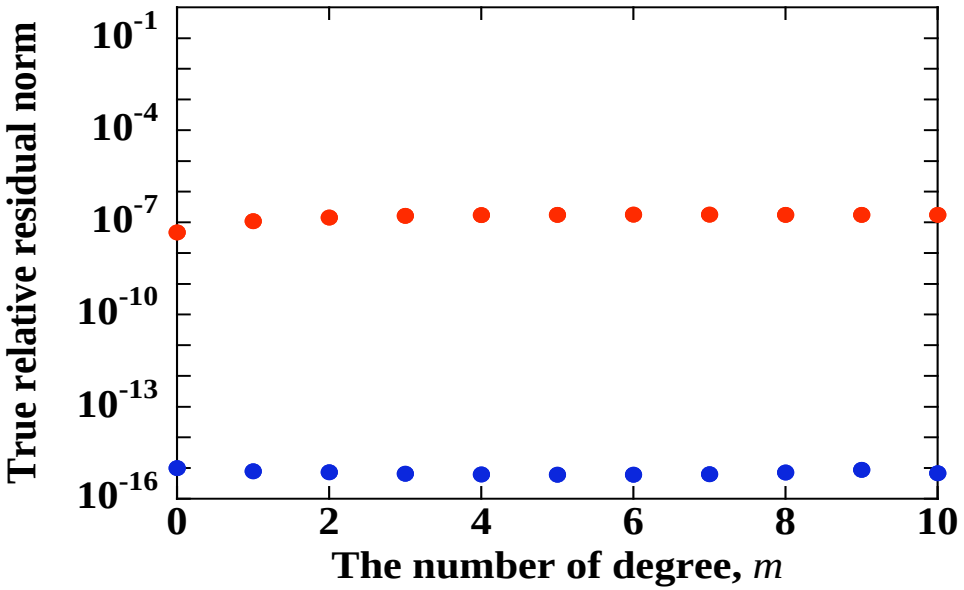


(b)　右前処理.

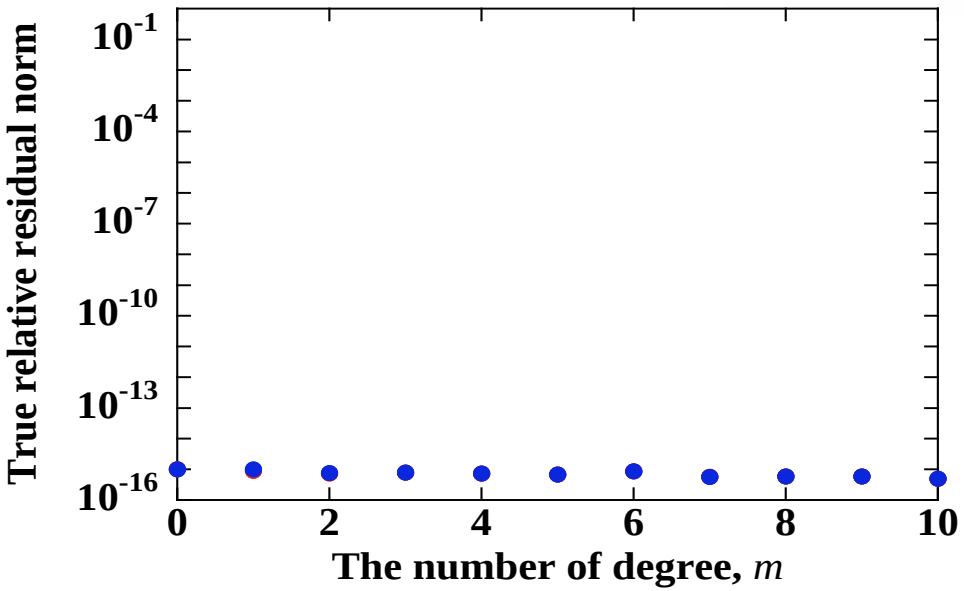
単精度前処理と倍精度前処理の真の相対残差（$\kappa = 0.15$）.

●：単精度前処理,　●：倍精度前処理.

実験結果



(a)　左前処理.



(b)　右前処理.

単精度前処理と倍精度前処理の真の相対残差（$\kappa = 0.17$）.

●：単精度前処理,　●：倍精度前処理.

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.
- Krylov 部分空間反復法における，精度混合型アルゴリズムについて述べた.

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れ．

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れ．

3. 双共役勾配法のプログラムを作り，連立一次方程式$A\boldsymbol{x} = \boldsymbol{b}$ を解け．但し，$\boldsymbol{b} = A[1, 1, \ldots, 1]^{\mathrm{T}}$ とする．行列のパラメータは，$0 < \gamma < 1$ に設定せよ．$\|\boldsymbol{r}_k\|_2 / \|\boldsymbol{b}\|_2 \leq 1.0 \times 10^{-10}$ を満たしたら反復を停止せよ．

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せよ．