

k 段飛ばし MrR 法の数値特性と並列化効率の数値的検証

森下 貴康[†] 董 然^{††} 阿部 邦美^{†††} 生野 壮一郎^{††}

東京工科大学大学院バイオ・情報メディア研究科コンピュータサイエンス専攻[†]

東京工科大学コンピュータサイエンス学部^{††}

岐阜聖徳学園大学経済情報学部^{†††}

1. はじめに

物理的・工学的現象を数値的に再現する際、現象を定式化し偏微分方程式の初期値境界値問題に帰着させ、有限要素法や有限差分法で離散化することで、大規模疎行列を係数行列にもつ連立1次方程式を得る。同方程式を数値的に解くことにより現象を数値的に解析することとなる。一般に、大規模疎行列を係数行列にもつ連立1次方程式の解法には共役勾配法（CG 法）などの Krylov 部分空間解法が適用される。CG 法は行列ベクトル積、ベクトルの内積、ベクトルの加減算で構成されておりアルゴリズムがシンプルであることから並列化が容易であるという利点がある一方で、並列化された内積演算では計算ノード間の物理的な通信上の制約から並列化効率の劣化が起こることが知られている。これらの難点を改善するために、Krylov 部分空間解法に対する様々な通信回避技術が提案されている。

Mister R 法（MrR 法）は共役残差法（CR 法）と数学的に等価な解法であるが、アルゴリズムおよび実装方法が異なる手法であり、CR 法と比較して収束特性が良いことが示されている[1]。

本研究の目的は、単調減少性の収束特性をもつ MrR 法に対して通信回避技術の一つである k 段飛ばし CG 法の実装方法を適用し、その収束特性を数値的に検証することである[2]。また、 k 段飛ばし MrR 法をメモリ分散型並列計算機に実装し、並列化効率の評価を行う。

2. k 段飛ばし MrR 法

k 段飛ばしアルゴリズムの基本概念は、各反復内で計算する Krylov 基底を事前に計算することで、反復をスキップすることができることである。即ち、各反復内で計算される内積演算を事前に求めた基底を用いてスカラー値として扱うことで、並列化による集団通信を回避するのである。

図1に k 段飛ばし MrR 法のアルゴリズムを示す。 k 段飛ばし MrR 法は通常の MrR 法に現れる3つの内積演算 $(\mathbf{r}_{n+1}, \mathbf{A}\mathbf{r}_{n+1})$, $(\mathbf{y}_{n+1}, \mathbf{A}\mathbf{r}_{n+1})$, $(\mathbf{y}_{n+1}, \mathbf{A}\mathbf{y}_{n+1})$ を更新式：

$$\mathbf{y}_{n+1} = \eta_n \mathbf{y}_n + \zeta_n \mathbf{A}\mathbf{r}_n$$

$$\mathbf{r}_{n+1} = \mathbf{r}_n + \mathbf{y}_{n+1}$$

を用いて展開し、事前に求めた Krylov 基底に置き換えることで導出することができる。

3. 評価と考察

本研究では、九州大学情報基盤研究開発センタースーパーコンピュータシステム ITO サブシステム B を4ノード用いて行う[3]。各アルゴリズムの実装は Python 3.6.2 を用いている。また、スレッド並列の実装は CuPy 9.6.0 を使い、CuPy より CUDA 10.1 を呼び出し実行する。またノード間のプロセス並列の実装は mpi4py 3.1.3 を用いており、同実装から OpenMPI 3.1.3 と Intel C++ Compiler 18.3、または MVAPICH2 gdr2.3 と GCC 4.8.5 を呼び出し実行する。マルチプロセス実行時の総プロセス数は16または8とし、それぞれスレッドごとに1または2台のGPUを割り当てており、また、GPU内のデータはCUDA-awareで転送される。

対象問題には、MatrixMarket から取得した有限要素法で円筒胴を離散化した次元数 $N = 5489$ を係数行列とする連立1次方程式を用いる[4]。初期解は零ベクトルとし、収束判定子を 10^{-6} 、最大

Numerical Evaluation of Parallelization Efficiency of k -skip MrR

[†] T. Morishita: Tokyo University of Technology Graduate School

^{††} R. Dong, S. Ikuno: School of Computer Science, Tokyo University of Technology

^{†††} K. Abe: Faculty of Economics and Information, Gifu Shotoku University

反復回数は次元数の N 回とする。また、評価に用いたスキップ数は $k = 0, 2, 4, 6, 8$ とする。ここで、 $k = 0$ は通常の MrR 法であることに注意されたい。

図 3, 4 に k 段飛ばし MrR 法を OpenMPI と MVAPICH を用いて k ごとの収束までの実行時間を示す。同図よりわかるように、OpenMPI の実装と比較して MVAPICH の実装の方が実行時間がかかっていることがわかる。また、何の実装でも MPI プロセス数による実行時間に違いがあることがわかる。これらの結果より、アーキテクチャに則したプロセス数の選択が必須であることがわかる。加えて、 k の増加とともに実行時間が増加しているが、その増加率は一定に漸近していることから、問題の大規模化及び並列分散数の増加によって、通信回避の効果は向上すると考えられる。

```

Let  $\mathbf{x}_0$  be an initial guess.
Set  $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0, \zeta_0 = \frac{(\mathbf{r}_0, A\mathbf{r}_0)}{(A\mathbf{r}_0, A\mathbf{r}_0)}$ ,
 $\mathbf{y}_1 = \zeta_0 A\mathbf{r}_0, \mathbf{z}_1 = -\zeta_0 \mathbf{r}_0, \mathbf{r}_1 = \mathbf{r}_0 - \mathbf{y}_1, \mathbf{x}_1 = \mathbf{x}_0 - \mathbf{z}_1$ 
for  $n = 1, 2, \dots$ , until  $\|\mathbf{r}_n\|_2 / \|\mathbf{b}\|_2 \leq \epsilon$  do
  calculate  $A\mathbf{r}_n, A^2\mathbf{r}_n, \dots, A^{k+1}\mathbf{r}_n$ 
  calculate  $A\mathbf{y}_n, A^2\mathbf{y}_n, \dots, A^k\mathbf{y}_n$ 
   $\alpha_{n,0} = (\mathbf{r}_n, \mathbf{r}_n), \dots, \alpha_{n,2k+2} = (\mathbf{r}_n, A^{2k+2}\mathbf{r}_n)$ 
   $\beta_{n,0} = 0, \dots, \beta_{n,2k+1} = (\mathbf{y}_n, A^{2k+1}\mathbf{r}_n)$ 
   $\delta_{n,0} = (\mathbf{y}_n, \mathbf{y}_n), \dots, \delta_{n,2k} = (\mathbf{y}_n, A^{2k}\mathbf{y}_n)$ 
  for  $i = n, n+1, \dots, n+k$  do
     $\sigma_i = \alpha_{i,2} \delta_{i,0} - \beta_{i,1}^2$ ,
     $\zeta_i = \alpha_{i,1} \delta_{i,0} / \sigma_i, \eta_i = -\alpha_{i,1} \beta_{i,1} / \sigma_i$ 
     $\mathbf{y}_{i+1} = \eta_i \mathbf{y}_i + \zeta_i A\mathbf{r}_i, \mathbf{z}_{i+1} = \eta_i \mathbf{z}_i - \zeta_i \mathbf{r}_i$ 
     $\mathbf{r}_{i+1} = \mathbf{r}_i - \mathbf{y}_{i+1}, \mathbf{x}_{i+1} = \mathbf{x}_i - \mathbf{z}_{i+1}$ 
    for  $j = 0, 1, \dots, 2(k-i)$  do
       $\delta_{i+1,j} = \eta_i^2 \delta_{i,j} + 2\eta_i \zeta_i \beta_{i,j+1} + \zeta_i^2 \alpha_{i,j+2}$ 
       $\tau_j = \eta_i \beta_{i,j} + \zeta_i \alpha_{i,j+1}$ 
       $\beta_{i+1,j} = \tau_j - \delta_{i+1,j}$ 
       $\alpha_{i+1,j} = \alpha_{i,j} - \tau_j - \zeta_i \alpha_{i,j+1}$ 
    end for
  end for
end for

```

図 1: k 段飛ばし MrR 法のアルゴリズム

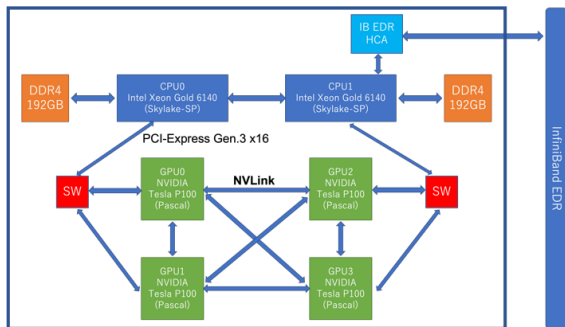


図 2: ITO ハードウェア構成の概念図

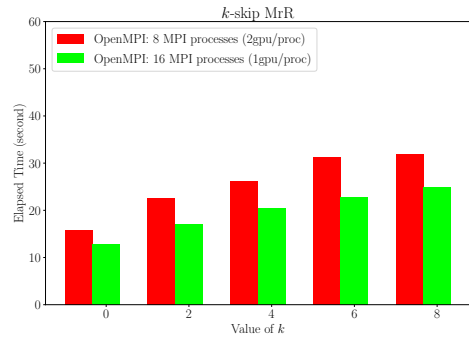


図 3: OpenMPI を用いた場合の k に対する実行時間ただし、赤: 8MPI プロセス、緑: 16MPI プロセス。

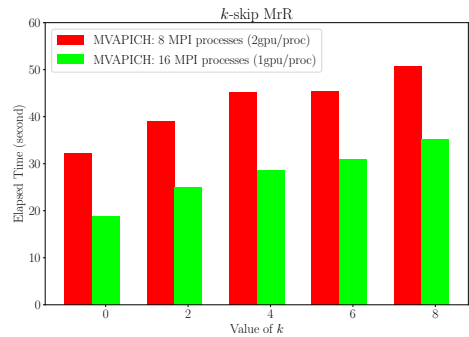


図 4: MVAPICH を用いた場合の k に対する実行時間。ただし、赤: 8MPI プロセス、
参考文献

1. 阿部 邦美, 藤野 清次, 線型方程式を解くための MrR (ミスター R) 法について, 情報処理学会研究 報告, 2016.
2. 本谷 徹, 須田 礼二, k 段飛ばし共役勾配法: 通信を回避することで大規模並列計算で有効な対称正定 値疎行列連立 1 次方程式の反復解法, 情報処理学 会研究報告, Vol.2012-HPC-133 No.30.
3. 九州大学情報基盤研究開発センター, ITO 紹介, https://www.cc.kyushu-u.ac.jp/scp/system/ITO/01_intro.html
4. S1RMQ4M1: Finite element analysis of cylindrical shellsCylindrical shell <https://math.nist.gov/MatrixMarket/data/misc/cylshell/s1rmq4m1.html>