Vol.2018-HPC-167 No.28
2018/12/18

建物の地震動応答シミュレーションに現れる前処理付き共役勾配法の並列化

後藤 啓[1,a)]　横川 三津夫[1]　坂 敏秀[2]

概要：地震の多い我が国では建築物に対する耐震性の要求が高い．建築物の耐震性を調べるために，建物・地盤の地震動応答を求める数値シミュレーションが行われている．本研究で扱うシミュレーションコードは建物と地盤を 3 次元有限要素法で離散化し，各節点について立てられた運動方程式に平均加速度法を適用したものである．そうして得られた連立一次方程式を前処理付き共役勾配法の一種である PSCCG 法で解いている．本研究では，本シミュレーションの実行時間の大部分を占める PSCCG 法のプロセス並列化を行った．スレッド数を 1 に固定しプロセス数を増やしながら，プロセス並列の実装評価を行った．その結果 2 プロセスでの実行時間を基準として，8 プロセスでは最大で 3.2 倍の速度向上が確認できた．

キーワード：地震動応答シミュレーション，前処理付き共役勾配法，Xeon Phi^{TM}（KNL），並列計算

1. 序論

地震の多い我が国では建築物の耐震性への要求が高い．社会的に重要な建築物の耐震性を確保するためには，地盤の影響を考慮した上で地震時の建物応答を適切に評価し，設計基準を満たしていることを確認する必要がある．その手段のひとつに，地盤を有限要素法でモデル化して，建物の地震動応答を求める数値解析法がある．この方法では有限要素法で地盤を詳細にモデル化するため，解析モデル全体の総自由度が増大し，計算時間が長時間化する傾向がある．この解析モデルをなるべく短時間に精度良く計算できることが必要である．

地震動シミュレーションコードのひとつに，坂・小磯らによって開発された地震動シミュレーション [1] がある．空間方向の離散化には 3 次元有限要素法が用いられており，時間方向の離散化には平均加速度法 [2] が用いられている．3 次元有限要素法によって離散化された建築物と地盤の運動は，各要素毎の運動方程式からなる連立一次方程式で表される．

連立一次方程式の解法は大きく直接法と反復法の 2 つに分類される．直接法は有限回の演算で解が求まるという利点がある．また時間発展型の問題に適用する場合には，時間発展ごとに変化するのが右辺のみで，対象とする連立一次方程式の係数行列が時刻によって不変であれば，コレスキー分解などを用いることで 2 回目以降の求解を高速に行うことができる．一方，反復法は直接法に比べて必要なメモリ容量が小さい．また時間発展型の問題に適用する場合には，各時間ステップにおける反復法の初期値として解に近いものが得られていることが多いため，計算時間の観点からも有効である．

さて，地震動シミュレーションでは地盤領域を広くとり，それを詳細にモデル化する．しかし大規模なモデルになると長い計算時間と大容量のメモリが必要になる．また本シミュレーションで得られる連立一次方程式の係数行列は実対称正定値行列である．以上より本シミュレーションには反復法の共役勾配法（Conjugate gradient method，CG 法）が適していると考えられる．

疎な実対称正定値行列を係数行列とする連立一次方程式には，前処理付き共役勾配法（Preconditioned CG method，PCG 法）を用いることが多い．本シミュレーションコードでは，建物要素と建物・地盤の接するインターフェース層からなる部分行列に対して直接法を基にした前処理を施し，地盤要素からなる部分行列には節点単位のブロック対角スケーリングによる前処理が適用されている．この前処理は坂・小磯らによって提案された PSC（Partial Sparse Cholesky）前処理 [1] で，特に本研究ではこの前処理を用いた CG 法を PSCCG 法と呼ぶ．

本シミュレーションの実行時間の多くを時間発展部分の計算時間が占めており，その大部分が PSCCG 法にかかる

[1] 神戸大学大学院システム情報学研究科
[2] 鹿島建設株式会社
[a)] kgoto@stu.kobe-u.ac.jp

時間である．そこで本研究では，PSCCG 法に重点を置いたプロセス並列化を行う．そして並列化実装の評価を負荷分散とスケーリングの観点から行う．

2. 数値解法

本章では時間方向の離散化に用いられている平均加速度法と連立一次方程式の解法である PSCCG 法について説明する．

2.1 平均加速度法

平均加速度法は数値積分法の Newmark の β 法 [3] の一種である．これは時刻 $t = t_n, t_{n+1}$ での加速度が与えられたとき，$[t_n, t_{n+1}]$ 区間内での加速度の時間変化を仮定し，積分することで速度と変位を求める手法である．本シミュレーションでは，加速度の時間変化が時刻 $t = t_n, t_{n+1}$ での加速度の平均値で一定であると仮定した平均加速度法が用いられている．ここでは平均加速度法に限定して説明を行う．

質量 m，粘性減衰係数 c，剛性 k の質点が地面に接しているとする．さらに地面に対して地動加速度 $\ddot{y}_{\mathrm{o}}$ が働いているとする．加速度 $\ddot{y}$，速度 $\dot{y}$，変位 y と表すと，この質点の運動方程式は

$$m\ddot{y} + c\dot{y} + ky = -m\ddot{y}_{\mathrm{o}} \tag{1}$$

となる．ここで時刻 t_n での $\ddot{y}, \dot{y}, y$ をそれぞれ $\ddot{y}_n, \dot{y}_n, y_n$ と表すと，区間 $[t_n, t_{n+1}]$ での加速度，速度，変位は

$$\ddot{y}(t) = \frac{\ddot{y}_n + \ddot{y}_{n+1}}{2} \tag{2}$$

$$\dot{y}(t) = \dot{y}_n + \int_{t_n}^{t} \ddot{y}(t)\mathrm{d}t \tag{3}$$

$$y(t) = y_n + \int_{t_n}^{t} \dot{y}(t)\mathrm{d}t \tag{4}$$

と表せる．$t = t_{n+1}$，$\Delta t = t_{n+1} - t_n$ として計算すると，速度と変位は

$$\dot{y}_{n+1} = \dot{y}_n + \frac{1}{2}(\ddot{y}_n + \ddot{y}_{n+1})\Delta t \tag{5}$$

$$y_{n+1} = y_n + \dot{y}_n\Delta t + \frac{1}{4}(\ddot{y}_n + \ddot{y}_{n+1})\Delta t^2 \tag{6}$$

となる．したがって運動方程式は式（1），(5)，(6）から

$$(m + \frac{c}{2}\Delta t + \frac{k}{4}\Delta t^2)\ddot{y}_{n+1} = -m\ddot{y}_{\mathrm{o}_{n+1}} - ca - kb \tag{7}$$

となる．ただし，

$$a = \dot{y}_n + \frac{1}{2}\ddot{y}_n\Delta t \tag{8}$$

$$b = y_n + \dot{y}_n\Delta t + \frac{1}{4}\ddot{y}_n\Delta t^2 \tag{9}$$

である．

多自由度系おいては物理量をベクトルに，係数を行列に置き換えればよく，質量マトリクス M，減衰マトリクス C，剛性マトリクス K を用いて

$$(M + \frac{C}{2}\Delta t + \frac{K}{4}\Delta t^2)\ddot{\boldsymbol{y}}_{n+1} = -M\ddot{\boldsymbol{y}}_{\mathrm{o}_{n+1}} - C\boldsymbol{a} - K\boldsymbol{b} \tag{10}$$

となる．これが本シミュレーションで PSCCG 法を用いて解くべき連立一次方程式である．以後断りなく $A\boldsymbol{x} = \boldsymbol{b}$ と書いた時は，式（10）を表す．

2.2 PSCCG 法

前処理付き共役勾配法の前処理は計算上では $\boldsymbol{z} = P^{-1}\boldsymbol{r}$ を行えばよい．ここで P は前処理行列で，$\boldsymbol{z}, \boldsymbol{r}$ はベクトルである．

$A\boldsymbol{x} = \boldsymbol{b}$ の係数行列 A は

$$A = \begin{pmatrix} A_{\mathrm{BB}} & A_{\mathrm{IB}}^{\top} & O \\ A_{\mathrm{IB}} & A_{\mathrm{II}} & A_{\mathrm{SI}}^{\top} \\ O & A_{\mathrm{SI}} & A_{\mathrm{SS}} \end{pmatrix} \tag{11}$$

と書ける．ここで添え字 B は建物要素由来の成分，添え字 S は地盤要素由来，I は建物と地盤の接するインターフェース層（IF 層）由来の成分を表す．O は零行列である．この A に対して，P を

$$P = \begin{pmatrix} A_{\mathrm{BB}} & A_{\mathrm{IB}}^{\top} & O \\ A_{\mathrm{IB}} & A_{\mathrm{II}} & O \\ O & O & P_{\mathrm{SS}} \end{pmatrix} \tag{12}$$

ととる．P_{SS} は節点単位の対角スケーリング [4] である．ベクトル $\boldsymbol{z}, \boldsymbol{r}$ の各要素を建物，インターフェース，地盤毎に分けて

$$\boldsymbol{z} = (\boldsymbol{z}_{\mathrm{B}}, \boldsymbol{z}_{\mathrm{I}}, \boldsymbol{z}_{\mathrm{S}})^{\top} \tag{13}$$

$$\boldsymbol{r} = (\boldsymbol{r}_{\mathrm{B}}, \boldsymbol{r}_{\mathrm{I}}, \boldsymbol{r}_{\mathrm{S}})^{\top} \tag{14}$$

と置く．このとき，$\boldsymbol{z}$ の各要素は

$$\begin{pmatrix} A_{\mathrm{BB}} & A_{\mathrm{IB}}^{\top} \\ A_{\mathrm{IB}} & A_{\mathrm{II}} \end{pmatrix} \begin{pmatrix} \boldsymbol{z}_{\mathrm{B}} \\ \boldsymbol{z}_{\mathrm{I}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{r}_{\mathrm{B}} \\ \boldsymbol{r}_{\mathrm{I}} \end{pmatrix} \tag{15}$$

$$P_{\mathrm{SS}}\boldsymbol{z}_{\mathrm{S}} = \boldsymbol{r}_{\mathrm{S}} \tag{16}$$

で計算される．式（15），つまり建物・インターフェース部分（BI 部分）の前処理（BI 前処理）は，一般的な建物の構造設計に用いられる建物モデルの規模であれば，それ程大きな問題でないため直接法で実行できることが利用されている．本シミュレーションコードでは，直接法に疎なコレスキー分解を行う公開ライブラリのひとつで，Intel MKL にも含まれている PARDISO[5], [6] が用いられている．各反復で P は不変であるため，PARDISO は分解を 1 回だけ行い，あとは前進後退代入のみを行う．一方，式（16），つまり地盤部分（S 部分）の前処理（S 前処理）はブロック単位で逆行列を求めて $\boldsymbol{z}_S$ を計算する．**Algorithm 1** が PSCCG 法のアルゴリズムである．

Algorithm 1 PSCCG method

1: $\boldsymbol{x} = \boldsymbol{x}_0$
2: $\boldsymbol{r} = \boldsymbol{b} - A\boldsymbol{x}$
3: $\boldsymbol{r} = (\boldsymbol{r}_{\mathrm{B}}, \boldsymbol{r}_{\mathrm{I}}, \boldsymbol{r}_{\mathrm{S}})$, $z = (\boldsymbol{z}_{\mathrm{B}}, \boldsymbol{z}_{\mathrm{I}}, \boldsymbol{z}_{\mathrm{S}})$
4: BI precondition :
$$\text{solve} \begin{pmatrix} A_{\mathrm{BB}} & A_{\mathrm{IB}}^{\top} \\ A_{\mathrm{IB}} & A_{\mathrm{II}} \end{pmatrix} \begin{pmatrix} \boldsymbol{z}_{\mathrm{B}} \\ \boldsymbol{z}_{\mathrm{I}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{r}_{\mathrm{B}} \\ \boldsymbol{r}_{\mathrm{I}} \end{pmatrix}$$
5: S precontition : $\boldsymbol{z}_{\mathrm{S}} = P_{\mathrm{SS}}^{-1}\boldsymbol{r}_{\mathrm{S}}$
6: $\boldsymbol{p} = \boldsymbol{z}$
7: $\rho_0 = 0, \rho_1 = \boldsymbol{r}^{\top}\boldsymbol{z}$
8: τ : tolerance, $\delta = \tau \times |b|_2$, $k = 0$
9: **while** $\sqrt{\rho_1} > \delta$ **do**
10: $k = k + 1$
11: $\boldsymbol{q}_k = A\boldsymbol{p}_k$
12: $\alpha = \rho_1 / \boldsymbol{p}_k^{\top}\boldsymbol{q}_k$
13: $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha\boldsymbol{p}_k$
14: $\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha\boldsymbol{q}_k$
15: BI precondition :
$$\text{solve} \begin{pmatrix} A_{\mathrm{BB}} & A_{\mathrm{IB}}^{\top} \\ A_{\mathrm{IB}} & A_{\mathrm{II}} \end{pmatrix} \begin{pmatrix} \boldsymbol{z}_{\mathrm{B}k+1} \\ \boldsymbol{z}_{\mathrm{I}k+1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{r}_{\mathrm{B}k+1} \\ \boldsymbol{r}_{\mathrm{I}k+1} \end{pmatrix}$$
16: S precondition : $\boldsymbol{z}_{\mathrm{S}} = P_{\mathrm{SS}}^{-1}\boldsymbol{r}_{\mathrm{S}}$
17: $\rho_0 = \rho_1, \rho_1 = \boldsymbol{r}^{\top}\boldsymbol{z}$
18: $\beta = \rho_1 / \rho_0$
19: $\boldsymbol{p}_{k+1} = \boldsymbol{z}_{k+1} + \beta\boldsymbol{p}_k$
20: **end while**

3. プロセス並列化実装の方針

このシミュレーションコードに対し MPI によるプロセス並列化を行う．**図 1** は PSCCG 法を逐次実行（1 プロセス 1 スレッド）したときの実行時間の内訳である．BI 前処理には cluster_sparse_solver を使用した．図 1 の SpMV は行列ベクトル積を表し，**Algorithm 1** の 11 行目に対応する．また BI 前処理は **Algorithm 1** の 15 行目に対応する．この実行時間の内訳より，PSCCG 法の実行時間の多くを BI 前処理が占めることがわかる．ここで BI 前処理には，例えば $(\boldsymbol{z}_{\mathrm{B}}, \boldsymbol{z}_{\mathrm{I}})$ が必要である．$(\boldsymbol{z}_{\mathrm{B}}, \boldsymbol{z}_{\mathrm{I}})$ を複数プロセスに分散して更新すると，通信が必要になりプログラムが複雑になる．したがって今回の実装では，BI 部分の要素をひとつのプロセスだけが持つようにする．つまり，ランク 0 に BI 部分の BI 前処理以外の計算を担当させ，他のランクに S 部分の計算を分担させる．そして最も時間のかかる BI 前処理のみ全プロセスで処理する．もともと直接法に用いられている PARDISO をプロセス並列版である PARDISO for Clusters[7] に変更し，全プロセスで cluster_sparse_solver をコールするように実装する．また S 部分行列の行に関して均等なブロック分割を行うと，行列ベクトル積のプロセス間の負荷が均等にならない．これは**図 2** から分かるように，行列要素が特定の行に偏っているためである．この偏りは IF 層付近の地盤に関する行に多く見られる．したがって S 部分の計算領域は，S 部分を担当するプロセス間で行列ベクトル積の負荷が均一になるようにブロック分割を行った．

4. プロセス並列化実装の評価

4.1 本研究で用いる係数行列

本研究では**表 1** の係数行列を用いた．これは**図 3** の建物・地盤モデルについて立てられた連立一次方程式の係数行列である．

4.2 実行環境

実装評価のための実行には Intel®Xeon Phi™Processor 7250（KNL）を搭載したシステムを用いた．1 ノードあた

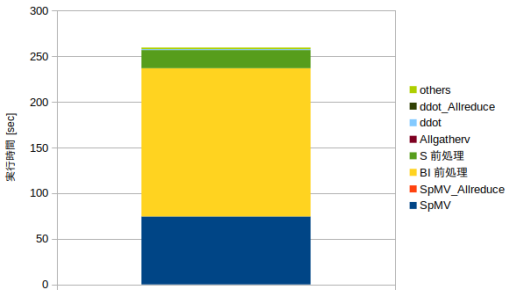

図 1 PSCCG 法の実行時間内訳（逐次）

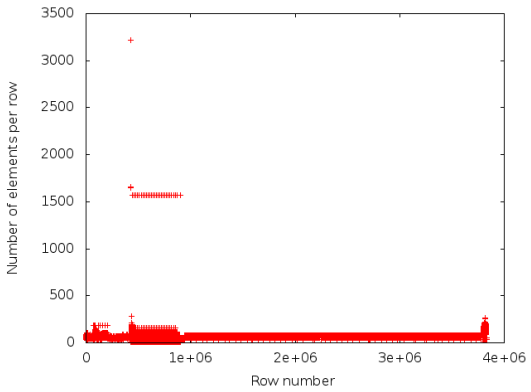

図 2 係数行列 A の行毎の要素数

表 1 係数行列 A

	全体	BI 部分	S 部分
行列サイズ	3,837,294	429,180	3,408,114
非零要素数	150,154,543	13,403,955	136,181,545

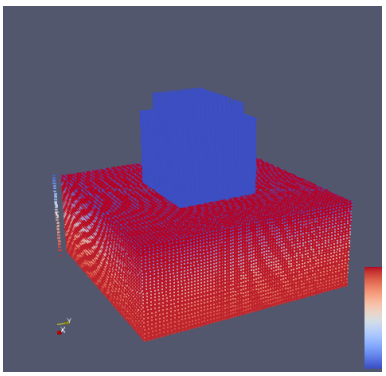

図 3 建物・地盤モデル

り 68 コアあり，4 ノードがネットワークで接続されている．コンパイラは Intel Fortran を用い，数値計算ライブラリは Intel MKL v11.3 を用いる．通信の評価を簡単にするため使用するのは 1 ノードだけとし，ノード内でプロセス並列を行うことにする．

4.3 性能評価

スレッド数を 1 スレッドに固定し，プロセス数を 2，4，8 に変えながら PSCCG 法部分の実行時間の測定を行った．このときの各ランク毎での実行時間は**図 4** のとおりである．まずプロセス数 2 では BI 前処理は 2 つのプロセスが行う．BI 前処理以外の BI 部分の計算は 1 つのプロセス（ランク 0）が行い，S 部分の計算も 1 つのプロセス（ランク 1）が行う．S 部分の行列ベクトル積の負荷が BI 部分の行列ベクトル積の負荷に比べて大きいため，ランク 1 が全体のボトルネックとなっている．

次にプロセス数 4 ではプロセス数 2 のときとは異なり，BI 前処理を行うプロセスは 4 つ（ランク 0〜3）に増える．また S 部分の計算を行うプロセスが 3 つ（ランク 1〜3）に増える．プロセス数 2 のとき同様，S 部分の行列ベクトル積の負荷が BI 部分の行列ベクトル積の負荷に比べて大きいため，ランク 1 から 3 が全体のボトルネックとなっている．

最後にプロセス数 8 では BI 前処理を行うプロセスは 8 つ（ランク 0〜8）に増える．また S 部分の計算を行うプロセスが 7 つ（ランク 1〜7）に増える．このとき，BI 部分の行列ベクトル積の負荷と，S 部分の行列ベクトル積の負荷を担当プロセスの数 7 で割った値が近くなる．したがって S 部分の行列ベクトル積によるボトルネックが解消されている．

次に PSCCG 法全体の速度向上率を**図 5** に示す．このときの実行時間は最も時間のかかるプロセスの実行時間である．また実行時間の大部分を占める BI 前処理に使用した cluster_sparse_solver の速度向上率を**図 6** に示す．プロセス数 2 のときの実行時間を基準にして，プロセス数 4 で 2 倍，8 で 3.5 倍の速度向上が確認できた．

また 1 コア 1 プロセスで実行したときのピーク性能を Intel®VTune™Amplifier XE 2017 で測定したところ，ピーク性能は 1.5[GFLOPS] であった．KNL の理論ピーク性能は 272 スレッド（68 コア × ハイパースレッディング 4）で 3046[GFLPOS] である．したがって 1 プロセスでの理論ピーク性能は 11.2[GFLOPS] で，約 13%の性能が出ていることが確認できた．

5. 結論

5.1 まとめ

本研究の最終目標は本地震動シミュレーションコードの並列化による高速化である．そのためのアプローチのひとつとして，MPI によるプロセス並列化を行った．

まず本シミュレーションコードで解かれる連立一次方程式について，平均加速度法の説明を行い明らかにした．次に本コードに用いられている前処理付き共役勾配法の PSCCG 法の紹介をし，そのアルゴリズムを掲載した．

次に本コードのプロセス並列化方針を示した．本コードのホットスポットは PSCCG 法であるため PSCCG 法部分のプロセス並列化を重要視した．PSCCG 法のホットスポットは BI 前処理に用いられている PARDISO である．したがって BI 前処理を PARDISO のプロセス並列版である PARDISO for Clusters を用いて，全てのプロセスで実行するようにした．また S 部分の計算において，行に関する均等なブロック分割を行うと行列ベクトル積の負荷がプロセス間で不均一になる．したがって行列ベクトル積の負荷が均一になるように S 部分の負荷分散を行った．つまり BI 部分の計算は BI 前処理のみを全てのプロセスで行い，BI 前処理以外の計算はランク 0 のプロセスで行った．一方，S 部分の計算はランク 0 以外のプロセスで行った．

最後に KNL を用いてプロセス並列実装の評価を行った．1 ノード，ノード内のプロセス並列で，プロセス数を 2 から 8 に変えながら PSCCG 法の実行時間とその内訳を測定した．2，4 プロセスでは S 部分の行列ベクトル積の負荷が大きく，全体のボトルネックとなっていた．8 プロセスではランク 0 での行列ベクトル積の負荷と，それ以外のプロセスでの行列ベクトル積の負荷がほぼ等しくなった．また PSCCG 法の速度向上率を計測した．2 プロセスでの実行時間を基準とし，8 プロセスで PSCCG 法全体で 3.2 倍の速度向上が確認できた．また BI 前処理の PARDISO for Clusters は 3.5 倍の速度向上が確認できた．さらに 1 コア 1 プロセスでのピーク性能を測定すると 1.5[GFLOPS] で，およそ 13%の性能がでていることが確認できた．

5.2 今後の課題

本研究では 8 プロセスまでの実行時間の計測しか行えなかった．したがって 16 プロセス以上での実行時間を計測し，速度向上率を評価する．さらに通信について評価し，通信の最適化を行う必要がある．

また本研究で実装したプロセス並列化方針では，16 プロセス以上で実行したときに，BI 部分の行列ベクトル積がボトルネックになることが容易に想定できる．したがって，プロセス数によらず行列ベクトル積の負荷が均一になる実装に変更しなければならない．

最後にピーク性能については，現状メニーコアプロセッサ向けの最適化が進んでいない．最適化を行ってから再度評価する必要がある．また通常の Intel®Xeon プロセッサでの実行，評価も行う必要がある．

謝辞 本研究成果の一部は，JSPS 科研費 JP16H02822，JP18K11325 の助成を受けたものです．

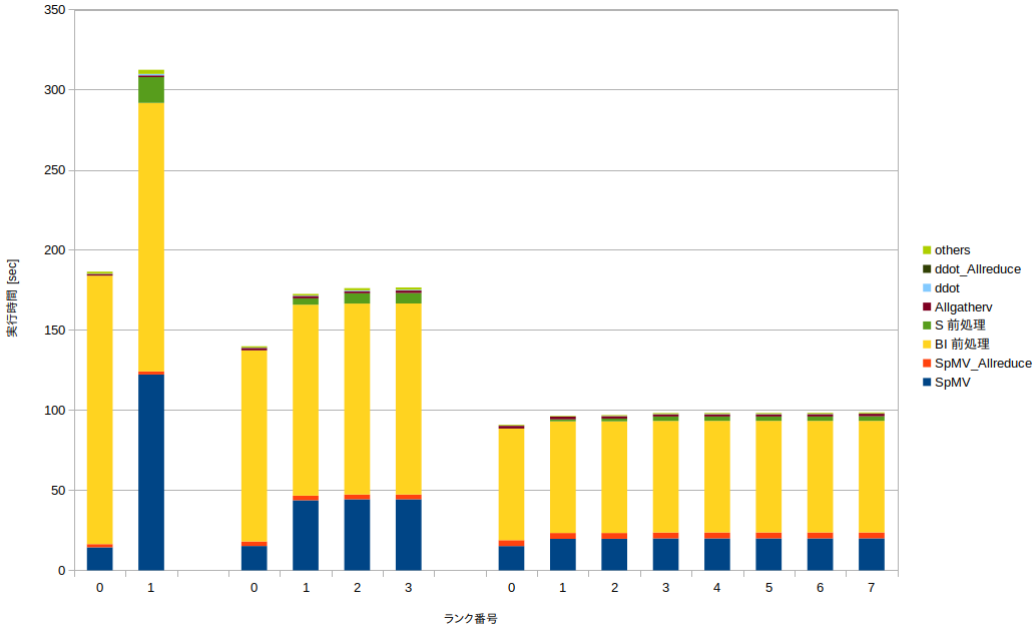

図 4 プロセス数 2，4，8 での
各ランク毎の処理別実行時間

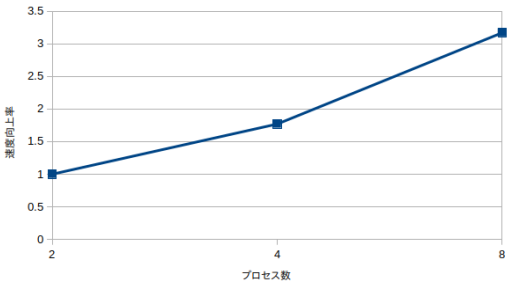

図 5 PSCCG 法の速度向上率

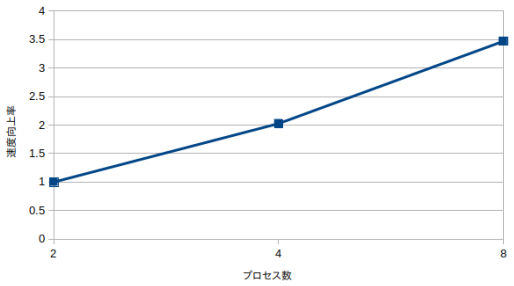

図 6 cluster_sparse_solver の速度向上率

参考文献

[1] 坂　敏秀，小磯利博：建物－地盤動的相互作用問題向けの疎な部分コレスキー分解前処理付き共役勾配法，土木学会論文集 A2（応用力学），Vol. 72, No. 2, pp. I_187–I_196（オンライン），DOI: 10.2208/jscejam.72.I_187 (2016).

[2] 柴田明徳：最新 耐震構造解析 第 3 版，森北出版株式会社 (2015).

[3] Newmark, N. M.: A Method of Computation for Structural Dynamics, *Proc. ASCE*, Vol. 85, No. 3, pp. 67–94 (1959).

[4] Nakajima, K. and Okuda, H.: Parallel iterative solvers with selective blocking preconditioning for simulations of fault-zone contact, *Numerical Linear Algebra with Applications*, Vol. 11, No. 8 - 9, pp. 831–852 (online), DOI: 10.1002/nla.349 (2004).

[5] PARADISO-Project: PARDISO 6.0 Solver Project, pardiso-project (online), available from ⟨https://www.pardiso-project.org/⟩ (accessed 2018-11-12).

[6] Intel: Developer Reference for Intel® Math Kernel Library - Fortran, Intel (online), available from ⟨https://software.intel.com/en-us/mkl-developer-reference-fortran-intel-mkl-pardiso⟩ (accessed 2018-11-12).

[7] Intel: Parallel Direct Sparse Solver for Clusters, Intel (online), available from ⟨https://software.intel.com/en-us/articles/intel-math-kernel-library-parallel-direct-sparse-solver-for-clusters⟩ (accessed 2018-11-12).