Vol.2015-HPC-152 No.14
2015/12/17

大規模連立一次方程式の反復解法の実問題への適用と性能評価

俵谷 健太郎[1,a)]　横川 三津夫[1]

概要：地盤とその上の大規模構造物に対する地震動解析においては，有限要素法などで離散化して得られる対称疎行列を係数行列に持つ大規模な連立一次方程式を，高速かつ複数のケースについて解くことが求められる．特に，剛性が大きく異なる計算対象では，係数行列の条件数が大きくなり通常の反復解法では解が得られないことが多く，企業等の現場では確実に解が得られる直接法が使用されている．しかし，一般に大規模な計算の場合，同等の解が得られるならば，直接法と比較してメモリ使用量，計算量が少ない反復解法が有効である場合が多い．連立一次方程式の反復解法として，共役勾配法（CG 法），共役残差法（CR 法）等が知られている．
本研究では，反復解法を実問題に適用し，その性能を評価した．地盤とその上の大規模構造物の地震動解析において現れる条件数の異なる大規模対称疎行列に対して，数値実験を行った結果，行列に応じて，有効な反復解法が異なることがわかった．

1. はじめに

地盤とその上の大規模構造物に対する地震動解析においては，有限要素法等で離散化して得られる対称疎行列を係数行列に持つ大規模連立一次方程式を，高速かつ複数のケースについて解くことが求められる．また，解析の現場では，メッシュ等の情報がなく，係数行列と右辺ベクトルだけが与えられ，連立一次方程式が解ける数値計算ライブラリが必要とされる場合がある．本研究では実問題の例として，3 つの異なる計算対象の地震動解析で得られる連立一次方程式を扱う．特に，地盤とその上の構造物で剛性が大きく異なるような計算対象では，係数行列の条件数が大きくなり通常の反復解法では解が得られないことが多く，企業等の現場では確実に解が得られる直接法が使用されている．しかし，一般に大規模な計算の場合，同等の解が得られるならば，直接法と比較して，メモリ使用量，計算量が少ない反復解法が有効である場合が多い．連立一次方程式の反復解法の一つとして，共役勾配法や共役残差法が知られている．また，これらの反復解法は適切な前処理を施すことにより，さらに速い収束が期待できる．反復解法の前処理として，対角項スケーリング，対称ガウスザイデル法，不完全コレスキー分解，代数的マルチグリッド等が存在する．

[1] 神戸大学 大学院 システム情報学研究科
Graduate School of System Informatics, Kobe University.
[a)] k_hyotani@stu.kobe-u.ac.jp

本研究では，これらの反復解法の実問題への適用，および，その性能評価を行った．地盤とその上の大規模構造物の地震動解析において，複数のモデルに対し有限要素法で離散化した時に得られる，条件数の異なる大規模対称疎行列に対して，数値実験を行い，高速かつ，どのような場合にも適用可能な解法を見出すことが目標である．

2. 対称行列向け反復解法

2.1 前処理付き共役勾配法

実対称正定値行列 $A \in \mathbb{R}^{n\times n}$，実ベクトル $b \in \mathbb{R}^n$ に対し，

$$Ax = b \tag{1}$$

を解く．共役勾配法（ Conjugate Gradient method : CG 法）は解の修正方向として，互いに A 共役となる方向を用いる反復解法である．

また，A を P_1AP_2 ($P_1, P_2 \in \mathbb{R}^{n\times n}$: 正則行列) に置き換えることで，元の行列よりも条件の良い行列になる場合がある．すなわち，

$$(P_1AP_2)\,P_2^{-1}x = P_1b \tag{2}$$

に CG 法を適用し，アルゴリズムを再構築する．これを前処理付き共役勾配法（ Preconditioned Conjugate Gradient method : PCG 法）という．P_1, P_2 は，$P = (P_2P_1)^{-1} \approx A$ となるように設定する．Algorithm 1 に PCG 法の手順を示す [1]． アルゴリズムの再構築により，一般に前処理は，1

Algorithm 1 PCG method

$k = 0, \delta = tol \cdot \|b\|_2, x = x_0, r = b - Ax$
while $\|r\|_2 > \delta$ **do**
  Solve $P\tilde{r} = r$
  $\rho_c = r^\top \tilde{r}$
  $k = k + 1$
  if $k = 1$ **then**
    $p = \tilde{r}$
  else
    $p = \tilde{r} + (\rho_c/\rho_-)p$
  end if
  $w = Ap$
  $\mu = \rho_c / p^\top w$
  $x = x + \mu p$
  $r = r - \mu w$
  $\rho_- = \rho_c$
end while

Algorithm 2 PCR method

$k = 0, \delta = tol \cdot \|b\|_2, x = x_0, r = b - Ax$, Solve $P\tilde{r} = r$
while $\|r\|_2 > \delta$ **do**
  $\rho_c = \tilde{r}^\top A\tilde{r}$
  $k = k + 1$
  if $k = 1$ **then**
    $p = \tilde{r}$
    $w = A\tilde{r}$
  else
    $p = \tilde{r} + (\rho_c/\rho_-)p$
    $w = A\tilde{r} + (\rho_c/\rho_-)w$
  end if
  Solve $P\tilde{w} = w$
  $\mu = \rho_c / w^\top \tilde{w}$
  $x = x + \mu p$
  $\tilde{r} = \tilde{r} - \mu\tilde{w}$
  $\rho_- = \rho_c$
end while

反復ごとに $P\tilde{r} = r$ の解 $\tilde{r}$ を求める処理に置き換えられる．また，$P \approx A$ より，これは $A\tilde{r} = r$ の近似解を得る処理とも言える．

2.2 前処理付き共役残差法

エルミート行列向けの反復解法として共役残差法（Conjugate Residual method : CR 法）[2] が存在する．CR 法では残差ベクトル r が互いに A 共役となる．

CG 法と同様に，CR 法にも前処理の適用が有効である．これを前処理付き共役残差法（ Preconditioned Conjugate Residual method : PCR 法）という．Algorithm 2 に PCR 法の手順を示す．

3. 反復解法の前処理

今回検討した 4 種類の前処理について，それぞれ説明する．

3.1 対角項スケーリング

行列 A に対し，その対角項 $D = \mathrm{diag}\,[a_{11}, a_{22}, \ldots, a_{nn}]$ によってスケーリングを行う．すなわち，

$$P_1 = P_2 = D^{-1/2}, \quad P = D, \tag{3}$$

$$\left(D^{-1/2} A D^{-1/2}\right)\left(D^{1/2} x\right) = D^{-1/2} b \tag{4}$$

のように前処理を施す [3]．以下，簡単のために単に S と書いて対角項スケーリング前処理を表す．

係数行列の対角要素のみを扱うので，実装が簡単で，前処理としての処理も軽い．しかし，その分効果も小さく，問題によっては収束しない場合も多い．

3.2 対称ガウスザイデル法

定常反復解法であるガウスザイデル法（Gauss Seidel method : GS 法）を使って，$A\tilde{r} = r$ の近似解を得る前処理が存在する．対称行列向け反復解法の前処理として GS 法を使う場合，1 反復につき，順方向に更新（順方向 GS 法）した後，逆方向にもう一度更新（逆方向 GS 法）する．これを対称ガウスザイデル法（Symmetric Gauss Seidel method : SGS 法）という．

対称ガウスザイデル法の計算量は行列の非零要素数に比例し，対角項スケーリングよりも時間はかかるが，収束を加速させる効果は一般に対角項スケーリングより上である．

3.3 不完全コレスキー分解

A が対称行列の時，

$$A = LDL^\top \tag{5}$$

のように，下三角行列 L，対角行列 D に分解可能である．L, D の各要素は

$$d_{ii} = a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2 d_{kk}, \quad \text{for} \quad i = 1, \ldots, n, \tag{6}$$

$$l_{ij} = \frac{1}{d_{jj}}\left(a_{ij} - \sum_{k=1}^{j-1} l_{ik} d_{kk} l_{jk}\right), \quad \text{for} \quad i > j. \tag{7}$$

のように計算される．

不完全コレスキー分解（Incomplete Cholesky decomposition : IC）では，元の行列 A で $a_{ij} = 0$ ならば，分解後の行列 L でも $l_{ij} = 0$ とすることで，行列の分解による Fill-in を回避する．前処理としては，分解後に反復の中で，$P = LDL^\top$ として，$LDL^\top \tilde{r} = r$ を前進代入，後退代入で解く．

不完全コレスキー分解，および前進・後進代入の計算量は，対称ガウスザイデル法と同じく，行列の非零要素数に比例する．収束を加速させる効果も一般に対角項スケーリングより高い．また，反復処理に入る前の行列の分解処理にやや時間がかかるので，非零要素数が多い場合に問題となりやすい．

3.4 代数的マルチグリッド

（幾何的）マルチグリッド法（MultiGrid method : MG 法）[4], [5] は解きたい対象の離散格子よりも粗い格子を段階的に複数構築して，各格子上で緩和法（定常反復法）により，誤差ベクトルの各周波成分を減少，修正させる方法である．MG 法は現在，特に収束の速い（問題の規模が反復数に依存しない）反復解法として知られており，前処理としても多く使われる．

しかし，（幾何的）MG 法は粗い格子の構築に，メッシュデータ等の方程式の離散化の情報を使うので，今回のように，離散化の情報がない場合には適用できない．そこで，係数行列のみから代数的に粗い格子を構築する代数的マルチグリッド法 (Algebraic MultiGrid method : AMG 法)[6], [7], [8] を使用する．今回は主に Classical AMG[8] を元に実装を行った．

3.4.1 緩和法

定常反復法（Gauss-Seidel 法等）は，離散化した偏微分方程式等の高周波（強く振動する）誤差を速く減衰させる，すなわち，平滑化作用を持つことが知られている．

AMG において，このような高周波誤差を減衰させるために使われる手法を，緩和法という．誤差の高周波成分を除いた後，より粗い格子上で低周波成分を近似的に求められれば，誤差の良い修正量として用いることが出来ると考えられる．

3.4.2 代数的滑らかさ

低周波誤差を効率よく減少させるためには，低周波誤差の特徴をなるべく保存するような粗い格子を得る必要がある．係数 a_{ij} を

$$a_{ij}^- = \begin{cases} a_{ij} & (\text{if} \quad a_{ij} < 0) \\ 0 & (\text{if} \quad a_{ij} \geq 0) \end{cases}, \quad a_{ij}^+ = \begin{cases} 0 & (\text{if} \quad a_{ij} \leq 0) \\ a_{ij} & (\text{if} \quad a_{ij} > 0) \end{cases} \tag{8}$$

のように正負で分け，さらに，$a_{ii} \approx \sum_{j \neq i} |a_{ij}|$ である時，それぞれの i について平均的に，

$$\sum_{j \neq i} \frac{|a_{ij}^-|}{a_{ii}} \frac{(e_i - e_j)^2}{e_i^2} + \sum_{j \neq i} \frac{a_{ij}^+}{a_{ii}} \frac{(e_i + e_j)^2}{e_i^2} \ll 1, \tag{9}$$

が成り立つ．

よって，緩和法による誤差は，$|a_{ij}^-|$ が大きい時，$e_j \approx e_i$（滑らか）となり，a_{ij}^+ が大きい時，$e_j \approx -e_i$ となる（振動する）傾向があると言える．このことから，誤差の特徴をなるべく保存するような粗い格子を得るためには，$|a_{ij}^-|$ が大きい方向に格子を粗くすれば良いと考えられる．

これらの情報をアルゴリズムの構築に使うために，「強接続」という概念を定義する．パラメータ $0 < \theta \leq 1$ が与えられているとして，Classical AMG では (10)　の定義がよく使われる．

定義 1

表 1: AMG の集合の定義
Table 1 Definition of sets for AMG.

集合	定義	意味
N_i	$\{j \mid j \neq i, a_{ij} \neq 0\}$	i が接続する点の集合
S_i	(10)　または (11)	i が強接続する点の集合
$S_i^\top$	$\{j \mid i \in S_j\}$	i が強接続される点の集合
C	Algorithm 3 の中で定義	粗い格子に含まれる点の集合
C_i	$C \cap S_i$	i が強接続する C 点の集合
C_i^+	$\{j \mid j \in C_i, a_{ij} > 0\}$	i が正の強接続をする C 点の集合
C_i^-	$\{j \mid j \in C_i, a_{ij} < 0\}$	i が負の強接続をする C 点の集合
F	Algorithm 3 の中で定義	粗い格子に含まれない点の集合

Algorithm 3 Modified standard coarsening algorithm

```
1: F := {i | S_i = ∅}, C := ∅, U := {1, ..., n}
2: λ_i = |S_i^⊤ ∩ U| + 2|S_i^⊤ ∩ F|
3: while U ≠ ∅ ∩ λ_i > 0 (i ∈ U) do
4:     max λ_i である i について，C := C ∪ {i}, U := U\{i}
5:     for all j ∈ S_i^⊤ ∩ U do
6:         F := F ∪ {j}, U := U\{j}
7:         for all k ∈ S_j ∩ U do
8:             λ_k ← λ_k + 1
9:         end for
10:    end for
11:    for all l ∈ S_i ∩ U do
12:        λ_l ← λ_l − 1
13:    end for
14: end while
15: if C = ∅ then
16:    C := C ∪ U
17: else
18:    F := F ∪ U
19: end if
```

$$-a_{ij} \geq \theta \max_{k \neq i, a_{ik} < 0} \{-a_{ik}\}. \tag{10}$$

を満たすなら，i は j に「強接続」するという．

しかし，係数行列が M 行列ではない場合，この定義は有効ではない場合が多い．そこで今回は Smoothed Aggregation AMG[6] でよく使われる，次の定義を採用した．

定義 2

$$-a_{ij} \geq \theta \sqrt{a_{ii} a_{jj}}. \tag{11}$$

を満たすなら，i は j に「強接続」するという．

3.4.3 制限：粗い格子点選択

今回は粗い格子の構築（制限）アルゴリズムに，Standard coarsening algorithm[6] と呼ばれる手法を，AMG のオープンソースライブラリである AMGCL[9] を参考に変形して，実装を行った．この手法では粗い格子上の点は，元の格子点の部分集合として選択される．

説明のために，**表 1** にいくつかの集合を定義する．

実装した制限アルゴリズムを Algorithm 3 に示す． 変更点は以下のとおりである．

- F の初期設定を $F := \emptyset$ から $F := \{i \mid S_i = \emptyset\}$ とした．
- $\lambda_i \leq 0$ となった場合，$C = \emptyset$ なら，U に残った点を全て C に，$C \neq \emptyset$ なら，U に残った点を全て F にした．AMGCL では，$\lambda_i \leq 0$ となった場合，U に残った点を全て C にしているが，今回の研究では格子点数を減らして計算量を削減するために，修正した．

3.4.4 補完行列 P の構築

補完行列 P の構築には，比較的シンプルな手法である Direct interpolation[6] を使用した．

補完行列 $P \in \mathbb{R}^{n \times |C|}$ を以下のように与える．

$$(Pe)_i = \begin{cases} e_i & \text{if } i \in C, \\ \sum_{j \in C_i} w_{ij} e_j & \text{if } i \in F, \end{cases} \tag{12}$$

このとき，以下のように w_{ik} を設定する．

$$w_{ik} = \begin{cases} -\alpha_i a_{ik}/a_{ii} & (k \in C_i^-) \\ -\beta_i a_{ik}/a_{ii} & (k \in C_i^+) \end{cases} \tag{13}$$

もし，$C_i^+ = \emptyset$ なら，

$$\beta_i = 0, \tag{14}$$

$$w_{ik} = -\alpha_i \frac{a_{ik}}{a_{ii} + \sum_{j \in N_i} a_{ij}^+} \quad (k \in C_i^-) \tag{15}$$

とする．

また，制限行列 R，粗い格子上の係数行列 A_{2h} は，以下のように構築した．

$$R = P^\top,$$

$$A_{2h} = P^\top A P.$$

3.4.5 AMG アルゴリズム

適当な段数の格子のそれぞれについて，P，R，A_{2h} を構築した後，反復処理により方程式を解く．AMG の反復処理では，**図 1** に示す 2 段グリッド法を再帰的に適用する．

粗い格子上の低周波誤差 e_{2h} を求める部分に，2 段グリッド法を再帰的に適用することで，マルチグリッド法の 1 反復となる．最も粗い格子上では，緩和法を数回適用して e_{2h} の近似解を得た．

4. 数値実験

4.1 対象とする数値データ

4 種類の行列を反復解法の評価対象とした．各係数行列のサイズと非零要素数を**表 2** に示す．いずれも対称疎行列である．

Matrix 0

$$-u_{xx} - u_{yy} = 2\Big((1-6x^2)y^2(1-y^2) + (1-6y^2)x^2(1-x^2)\Big) \quad \text{in} \quad \Omega, \tag{16}$$

$$u = 0 \quad \text{on} \quad \partial\Omega. \tag{17}$$

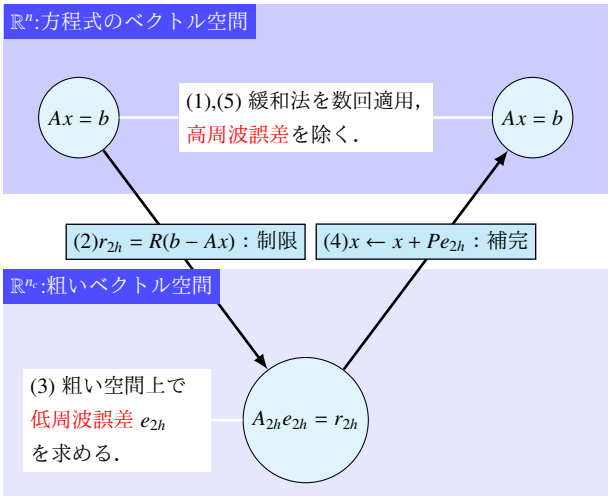

図 1: 2 段グリッド法
Fig. 1 Two-Grid method

を有限差分により離散化（離散格子点数 $N = 512^2$）して得られる方程式の係数行列である [7]．

Matrix 1

約 400 m × 400 m × 90 m の範囲の地盤に，有限要素法を適用して得られる連立一次方程式の係数行列である．

Matrix 2

Matrix 1 の場合よりやや狭い範囲の地盤に，節点あたり 6 自由度，81,000 の直方体要素の有限要素法と，その上の建物部分にバネ-質点系モデルを適用して得られる連立一次方程式の係数行列である．地盤部分と建物部分では剛性が異なっている．

Matrix 3

約 2 万節点の有限要素モデルより得られる連立一次方程式の係数行列である．

4.2 実行環境

プログラムは Fortran で実装し，神戸大学の π-VizStudio (SGI UV 300) で実行した．π-VizStudio の性能を**表 3** に示す．実験はすべて，逐次処理による結果である．また，使用したコンパイラおよび，そのオプションは ifort -xAVX -O3 -ipo -no-prec-div である．

4.3 実行時パラメータ

- 初期解：$x_0 = 0$.
- 収束判定：相対残差 L_2 ノルム $\frac{\|b - Ax\|_2}{\|b\|_2} \leq 1.0 \times 10^{-8}$
- AMG のパラメータ
  - 係数行列のサイズ n が $n \leq 100$ になるまで，格子を構築する．
  - 緩和法は図 1 において，(1) 順方向 GS，(5) 逆方向 GS を 1 反復ずつ行う．
  - 最も粗い格子上では，SGS を 10 反復適用して e_{2h} の近似解を得る．

表 2: 行列の要素数
Table 2 Number of elements of matrices

	Matrix 0	Matrix 1	Matrix 2	Matrix 3
Matrix size($n \times n$)	262144^2	1446863^2	249841^2	89014^2
The number of nonzero elements	1308672	114480579	19040429	5227256
Nonzero elements rate	1.9043684E-5	5.4686105E-5	2.0670604E-05	6.5971618E-4

表 3: π-VizStudio (SGI UV 300) の仕様
Table 3 Spec of π-VizStudio (SGI UV 300)

CPU		Intel Xeon E7-8857 v2
	Cores	12
	Operating frequency	3.0 GHz
	LLC	30 MB
Node	Memory	16 TiB (DDR3 32GiB * 512)
Compiler driver	Intel Parallel Studio XE Cluster Edition (ifort version 15.0.3)	
Numerical Libraries	Intel Math Kernel Library (2015.3.187)	

表 4: AMG のパラメータ
Table 4 Parameter for AMG.

Matrix	0	1	2	3
θ	0.05	0.40	0.45	0.60

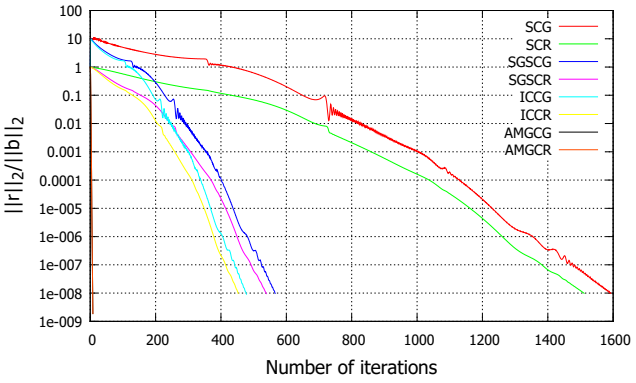

図 2: Matrix 0 の相対残差 L_2 ノルム（片対数グラフ）
Fig. 2 Relative residual L_2 norm for Matrix 0 (semilog graph).

– 強接続のパラメータ θ については，0.05 から 0.95 の間を 0.5 刻みで試行し，AMGCG か AMGCR が最速となった値を用いた．**表 4** にその値を示す．

4.4 結果

4.1 小節の数値データに対して，2 節の反復解法の適用を行った．その数値実験の結果を以下に示し，性能を評価する．

4.4.1 収束履歴

各数値データに対する，各反復解法の収束履歴を **図 2–5** に示す．

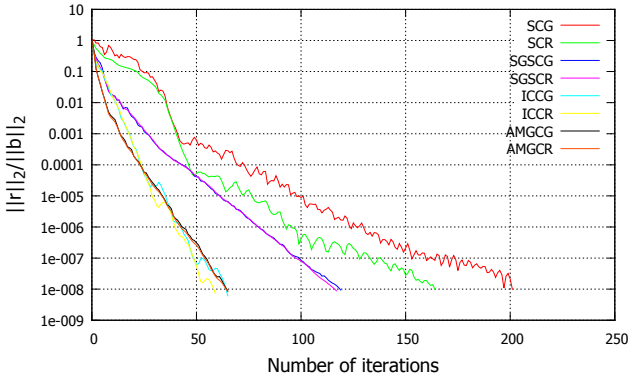

図 3: Matrix 1 の相対残差 L_2 ノルム（片対数グラフ）
Fig. 3 Relative residual L_2 norm for Matrix 1 (semilog graph).

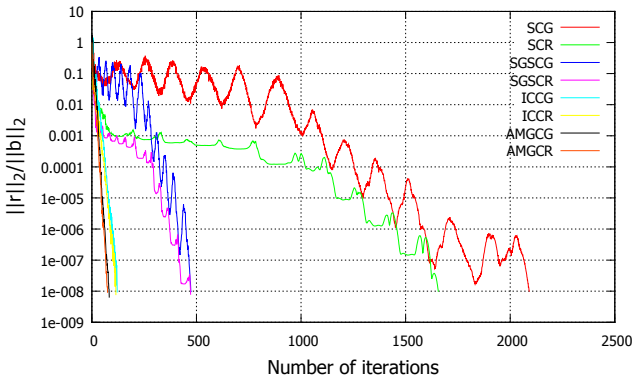

図 4: Matrix 2 の相対残差 L_2 ノルム（片対数グラフ）
Fig. 4 Relative residual L_2 norm for Matrix 2 (semilog graph).

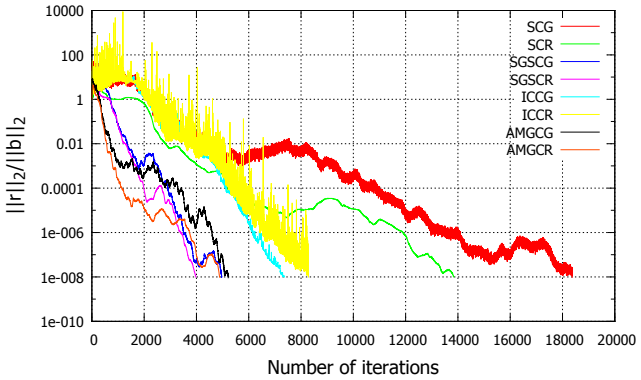

図 5: Matrix 3 の相対残差 L_2 ノルム（片対数グラフ）
Fig. 5 Relative residual L_2 norm for Matrix 3 (semilog graph).

Matrix 0

各前処理については，おおよそ AMG<IC<SGS<S の順に，収束までの反復回数が少なかった．AMGCG・CR は収

束までの反復回数が 8 回となっており，他の解法に比べて極端に少ないことがわかった．また，AMG 以外では，CG よりも CR のほうが僅かに収束が早いという結果になった．

Matrix 1

各前処理については，おおよそ IC≈AMG<SGS<S の順に，収束までの反復回数が少なかった．IC と AMG ではおおよそ同等となっている．また，S を除いては，CG と CR で反復回数に際立った差が見られないが，AMG 以外では，やや CR の方が少なくなっている．

Matrix 2

各前処理については，おおよそ AMG<IC<SGS<S の順に，収束までの反復回数が少なかった．IC と AMG での差はおおよそ 1.5 倍程度である．また，振動の大きい S と SGS では，CG と CR の収束過程の差が大きいが，SGS の最終的な反復回数はほぼ変わらなかった．

Matrix 3

各前処理については，おおよそ SGS≈AMG<IC<S の順に，収束までの反復回数が少なかった．反復の初期部分では SGS よりも AMG の方が収束が早いが，反復回数が増えるにつれて AMG の収束が鈍くなっていく．また，Matrix 1,2 と同じく，S では CG よりも CR の方が反復回数が際立って少ない．しかし，IC では，CG よりも CR の方がかなり強く振動して，収束がやや遅い．

本実験では AMG の緩和法に GS 法を使用しているため，AMG 前処理は SGS 前処理を複数の格子上で行う拡張版と捉えることが出来る．しかし，SCR と AMGCR を比較すると，AMGCR の方が反復回数が多い．このことから Matrix 3 に対しては，複数の格子を使った処理は逆効果となっていると考えられる．

全体として，単純な偏微分方程式の離散化による Matrix 0 と，実問題である Matrix 1–3 とでは，特に AMG 前処理付き反復解法の収束性に顕著な違いが見られた．

4.4.2 実行時間

各数値データに対する，各反復解法の実行時間の結果を **図 6–9** に示す．また，実務で使用されている PARDISO[10] という直接解法ソルバを試した実行時間の結果も合わせて示す．PARDISO の実行時のパラメータは，すべてデフォルトに設定した．

Matrix 0

最も実行時間が短く，有効だったのは AMGCG であった．

また，PARDISO の結果と比較すると，AMGCG のほうが 3 倍以上高速であることがわかった．このことから Matrix 0 に対しては，直接解法よりも反復解法の方が有効であると考えられる．

Matrix 1

最も実行時間が短く，有効だったのは ICCR であった．反復回数の観点でも ICCR は最も少ない．ICCR は AMGCG・AMGCR と比較して，反復回数は 6 回しか変わらないが，

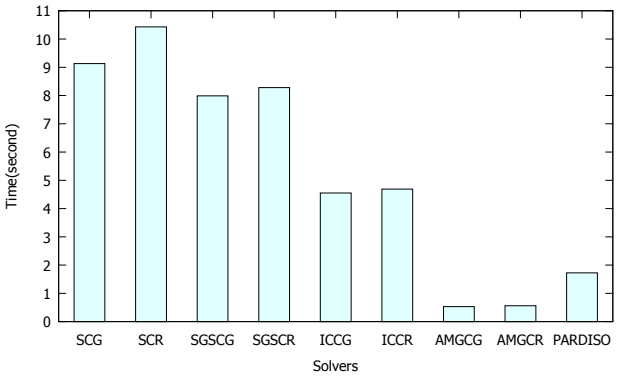

図 6: Matrix 0 の実行時間

Fig. 6 Execution time for Matrix 0.

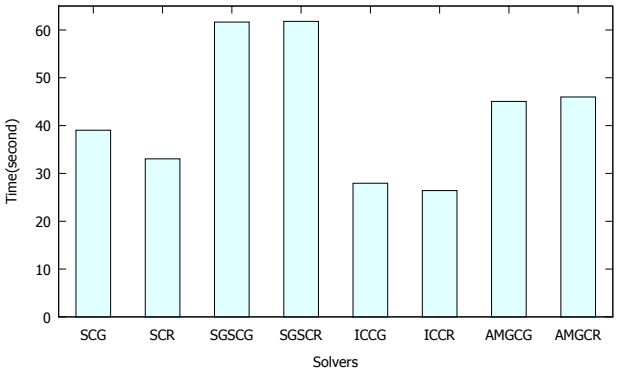

図 7: Matrix 1 の実行時間

Fig. 7 Execution time for Matrix 1.

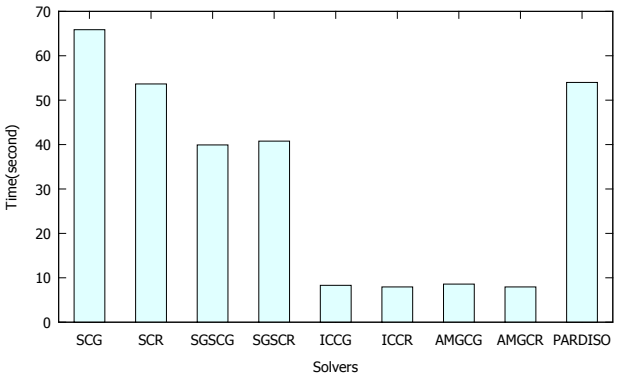

図 8: Matrix 2 の実行時間

Fig. 8 Execution time for Matrix 2.

AMG の方が IC よりも 1 反復での処理が重いため，ICCR の方が明らかに高速という結果となった．

Matrix 1 に PARDISO を適用したところ，行列の分解処理中にエラーが発生し，結果の確認ができなかった．行列の規模が大きすぎたためと考えられる．PARDISO の実行時パラメータを調整することによって改善される可能性はあるが，エラーが起こった分解処理の前の，行列のオーダリング処理の時点で既に 40.58 s を要していた．このことから Matrix 1 に対しては，直接解法よりも反復解法の方が有効であると考えられる．

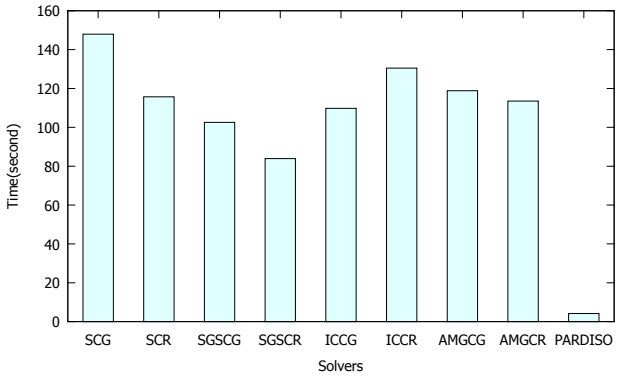

図 9: Matrix 3 の実行時間
Fig. 9 Execution time for Matrix 3.

Matrix 2

最も実行時間が短く，有効だったのは Matrix 1 と同じく ICCR であった．反復回数は AMGCR より ICCR の方が 1.5 倍ほど多いが，Matrix 1 の場合と同様に AMG の方が処理が重いため，ICCR のほうが僅かではあるが高速であった．

今回は AMG の緩和法に SGS 法を使用しているが，その代わりに対称 SOR 法（Symmetric SOR method : SSOR 法）を用いて更にその加速パラメータを調整すれば，AMGCR の方が速くなる可能性がある．

また，PARDISO の結果と比較すると，ICCR のほうが約 7 倍高速であることがわかった．このことから Matrix 2 に対しても，直接解法よりも反復解法の方が有効であると考えられる．

Matrix 3

最も実行時間が短く，有効だったのは SGSCR であった．Matrix 1,2 で最も高速だった ICCR は，Matrix 3 では反復回数が多く効果が小さかった．

また，PARDISO の結果と比較すると，PARDISO のほうが SGSCR よりも約 20 倍高速であることがわかった．Matrix 3 が Matrix 1,2 よりも非零要素数が少ないことと，多くの反復回数を要すること等が理由としてあげられる．このことから Matrix 3 に対しては，反復解法よりも直接解法の方が有効であると考えられる．

5. まとめ

本研究では，各種反復解法の構造物の地震動解析問題への適用，およびその性能評価を行った．

数値実験によって，Matrix 0 では AMGCG が有効であったが，今回対象とした実問題である Matrix 1,2 では ICCR，Matrix 3 では SGSCR が有効であった．また，反復・直接解法を比較した結果，Matrix 0,1,2 では反復解法，Matrix 3 では直接解法がそれぞれ，より高速であることを確かめた．より大きい問題を扱うような場合には，問題の規模が反復数に依存しないと言われている AMG が，有効となる可能性がある．

今後の課題としては，問題の性質・サイズによって変化する反復・直接解法の有効性の調査．解法の改良，および，他の解法の採用．また，より幅広い実問題についての反復解法の性能評価が挙げられる．

並列化については，対角項スケーリング前処理付き反復解法の並列性能が高く，ほぼ並列数にスケールすることがわかっている．ICCR や SGSCR の並列化を考えた場合，高並列時でも SCR より高速であるかどうかについては今後検討したい．

謝辞 本研究に際して，小柳義夫特命教授，谷口隆晴准教授，森下浩二特命助教，および研究室の先輩・同輩の皆様からは日頃より多くのご助力を頂きました．ここに感謝の意を表します．最後に，本研究成果の一部は，JSPS 科研費 26286087 の助成を受けたものです．

参考文献

[1] Golub, G. H. and Van Loan, C. F.: *Matrix Computations*, JHU Press, 4th edition (2014).
[2] Saad, Y.: *Iterative methods for sparse linear systems*, SIAM (2003).
[3] 速水謙，原田紀夫：対角項スケーリングを施した共役勾配法のベクトル計算機における有効性について，情報処理学会論文誌，Vol. 30, No. 11, pp. 1364–1375 (1989).
[4] 建部修見，小柳義夫：マルチグリッド前処理付き共役勾配法，情報処理学会研究報告，92-HPC-42, SWoPP'92，pp. 9–16 (1992).
[5] 建部修見，小柳義夫：マルチグリッド前処理付き共役勾配法の並列化，並列処理シンポジウム JSPP93 論文集，pp. 387–394 (1993).
[6] Trottenberg, U., Oosterlee, C. W. and Schuller, A.: *Multigrid*, Academic press (2000).
[7] Briggs, W. L., McCormick, S. F. et al.: *A multigrid tutorial*, Siam (2000).
[8] Falgout, R. D.: An Introduction to Algebraic Multigrid (2006).
[9] : AMGCL: Main Page, http://ddemidov.github.io/amgcl/ (Accessed: 2015-11-10).
[10] : Intel MKL PARDISO - Parallel Direct Sparse Solver Interface | Intel® Developer Zone, https://software.intel.com/en-us/node/470282 (Accessed: 2015-11-10).