前処理行列の改良による反復法ソルバーの高速化について

細井聡　　鷲尾巧　　岡田純一　　久田俊明*

1　はじめに

我々は，マルチスケール・マルチフィジックス心臓シミュレータを開発している。そのシミュレーション時間の大半をミクロ部（心筋細胞）の計算が占め，さらにその中でもソルバー部の占める割合が大きい。このようなわけで，ソルバーの高速化はマルチスケール・マルチフィジックス心臓シミュレータにとって極めて重要である。

GMRES[1] を始めとする Krylov 部分空間法の収束性に関する問題点は，生成される Krylov 部分空間が初期誤差ベクトルの低周波成分を含むようになるまで収束が非常に緩やかであることである [2]。そこで，係数行列 A の小さな固有値に対応する反復解の誤差を素早く除去して反復回数を削減することを考える（前処理行列 M が適用される場合は AM^{-1} を係数行列と見なす)。

一方，非線形問題においては，連立 1 次方程式を繰り返し解くが，充分短期間にはその左辺の係数行列は大きくは変化しないと考えられる。そこで，既に解いた方程式から得られる情報を用いて前処理行列を改良し，新規にこれから解く方程式に対する反復回数を削減することを試みる。GMRES の場合

$r_0 = b - Ax_0;\quad \beta = \|r_0\|_2;\quad v_1 = r_0/\beta$
for　$j = 1, \ldots$
　$w_j = M^{-1}v_j$
　$v_{j+1} = Aw_j$
　for　$i = 1, j$
　　$h_{i,j} = v_{j+1}^T v_i;\quad v_{j+1} = v_{j+1} - h_{i,j}v_i$
　endofor
　$h_{j+1,j} = \|v_{j+1}\|_2;\quad v_{j+1} = v_{j+1}/h_{j+1,j}$
　if (収束条件を満たす) $m = j$ として j ループを終了
endfor
$V_m = [v_1, \cdots, v_m],\quad \bar{H}_m = \{h_{i,j}\}_{1\le i\le j+1, 1\le j\le m}$とする
$\|\beta e_1 - \bar{H}_m\|_2$を最小にする y を求める
$x = x_0 + M^{-1}V_m y$ が解となる

図 1　通常の前処理付きフル GMRES

* 東京大学新領域創成科学研究科人間環境学専攻

- Hessenberg 行列
- 直交基底ベクトル
- 固有値
- 固有ベクトルの張る空間

などを有効に使って低周波モードの素早い除去が可能である [3][4][5][6]。また，一度に 1 つの方程式だけしか解くことができないので，いわゆるグローバル GMRES[7] のような手法は使えない。本原稿では，図 1 に示すような前処理付きフル GMRES の前処理行列の改良を試みる。また、対象は制約条件付きの問題であるので、係数行列は負定値となり通常の ILU 前処理では解くのが困難な問題である。以下、 2 章では基底ベクトルを用いた前処理行列の改良手法とその効果について述べ， 3 章では粗いグリッドを用いた前処理行列の改良とその評価を行ない，最後にまとめを行なう。

2 基底ベクトルを用いた前処理行列の改良

ある部分空間基底ベクトルを用いて，元の前処理行列 M を $\tilde{M}$ に改良することを考える。ここで，ある部分空間とは $(AM^{-1})^T AM^{-1}$ の小さな固有値で張られる部分空間である。具体的には，図 2 のように通常の GMRES（図 1）を呼び出す前に (a)(b) の処理を行ない，GMRES のメインループ（j ループ）中の $w_j = M^{-1}v_j$ を $w_j = \tilde{M}^{-1}v_j$（図 2(c)）に変更する。ここで，$\tilde{M}$ に相当する処理を図 3 に示す。これは，陽に $\tilde{M}$ という行列を生成するわけではないが図 2(c) において v_j から w_j を生成する際に元の前処理行列 M の代わりに $\tilde{M}$ を適用したと見なすことができる。以上が，前処理行列を改良するという意味である。

なお，前処理行列を改良した効果を評価する指標として以下の 2 つを用いる。

$$\text{反復回数比} = \frac{\text{改良した前処理行列}\tilde{M}\text{を用いた場合の反復回数}}{\text{元の前処理行列 } M \text{ を用いた場合の反復回数}} \tag{1}$$

$$\text{スピードアップ} = \frac{\text{元の前処理行列 } M \text{ を用いた場合の実行時間}}{\text{改良した前処理行列}\tilde{M}\text{を用いた場合の実行時間}} \tag{2}$$

反復回数比の値が小さい程，反復回数が減少したことを表す。

2.1 実装についての補足

- 図 3(c_2): $W = AM^{-1}\hat{U}$（ただし $\hat{U} = [\hat{\mu}_1 \cdots \hat{\mu}_L](1 \le k \le L)$）を計算し
$$W^T W[\alpha_1 \cdots \alpha_L]^T = W^T v_j \tag{3}$$
を満たす $\alpha_k(1 \le k \le L)$ を求める。$W^T W$ を GMRES メインループ（j ループ）に突入する前に LU 分解あるいは特異値（SVD）分解しておけば，メインループ中では前進・後退代入あるいは V, w, U 行列との乗算により解を求めることができる。
- 図 3(c_3): $\hat{v}_j = \sum_{k=1}^{L} \alpha_k M^{-1}\hat{\mu}_k = M^{-1}[\hat{\mu}_1 \cdots \hat{\mu}_L][\alpha_1 \cdots \alpha_L]^T = (M^{-1}\hat{U})[\alpha_1 \cdots \alpha_L]^T$ として計算する。
- 反復法ソルバーにおいて疎行列ベクトル積は実行時間の大きな割合を占める。そこで、$M^{-1}\hat{U}$ および $AM^{-1}\hat{U}$ の計算においては複数の疎行列ベクトル積を同時に実行するような実装として高速化した。同時に実行することにより、間接参照が減り、BLAS Level3 の利用が可能となり，また，効率的な SIMD 命令の生成が可能となる。

(a) 前回の $GMRES$が m 回で収束した場合, $\bar{H}_m^T \bar{H}_m$の
$s = min(m, l)$ 個の小さな固有値に対応した
基底ベクトル$\hat{\mu}_1, \ldots, \hat{\mu}_s$を選ぶ
ただし $\bar{H}_m : (m+1) \times m$ の $Hessenberg$行列
l : あらかじめ定めた正定数
$\hat{\mu}_k = V_m \mu_k (1 \leq k \leq s)$
$\mu_k : \bar{H}_m^T \bar{H}_m$の固有ベクトル
$(b)(a)$ で選択した s 個の基底ベクトル
をこれまで収集した基底ベクトルの
集合 P に追加

$r_0 = b - Ax_0; \quad \beta = \|r_0\|_2; \quad v_1 = r_0/\beta$
$for \quad j = 1, \ldots$
$w_j = \tilde{M}^{-1} v_j \ (c)$
$v_{j+1} = Aw_j$
$for \quad i = 1, j$
$h_{i,j} = v_{j+1}^T v_i; \quad v_{j+1} = v_{j+1} - h_{i,j} v_i$
$endofor$
$h_{j+1,j} = \|v_{j+1}\|_2; \quad v_{j+1} = v_{j+1}/h_{j+1,j}$
if (収束条件を満たす) $m = j$ として j ループを終了
$endfor$
$V_m = [v_1, \cdots, v_m], \quad \bar{H}_m = \{h_{i,j}\}_{1 \leq i \leq j+1, 1 \leq j \leq m}$とする
$\|\beta e_1 - \bar{H}_m\|_2$を最小にする y を求める
$x = x_0 + M^{-1} V_m y$ が解となる

図 2　改良された前処理行列を用いたフル GMRES

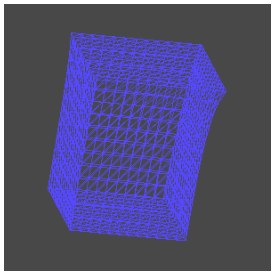

図 4　超弾性体

(c_1) これまでに収集してきた基底ベクトルの集合 P
から L 個の基底ベクトル$\hat{\mu}_k (1 \leq k \leq L)$ を選ぶ[a]
$(c_2)\tilde{v}_j = AM^{-1} \sum_{k=1}^{L} \alpha_k \hat{\mu}_k$とおき
$\|v_j - \tilde{v}_j\|_2$を最小にする$\alpha_k (1 \leq k \leq L)$ を求める
$(c_3)\hat{v}_j = M^{-1} \sum_{k=1}^{L} \alpha_k \hat{\mu}_k$
$(c_4)\tilde{r} = v_j - \tilde{v}_j$
$(c_5) M \Delta v_j = \tilde{r}_j$を解く
$(c_6) w_j = \hat{v}_j + \Delta v_j$

図 3　図 2(c) の $\tilde{M}^{-1}$ に相当する処理

[a] L 個の選び方としては 2.2 で示すように，1 回前の方程式の情報のみ用いる，過去に解いた全ての方程式の情報を用いるなど幾つかのヴァリエーションが考えられる。

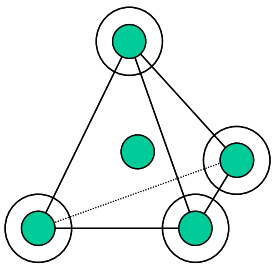

図 5　四面体有限要素

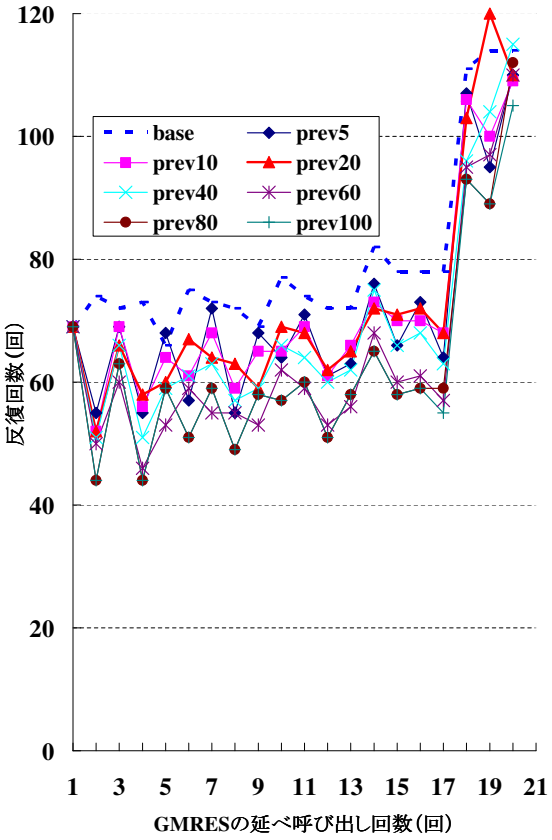

図 6　基底ベクトルを用いた前処理行列改良の効果
（1 回前に解いた方程式の情報のみを用いた場合）

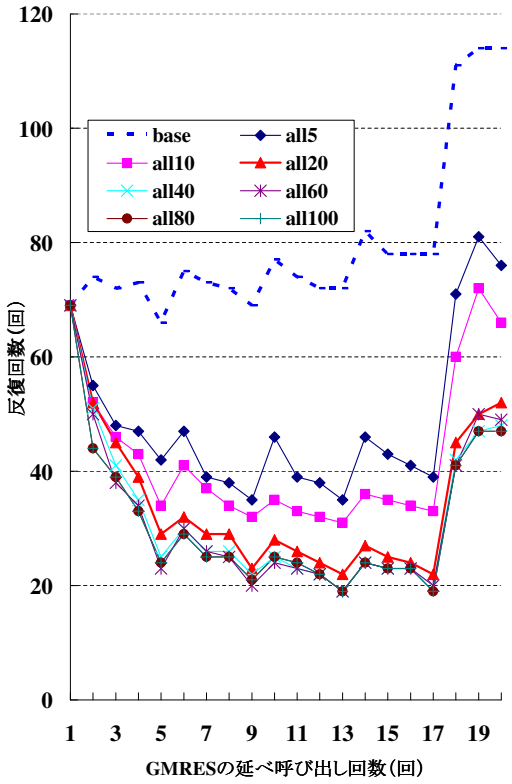

図 7　基底ベクトルを用いた前処理行列改良の効果
(過去に解いた全ての方程式の情報を用いた場合)

2.2　数値実験例

図 4 のような 3 次元超弾性体 (節点数 $16 \times 16 \times 16$ + バブル節点) の伸縮シミュレーションプログラムの線形ソルバーに対して本手法を適用した．離散化は, 4 面体 MINI 要素を用いて行った. すなわち, 4 面体の頂点に加えその重心に変位節点を追加し, 各頂点に静水圧点を加えたものである（図 5)。混合要素タイプの有限要素離散化を適用しているので係数行列は対称ではあるが静水圧の自由度の数だけ負の固有値を有し，通常の ILU 前処理では解くことが困難な問題である．本例題においては, 5 段階で境界応力を増加させたので, 全部で 5 回の非線形問題を解いている.

ここでは, 元の前処理行列M として文献 [8] により提案されたものを用い，変位自由度を消去する際に圧力対角に生じる fill-in を取り込むように ILU 分解する。そして，相対残差が 10^{-8} 以下となるまでの反復回数を比較する。

図 6 の base は元の前処理行列 M のみを用いた場合の反復回数である。一方，prev$k$ は $l = k$ とし，1 回前に解いた方程式のみの基底ベクトルを用いて前処理行列を改良した場合の反復回数で，反復回数比は大半の GMRES の呼び出しにおいて 0.7〜1.0 超であり，反復回数を充分削減できたとは言い難い。

図 7 の all$k$ は $l = k$ とし，過去 t 回解いた方程式に対して毎回 $min(m_k, l)$ $(1 \leq k \leq t)$ 個の小さな固有値

に対応した基底ベクトルを継続して収集し，合計 $L = \sum_{k=1}^{t} min(m_k, l)$ 個の基底ベクトルを用いて前処理行列を改良した場合の反復回数を表す（ただし，$m_k (1 \leq k \leq t)$ は，k 回目の GMRES 呼び出しの反復回数)。これから，以下のことが言える。

- $l \geq 20$ の場合の反復回数比に大きな差はない
- GMRES の述べ呼び出し回数が 3 回以下の場合は反復回数比がまだ充分小さくはない。この時点ではまだ充分なスピードアップが得られていないと予想される。
- 実行が進み，用いる基底ベクトル数が増えるに従い反復回数比は小さくなる。$l = 20$ の場合，GMRES の延べ呼び出し回数が 7 回目で反復回数比が始めて 0.4 以下となるが，そのためには基底ベクトルが既に 140 個必要となっている。

また，図 8 において以下のことが言える。

- first20 と first20×4 は，それぞれ最初の 1 回の GMRES 呼び出しのみの 20 個，最初の 4 回の GMRES 呼び出しのみの 20×4 個の基底ベクトルのみを用いた場合の反復回数である。いずれも次第に反復回数比は増加してしまうので，これでは最終的には充分なスピードアップが得られない。
- periodic1 は，4 回の GMRES 呼び出しにに 1 回収集した基底ベクトルを全て捨て再度収集を開始する場合である。定期的に反復回数比が 1 に戻ってしまうのが最大の問題であり，平均のスピードアップが充分に得られない。
- periodic2 は，最初の 4 回の GMRES の間基底ベクトルを収集し続け，次の 4 回は収集を中断し（ただし，これまでに収集した基底ベクトルは保持し続ける)，また次の 4 回は収集を継続 $\cdots$ とした場合である。periodic1 のように定期的反復回数比が 1 に戻ってしまうことはないし，all に比べれば用いる基底ベクトル数は少なくて済む。しかし，やはり反復回数比は増加傾向にあり，基底ベクトル数の増加も避けられないので，結局は十分なスピードアップは得られない。

以上のように，過去に解いた方程式のうちの一部の情報のみを用いれば反復回数は次第に増加してしまう。一方，全ての情報を用いれば基底ベクトル数が増大してしまい，いずれにしろ充分なスピードアップが得られないと考えられる。

一部の方程式の情報を用いるだけでは不充分で，過去に解いた方程式の情報を収集し続けなければならない理由の 1 つとして，右辺の違いの影響が考えられる。たとえば，$Ax = b$ を解いた結果得られる全ての基底ベクトルを用いて前処理行列を改良する場合を考える。もし，この前処理行列を全く同じ連立 1 次方程式に再び適用すると，丸め誤差の影響を無視すれば 1 回の反復で GMRES は収束する。しかし，右辺のみを変更した $Ax = b'$ に適用すると反復回数比はせいぜい 2/3 程度にしかならない。すなわち，1 つの連立 1 次方程式を解いた結果得られる基底ベクトルを全て用いても任意の右辺に対する誤差の低周波成分を表現するのにはまだ充分ではないと考えられる。

また，図 3 の前処理行列を改良する処理は，GMRES メインループ (j ループ) 中で毎回実行しないと反復回数をほとんど削減できないか場合によっては返って反復回数を増やしてしまう結果となる（図 9 の onlyInit)。これも，1 回だけでは初期誤差の低周波成分を全て除くことはできないためと考えられる。

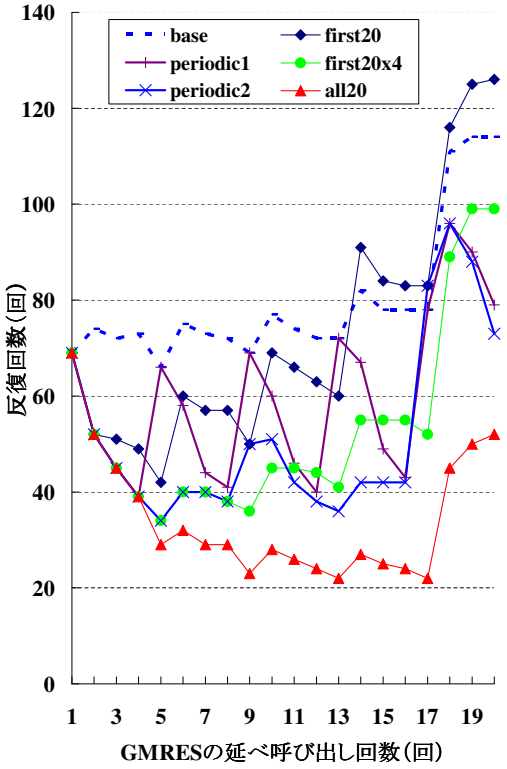

図 8　最初の基底ベクトルのみ用いる場合

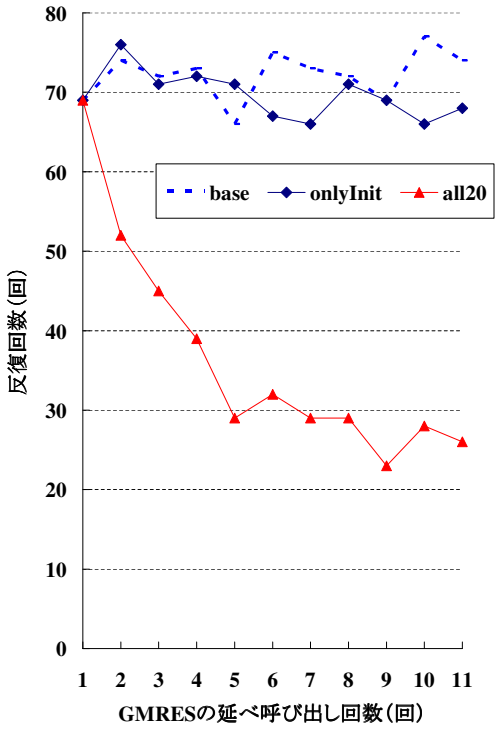

図 9　初期化時にのみ前処理行列の改良を行なう場合

2.3　基底ベクトルの線形結合を用いて基底ベクトル数を削減

2.2 の手法は，過去に解いた全ての方程式の情報を用いて始めて反復回数比を 0.3 程度にできるが，用いなければならない基底ベクトル数が多過ぎるという問題があった。そこで，極力少数の基底ベクトルで低周波部分空間を効率良く表現するために，収集した基底ベクトルの線形結合を用いることを考える。具体的には，t 回目の GMRES の呼び出しの反復回数を削減するための基底ベクトルを以下のようにして求める。なお，以下で m_k は k 回目の GMRES 呼び出しの反復回数である。

- $t = 1$ の場合，利用可能な基底ベクトルがないので通常のフル GMRES を実行
- $t = 2$ の場合，$t = 1$ 回目の GMRES の呼び出しの結果得られた基底ベクトルのうち小さな固有値に対応した $L_1 = min(m_1, l)$ 個を用いて前処理行列を改良
- $t \geq 3$ の場合，$t-2$ 回目の GMRES の呼び出しに用いた L_{t-2} 個の基底ベクトルを $\hat{\mu}_{t-2,k}(1 \leq k \leq L_{t-2})$，$t-1$ 回目の GMRES の呼び出しの結果得られる $L_{t-1} = min(m_{t-1}, l)$ 個の基底ベクトルを $\hat{\mu}_{t-1,k}(1 \leq k \leq L_{t-1})$ として，以下により t 回目の GMRES の呼び出し時に用いる $L_t = min(L_{t-2} + L_{t-1}, l)$ 個の基底ベクトルを求める。
  1. $\hat{\mu}_{t-1,k}(1 \leq k \leq L_{t-1})$ を $\hat{\mu}_{t-2,k}(1 \leq k \leq L_{t-2})$ に対して直交化
  2. $\hat{U} = [\hat{\mu}_{t-2,1} \cdots \hat{\mu}_{t-2,L_{t-2}}\ \hat{\mu}_{t-1,l} \cdots \hat{\mu}_{t-1,L_{t-1}}]$ とし $(AM^{-1}\hat{U})^T AM^{-1}\hat{U}$ の固有値を求め，そのうちの小さな L_t 個に対応した固有ベクトル $\mu_k(1 \leq k \leq L_t)$ を選ぶ

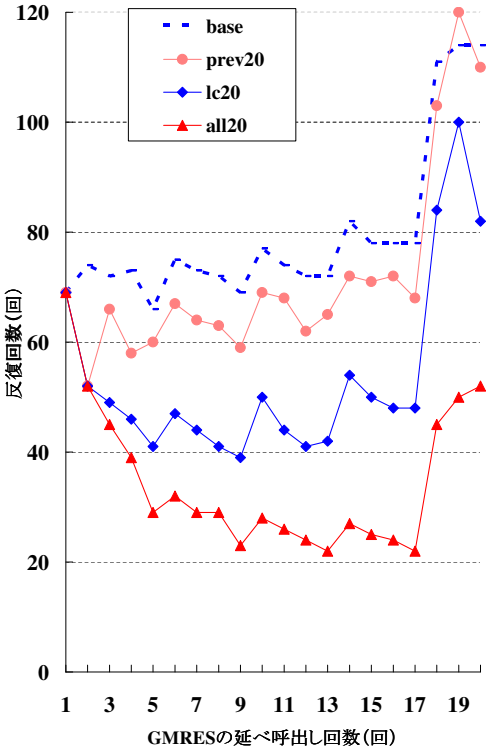

図 10　基底ベクトルを線形結合した場合の反復回数

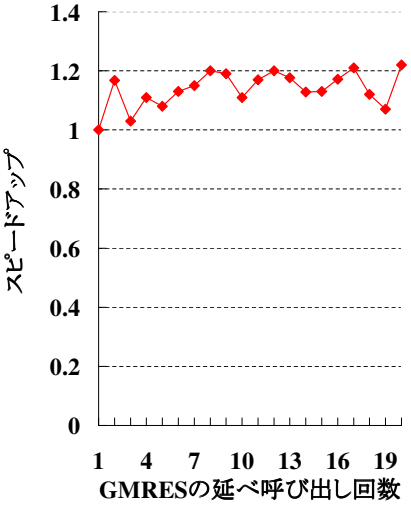

図 11　基底ベクトルの線形結合を用いた場合のスピードアップ

3. $\hat{\mu}_{t,k} = \hat{U}\mu_k (1 \leq k \leq L_t)$ を新たな基底ベクトルとし，t 回目の GMRES 呼び出しの反復回数削減に用いる

図 10 の lc20 は $l = 20$ として線形結合した基底ベクトルを用いた場合の反復回数である。1 つ前の方程式の情報のみを用いる場合（prev20）より反復回数比は充分小さくなっており，最小で 0.6 弱となっている。しかし，過去に解いた全ての方程式の情報を用いた場合（all20）に比べると反復回数比はまだ充分小さくなっているとは言えない。

次に，基底ベクトル線形結合を用いた場合のスピードアップを図 11 に示す。最大でも 20% のスピードアップしか得られておらず，実行環境により優位な差となって現れない場合もあり得ると考えられる。

3　粗いグリッドを用いた前処理行列の改良

2 では元の自由度と同じベクトルを用いたので、たとえそれらの線形結合を用いても初期誤差の低周波成分を完全に表現するには多くのベクトルが必要であった。一方、粗いグリッドを用いれば少数のベクトルでそれを表現できると考えられる。そこで、前処理行列の改良に粗いグリッドを併用することを考える [1][9][10]。マルチスケール・マルチフィジックス心臓シミュレータのミクロ部では実際の心筋細胞を模擬した図 12 のような数値細胞の動きをシミュレートする。図 12 の右端の図は 2 つの心筋細胞を連結したモデルである。図 12 の数値細胞に対して細かいメッシュ (fine メッシュ) と粗いメッシュ（coarse メッシュ）を作り，これら 2 つのメッシュを用いて前処理行列の改良を試みる。しかし，問題の規模や複雑さを考えると，実際の数値細胞に

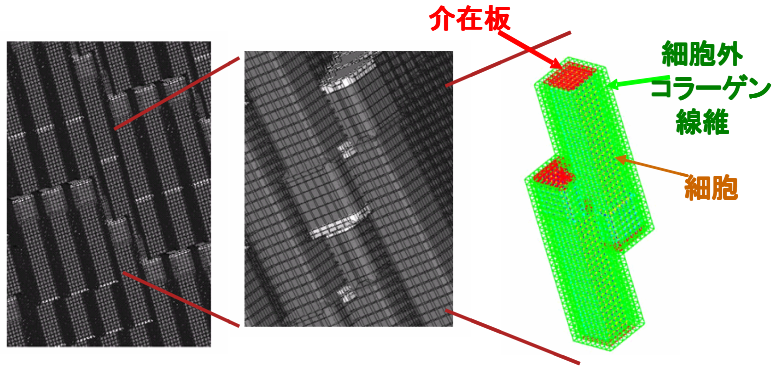

図 12　数値細胞

表 1　マテリアル情報

ギャップ・ジャンクション	$\{(x,y,z)\|2 \leq x \leq 15, 2 \leq y \leq 15, z = 1 \ or \ 76\}$
筋原繊維	$\{(x,y,z)\|2 \leq x \leq 15, 2 \leq y \leq 15, 2 \leq z \leq 75\}$
細胞外セル	上記以外の範囲の要素

最初から取り組むのは必ずしも得策ではない。そこで，まずは図 4 のような単純な直方体を 1 個の心筋細胞に見立て，六面体有限要素を用いて細かいメッシュと粗いメッシュを作成する。六面体要素の各頂点には変位節点（自由度 3），重心には圧力節点（自由度 1）を配置する。そして，このような簡単な例題に対して粗いグリッドを用いて前処理行列を改良する効果を確かめることにする。

図 4 に対して以下のような fine メッシュを作成して，心筋細胞 1 個を模擬する。

- 直方体の x, y, z 方向の長さの比を 1.0 : 1.0 : 5.0 とし，fine メッシュはこれを $15 \times 15 \times 75$ 個の六面体有限要素で分割する（節点数は $16 \times 16 \times 76$ 個）。
- fine メッシュの各有限要素に対して表 1 のようにマテリアル情報を与える。なお，ここで (x, y, z) は直方体の節点座標を表す。

3.1　coarse メッシュから fine メッシュへの変換

coarse メッシュから fine メッシュの生成は、異なるマテリアル情報を跨がないように行なう。また、coarse メッシュから fine メッシュへの変換に際しては，変位節点と圧力節点とを以下のように分けて考える。

- fine メッシュ上における変位は，coarse メッシュ上での変位から空間補完して求める。たとえば図 13 は coarse メッシュ上の節点の中間点に fine メッシュ上の節点を生成した場合に、fine メッシュでの変位を求めた様子を示している。
- fine メッシュ上の圧力は，coarse メッシュにおける圧力値の恒等写像により求める。図 14 は図 13 と同様な fine メッシュを生成した場合に、coarse メッシュ上の各有限要素の重心に位置する圧力を恒等写像した場合である。

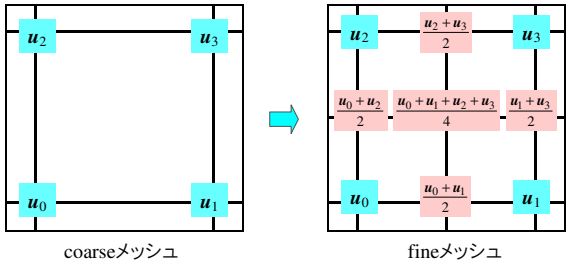

図 13　coarse メッシュから fine メッシュへの変換例（変位）

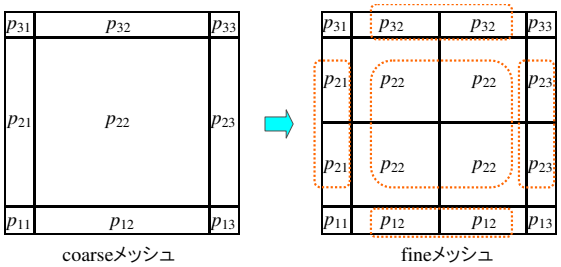

図 14　coarse メッシュから fine メッシュへの変換例（圧力）

$(a) B = AP, A_c = P^T B$ を計算
$(b) A_c$の LU 分解結果を改めて A_cとする

$r_0 = b - Ax_0;\quad \beta = \|r_0\|_2;\quad v_1 = r_0/\beta$
for　$j = 1, \ldots$
　$w_j = \tilde{M}^{-1} v_j$ (c)
　$v_{j+1} = Aw_j$
　for　$i = 1, j$
　　$h_{i,j} = v_{j+1}^T v_i;\quad v_{j+1} = v_{j+1} - h_{i,j} v_i$
　endofor
　$h_{j+1,j} = \|v_{j+1}\|_2;\quad v_{j+1} = v_{j+1}/h_{j+1,j}$
　if (収束条件を満たす) $m = j$ として j ループを終了
endfor
$V_m = [v_1, \cdots, v_m],\quad \bar{H}_m = \{h_{i,j}\}_{1 \le i \le j+1, 1 \le j \le m}$とする
$\|\beta e_1 - \bar{H}_m\|_2$を最小にする y を求める
$x = x_0 + M^{-1} V_m y$ が解となる

図 15　粗いグリッドを用いて改良された前処理行列を用いたフル GMRES

$(c_1) A_c q_c = P^T v_j$を前進・後退代入により解く
$(c_2) q_0 = P q_c$
$(c_3) \tilde{v}_j = v_j - Aq_0 = v_j - Bq_c$
$(c_4) M q_1 = \tilde{v}_j$を解く
$(c_5) w_j = q_1 + q_0$

図 16　図 15(c) の $\tilde{M}^{-1}$ に相当する処理

今 coarse メッシュから fine メッシュへの変換を P とするとき[*1]，fine メッシュ上での連立 1 次方程式の左辺の係数行列 A に対応する coarse メッシュ上での係数行列 A_c は

$$A_c = P^T A P \tag{4}$$

で与えられる。今，GMRES のメインループ中において $Aq = r$ を解く場合を考える。両辺に左から P^T を掛けて

$$P^T A q = P^T r \tag{5}$$

[*1] この P は図 14 の p_{ij} とは異なり，変位と圧力の自由度とを合わせた coarse メッシュ上のベクトルを fine メッシュ上へのベクトルへ変換する行列を表す。

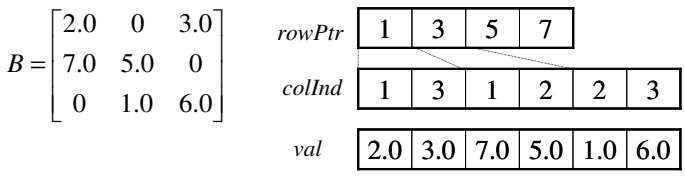

図 17　CRS 形式による疎行列の格納

ここで

$$q = Pq_c \tag{6}$$

とおけば

$$A_c q_c = P^T r \tag{7}$$

となり，これが coarse メッシュ上で解く連立 1 次方程式となる。以上を用いた前処理行列を改良するアルゴリズムの概要を図 15 および図 16 に示す。

- A_c の LU 分解は GMRES のメインループに突入する前に実行しておき（図 15(b)），メインループ内では前進・後退代入のみで coarse メッシュ上での連立 1 次方程式を解く（図 16(c_1)）。
- coarse メッシュ上での連立 1 次方程式の解 q_c を変換 P を用いて fine メッシュ上に写像する（図 16(c_2)）。

3.2　実装についての補足

- 粗いグリッドを用いた前処理行列の改良においては，coarse メッシュから fine メッシュへの変換行列 P は全実行中不変であることを仮定した。よって，P は最初に一度だけ生成すれば良い。
- A の非零要素の位置は全実行中不変。よって，$B = AP$ の非零要素の位置も全実行中不変。
  - 具体的には B の各要素は

$$B_{ij(1\leq N\leq i, 1\leq j\leq N_c)} = \begin{cases} 0 & \forall_{k(1\leq k\leq N)}\ \ A_{ik} = 0 \text{ あるいは } P_{kj} = 0 \\ \displaystyle\sum_{k=1}^{N} A_{ik}P_{kj} & otherwise \end{cases} \tag{8}$$

    となる（ここで，N は A の行数，N_c は A_c の行数)。なお，$\exists_{k(1\leq k\leq N)}\ \ A_{ik} \neq 0$ かつ $P_{kj} \neq 0$ ではあるが $\displaystyle\sum_{k=1}^{N} A_{ik}P_{kj} = 0$ となる B_{ij} は非零要素として処理する。
  - B に対していわゆる CRS 形式（図 17）の疎行列用のデータ構造を用いる場合
    - B の各行の最初の非零要素へのポインタを表す配列 $rowPtr$，および，各非零要素の列位置を表す配列 $colInd$ の領域確保と値の設定は最初に一度だけ行なえば良い。
    - 非ゼロ要素の値を保持する配列 val の領域確保は最初に一度だけ行なえば良く，次回以降は値のみ変更すれば良い。
- 図 16(c_3) における Aq_0 の計算は，$(AP)q_c$ により計算した方が効率的である。よって，AP の計算結果 B を保持しておく。

表 2　fine メッシュおよび coarse メッシュ

	変位節点座標 $\{(x,y,z)\|x \in N_x, y \in N_y, z \in N_z\}$	総変位 節点数	総圧力節点数 および 総要素数	総自由度 （総自由度比）
(a)fine	$N_x = \{x\|1 \le x \le 16\}, N_y = \{y\|1 \le y \le 16\},$ $N_z = \{z\|1 \le z \le 76\}$	19456	16875	75243 (1.000)
(b)coarse4×4×4	$N_x = N_y = \{1, 2, 15, 16\}$ $N_z = \{1, 2, 75, 76\}$	64	27	219 (0.003)
(c)coarse5×5×5	$N_x = \{1, 2, 8, 15, 16\}, N_y = \{1, 2, 9, 15, 16\},$ $N_z = \{1, 2, 38, 75, 76\}$	125	64	439 (0.006)
(d)coarse7×7×7	$N_x = N_y = \{1, 2, 5, 8, 12, 15, 16\}$ $N_z = \{1, 2, 20, 38, 55, 75, 76\}$	343	216	1245 (0.017)
(e)coarse7×7×11	$N_x = N_y = \{1, 2, 5, 8, 12, 15, 16\}$ $N_z = \{1, 2, 11, 20, 29, 38, 48, 57, 66, 75, 76\}$	539	360	1977 (0.026)
(f)coarse7×7×13	$N_x = N_y = \{1, 2, 5, 8, 12, 15, 16\}$ $N_z = \{1, 2, 9, 16, 23, 30, 37, 47, 54, 61, 68, 75, 76\}$	637	432	2343 (0.031)

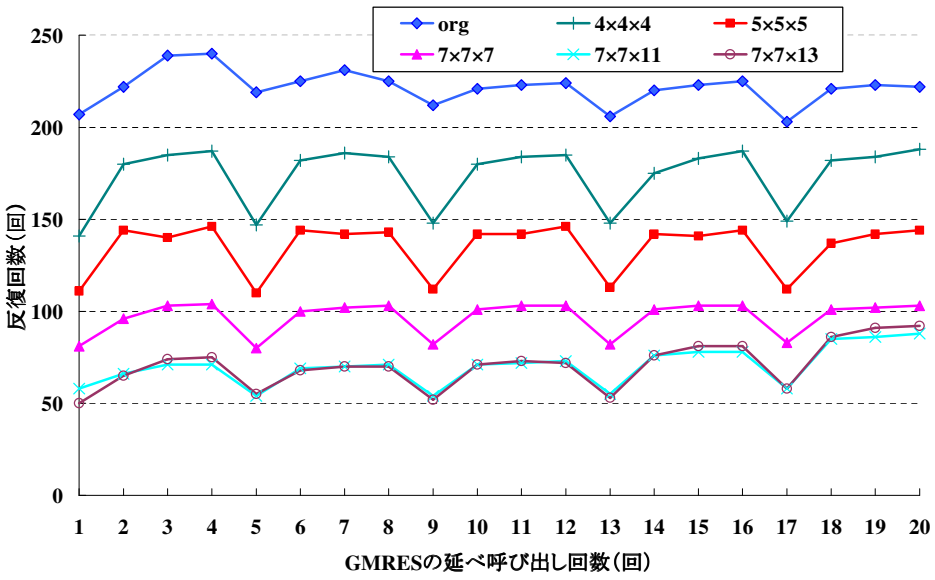

図 18　粗いグリッドを用いて前処理行列を改良した場合の反復回数

- $A_c = P^T AP$ も非零要素の位置は常に不変である。
- A_c の LU 分解と前進・後退代入には，Lapack(dgetrt+dgetrs) と後藤 BLAS を用いた。

3.3　数値実験例

2.2 と同様な伸縮シミュレーションを行ない，表 2(a) に示す fine メッシュに対して元の前処理列 M を適用した場合と，その fine メッシュと表 2(b)〜(f) のいずれか 1 つの coarse メッシュとを併せて用いて前処理行列を改良した場合の性能や反復回数を比較する。なお、表 2 中の N_x, N_y, N_z はそれぞれ x, y, z 方向の節点

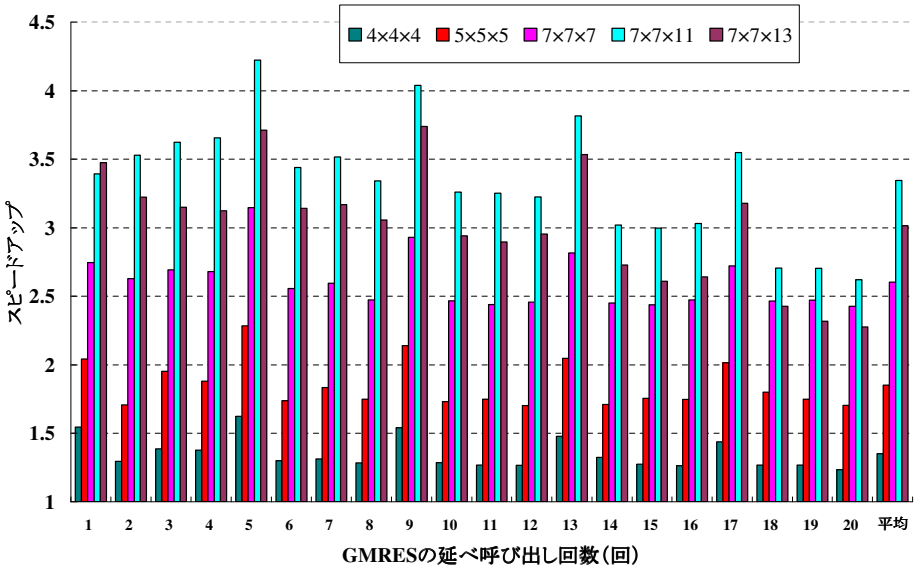

図 19　粗いグリッドを用いて前処理を改良した場合のスピードアップ

の集合を表す。また、総自由度比とは，fine メッシュの総自由度数に対する各メッシュの総自由度数である。

図 18 において，org は fine メッシュに対して元の前処理列 M を適用した場合の反復回数，4×4×4 は fine メッシュと表 2(b) の coarse4×4×4 を併せて用いて前処理行列を改良した場合の反復回数を表す（以下，5×5×5 などについても同様)。また，図 19 において，各 GMRES の呼び出しに対する 5 本の棒のうち最左は coarse4×4×4 を併せて用いた場合のスピードアップを表している（以下順に，coarse5×5×5 を併用した場合 ··· で，最右が coarse7×7×13 を併用した場合のスピードアップである)。なお，最右の 5 本は 20 回の GMRES の呼び出しのスピードアップの平均値である。図 18 および図 19 より以下のことが言える。

- 表 2 の中で最も粗いメッシュである coarse4×4×4 を用いた場合，反復回数比は 0.7〜0.8 程度であるが最大で 1.6 倍以上のスピードアップが得られている。基底ベクトルを用いた手法と比べると，反復回数比の割にはスピードアップが大きい。しかし，平均のスピードアップは 1.35 倍でありまだ充分ではない。
- 一方，coarse7×7×7 や coarse7×7×11, $7 \times 7 \times 13$ を用いた場合は平均で 2.5 倍以上のスピードアップが得られている。特に coarse7×7×11 の場合最大で 4.2 倍，平均で 3.3 倍のスピードアップとなっている。
- coarse7×7×11 と coarse7×7×13 との反復回数ほとんど変わらず，ごく僅かではあるがむしろ後者の方が少ない時もあるのに，スピードアップは常に前者の方が大きく最大で 15% 以上，平均でも 10% 以上高速である。以上の理由については現在調査中である。

4　まとめ

問題の非線形性を利用した反復法ソルバーの高速化についての検討を簡単な例題に対して行なった。

1. まず，既に解いた連立 1 次方程式から得られる基底ベクトルを用いて前処理行列を改良し反復回数の削減を試みた。過去に解いた全ての方程式から得られる基底ベクトルを収集し続けた場合，最初の数回の GMRES 呼び出し以降は反復回数比を 0.3 前後とすることができた。しかし，実行が進むに従い用いる

基底ベクトル数は単調増加してしまうので，次第に前処理行列を改良するコストも増加し充分なスピードアップが得られないようになってしまう。逆に，どこかで基底ベクトルの収集を中止してしまうと次第に反復回数比は多くなってしまうので，やはり充分なスピードアップは得られない。
繰り返し解く連立 1 次方程式の右辺の違いに対処しなければならないこと，および，非線形問題とはいえ充分時間が経てば左辺の係数行列も大きく変化してしまうと考えられることが基底ベクトルの収集を継続しなければならない理由の一因であると考えられる。

2. 次に，基底ベクトルの収集を継続する一方でその数の増加を抑制するために，得られた基底ベクトルの線形結合を用いることを試みた。一部の方程式から得られる情報のみを用いた場合と比べると，用いる基底ベクトル数を一定以下に抑えつつ，反復回数比を小さくし続けることが可能となった。しかし，基底ベクトル数に制限を設けないで収集し続けた場合に比べるとまだ反復回数比が大きく，実際のスピードアップも高々 1.2 程度であった。
3. さらに，粗いグリッドを用いて前処理行列を改良することを試した。非常に簡単な例題に対してではあるが，最も効果があった場合には反復回数比を 0.25 前後に保つことができ，また，併用する coarse メッシュにより平均で 2.5〜3 倍強のスピードアップが得られた。

今回用いた例題に比べると細胞はずっと複雑であるが，今後，粗いグリッドを用いて前処理行列を改良する手法をベースにミクロ部のソルバーの高速化に取り組む予定である。

謝辞

本研究は，JST 産学協同シーズイノベーション化事業育成ステージ「心臓シミュレータの医療への実用化研究」，並びに，平成 20 年度 T2K オープンスパコン (東大) 共同研究プロジェクトの一環として行なわれました。東京大学情報基盤センター・中島研吾先生を始めとする関係各位に感謝の意を表します。また，本論文で用いた 3 次元超弾性体伸縮シミュレーションプログラムは，東京大学大学院新領域創成科学研究科の渡邊浩志先生からご提供頂きました。ここに厚く御礼申し上げます。

参考文献

[1] Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, second edition, 2003.

[2] H. A. van der Vorst and C. Vuik. The superlinear convergence behaviour of GMRES. *J. Comput. Appl. Math.*, Vol. 48, No. 3, pp. 327–341, 1993.

[3] Ronald B. Morgan. GMRES with deflated restarting. *SIAM J. Sci. Comput.*, Vol. 24, No. 1, pp. 20–37, 2002.

[4] Andrew Chapman and Yousef Saad. Deflated and augmented Krylov subspace techniques. *Numer. Linear Algebra Appl*, Vol. 4, pp. 43–66, 1996.

[5] Kevin Burrage. On the performance of various adaptive preconditioned GMRES strategies, 1997.

[6] J. Baglama, D. Calvetti, G. H. Golub, and L. Reichel. Adaptively preconditioned GMRES algorithms. *SIAM J. Sci. Comput*, Vol. 20, pp. 243–269.

[7] Gerard. A. Meurant. *The Lanczos and Conjugate Gradient Algorithms:From Theory to Finite Precision Computations*. SIAM, 2006.

[8] T.Washio, T.Hisada, H.Watanabe, and T.E.Tezduyar. A robust preconditioner for fluid–structure interaction problems. *Comput. Methods Appl. Mech. Engrg.*, 2005.

[9] George Em Karniadakis and Robert M. Kirby. *Parallel Scientific Computing in C++ and MPI.* CAMBRIDGE UNIVERSITY PRESS, 2003.

[10] Thoman. *Multigrid Methods on GPUs.* VDM Velag, 2008.