RC-004

インタラクティブシミュレーションへの応用を前提としたマルチコアプロセッサ上でのSMW公式を用いた高速逆行列計算

岩永　翔太郎, 福間 慎治, 森 眞一郎

Shotaro Iwanaga, Shinji Fukuma, Shin-ichiro Mori

1　はじめに

近年の計算機性能の急速な向上に伴い, インタラクティブな実時間数値シミュレーションへの期待が高まっている. 中でも実時間内での数値シミュレーションは, 大規模な 3 次元データを必要とする医療などの分野において高度な技術が要求されている分野である. 我々は次世代のシミュレーション技術として, 数値シミュレーションにユーザが直接介入し, シミュレーションのシナリオを実時間かつ対話的に変更するインタラクティブシミュレーション技術の研究を行っている.

本研究では, 多くのシミュレーション中に現れる線形方程式 $Ax = b$ の求解問題に着目する. 特に, シミュレーション中のユーザからのインタラクションや時間経過によって係数行列 A が微小な変化を繰り返す状況でのリアルタイムシミュレーションの実現を研究のターゲットとする.

通常, このような線形方程式の求解問題では係数行列 A の逆行列を求めて解 x を計算する手法は計算に必要な計算量, ならびに記憶領域の問題から推奨されていない.[2] しかしながら, A の近似行列 A' の逆行列 $(A')^{-1}$ が既知の場合, この逆行列を用いた補正計算により高速に A^{-1} を求める方法が存在すれば, A の逆行列を導出して方程式の解を求める方法も, 十分に有効な手法になりえる. このような目的で使用可能な数学公式のひとつに SMW 公式 (Sherman-Morrison-Woodbury formula)[1, 3] がある.

本論文では,SMW 公式を用いた逆行列計算に基づく線形方程式求解問題をマルチコア CPU 上に実装し, 近似行列の近似度の影響, 並列化効果, ならびに, 係数行列が疎行列の場合の効果を調査し, インタラクティブシミュレーションへの応用の可能性を検討した結果を報告する.

2　研究の背景

2.1　SMW 公式による逆行列計算

問題サイズ $n \times n$ の行列 A に対して, 問題サイズ $n \times s$ ならびに $s \times n$ の行列 B ならびに C を考え, A の逆行列を A^{-1} とすると, SMW 公式は次式で与えられる.

$$(A + BC)^{-1} = A^{-1} - A^{-1}B(I + CA^{-1}B)^{-1}CA^{-1} \quad (1)$$

いま,A が時間経過によって A' へと微少変化したと仮定し,

$$A' = A + \Delta A \quad (2)$$

とする. さらに ΔA を

$$\Delta A = \Delta A_c \times I \times E_c \quad (3)$$

[0]福井大学大学院工学研究科, University of Fukui

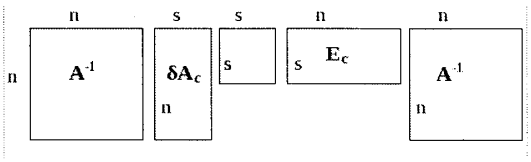

図 1: SMW 公式の行列積部分の行列サイズ

として,$B = \Delta A$,$C = E_c$ と考えると, 式 (1) より $(A')^{-1}$ が求まることになる (式 (4)). ΔA_c は,ΔA から全ての要素が 0 である列を取り除いた行列で, E_c は 0 と 1 からなる行列である.　なお, ΔA の各行での非零要素数のうち最大のものを s とする.

$$(A + \Delta A)^{-1} = A^{-1} - A^{-1}\Delta A_c(I + E_cA^{-1}\Delta A_c)^{-1}E_cA^{-1} \quad (4)$$

式 (4) は $A'^{-1} = A^{-1} -$ (補正項) と見ることができ, A^{-1} を用いて A'^{-1} が求まることを表している. 式 (4) において, 右辺第二項で行う行列積計算に使用する行列サイズの概要を図 1 に示す.

ここで, 行列 $(D^{-1} + CA^{-1}B)^{-1}$ は図 1 の中央部分にある大きさ s×s の行列に対応し, この逆行列を求めるには $O(s^3)$ の時間を要する. その他の行列積の計算では $O(sn^2)$ の時間を要する.

このとき, インタラクティブシミュレーションの連続するタイムステップにおける係数行列の時間変化 (A から A' への変化) は微小 ($n >> s$) なものであると仮定すると,SMW 公式を用いた逆行列計算の計算量は $O(sn^2)$ となる. これは, LU 分解を用いた線形方程式の求解問題に比べて高速に解が求まる可能性を示唆している. 一方, 係数行列 A が疎行列の場合には, ICCG 法などに代表される反復法による線形方程式の求解が広く用いられている. ICCG 法の一回の反復計算に必要な計算量は $O(n^2)$ であり, k 回の反復で実用収束した場合の計算量は $O(kn^2)$ となる. 一般に k の値は n に比べて小さいことが知られており, SMW 公式を用いた求解法との比較においては, s と　k の関係によって優劣が逆転することが考えられる.　また, メモリ使用量の比較では A が疎行列であっても A^{-1} は必ずしも疎行列にはならないため, 実用上は A に含まれる非零要素数程度のオーダのメモリしか使用しない ICCG 法と比較すると, SMW 公式を用いた場合の欠点は否めない. そこで, 以下の議論はメモリ消費量で一意に優劣が決まらない程度の問題サイズ n を仮定する. なお, この仮定は 1 ステップの計算を最悪でも 100msec のオーダで終了することを求めるインタラクティブシミュレーションにおいては, 現在利用可能な計算機資源を考えると現実的な問題設定であると考える. (表 1 参照)

表 1: 各種解法の計算量およびメモリ消費量の比較

	計算量	メモリ消費量
直接法 (LU 分解)	$O(n^3)$	$O(n^2)$
SMW 公式	$O(sn^2)$	$O(n^2)$
ICCG 法 (k 反復分)	$O(kn^2)$	$O(n^2)$

2.2 SMW 公式による逆行列計算プログラムの実装と評価環境

2.2.1 プログラムの実装

プログラムの実装にあたっては，SMW 公式を以下の 8 つのステップに分けて実装を行った．

1. $A^{-1} \times \Delta A_c$
2. $E_c \times A^{-1}\Delta A_c$
3. $E_c \times A^{-1}$
4. $I + E_c A^{-1}\Delta A_c$
5. $(I + E_c A^{-1}\Delta A_c)^{-1}$ を直接法で計算
6. $(I + E_c A^{-1}\Delta A_c)^{-1} \times E_c A^{-1}$
7. $A^{-1}\Delta A_c \times (I + E_c A^{-1}\Delta A_c)^{-1} E_c A^{-1}$
8. $A^{-1} - A^{-1}\Delta A_c (I + E_c A^{-1}\Delta A_c)^{-1} E_c A^{-1}$

ステップ 5,8 はハンドコーディングを行い，それ以外のステップでは数値計算ライブラリ (Intel 社の MathKernelLibrary[MKL]) を用いた．マルチコアプロセッサ環境では，MKL 版の BLAS ライブラリが OpenMP ベースの自動並列化されるため，この時点でステップ 5 以外はマルチコアプロセッサ対応のプログラムとなる．並列化の際は，各ステップの主要部分を構成する多重ループのうち最外ループを均等に分割し，分割した各領域をそれぞれ利用可能なコアに割り当てた．今回の実験では，計算全体に占めるステップ 5 の割合が無視できることを確認のうえ，ステップ 5 の最適化および並列化は行っていない．

なお上述の MKL は，デフォルトでは実行時に利用可能なコア数を自動認識し並列度を決定する．そこで，並列処理効果の検証を行う場合には，実行前に環境変数を設定し利用可能なコア数を明示的に指定した．

2.2.2 実行環境と実験データ

表 2 に今回の実験で使用した実行環境の概要を示す．表 3 には実験に用いた 4 種類の係数行列 A の概要を示す．いずれも，同一の生体臓器を要素数の異なる有限要素モデルで表現した際の剛性マトリックスである．

3 SMW 公式を用いた逆行列計算の評価実験

3.1 逐次処理環境による予備実験

並列処理による高速化効果を検証するため，まずは実行スレッド数を 1 に設定して実験を行った．

表 2: 実験環境

CPU	IntelCore2Quad Q6600 2.4GHz
Memory	2GB
L2 Cache	4MB×2
OS	Linux2.6.23.17-88.fc7
compiler	icc 10.1
compiler option	-O3
library	MKL 10.0.011

表 3: 実験に用いた係数行列の性質

問題サイズ (n)	780	1089	2427	3759
非零要素数の割合 [%]	4.36	3.17	1.50	1.00

最初の実験では，s の値を 32 で固定し，係数行列 A のサイズを変化した時の実行時間を測定した．その結果を表 4 に示す．表中の解計算は A^{-1} が求まった後に $A^{-1}b$ を計算する時間である．なお，解計算部分でも BLAS ライブラリを用いた．

表 4: 逐次処理時の実行時間 (s=32)

n	実行時間 [ms]		
	逆行列計算	解計算	合計
780	29.0	1.24	30.2
1089	56.0	2.2	58.2
2427	274.3	10.2	284.5
3759	653.0	23.9	676.9

表 4 で n の値が 780 の場合と,2427 の場合とを比較すると,n が約 3 倍の大きさになったのに対し，計算時間は約 9 倍になっている．同様に,n の値が 1089 の場合と 3759 の場合とを比較すると,n が 4 倍弱の大きさになったのに対し，計算時間は 12 倍程度となった．計算時間は n^3 ではなく，ほぼ n^2 に比例していることがみてとれる．また,2.2.1 節にて示した各計算ステップごとに計算に要した時間を表 5 に示す．ただし，ステップ 4 と 5 はまとめて測定を行った．

$A^{-1}\times\Delta A_c$(ステップ 1),$E_c\times A^{-1}$(ステップ 3), $A^{-1}\Delta A_c\times(I + E_c A^{-1}\Delta A_c)^{-1}E_c A^{-1}$ (ステップ 7) において，おおまかではあるが，計算時間が $O(n^2)$ にしたがっていることが確認できる．

次に，逆行列が既知の近似行列と，求めたい行列との近似度が実行時間に与える影響を評価するため問題サイズ n を 2427 に固定し，s の値を変化させた場合の実行時間を測定した．その結果を表 6 に示す．

s の値が 2 倍になると処理時間も 2 倍弱になっていることが確認できる．SMW 公式の計算量は $O(s^3 + sn^2)$ であるが,n>>s であるため，s^3 の部分が計算時間に及ぼす影響が小さいということも確認ができる．同じように，各ステップごとの実行時間を表 7 に示す．

表 7 から，$A^{-1} \times \Delta A_c$(ステップ 1),$E_c \times A^{-1}$(ステップ 3), $A^{-1}\Delta A_c \times (I + E_c A^{-1}\Delta A_c)^{-1}E_c A^{-1}$ (ステップ 4) において,s の値にほぼ比例して処理時間が増大していくことが確認できる．また,$(I + E_c A^{-1}\Delta A_c)^{-1}$(ステップ 4+5) においては，s の 3 乗に比例して処理時間が増大していくこと

表 5: 逐次処理時の逆行列計算時間の内訳 (s=32)

ステップ	実行時間 [ms]			
	係数行列の次数 n			
	780	1089	2427	3759
1	8.4	17.0	87.6	214.3
2	0.3	0.5	1.2	1.8
3	8.5	16.2	78.9	184.7
4+5	0.1	0.1	0.1	0.1
6	0.3	0.5	1.1	1.6
7	7.0	13.9	67.8	160.7
8	4.3	7.9	37.8	89.8
合計	29.0	56.0	274.3	653.0

表 6: s の値と計算時間の関係 (n=2427)

s の値	実行時間 [ms]		
	逆行列計算	解計算	合計
16	186.5	10.1	196.6
32	274.3	10.2	284.5
64	472.4	10.2	482.6
128	867.1	10.1	877.2

が確認できる．これらのことから,s が n の 5%程度であれば，SMW 公式は掃き出し法 ($O(n^3)$) や，ガウスの消去法 ($(n^3/3+O(n^2))$ よりも十分高速な解計算が行えると予測できる．

3.2 並列化効果の評価実験

前節のプログラムを並列化することで, 計算時間の改善がどの程度行われるのかを調査するための実験を行う．本実験では, s の値は全て 32 で固定した．実行スレッド数を 1 から 4 まで増やしつつ実験を行った結果を表 8 と図 2 に示す．係数行列の次数が小さいと, 差が分かりづらいため, 表には n が 3759 の場合のデータを記載した．また,n が 3759 の場合の実行ステップごとに要した時間を表 9 に示す．

表 8 から, 実行スレッド数を増やすことによって逆行列導出にかかる時間に関して 2.4 倍の高速化が確認できた．図 2 から，実験を行ったその範囲において 2.33〜2.44 倍の高速化がされていることが確認できた．並列処理を行うことによる処理時間全体の高速化については有効性が確認できたが, 解計算部 $A'^{-1}b$ の計算速度と, $A^{-1}-A^{-1}\Delta A_c(I+E_cA^{-1}\Delta A_c)^{-1}E_cA^{-1}$ (ステップ 8) の計算速度において, 2 スレッドで実行した場合のほうが 3,4 スレッドで実行した場合よりも速くなっている．ステップ 8 においては,n^2 のサイズの行列データ 2 つを処理する際に, 並列化することによって主記憶アクセスの競合が起こっている可能性が考えられる．解計算部分の速度低下については，ステップ 8 実行直後のキャッシュの状態の影響によるものではないかと推測する．

表 7: 逐次実行時の逆行列計算時間の内訳 (n=2427)

ステップ	実行時間 [ms]			
	s の値			
	16	32	64	128
1	57.0	87.6	153.2	280.0
2	0.4	1.2	4.0	14.5
3	46.1	78.9	140.0	265.7
4+5	0.02	0.1	0.8	5.4
6	0.3	1.1	3.6	13.6
7	38.5	67.8	126.4	244.9
8	44.2	37.8	44.3	43.0
合計	186.5	274.4	472.4	867.1

表 8: 複数スレッドで実行した場合の逆行列計算時間 (n=3759,s=32)

スレッド数	実行時間 [ms]		
	逆行列計算	解計算	合計
1	653.0	23.9	676.9
2	386.1	17.7	403.8
3	310.5	19.5	330.1
4	271.6	19.6	291.2

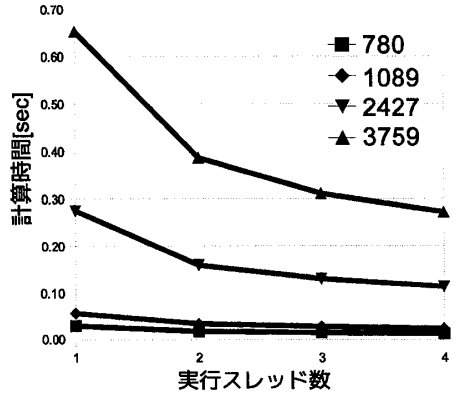

図 2: 実行スレッド数と逆行列計算時間の関係 (s=32)

3.3 SMW 公式における疎行列性の利用

係数行列 A が密行列か疎行列かによらず, A の時間変化の度合いが微小であれば,ΔA が疎行列となり，ΔA_c,E_c は疎行列となる．そこで, E_c および ΔA を圧縮形式で保存するとともに，零要素に対する演算を省略することによる高速化効果を評価する．

具体的には,SMW 公式の計算過程において, 常に行列積の左側に現れる行列 E_c に対して CRS 形式 (Compressed Row Storage format) での圧縮を，常に行列積の右側に現れる行列 ΔA に対して CCS 形式 (Compressed Column Storage format) で圧縮を行った．

計算途中で現れる他の行列に関しても，疎行列となる可能性が高いが，計算途中での fill-in によるオーバヘッドの増加などの影響を鑑み圧縮形式とはしなかった．

表 9: 複数スレッドで実行した場合の
実行時間内訳 (n=3759,s=32)

ステップ	実行時間 [ms]			
	スレッド数			
	1	2	3	4
1	214.3	121.1	88.8	76.0
2	1.8	1.2	1.1	1.0
3	184.7	95.3	74.5	57.5
4+5	0.1	0.1	0.1	0.1
6	1.6	1.0	0.6	0.5
7	160.7	82.7	59.1	50.1
8	89.8	85.0	86.3	86.5
合計	653.0	386.1	310.5	271.6

その結果，節 2.2.1 に示した計算の手順に，行列 ΔA_c,E_c を疎行列形式で圧縮するというステップが追加され，以下のような手順となる．

1. ΔA_c を CCS 形式で圧縮
2. $A^{-1} \times \Delta A_c$
3. E_c を CRS 形式で圧縮
4. $E_c \times A^{-1}\Delta A_c$
5. $E_c \times A^{-1}$
6. $I + E_c A^{-1}\Delta A_c$
7. $(I + E_c A^{-1}\Delta A_c)^{-1}$ を直接法で計算
8. $(I + E_c A^{-1}\Delta A_c)^{-1} \times E_c A^{-1}$
9. $A^{-1}\Delta A_c \times (I + E_c A^{-1}\Delta A_c)^{-1} E_c A^{-1}$
10. $A^{-1} - A^{-1}\Delta A_c (I + E_c A^{-1}\Delta A_c)^{-1} E_c A^{-1}$

この手順にしたがい，s=32 として n の値を変化させ，疎行列圧縮の高速化の度合いを検証する．まず最初に，純粋に疎行列圧縮の効果のみを検証するため，逐次実行環境において実験を行った．結果を表 10 に示す．

表 10: 疎行列圧縮時の実行時間 (n=3759,s=32)

n	実行時間 [ms]		
	逆行列計算	解計算	合計
780	16.7	1.2	17.8
1089	29.5	2.2	31.7
2427	121.6	9.7	131.4
3759	282.1	24.6	306.7

更に，各ステップごとの処理時間を表 11 に示す．ただし，ステップ 6 と 7 はまとめて測定を行った．

疎行列圧縮を行うことによって，非圧縮状態 (表 4) と比較して,1.76〜2.31 倍の高速化がされていることを確認できた．疎行列圧縮の効果が最もよくあらわれるステップ 2 と 5 においては，最高で 100 倍以上の速度改善が得られた．

疎行列圧縮による高速化が確認できたところで，疎行列圧縮をしつつ並列処理することによる高速化について検証を行った．係数行列のデータはともに n の値が 3759 のものを用い，s の値は 32 で固定した．圧縮をしつつ実行スレッド数を変更した結果を表 12 に示す．

表 11: 疎行列圧縮時の実行時間内訳 (s=32)

ステップ	実行時間 [ms]			
	係数行列の次数 n			
	780	1089	2427	3759
1	0.1	0.2	0.4	0.6
2	2.2	3.0	6.5	10.1
3	0.1	0.1	0.2	0.3
4	0.7	0.7	0.7	0.7
5	0.3	0.5	0.9	1.5
6+7	0.1	0.1	0.1	0.1
8	0.5	0.7	1.2	1.8
9	7.2	14.0	68.5	162.2
10	5.5	10.3	43.1	104.8
合計	16.7	29.5	121.6	282.1

表 12: 疎行列圧縮時の並列処理効果 (n=3759,s=32)

スレッド数	実行時間 [ms]		
	逆行列計算	解計算	合計
1	282.1	24.6	306.7
2	178.4	17.6	196.0
3	152.0	19.6	171.6
4	142.0	19.5	161.5

疎行列圧縮を行うことによって，非圧縮状態 (表 4) と比較して,1.76〜2.31 倍の高速化がされていることを確認できた．また，係数行列 A のサイズが大きければ大きいほど疎行列化の効果が大きいことが確認できた．これは,s を固定して計測を行ったために，n が大きくなればなるほど n の値に対する s の値の割合が小さくなるためだと考えられる．また，解計算部分 $A^{-1}b$ について，今回は b を密ベクトルとして扱ったため，計算量は n^2 であるが，実際には b は疎ベクトルであることが多く，適切に疎行列圧縮を行えば更に処理の高速化が見込めるのではないかと考えられる．

また，疎行列圧縮，並列処理の両方を用いて処理を行うことで，表 4 と比べて 4.19 倍の高速化が得られた．並列化のみで 2.4 倍，疎行列圧縮のみで 2.3 倍の高速化が得られたため，

表 13: 実行スレッドが複数の場合の
各ステップにおける実行時間 (s=32)[ms]

ステップ	スレッド数			
	1	2	3	4
1	0.6	0.6	0.5	0.6
2	10.1	5.2	3.7	3.4
3	0.3	0.3	0.3	0.3
4	0.7	0.7	1.5	0.8
5	1.5	1.5	1.4	1.3
6+7	0.1	0.1	0.1	0.1
8	1.8	1.2	1.0	0.9
9	162.2	82.1	58.7	50.3
10	104.8	86.7	84.8	84.3
合計	282.1	178.4	152.0	142.0

5 倍程度の高速化を予想していたが, そこまで効率はあがらなかった.

3.4 ICCG 法との比較

ここまで,SMW 公式についての高速化について述べてきたわけであるが, ここで一般的な連立一次方程式の解法の一つである ICCG 法 (不完全コレスキー分解付共約勾配法) と SMW 公式を用いた解法との簡単な比較を行う.

実験には表 3 の n の値が 2427 と 3759 のものを用い, s の値は 32 のものを用いた. 表 14 の前処理とは,SMW 公式では逆行列を求めるまで, ICCG 法では解計算の反復回数を減らすための前処理行列の導出にかかる時間を表している. 解計算とは,SMW 公式では $A^{-1}b$ の計算の部分で,ICCG 法では反復計算の部分を示している.

表 14: それぞれの手法による実行時間 [ms]

n	並列度	手法	前処理	解計算	合計
2427	1	SMW 公式	121.7	9.7	131.4
		ICCG 法	103.6	87.6	191.2
	4	SMW 公式	62.3	8.6	70.9
		ICCG 法	105.6	65.3	170.9
3759	1	SMW 公式	282.0	24.6	306.7
		ICCG 法	251.02	197.49	448.51
	4	SMW 公式	142.0	19.5	161.5
		ICCG 法	250.72	145.62	396.34

前処理段階の数値に注目すると,SMW 公式は逐次処理の場合,ICCG 法とほぼ変わらない速度が出ることが確認できる. 解計算の処理については ICCG 法よりもはるかに高速で, 処理全体の時間を比べても高速であることが確認できる. 4 スレッドで実行した場合には, 約 2 倍程度の高速化に成功している. 今回,ICCG 法における前処理の工程は並列処理を行っていないため, 実際にはより高速な数値が得られることが考えられる. しかし,ICCG 法における前処理行列の計算が, 逐次性が高く, あまり並列化の効果が上がらないことを踏まえると, 逆行列を求めての求解が反復法による求解とほぼ同速, もしくはそれ以上の速度で行えることを示していると考えられる.

4 考察

対象とする行列の疎行列性, 並列性を利用して SMW 公式を用いた逆行列計算の高速化を行った結果, 4 倍以上の高速化が達成できた. しかしながら, 今回の実験結果では 3.3 節のステップ 9 やステップ 10 等の本来並列化効果が高い部分での速度向上が得られていない. 実験環境のキャッシュやメモリ等のハードウェア環境に依存する現象とも考えられるため, ハードウェアパラメータを考慮した最適化を行うことで更なる高速化が期待できる. さらに, これらのステップで得られる計算結果は対称行列であることが既知であるため, その性質を利用することで最大で 2 倍の高速化の可能性も残っている. これらの効果を考慮すると, SMW 公式を利用した方式が反復法に対して更なる優位性が得られることが期待できる.

一方で, シミュレーションの進行により s のサイズが大きくなると s^3 の効果が表れてくるが, s 自体の疎行列性, 対称性を利用した更なる最適化の可能性が考えられる.

さらに, n や s がともに大きくなった場合には, 不完全コレスキー分解の発想と同様に, SMW 公式を用いて一部不完全な近似逆行列を作り, CG 法の前処理行列に利用する等の組み合わせの可能性についても考えることが可能である.

5 まとめ

本稿ではインタラクティブシミュレーションにおいて頻出する $Ax = b$ という連立一次方程式について, 時間経過によって A が微小な変化を行い A' となる, という前提のもと SMW 公式を用いて逆行列計算の高速化を行った. さらに,SMW 公式を用いる際に並列処理, 疎行列性を利用するためのデータ圧縮といった高速化を施して, それぞれの手法に対する評価を行った. 実装したプログラムでは並列処理を行うことで 2.33〜2.44 倍の高速化が, 疎行列性を利用することで 1.76〜2.31 倍の高速化が, 両方を組み合わせて用いることで 4.19 倍の高速化がそれぞれ得られた.

また, 反復解法のひとつの例として,ICCG 法との処理速度の比較を行った. 係数行列 A が変化しなかった場合に, 高速に求解が行える点を踏まえて, 逆行列を用いての求解も十分有効な手法と成りえると考えられる.

今後,s の値が大きくなった場合にも処理を高速に行うためのアルゴリズムの改善や, 並列処理を行っているにも関わらず速度が向上しない部分の検証を行っていくことが重要である.

謝辞

本研究の一部は, 科学研究費補助金（基盤研究 (S)16100001, 基盤研究 (C)22500044）の補助を得て実施したものである.

本研究を進める上で, 日頃から有益な議論, 助言をいただいた京都大学 富田眞治教授ならびに福井大学 森・福間研究室の諸氏に感謝いたします.

本研究の実施にあたり, 実シミュレーションで用いられる剛性マトリックスのデータを提供していただいた京都大学附属病院医療情報部の皆様に感謝いたします.

参考文献

[1] G.H. Golub and C.F.Van Loan, “Matrix Computations,”

[2] 二宮市三, 吉田年雄, 長谷川武光, 秦野やすよ, 杉浦　洋, 櫻井鉄也：数値計算のつぼ (2004)

[3] 依藤 逸:手術シミュレータにおける SMW 公式を用いた疎行列計算の高速化. 京都大学卒業論文特別報告書 (2006).

[4] インテル©マス・カーネル・ライブラリー　リファレンス・マニュアル (Version 9.0)