特別研究報告書

共役勾配法による手術シミュレータ計算の高速化

指導教員　　富田 眞治 教授

京都大学工学部情報学科

吉田 智一

平成18年2月10日

共役勾配法による手術シミュレータ計算の高速化

吉田 智一

内容梗概

近年の外科分野では，小切開創による低侵襲手術やロボットを用いたマイクロ手術など，手術の高度化が注目される．一方，手術の安全な執刀には，一般的に数百例以上の開腹手術を経験しなければならい．しかし，短期間に経験できる手術数は限られており，技術の高度化と医師の技術向上が釣り合っていないのが現状である．

そこで，実際の手術に代わって，手術シミュレータを用いた医療技術の習得訓練を行うことが望まれている．実際の手術を疑似体験できるようなインタラクティブで精巧なシミュレーションが行えれば，医師の技術が効率的に向上すると見込まれる．現在の手術シミュレータにおける計算は，触診等の小変形の挙動表現に対してはインタラクティブに行える．しかし，切断等を伴う変形では挙動表現のインタラクティブ性は低い．

本稿では，反復法である共役勾配法を用いて手術シミュレータ計算を高速に行い，切断を伴う変形に対してもインタラクティブな計算を行うことを目指す．

シミュレータで扱う臓器等をモデル化し，応力による変位を表現するためには，有限要素モデルを用いるのが一般的である．

従来の手術シミュレータ向けの計算ライブラリでは，有限要素モデルにおける剛性マトリクスを用いた応力-変位計算に対し，変形を小変形に限定し，線形性と固定境界条件が変わらないという制約を与えて計算を行う手法が用いられてきた．この制約のもとでは，全体剛性マトリクスは変形の状態に依存しない．よって，あらかじめ全体剛性マトリクスの逆行列を 1 回求めておき，それを用いて直接法で変形毎の応力-変位計算を行う．すると，逆行列を求める必要はなく，計算量は比較的少ないため，高速に計算できる．

しかしながら，この手法は切断を伴う変形に対する応力-変位計算には利用できない．これは，全体剛性マトリクスは変形の状態に依存し，あらかじめ求めておいた逆行列を用いた直接法計算は行えないためである．また，切断を伴う変形が発生した時点で，逆行列を計算し，それを用いて直接法で応力-変位計算を行うと，変形毎に逆行列を求める計算を要する．しかし，逆行列の導出は $O(n^3)$ の計算量を要し，シミュレーションのインタラクティブ性を維持するような高

速な計算は行えない．

そこで，切断を伴う変形に対してもシミュレーションのインタラクティブ性を維持すべく，反復法を用いて応力-変位計算を高速に行う．ここでは，あらかじめ用意した剛性マトリクスの逆行列を用いた近似計算をするのではなく，変化毎のモデルに対応する剛性マトリクスを用いて，反復法により応力-変位計算を行う．シミュレータのリフレッシュレートを満たすよう高速に行えれば，切断を伴う変形が起きた場合も，インタラクティブ性を失わない計算ができる．

反復法アルゴリズムとして，疎行列を扱うのに適し，収束が速く，並列化による高速化が見込める，共役勾配法を用いる．

共役勾配法の実装では，不完全コレスキー分解による前処理，数値計算ライブラリの使用，行列データ構造のリスト化，並列化などを用いて高速化する．

今回扱う共役勾配法で行う計算は，大動脈の触診をシミュレートした際の応力-変位計算である．大動脈モデルは有限要素モデルで，ノード数 197，799，1661 の 3 モデルである．また，比較のために，LU 分解を用いた直接法による計算も行った．

実行結果は，ノード数 197 のモデルでは，スカイライン法を用いた LU 分解による直接法の方が高速で，0.016208[s] であった．また，ノード数 799，1661 の 2 モデルでは，行列データ構造のリスト化および，並列化を施した共役勾配法を用いることにより，ノード数 799 のモデルではクラスタ 4 台で 0.472755[s] であった．また，ノード数 1661 のモデルではクラスタ 8 台で 1.224938[s] であった．

よって，今回の研究で，切断を伴う変形時に広田手法を用いることが出来なくなった場合の応力-変位計算は，’aorta197’ 程度のノード数の小さいモデルでは直接法の方が有効で，’aorta799’ 以上のノード数の大きいモデルでは反復法の共役勾配法で計算することが有効であることが分かった．

The speedup of a Surgical Simulator by Conjugate Gradient Method

Tomokazu YOSHIDA

Abstract

Recently in the field of surgery, the surgery with a finesse incision and a micro-surgery using a robot are the center of attention. They need high technologies. On the oter side, a doctor must operate for more than 100 patients for safe surgery. Actually a doctor can operate for few patients, so they cannot improve thieir surgical skill easily. It is wanted that doctors improve their surgical skills by using a surgical simulator. If doctors practice surgery as if they really operate for patients by using surgical simulator, their surgical skills will improve effectively.

Now the surgical simulator can perform a calculation for behavior expression of small changes of an object interactively, but cannot perform a calculation for behavior expression of changes with cutting of an object interactively.

In this papaer, a calculation for behavior expression is performed by Conjugate Gradient method fast to perform a calculation for behavior expression of changes with cutting of an object interactively.

It is common that an organ substitute a finite element model to express displacement by stress. A conventional Real Time Physics-Based Simulation Library for the surgical simulator has used Hirota method to perform a calculation with a Stiffness Matrix of finite element model. Hirota method limits transformation to small one, and perform a calculation under the restriction that liner property and fixed boundary condition do not change. Under this restriction, a Stiffness Matrix is not changed by transformation. Therefore once the inverse matrix of a Stiffness Matrix is found, a calculation for expression of displacement by stress is performed by direct method using the inverse Matrix. Then it is not needed to find the inverse Matrix , so the calculation is performed fast with small calculational amount. In this way, under the restriction, a calculation for expression of displacement by stress is performed fast by Hirota method.

A calculation for behavior expression of changes with cutting cannot be per-

formed by Hirota method. Changes with cutting make the restriction meaningless. When the Stiffness Matrix is changed the calculation cannot be performed by direct method using the inverse Matrix found previously.

If the calculation for changes withcutting is performed by direct method using the inverse Matrix, it is needed that the inverse Matrix is found every transformation. To find the inverse Matrix, it is needed large amount of calculation about $O(n^3)$, so a fast calculation maintaining an interactivity of the simulation cannot be performed.

To maintain the interactivity, the calculation is performed fast by a repetitional method without the inverse Matrix found previously. If the calculation is performed by the repetition method very fast, the interactivity is maintained. In many repetition methods, I chose Conjugate Gradient method. I implemented Conjugate Gradient method with preprocessing by the incomplete Cholesky resolution, using of a numerical computation library, using a list data structure of Matrix data, and using parallel computing , to perform the calculation more fast.

The problem to solve by Conjugate Gradient method is a calculation for expression of displacement by stress of the simulation of palpation of aorta. There are three finite element models of aorta. The first model has 197 nodes, second has 799 nodes, and third has 1661 nodes. In addition, for comparison I performed the same calculation by the direct method with LU resolution.

As a result, in the model of 197 nodes, the direct method with LU resolution was higher-speed than with Conjugate Gradient method and was carried out in 0.016208[s]. In the model of 799 nodes and the model of 1661 nodes, the Conjugate Gradient method using a list data structure of Matrix data and using parallel computing was carried out in 0.472755[s] and 1.224938[s].

Therfore a calculation for expression of displacement by stress of changes with cutting without Hirota method is effectively performed by direct method in a particular model with 197 nodes, and is effectively performed by Conjugate Gradient method of repetition method in a coarse model with 799 and 1661 nodes.

共役勾配法による手術シミュレータ計算の高速化

目次

第1章　はじめに

近年の医療技術の進化と発展とともに，その難度も上昇し，医師には高度な技能が求められている． 特に外科分野では新しい手術手技に対して，外科医のスキルを如何に効率的に向上させ，一定の水準に到達させるかが大きな課題となっている．この課題に対し，人体組織の力学や生理をモデル化した仮想物体を用いて手術シミュレーションを実現し，スキル向上に役立てるアプローチが考えられる．我々は仮想的な空間において手術シミュレーションを容易に行える，インタラクティブな手術シミュレータの完成を，京都大学医学部附属病院医療情報部と共同で目指している [1][2].

臓器の触診等で加わる応力と，その応力による変位の表現には，有限要素モデルを用いるのが一般的である．中尾らが開発した手術シミュレータ向け実時間力学計算手法のライブラリ [8] における，有限要素モデルに基く反力，変形計算手法では，広田らが提案した近似解法 [5](以下，広田手法と呼ぶ) が用いられている．有限要素モデルにおいて，線形性と固定境界条件が変わらないという制約のもとで，全体剛性マトリクスは変形の状態に依存しない．このことから，逆行列をあらかじめ求めておくことで，直接法で応力-変位計算を容易に，高速に行うことを可能にする手法である．線形性と固定境界条件が変わらないという制約は，臓器の触診といった小変形では適用可能である．

しかし，切開手術では，シミュレーション対象物体の幾何学的な構造 (トポロジ) 自体が変化するため，切開を伴う変形が発生した場合には，剛性マトリクス自体が変化してしまう．従って，広田手法が仮定している線形性が成立しなくなる．そこで，並列化効率が悪く，$O(n^3)$ の計算時間が必要な，剛性マトリクスの逆行列が介在した近似計算ではなく，本来必要とされている応力-変位方程式の解そのものを高速に求解することで，実時間応答性を改善しようというのが本提案の目的である．

応力-変位方程式の直接求解においては，剛性マトリクスが係数が定数である対称疎行列であることを利用して，並列化効率が良く，かつ反復計算 1 回分の計算時間が $O(n^2)$ である共役勾配法を用いた反復解法を利用する．

以下，第 2 章で背景，第 3 章で高速化を視野に入れた実装について述べた後，第 4 章で実行結果の評価，第 5 章で考察，第 6 章でまとめを行う．

第2章　背景

現在求められている手術手技シミュレーションの課題としては，臓器の切開等のリアルな挙動表現がある．操作による挙動をリアルタイムに表現するために，従来は小変形のみを仮定してシミュレートされてきた．しかし，これではシミュレートできる手技が限定され，切断等の切断を伴う変形を引き起こす手技の挙動が実時間で表現できない．手術においては患部の治療のために切断手技は不可欠である．しかし，切断手技はミスが患者の生命に関わる重要な手技である．よって，切断手技の挙動をリアルタイムで忠実に表現できることは医師の切断技術向上につながり，結果としてスムーズで安全な手術を行えることにもつながる．つまり，手術シミュレータ完成には欠かせない要素と言える．

リアルな挙動を表現するために，臓器のモデル化としては有限要素法を用いる [8]．有限要素法における応力-変位計算について，従来の計算手法では小変形のみを仮定している．仮定においては剛性マトリクスは変化せず，あらかじめ逆行列を求めておけば，応力から変位の計算は直接法により高速に行えるとするものであった．しかし，仮定に該当しない切断を伴う変形では剛性マトリクスは変化し，直接法で応力-変位計算を行うには，新たな逆行列の導出が必要となる．しかし，これには多大な計算時間を要し，リアルタイムな表現と言う観点からは妥当でない．そこで，今回提案する応力- 変位計算手法では小変形に限定せず，操作による変形毎に導出される剛性マトリクスを用いて反復法で高速に計算を行い，切断等の切断を伴う変形の挙動もリアルタイムに表現できることを目指す．扱う反復法としては，剛性マトリクスの疎行列性と並列化に適した，共役勾配法を用いる．

本章では有限要素モデルの概要，手術シミュレータにおける従来の応力-変位計算手法，今回提案する手法，共役勾配法について述べる．

2.1　有限要素モデル

2.1.1　概要

有限要素モデルとは，変形に対し無限の自由度を持つ物体を，有限の自由度を持つ要素（有限要素）の集合体と近似し，この集合体に加わる力とその変位の関係に対して成立する方程式（連立一次方程式）を解くモデルである．全自由度に加わる力のベクトルを$\boldsymbol{F}$，全自由度の変位ベクトルを$\boldsymbol{u}$とする．係数行列

を K とすると，この連立一次方程式は，

$$\boldsymbol{F} = K\boldsymbol{u} \tag{1}$$

と表される．ここで K は，物体に加わる力とその変位の関係を表すマトリクスとして，全体剛性マトリクスと呼ばれる．全体剛性マトリクスは，要素毎の要素内全自由度の力-変位関係を表現する要素剛性マトリクスを，要素の集合全体について足し合わせることで得られ，モデルの持つすべての自由度に対して変位が与えられた際に，各自由度に発生する力を与えるものである．

ここで線形なモデルに限って考えると，全体剛性マトリクスは定数行列となる．すると，全体剛性マトリクス及び全自由度の力-変位関係は次のように与えられる．

$$\begin{bmatrix} \boldsymbol{f}_i \\ \boldsymbol{f}_o \end{bmatrix} = \begin{bmatrix} K_{ii} & K_{io} \\ K_{oi} & K_{oo} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_i \\ \boldsymbol{u}_o \end{bmatrix} \tag{2}$$

ここで，$\boldsymbol{u}_i$ および $\boldsymbol{f}_i$ は力が与えられた接触点のノードにおける変位及び力，$\boldsymbol{u}_o$ および $\boldsymbol{f}_o$ は接触点以外の点のノードにおける変位及び力である．手術シミュレータでは要素（メッシュ）として四面体を用い，各節点（ノード）における自由度は x，y，z の 3 つである．図 1 に物体のメッシュ分割のイメージを示す．

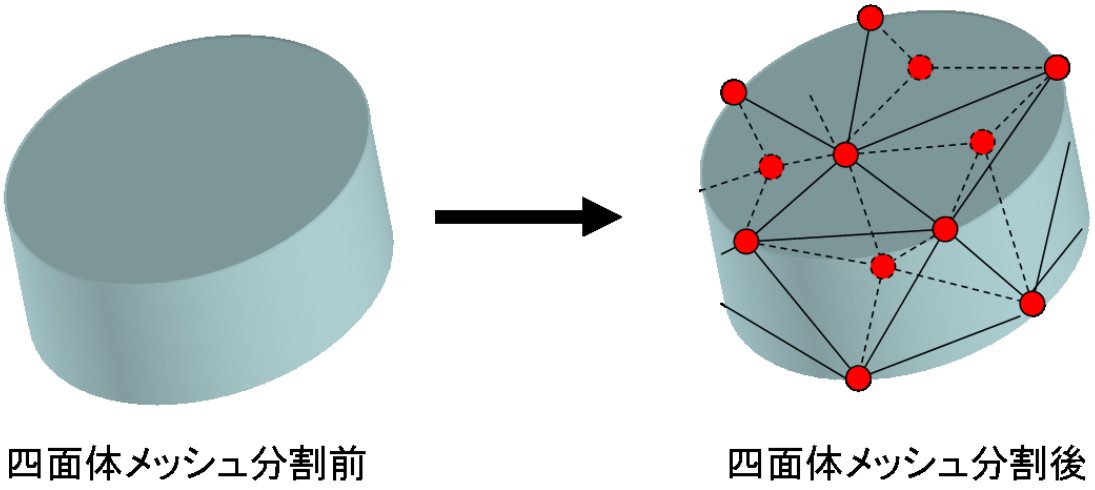

図 1: 要素 (メッシュ) 分割

2.2　これまでの医療シミュレータ計算

起こる操作を小変形に限定したシミュレーションにおいて，接触点における力が分かっていて，接触点以外では力が加わらないと仮定する．有限要素モデルにおける各自由度における変位の導出は，以下のようになる．まず，式（2）より

$$\boldsymbol{f}_i = K_{ii}\boldsymbol{u}_i + K_{io}\boldsymbol{u}_o \tag{3}$$

$$\boldsymbol{f}_o = K_{oi}\boldsymbol{u}_i + K_{oo}\boldsymbol{u}_o \tag{4}$$

である．式（4）にて，仮定より $\boldsymbol{f}_o = \boldsymbol{0}$ であるから，

$$\boldsymbol{u}_o = -K_{oo}^{-1}K_{oi}\boldsymbol{u}_i \tag{5}$$

式（3）へ代入して，

$$\boldsymbol{f}_i = K_{ii}\boldsymbol{u}_i - K_{io}K_{oo}^{-1}K_{oi}\boldsymbol{u}_i \tag{6}$$

となる．

式 (5)，(6) で逆行列を求める計算は，マトリクスが数千次となるため，多大な計算時間を要する．一方，手術シミュレータの変形計算において，実時間性を維持するためには 30Hz のリフレッシュレートを要する [8]．このリフレッシュレート実現のために，中尾ら [8] は広田手法を用いて式 (5)，(6) の逆行列を扱ってきた．触診等の小変形においては，変形前後で剛性マトリクスの変化は非常に小さく，近似により剛性マトリクスは同一のものと見なすことができる．よって，一連の操作によって小変形しか起こらないとした仮定では，式 (5)，(6) の逆行列を一度求めておけば，各時点での応力-変位計算はその逆行列を用いることができ，直接法により容易に計算できる．広田手法はこのように仮定を設けて応力-変位計算手法を簡単化して，高速に行う手法である．しかし，切断を伴う変形が起こると，変形の前後で剛性マトリクスが大幅に変化し，変形前の逆行列を利用できない．その結果，変化毎に新たな逆行列を導出する必要が生じ，その導出に多大な時間を要する．

2.3 提案する医療シミュレータ計算

2.3.1 変形前の状態に依存しない計算

広田手法では，操作による変形を小変形を仮定し，剛性マトリクスの変化はないと近似した．そして，あらかじめ求めておいた逆行列を用いた直接法により，応力-変位計算を行ってきた．今回提案する手法では，変形を小変形に限定せず，切断を伴う変形への対応も想定する．そのために，変形毎に導出される剛性マトリクスから，応力-変位計算を，2.2 節で示したリフレッシュレートを満たすよう高速に行うことを考える．リフレッシュレートを満たすよう高速に計算できれば，広田手法の仮定以外の切断操作の挙動も，リアルタイムに表現できると考えられる．

剛性マトリクスから応力-変位計算を行う，つまり，剛性マトリクスを係数行列として持つ連立一次方程式を解く方法としては，直接法と反復法が考えられる．直接法は，逆行列を用いる行列演算に帰着する．広田手法の仮定のもとでは，あらかじめ導出した逆行列を用いれば，逆行列の導出は不必要であった．しかし，大変形のもとでは剛性マトリクスの変化は大きく，新たな剛性マトリクスに対する逆行列の導出が必要となる．そのため，$O(n^3)$ の計算時間を要する．一方，反復法は基本的には $O(n^2)$ の計算時間である．よって，剛性マトリクスが数千次にも及ぶ状況で，切断を伴う変形をリアルタイムに表現するために，変形前の状態に依存しない計算法として反復法を用いるのが妥当である．

ここで，有限要素モデルにおいて，ある自由度の離れた自由度に対する影響は小さいという性質がある．この性質により，剛性マトリクスは疎行列となる [6]．

また，2.2 節で示したリフレッシュレートを満たす応力-変位計算を行うために，反復法では高速な収束が見込める方法が望ましい．

さらに，計算の高速化の一つの方法としては，並列化が考えられる．

以上より，係数行列が疎行列の連立一次方程式に適し，最大次数 n 回のループで計算が終了し，並列化の容易な共役勾配法を，反復法のアルゴリズムとして適用する．

2.4 共役勾配法 (CG 法)

有限要素モデルにおいて成立する連立一次方程式の解法として，大きく分けて直接法と反復法の二つがある．しかし，有限要素モデルにおける連立一次方程式では，数千次のものを扱う場合も多く，計算量が方程式の元 n の 3 乗に比例する直接法の使用は現実的ではない．そこで，解法として直接法より格段に計算の速い反復法を用いることとする．現在，反復法の中でよく用いられるものには共役勾配法（CG 法：Conjugate Gradient）がある．そこで本節では，共役勾配法をもとにした連立 1 次方程式の反復解法について説明する．いま，与えられた連立方程式を

$$A\boldsymbol{x} = \boldsymbol{b} \tag{7}$$

とする．但し，A は係数行列，$\boldsymbol{x}$ は未知量ベクトル，$\boldsymbol{b}$ は荷重ベクトルである．共役勾配法の基本原理は最適化法に由来している．共役勾配法では評価関数を次のように設定し，その最小化を考える．但し，A を n 次対称正定値行列とする．

$$F(\boldsymbol{x}) = (\boldsymbol{x}, A\boldsymbol{x}) - 2(\boldsymbol{x}, \boldsymbol{b}) \tag{8}$$

ここで，　$(\boldsymbol{x},\boldsymbol{b})$ は $\boldsymbol{x}$ と $\boldsymbol{b}$ の内積を表すものとする．また A として正定値を仮定していることから，この $F(\boldsymbol{x})$ は最小値を持ち，A の対称性から

$$\frac{\partial F}{\partial x_k} = \sum_{j=1}^{n}(a_{kj} + a_{jk})x_j - 2b_k = 2\sum_{j=1}^{n}(a_{kj} - b_k) = 0 \quad (k = 1, 2, \cdots, n) \tag{9}$$

を満たすことになる．これは与えられた方程式 (7) そのものであり，$F(\boldsymbol{x})$ を最小にする $\boldsymbol{x}$ は与えられた連立方程式の解となる．そこで，連立 1 次方程式を解くために，適当な初期値 $\boldsymbol{x}_0$ から出発して順次修正を加え，近似ベクトル $\boldsymbol{x}_{k+1}$ を求めるアルゴリズムを以下のように構築する．

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k \tag{10}$$

ここに $\boldsymbol{p}_k$ は近似ベクトル $\boldsymbol{x}_k$ の探索方向であり，　α_k は $\boldsymbol{p}_k$ 方向への修正量である．いま，任意のベクトル $\boldsymbol{h} \neq \boldsymbol{0}$ について

$$\begin{aligned} F(\boldsymbol{x} + \boldsymbol{h}) &= (\boldsymbol{x} + \boldsymbol{h}, A(\boldsymbol{x} + \boldsymbol{h})) - 2(\boldsymbol{x} + \boldsymbol{h}, \boldsymbol{b}) \\ &= F(\boldsymbol{x}) + (\boldsymbol{h}, A\boldsymbol{h}) - 2(\boldsymbol{h}, \boldsymbol{b} - A\boldsymbol{x}) \end{aligned} \tag{11}$$

であることから，式（10）を式（8）へ代入し，$F(\boldsymbol{x})$ を最小にする α_k を決定する． $\boldsymbol{p}_k \neq \boldsymbol{0}$ のとき，

$$\begin{aligned} F(\boldsymbol{x}_{k+1}) &= (\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k, A(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k)) - 2(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k, \boldsymbol{b}) \\ &= F(\boldsymbol{x}_k) + \alpha_k^2 (\boldsymbol{p}_k, A\boldsymbol{p}_k) - 2\alpha_k (\boldsymbol{p}_k, \boldsymbol{r}_k) \end{aligned} \tag{12}$$

となる．但し， $\boldsymbol{r}_k$ は近似ベクトル $\boldsymbol{x}_k$ の残差で

$$\boldsymbol{r}_k = \boldsymbol{b} - A\boldsymbol{x}_k \tag{13}$$

である．ここで最小化の手順に従い， $\partial F/\partial \alpha_k = 0$ として α_k を求めると，

$$\alpha_k = \frac{(\boldsymbol{p}_k, \boldsymbol{r}_k)}{(\boldsymbol{p}_k, A\boldsymbol{p}_k)} \tag{14}$$

となる．

また，第 k ステップにおける探索方向 $\boldsymbol{p}_k$ は， $\boldsymbol{r}_k$ に前ステップの探索方向の修正値を加えて，

$$\boldsymbol{p}_k = \boldsymbol{r}_k + \beta_{k-1} \boldsymbol{p}_{k-1} \tag{15}$$

とおく．ここで $\boldsymbol{p}_k$ が $(\boldsymbol{p}_k, A\boldsymbol{p}_{k-1}) = 0$ を満たすものとすれば β_k が次のように決定される．

$$\beta_k = -\frac{(\boldsymbol{r}_{k+1}, A\boldsymbol{p}_k)}{(\boldsymbol{p}_k, A\boldsymbol{p}_k)} \tag{16}$$

以上の手順を実装用アルゴリズムとしてまとめると，

1. $k = 0$ とし，適当な初期値 $\boldsymbol{x}_0$ を与え以下を計算する．

$$\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0 \tag{17}$$

$$\boldsymbol{p}_0 = \boldsymbol{r}_0 \tag{18}$$

2. 以下の計算を行う．

$$\alpha_k = \frac{(\boldsymbol{p}_k, \boldsymbol{r}_k)}{(\boldsymbol{p}_k, A\boldsymbol{p}_k)} \tag{19}$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k \tag{20}$$

$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k A\boldsymbol{p}_k \tag{21}$$

3. $||\boldsymbol{r}_{k+1}||/||\boldsymbol{r}_0|| < \varepsilon$ ならば計算終了．そうでなければ，以下の計算を行う．

$$\beta_k = -\frac{(\boldsymbol{r}_{k+1}, A\boldsymbol{p}_k)}{(\boldsymbol{p}_k, A\boldsymbol{p}_k)} \tag{22}$$

$$\boldsymbol{p}_{k+1} = \boldsymbol{r}_{k+1} + \beta_k \boldsymbol{p}_k \tag{23}$$

4. $k = k + 1$ として 2. へ戻る．

となる．

以上の方法によって連立 1 次方程式の解を求める方法が，共役勾配法（CG 法）である．共役勾配法は，原理的に n 回の探索で解に到達する [7] が，丸め誤差の影響で n 回の探索では必ずしも解に到達しない場合もある．

第 3 章　共役勾配法の実装

2.4 節に示したアルゴリズムをもとに実装を行う．なお，収束判定条件は ERR を打ち切り誤差とすると，

$$\frac{\| \boldsymbol{r}_{k+1} \|}{\| \boldsymbol{r}_0 \|} < ERR \tag{24}$$

とする．高速化手法を以下に示す．

3.1　不完全コレスキー分解による前処理

共役勾配法は，係数行列の固有値に重複がある場合や，密集固有値がある場合に収束が速い．つまり，係数行列の固有値の分布に密集があるほど，少ない探索回数で真の解に到達できる．例として，係数行列が単位行列となる場合である．この場合固有値は 1 に密集しており，探索回数は 1 である．このように係数行列が単位行列に近いような連立一次方程式の場合，共役勾配法の収束を速くできることが知られている．そこで収束を速くするために，係数行列に対して何らかの前処理を施し，係数行列を単位行列に近くした上で，共役勾配法を適用する方法がある．

ここでは，共役勾配法の収束を早める前処理として一般的に利用される，不完全コレスキー分解について述べる．

3.1.1　コレスキー分解

連立一次方程式，

$$A\boldsymbol{x} = \boldsymbol{b} \tag{25}$$

について，係数行列 A を下三角行列 L と上三角行列 U の積として分解する．

$$A = LU, \quad L = (l_{ij}), \quad U = (u_{ij}) \tag{26}$$

ここで，下三角行列及び上三角行列の要素 l_{ij}，u_{ij} は次の手順で求められる．

$$l_{ij} = a_{ij} - \sum_{k=1}^{j-1} l_{ik}u_{kj}, \quad (i \geq j) \tag{27}$$

$$u_{ij} = \frac{a_{ij} - \sum_{k=1}^{i-1} l_{ik}u_{kj}}{l_{ii}}, \quad (i < j) \tag{28}$$

このように，A を L と U に分解することを LU 分解と呼ぶ．A が対称行列とした場合の分解はコレスキー分解と呼ばれ，一般に対称行列に対しては改訂コレスキー分解と呼ばれる分解を行う．

改訂コレスキー分解の時には，LU 分解の時に使用した下三角行列 L から作った対角行列 D を上三角行列 U，すなわち L^T に掛ければよい．この時の各行列の成分は，

$$d_{ii} = a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2 d_{kk}, \quad (i = 1, 2, \cdots, n) \tag{29}$$

$$l_{ij} = \frac{a_{ij} - \sum_{k=1}^{j-1} l_{ik}l_{jk}d_{ii}}{d_{jj}} \tag{30}$$

となる．

3.1.2 不完全コレスキー分解

改訂コレスキー分解をアルゴリズム通りに行うと，計算時間を要する．ここでは扱う行列の疎行列性を活かし，

$$A \cong LDL^T \tag{31}$$

のようにコレスキー分解を不完全に行うことで前処理を簡単に行う手法がある．これが不完全コレスキー分解である．不完全コレスキー分解では行列 A の成分が 0 であった部分に対応する l_{ij} では，値を強制的に 0 にして式 (30) の計算をしない [7]．

不完全コレスキー分解を実行した後，D の成分の平方根を対角成分に持つ行列を $D^{\frac{1}{2}}$ とし，

$$LDL^T = LD^{\frac{1}{2}}(LD^{\frac{1}{2}})^T, C = LD^{\frac{1}{2}} \tag{32}$$

とする．すると，式 (25) の解は

$$C^{-1}A(C^T)^{-1}x' = C^{-1}\boldsymbol{b} \quad (33)$$

$$x' = (C^T)x \quad (34)$$

を解くことに等しい．

A の改訂コレスキー分解を不完全でなく，完全に行う，つまり $A = LDL^T$ となるように行うと，$C = LD^{\frac{1}{2}}$ として，$C^{-1}A(C^T)^{-1}$ は単位行列となるが，不完全分解のため単位行列とはならない．しかし，改訂コレスキー分解に近い操作は行っているため，$C^{-1}A(C^T)^{-1}$ は単位行列に近い形をしている．よって，式 (33) の連立一次方程式に共役勾配法を適用すれば，わずかな探索回数で解を得ることが期待できる．

前処理としてこのような不完全コレスキー分解を行った後，共役勾配法を適用する方法は ICCG 法と呼ばれる．

3.2 数値計算ライブラリの使用

共役勾配法の計算においてはベクトル演算が多用される．このベクトル演算を BLAS，ATLAS を併用して高速化することを考える．

3.2.1 BLAS

BLAS とは，行列とベクトルの基本演算を行うルーチンを集めたライブラリで，

- Level 1 BLAS（ベクトル-ベクトル演算）
- Level 2 BLAS（行列-ベクトル演算）
- Level 3 BLAS（行列-行列演算）

で構成されている．共役勾配法のアルゴリズムにおいて，Level 1 BLAS のスカラーとベクトル積，ベクトルの加減算，Level 2 BLAS の行列-ベクトル積のルーチンを用いる．

3.2.2 ATLAS

使用する CPU に対して最適化された BLAS を用いれば，計算のパフォーマンスが飛躍的に向上する．ATLAS とはシステムに最適化された BLAS を自動的に生成するものである．

3.3 行列のデータ構造のリスト化

有限要素モデルにおいて，剛性マトリクスは疎行列である場合が多い．従って，ゼロ要素が多く，行列演算の際に，ゼロを含む意味のない演算が増えてしまう．ここで，図 2 のように，疎行列対応のデータ構造として，行列のゼロ以外の要素で構成されるリストを作ることを考える．リストは，

- 非ゼロ要素のインデックス
- 非ゼロ要素の値
- 次リストへのポインタ

より構成される．また，ベクトルとの演算を容易にするため，リストのインデックスの並びは行指向とする．このようなリストを用いて行列演算を行えば，演算量は小さくなり，処理時間が短くなることが予想される．実際，大動脈の有限要素モデルの剛性マトリクスデータ ('aorta197.mtx') を見てみると，剛性マトリクスの次数は 438，つまり要素数は $438^2 = 191844$ である．この中で非ゼロ要素数は 17801 であり，1/10 以下であることがわかる．つまり，リスト処理により行列演算の演算量が減り，高速化されると考えられる．

3.4 並列化

共役勾配法は多次元ベクトル演算が多く，ベクトルを分割して並列に処理すれば，処理速度の向上が見込まれる．並列化にあたっては PC クラスタを使用し，プログラミングにあたっては MPI(Message Passing Interface)[3] を使用する．また，並列化部位に関しては

- ベクトル演算全体 (行列-ベクトル積の部位を含む)
- 行列-ベクトル積の部位のみ

の 2 種類で実装し，比較した.

3.4.1 PC クラスタ

PC クラスタとは，複数の PC をネットワークで接続し，仮想的に 1 台の並列コンピュータとして利用することでパフォーマンスを向上させた PC 群を言う．PC クラスタは 2 台から 8 台の小規模な PC クラスタから，数千台規模でスーパーコンピュータの性能を発揮する PC クラスタを構成することが可能である．また近年の PC の高性能化，低価格化に伴うコストパフォーマンスの向上により PC クラスタの構成もコストパフォーマンスの向上が図られた．また，ネッ

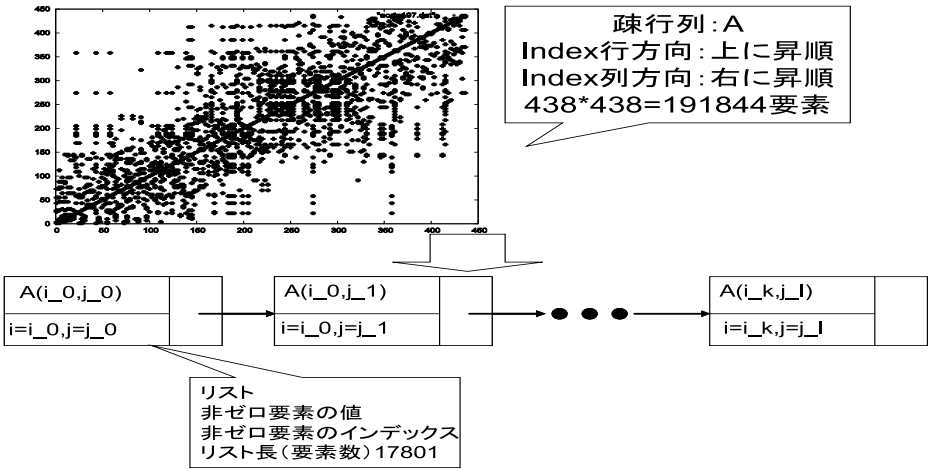

図 2: 行列の非ゼロ要素のリスト化

トワークにおいても，100M Ethernet から Gigabit Ethernet や Myrinet などの高速ネットワークの登場で，PC の高性能化に対するバランスが取れるようになり，大規模な PC クラスタで発生するネットワークのオーバヘッドを最小限に抑えることが可能となった．なお，今回使用するクラスタは，8 台の小規模なクラスタである．

3.4.2 MPI

本研究室のクラスタは分散メモリ型である．よって，プログラミングにあたってはデータを通信する必要が生じ，これを実装するためにメッセージパッシング型のプログラミングである MPI(Message Passing Interface) を使用する．MPI は，メッセージ通信のプログラムを記述するために広く使われる標準化を目指して作られた，メッセージ通信の API 仕様である．MPI は，ネットワーク上のプロセス間の効率の良い通信が可能であること，異なるプラットフォーム上

での移植のし易さの保証をすること，信頼できる通信インターフェースの提供，PVM など既存のものと同様の使いやすさとより高い融通性を実現していることを特徴とする．

3.4.3 ベクトル演算全体の並列化

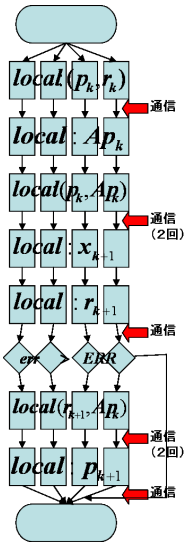

図 3: ベクトル演算全体の並列化

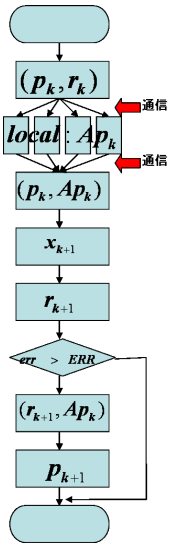

図 4: 行列-ベクトル積の部位のみの並列化

共役勾配法の特徴として，ベクトル演算の多さが挙げられる．有限要素モデルの計算では，このベクトルの次数は数千次に及ぶことも多い．図 3 のようにベクトルを分割して演算を並列に処理できれば処理速度の向上が見込まれる．共役勾配法において現れるベクトル演算としては

- スカラーとベクトルの積
- ベクトル同士の加算, 減算
- ベクトルの内積
- 行列とベクトルの積

が挙げられる．ベクトル演算を並列に行うに際して，まずベクトルを各ノードに分配する．次数 n のベクトルを p ノードに分配する場合，1 ノードあたり次数 n/p を持つ．但し，n が p の倍数でない場合はある 1 ノードのみが次数 $n/p + s$ の部分ベクトルを持つ．ここで s は n の p に対する剰余である．同様に，n 行 n 列の行列を p ノードに分配する場合，1 ノードあたり n/p 行 n 列の部分行列を

持つ．ここでも，n が p の倍数でない場合は，ある 1 ノードのみが $n/p + s$ 行 n 列の部分行列を持つ．

分割の後，ベクトル同士の加算，減算は完全に並列に行える．スカラーとベクトルの積については，スカラーを全ノードに渡す通信が必要である．ベクトルの内積については，各ノードで行われる部分ベクトル同士の内積を集めて足し込む必要があるため，通信が必要である．行列とベクトルの積は各ノードで部分行列-ベクトル積を行う際は，ベクトルは次数 n 分の情報が必要となる．そのため，あるノードが持つ部分ベクトル以外にも他ノードのが持つ部分ベクトルも必要となり，それを集める通信が必要となる．

3.4.4　行列-ベクトル積の部位のみの並列化

共役勾配法に多く存在するベクトル演算の中でも，とくに大きな割合を占めるのが行列-ベクトル積の演算である．これは，例えばベクトル同士の和算，引算，乗算といった演算は次数 n に比例した処理時間がかかる．しかし，行列-ベクトル積の演算は行列が次数 n のベクトルの n 個の集合と考えられるため，n^2 に比例した時間を要する．そこで，図 4 のように処理時間の大きいこの部分のみを並列化する．この場合，その他のベクトル演算は 1 ノードで行うこととする．そのため，行列-ベクトル積以外のベクトル演算での処理時間の向上は，ベクトル演算全体の並列化よりは見込めない．しかし，スカラーとベクトルの積，ベクトルの内積を行う際の通信は要らず，その部分で有利である．

第 4 章　評価

4.1　評価環境

評価環境を表 1 に示す．使用するクラスタは，マスターが 1 台でスレーブが 8 台である．

表 1: 評価環境

ノード数	8 台
CPU	Pentium4 3 GHz
OS	Linux 2.4.22
コンパイラ	gcc 3.3.5
コンパイラオプション	-O3 -mfpmath=sse -march=pentium4

4.2 実行結果

まず，共役勾配法で扱う問題としては，大動脈の触診に関する応力-変位計算である．大動脈を有限要素法でモデル化し，1 ノードに力を加えた場合を考える．これは，大動脈の有限要素モデルにおける剛性マトリクス (K) を係数行列とする連立一次方程式を解き，応力 ($\boldsymbol{f}$) から変位 ($\boldsymbol{u}$) を計算する問題に帰着する．

$$K\boldsymbol{u} = \boldsymbol{f} \tag{35}$$

と書くと，共役勾配法で解くにあたり，右辺ベクトル$\boldsymbol{f}$は

- 応力の掛かるノードに関する要素:応力の値
- 応力の掛かるノード以外のノードに関する要素:ゼロ (応力ノード以外に力は掛からないという仮定より)

であり，ベクトル$\boldsymbol{u}$の初期値ベクトルは$\mathbf{0}$とする．

ここで，大動脈の有限要素モデルとしては，3 モデルを考える．すなわち

1. aorta197(ノード数 197)
2. aorta799(ノード数 799)
3. aorta1661(ノード数 1661)

である．これらの有限要素モデルに対して，Matrix Builder[8] を用いて生成された剛性マトリクスを用いる．但し，Matrix Builder では直接法で変位計算が行えるように，逆行列の形で生成される．そのため，逆変換してから，共役勾配法で扱う連立一次方程式の係数行列として利用する．

各モデルに対し実装したプログラムを用いた，応力-変位計算 (連立一次方程式) の実行時間を表 2，表 3，表 4，表 5，表 6，表 7，表 8 に示す．

各表の説明を行う．まず，表 2 について，共役勾配法プログラムに対し，各高速化を行ったプログラムの実行時間の比較を示す．また，反復法である共役勾配法との比較を示すために，3.1.1 節で述べた LU 分解を用いた直接法によるプログラムの実行時間も示す．表中最左欄は用いたプログラムを示し，

CG 2.4 節で示した共役勾配法のアルゴリズム通りに実装したプログラム．

LibCG (a) に，3.2 節で示した，行列演算やベクトル演算に対し数値計算ライブラリ BLAS のルーチンを ATLAS にて高速化したものを用いたプログラム．

SparseCG (a) に，3.3 節で示した，係数行列のデータ構造を，疎行列対応のリスト構造にしたプログラム．

LU　反復法の共役勾配法との比較のための，LU 分解を用いた直接法のプログラム．LU 分解後は前進代入，後退代入により解を求める．

SparseLU　LU で，スカイライン法を用いた LU 分解を行うプログラム．スカイライン法とは，疎行列向け LU 分解高速化手法で非ゼロ要素の分解計算をスキップすることにより高速化する手法である．

とする．また，表中最上欄は使用したモデルを示す．

表 2: 各プログラムの実行時間

	aorta197	aorta791	aorta1661
CG	0.188503[s]	4.892315[s]	27.473117[s]
LibCG	0.102798[s]	2.825731[s]	16.323603[s]
SparseCG	0.061617[s]	0.508868[s]	2.069610[s]
LU	0.234710[s]	10.845323[s]	92.085955[s]
SparseLU	0.016208[s]	0.535902[s]	3.653248[s]

表 3，表 4，表 5 について，各モデルに対し，'SparseCG' に不完全コレスキー分解による前処理を施したプログラム (Cholesky(a)) の実行時間を示す．表中最上欄は，

TimeOfCG　共役勾配法部分のみの実行時間

TimeOfAll　不完全コレスキー分解による前処理の時間を含めた全体の実行時間

Loop　共役勾配法のループ回数

とする．

また，表 6，表 7，表 8 について，各モデルに対し，表 2 の'CG' を並列化したプログラムの実行時間を示す．なお，3.4.3 節で示したベクトル演算全体の並列化，3.4.4 節で示した行列-ベクトル積のみの並列化の 2 種を行う．表中最左欄は用いたプログラムを示し，

ParallelCG(a)　CG のベクトル演算全体を並列化したプログラム

ParallelCG(b)　CG の行列-ベクトル積のみを並列化したプログラム

とする．表中最上欄には並列化に用いるクラスタの台数を示す．

表 3: 不完全コレスキー分解による前処理効果 aorta197

	TimeOfCG	TimeOfAll	Loop
SparseCG	0.061617[s]	0.061617[s]	320[回]
Cholesky(a)	0.051107[s]	6.009937[s]	241[回]

表 4: 不完全コレスキー分解による前処理効果 aorta799

	TimeOfCG	TimeOfAll	Loop
SparseCG	0.508868[s]	0.508868[s]	514[回]
Cholesky(a)	0.400253[s]	702.852370[s]	399[回]

表 5: 不完全コレスキー分解による前処理効果 aorta1661

	TimeOfCG	TimeOfAll	Loop
SparseCG	2.069610[s]	2.069610[s]	700[回]
Cholesky(a)	1.549494[s]	21630.654495[s]	523[回]

また，並列化効果を見るために，2 種の並列化プログラムの各モデルに対するクラスタ台数と実行時間の関係を示したグラフを示す．'ParallelCG(a)' については図 5 に，'ParallelCG(b)' については図 6 に示す．

表 6: 2 種の並列化の実行時間 aorta197

	1 台	2 台	4 台	8 台
ParallelCG(a)	0.618354[s]	0.970383[s]	0.855004[s]	1.604196[s]
ParallelCG(b)	0.196604[s]	0.292872[s]	0.256248[s]	0.324446[s]

表 7: 2 種の並列化の実行時間 aorta799

	1 台	2 台	4 台	8 台
ParallelCG(a)	16.660599[s]	9.128374[s]	4.979070[s]	3.809046[s]
ParallelCG(b)	4.918082[s]	2.728668[s]	1.616742[s]	1.135356[s]

表 8: 2 種の並列化の実行時間 aorta1661

	1 台	2 台	4 台	8 台
ParallelCG(a)	96.494240[s]	62.579170[s]	31.983287[s]	15.060825[s]
ParallelCG(b)	27.928371[s]	14.519821[s]	7.861081[s]	4.627766[s]

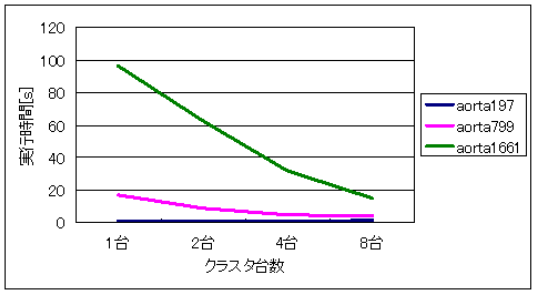

図 5: 並列化効果 ParallelCG(a)

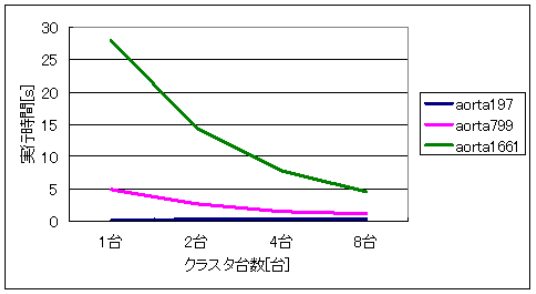

図 6: 並列化効果 ParallelCG(b)

第 5 章　考察

5.1　共役勾配法の有効性

表 2 の結果を見ると，'aorta197' については'SparseLU' が最速であることが分かる．一方'aorta799'，'aorta1661' では'SparseCG' が最速であることが分かる．通常，共役勾配法の計算量は $O(n^2)$ であり，直接法の計算量は $O(n^3)$ である．また，3.3 節に示した通り剛性マトリクスの次数は 438 で，共役勾配法のループ回数を見ると 320 回である．このように次数とループ回数が比較的近い値であり，'aorta197' に対する共役勾配法の計算では，計算量が $O(n^3)$ に近くなってしまったのが'aorta197' で'SparseLU' が最速である原因と考えられる．

よって，医療向けモデルに対し，切断を伴う変形時に応力-変位計算をする際の解法として，共役勾配法と LU 分解による直接法を比べた場合，'aorta197' 程度のノード数の小さいモデルでは LU 分解による直接法が，'aorta799'，'aorta1661' といったノード数の大きいモデルでは共役勾配法が有効であると言える．

5.2　不完全コレスキー分解による前処理

表 3，表 4，表 5 より，不完全コレスキー分解による前処理により，ループ回数に改善が見られ，収束が速くなっている．実際，'aorta197' において，元の剛性マトリクスの形と不完全コレスキー分解による前処理後の形を示した図を図

7，図 8 に，固有値を示した図を図 9，図 10 に示す．するとマトリクスの形の変化は少ないものの，固有値の分布が元の剛性マトリクスでは約 0 から 500 程度であったのに対し，不完全コレスキー分解による前処理後では約 0 から 4 程度となっている．このように不完全コレスキー分解による前処理により固有値のばらつきが格段に小さくなったため，収束が速くなったと考えられる．

ただ，全体実行時間としてはプログラムの最適化が未実装なため，現実的な数値が得られなかった．プログラム最適化後，不完全コレスキー分解による前処理自体に掛かる時間以上に，収束改善による高速化が得られれば，不完全コレスキー分解による前処理は有効であると言える．

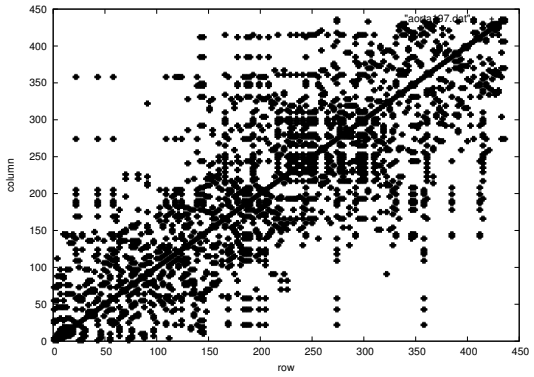

図 7: マトリクスの形

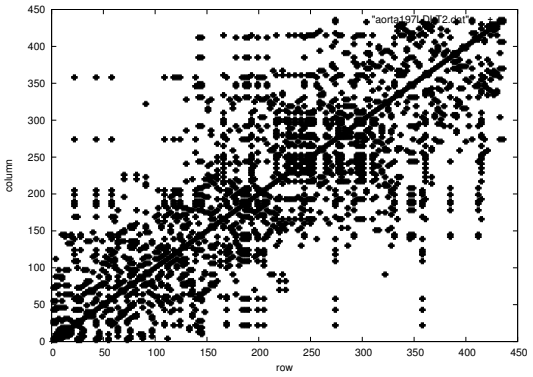

図 8: マトリクスの形 不完全コレスキー分解による前処理後

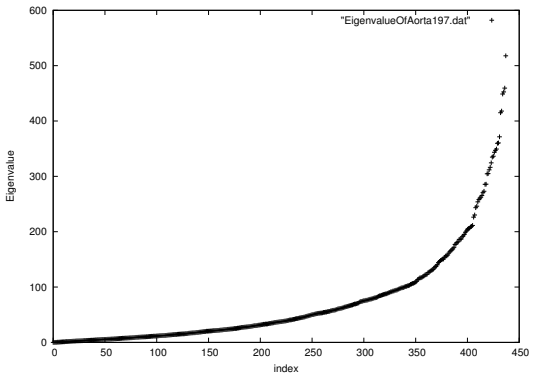

図 9: 固有値

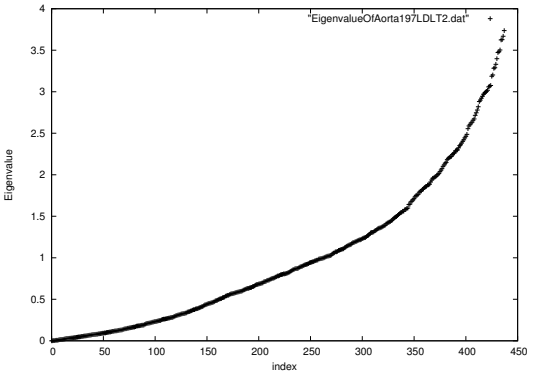

図 10: 固有値 不完全コレスキー分解による前処理後

5.3 最適な前処理

今回，共役勾配法の前処理として一般的な，不完全コレスキー分解を実装した．前処理には多くの種類があり，解くべき問題に対しどの前処理が最適かを知るために，ITBL(Information Technology Based Laboratory) ポータル上の，反復法を用いた連立一次方程式評価システム TiS(Test of Iterative Solvers)[4] がある．

そこで今回，TiS を用いて’aorta197’，’aorta799’，’aorta1661’ の 3 モデルの応力-変位計算の問題に対し，不完全コレスキー分解前処理が適当かどうかを調べる．さらにどの前処理が最適かも併せて調べる．

例として’aorta197’ の応力-変位計算の問題に対し，TiS を用いて得られたグラフを図 11 に示す．

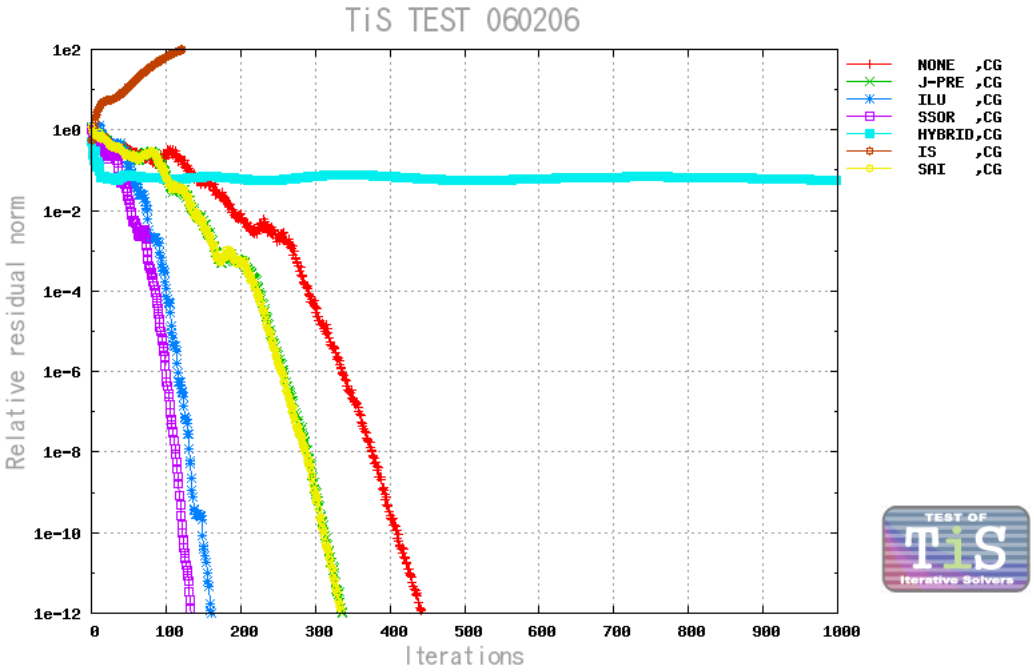

図 11: TiS aorta197

グラフより，実装した不完全コレスキー分解による前処理 (ILU) は 2 番目に速い収束が見られる．一方，最も収束が速いのは対称逐次過剰緩和法 (SSOR) による前処理を行った場合である．この傾向は’aorta799’，’aorta1661’ においても同様であった．

よって，TiS から得られたグラフによれば，3 モデルに対する問題を共役勾配

法で解く際の前処理としては，SSOR による前処理が収束速度の観点からは最も有効と考えられる．

5.4 共役勾配法の並列化

表 6，表 7，表 8 より，並列化に関してベクトル演算全体を並列化したプログラム (ParallelCG(a)) よりも，行列ベクトル積のみ並列化したプログラム (ParallelCG(b)) の方が高速であるとの結果が得られた．これは'ParallelCG(a)' において，各ノードで行列ベクトル積だけでなく，スカラ-ベクトル積やベクトルの加減算といったベクトル計算を並列化して処理するメリットよりも，並列化したことにより増加する各ノード間の通信に要する時間のデメリットが影響した結果と言える．

通信に要するデメリットを見るために，2 種の並列化プログラムにおける，1 ループ内で主な処理の実行時間の割合を示す．

'ParallelCG(a)' については表 9 に，'ParallelCG(b)' については表 10 に示す．各々の並列化で，モデル'aorta197' を用いて，クラスタ 4 台で実行した．

ここで，表中の各処理について述べる．なお，各処理で'local' と付いたものは，並列に処理する部分を示す．

$(\boldsymbol{p}_k, \boldsymbol{r}_k)$　探索方向ベクトル $\boldsymbol{p}_k$ と残差ベクトル $\boldsymbol{r}_k$ の内積．

$A\boldsymbol{p}_k$　係数行列 A とベクトル $\boldsymbol{p}_k$ の積．積の結果をベクトル $A\boldsymbol{p}_k$ とする．

$(\boldsymbol{p}_k, A\boldsymbol{p}_k)$　ベクトル $\boldsymbol{p}_k$ とベクトル $A\boldsymbol{p}_k$ の内積．

$\boldsymbol{x}_{k+1}, \boldsymbol{r}_{k+1}, \boldsymbol{r}_{k+1}$　次ステップの各ベクトルの導出．

Bcast:$\boldsymbol{p}_k$　MPI の集団通信関数 (MPIBcast) を用いてベクトル $\boldsymbol{p}_k$ をマスターから全スレーブに通信する．(ブロードキャスト)

Bcast:err, α, β　MPI の集団通信関数 (MPIBcast) を用いてスカラー err, α, β をマスターから全スレーブに通信する．(ブロードキャスト)

Gather:local:$A\boldsymbol{p}_k$　MPI の集団通信関数 (MPIGather) を用いて各スレーブで並列に処理した $A\boldsymbol{p}_k$ の部分計算結果を，マスターに集める．(ギャザ)

Reduce:local:$(\boldsymbol{p}_k, \boldsymbol{r}_k), (\boldsymbol{p}_k, A\boldsymbol{p}_k), (\boldsymbol{r}_{k+1}, A\boldsymbol{p}_{k+1})$　集団通信関数 (MPIReduce) を用いて各スレーブで計算した各内積の部分内積を結果を，和を取りながらマスターに集める．(レデュース)

表 10 より，行列-ベクトル積のみの並列化では通信が 2 箇所 (Bcast:$\boldsymbol{p}_k$, Gather:local:$A\boldsymbol{p}_k$) である．これに対し表 9 よりベクトル演算全体の並列化では

通信が 7 箇所 (Reduce:local:($\boldsymbol{p}_k, \boldsymbol{r}_k$),Reduce:local:($\boldsymbol{p}_k, A\boldsymbol{p}_k$),Bcast:$\alpha$, Bcast:$err$,Reduce:local:($\boldsymbol{r}_{k+1}, A\boldsymbol{p}_k$),Bcast:$\beta$,Gather:local:$\boldsymbol{p}_{k+1}$) である．通信に要する時間は比較的大きく，共役勾配法のループが多くなるとこの通信時間の影響も大きくなると考えられる．

よって，並列化手法として，2 種の並列化では行列ベクトル積のみの並列化が妥当と言えよう．

表 9: 各部分の実行時間 'ParallelCG(a)'

local:($\boldsymbol{p}_k, \boldsymbol{r}_k$)	0.000429[s]
Reduce:local:($\boldsymbol{p}_k, \boldsymbol{r}_k$)	0.000453[s]
local:$A\boldsymbol{p}_k$	0.001005[s]
local:($\boldsymbol{p}_k, A\boldsymbol{p}_k$)	0.000571[s]
Reduce:local:($\boldsymbol{p}_k, A\boldsymbol{p}_k$)	0.000590[s]
Bcast:α	0.000570[s]
local:$\boldsymbol{x}_{k+1}$	0.000571[s]
local:$\boldsymbol{r}_{k+1}$	0.000571[s]
Bcast:err	0.000572[s]
local:($\boldsymbol{r}_{k+1}, A\boldsymbol{p}_{k+1}$)	0.000571[s]
Reduce:local:($\boldsymbol{r}_{k+1}, A\boldsymbol{p}_{k+1}$)	0.000592[s]
Bcast:β	0.000571[s]
local:$\boldsymbol{p}_{k+1}$	0.000570[s]
Gather:local:$\boldsymbol{p}_{k+1}$	0.000591[s]

表 10: 各部分の実行時間 'ParallelCG(b)'

($\boldsymbol{p}_k, \boldsymbol{r}_k$)	0.000500[s]
Bcast:$\boldsymbol{p}_k$	0.000474[s]
local:$A\boldsymbol{p}_k$	0.000518[s]
Gather:local:$A\boldsymbol{p}_k$	0.000526[s]
($\boldsymbol{p}_k, A\boldsymbol{p}_k$)	0.000498[s]
$\boldsymbol{x}_{k+1}$	0.000500[s]
$\boldsymbol{r}_{k+1}$	0.000500[s]
$\boldsymbol{p}_{k+1}$	0.000500[s]

並列化の効果について，モデルのノードが多くるほど，すなわちモデルが詳細になる程並列化の効果が得られた．これは処理するデータ量が多くなり，並列処理のメリットが活かされたためであろう．実際，'ParallelCG(b)' において，グラフ 6 より，'aorta799' ではクラスタ 8 台で約 4.3 倍，'aorta1661' ではクラスタ 8 台で約 6.4 倍高速化が達成された．

ここで，並列化と 3.1 節，3.2 節，3.3 節で述べた高速化手法を組み合わせる

ことを考える．表 2 と表 6，表 7，表 8 より，’aorta799’，’aorta1661’ では，行列-ベクトル積の並列化と，疎行列性を活かした剛性マトリクスのデータ構造のリスト化を組合わせると，最も高速化されると考えられる．よってこれを実装し，各モデルに対し実行した結果を表 11 に示す.

表 11: 並列化とリスト処理を組み合わせた時の実行時間

	1 台	2 台	4 台	8 台
aorta197	0.066832[s]	0.161722[s]	0.204712[s]	0.323220[s]
aorta799	0.535384[s]	0.509541[s]	0.472755[s]	0.510860[s]
aorta1661	2.316988[s]	1.719447[s]	1.510846[s]	1.224938[s]

すると，確かに’aorta799’ ではクラスタ 4 台で 0.472755[s]，’aorta1661’ ではクラスタ 8 台で 1.224938[s] となり，実装した範囲では最速となった.

5.5 さらなる高速化に対する考察

さらなる高速化手法として，共役勾配法におけるネスティング手法の適用を考える.

まず，共役勾配法におけるネスティング手法を説明する．ある物体の有限要素モデルでメッシュの粗いモデル，細かいモデルの 2 つを用意し，各々の剛性マトリクスにより共役勾配法で応力-変位計算を考える．メッシュの細かいモデルの解を求めたい場合，まず粗いモデルにおける剛性マトリクスを用いて共役勾配法により解を高速に求める．この解を補間して，メッシュの細かいモデルにおける共役勾配法の初期値として代入する．すると，初期値は解に近く，収束も速くなり，結果的に細かいメッシュの解が高速に求められると考えられる．これがネスティング手法である.

ここで解に近い値を初期値として代入した場合の，共役勾配法の収束の高速化を検証する.

そこで，小変形を考え，ある時点の解を共役勾配法で解く際，前の時点の解を初期値として与えた場合の収束具合を調べる.

初期値をゼロとした場合と，初期値を前の時点の解とした場合の収束回数と実行時間を表 12，表 13，表 14 に示す．共役勾配法では 3.3 節のリスト処理を

したもの (SparseCG) を利用する．

表 12: 初期値:ゼロ, 初期値:前の時点の解の比較 aorta197

	実行時間	ループ回数
初期値:ゼロ	0.061617[s]	320[回]
初期値:前の時点の解	0.045090[s]	220[回]

表 13: 初期値:ゼロ, 初期値:前の時点の解の比較 aorta799

	実行時間	ループ回数
初期値:ゼロ	0.508868[s]	514[回]
初期値:前の時点の解	0.476992[s]	446[回]

表 14: 初期値:ゼロ, 初期値:前の時点の解の比較 aorta1661

	実行時間	ループ回数
初期値:ゼロ	2.069610[s]	700[回]
初期値:前の時点の解	1.837363[s]	615[回]

表 12，表 13，表 14 のように，収束は速くなる．よってネスティングによって収束を速めることは可能と考えられる．しかし，実際のネスティングでは，粗いメッシュにおける求解にかかる時間も考慮しなければならない．よって，ネスティングを行う際は，ある程度高速に求解でき，かつその解を補間して初期値として用いればある程度高速に収束する，メッシュの粗さを決める必要がある．

また，物体の 2 つの有限要素モデルは，粗いメッシュのノード集合を N_1，細かいメッシュのノード集合を N_2 とすると，

$$N_1 \subset N_2 \tag{36}$$

を満たさなければならない．よってこれを満たすモデル作成も重要である．

第 6 章　まとめ

本稿では切断手技等の切断を伴う変形を伴う挙動をインタラクティブにシミュレートするために，有限要素モデルで剛性マトリクスを用いた応力-変位計算を共役勾配法で行った．さらに，共役勾配法に対し前処理，疎行列対応のデータ構造への改良 (行列データ構造のリスト化)，並列化といった高速化を施し，各

高速化について評価した．

その結果，実装したプログラムにおいては，’aorta197’ではスカイライン法を用いたLU分解による直接法プログラムが最速であり，’aorta799’ではクラスタ4台で，’aorta1661’ではクラスタ8台で，剛性マトリクスのリスト化と並列化を併用した共役勾配法プログラムが最速であることが分かった．よって，’aorta799’，’aorta1661’といったノード数の大きいモデルでは，切断を伴う変形時，あらかじめ用意しておいた剛性マトリクスを用いることができなくなった場合の応力-変位計算に対して，リスト化，並列化を行った上で，共役勾配法の使用は有効である．ただ，リフレッシュレートを満たす実行時間で計算が行えたのは，’aorta197’で，スカイライン法を用いたLU分解による直接法プログラムを用いた場合のみであった．

今後，リフレッシュレートを満たした共役勾配法によるシミュレータ計算の実現に向けて，高速化手法として5.5節で提案したネスティング手法や，詳細な計算を応力ノードとその周辺といった一部分に限定して，計算を高速に行うLOD(Level Of Detail)手法などがある．こういった新たな高速化手法を確立し，本稿で述べた高速化と併用して計算時間をリフレッシュレートに近づけることが重要である．

謝辞

本研究の機会を与えていただいた富田眞治教授に深甚なる謝意を表します．また，本研究に関して適切なご指導を賜った森眞一郎助教授，中島康彦助教授，嶋田創特任助手，三輪忍助手に心から感謝いたします．

さらに，日頃からご討論いただく京都大学情報学研究科通信情報システム専攻富田研究室の諸兄に心より感謝いたします．Revolverグループとして有益なアイデアと御助言を頂いた本学修士課程2回生岡村大氏，小松原誠氏，篠本雄基氏，吉村知晋氏，本学学士課程4回生野田裕介氏には特に感謝いたします．

また，こちらの様々な要望を聞いていただき，お忙しい中熱心なご指導までしていただいた，京都大学医学部附属病院医療情報部の黒田嘉宏助手にも感謝の意を述べさせていただきます．

参考文献

[1] Yoshihiro Kuroda. A Study on Virtual Reality based Palpation Simulator. 京都大学大学院情報学研究科博士論文, 2005.

[2] Megumi Nakao. Cardiac Surgery Simulation with Active Interaction and Adaptive Physics-Based Modeling. 京都大学大学院情報学研究科博士論文, 2003.

[3] P. パチェコ. MPI 並列プログラミング. 培風館, 2001.

[4] 福井義成, 長谷川秀彦. ITBL における反復解法を用いた連立 1 次方程式の評価システム. 第 34 回数値解析シンポジウム講演予稿集, 2005.

[5] 広田光一, 金子豊久. 柔らかい仮想物体の力覚表現. 情報処理学会論文誌, 1998.

[6] 三好俊郎. 有限要素法入門. 培風館, 1978.

[7] 小国力. 行列計算ソフトウエア WS、スーパーコン、並列計算機. 丸善, 1991.

[8] 中尾恵, 黒田嘉宏. 実時間力学計算手法のライブラリ化と手術シミュレータの開発. 平成 14 年度 IPA 未踏ソフトウェア創造事業「デジタル・ヒューマンを実現する人間機能のモデリングとその応用ソフトウェア」研究報告会, 2003.

付録

A.1　ネスティング

A.1.1　概要

ネスティングとは，有限要素モデルの計算で，荒いメッシュにおける節点の解を求め，その解を補間した上で反復法における初期値として用い，細かいメッシュにおける節点の解を求めることにより，反復法の収束を早めようとした手法である．

ネスティングにおいて，荒いメッシュを構成する節点は細かいメッシュを構成する節点に含まれるものでなければならない．しかし，メッシュ生成ツールではある物体を荒いメッシュと細かいメッシュで分割した際，必ずしも節点の対応関係は成り立たない．（荒いメッシュで分割した場合のある節点の位置に，細かいメッシュで分割した場合に必ずしも節点が存在するとは限らない）

よって，まずネスティング実行のためのモデルを作成すべく，荒いメッシュに新たな節点を置きメッシュを細かくする手法を提案する．その上で，元の荒いメッシュで解を求めて補間する．その後，提案する手法を用いて細かくしたメッシュにおいて，補間した値を初期値として反復法（ここではCG法）により解を求めるといったネスティングを提案する．

A.1.2　メッシュの細分化と剛性マトリクスの生成

まず，剛性マトリクスはMatrixBuilderにおいて物体のメッシュ化されたデータから剛性マトリクスを生成する際，Condensationにより物体内部ノードは省略されているため，表面自由ノードについてのみしか考えられていない．そこで，メッシュを細かくする際，新たに加える節点が物体内部ノードでは解に反映されないため，物体の表面，特に簡単のため表面の辺上，節点間の中点に置くこととする．また，新たに節点を置く辺は，加えた力が影響する範囲は，接触ノード及びその周辺と考えられるため，接触ノードを含む辺，あるいはその周囲から優先して選ぶものとする．

ここで，分割する四面体ghijを考える．h，i，jは表面ノード，gは内部ノードである．今，辺hiの中点に新たな節点kを置くものとする．すると，新たな四面体ghjk，gijkが生成される．この新たな四面体について，要素剛性マトリクスを作成する．図A.1にメッシュ分割の様子を，図A.2に剛性マトリクス生成の様子を示す．

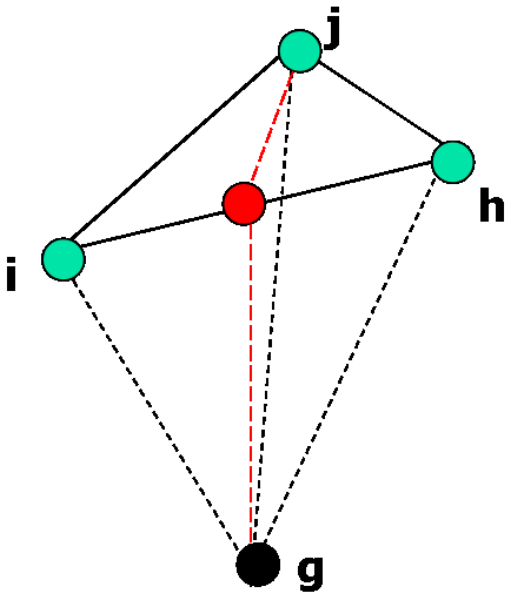

図 A.1: 四面体分割

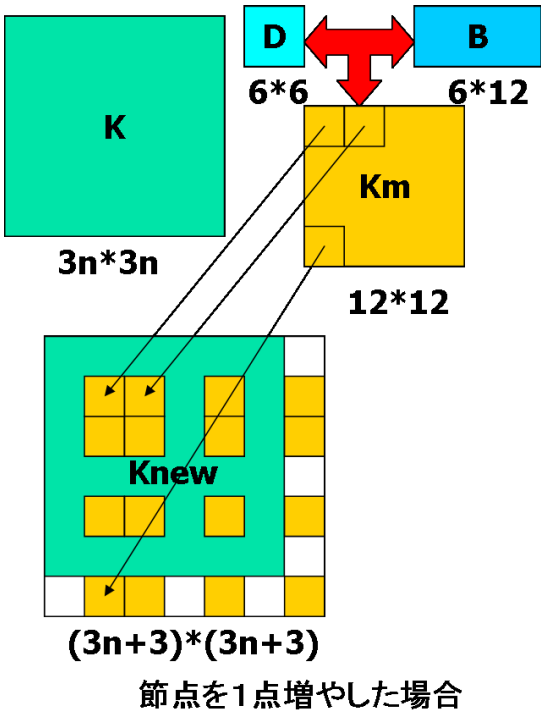

図 A.2: 剛性マトリクス生成

この要素剛性マトリクスを新たな剛性マトリクス（元の剛性マトリクスを 3n*3n とすると，(3n+3)*(3n+3)．3n*3n の部分は元の剛性マトリクスの値が格納されており，新たな節点用に 3 行 3 列外側に付け加えたもの）の対応する箇所に足し込む．こうすることにより，新たに細かいメッシュにおける剛性マトリクスができる．

要素剛性マトリクスは以下のように記述される．

$$K_m = VB^T DB \tag{A.1}$$

但し，V は四面体の体積，B はひずみ-変位マトリクス，D は応力-ひずみマトリクスである．

この要素剛性マトリクス K_m を導出するために，ひずみ-変位マトリクス B，応力-ひずみマトリクス D の求め方を以下に示す．ひずみ-変位マトリクス B 四面体 ghjk を考える．メッシュ生成の仕様より，四面体要素内で，応力，ひずみが一定である．すると，これは要素内で変位が直線的に変化すると仮定したことに等しく，要素内の変位場を以下のように仮定できる．

$$u = \alpha_1 + \alpha_2 x + \alpha_3 y + \alpha_4 z$$

$$
\begin{aligned}
v &= \alpha_5 + \alpha_6 x + \alpha_7 y + \alpha_8 z \\
w &= \alpha_9 + \alpha_{10} x + \alpha_{11} y + \alpha_{12} z
\end{aligned} \tag{A.2}
$$

但し，u:x 方向の変位，v:y 方向の変位，w:z 方向の変位である．この式を変位関数という．

定数 α_1，α_2，$\cdots$,α_{12} を節点変位 u_g，v_g，w_g，u_h，$\cdots$,w_k，および節点座標 x_g，y_g，z_g，x_h，$\cdots$,z_k で表す．まず，u_g，v_g，w_g，u_h，$\cdots$,w_k を定数 α_1，α_2，$\cdots$,α_{12} と x_g，y_g，z_g，x_h，$\cdots$,z_k で表すと，以下のようになる．

$$
\begin{aligned}
u_g &= u(x_i, y_i, z_i) = \alpha_1 + \alpha_2 x_i + \alpha_3 y_i + \alpha_4 z_i \\
v_g &= v(x_i, y_i, z_i) = \alpha_5 + \alpha_6 x_i + \alpha_7 y_i + \alpha_8 z_i \\
w_g &= w(x_i, y_i, z_i) = \alpha_9 + \alpha_{10} x_i + \alpha_{11} y_i + \alpha_{12} z_i \\
u_h &= u(x_h, y_h, z_h) = \alpha_1 + \alpha_2 x_h + \alpha_3 y_h + \alpha_4 z_h \\
&\cdots\cdots, \\
w_k &= w(x_k, y_k, z_k) = \alpha_9 + \alpha_{10} x_k + \alpha_{11} y_k + \alpha_{12} z_k
\end{aligned} \tag{A.3}
$$

これらを解くと，α_1，α_2，$\cdots$,α_{1_2} が求まる．マトリクス表示すると，

$$
C\boldsymbol{\alpha}_u = \boldsymbol{u}, C\boldsymbol{\alpha}_v = \boldsymbol{v}, C\boldsymbol{\alpha}_w = \boldsymbol{w} \tag{A.4}
$$

となる．但し，

$$
C = \begin{bmatrix} 1 & x_g & y_g & z_g \\ 1 & x_h & y_h & z_h \\ 1 & x_j & y_j & z_j \\ 1 & x_k & y_k & z_k \end{bmatrix} \tag{A.5}
$$

$$
\boldsymbol{\alpha}_u = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix}, \boldsymbol{\alpha}_v = \begin{bmatrix} \alpha_5 \\ \alpha_6 \\ \alpha_7 \\ \alpha_8 \end{bmatrix}, \boldsymbol{\alpha}_v = \begin{bmatrix} \alpha_9 \\ \alpha_{10} \\ \alpha_{11} \\ \alpha_{12} \end{bmatrix} \tag{A.6}
$$

$$
\boldsymbol{u} = \begin{bmatrix} u_g \\ u_h \\ u_j \\ u_k \end{bmatrix}, \boldsymbol{v} = \begin{bmatrix} v_g \\ v_h \\ v_j \\ v_k \end{bmatrix}, \boldsymbol{w} = \begin{bmatrix} w_g \\ w_h \\ w_j \\ w_k \end{bmatrix}, \tag{A.7}
$$

である．今，C の逆行列を C^{-1} とすると，

$$\boldsymbol{\alpha}_u = C^{-1}\boldsymbol{u}, \boldsymbol{\alpha}_v = C^{-1}\boldsymbol{v}, \boldsymbol{\alpha}_w = C^{-1}\boldsymbol{w} \tag{A.8}$$

とすることにより，$\boldsymbol{\alpha}_u$，$\boldsymbol{\alpha}_v$，$\boldsymbol{\alpha}_w$ ，つまり，α_1，α_2，$\cdots$,α_{12} が求まる．いま，定数 a_g，a_h，a_j，a_k，b_g，$\cdots\cdots$,d_k を用いて

$$C^{-1} = \begin{bmatrix} a_g & a_h & a_j & a_k \\ b_g & b_h & b_j & b_k \\ c_g & c_h & c_j & c_k \\ d_g & d_h & d_j & d_k \end{bmatrix} \tag{A.9}$$

と置けて，

$$\alpha_1 = a_g u_g + a_h u_h + a_j u_j + a_k u_k \tag{A.10}$$

$$\alpha_2 = b_g u_g + b_h u_h + b_j u_j + b_k u_k \tag{A.11}$$

$$\alpha_3 = c_g u_g + c_h u_h + c_j u_j + c_k u_k \tag{A.12}$$

$$\alpha_4 = d_g u_g + d_h u_h + d_j u_j + d_k u_k \tag{A.13}$$

$$\alpha_5 = a_g v_g + a_h v_h + a_j v_j + a_k v_k \tag{A.14}$$

$$\cdots\cdots, \tag{A.15}$$

$$\alpha_{12} = d_g w_g + d_h w_h + d_j w_j + d_k w_k \tag{A.16}$$

$$\tag{A.17}$$

と表すことができる．これを式 (A.3) に代入すれば，変位-節点変位関係が次のように表される．

$$\begin{aligned} u = {} & (a_g + b_g x + c_g y + d_g z)u_g + (a_h + b_h x + c_h y + d_h z)u_h \\ & +(a_j + b_j x + c_j y + d_j z)u_j + (a_k + b_k x + c_k y + d_k z)u_k \end{aligned}$$

$$\begin{aligned} v = {} & (a_g + b_g x + c_g y + d_g z)v_g + (a_h + b_h x + c_h y + d_h z)v_h \\ & +(a_j + b_j x + c_j y + d_j z)v_j + (a_k + b_k x + c_k y + d_k z)v_k \end{aligned} \tag{A.18}$$

$$\begin{aligned} w = {} & (a_g + b_g x + c_g y + d_g z)w_g + (a_h + b_h x + c_h y + d_h z)w_h \\ & +(a_j + b_j x + c_j y + d_j z)w_j + (a_k + b_k x + c_k y + d_k z)w_k \end{aligned}$$

また，ひずみについて，垂直ひずみε，せん断ひずみγに関して，四面体要素では

$$\varepsilon_x = \frac{\partial u}{\partial x}, \varepsilon_y = \frac{\partial v}{\partial y}, \varepsilon_z = \frac{\partial w}{\partial z} \tag{A.19}$$

$$\gamma_{xy} = \frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}, \gamma_{yz} = \frac{\partial w}{\partial y} + \frac{\partial v}{\partial z}, \gamma_{zx} = \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \tag{A.20}$$

であるから，ε_x，ε_y，ε_z，γ_{xy}，γ_{yz}，γ_{zx}は，u_g，v_g，w_g，u_h，$\cdots$,w_kの式で，

$$\varepsilon_x = b_g u_g + b_h u_h + b_j u_j + b_k u_k \tag{A.21}$$

$$\varepsilon_y = c_g u_g + c_h u_h + c_j u_j + c_k u_k \tag{A.22}$$

$$\varepsilon_z = d_g u_g + d_h u_h + d_j u_j + d_k u_k \tag{A.23}$$

$$\gamma_{xy} = c_g u_g + c_h u_h + c_j u_j + c_k u_k + b_g v_g + b_h v_h + b_j v_j + b_k v_k \tag{A.24}$$

$$\gamma_{yz} = d_g v_g + d_h v_h + d_j v_j + d_k v_k + c_g w_g + c_h w_h + c_j w_j + c_k w_k \tag{A.25}$$

$$\gamma_{zx} = b_g w_g + b_h w_h + b_j w_j + b_k w_k + d_g u_g + d_h u_h + d_j u_j + d_k u_k \tag{A.26}$$

と表される．これをマトリクス表示すると，以下のようになる．

$$\boldsymbol{\varepsilon} = B\boldsymbol{\delta} \tag{A.27}$$

但し，

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{zx} \end{bmatrix}, \boldsymbol{\delta} = \begin{bmatrix} u_g \\ u_h \\ u_j \\ u_k \\ v_g \\ \vdots \\ w_k \end{bmatrix}, \tag{A.28}$$

$$B = \begin{bmatrix} b_g & b_h & b_j & b_k & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & c_g & c_h & c_j & c_k & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & d_g & d_h & d_j & d_k \\ c_g & c_h & c_j & c_k & b_g & b_h & b_j & b_k & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & d_g & d_h & d_j & d_k & c_g & c_h & c_j & c_k \\ d_g & d_h & d_j & d_k & 0 & 0 & 0 & 0 & b_g & b_h & b_j & b_k \end{bmatrix} \tag{A.29}$$

であるこの B をひずみ-変位マトリクスと呼ぶ.

応力-ひずみマトリクス D 弾性体において，フックの法則が成立し，応力とひずみの関係は

$$\sigma = E\varepsilon \tag{A.30}$$

で表される．なお，E はヤング率である．また，ある一様断面の棒を考えた際，軸方向に引っ張られて ε のひずみを生じる時は，これと直角な方向に ε' なるひずみが生じ，

$$\varepsilon' = -\nu\varepsilon = -\nu\frac{\sigma}{E} \tag{A.31}$$

の関係が成立することが知られている（ポアソン効果）．ここで ν はポアソン比である．ここで，引っ張る方向を x 軸に取った場合，各方向のひずみは，式 (A.33)(A.34) に対応して，

$$\varepsilon_x = \frac{1}{E}\sigma_x, \varepsilon_y = -\nu\varepsilon_x = -\frac{\nu}{E}\sigma_x, \varepsilon_z = -\nu\varepsilon_x = -\frac{\nu}{E}\sigma_x \tag{A.32}$$

と表せる．これは x 方向の応力 σ_x が単独で x，y，z 方向に生じさせるひずみを示したものであるから，一般の応力状態，すなわち σ_x，σ_y，σ_z が同時に存在する場合のひずみは，σ_x，σ_y，σ_z が各々単独で作用する場合の結果を重ね合わせて得られる．σ_y，σ_z についても式 (A.35) に対応した式を求め，それらを重ね合わせると，以下のようになる．

$$\varepsilon_x = \frac{1}{E}\{\sigma_x - \nu(\sigma_y + \sigma_z)\}, \varepsilon_y = \frac{1}{E}\{\sigma_y - \nu(\sigma_z + \sigma_x)\}, \varepsilon_z = \frac{1}{E}\{\sigma_z - \nu(\sigma_x + \sigma_y)\} \tag{A.33}$$

また，この時のせん断応力 τ とせん断ひずみ γ の関係は以下のように表される．

$$\tau_{xy} = \frac{E}{2(1+\nu)}\gamma_{xy}, \tau_{yz} = \frac{E}{2(1+\nu)}\gamma_{yz}, \tau_{zx} = \frac{E}{2(1+\nu)}\gamma_{zx} \tag{A.34}$$

式 (A.36)(A.37) より応力成分 σ_x，σ_y，$\vdots$，τ_{zx} について解くと，以下のようになる．

$$\sigma_x = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)}\{\varepsilon_x + \frac{\nu}{1-\nu}(\varepsilon_y + \varepsilon_z)\}, \tag{A.35}$$

$$\sigma_y = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)}\{\varepsilon_y + \frac{\nu}{1-\nu}(\varepsilon_z + \varepsilon_x)\}, \tag{A.36}$$

$$\sigma_z = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)}\{\varepsilon_z + \frac{\nu}{1-\nu}(\varepsilon_x + \varepsilon_y)\} \tag{A.37}$$

$$\tau_{xy} = \frac{E}{2\left(1+\nu\right)}\gamma_{xy}, \tau_{yz} = \frac{E}{2\left(1+\nu\right)}\gamma_{yz}, \tau_{zx} = \frac{E}{2\left(1+\nu\right)}\gamma_{zx} \tag{A.38}$$

これをマトリクス表示すると，以下のようになる．

$$\begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{xy} \\ \tau_{yz} \\ \tau_{zx} \end{bmatrix} = \begin{bmatrix} 1 & \frac{\nu}{1-\nu} & \frac{\nu}{1-\nu} & 0 & 0 & 0 \\ \frac{\nu}{1-\nu} & 1 & \frac{\nu}{1-\nu} & 0 & 0 & 0 \\ \frac{\nu}{1-\nu} & \frac{\nu}{1-\nu} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} \end{bmatrix} \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{zx} \end{bmatrix} \tag{A.39}$$

式 (A.42) の係数マトリクス D が応力-ひずみマトリクスである．