Vol.2014-HPC-143 No.13
2014/3/3

操作の連続性を考慮した投機計算を行うインタラクティブシミュレーションシステム

松井 祐太[1]　岩永 翔太郎[1]　福間　慎治[1]　森　眞一郎[1]

概要：本論文では，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の高速化を目的としている．シミュレーション対象は有限要素モデル化され，操作に対する反応のシミュレーションは連立一次方程式の前処理付反復解法によって得られるものとする．この時，1 秒間に 10〜30 回程度のシミュレーションを連続的に繰り返す状況を考えると，従来の並列処理効果だけに依存した最適化のみでは，実時間処理可能な有限要素モデルの次元数は低く抑えられてしまい，精度が低いシミュレーションとなる．そこで本論文では，より次元数の高い，すなわち高精度なシミュレーションを実現する手法として，対話的な操作の連続性に着目し，係数行列の変化を予測して求解時に必要となる前処理行列を投機的に計算する投機実行モデルを提案する．また，提案モデルに基づいたプロトタイプシステムの実装ならびに評価を行い報告する．

キーワード：インタラクティブシミュレーション，実時間シミュレーション，投機実行，前処理付き反復計算，並列処理，スーパーコンピューティング

1. はじめに

我々は次世代のシミュレーション技術として，高性能な計算サーバ上での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．従来のスーパーコンピュータ上でのバッチ処理的なシミュレーションに比べて実時間性が極めて重視される一方で，シミュレーションの精度に対しては，真の物理現象と大幅にかけ離れていない限りにおいて人間の視覚，触覚等の識別能力を超える精度は求められない．

このようなシミュレーションの代表例として，医療分野における手術シミュレーションが挙げられる．近年の医療技術の進化と発展に伴い医師には高度な技能が求められており，特に外科分野における医師や研修生に対する効率的で倫理的に安全なスキル向上手法として，人体組織の力学や生理をモデル化した仮想物体を用いた手術シミュレーションの実現が期待されている [1][2][3].

また，次世代プロセッサのメニーコア化は非常に多くの並列処理環境を我々に提供する．しかしながら，中規模程度のアプリケーションでは並列処理を行う際にこれらの計算資源を十分に活用することが困難である．本論文では，インタラクティブシミュレータの実用化に向けて，1 台のコンピュータでは実現不可能な精度のシミュレーションを，有り余るコンピュータ資源を用いた並列処理とインタラクティブシミュレーションにおけるユーザ介入の連続性を利用した投機計算による高速化により実現する手法を提案し，実装する．

具体的なシミュレーション対象のモデルとしては，破壊変形をともなう構造変形シミュレーションを取り上げる．そして，有限要素法を用いたシミュレーションにおいて必須とされる剛性方程式と呼ばれる連立一次方程式の求解問題にについての記述を行う．

連立一次方程式の数値解法としては直接解法と反復解法に大別できる．今回対象とするシミュレーションの次元数としては数千のオーダを想定しており，かつ，対称疎行列を想定しているため，数値解法としては反復法を前提とする [4]. なお，時間制約があるインタラクティブシミュレーションにおいて反復法を利用するメリットとして，制約時間内に得られた最善の解を近似解として提示することが可能である点があげられる．以下，本論文では反復法のソルバとしては，共役勾配法に前処理を施すことにより収束速度を高めた前処理付共役勾配法（PCCG 法）を前提とする．特に今回は前処理には不完全コレスキー分解を用いることとする．

[1] 福井大学大学院　工学研究科

2. 研究背景

本論文で用いるシミュレーション物体は，有限要素法を用いて処理を行っている．有限要素法 (Finite Element Method) とは，変形に対し無限の自由度を持つ弾性体を有限の自由度を持つ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）を解く手法である．

弾性体は四面体メッシュによって構成されていると仮定する．図 1 に弾性体の四面体メッシュ分割の様子を示す．

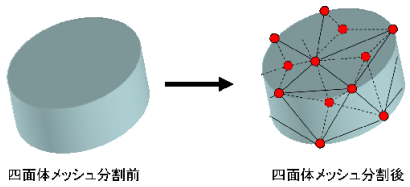

図 1 四面体メッシュ分割

これらの点に加わる力と変位の関係は，剛性マトリクスと呼ばれる係数行列 K を用いて式（1）のような形で表される．

$$Ku = f \tag{1}$$

また，この方程式は剛性方程式と呼ばれる．本来であれば，直接的に加わる力だけでなく慣性項や粘性項を考慮する必要があるが，計算コストの問題から考慮しないものとする．[5][6] 我々が取るこの解法を陰解法と呼ばれる解法である．陰解法では一般的にシミュレーションのタイムステップを自由に設定することが可能であると言われている．要素の計算領域を小さくすると同時にシミュレーションのタイムステップを細かく設定する陽解法と呼ばれる手法を用いることも考えられるが，そのタイムステップをどれくらいに設定すればある程度の精度が得ることができるのか不明であるので，今回の解法には陰解法を用いるものとした．

シミュレーションにおいて，シミュレーション物体を構成する四面体が破壊されるような場合は，剛性マトリクスが変化する．逆に言えば，シミュレーション物体を構成する四面体に力が加わった場合でも，それを構成する四面体が破壊さえされなければ剛性マトリクス K は変化しない．以下では，剛性マトリクス K が変化しないシミュレーションを非破壊変形，変化するシミュレーションを破壊変形と呼ぶ．有限要素法における対象物体では，剛性マトリクスは対象正定値な疎行列となる．そのため，この種の問題に対しては反復計算法として一般に共役勾配法 (CG 法) が有効であると考えられる．今回用いる解法では，CG 法に不完全コレスキー分解による前処理を加えた前処理付き共役勾配法（ICCG 法）を採用する．

通常の大規模数値シミュレーションでは，初期状態といくつかのパラメータをシミュレーションを開始する前に設定して，大型計算機等で計算を行い，後にシミュレーションの最終結果の可視化を行うことが一般的であり，そのパラメータを計算の途中で変更することはきわめて稀である．

しかし，われわれが想定する構造変形シミュレーションは対話性の要求されるシミュレーションであり，計算途中にユーザの判断によってパラメータを変更することがしばしば起こり，それに伴ってそのような時々刻々と変わる状況を実時間でシミュレーションしていかなければならない．このような対話性の要求されるシミュレーション環境では高精度なシミュレーションはもちろんのこと，従来の数値シミュレーションではあまり重視されていなかった実時間での応答が必要とされる．

しかしながら，本論文で採用する ICCG 法は，大規模問題の計算においては高い並列化効果を期待できるが，並列処理を行う各ノードで秒間 30 ステップ程度の実時間処理可能な規模の問題サイズを考えると，ノード間通信のオーバーヘッドを始めとする並列化オーバーヘッドの影響により，単純なデータ並列性のみを利用して高速化の恩恵を預かることができない．また，反復法の反復計算部分に対しては計算精度とのトレードオフで実時間処理が不可能ではないが，剛性マトリクスが変化した際に必要となる前処理行列の更新手順では制約時間内で計算を打ち切る手法は使えない．そこで，この前処理行列導出時間の隠蔽を図る手法を提案し，そのフレームワークの構築を行う．

3. 操作の連続性を利用した投機実行モデル

本章では物体に対する操作の連続性を意識した投機予測による前処理行列導出時間の隠蔽についての議論を行う．

3.1 剛性マトリクス K の連続性の利用

破壊変形を伴うシミュレーションにおいては破壊変形が起きた際に剛性マトリクスが新たに構成される．実時間シミュレーションを実現するためには，視覚的に違和感が無い程度の応答速度が必要である．すなわち，シミュレーションのタイムステップを 33[ms] 以下にする必要があるが，破壊変形は毎ステップ生ずるとは限らない．以下，話を整理するため，シミュレーションのタイムステップ（Step）に対して，破壊変形が起こるタイムステップをステージ（Stage）と呼ぶ．具体的なシミュレーションの流れを図 2 に示す．図中において横軸は Step を表しており，K_i の矢印が指す Step において，破壊変形が生じ剛性マトリクス K が更新されることを表している．ここで添字 i は破壊変形の Stage 数を表している．

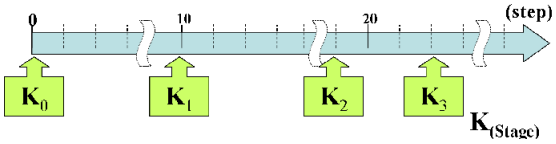

図 2 破壊変形を伴うシミュレーションの流れ

3.2 剛性マトリクス K の投機予測

有限要素法で作成された物体に対してはマクロな視点から見ると高い連続性が期待できるため，次に発生する破壊変形を高い確率で予測することが可能であると考えられる [7]．また，われわれが採用するモデルでは破壊変形の前後で要素数が不変であり，ランク落ちも発生しないという仮定がある [8]．そのため，破壊変形前の剛性マトリクス K_t と破壊変形後の剛性マトリクス K_{t+1} は類似度が高いと期待できる．そこで，次の状態として想定される複数の破壊変形を予測し，それらに応じた剛性マトリクス K_{t+1} を生成し，投機的に前処理行列 P_{t+1} の計算を開始することで前処理更新までのタイムラグを短縮することを考える．

この際に剛性マトリクスそのものの投機予測を行った場合，予測が外れた際には最初から再計算が必要であるのに対して，ICCG 法では前処理行列が適度な近似精度を持った逆行列であればよいという性質を活用することで，予測が完全に一致しなかった場合においても，最も類似した破壊パターンに対する前処理結果を前処理行列として採用することで，再計算を不要とすることができる．[9] これにより，投機の効果が少なからず回収できる仕組みを実現可能である．予備評価より，構造変形シミュレーションにおける連立一次方程式の解法において ICCG 法を用いた際に前処理計算と反復計算を比較した場合，前処理計算に多くの時間をとられていることがわかっている．このことからも，処理に多くの時間が必要とされる前処理の投機を行うのは妥当なことであると考える．

投機実行の処理フローとしては，図 3 のようなものを考える．横軸には Stage と Step の 2 つの時間軸をとっている．その処理は，反復解法を行う処理プロセスと投機的に前処理を行う処理プロセスに分かれる．図 3 のように，投機的に前処理を生成し，完了次第投機判定を行い，適当なプロセスを選び，その前処理をもとに反復解法を行うように処理プロセスを切り替える．このようなシステムにより，収束速度の低下を防ぐことができると考える．

このような前処理の投機実行モデルには，大きくわけて 2 つのパターンが考えられる．一つのモデルは，投機計算による前処理行列の導出を次の破壊変形が発生するまでに終えることができる場合であり，もう一つのモデルは投機計算による前処理行列の導出を次の破壊変形が発生するまでに終えることができないモデルである．前者の場合，話は簡単である．破壊変形が発生した瞬間に，その剛性マトリクスに対する前処理行列が用意されている．正しいと判

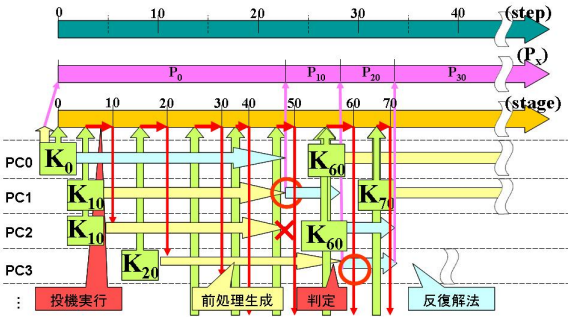

図 3 投機処理フロー

定された前処理行列を用いて計算すればよい．

後者の場合は，話が少し複雑になる．破壊変形が発生した瞬間には前 Step での投機結果は回収されない．そのため数 Stage 先の投機を行う必要が出てくる．このモデルにおけるシミュレーションの流れは 3.3.4 項で解説する．

3.3 システムモデル

本システムは，外部からの入力を受け付ける操作端末，操作端末からの入力および操作端末への出力を受け付ける管理ノード，そして，前処理行列の投機計算を行う投機計算ノード群，解計算を行う解計算ノード群に分けられる．図 4 に概念図を示す．

3.3.1 各ノードの動作

- 操作端末
  ユーザの動きを入力として，剛性方程式における f の値を管理ノードに送信する．また，管理ノードより解計算の結果として，剛性方程式における変位 u を受け取る．f および u は毎 Step 送受信される必要がある．

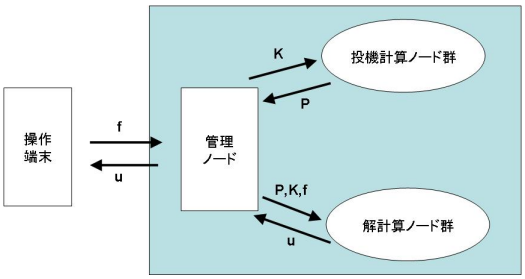

図 4 投機システムの概念モデル

- 管理ノード
  操作端末との通信並びに投機計算及び解計算を管理する役割を持つ．本ノードは投機計算ノード群および解計算ノード群に対する情報を常に管理している．投機計算ノード群に対しては，Δs 時刻後に発生するであろう破壊変形についての剛性マトリクス $K_{i+\Delta s}$ を送信する．そして，投機した結果として予測された形状に対する前処理行列 $P_{i+\Delta s}$ をその $K_{i+\Delta s}$ をもとに計算させ，その前処理行列を受信する．解計算ノード群

に対しては，受信した前処理行列から最適なものを選択し，その時刻 i において既知の前処理行列 P_i 及び剛性マトリクス K_i，力 f_t を送信し，計算結果として u_t を受信する．

- 投機計算ノード群
  管理ノードから受信した剛性マトリクス K に対して不完全コレスキー分解を行い，前処理行列 P を計算する．求まった P を管理ノードに送信する．
- 解計算ノード群
  管理ノードより前処理行列 P，剛性マトリクス K，力 f を受信し，不完全コレスキー分解を前処理に用いた共役勾配法である ICCG 法の処理を行い，その時刻に対する解 u を計算する．そして，求まった解 u を管理ノードに送信する．

3.3.2 通信の削減

投機実行システムのモデルは 3.3.1 項で示した．ここでは，実態に即したものを解説する．3.3.1 項で投機計算ノード群は管理ノードから剛性マトリクスを受信し，前処理行列を管理ノードに送信する．また，管理ノードは受信した前処理行列を解計算ノード群に送信する，と述べた．しかし，剛性マトリクスおよび前処理行列の容量は，要素数 1024 のモデルにおいて，double 型のデータを用いた場合には 72MB 必要となる．ギガビットイーサを利用したと仮定してもこれでは通信遅延が非常に大きな問題となる．したがって，各ノードの役割を固定するのは適当ではない．投機計算ノードと解計算ノードの役割を固定すると，前処理行列 P が求まる度に，投機計算ノードから管理ノードへ，管理ノードから解計算ノードへ通信が発生する．これを回避するため，前処理行列 P を導出し，成功判定に該当した投機ノードはそのまま解計算ノードとして用いること，また，$i-1$ のタイムステップで解計算を行っていた解計算ノードは，投機ノードとして割り当てることで，導出した前処理行列 P を通信することなく解計算が可能となる．このようなクラスタ内の PC の役割が流動的に変化するシステムを採用することで，データ通信量を大きく削減することが可能となる．

さらにここで，剛性マトリクス K の作成方法について考えてみる．有限要素法における剛性マトリクスは，仮想物体を構成する各要素が，まわりの要素に及ぼす力の係数の値を重ね合わせることで表現されている．そのため，破壊変形によって物体構造が変化する場合というのは，仮想物体を構成する頂点の座標が変化した場合のことを表している．また，仮想物体において，ある頂点の変位が及ぼす力は，その頂点の近傍にのみ届く．つまり，ある時刻における剛性マトリクス K_i とそこから破壊変形が発生した後の剛性マトリクス $K_{i+\Delta s}$ を比較すると，Δs が小さければ変化した剛性マトリクスの制分数はそれほど多くない．したがって，剛性マトリクス K_i を所持しているノードに対しては，変化後の剛性マトリクス $K_{i+\Delta s}$ と剛性マトリクス K_i との差分行列を送信すれば当該ノードにおいては剛性マトリクス $K_{i+\Delta s}$ を復元することができる．この差分行列については，変化する剛性マトリクスの成分数がそれほど多くないため，ほとんどの要素がゼロとなる．したがって，ゼロ要素を省略するデータ構造である CRS 形式を用いれば，非常に小さい通信量となる．差分行列を受け取ったノードでは $K_{i+\Delta s}$ の復元処理が必要であるが，$K_{i+\Delta s}$ の送受信に必要なコストと比較して，はるかに小さいコストとなる．

3.3.3 最尤判定

前処理行列の投機計算を行った場合に，どの投機予測が正しい変化にもっとも近いのかを判定する必要がある．言い換えると，収束改善率がもっとも高い前処理行列を探すということになる．理論的に考えると，次のような判定手法が考えられる．

- 投機により求めた前処理行列 P_i，そしてそのステージにおける剛性マトリクス K_i の積をとり，その行列式 $|P_iK_i|$ が 1 に近いか調査
- P_iK_i の固有値を計算

しかし，これらの判定手法は数学的な処理を行うこととなり，実時間での判定は困難である．そこで，計算量の少ない判定方法を考える．候補としては以下のようなものが考えられる．

- 試行計算による判定
- 対象構造の類似性

まず，試行計算による判定では，投機計算の結果として得られた前処理行列に対して，実際に変化した係数行列を用いて反復計算の試行を行い，反復回数の少ない前処理行列を採用する手法である．この判定方法では計算量は反復計算 1 回分のみとなる．次に，対象構造の類似性による判定は以下の 2 つの方針からなる．

(1) 投機した形状が正しい予測であればその予測が正解
(2) 剛性マトリクスの要素毎の差の 2 乗和の小さいものが正解

1 のケース，つまり予測した形状と同様の破壊変形が生じた場合は即座にその予測を正解とすればよい．仮に 1 のケースに該当しなかった場合，2 の選択肢を採用する．2 の選択肢では計算を行う必要があるが，前述の固有値計算などの選択肢と比べると，計算量は非常に小さくなる．

このようにいくつか判定手法が考えられるが，今回我々が対象とするシステムでは試行計算による判定を採用し，収束改善率のみによる評価とする．

3.3.4 破壊変形の頻度が高い場合への対応

ここでは，前処理行列の計算を行うのに必要とされる時間について考える．破壊変形が起こった Step から，次に破壊変形が起こる Step までを 1 Stage とし議論を進める．もし仮に，1 ステージ以内に前処理行列を求めることがで

きるならば，その時点での解計算ノードを除くすべての計算資源を，前処理行列の計算に使用することができる．次の破壊変形が起こるまでに解の計算が終了するため，毎回すべての資源を使用することができる．

しかし，前処理行列の計算は常に 1 ステージ以内に終了するとは限らないため，前処理行列を計算するために，m ステージかかる場合のシミュレーションについて以下の 2 つから考えてみる．

(1) 次の投機計算ノードの割り当て
(2) 解計算にどの前処理行列を使うか

まず，資源を複数のグループに分割し，この時のグループ数を x とする．つぎに，一度に y 台投機するような状況を考える．また，最初に投機を行うノード群以外は，シミュレーションスタート直後は何も役割を与えられていない．この役割を与えられていないノード群を，以降，資源ノード群と呼ぶ．このような仮定のもとにシミュレーションの流れを説明する．

資源ノード群から選んだ group を，以降単純に group と呼ぶことにする．図 5 はシミュレーションの流れ図である．t は時刻を表しており，時刻 t は左から右に進んでいく．また，ステージ番号とその時の投機計算にかかる時間も各 group 毎に示している．

破線については，i ステージでの投機計算が終了する時刻を表している．group0 は i ステージでの投機計算が終わったならば，破線以降でまた投機計算をする資源として利用することができる．

また，計算資源をこのように分割して使うことで，差分行列を一定の間隔で更新していくことができる．

シミュレーションにおける資源ノードの状態遷移に関する図を図 6 に示す．管理ノードは，ある一定以上の台数が投機状態として残っている場合には，投機ノード群から前述の投機アルゴリズムに従って前処理行列を計算させるためのノードを選択する．そこで，選択されたノードはこれ以降投機計算ノードとして振る舞うこととなる．その時刻において，投機計算を行わないノードは待機ノードとして残す．投機計算ノードでは，与えられた情報をもとにして，前処理行列の計算を行う．前処理行列の導出が終わると，試行計算のためのデータ受信待ち状態になる．データを受け取ったら試行計算を行い，そのときの反復回数を管理ノードに返す．この時の管理ノードの投機予測の当たり判定によって，試行計算ノードのこれからの振る舞いが決まってくる．投機予測が当たりと判定された試行計算ノードは解計算ノードへと移行する．一方で，投機予測が外れと判定された試行計算ノードは待機ノードへと追加される．解計算ノードへと状態遷移したノードは，持っている前処理行列をもとに ICCG 法の反復計算を行い，解を導出する．

本シミュレーションにおいて，新しい前処理行列の計算が終了するまでこの処理を繰り返すことになる．そして，新しい前処理行列が計算され，導出された後に試行計算ノードを解計算ノードに切り替える．その上で，一つ前の処理で解計算を行っていた解計算ノードを待機ノードへと移行させ，投機ノード群に追加する．この解計算ノードを切り替える操作については，管理ノードがその役割を担うこととなる．

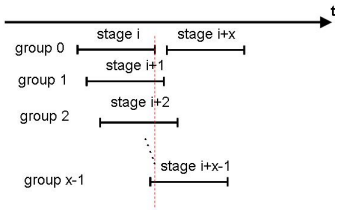

図 5 各資源 group と投機計算の関係

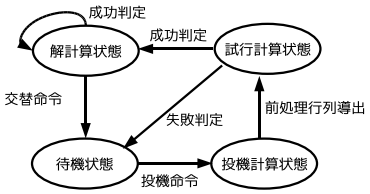

図 6 資源ノードの状態遷移

3.3.5 管理ノード

ここでは，管理ノードの役割について考える．まず，管理ノードの 1 つ目の役割として，投機制御が挙げられる．多くの計算資源を管理しつつ，1Stage 毎の計算を複数並行に処理させるスループット処理の計算を行う．2 つ目の役割として，解計算制御が挙げられる．少ない計算資源で 1Step 毎の応答が必要な処理の制御を行う．

これらの役割を管理ノードが単一プロセスで管理を行っていた場合，異なる性質の仕事が混在することとなるため，管理ノードの処理が煩雑になることが考えられる．そこで，管理ノード処理を投機計算管理プロセスと解計算管理プロセスの 2 つに分ける．投機計算管理プロセスは，投機予測に関する情報の管理を行うプロセスである．解計算管理プロセスは解計算についての管理を行うプロセスであり，操作端末に解計算の結果の送信も行う．

4. プロトタイプシステムの実装

3.3 節で述べたモデルの詳細を適用した概念図を図 7 に示す．基本的にはこのモデルに沿って実装しているが，実装の都合上一部で変更点が生じている．図 8 に 1 段階での投機システムの流れを示す．このとき，ある時刻での剛性マトリクスを K_i とし，投機を行うノード数 p が 2，予測する K_{i+s} の s は 1 である．図 8 において，予測すべき係数行列の情報を $pattern$ と表現している．$pattern$ は予測すべき未来の係数行列 $\Delta K_{i+s}, a(a = 0, 1...p-1)$ を生成し

それぞれのノードに送信する方法や，$\Delta K_{i+s}, a$ の生成規則をあらかじめ用意しておき，予測のためのパラメータだけを送信し，投機計算ノード側で $\Delta K_{i+s}, a$ を生成する方法など，様々な手法が考えられる．今回の実装では予測のためのパラメータだけを送信するようにした．ある時点での剛性マトリクス K_y を改め，K_{base} と表現する．このとき，K_{base} を K_i に更新するためには，ΔK_{base+1} から K_i までの差分行列を必要とし，この差分行列を足しあわせた行列を ΔK_{patch} とする．また，$result$ は反復回数を表す．

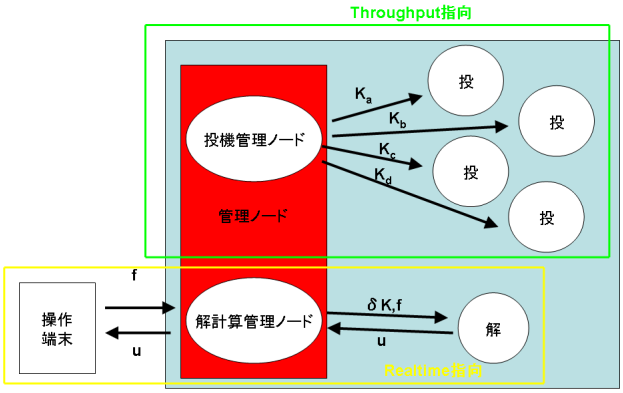

図 7 投機実行モデルの構成

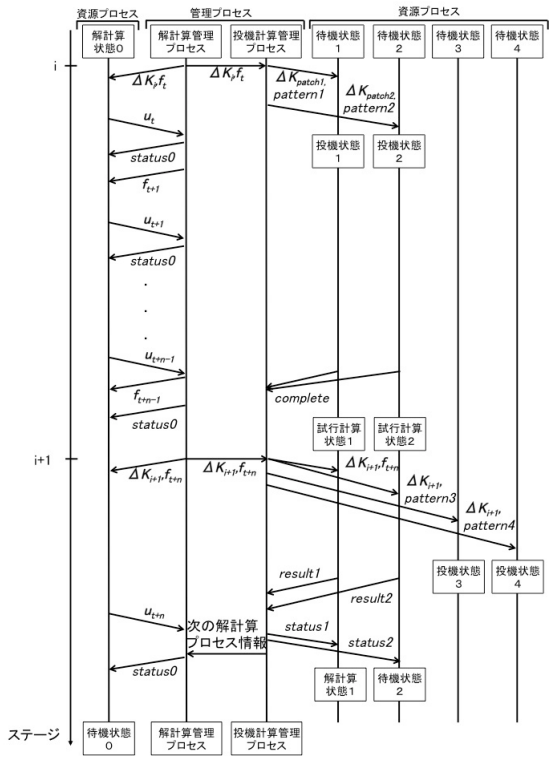

図 8 1 段階投機で 2 パターン予測した場合

4.1 管理ノードの実装

4.1.1 解計算管理プロセス

図 9 では解計算管理プロセスが他のプロセスとの通信処理とその順番について示し，図 10 で解計算管理プロセスの処理フローについて示す．

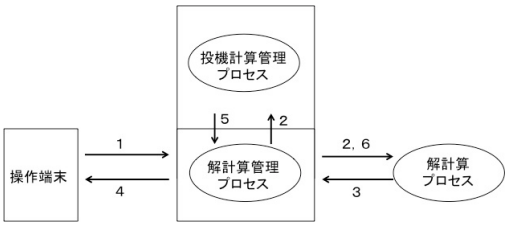

図 9 解計算管理プロセスの通信処理

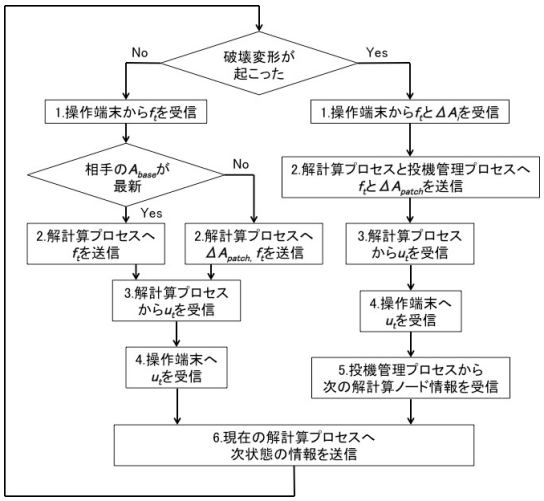

図 10 解計算管理プロセスの処理フロー

1. 操作端末から f_t と ΔK_i を受け取る．
   破壊変形が起こらなかった場合は ΔK_i の受信を行わない．
   - 破壊変形が起こった場合

   2.a. 解計算プロセスに f_t と ΔK_{patch} を送信する．
   投機計算管理プロセスに f_t と $varDeltaK_i$ を送信する．投機計算管理プロセスとの通信は MPI の Sendrecv を用いるが同一ノード内のため通信時間は無視できる程度のものである．
   - 破壊変形が起こらなかった場合
     - 解計算プロセスの A_{base} が最新でない場合
       2.b. 解計算プロセスに f_t と ΔK_{patch} を送信する．
       解計算プロセスが交代して新しい解計算プロセスと通信する際にこの状況が発生する．
     - 解計算プロセスの A_{base} が最新である場合
       2.c. 解計算プロセスに f_t を送信する．
3. 解計算プロセスから u_t を受信する．
4. 操作端末へ u_t を送信する．
   - 破壊変形が起こった場合

5. 投機計算管理プロセスから次の解計算プロセスの番号と保持している A_{base} のバージョン情報を受信.
6. 現在の解計算プロセスへ次状態の情報 ($status$) を送信．今回の実装では毎ステップこの通信を行うものとし，$status$ の内容は「解計算を続行する」か「待機状態に遷移する」かのどちらかである．

4.1.2 投機計算管理プロセス

投機計算管理プロセスでは，待機プロセス，投機計算プロセス，試行計算プロセスをそれぞれ管理し，どのプロセスがどの時点の K_{base} を保持しているかも管理する．また，ステージ毎の ΔK も管理する．前処理行列の導出を行わせるため資源プロセス群に剛性マトリクス K の差分行列 ΔK_{patch} を送信する．今回の実装では，資源プロセス群の待機プロセスから T 台のプロセスを選び投機計算プロセスとする．投機計算プロセスからの試行計算の結果を受け取り，次の解計算プロセスの選出を行う．

図 11 では投機計算管理プロセスが他のプロセスとの通信処理とその順番について示し，図 12 で投機計算管理プロセスの処理フローについて示す．

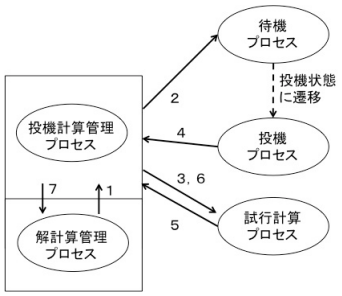

図 11 投機計算管理プロセスの通信処理

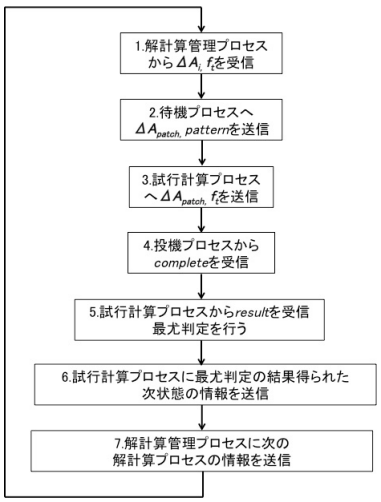

図 12 投機計算管理プロセスの処理フロー

(1) 解計算管理プロセスから剛性マトリクスの差分行列 ΔK_i と力 f_t を受け取る．
(2) 待機プロセスに，ΔK_{patch} と予測すべきパターンの情報 ($pattern$) を送信し，前処理行列の計算をさせる．
(この情報を受け取った時点で待機プロセスは投機プロセスに状態遷移する)
送信先のプロセスが保持している剛性マトリクスの Stage 情報を更新する．
(3) 試行計算プロセスに ΔK_{patch} と f_t を送信し，試行計算を行わせる．
送信先のプロセスが保持している剛性マトリクスの Stage 情報を更新する．
(4) 投機計算プロセスから前処理行列導出完了のメッセージ ($complete$) を受信．
(5) 試行計算プロセスから試行結果の反復回数 ($result$) を受け取る．受け取った反復回数をもとに次の解計算プロセスを決める．
現在の実装では，3.3.3 節でも述べたように，投機予測の成否判定として反復回数の最も少ないものを採用する．
(6) 試行計算プロセスに次の状態遷移のメッセージを送り，リストを更新する．
(7) 次の解計算プロセスの番号と，保持している K_{base} のステージ数を解計算管理プロセスに送る．

ここで，破壊変形が起こった後の形状としていくつかのパターンの剛性マトリクスを予測する必要がある．そこで，p パターンの投機予測を行うと考える．1 回目の投機で p 台の投機計算プロセスに前処理行列の計算を行わせるとすると，i ステージで投機計算を開始したプロセスは $i+s$ ステージ目で投機計算が終了し，$i+s+1$ ステージ目で試行計算を行う．先ほどの図 8 で説明すると $s=1, p=2$ ということになる．

このことから，この投機システムが動作するには解計算プロセスも含めて最低でも $(s+1)\times p+1$ 台の資源プロセスと管理プロセスが 2 プロセスが必要となる．図 8 の例では 5 台の資源プロセスと管理プロセスが 2 つの構成となっている．

4.2 資源プロセスの実装

本システムにおいて，前処理行列の予測や解計算を行うためのプロセスを資源プロセスと呼ぶ．資源プロセスには図 6 のように待機状態，投機計算状態，試行計算状態，解計算状態の 4 つの状態が存在する．以下ではその 4 つの状態についての説明を述べる．

4.2.1 待機状態

資源プロセスのうち，待機状態にあるものを待機プロセスと呼ぶ．図 13 は待機状態の処理フローである．待機プロセスは投機計算管理プロセスからのデータを待ち続ける．データが送られてきたらそれをトリガーとして状態遷移し，投機計算を行っていく．

(1) 投機計算管理プロセスから，$\Delta K_{patch}, pattern$ が送ら

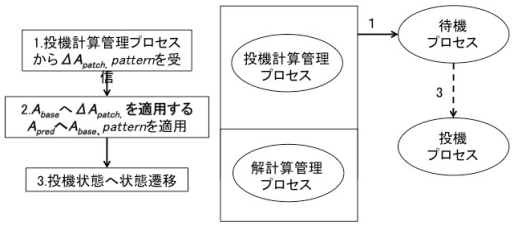

図 13 待機状態プロセスの処理フロー

れてくるのを待つ．

(2) 受信した場合 $K_{base} += \Delta K_{patch}$ とし K_{base} を最新の状態にする．更に $pattern$ をもとに K_{pred} を生成し，予測する係数行列 $K_{pred} += K_{base}$ とする．

(3) 以上の投機計算に必要な準備の後，投機計算状態に遷移する．

4.2.2 投機計算状態

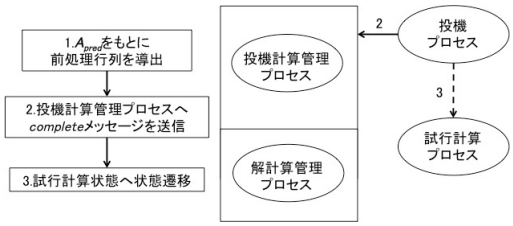

図 14 投機計算状態プロセスの処理フロー

資源プロセスのうち，投機計算状態にあるものを投機計算プロセスと呼ぶ．図 14 は待機状態の処理フローである．投機計算プロセスでは，投機計算を行うために必要な情報のやりとりを投機計算管理プロセスと行う．そして，投機計算として前処理行列の計算を行う．

(1) 待機状態時に生成した K_{pred} をもとに前処理行列 P を導出する．

(2) 前処理行列の導出が完了したこと (complete) を投機計算管理プロセスに送信する．

(3) 試行計算状態に遷移する．

4.2.3 試行計算状態

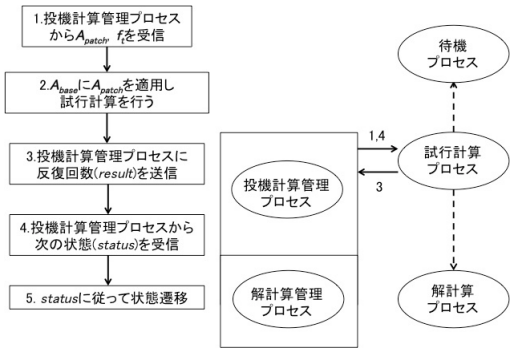

図 15 投機計算状態プロセスの処理フロー

資源プロセスのうち，試行計算状態にあるものを試行計算プロセスと呼ぶ．図 15 は試行計算状態の処理フローである．試行計算プロセスでは，試行計算を行うために必要な情報のやりとりを投機計算管理プロセスと行う．そして，投機状態時に導出した前処理行列 P を用いて試行計算を行う．

(1) 投機計算管理プロセスから最新の ΔK_{patch} を受信する．

(2) $K_{base} += \Delta K_{patch}$ とし，K_{base} と P を用いて反復解法を行う

(3) 求解までにかかった反復回数 ($result$) を投機計算管理プロセスに送信する．

(4) 投機計算管理プロセスから次に遷移するべき状態の情報 ($status$) を受信する．

(5) status に応じて解計算プロセスか待機プロセスに遷移する．

4.2.4 解計算状態

資源プロセスのうち，解計算状態にあるものを解計算

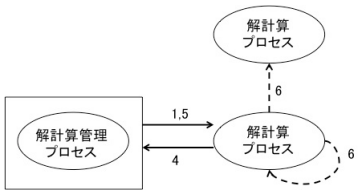

図 16 投機計算状態プロセスの処理フロー

プロセスと呼ぶ．図 16 は解計算状態の処理フローである．解計算プロセスでは，解計算を行うために必要な情報のやりとりを解計算管理プロセスと行う．

(1) 解計算管理プロセスから最新の f_t と ΔK_{patch} の情報を受信する．
破壊変形が起こった場合のみ ΔK の受信を行う．

(2) $K_{base} += \Delta K$ とし K_{base} を最新の状態に更新．
この作業は ΔK を受信した場合のみ行う．

(3) K_{base} と P，f_t をもとに ICCG 法を行い，u_t を導出．

(4) u_t を解計算管理プロセスへ送信する．

(5) 次状態の情報 (status) を受信する．

(6) status に応じて状態遷移を行う．

5. 評価

計測には肝臓を有限要素モデル化したものを係数行列として用いた．行列のサイズは 3759×3759 で，非零要素数の割合は約 1.0[%] である．ある頂点 1 つに注目し，隣接した頂点との連結を強くするような変化（肝組織の硬化に相当）を加え実験を行った．実験の実行環境は CPU:Core2Duo 2.66[GHz],L2Cache:4[MB],Memory:2[GB],OS:Linux 2.6.19-1.2895.fc6 を 8 ノード用意し，Gigabit Ethernet で結合した．本システムを使用した場合と，本システムを使わず操作端末と解計算プロセスが 1:1 で通信を行う場合の，ΔA_i の非零要素数が 169 の時のデータを表 1 に示す．本システムの手法では 1 ステージに前処理行列導出が完

了するとし，1 ステージ内で 2 パターンの投機予測を行ったものである．ただし，今回の実験では複雑な予測計算は行っておらず，$pattern$ として A_i から前処理行列を導出するか，前処理行列として単位行列を用いるかの 2 パターンを用意し実験を行った．表 1 の応答時間については，操作端末が b を送信終了した直後から x を受信終了直後までの時間で，操作端末側で計測を行った．

表 1 実験結果

	本システム	逐次処理
前処理行列導出 [ms]	211	203
反復計算 [ms]	137	135
応答時間 [ms]	143	342

6. 大規模並列処理環境への対応

本フレームワークでは計算や通信の最適化がされておらず，フレームワーク内で用いている計算の並列化や投機パターンの増加などにより大規模並列処理環境へ適用した場合，投機計算管理プロセスでの通信時間が増大し，前処理行列の導出が大幅に遅くなる可能性がある．そのため，本章では通信時間の減少を試みる．ギガビットイーサ環境における差分行列配信時の通信遅延状況を表 2 に示す．なお，差分行列は CRS 形式で圧縮されており，測定した時点での差分要素数は 106 である．表 2 を参考に計算を行うと，投機ノード数が 1024 程度の環境においては，通信遅延はおおよそ 10[msec] となる．投機ノード数が 2 の時の投機管理プロセス全体の所要時間がおおよそ 140msec 程度であるから，投機ノード数が 1024 の場合では，投機計算管理プロセス全体で 150[msec] となり，通信遅延はそのうちの約 7%を占める．さらに投機ノード数を増加させた場合，もしくは差分要素数が増大した場合，通信遅延が増大することが見込まれる．そこで，この通信遅延を軽減する方法を考える．

表 2 ノード数による通信遅延 [msec]

投機ノード数	所要時間
2	0.036
4	0.036
8	0.075
16	0.149
32	0.291
64	0.602

6.1 投機計算管理サブプロセスの導入

現在のアルゴリズムでは，資源ノードの総数を n 台とすると，投機計算管理プロセスが n 台すべてにデータを送信している．この状態では，$O(n)$ で通信時間が増加してしまう．そこで，この通信時間を減少させるために，既存の投機計算管理プロセスと平行してデータを送信する投機計算管理サブプロセスの導入を行う．投機計算管理サブプロセスは 1 台だけではなく，複数台での運用を想定している．通信時のアルゴリズムとしては，

(1) 投機計算管理プロセスは，投機計算管理サブプロセスが送信すべき宛先の情報として，ビットマップデータを生成し，送信する
(2) 投機計算管理サブプロセスは，受信したビットマップデータから送信先を抽出し，データを送信する．
(3) 投機計算ノードは，あらかじめ MPI 関数である Irecv 命令を投機計算管理プロセスおよびサブプロセスの数だけ発行しておき，どのプロセスからでも受信可能な状態とする

このアルゴリズムにより，投機計算管理プロセスと投機計算管理サブプロセス間で通信が行われるデータ量は $n$bit である．現段階では実装中のため，データを示すことができないが，このアルゴリズムにより投機ノードの台数 n が 10000 程度になったとしても，投機計算管理サブプロセスを複数用いることで，投機計算管理プロセスの所要時間に占める通信遅延の割合を減少することが可能であると考える．

7. まとめ

本論文では，前処理行列を投機的に計算することで実時間性を保証する投機システムのモデルおよび実装について述べた．簡単な評価を行った結果，投機計算により前処理行列導出時間が隠蔽されていることが確認できた．さらに，反復計算時の管理ノードのオーバーヘッドも十分小さいことが確認できた．大規模並列処理環境への適用を目指して，通信アルゴリズムの改良を提案した．今後，実装並びに評価を行っていく．

謝辞

本研究の一部は,JSPS 科研 25280042 ならびに 25540043 助成による。本研究の実施にあたり、日頃より計算機環境の維持管理に貢献いただいている松山幸雄技術職員、ならびに、活発な議論を通して貴重な御意見を頂いている森・福間研究室の諸氏に心より感謝いたします。

参考文献

[1] Nakao,M.：*Cardiac Surgery Simulation with Active Interaction and Adaptive Physics-Based Modeling*，PhD Thesis，京都大学大学院情報学研究科 (2003).
[2] Kuroda,Y.：*A Study on Virtual Reality based Palpation Simulator*，PhD Thesis，京都大学大学院情報学研究科 (2005).
[3] 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).
[4] 野田裕介：共役勾配法による手術シミュレータ高速化の予備評価，情報処理学会関西支部大会，(2006).
[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，(1989).
[6] 渡辺隆史，大谷淳， 棚沢順，徳永幸生：2 段階境界要素法を用いる三次元弾性物体の変形と移動の実時間アニメーション法，電子情報通信学会論文誌，(2005).
[7] Kume, N.: *Distributed Massive Simulation for Haptic Virtual Reality Based Surgical Skill Transfer*, PhD Thesis, 京都大学大学院情報学研究科 (2006).
[8] 中尾恵，河本敏孝，湊小太郎：柔らかい物体に対する切開の実時間シミュレーション方法，Visual Computing グラフィックスと CAD 合同シンポジウム 07，(2007).
[9] 野田裕介：手術シミュレータにおける並列化及び操作の連続性を考慮した高速化，京都大学大学院情報学研究化，修士論文 (2008).
[10] 依藤逸，野田裕介，粂直人，嶋田創，中尾恵，森眞一郎，中島浩，富田眞治：操作の連続性を考慮した投機計算を利用するインタラクティブシミュレーション，先進的計算基盤シンポジウム SACSIC2009 (2009).
[11] 岩永翔太郎，松井祐太，福間慎治，森眞一郎：インタラクティブシミュレーションのための投機計算システムの実装，先進的計算基盤シンポジウム SACSIS2012 (2012).