Vol.2013-HPC-138 No.1
2013/2/21

大規模反復型数値計算過程の視覚的解析支援ツールの開発

花房 秀光[1,a)]　小寺 晋[1]　木内 俊介[1]　山崎 勇輔[1]　福間 慎治[1]　森 眞一郎[1]

概要：数値シミュレーションで多用される反復型数値計算の高速化では, いかに少ない反復回数で十分な精度の近似解が得られるかが重要である. しかしながら, その計算手法の高速化を図るには数値解析の専門家以外では困難となっているが, シミュレーションを行うユーザーは一般に解こうとしている問題の性質に関する事前知識を持っており, 計算過程からこの事前知識を上手く活用できれば, シミュレーションの高速化に関する示唆が得られる可能性がある. 本論文では今までユーザーに提示されることが少なかった反復計算の収束過程に注目し, 残差分布を可視化してユーザーに提示することで, 計算過程の解析を支援するツールについて報告を行う.

キーワード：可視化, 反復型数値計算

1. はじめに

数値シミュレーションで多用される反復型数値計算の高速化では, いかに少ない反復回数で十分な精度の近似解が得られるかが重要である. そのためには計算手法の高速化が重要になってくる. しかしながら, その計算手法の高速化を図るのは数値解析の専門家以外では困難だと考えられる. また, 反復型数値計算手法としてよく使われる共役勾配法や GMRES 法の収束過程の残差ノルムのグラフから何故そのような収束結果が得られたかは専門家以外わからない.

しかし, シミュレーションを行っているユーザーは一般に解こうとしている問題の性質に関する事前知識を持っており, この事前知識を上手く活用できれば, シミュレーションの高速化に関する示唆が得られる可能性がある. そこで我々は今までユーザーに提示されることが少なかった反復計算の収束過程に注目し, 残差分布を可視化して, ユーザーに提示することで, プログラムの解析を支援する手法の検討を行っている. また, より効果的な可視化手法を検討し, 解析支援ツールの構築を目的とする.

以下その結果を報告する.

1 福井大学大学院工学研究科
University of Fukui
†1 現在，福井大学大学院工学研究科
Presently with University of Fukui
a) hanafusa@sylph.fuis.u-fukui.ac.jp

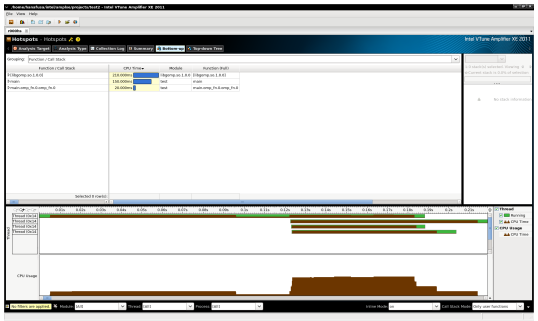

図 1　プログラム最適化ツールの例

2. 研究の背景

2.1 関連研究

プログラムの最適化支援に関しては様々な研究が行われており, 実用化・商用化され数多くのツールが存在する. 例えば高速化支援ツールの一つである Intel Parallel Studio XE[1] の Vtune Amplifier XE で OpenMP を用いた並列プログラムを解析すると, 図 1 のような解析結果が得られ, そこからプログラム中のどこにボトルネックが存在するか分かる. ツールの利用者は提示された情報と自らが持っているプログラムの内容に関する知識を連携させ, プログラムの最適化を行う.

これらのツールの多くはトレースベースの解析ツールであり, プロファイル付きで一度プログラムを実行し, 得られた多くのトレースデータを実行後に解析するためのツールである. 本研究では実行中のプログラムを対話的に解析し, アルゴリズムの最適化を支援するツールを開

ⓒ 2013 Information Processing Society of Japan

発する.

数値計算過程の可視化という事では, 多くの人が流れの可視化などの時系列シミュレーションにおける, シミュレーション手法の可視化 [2][3] を思い浮かべるであろう. これに対して本研究は, 時系列シミュレーションにおける 1 回分の数値計算で行われる反復改善における, 収束特性の可視化に注目し, これを用いてプログラムの実行時間最適化を行う. これが最終目標である.

藤野 [4] らは反復解法の本質的な疑問に対し答えるには残差ノルムのグラフだけでは不十分とし, 収束過程, 物理平面における近似解や残差ベクトルの可視化などを行った. その結果, 反復解法の収束過程の可視化は, 収束特性を調べる上で非常に強力な手段であることを実証し, 近似解や残差ベクトルの可視化は反復解法の性質の理解に役に立つとしている.

前章で述べた通り, 多くのユーザは解くべき問題の性質は把握しているが数値解析の専門家では無い. そこで数値計算過程で起きている現象を把握するために, 収束過程の可視化というアプローチがある. 後藤 [5] らは数値シミュレーションの収束過程を 2 次元の物理空間上に可視化し, シミュレーション対象の要素毎に値の変化に応じて色付けを行っている. これにより収束過程の途中経過を理解することが容易になったと実証している. しかし, その際に色の指定の最適な方法や時間ステップに即した表示の方法が必要であるとしている.

可視化のためにシミュレーションコードに与える影響を軽減する手法として,VisTrace[7] は可視化対象をグローバル変数宣言（あるいは Common 宣言) されたデータに限定することで, シミュレーションコード自体に変更を加えることなく可視化ライブラリをリンクするだけで可視化を可能にする手法を提案している. これに対して RSVLib はプログラム中に可視化のための関数を埋め込むことで可視化対象となるデータを可視化ライブラリに渡す実装を行っている. シミュレーションコードの改編は必要であるが我々も後者の方法を採用している. これらの議論はプログラミング負担の影響を軽減するという観点の議論であるが, 実行時オーバヘッドという観点での議論はほとんど行われていない. これに対し我々は, シミュレーションコードと可視化コードの弱い連携による双方向インタラクションを実現することを目指している.

数値データを可視化する際に重要な要素として, データと色情報を関連づける伝達関数の設定がある. 適切な設定を行わなければ, 数値データに含まれる重要な情報を視覚的に提示できないためである. 竹島 [8] らは数値データに含まれる幾何学的位相情報を用いて特徴点を抽出し可視化時に特徴点が失われないような伝達関数の自動設定に関する研究を行っている. 本来はこのような自動設定のツールも導入すべきであるがツール全体の実時間性を重視し, 今回の実装ではユーザが手動で伝達関数のパラメータ設定を変更可能な実装とした.

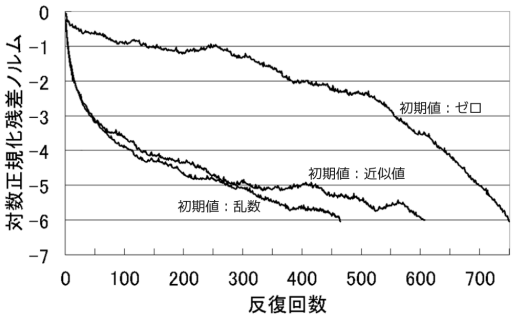

図 2　一般的な収束過程の表現

A

$$\begin{pmatrix} 543210 & \cdots & 000000 \\ 454321 & \cdots & 000000 \\ 345432 & \cdots & 000000 \\ \cdots & \ddots & \cdots \\ 000000 & \cdots & 234543 \\ 000000 & \cdots & 123454 \\ 000000 & \cdots & 012345 \end{pmatrix}$$

図 3　1000 次元の帯行列

2.2　反復型数値計算の収束過程の可視化

一般に良く用いられる正規化残差ノルムの時間変化をグラフ表示したものが図 2 である. このグラフからは反復回数が進むにつれ残差の値が収束に向っていることが読み取れる. しかし, 計算の過程で個々の変数で何が起きてるかは, このグラフだけからでは解析できない. そこで本研究では残差ノルムではなく残差分布に注目し, 反復計算の過程で個々の変数に対する残差の収束過程を可視化する. 具体的には, 残差の絶対値をカラーマップで表現することで可視化を行った.

3.　残差拡散と可視化

3.1　収束過程の可視化

予備実験として, 次元数 1000 の単純な行列 A(図 3) からなる線形方程式を考え $x[i] = 1(0 <= i <= 999)$ とした時の b に対し,b[250] の要素 1 点に変位を加えた場合を共役勾配法で解き, その収束過程を可視化した. 横軸は個々の変数に対応し, 縦軸が時間方向である. 残差情報を色で表現した.

残差情報の色の付け方として残差の絶対値の値によって色の濃淡を変化させて可視化した. その結果を図 4 に示す.

図 4 から変位を与えた 250 番目の要素に対応する X から一旦全体に残差拡散し, そのあと収束が始まっていることが分かる. この色付けの方法によって変数空間上で

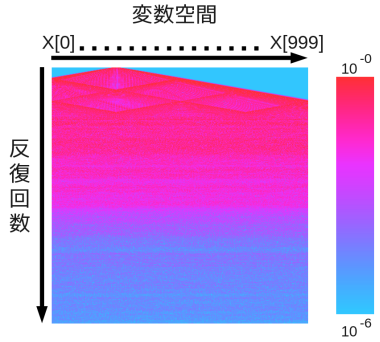

図 4　残差の収束過程 グレースケール

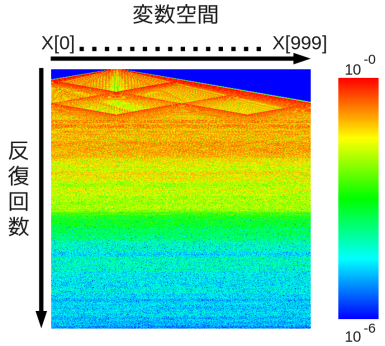

図 5　残差の収束過程 カラー

の残差の収束の様子を可視化することができた.

これは残差の情報を 2 種類の色の濃淡でで表現した. しかし, 色の濃淡だけでは残差の収束時の各変数の様子が分かりにくい場合がある. そこで, 残差の色付けの仕方としてより多くの色を用いることで, より詳しく残差の収束の様子を可視化できるようにした.

図 5 は図 4 と同じデータを色の付け方だけ変更した結果である. 図 5 を見ると, 図 4 では見られなかった周期特性が現れている様子が見て取れる. 残差分布を可視化することで残差ノルムのグラフだけでは分からなかった収束の特性を見つけることができた. なぜこのような性質が現れたのかは現段階では分からないが, 収束過程の解析を進めていく上で次はこの菱形が現れる原因を探るなど次の解析へ進めることができる.

またこの残差分布の可視化でより効果的に情報を提示するために残差の色付けの範囲の最適な範囲を考える必要がある. 例えば, この残差分布図からユーザが指定した範囲の残差ノルムの最大値を赤色とし最小値を青色とするなど, そうすることでより詳細な残差分布の収束の様子を可視化することが出来る. 今回の実装では UI 上から log スケールの値で赤色となる上限の値と青色となる下限の値を指定できるようにした.

4. 一般的な行列問題に対する可視化

前章では簡単な線形方程式に対しての残差分布の可視化を行った. そこで他の一般的な問題に対して可視化を

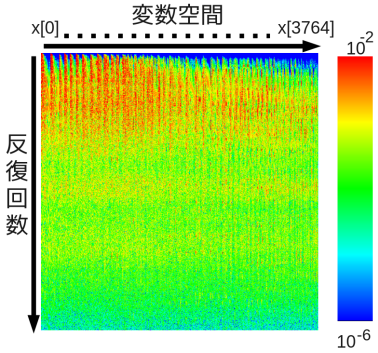

図 6　大動脈データの残差分布表示

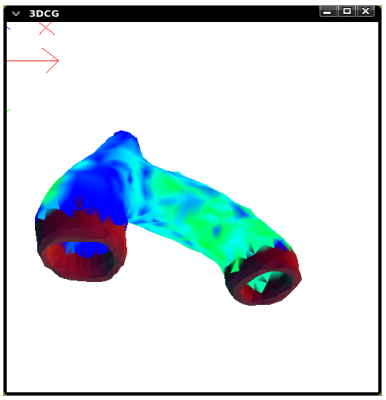

図 7　物理モデルへの対応付け

行い, その有効性を検証した. 今回は次元数 3765 の大動脈データに対して可視化を行った. 図 6 はその結果である. 実データでのシミュレーションでは係数行列の構造も複雑で残差分布上では残差がすぐに分散してしまいここから解析を行うことは難しい. これまでは残差の拡散を変数空間で可視化してきた. ところで残差の拡散には物理的な性質が関係すると考えられる. そこで, シミュレーション対象の物理構造データがある場合は物理構造データを用いて, 物理構造データがない場合はシティマップを用いて残差分布をマッピングし可視化を行った.

4.1 物理構造データを用いた可視化

物理構造データが既知である場合, そこから 3 DCG モデルを構築し, そこへ残差分布を対応付けることで物理空間における残差の拡散の視認を図る. 対応付けの方法として,CG モデルの表面にこれまでと同様に残差の大きさに応じた色付けを行うことで対応つける. 面の色の決定は, 面を構成する 3 点の残差で行う. その様子を図 7 に示す.

4.2 シティマップを用いた可視化

物理構造データが未知である場合, 本研究ではシティマップ [12][13] に基づく可視化手法を実装することにした. シティマップとは反復計算の対象となる方程式にお

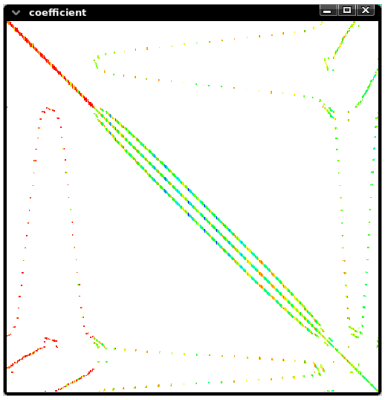

図 8　2 次元残差分布表示

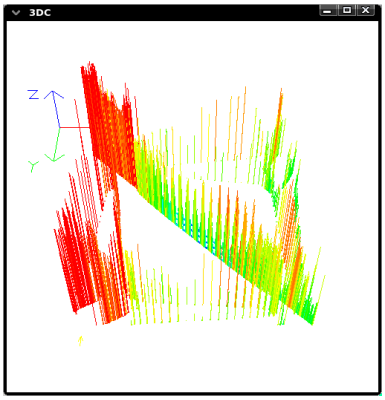

図 9　3 次元残差分布表示

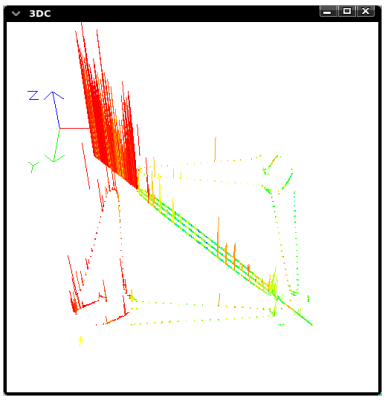

図 10　右辺ベクトルを考慮した残差分布表示

ける係数行列のうち, 非ゼロとなる要素を ”点 ”で表現することで係数行列の概形を一目で把握可能にしたものである.

ここでは残差分布の空間方向, 相互関係に対応するため残差分布を対角要素にマッピングした. しかし, 問題が大規模化するにつれ, マッピングされた点が視認出来なくなる. その対策として, 残差分布を列方向へマッピングした（図 8）. この方法でより見やすくなったが, 構成する要素数が少ないとやはり見落としてしまう. そこで, 立体化による強調を行った.

3 次元表示の際, 高さを残差の値で表示した $Z(i,j) = r(i,j)$（図 9）. また, 右辺ベクトルの大きさを考慮した残差の相対値による残差分布表示も実装した $Z(i,j) = A(i,j)r(i)/b(i)$（図 10）.Z は高さ方向,A は係数行列,r は残差分布,b は右辺ベクトルとする. さらに, 視点を回転させることで 2 次元表示や単なる残差分布のグラフとして見ることも出来る.

4.3　Cone Tree を用いた残差表示

前節ではシティマップに残差分布をマッピングすることにより, 係数行列上での残差分布の収束の様子を観測できるようになった. しかし, このマッピング法では真の解と異なる値を与えたノードから残差がどの隣接ノードを通って残差が広がっていくか, ということは分かりにくい.

そこで, 注目すべきノードを根ノードとした木構造にすることによって隣接関係を重視した残差の広がり方の様子を表示する. この木構造は注目したノードを根ノードとし, そのノードと隣接しているノード（1 次隣接ノード）を根ノードの子ノードとする. 次に 1 次隣接ノードに隣接しているノード（2 次隣接ノード）を 1 次隣接ノードの子ノードとする. このように徐々に次数を上げていき, これを隣接しているノードが全て決まるまで繰り返す. また, 木構造グラフは 3 次元空間に表示し, 子ノードは根ノードを中心とする円周上に配置し, 円錐状に表示する. この木構造に色で表現された残差分布をマッピングする（図 11）.

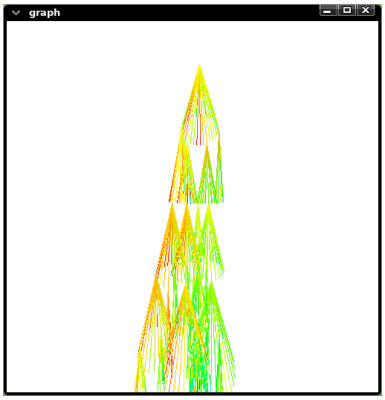

図 11　Cone Tree の残差分布表示

このような円錐状の木構造で表されたグラフを Corn Tree[14] と呼ぶ.Corn Tree は通常の 2 次元の木構造と比べ子ノードが多数存在しても円錐の半径等を調節すればアスペクト比の増大を抑えることが出来るメリットがある.

以上により, 残差が広がっていく過程で, 色に余り変化が無く残差が広がりにくいノードと, 色の変化が大きく残差が広がりやすいノードが分かるようになり, 残差がどの隣接ノードを通って広がっていくかを観測できる.

しかし問題が大規模になるにつれ木構造のグラフを表示することが負荷となり表示に時間が掛かるようになってしまった. そこで, ツールのオプションとしてグラフ表示の描画をシミュレーションの毎タイムステップごとではなくユーザーの任意のタイムステップごとに更新できるようにし, また全隣接関係を表示するのではなくグラフ表示ウィンドウに表示されている範囲だけ表示する機能を実装した.

5. 解析支援システムの構築

ここまでの可視化手法をまとめ解析支援システムを構築した. 本章では構築したシステムの説明を行う.

5.1 可視化システムの構築の要件

前章で説明した可視化手法に加えシステムを構築する上で求められる要件を分析する. 本システムは対象とするシミュレーションを実行しながら解析を行うことを目的としている. 従って, シミュレーションを行うノード, 可視化を行うノードを並列に非同期で動作させることで実現させる. また一般に反復法では, 収束までの反復回数は問題サイズに比例して増加する. そのため, 可視化対象となるデータのサイズは空間方向, 時間方向それぞれの問題サイズに比例して増加する. しかしながら, 画面上に表示可能なピクセル数には限界があるため, 大規模な問題を表示する際には何らかの情報圧縮が必要となる. さらに数値計算に要する時間について考えると, 問題の大規模化に伴い,1 回の反復に要する時間も増大する. そのためシミュレーションを, 可視化のパラメータを変えて何度もシミュレーションを再実行するという試行実験は困難となる.

5.2 システムの特徴

可視化システムの構築の要件より, 時間的・空間的に様々な視点から数値計算の解析支援システムを構築した. システムの主な特徴を以下に示す.

・物理構造データが既知の場合 3DCG における残差分布の様子を任意の視点で表示.
・物理構造データが未知の場合シティマップににおける３次元残差分布の様子を任意の視点で表示.
・タイルドディスプレイを用いた表示.
・以上の残差分布の表示を任意時刻で閲覧可能.
・シミュレーションと可視化の非同期での動作.

またこの他にも対話機能として, 残差の色の範囲を任意の値に変更可能, また残差ノルムグラフも表示し横軸を反復回数と時間で切り替えることができ, 直近のデータを残差ノルムグラフ上に表示することで比較, 解析できるようにした. これらを GUI 制御 [15] し, シミュレーションの解析を行う. その様子を図 12 に示す.

5.3 残差ノルムの可視化

３ DCG, シティマップ,Cone Tree に残差分布を対応付けすることで従来の可視化手法では明かされなかった部分が観測できるようになった. しかし, これらは各要素毎の残差の値を可視化したもので, 全体の残差ノルムはわからない. そこで従来の可視化手法である残差ノルムの表示（図 13）も同時に行う. これらの可視化手法を併用することによって高速化への解析を行っていく. また, 通常は横軸は反復回数だが, 今回は横軸時間とした残差ノルム表示も実装した. リスタート GMRES 法では, 一反復毎に計算量が増加するので計算時間も増えある程度反復したらリスタートを行う. しかし, 一般にはリスタートはなるべく行わない方が早く収束する. そこで, 時間のグラフを見ることで反復計算を継続するか, このタイミングでリスタート行うか, などといったリスタートのタイミングの解析を行うことを目的としている. リスタート GMRES 法の残差ノルムのグラフで図 13 は横軸を時間とし 1 反復毎に紫の縦線でグリッド表示し, リスタートしたタイミングで緑の縦線を表示している. この図から反復回数が増加するにつれ計算時間が増加し, リスタートしたタイミングで計算時間がまた短くなっていることがわかる.

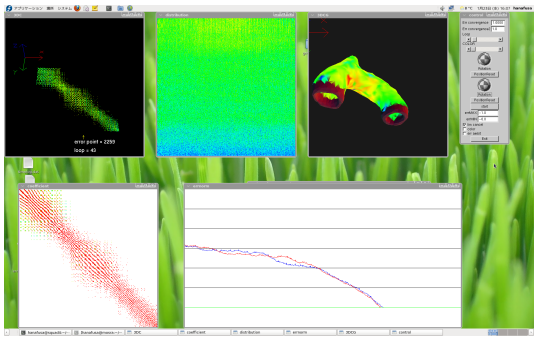

図 12 実行の様子

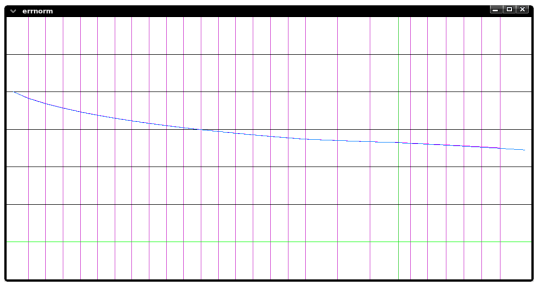

図 13 横軸時間軸での残差ノルムの表示

また, この残差ノルムのデータを保存し, 次回のシミュレーションの際に直近のデータとして表示し比較することが出来る. これは時系列シミュレーションにおいて, 連続した数値計算で行われる反復計算の収束特性は似かよると考えられる. そこで直近のデータと比較することで解析の手助けを行う機能である.

5.4 時系列プレイバック機能

シミュレーション終了後に残差分布の収束過程を詳しく解析できるように, 収束過程のプレイバック機能を実装した. これはシミュレーション中の各ノードの残差の

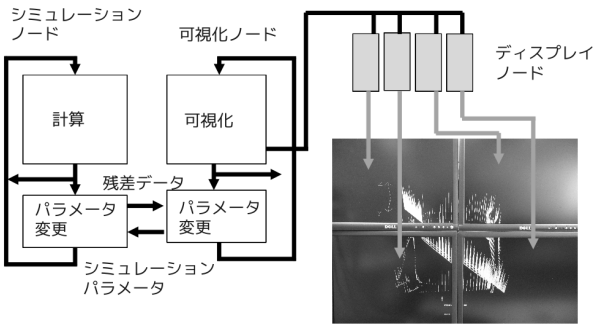

図 14　ツールを使用した場合のデータの流れ

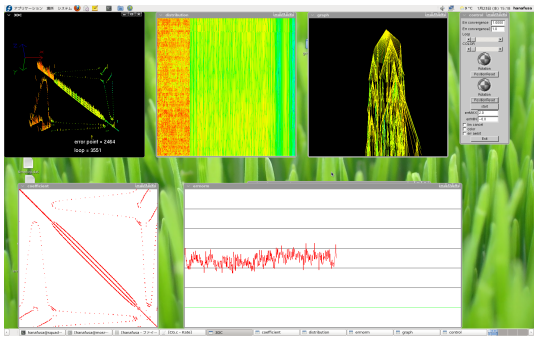

図 15　ツールの描画

値をファイルに記録しておくことで, シミュレーションの終了後にスクロールバーを用いて任意の時系列の残差分布の様子を観測できる. また, 残差分布の収束過程と残差ノルムのグラフはタイムスケールを任意に変更することができ,1 タイムステップ毎の解析だけでなく, 全体の大まかな収束過程の傾向を見ることも出来る.

5.5　タイルドディスプレイを用いた高解像度解析

通常, ディスプレイの画素数には限界があり, 問題が大規模になるにつれ 1pixel に圧縮される情報量が増えてしまう. その結果, 必要な情報が隠れてしまう可能性がある. そこで, タイルドディスプレイを用い使用できる画素数を増やすことで,1pixel に含まれる情報量を減らし, 情報の見落としを防ぐことが出来るようにした.

タイルドディスプレイとは複数のディスプレイを組み合わせて一つの大きなディスプレイとして使用する技術で, アプリケーションのソースコードに SAGE[16] の API コードを加えることでタイルドディスプレイクラスタに表示させたい対象のビットマップ, またはテクスチャを送信しディスプレイに表示する. 本環境では, 可視化ノードからタイルドディスプレイクラスタにシティマップを用いた残差分布表現もしくは Cone Tree を用いた残差分布表現の描画された結果を送信することでタイルドディスプレイを用いて結果の表示を行っている（図 14）.

6.　支援ツールの対話性改善

6.1　ツールの描画速度

シミュレーションの問題規模が大規模になるにつれツールで処理する情報量が飛躍的に増加するため,1 タイムステップ毎の描画に時間が掛かる. 図 15 はシティマップ, シティマップを用いた残差分布の可視化, 残差分布の収束過程,Cone Tree を用いた残差分布の可視化, 残差ノルムのグラフを表示した例である. この状態でシミュレーションと提案する可視化ツールを表 1 で示す同一端末内で両方起動した場合の 1 タイムステップ毎の描画時間を表 3 に示す. また使用した行列のデータは表 2 に示す.

表 3 の結果より, 次元数, 非ゼロ要素数が増加するにつれシミュレーション 1 タイムステップの結果を描画する時間が増加していることがわかる. 特に EC1 では一回の描画に 0.6 秒ほどかかっている. しかし, これは 100 万ノードの Corn Tree のグラフを全て描画した場合である. 前節で説明した任意のノードだけ表示する機能を使用し, 表示ウィンドウ内で見ることができる第 6 隣接まで表示し残りを表示しないことで, 描画するノードは 85 ノードでよく,1 回の表示に掛かる時間を 0.445 秒まで改善することができた.

6.2　対話機能の実装

大規模な問題をシミュレーションすると,1 反復の計算時間も増加する. そこで, シミュレーションの計算をするノードと可視化をするノードを分離し, 並列処理を用いてそれぞれ独立して処理を行う. シミュレーションを行

表 1　ツール使用環境

CPU	Intel Core i7 860 2.80GHz
OS	Linux 2.6.32
Memory	2.9GB
Gfx Card	GeForce9800GT
Gfx Memory	512MB
コンパイルオプション	-O3

表 2　評価に使用した行列

行列名	次元数	非ゼロ要素数
bcsstk09	1083	9760
bcsstk24	3562	81736
EC1	1000000	2998000
(第 6 隣接まで)	(85)	-

表 3　タイムステップの処理時間

行列名	描画時間 [ms] (CornTree)	1 反復の計算時間 [ms]
bcsstk09	18(0.3)	6.1
bcsstk24	30(0.8)	7.2
EC1	636(193)	38
(第 6 隣接まで)	326(3.2)	38

ⓒ 2013 Information Processing Society of Japan

$$A = \begin{pmatrix} 1 & -0.5 & 0 & \cdots & 0 & 0 & 0 \\ -0.5 & 1 & -0.5 & \cdots & 0 & 0 & 0 \\ 0 & -0.5 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & -0.5 & 0 \\ 0 & 0 & 0 & \cdots & -0.5 & 1 & -0.5 \\ 0 & 0 & 0 & \cdots & 0 & -0.5 & 1 \end{pmatrix}$$

図 **16**　1000 次元の行列

う計算ノードでは 1 タイムステップごとの残差のデータを可視化ノードに送信し, 可視化ノードでは残差データから可視化を行い, ユーザからパラメータ変更などの操作があれば計算ノードと通信してシミュレーションに反映させる. そうすることで,1 度のシミュレーション中に様々な観点からの対話的な解析を可能にし, 高速化の支援に繋げる事ができる（図 14）.

またこれらの通信を非同期で実装した. 非同期で実装する事で, 計算速度か描画速度どちらが上回っていても遅延無く処理を行う事が出来る. 例えば表 3 の EC1 のツール全体の描画時間と 1 反復の計算時間を比較する. するとツールの描画時間は 636[ms] に対し, 計算時間は 38[ms] である. もし同期通信で実装していたら, 計算の方に通信の待ち時間が発生し遅延が発生するが, 非同期で実装したことによりその遅延を回避した.

また, 計算ノードと可視化ノードを分けたことによりシミュレーションの計算プログラムを変更するだけで簡単に様々な反復法のシミュレーションに対して可視化ツールを用いることができ, さらに前回のシミュレーション結果をガイドラインとして表示し, その結果と現在のシミュレーションの結果を比較してユーザが操作するといった機能も実装している.

7. ツールを用いた解析

7.1 収束過程の解析

次の 1000 次元の行列を用いた方程式を反復解放で解くことに対してツールを用いて解析を行った.x の値を $(i-500)^3/(1000)^3$ としたものを真値とし,b の 250,251,252 の 3 点に 10 %の変位を加え共役勾配法で解いた場合の収束過程をツールを用いて解析した.

図 17 はその結果である.b に変位を与えた場合残差は一度全体に拡散した後収束している. また収束過程に現れた特徴として菱形のような周期性が見られた. この菱形の特徴をより詳しく調べるために, 次は残差の拡散する範囲の制限を行った. 具体的には x[150]〜x[500] 以外の x の補正を行わないで反復計算を行った.

図 18 を見ると, 残差が制限された領域 (x[150]) に到達すると反対側に広がっていた残差の先端から残差の値が

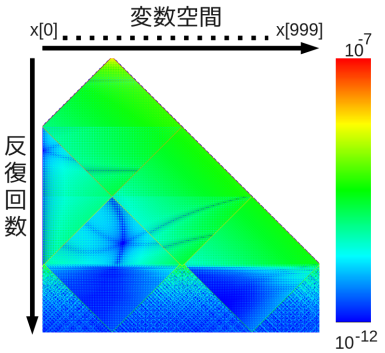

図 **17**　残差の収束過程

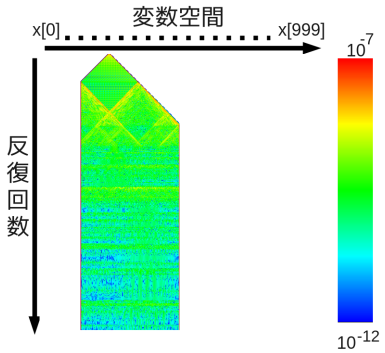

図 **18**　残差拡散の抑制

戻ってきている. 残差の拡散範囲を狭めたことにより菱形の周期が短くなっており, 残差が端まで広がり反射した際になんらかの影響がでていると考えられる. ただし, 残差の拡散範囲外には範囲内の x の値を修正したことによる残差が生じるが, 範囲外の x は修正しないため残差の拡散制限を解除するまで収束しないことがわかった.

図 17 ではこの反射した際に特に大きく残差ノルムは減少しており, この原因が分かればそこから高速化のアルゴリズムが得られるかもしれない.

7.2 残差拡散制限

前節で行った残差拡散制限法を用いて高速化を達成した例と上手くいかなかった例を示す.

3 章の実験で用いた線形方程式 (図 3) に対してまず $x[i] = 1(0 <= i <= 999)$ とした時の b に対し ,$x[250]$ の要素 1 点に変位を加えたものを初期値として共役勾配法で解いた. この収束過程は図 19 のようになった. 残差は一度全体に拡散した後に収束しており, 全体に残差が広がることは不必要であると考えた. したがってこのような不必要な残差の拡散が高速化の妨げとなっている可能性が考えられる.

そこで収束条件 10^{-6} に対し残差が 10^{-5} となるまで要素番号 0 から 500 の変数以外に対して, 反復解法における解探索方向に残差が拡散しないように制約を加え反復改善を行った. 図 20 はその収束過程を示したものである.

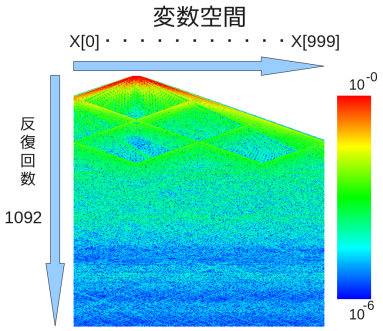

図 19　X に変位を与えた場合の残差の収束過程

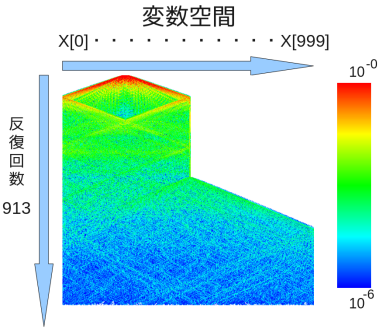

図 20　X に変位を与えた場合の残差拡散の抑制

図 19 と比較すると,x に変位を与えた場合は残差の拡散を抑えることで高速化が達成できていることがわかる.

x に変位を与えた場合に同様の制限法を実際の大動脈弓データに対して適用する. 制限方法としては, 強制的な変形が加えられた場所（ノード）に対応する線形マトリクスの非ゼロ要素に対してのみ残差補正を行う方法を試験的に適用した. 表 4 に制限前後の反復回数を示す. 残差拡散制御の結果, どの次元数においても反復回数が大幅に削減され, 期待通りの高速化が達成できた.

次に b[250] の要素に変位を与えた場合を考える (図 21).b に変位を与えた場合に対して,x に変位を与えた場合と同様に残差拡散を制限すると, 制限範囲外に残差が生じてしまい, いつまでも収束しない結果となった (図 22).b に変位を与えた場合 x の値全てが変わらなければ真値にならないため, 残差拡散制限で残差の変化を制限するといつまでも収束しないと考えられる. しかし, 一概に残差拡散を制限すると必ず遅くなるとは言えず, 制限を掛ける範囲や制限を解除するタイミングなどによっては通常より早く収束するかもしれない. そこでツールから残差拡散の制限範囲を選択でき, またユーザの好きな

表 4　残差拡散制御の効果

次元数 N	最適化なし 反復回数	最適化あり 反復回数
438	284	88
1836	452	49
3765	607	71

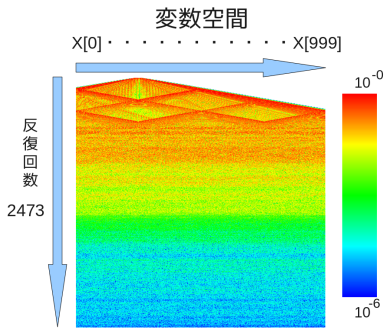

図 21　b に変位を与えた場合の残差の収束過程

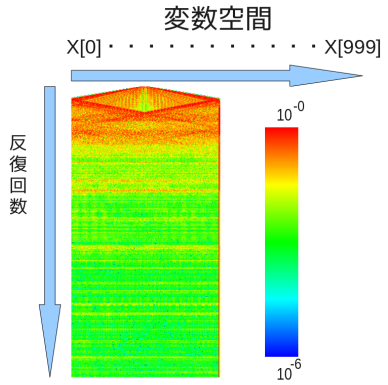

図 22　b に変位を与えた場合の残差拡散の抑制

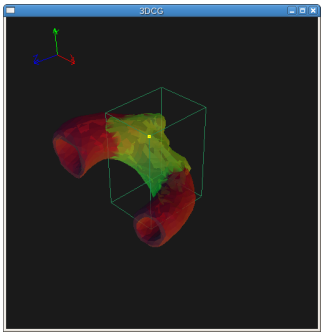

図 23　物理構造データ上からの範囲制限

タイミングで制限を解除できるようにした.

図 23 のように物理構造データがある場合は 3 DCG 上から, 無い場合は図 24 のようにシティマップ上から指定でき, 選択した範囲は黄緑色になる. これは例えば 3 D シミュレーションなどでは構造データを 3 次元空間に固定する必要があり, その部分が収束に影響を与えてしまう場合がある. その際に固定している付近を避けて範囲を指定する, といった使い方もできる.

7.3　反復解法毎の比較

計算ノードと描画ノードを分けたことにより, 計算ノードで行うシミュレーションを変更するだけで容易にさまざまなシミュレーションに対して可視化を行うことができる. 図 3 の線形方程式を b の 250 番目の要素に変位を与えた場合を解く際に共役勾配法で解く場合と GMRES

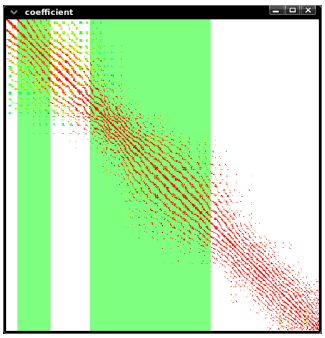

図 24　シティマップ上からの範囲制限

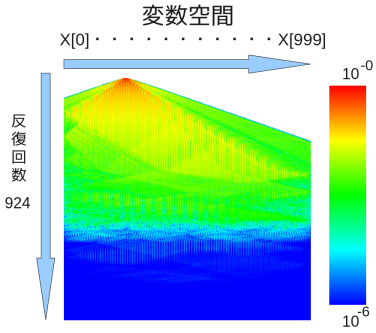

図 25　GMRES 法での残差分布

法で解く場合とで比較した．図 25 は GMRES 法で解いた際に残差分布を可視化したものである．このように，さまざまな反復法の収束特性を容易に比較できるので，ぜひ解析に役立てていただきたい．

8. まとめ

インタラクティブシミュレーションの解析支援を目的として，反復計算における収束過程の残差分布の可視化を行うツールを開発した．物理構造データが既知の場合は物理モデルへの対応付けを行い，物理空間における残差の拡散も観測できるようになった．物理構造データが未知の場合はシティマップを用いた残差分布の 3 次元可視化，右辺ベクトルを考慮した相対値表示により残差分布の空間の広がり方が分かるようになり，Cone Tree を用いた残差分布の可視化で残差分布の広がる経路が分かるようになった．このシティマップと Cone Tree を組み合わせることによってユーザーにとってより有意義な情報を提示することができる．これらを統合したシステムを構築し，様々な検証が出きるようになった．

今後の課題として，ツールを用いた新たな高速化アルゴリズム開発事例の開拓，またユーザインタフェースの向上などが必要である．

謝辞

本研究の実施にあたり，京都大学付属病院医療情報部にデータを提供していただいた．ここに深く感謝の意を表します．

本研究は一部 Jsps 科研費基盤 (c)22500044 および萌芽 23650052 の助成による．

参考文献

[1] IntelParallelStudioXE http://www.xlsoft.co.jp/jp /products/intel/studio __ xe/index.html/
[2] RVSLIB(RealTimeVisualSimulationLibrary),http://www.nec.co.jp/ solution/hpc/library/rvslib/index.html
[3] リアルタイム可視化用ソフトウェア Vis-Link,http://jp.fujitsu.com/solutions/hpc/app/vislink/
[4] 藤野清次, 張紹良, 森正武,Bi-CG STAB 法の収束過程の可視化, 数理解析研究所講究録,vol.0791(19920600),pp.16-30
[5] 後藤和一, 梅谷征雄, 数値シミュレーション過程の可視化トレース, 第 5 回問題解決環境ワークショップ論文集,pp,48-51,2002
[6] 宮地英生,PC クラスタにおける可視化技術の動向とその事例,PCCC ワークショップ資料 2008/3/7.
[7] 藤井孝蔵, 小笠温滋, 中島拓之, 田村善昭, 森屋光弘,VisTrace-リアルタイム大規模データの虫眼鏡
[8] 竹島由里子, 高橋成雄, 藤代一成, 位相属性を用いた多次元伝達関数設計
[9] 山崎勇輔, 森眞一郎：収束過程の可視化によるアルゴリズム設計支援 –インタラクティブシミュレーションの高速化–, 第 35 回 可視化情報シンポジウム講演論文集, pp.65–66, July 2007.
[10] 小寺　晋. 共役勾配法における収束過程の可視化による高速化支援システムの検討. 福井大学情報・メディア工学専攻修士論文.2008
[11] 木内俊介. 反復型数値計算における収束過程の可視化による高速化支援システムの改善. 福井大学大学院工学研究科情報・メディア工学専攻修士論文.2009
[12] MATRIXMARKET http://math.nist.gov/MatrixMarket/
[13] FloridaMatrix http://www.cise.ufl.edu/research /sparse/matrices/
[14] G.G.Robertson,J.D.Mackinlay and S.K.Card Cone Trees:Animated 3D Visualizations of Hierachical Information. Reaching through technology (1991), pp. 189-194.
[15] GLUI http://www.cs.unc.edu/ rademach/glui/
[16] SAGE http://www.sagecommons.org/

ⓒ 2013 Information Processing Society of Japan