大規模非定常数値シミュレーションのリアルタイム可視化 ——並列計算サーバによる可視化方式の実用化に向けて

武井利文[†] 松本秀樹[††] 土肥俊[†]

本論文では，並列ベクトル型のスーパコンピュータなどの並列計算サーバの上で実行される大規模な非定常数値シミュレーションの結果を，ネットワーク上の PC などの軽いクライアントを通して可視化することを目的として開発されたリアルタイム可視化システム RVSLIB（Real-time Visual Simulation Library）の概要を，可視化処理性能を含めて紹介する．数百万自由度以上の計算格子を用いた流体解析などの大規模非定常計算においては，従来のポスト処理手法による可視化は困難になる．RVSLIB は並列計算サーバの各プロセッサ上で，流体などの解析と可視化画像生成までのグラフィックス処理を行い，画像を圧縮し，ネットワークを通して端末へ転送する．これによって，計算サーバから PC などのユーザ端末に計算結果を転送する場合に比べて，データ転送量を大幅に削減し，インターネットを含む広域のネットワーク分散環境での可視化を可能にしている．グラフィックス処理はベクトル化と並列化により高速化している．RVSLIB を実際のシミュレーションプログラムに組み込んで NEC SX-4 上で性能評価を行い，その結果，リアルタイム可視化による経過時間の増加率は 1%以内に抑えられることを確認している．

A Real-time Visualization for Large-scale Unsteady Simulations ——Toward a Practical Visualization on Parallel Computation Servers

Toshifumi Takei,[†] Hideki Matsumoto[††] and Shun Doi[†]

We have developed the RVSLIB (Real-time Visual Simulation Library) as a concurrent system for producing visualizations. This paper shows the effectiveness of the system when it is applied to large-scale unsteady numerical simulations, for which more than millions of grid points are used, on high-performance parallel vector supercomputers in a network-computing environment. Conventional post-processing may no longer be applicable to such large-scale simulations. The system concurrently performs almost all of the visualization tasks on a computation server and uses compressed visualized image data for communications between the server and the user terminal. Consequently, the volumes of data transferred over the network are much smaller than the raw data and efficient visualization of large volumes of data in wide-area network environments that include the Internet becomes possible. We have realized several ideas, such as high-speed visualization algorithms for use on parallel vector supercomputers, to make the system practical. The system was applied to an actual CFD code (a simulation of the fluid flow around a baseball) on an NEC SX-4 and we observed, in this case, that the computational time increase due to the concurrent visualization was less than 1%.

1. はじめに

計算機の高速化，数値シミュレーションの高精度化/大規模化とともに，計算結果の可視化技術の重要性が増している．数値シミュレーション結果の可視化といえば，従来はポストプロセッシングを意味していた．すべての結果を計算サーバ上のディスクにいったん出力し，これを端末に転送してから市販のシステムで可視化する．しかし，計算機の性能が向上してくると，シミュレーションの進行状況をライブで可視化して見たいというのは自然な要求である[2),4),6)]．このような可視化手法を，本論文ではリアルタイム可視化と呼ぶことにする．“リアルタイム” という言葉は，“シミュレーションと同時に”，あるいは “シミュレーションの実行に合わせて” 可視化を行うという意味で用いている．可視化のためのパラメータを変更しながら計算進行状況を追尾（トラッキング）したり，シミュレー

† NEC 情報通信メディア研究本部
Computer & Communication Media Research, NEC Corporation
†† NEC 情報システムズ
NEC Informatec Systems, Ltd.

ションプログラムの実行自体を途中で制御（ステアリング）できれば，計算結果評価の効率性は飛躍的に向上することが期待される．

さらに，最新鋭のスーパコンピュータ上で実行される大規模シミュレーションの出力結果は膨大であり，1時刻ステップあたりで数十 MB から数百 MB，シミュレーション全体では数 TB に達することもある[10]．この場合，従来の可視化手法ではいくつかの問題に直面する．第 1 に大規模な計算結果を端末側へネットワーク上を転送しなければならないこと，第 2 に計算サーバ側および端末側に大規模データを格納するためのディスクスペースが必要なこと，第 3 に端末側に大規模データを扱うことができる大容量のメモリスペースが必要なことである．計算サーバの性能が許す限り大規模なシミュレーションを行いたいという要求はつねに存在するため，すでにこれらの問題を克服するための様々な可視化手法が議論されている．1 つには，解析結果を一種の階層構造に格納しておき，可視化したい部分や可視化精度に応じて，必要な一部のデータを転送してポストプロセッシングする方法がある[1),5),9),13)]．この方法は，ネットワーク上を転送し端末側で扱うべきデータ量を削減する点では効果的である．しかし，階層化のために多くの計算時間を必要とするうえ，結局解析結果全体を保存しておかなければならないなど，シミュレーションの大規模化に対する根本的な解決にはなっていない．他の方法としてリアルタイム可視化がある．計算結果を数値として残すかわりに，計算と同時に計算サーバ上で動画まで作成し，これを再生してシミュレーション結果を評価するという方法である．これによって，数値データ自体を保存しておく場合に比べ，1 回のシミュレーションに必要なディスク容量は激減する．たとえば 1 ピクセルあたり 3 B，1600×1200 程度の大きさの高精細フルカラー画像を作成する場合を考えてみる．10 ステップ程度に 1 回画像を作成したとすると，必要なディスク容量はシミュレーション全体でたかだか数 GB 程度である．画像圧縮技術を用いれば，この容量はさらに数十分の 1 に削減される．したがって，結果を十分に保存できない大規模非定常数値シミュレーションにとって，リアルタイム可視化はきわめて重要な技術の 1 つになってくる．

リアルタイム可視化だけでシミュレーション結果を完全に評価できるわけではない．決定的な現象が発生した計算結果などは，ポストプロセッシングにより詳細な可視化を行いたいものである．このような計算結果を計算サーバ側に保存することができたとしても，これを転送して端末側で扱うことができなければ，可視化処理は計算サーバ側で行わざるをえない．

著者らのリアルタイム可視化システム RVSLIB（Real-time Visual Simulation Library）[3),12)]の目的は，以下の 3 つである．

- 大規模非定常数値シミュレーションに対するリアルタイム可視化
- 大規模蓄積データに対するポストプロセッシング
- 動画作成の簡便化

このシステムでは，ほとんどの可視化処理を計算サーバ上で実行し，生成した画像を圧縮し，ネットワークを通して端末へ転送する．これによって，計算サーバから PC などのユーザ端末に計算結果を転送する場合に比べて，データ転送量を大幅に削減し，インターネットを含む広域のネットワーク分散環境での可視化を可能にしている．一方，可視化処理を計算サーバ上で行うことにともなう課題もある．たとえば，可視化処理は計算サーバの CPU リソースを消費するため，その高速化が必要である．

本論文では，RVSLIB がネットワーク分散環境における大規模非定常数値シミュレーション結果の可視化のための実用的な手段を与えることを示す．初めに 2 章では，関連する従来の研究を概観する．3 章では RVSLIB のシステム構成を概観し，画像をベースとした通信方式の利点について述べる．4 章では，本システムを実用的なものにするうえで解決しなければならなかったいくつかの課題について述べる．5 章ではこれらの課題を克服するために，著者らが行ったいくつかの工夫について解説する．6 章では，本システムを実際のシミュレーションに適用して可視化処理性能を評価する．最後に，今後の研究課題について議論する．

2. 関連する研究

ここでは，計算サーバとユーザ端末を，インターネットを含む比較的バンド幅の狭いネットワークで結んだ環境を想定して，これまでの研究を振り返る．これは大規模非定常シミュレーションを行ううえで，最も一般的な環境である．計算サーバは，大規模非定常シミュレーションを実行するスーパコンピュータなどである．ユーザ端末は，比較的仕様の低い PC やワークステーションなどである．

リアルタイム可視化システムとして，すでにいくつかの提案がなされている．たとえば，MIT で開発されている pV3 がある[6)]．このシステムでは，計算サーバ上でグラフィカルオブジェクトを並列処理により生成し，これをクライアント側へ送信する．クライア

ント側では，OpenGL などの汎用グラフィックスライブラリを用いてレンダリング処理を行う．専用のグラフィックスボードを用いれば，高速なレンダリング処理が可能である．また，並列計算サーバ上に分散された計算結果を集めてクライアント側へ送信し，そこで AVS などの市販のシステムを用いて可視化を行うためのライブラリが Oak Ridge 国立研究所で開発されている[4]．部分領域あるいは間引きを行った計算結果の送信や，シミュレーションのステアリングが可能である．しかし次章で見るように，これらのシステムでは，ネットワーク上を転送しユーザ端末側で扱うべきデータの大きさが，シミュレーション規模とともに大きくなるという欠点がある．

レンダリング処理までを計算サーバ上で行うライブラリは，NASA Langley 研究所から報告されている[2]．これは基本的に可視化ライブラリのみの提供であり，システムとして完成されたものになっていない．特に提供されるライブラリは比較的低レベルのものであり，十分なユーザインタフェースも提供されていないなど，実用化のための課題が解決されているとはいえない．

一方，大規模蓄積データ向けのポストプロセッサとして開発が進められているものに，ASCI（Accelerated Strategic Computing Initiative）プロジェクトの，“Terascale Visualization” がある[9]．このシステムでは，大規模データをいかに削減してクライアント側に転送し可視化するかに重点が置かれている．また領域全体の可視化を粗い精度で行った後，ズームアップしてローカル領域を高精度で可視化することもできる．解析データに 3-D および 4-D ウェーブレット変換を施し，階層型のデータ構造にすることでこれを実現している．1 つのデータを様々な視点から可視化する場合は，階層型のデータ構造構築に見合うメリットが得られる．しかし，階層化のために多くの計算時間を必要とするうえ，結局解析結果全体を保存しておかなければならない．したがって，多くの時刻ステップのデータや様々なパラメータセットのシミュレーション結果の可視化を行う場合はあまり効率的ではない．

3. リアルタイム可視化システム RVSLIB

3.1 システム構成

RVSLIB はサーバクライアント型のシステムである．システム構成を**図 1** に示す．

RVSLIB のサーバモジュールは計算サーバ上で動作し，利用者のシミュレーションプログラムからサブルーチンコールの形で呼び出される．シミュレーションの結果が格納されるメモリ領域を直接参照すること

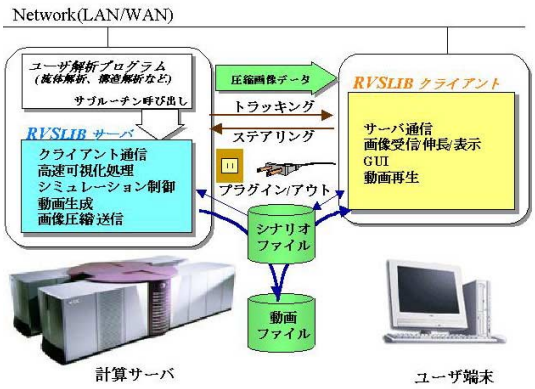


図 1 RVSLIB システム構成
Fig. 1 The system configuration of RVSLIB.

により，高速処理が可能になると同時に，所要メモリ容量節約にもつながる．計算結果をいったんファイルに出力する必要もない．ほとんどの可視化処理はこのサーバモジュールが行い，可視化画像を生成する．可視化処理には特定のグラフィックスライブラリを用いていないため，高いポータビリティを保っている．生成された画像は圧縮され，クライアント側へ送信される．画像圧縮は，前時刻ステップの画像との差分をとり，これに対してランレングスを Huffman 符号化することにより実現している．さらに，JPEG（Joint Photographic Experts Group）による圧縮も可能である．生成された画像は動画ファイルとして，サーバ側またはクライアント側に蓄積することができる．動画ファイルの形式としては，AVI や MPEG2 などがサポートされている．したがって，Adobe Premiere などの市販ソフトを用いれば，生成された動画のノンリニア編集やビデオテープへの録画が容易に行える．

RVSLIB のクライアントモジュールはユーザ端末上で動作し，システム操作のための GUI を表示する．入力された可視化パラメータはサーバ側へ送信され，次の時刻ステップにおける可視化処理で利用される．また一部のパラメータはサーバモジュールを通してシミュレーションプログラムに渡されるため，シミュレーション自体の実行制御に利用できる．シミュレーションプログラムに渡されるパラメータは，利用者自身が自由に設定でき，これに応じた GUI が自動生成される．サーバ側から送信された圧縮画像はクライアント側で復元され，GUI 上に表示される．クライアントモジュールは，Java アプリケーションおよび Java アプレットとして実現されている．**図 2** に GUI の表示例を示す（115 ページ参照）．

システムの動作モードとして以下の 4 つを用意して

おり，これを随時切り替えることができる．

- リアルタイムモード：
  数値シミュレーションと可視化処理の両方が同時に実行される．可視化処理は，毎時刻ステップまたは数時刻ステップに 1 回実行される．
- 可視化表示中断モード：
  可視化処理が中断され，数値シミュレーションだけが実行される．これは，しばらくの間可視化表示が必要ない場合や，数値シミュレーションだけを実行して計算効率を上げたい場合などのために用意されている．適当な時点で可視化処理を再開することができる．
- シミュレーション中断モード：
  数値シミュレーションの実行は一時停止され，可視化処理だけが実行される．これは，いくつかの可視化パラメータの入力を同時に行いたい場合や，特定時刻ステップの結果を様々な角度から可視化したい場合などのために用意されている．適当な時点で数値シミュレーションを再開することができる．
- シミュレーション・可視化表示中断モード：
  数値シミュレーションと可視化処理の両方が一時停止される．適当な時点で，実行を再開することができる．

3.2 処理の分散方式

ここでは，リアルタイム可視化や大規模蓄積データのポストプロセッシングにおいて，解析から画像表示までの一連の処理を，サーバ側（計算サーバ）とクライアント側（ユーザ端末）へ分散させる方法について議論する．分散させるべき処理には，大きく以下の 4 つがある．

- 数値シミュレーション：
  三次元流体解析などの大規模非定常数値シミュレーションを行う．あるいは，大規模蓄積データを読み込む．
- マッピング処理：
  格子点上に定義されたシミュレーション結果から，ポリゴンやポリラインなどの三次元グラフィカルオブジェクトを生成する．
- レンダリング処理：
  グラフィカルオブジェクトに対して，投影変換，陰影処理，隠面処理などを行い，可視化画像を生成する．
- ユーザインタフェース：
  パラメータ入力のための GUI や可視化画像を表示する．

これらのうち，数値シミュレーションは計算サーバ上で，ユーザインタフェースはユーザ端末上でそれぞれ動作するものとすると，マッピング処理およびレンダリング処理の分散のさせかたによって，以下の 3 つのアプローチが考えられる．

- アプローチ A：
  マッピング処理およびレンダリング処理を，ともにクライアント側で行う．サーバ側からクライアント側へ転送されるデータは計算結果である．この場合，計算サーバをシミュレーションに占有できる利点がある．しかし，クライアント側へ全領域の計算結果を毎時刻ステップ送信して可視化することは，ネットワークおよびユーザ端末の性能上困難であり，表示に必要なデータ部分の切り出しを行うための何らかの方策が必要となる．Oak Ridge 国立研究所のシステムや “Terascale Visualization” では，このアプローチを採用している．
- アプローチ B：
  マッピング処理をサーバ側で，レンダリング処理をクライアント側でそれぞれ行う．サーバ側からクライアント側へ転送されるデータはグラフィカルオブジェクトデータである．この場合，マッピング処理に計算サーバの高速演算器を利用できる．また，クライアント側に専用のグラフィックスボードが実装されていれば，レンダリング処理でこれを有効活用できる．しかし，グラフィカルオブジェクトデータの大きさも解析規模とともに大きくなるため，アプローチ A と同様の問題がある．また，グラフィカルオブジェクトを用いないボリュームレンダリングなどでは，このアプローチをとることはできない．MIT のシステムでは，このアプローチを採用している．
- アプローチ C：
  マッピング処理およびレンダリング処理を，ともにサーバ側で行う．サーバ側からクライアント側へ転送されるデータは可視化画像である．可視化画像のデータサイズは，表示ピクセル数と色数のみに依存し，シミュレーション規模や表示方法などには依存せず一定である．さらに JPEG などの画像圧縮技術により，転送データ量を大幅に削減することが可能である．しかし，次章でみるように，この方法にもいくつかの問題がある．NASA Langley 研究所のシステムでは，このアプローチを採用している．

上記のうち，RVSLIB ではアプローチ C を採用した．最大の理由は，ネットワーク上を転送し，ユーザ端末

上で処理すべきデータ量であり，1 方向の格子点数を n とするとき，アプローチ A および B では $O(n^2)$ から $O(n^3)$ であるのに対し，アプローチ C では n に依存しないことである．これは，今後予想されるシミュレーション規模の増大に際して非常に有利である．また，ユーザ端末に要求される仕様が最も低くて済むのも魅力的である．

4. 実用化のための課題

ここでは，RVSLIB 方式の可視化システムを実用的なものにするうえでの主な課題について述べる．

4.1 シミュレーションプログラムへの組み込み

シミュレーションプログラムとの結合を容易かつ効率的なものとするため，サーバモジュールはライブラリ形式で提供されるべきである．このとき利用者は，シミュレーションプログラムにこれらを呼び出す処理を追加する必要がある．これにともなうプログラムの修正や，利用者が新たに作成しなければならないプログラムの量は最低限でなければならない．したがってここでは，シミュレーションプログラムのデータ形式との親和性を考慮した高レベルの可視化ライブラリの設計が重要な課題となる．

4.2 計算サーバの有効活用

計算サーバの CPU リソースは，基本的に数値シミュレーションに割り当てられなければならない．マッピング処理およびレンダリング処理を計算サーバ上で行う場合は，これに必要な CPU 時間が，シミュレーション自体に必要な CPU 時間に比較して小さくなければならない．計算サーバ上には，通常グラフィックスボードのような専用のハードウェアが付属していないため，これらはソフトウェアで行わなければならない．したがって，これを効率的に行う可視化アルゴリズムの開発が課題である．

4.3 長時間シミュレーションへの適用

大規模シミュレーションの実行には，通常多くの経過時間を要する．最新鋭のスーパコンピュータを利用しても，1 回のシミュレーションに，数日から数週間かかることも珍しくない．このような場合，サーバクライアント間の通信を確立したままリアルタイム可視化を継続することは困難である．また，興味ある現象が発生するまでは，シミュレーションの進行状況を定期的にチェックしたり，それまでに生成された動画を再生して概要を把握したりするだけで十分なことも多い．このような長時間シミュレーションへの適用に対してどのような手段を与えるかが課題である．

4.4 バッチ処理環境への適用

大規模シミュレーションが実行されるスーパコンピュータセンタなどではバッチ処理運用を基本とする場合が少なくない．バッチ処理中のジョブとの通信は困難であり，禁止されている場合もある．これは，ユーザからのパラメータ入力待ちなどで CPU がアイドル状態になるのを極力避け，効率的な利用を促すための運用方針として設定されていることが多い．この場合，通常の会話型の可視化処理は不可能であり，これにかわる方策を考える必要がある．

4.5 不可逆性

リアルタイム可視化では通常，1 時刻ステップまたは数時刻ステップに 1 枚の可視化画像を生成していく．現在の時刻ステップの結果に対して，別の視点からの可視化や別の図種を用いた可視化を行いたいという場合は，シミュレーションの進行を一時停止させたうえで可視化パラメータを変更すればよい．一方，すでに計算し終えた時刻ステップの結果に対してこれを行いたいという場合は，計算結果を残しておいたうえでポストプロセッシングに頼る必要がある．しかし，計算結果を保存しなかった場合や，計算結果の保存が不可能な大規模シミュレーションの場合は，最悪シミュレーションを再実行する必要がある．このような事態は，なるべく避けるようにしなければならない．

4.6 グラフィックス操作の困難性

クライアント側では二次元の画像データしか持たないため，グラフィカルオブジェクトの拡大/縮小や回転などといったローカルな三次元操作には不利である．可視化画像に対して，マウス操作でこれをスムーズに行おうとすると，マウスの動きに追随してサーバ側で可視化画像を作成し，クライアント側へ連続送信しなければならない．これはネットワーク性能上困難であり，サーバ側の負荷も高くなる．前章でみたアプローチ C のアプローチ A または B に対する最大の弱点である．リアルタイム可視化を行う場合は，多くの時刻ステップでこのような操作を行う必要性は低いといえる．しかし，決定的現象が発生した時刻ステップにおいてはこのような操作が多用される可能性がある．多くの時刻ステップからなる非定常データのポストプロセッシングを行う場合も同様である．

5. 課題の克服

5.1 メモリ参照モデルとライブラリインタフェース

典型的な非定常シミュレーションプログラムにおける，RVSLIB ライブラリの呼び出し順序を**図 3** に示す．呼び出されているサブルーチンの役割は，以下のとおり

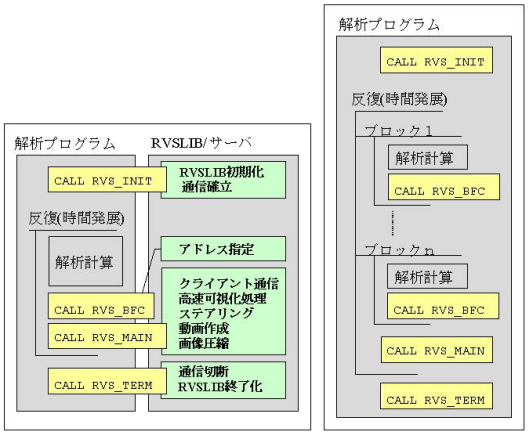


図 3 典型的なライブラリ呼び出し順序．左図：シングル BFC 格子の場合，右図：マルチブロック BFC 格子の場合

Fig. 3 Calling sequence of RVSLIB library in a typical simulation program using a single BFC grid (left) or a multi-block grid (right).

である．つまり，RVSLIB の初期化およびクライアントとの間の通信確立（RVS_INIT），格子データおよび計算結果が格納された配列のアドレス指定（RVS_BFC），クライアント側からの可視化パラメータ獲得および可視化処理（RVS_MAIN），RVSLIB 終了化および通信切断（RVS_TERM）である．ここでは，三次元 BFC（Boundary Fitted Coordinate）格子を用いたシミュレーションを想定している．非構造四面体格子，非構造六面体格子，または多数粒子を用いたシミュレーションプログラムの場合，RVS_BFC のかわりに RVS_TET1, RVS_HEX1, または RVS_PARTICLE を呼び出す．それぞれのデータ種別に応じて，専用の可視化処理プログラムが RVS_MAIN から呼び出されることになる．マルチブロック格子の場合は，各ブロックごとにこれらのアドレス指定ルーチンを呼び出し，全ブロックで 1 時刻ステップ分の計算が終わった段階で RVS_MAIN を呼び出せばよい．このライブラリインタフェースの特徴は，すべての可視化処理が RVS_MAIN に集約されていることである．視点の位置などのビュー情報を設定したり図種ごとに必要な情報を設定するサブルーチンをさらに呼び出す必要はない．クライアント側から指定されたこれらの情報を，RVS_MAIN が受信する．このライブラリ形式は，時間発展問題を解析するシミュレーションプログラムと自然に同期がとれるようになっている．つまり，1 時刻ステップ分の計算が終了するたびに，1 つまたは複数のアドレス指定ルーチンとともに RVS_MAIN を 1 回呼び出せばよい．RVSLIB をポストプロセッサとして利用する場合は，1 時刻ステップ分の計算結果を読み込むたびに，これらのサブルーチンを呼び出すことになる．

さらに格子データおよび計算結果が格納された配列に対しては，整合寸法を用いたインタフェースが採用されている．これは数学ライブラリのインタフェースで採用されているものであり，データ形式に対する制約を大幅に緩和する．

以上のサブルーチンをシミュレーションプログラムから呼び出すだけでひととおりの可視化処理ができるようになっているが，格子データや計算結果以外のシミュレーション情報が必要となる可視化に備え，ユーザ関数が定義されている．利用者が必要に応じて作成したこれらのユーザ関数は，RVS_MAIN から内部的に呼び出され，シミュレーション情報をやりとりするために利用される．たとえばトレーサ計算に必要な時間刻み幅は，ユーザ関数の 1 つを通してシミュレーションプログラムから獲得する．また，クライアント側で指定されたシミュレーション制御のためのパラメータ値は，最終的にユーザ関数の 1 つを通してシミュレーションプログラムに渡される．ユーザ関数とシミュレーションプログラムとは，COMMON 変数（FORTRAN 用語で）を通じて情報をやりとりする．主なユーザ関数には，以下のものがある．

RVS_USER_OBJECT_BFC：オブジェクト（障害物など）のブロックデータを RVSLIB に与える．

RVS_USER_OBJECT_POLYGON：オブジェクト（障害物など）のポリゴンデータを RVSLIB に与える．

RVS_USER_CUT_BFC：O 型，C 型などの BFC 格子のトポロジー的切断面の位置を RVSLIB に与える．

RVS_USER_TIME：時間刻み幅を RVSLIB に与える．

RVS_USER_STEERING：ステアリングのためのパラメータをシミュレーションプログラムに与える．

5.2 高速可視化処理（ダイレクトイメージ生成）

計算サーバ上でほとんどの可視化処理を行う場合，マッピングおよびレンダリング処理をいかに効率的に行うかが重要となる．RVSLIB では，複数のベクトルプロセッサを持つ共有メモリ並列型のスーパコンピュータ上での利用を想定し，処理の高速化を図っている．これは，このようなスーパコンピュータが大規模数値シミュレーション実行のための主要なプラットフォームの 1 つだからである．その具体的な処理内容を，三次元空間内の任意の断面上に等高線を描く場合を例として説明する．

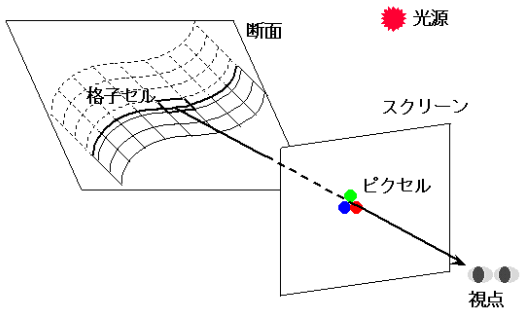


図 4 ダイレクトイメージ生成の模式図
Fig. 4 Explanatory figure for the direct image generation.

等高線図では，ダイレクトイメージ生成という手法を導入することにより，その処理コストを低減している．これは，表示する計算格子の格子番号やその格子内での内挿係数を表示画面の全ピクセルについてあらかじめ計算して保存し，以降の処理においてはこれらの係数を用いて計算結果から直接可視化画像を生成する方法である．一般にリアルタイム可視化を行う場合，時刻ステップごとに可視化パラメータが変更されることは少ないと考えられる．この場合，連続する複数の時刻ステップを通じて，1 つのピクセルは断面上の同一点をつねに表示する（**図 4**）．この点におけるデータ値 D は，この点を含む格子セルを形成する格子点上のデータ値から補間により次のように求められる．

$$D = \sum_{l=1}^{n} w_l D_l$$

ここで n は 1 つの格子セルを構成する格子点の数であり，w_l $(l = 1, \ldots, n)$ は内挿係数である．内挿係数は対応する格子点と表示点との距離が小さくなるほど大きくとられるが，上記条件の下では複数の時刻ステップを通じて一定に保たれる．格子点上のデータ値 D_l $(l = 1, \ldots, n)$ だけは毎時刻ステップ変化する．こうして補間したデータ値に対応する色と，断面に当たる光の強さを掛け合わせたものがピクセルの輝度値となる．各ピクセルに対する内挿係数を保存しておけば，各時刻ステップで必要な処理は上記補間計算のみとなる．必要なメモリ容量は画像の大きさのみに依存する．可視化処理のコストはシミュレーション規模に依存しない．さらにこの処理はベクトル化および並列化が可能であるうえ，ベクトルプロセッサを有しない一般の計算機上でも可視化処理のコストが削減できる．

5.3 プラグイン/プラグアウト

RVSLIB では，サーバクライアント間の通信を自由に接続/切断する機能を実現している．これをクラグイン/プラグアウト機能と呼ぶ．クライアントからサーバモジュールへの通信を切断（プラグアウト）すると，クライアント側の処理は停止し，サーバ側の処理だけが行われるようになる．さらにこのとき，ユーザ端末をシャットダウンすることも可能である．その後，クライアントモジュールからサーバ側へ通信の接続（プラグイン）を行うと，通信切断時の可視化パラメータが自動的に復元され，可視化画像がユーザ端末に表示されるようになる．

プラグアウト時には，以降のサーバ側の動作モードとして，以下のものが選択できる．

- シミュレーション単独実行モード：
  RVSLIB のサーバモジュールの実行も停止し，数値シミュレーションだけが計算サーバ上で実行される．
- バッチ処理モード：
  数値シミュレーションと RVSLIB のサーバモジュールによる可視化処理が計算サーバ上で継続される．このとき，通信切断時の可視化パラメータの内容が使用される．ユーザ端末上に可視化画像は表示されない．ただし，通信切断時に，以降の可視化画像をサーバ側のファイルに保存するよう指定することができる．また，通信切断中でもトレーサ計算を継続し，再接続後にこれを正しく表示したい場合は，この動作モードを選択する必要がある．

このような機能は，長時間を要する大規模シミュレーションのリアルタイム可視化には特に有用である．さらに，リアルタイム可視化実行中に何らかの理由で通信が絶たれた場合も，サーバ側はシミュレーション単独実行モードに自動的に切り替わる．このように，不意の通信トラブルなどからシミュレーションの実行が保護されるような工夫もなされている．

5.4 シナリオ機能

視点の位置などの可視化パラメータの変更は，通常クライアント側の GUI から行う．一方，シミュレーションの進行にともなう可視化パラメータの変更の過程をあらかじめファイルに記述しておけば，リアルタイム可視化またはポストプロセッシングの際にこの“シナリオ”（台本）に則った可視化が自動的に行われる．これがシナリオ機能であり，シナリオを記述したファイルをシナリオファイルと呼んでいる．RVSLIB ではシナリオファイルを読み込むと，その構文解析や意味解析などを行い，可視化パラメータをシナリオに従って自動更新し，可視化処理を行っていく．

シナリオファイルはキーフレーム方式で簡単に記述

することができる．その例を以下に示す．

```
$beginframe step=300;
   $viewing;
        eye = 1.0 0.0 0.0;
        look_at = 0.0 0.0 0.0;
   $end;
$endframe;
#
$beginmotion;
   interpolation='polar';
$endmotion;
#
$beginframe step=600;
   $viewing;
        eye = 0.0 1.0 0.0;
   $end;
$endframe;
```

これは，時刻ステップ（step）300 から 600 にかけて，視点の位置（eye）を注視点（look_at）を中心として (1, 0, 0) から (0, 1, 0) にスムーズに回転させる操作を記述したものである．注視点はつねに原点である．キーフレーム間の可視化パラメータの補間方法は，キーワード “interpolation” を用いて指定する．引き続くキーフレーム間で変更されている可視化パラメータ（ここでは視点の位置）が補間される．補間方法としては，ここで示した極座標空間での補間に加え，線形補間および移動平均による補間が可能である．シナリオファイルには RVSLIB で用いている可視化パラメータの多くを記述できるが，変更が必要なパラメータのみ記述すればよい．他のパラメータについては，シナリオ機能利用前に設定されていた値が引き続き用いられる．

シナリオ機能は，バッチ処理環境下でも威力を発揮する．シナリオファイルを計算サーバ上に置き，RVSLIB を組み込んだシミュレーションプログラムをバッチ処理で実行する．この場合，クライアント側とは通信を行わない．これにより，シナリオに則った可視化が自動実行され，シミュレーション終了時点で動画ファイルが完成していることになる．もちろんシミュレーション途中であっても，この動画ファイルにアクセスすれば，その時点までのシミュレーションの進行状況を動画として確認することができる．

5.5 マルチカメラ

マルチカメラは，複数の可視化パラメータセットを用いて可視化を行う機能である．各可視化パラメータセットは，1 回のシミュレーションに対して別々の可視化画像を作成する．これにより，可視化する断面ごとあるいは利用する図種ごとに別々の画像にしたり，複数の視点から同時に眺めたりするなどのことができる．

たとえば，多くの荷電粒子の運動と，これによって引き起こされる電磁場をシミュレーションする場合を考えてみる．このとき，粒子の運動の様子は粒子図で，電磁場の変化の様子は等高線図やベクトル図などで可視化するのが一般的である．これらを別々の画像にして解析できれば便利である．また複数の現象が絡み合う複雑なシミュレーションを行う場合，それぞれの現象に適切な可視化手法は異なるであろう．たとえ 1 つの現象を 1 つの図種で可視化する場合であっても，これを複数の視点から同時に眺めることができれば，その理解は飛躍的に高まる．さらに，各時刻ステップの計算結果を数値データとして残しておくことが不可能な大規模シミュレーションにとっては，複数の可視化パラメータセットを許すことで，興味ある現象を見逃してしまう危険性を極力抑えることができる．

バッチ処理の場合は，複数のシナリオファイルを同時に指定できるようにすることで，マルチカメラ機能を実現している．会話型処理の場合は，特定のカメラの画像だけを表示したり，複数のカメラの画像を並べて表示したりできる．

5.6 アクションウィンドウ

アクションウィンドウは，可視化画像の移動，回転，拡大/縮小などといったグラフィックス操作をマウスで行うために用意されている．この機能を利用すると，解析領域の外形およびオブジェクト（シミュレーションにおいて障害物などとして定義された物体）の外形を表すワイヤフレームが三次元グラフィカルオブジェクトとしてサーバ側で用意され，クライアント側に送信される．クライアント側では，受信したグラフィカルオブジェクトを画像化し，可視化画像に重ねて表示する．この表示画像の上でマウスボタンを押しながらマウスカーソルを動かすと，マウスボタンが押された時点でのマウスカーソルの位置に応じて，ワイヤフレームだけが移動，回転，または拡大/縮小する．ワイヤフレームを所望の配置にした時点で画像更新指示を GUI 上から行うと，その配置に応じた新たな可視化パラメータがサーバ側へ送信され，更新後の画像が送られてくる．アクションウィンドウを利用した場合の表示例を**図 5** に示す．

可視化画像自体をマウスで動かして可視化パラメータを設定することはできない．しかし，アクションウィンドウを用いた操作で利用されるグラフィカルオブジェクトは通常小規模であり，ネットワーク上の転

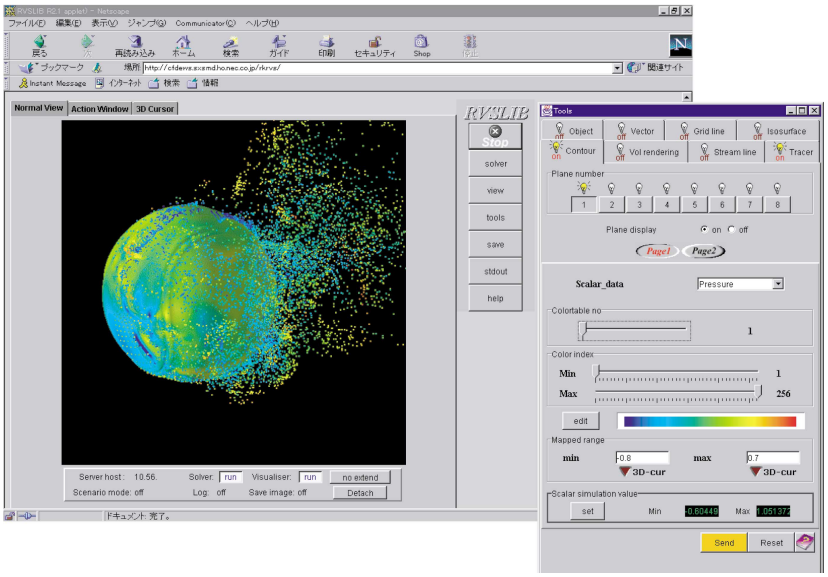


図 2 クライアントモジュールの表示例．Java アプレットとして実現されたモジュールが，Web ブラウザの中で動作している．野球ボールの周りの流れのリアルタイム可視化を行っている．

Fig. 2 The graphical user interface of RVSLIB/Client. The client module that has been developed as a Java applet runs on a Web browser. The concurrent visualization is performed for the flow calculation around a baseball with stitches. The influence of stitch line position on the fluid flow is investigated using the contour and tracer tools. The number of grid points is 337 × 181 × 101. This model is supplied by courtesy of the Institute of Physical and Chemical Research.

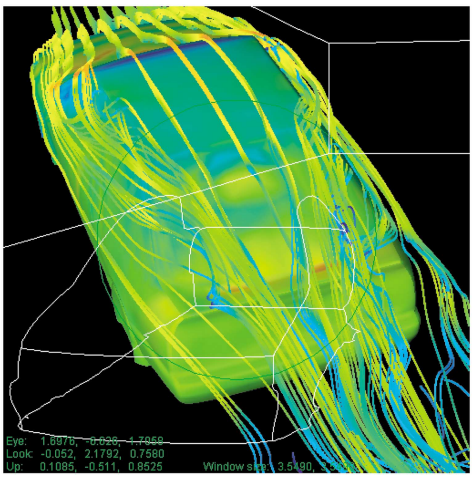


図 5 アクションウィンドウの表示例．白いワイヤフレームは，解析領域とオブジェクト（障害物）の外形を表している．モデルは日産プリメーラ．RVSLIB を組み込んだシミュレーションプログラムは，車の周りの流れの解析を行う．格子サイズは 173 × 101 × 80 である．

Fig. 5 The view of the action window. The white wireframe represents the extents of the computation domain and objects. This model is Nissan Primera supplied by courtesy of Nissan Motor Co., Ltd. The simulation program incorporating RVSLIB calculates fluid flows around the automobile. The number of grid points is 173 × 101 × 80.

送やクライアント側での操作の支障にならないという大きな特長がある．

6. 性能評価

RVSLIB を野球ボールの周りの流れのシミュレーションに適用し，性能評価を行った．このシミュレーションの目的は，ボールの縫い目が後流やボール自身に働く力に与える影響を調べるためである．これは，理化学研究所との共同研究により実施されている[7]．シミュレーションでは，非定常非圧縮性の粘性流体の運動量保存式（Navier-Stokes 方程式）と質量保存式（連続の式）から圧力の Poisson 方程式を導き，MAC 法に準拠して単一 BFC 格子上の差分法で解いている．対流項の微分には 3 次精度の上流差分を，他の空間微分には 2 次精度の中心差分を採用し，運動方程式の時間項は 1 次精度の陰的オイラー法を用いている．また，基礎方程式は運動するボールに固定した座標系で表現している．計算格子点数は本来 337 × 181 × 101 の約 620 万点である．計算時間節約のため 169 × 92 × 101 の約 160 万点で計算を行うことも多いが，格子点数の多いシミュレーションの方がより実験と一致する結果が得られる．4 つの変数（圧力および速度の 3 成分）が各格子点で計算されるため，1 時刻ステップあたり

の計算結果は，小さな計算格子の場合約 25 MB，本来の計算格子の場合約 100 MB になる．実際には 1 回のシミュレーションで 10,000 時刻ステップ以上の計算が行われるため，10 時刻ステップに 1 回の割合で計算結果を保存したとしても，最終的な規模は小さな計算格子の場合 25 GB，本来の計算格子の場合 100 GB 以上にも達する．さらに，ボールの縫い目の位置や，ボールの回転速度，ボールの進行速度などを様々に変更したシミュレーションを実行する必要がある．したがって，これらのシミュレーション結果を数値として残しておくことは不可能に近い．

大小 2 つの規模の計算格子を使用して，RVSLIB を組み込んだシミュレーションプログラムを NEC SX-4（CPU あたりのピーク性能は 2 Gflops）上でバッチ処理により 10,000 時刻ステップ実行した．等高線図とトレーサを用い，10 時刻ステップに 1 度可視化処理を行った．高精度なトレーサ計算を行うため，トレーサの時間積分計算は毎時刻ステップ行っている．また，可視化画像を SX-4 上に出力して動画ファイルを作成した．可視化処理を行うたびに視点の位置を変えた場合と，視点の位置をいっさい変えなかった場合との経過時間を，シミュレーションプログラムのみを実行した場合の経過時間とともに，**表 1** に示す．この表からすぐに分かることは，シミュレーションと同時に可視化処理を行うことによる経過時間の増加率が小さいということである．実際，これは最大でも 3.0%であり，計算規模とともに小さくなる傾向にある．より大規模な計算格子を使用した場合，これは 1.0%以下である．RVSLIB のマルチカメラ機能では，最大 8 台のカメラを設定できるが，この場合でも経過時間の増加率は 24%程度に抑えられる．

次に，SX-4 と端末（NEC EWS4800/460）をネットワーク接続し，同様の可視化処理を会話処理モードでも実行した．この場合，可視化画像はネットワーク上を転送されて端末側に表示される．ネットワークの実効性能は約 160 KB/sec であった．図 2 にこのときの可視化画像を示す．SX-4 側での画像圧縮，ネットワーク上の画像転送，および端末側での画像伸長に要する経過時間を**表 2** に示す．これらの経過時間もシミュレーションプログラムと比較して小さく，10 時刻ステップ程度に 1 度可視化処理を行う場合は問題にならないことが分かる．

以上の性能評価結果から，RVSLIB の可視化処理では十分な性能が達成されていること，これがシミュレーションに必要な経過時間に及ぼす影響は小さいことが分かる．このことは，RVSLIB が計算サーバ上でポストプロセッサとして利用される場合でも，他のシミュレーションプログラムに対する影響は小さいということも意味する．したがって，大規模非定常シミュレーション結果の可視化にとって，RVSLIB のような計算サーバ上での可視化方式が実用的な手段の 1 つになりうることが示されたといえる．

表 1 バッチ処理を行った場合の経過時間．上：小規模（160 万点）格子の場合，下：大規模（620 万点）格子の場合
Table 1 Elapsed times for the concurrent visualization in a batch-processing mode.

小規模格子	経過時間	増加率
CFD 単独	52080 sec (14.5 h)	-
+固定カメラ	53044 sec (14.7 h)	1.9%
+移動カメラ	53628 sec (14.9 h)	3.0%

大規模格子	経過時間	増加率
CFD 単独	242909 sec (67.5 h)	-
+固定カメラ	244832 sec (68.0 h)	0.79%
+移動カメラ	245280 sec (68.1 h)	0.98%

表 2 会話処理モードにおいて，画像の圧縮/伸長および画像転送に要する経過時間
Table 2 Elapsed times for transferring compressed images over a network.

	圧縮	転送	伸長
256 × 256 画像	0.057 sec	0.27 sec	0.085 sec
512 × 512 画像	0.20 sec	0.74 sec	0.50 sec

7. 今後の課題

7.1 可視化処理の分散メモリ並列化

RVSLIB の可視化アルゴリズムは，これまで主として共有メモリ並列型のベクトルプロセッサ向けに開発されてきた．しかし，大規模シミュレーションにおいては，分散メモリ並列型のスーパコンピュータの利用も主流になりつつある．

著者らは，分散メモリ並列型のスーパコンピュータに対応したリアルタイム可視化システムを日本原子力研究所と共同で試作している[11]．このシステムでは，重なりのない領域分割手法を用いたシミュレーションを想定しており，各部分領域の可視化は，この部分領域のシミュレーションを担当しているプロセッサが行う．これを，Owner Computation Rule と呼んでいる．各部分領域の可視化画像は最終的に 1 つのプロセッサに集められ，1 枚の可視化画像が生成される．このシステムは MPI を用いて実装されている．

したがって，この研究をさらに進め，シングルベクトル/共有メモリ並列/分散メモリ並列の 3 階層の計算機アーキテクチャに対して 1 つのシステムで統一的に

対応できるようにする必要がある．また，ここで開発された技術を，地球シミュレータ[8]用の可視化システムとして提供する予定である．

7.2 ライブラリインタフェースの汎用化

RVSLIBでは，BFC格子，非構造四面体格子，非構造六面体格子，および粒子系データに対するライブラリインタフェースと，それぞれに対する専用の可視化プログラムが用意されている．他の種類のデータに対するインタフェースと可視化プログラムは，利用者からの要求により順次整備していく必要がある．またRVSLIBをポストプロセッサとして利用する場合などを考慮して，いくつかの標準的なデータ形式に対するインタフェースも必要である．標準的なデータ形式としてはnetCDF（Network Common Data Form）などがある．

7.3 ローカルレンダリングによる対話性向上

特定の時刻ステップのデータに対してローカルな三次元グラフィックス操作を多用する場合，RVSLIBではアクションウィンドウを利用することになる．しかし，可視化結果としてのグラフィカルオブジェクトデータをクライアント側にもってくることができれば対話性はさらに向上する．また，計算規模と比較して，ネットワークやクライアント端末の性能に恵まれた環境であれば，グラフィカルオブジェクトの転送をベースとするアプローチBが有利になる可能性もある．大規模シミュレーションの場合，グラフィカルオブジェクトのデータ量も大きくなる．したがって，必要な可視化精度を維持しながらグラフィカルオブジェクトをサーバ側で効率的に削減するための手法の開発が必要である．

8. おわりに

非定常数値シミュレーションの計算結果はますます大規模化し，従来のポストプロセッシング手法で数百万自由度以上の計算格子を用いた大規模非定常数値シミュレーションの結果を可視化することは困難になる．その主な理由は3つある．第1に大規模な計算結果を端末側へネットワーク上を転送しなければならないこと，第2に計算サーバ側および端末側に大規模データを格納するためのディスクスペースが必要なこと，第3に端末側に大規模データを扱うことができる大容量のメモリスペースが必要なことである．本論文では，著者らが開発しているリアルタイム可視化システムRVSLIBの概要を紹介し，大規模非定常数値シミュレーションの結果の効率的な可視化をどのように可能にするかについて述べた．RVSLIBは計算サーバ上で可視化画像生成までのグラフィックス処理を行い，画像を圧縮し，ネットワークを通して端末へ転送する．これによって，計算サーバからPCなどのユーザ端末に計算結果を転送する場合に比べて，データ転送量を大幅に削減し，インターネットを含む広域のネットワーク分散環境での可視化を可能にしている．

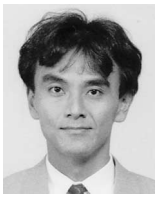
一方，可視化処理を計算サーバ上で行うことにともなう課題もあり，これを克服するためのいくつかの工夫を行った．たとえば，グラフィックス処理はベクトル化と並列化により高速化している．実際の流体解析コードに適用して行った性能評価によれば，リアルタイム可視化を行うことによる経過時間の増加率は大規模格子を用いた場合に1%以下に抑えられ，著者らの方式がますます大規模化しつつある数値シミュレーション結果の可視化に対して実用的であることが確認できた．

参考文献

1) Cignoni, P., Montani, C., Puppo, E. and Scopigno, R.: Multiresolution Representation and Visualization of Volume Data, *IEEE Trans. Visualization and Computer Graphics*, Vol.3, No.4, pp.352–369 (1997).
2) Crockett, T.W.: Design Considerations for Parallel Graphics Libraries, NASA CR-194935 (1994).
3) Doi, S., Matsumoto, H., Takei, T., Akiba, Y. and Schultheiss, B.C.: RVSLIB: A Library for Concurrent Network Visualization of Large-Scale Unsteady Simulation, *Proc. 21st SPEEDUP Workshop*, Vol.11, No.1, pp.59–65 (1997).
4) Geist, G.A., Kohl, J.A. and Papadopoulos, P.M.: CUMULVS: Providing Fault-Tolerance, Visualization and Steering of Parallel Applications, *International Journal of High Performance Computing Applications*, Vol.11, No.3, pp.224–236 (1997).
5) Guo, B.: A Multiscale Model for Structured-Based Volume Rendering, *IEEE Trans. Visualization and Computer Graphics*, Vol.1, No.4, pp.291–301 (1995).
6) Haimes, R.: pV3: A Distributed Systems for Large-Scale Unsteady Visualization, AIAA Paper 94-0321 (1994).
7) 姫野龍太郎，松本秀樹，土肥 俊：回転飛行する野球ボール周り流れの数値計算，第13回数値流体力学シンポジウム講演論文集 (1999).
8) http://www.gaia.jaeri.go.jp/
9) http://www.llnl.gov/terascale-vis/
10) Lane, D.A.: Scientific Visualization of Large-

Scale Unsteady Fluid Flows, *Scientific Visualization Overviews · Methodologies · Techniques*, Nielson, G.M., Hagen, H. and Muller, H. (Eds.), pp.125–145, IEEE Computer Soc. Press (1997).
11) Muramatsu, K., Matsumoto, H., Takei, T. and Doi, S.: A Real-Time Visualization System for Computational Fluid Dynamics on Parallel Computers, *Proc. Parallel CFD '98*, Elsevier Science (1998).
12) 武井利文，松本秀樹，土肥　俊：リアルタイム可視化システム RVSLIB，第 10 回計算力学講演会講演論文集，pp.413–414 (1997).
13) Wilhelms, J. and Van Gelder, A.: Multi-Dimensional Trees for Controlled Volume Rendering and Compression, *Proc. 1994 Symp. Volume Visualization*, pp.27–34 (1994).

(平成 12 年 4 月 27 日受付)
(平成 12 年 7 月 10 日採録)



武井　利文（正会員）
昭和 62 年京都大学大学院理学研究科物理学第一専攻修士課程修了．同年 NEC 入社．現在 NEC 情報通信メディア研究本部勤務．

松本　秀樹
昭和 63 年東北大学理学部化学科卒業．同年 NEC 技術情報システム開発（現 NEC 情報システムズ）入社．現在 NEC 情報システムズ科学技術システム事業部勤務．

土肥　　俊（正会員）
昭和 59 年北海道大学大学院工学研究科精密工学専攻博士課程修了．工学博士．同年 NEC 入社．現在 NEC 情報通信メディア研究本部勤務．日本計算工学会，可視化情報学会，日本応用数理学会各会員．