JAERI-Data/Code
98-014

並列計算機上での流体解析のための実時間可視化システムの開発

1998年3月

村松一弘・大谷孝之・松本秀樹*
武井利文**・土肥　俊**

日本原子力研究所
Japan Atomic Energy Research Institute

本レポートは，日本原子力研究所が不定期に公刊している研究報告書です。
入手の問合わせは，日本原子力研究所研究情報部研究情報課（〒319-1195　茨城県那珂郡東海村）あて，お申し越しください。なお，このほかに財団法人原子力弘済会資料センター〒319-1195　茨城県那珂郡東海村日本原子力研究所内）で複写による実費領布をおこなっております。

This report is issued irregularly.
Inquiries about availability of the reports should be addressed to Research Information Division, Department of Intellectual Resources, Japan Atomic Energy Research Institute, Tokai-mura, Naka-gun, Ibaraki-ken 319-1195, Japan.

編集兼発行　日本原子力研究所
印　　刷　㈱原子力資料サービス

JAERI－Data/Code 98－014

並列計算機上での流体解析のための実時間可視化システムの開発

日本原子力研究所計算科学技術推進センター
村松　一弘　・大谷　孝之・松本　秀樹*
武井　利文**・土肥　俊**

（1998年2月6日受理）

並列計算サーバ上での流体解析の結果を、ネットワークで接続されたクライアント上で解析と同時に可視化するとともに、解析および可視化のための種々のパラメータをクライアントのGUI（Graphical User Interface）で制御する実時間可視化システムを開発した。本システムでは、並列計算機上で流体解析から画像データ生成までの過程を並列処理することにより高速化するとともに、サーバからクライアントへの画像データ転送に画像圧縮技術を用いることにより、ネットワークの負荷を軽減している。可視化処理の並列化は、Owner Computation Ruleに基づいている。またクライアント側はOS非依存の実現方式として、Javaアプレットを利用している。これにより、Webブラウザさえインストールされていれば実時間可視化が可能になっている。

日本原子力研究所：〒153-0061　東京都目黒区中目黒2-2-54
*（株）NEC情報システムズ
**NEC C&Cメディア研究所

JAERI－Data/Code 98－014

Development of Real-Time Visualization System for Computational Fluid Dynamics on Parallel Computers

Kazuhiro MURAMATSU, Takayuki OTANI, Hideki MATSUMOTO*, Toshifumi TAKEI** and Shun DOI**

Center for Promotion of Computational Science and Engineering
Japan Atomic Energy Research Institute
Nakameguro, Meguro-ku, Tokyo

(Received February 6, 1998)

A real-time visualization system for computational fluid dynamics in a network connecting between a parallel computing server and the client terminal was developed. Using the system, a user can visualize the results of a CFD(Computational Fluid Dynamics) simulation on the parallel computer as a client terminal during the actual computation on a server. Using GUI(Graphical User Interface) on the client terminal, the user is also able to change parameters of the analysis and visualization during the real-time of the calculation. The system carries out both of CFD simulation and generation of a pixel image data on the parallel computer, and compresses the data. Therefore, the amount of data from the parallel computer to the client is so small in comparison with no compression that the user can enjoy the swift image appearance comfortably. Parallelization of image data generation is based on Owner Computation Rule. GUI on the client is built on Java applet. A real-time visualization is thus possible on the client PC only if Web browser is implemented on it.

Keywords:Real-time Visualization, Tracking and Steering, Computational Fluid Dynamics, Parallel Computer, Pixel Image Data Generation, Image Data Compression, Java Applet, Network Computing Environment, Domain Decomposition, Owner Computation Rule

* NEC Informatec Systems, Ltd.
** C&C Media Research Laboratories, NEC Corporation

目　　次

Contents

1 はじめに

近年、コンピュータの高速化、特に分散メモリ型の並列計算機による高速化が著しい。例えばアメリカのASCI Redでは、9152個のPEを用いて、LINPACKベンチマークで1.3TFLOPSの実効性能を実現している[1]。ASCI計画では、さらに数十TFLOPS級の並列計算機を計画しており、国内でも、数十TFLOPSの速度を有する並列計算機「地球シミュレータ」の計画がある。一方ネットワーク環境の方に視点を移すと、インターネットの世界ではデータ通信の高速化により、MPEGやgif形式による動画がポピュラーになりつつある。

このような計算機およびネットワーク環境下では、高速な並列計算サーバ上での解析結果をユーザ側のクライアントで実時間で可視化する指向性が生じてくる。また実時間可視化が実現すれば、解析の途中で結果を評価した上で、解析パラメータや可視化パラメータを変更したいという要求が高まってくる[2],[3]。従来このような方向を目指すシステムとしては、例えばMITのpV3がある[4]–[6]。そこでは、並列計算サーバ上でポリゴンなどのグラフィクスオブジェクトを生成し(マッピング処理)、それらをネットワーク上のクライアントに送る。クライアント上では、OpenGLなどの汎用グラフィクスライブラリやGWSを用いてレンダリング処理を行なう。一方、レンダリング処理を並列コンピュータ上で行うライブラリがNASA Langley研究所やカルフォルニア工科大学から報告されている[7]–[9]。

このような背景の下で、筆者らはCFD(Computational Fluid Dynamics)の解析計算からレンダリング処理までを計算サーバとしてのNEC SX-4上にて行い、生成されたイメージデータをユーザのクライアント上で実時間で可視化する(トラッキング)ソフトウェアRVSLIB(Real-time Visual Simulation LIBrary)を開発してきた[10]–[13]。このソフトウェアでは、クライアントのGUI(Graphical User Interface)からユーザが解析パラメータあるいは可視化パラメータを解析の途中で変更することも可能になっている(ステアリング)。さらに計算サーバからクライアントへのイメージデータ転送の際に、時間差分をベースにした独自の画像圧縮技術を用いることにより、ネットワークの負荷を軽減することが可能になっている。またこのソフトウェアはX Window、Motif、Socketなどの UNIXの標準的環境の上に構築されているため、高いポータビリティを持っている。さらにクライアント側はActiveX版への拡張も行なわれており、Windows95ベースのPCを利用することも可能である[14]。また同じ設計方針で筆者らは、RVSLIBとは別に分散メモリ型並列計算機Cenju-3上での実時間可視化システムも開発してきた[15],[16]。

本報告書では、Cenju-3上での実時間可視化システムをベースに、特に任意の並列計算機上で実時間可視化ができ、かつクライアント側もOS非依存の形式で実行できるシステムを、NECと共同研究により開発してきたものについて報告する[17]。具体的には、計算サーバ側のマルチプラットフォーム化として、原研にある種々の並列計算機、具体的には、NEC SX-4/FUJITSU VPP300/HITACHI SR2201/IBM SP/CRAY T94上での流体解析の実時間可視化を目指したものである。計算サーバ側のポータビリティを考慮して、本システムでは開発言語としてFORTRANおよびC、さらにプロセッサ間通信として標準的なMPI(Message Passing Interface)を採用している。クライアント側もOS非依存の実現方式として、Javaアプレットを利用して、Netscape NavigatorやInternet Explorerなどの Webブラウザさえインストールされていれば実時間可視化を可能にするように、GUIの拡張を行なっている。

第2章では、解析と可視化の分散方法について議論する。システムイメージと基本要件について第3章で述べる。第4章ではシステムの構成について、第5章ではシステムの特徴について解説する。第6章では、Owner Computation Ruleに基づく可視化処理の並列化について述べ、第7章では画像データ転送の性能評価結果について報告する。最後に、第8章で今後実現すべき課題について議論する。

2　解析の可視化の分散方法の比較

2.1　定性的議論

この章では、解析から可視化までの一連の処理 (解析、マッピング処理、レンダリング処理) のサーバとクライアントへの分割方法とその各方法の特質を議論する。ここで、“マッピング処理”は解析結果 (格子点上の物理量) からポリゴン・ベクトル等のグラフィカルオブジェクトを生成する処理を指し、その出力はこれらオブジェクトを含むグラフィクスコマンドである。また、“レンダリング処理”はこれら 3 次元情報に対して、投影・ライティング・陰面処理等を行なうことを指し、その出力はピクセルデータである。このとき、解析と可視化の分散方法には次の 3 通りが考えられる (表 1)。

アプローチ A　解析をサーバ、マッピング処理・レンダリング処理をクライアントでそれぞれ行なう。サーバからクライアントに転送されるデータは解析結果である。

アプローチ B　解析とマッピング処理をサーバ、レンダリング処理をクライアントで行なう。サーバからクライアントに転送されるデータは主にグラフィクスオブジェクトから成るグラフィクスコマンドである。

アプローチ C　解析・マッピング処理・レンダリング処理全てをサーバで行ない、ピクセル表示のみをクライアントで行なう。サーバからクライアントに転送されるデータはピクセルデータである。

それぞれの特質をまとめると次のようになる。

アプローチ A　サーバを解析に専有できるメリットがある。一方、解析結果全てをクライアントに転送するとすると通信ネックとなるため、表示に必要なデータ部位の切り出しとクライアント側でのその再構成が必要である。ただし、トレーサ計算では必要なデータ部位が動的に変化するので、例外的にサーバ側で行なうのが自然である。また、ボリュームレンダリングを行なおうとすると、基本的に全格子点のデータを必要とするので、これもサーバ側で行なわざるを得ない。すなわち、全てのマッピング処理をクライアント側のみで行なうことは不可能であり、実際にはサーバ・クライアントの処理の切り分けが必要となる。また、クライアントの CPU 性能によってはマッピング処理がボトルネックになりかねない。

アプローチ B　この場合、マッピング処理は全てサーバ側で行なうので、サーバとクライアントの切り分けは明確である。また、マッピング処理を十分ベクトル化すれば、それがボトルネックとなることはない。一方、レンダリング処理ではクライアントのグラフィクスハードを有効に活用できる。ただし、流体解析の可視化にとって、グラフィカルオブジェクトとしてポリゴンは必ずしも適してはいない (不必要にポリゴン数が増える)。

アプローチ C　特定のグラフィクスハードに依存しないのでポータビリティが高い。転送データサイズは表示ドット数と色数にのみ依存し、問題規模・表示法には依存せず一定である。また次節で示すように、転送データ量はアプローチ B に比較して一般に少ない。ただし、通常グラフィクスハードを用いて行なうレンダリング処理をソフト的に行なうため、その高速処理アルゴリズムの開発が必要となる。

以上をまとめると表 1 のようになる。

表 1: 解析・可視化処理の分散方法とその特質

アプローチ	A	B	C
サーバ	解析 一部のマッピング処理 (トレーサ計算、ボリュームレンダリング)	解析 マッピング処理	解析 マッピング処理 レンダリング
クライアント	マッピング処理 レンダリング	レンダリング	(ピクセル表示)
転送データ	解析結果	グラフィクスコマンド	ピクセルデータ
長所	解析処理にサーバを専有	グラフィクスハードの活用	ポータビリティ
短所	グラフィクス処理のCPU負荷 マッピング処理のサーバ/クライアントの切り分け 転送データ量	マッピング処理のベクトル化 グラフィカルオブジェクトの増大(ポリゴン)	ソフトウエアによるマッピング処理・レンダリング

2.2 定量的議論 −直交正方格子による例−

上記アプローチA,B,Cのデータ転送量を簡単な例で比較する。簡単のために、計算格子は$n_{\rm gr} \times n_{\rm gr} \times n_{\rm gr}$の直交正方格子とし、解析結果は各格子点で速度および圧力が出力されるとする。

アプローチAの場合　速度および圧力のデータは、倍精度実数とする。このとき、解析結果のデータ量は、

$$32 \times n_{\rm gr} \times n_{\rm gr} \times n_{\rm gr} \text{ bytes}$$

となる。これは、格子のサイズを$n_{\rm gr} = 51$とすると4.05MByte、$n_{\rm gr} = 100$とすると30.52MByteとなる。

アプローチBの場合　等高線、パーティクルトレーサ、オブジェクトを重ね合わせるものとする。それぞれの図種において、以下の仮定をおく。

等高線

1. 256本の等高線を生成する。
2. 最外側の等高線は断面上の$n_{\rm gr} \times n_{\rm gr}$の格子の最外周を通る。一方、最内側の等高線は中央の格子内に入る。その他の等高線は、その間に一様に分布するものとする。
3. 等高線は、格子点と等高線の交点をつなぐポリラインとして定義される。
4. 等高線毎に色データが加わる。

1等高線面当たりのデータ量

$$0.25 \times 4 \times n_{\rm gr} \times 256 \times 3 \times 4 \text{ bytes} = 3072 \times n_{\rm gr} \text{ bytes}$$

8等高線面では

$$D_{\mathrm{Contour}} = 8 \times 3072 \times n\mathrm{gr}\ \mathrm{bytes} = 24576 \times n\mathrm{gr}\ \mathrm{bytes}$$

パーティクルトレーサ

1. 10000個のパーティクルトレーサを描画するものとする。
2. 各トレーサ毎の転送データは、座標値と色データである。

10000個のトレーサのデータ量

$$D_{\mathrm{Tracer}} = 10000 \times 4 \times 4\ \mathrm{bytes} = 160000\ \mathrm{bytes}$$

オブジェクト

1. 格子内の立方体(複数)を描画する。
2. 一辺が$0.7n\mathrm{gr}$の大きな1つの直方体と$0.1n\mathrm{gr}$の小さな2つの立方体を想定する。
3. カラーマップされたワイアーフレームモデルを考える。
4. オブジェクトの位置は変わらない。従って、各タイムステップのデータ量にはカウントしない。
5. 格子点上の色情報を転送するものとする。

オブジェクトのデータ量

$$\begin{aligned} D_{\mathrm{Object}} &= 8 \times 0.7 \times 0.7 \times n\mathrm{gr} \times n\mathrm{gr} \times 4 + 2 \times 8 \times 0.1 \times 0.1 \times n\mathrm{gr} \times n\mathrm{gr} \times 4\ \mathrm{bytes} \\ &= 16.32 \times n\mathrm{gr} \times n\mathrm{gr}\ \mathrm{bytes} \end{aligned}$$

以上の仮定のもとで、転送するデータ量D_{Total}を計算すると、以下のようになる。

$$\begin{aligned} D_{\mathrm{Total}} &= D_{\mathrm{Contour}} + D_{\mathrm{Tracer}} + D_{\mathrm{Object}} \\ &= 160000 + 24576 \times n\mathrm{gr} + 16.32 \times n\mathrm{gr} \times n\mathrm{gr}\ \mathrm{bytes} \end{aligned}$$

これは、格子のサイズを$n\mathrm{gr} = 51$とすると1.39MByte、$n\mathrm{gr} = 100$とすると2.65MByteとなる。

アプローチCの場合　ここでは以下の仮定をおく。

1. RGBデータは4バイト整数としてストアする。
2. ビューポートのサイズは512×512とする。

このとき、1画面当りのデータ量は1MByteとなる。以上を$n\mathrm{gr} = 51$と$\mathrm{gr} = 100$の場合についてまとめると、表2のようになる。アプローチCではデータ量は計算格子サイズに依存せず一定である。この例に関する限り、アプローチCの方が転送データ量は少ない。ただし、アプローチBとの差は1～3倍程度である。アプローチCの場合、JPEG・MPEGといった画像圧縮技術を用いることによって、データ量を数十分の一から百分の一のオーダまで圧縮することが可能になる。一方、アプローチBの場合、数値データのため、圧縮率はせいぜい数分の一程度と思われる。したがって、データ転送に圧縮技術を用いることを考えるとアプローチCはさらに有利になってくる。アプローチCは昨今のマルチメディアパソコンでの動画利用の普及を考えると、今後よりポピュラーになってくると考えられる。すなわち、ユーザの端末としてパソコンを用いることによって、より安価かつ手軽に計算サーバ上の計算の実時間可視化を実現できる。

表 2: アプローチ B と C におけるデータ転送量の比較

アプローチ	データ量 (MByte)	
	$n$gr = 51	$n$gr = 100
A	4.05	30.52
B	1.39	2.65
C	1	

3 システムイメージと基本要件

この章では、システムが満たすべき要件について議論する。

3.1 システムのイメージ

われわれが目指すシステムのイメージは以下のようなものである。すなわち、ネットワーク分散環境下において、並列計算機などの高速計算サーバ上でのユーザプログラムによる流体解析をワークステーションないしパソコンなどのユーザ端末上で手軽に実時間 (オンライン) 可視化する Real-Time VisualizationSystem である。同時に、イメージデータを圧縮して保存することによって、オフラインで手軽に動画表示することができる。これは従来のアナログビデオに代わる機能である。以下では、われわれが目指すシステムが満たすべき要件を述べる。

3.2 基本要件

アプリケーションコードへの組み込み容易性 ユーザは、自身の解析コードに対して、上記実時間可視化の機能を手軽に付加することを期待する。したがって、システムはユーザの解析コードから簡単にコールするだけで利用できるライブラリ形式でなければならない。実際には、基本ルーチンとして、

- 実時間可視化機能たち上げ時に諸初期化を行なう初期化ルーチン
- 時間発展計算の中で毎回呼ばれ、画像生成と表示を行なう表示メインルーチン
- 実時間可視化機能を終了するための終了化ルーチン

などがある。実際には、データの受渡しなどにさらにいくつかのルーチンが用意されることになる。

移植容易性 ユーザは自身の机上の端末をクライアントとして利用したい。そこでは、グラフィクス専用ハードやあるいはグラフィクスライブラリが装備されているという保証はない。したがって、クライアントシステムは以下の要件を満たす必要がある。

- グラフィクスライブラリが不要である。
- FORTRAN77、C、UNIX の標準環境 (Xwindow、Motif、Socket など) や OS 非依存の上に構築される。このため、異なるハードウェア・プラットフォームへの移植が容易である。

同様のことは計算サーバ側のシステムにも要求される。

性能 計算実行と同時に可視化を行なうため、可視化処理に要する CPU 負荷やクライアント/サーバ間のデータ転送に要する通信負荷が計算実行に比べて充分に小さい必要がある。後に示す

ように、本システムでは、表示ルーチン用にベクトル計算アルゴリズムを開発し、計算サーバがベクトルマシンの場合、CPU を有効利用する。また Owner Computation Rule に基づいた並列化も行なっているため、並列計算機の資源も利用できる。

リアルタイム性 現在進行中の計算結果をグラフィカルに確認することによって、

- プログラム開発段階においては、デバッグの迅速化が期待できる。
- 実行時においては、例えば、計算に必要な種々のパラメータについて、その値が妥当であるかを、計算の立ち上がり時に容易に確認できる。
- 種々のパラメータに対して計算を行なうことによって設計最適化を行なう場合、リアルタイムにパラメータを変更することによって、パラメータ最適化に伴う試行錯誤の時間を (従来のポストプロセッシングに比べて) 大幅に短縮できる。

対話性 どのように計算結果を可視化するかを指定する表示パラメータの設定は GUI を用いて対話的かつ容易に行なえる必要がある。その場合、計算を一旦止めて表示パラメータを試行錯誤的に決定することができる必要がある。このために本システムでは計算処理と可視化処理の実行に関して、3 つのモードを用意している。すなわち、

- Real-Time Mode: 計算と可視化を同時に実行する。
- Halt_Solver Mode: 可視化パラメータの調整を行なうために、計算をストップし，可視化モジュールのみを動作させる。
- Halt_Visualizer Mode: 実時間の可視化の必要がない場合、可視化モジュールをストップし，計算のみを実行する。

汎用性 実時間可視化を行なおうとするユーザのプログラムは様々な領域 (流体・熱・構造・電磁場解析、およびそれらの連性解析など) におよび、またデータ形式もいろいろである。本システムは主に流体解析向きに開発されているが、他の場の問題にも適用可能である。ただし、現バージョンはデータ形式が境界適合座標系に限られており、有限要素法などの非構造格子系や、マルチブロックソルバなどに対しては、補間処理などを行なう必要がある。データ形式の汎用化は今後の課題の一つである。

ネットワーク分散環境への親和性 通常の Ethernet 上に構築された分散環境上でリソースを有効活用することは必須の条件となる。すなわち、比較的バンド幅の狭いネットワークでもクライアント/サーバ間のデータ転送がネックにならないようなシステムである必要がある。本システムでは、画像圧縮を行なうことにより、イメージデータ転送時のネットワーク負荷を最小限にする。なお、ATM ネットワークなどのバンド幅の大きなネットワークでは、逆に画像圧縮を行わないことで、CPU 負荷を少なくすることが必要である。すなわち、実際のユーザのネットワーク環境に応じてシステム計算/通信負荷バランスを調整できることが重要である。

アニメーションの実現容易性 従来、計算結果のアニメーション化においてはビデオを使っていたが、一般にビデオのコマ撮りは時間のかかる処理であり、その処理の短縮化・容易化が強く望まれていた。一方画像圧縮の世界では、Motion JPEG や MPEG といった高性能な画像圧縮方式の普及により、ワークステーションやパソコンでもかなり容易に動画を再現できるようになっている。現状では、画素数が多い場合には処理速度に問題があるが、今後の CPU 性能の向上により、この性能問題は比較的早期に解決されることが期待される。そのような環境においては、圧縮されたデジタルイメージによるアニメーションが従来のビデオにとって代わると予想される。本システムはネットワーク転送データに圧縮イメージを使っており、このようなアニメーションシステムとの親和性は元来高い。アニメーション作成に必要なシナリオの記述やそれに従った動画作成が容易に行なえることが求められる。

4 システム構成

4.1 全体構成

本システムは、ハードウェアとしては、計算サーバが単一PEのベクトル計算機、共有メモリ型のベクトル計算機、分散メモリ型並列計算機の3階層のプラットフォームを、クライアント側ではWebブラウザがインストールされていることを前提としている。さらに、データ中継の機能を果たすWebサーバの存在を仮定しており、計算サーバと高速なネットワークで接続されているものとする。そしてソフトウェアは、計算サーバ・Webサーバ・クライアントの各ハードウェア上で動作する、可視化モジュール・データ中継モジュール・GUIモジュールから構成される。

例えば、計算サーバが分散メモリ型並列計算機の場合、本システムの全体構成は図1のようになる。本システムでは並列計算機の特徴を生かして、PE(Processing Element)を解析・画像生成を行なうPE(以下、解析PEと略す)群と合成・圧縮を行なうPE(以下、合成PEと略す)とに分けている。さらに、解析・画像生成と合成・圧縮とがパイプライン的に処理され、実行速度の向上が図られている。なお解析・画像生成に関する並列化は領域分割法に基づいており、これについては第6章で述べる。

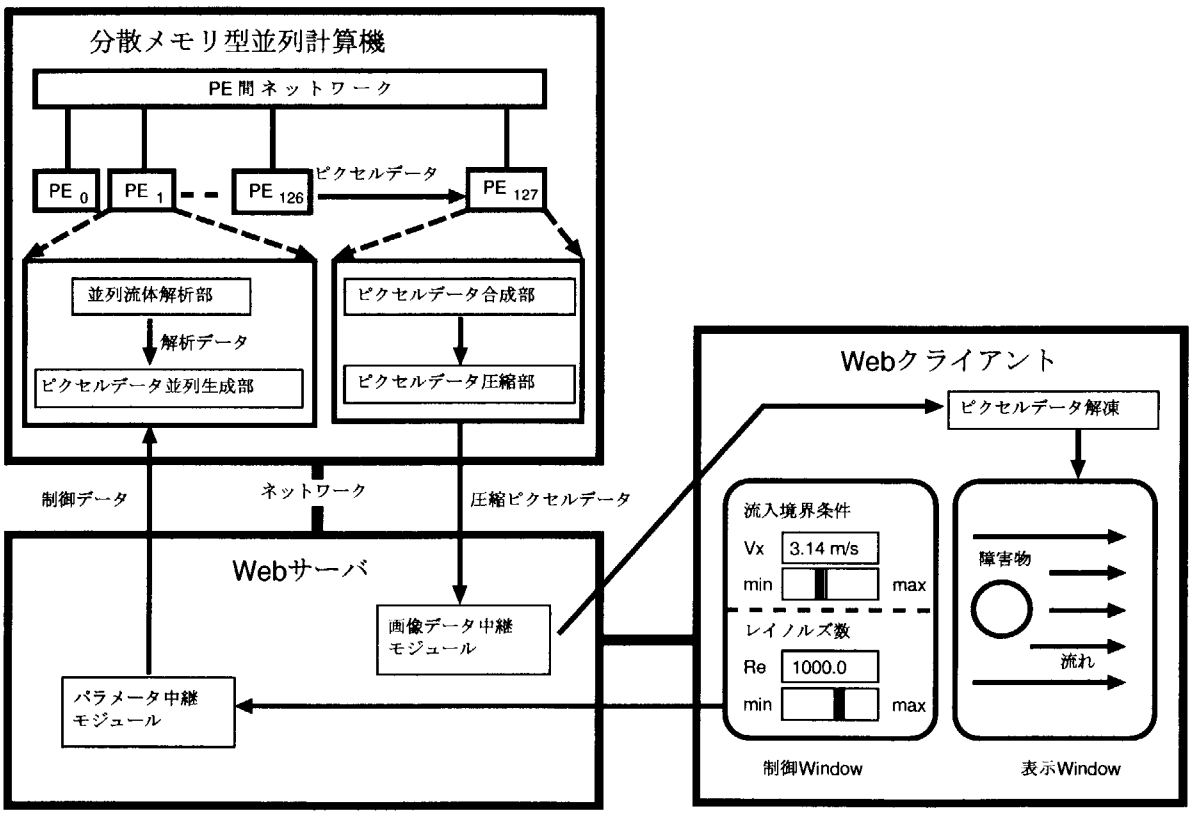


図 1: システムの全体構成

また本システムにおけるGUIは、OS非依存のWebベースで実現するために、Javaアプレットとして作成されている。そのため、計算サーバとクライアントとの間の画像データおよび制御パラメータの転送は、Webサーバを中継して行なわれる。

トラッキングの処理手順は、以下のようになる。

Step1 各解析PEで並列に解析を行なう。

Step2 各解析PEで解析データおよび可視化パラメータに基づいて、並列にピクセルデータを生成する。

Step3 ピクセルデータを合成PEに送信する。

Step4 合成PEでピクセルデータを合成し、1フレームの画像を生成する。

Step5 さらに画像データが、前画面とのデータ差分をHuffmanコードで符号化することにより圧縮される。

Step6 圧縮ピクセルデータを計算サーバからWebサーバに転送する。

Step7 Webサーバ上の画像中継モジュールにより、圧縮ピクセルデータがWebサーバからクライアントに転送される。

Step8 クライアントにて、ピクセルデータを解凍する。

Step9 解凍されたピクセルデータをJavaの2次元グラフィクスライブラリにより表示する。

ステアリングの処理手順は、以下のようになる。

Step1 制御Windowで決定された制御パラメータをクライアントからJavaアプレットの通信機能によりWebサーバに転送する。制御パラメータとしては、解析パラメータ、可視化関係では視点・カラーテーブル・光源の位置・カラーマッピングするデータの範囲などを計算の途中で変更することができる。

Step2 Webサーバ上のパラメータ中継モジュールにより、パラメータがWebサーバから計算サーバに転送され、計算サーバの磁気ディスクに書き込まれる。

Step3 計算サーバにおいて、PE0が制御パラメータを読み込み、他の解析PEにブロードキャストする。

Step4 各解析PEでは、新しいパラメータに基づいて計算およびピクセルデータ生成を行なう。

以下に各モジュールについて詳述する。

4.2 ピクセルデータ並列生成部

ピクセルデータ並列生成部の構成を図2に示す。ピクセルデータ生成部は流体解析プログラムからライブラリモジュールを呼び出すことにより実行される。図種としてはオブジェクト表示(障害物等の3次元物体の表示)・等高線図・ベクトル図・パーティクルトレースおよびそれらの重ね合わせ(等高線図・ベクトル図は複数断面)あるいはボリューム・レンダリングが可能である。各表示機能は、解析結果とカラーテーブル・ビュー情報・光源等の可視化制御情報を参照しながら、所定の可視化処理を行なう。出力画像の大きさは任意に設定可能である(標準は512×512)。

このピクセルデータ生成部の基本処理手順は以下の通りである。

Step1 指定された表示図種による可視化計算を実行し、互いに独立なRGB値配列とZ値配列に値を格納する。

Step2 **Step1**で得られた複数のRGB値配列を対応するZ値配列を参照しながら隠面処理し、1個のRGB値配列(イメージデータ)を得る。

Step3 得られたイメージデータを、合成PEに送信する。

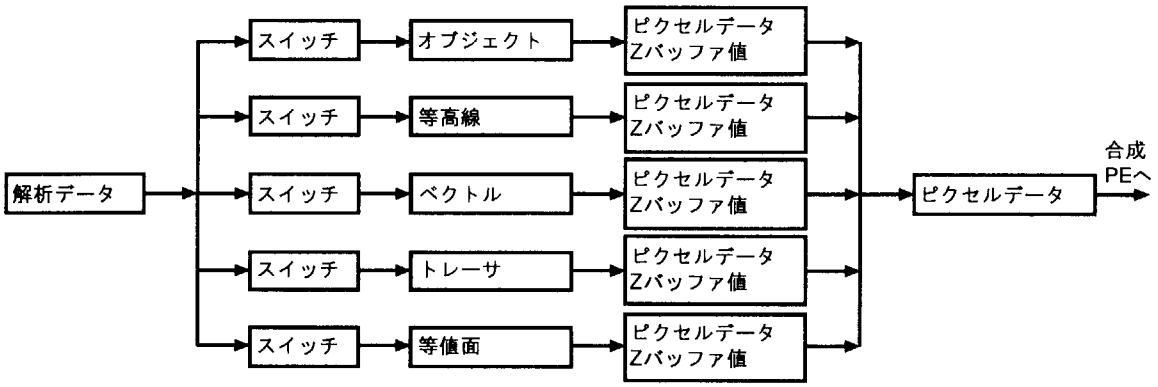


図 2: ピクセルデータ並列生成部の構成

ピクセルデータ生成部は、ほぼ全て標準 FORTRAN77 および C で記述されており、高いポータビリティを備えている。さらに、スーパーコンピュータや並列計算機での利用も想定し、ベクトル化および並列化されたアルゴリズムが採用されている [10]。特に並列化されたアルゴリズムの採用に当たっては、領域分割法による解析結果の形式を前提とし (Owner Computation Rule)、プロセッサ間通信量が最小限になるよう配慮されている [16],[17]。これについては、第 6 章で述べる。各図種の表示方法は、以下の通りである。

4.2.1 オブジェクト表示

本機能は 3 次元格子中の一部分を占める物体を表示するものである。一般的に、物体表面の頂点 (i,j,k) の組で表現される格子面は平面上になく 4 面体を形成する。この 4 面体の向きおよび凹凸と視点との関係から、見える 4 面体上の 3 角形を選び出し、イメージデータと Z 値の計算して表示処理を行っている。

表示タイプとして、色を RGB により指定するか変数データのカラーマッピングを行なうかを選択する。またシェーディングの手法はコンスタントシェーディング (格子面内は一様) である。

なお、表示アルゴリズムの詳細は以下の通りである。

Step1 3 次元格子において、$(i,j,k),(i+1,j,k),(i+1,j+1,k),(i,j+1,k)$ の頂点で表現される格子面を処理の単位とする。

Step2 物体表面の格子面の頂点を物理座標に変換する。4 頂点は一般的には 4 面体を形成する。

Step3 格子面の向きを表すベクトルを求める。

Step4 格子面の向きを表すベクトルに対して、頂点が形成する 4 面体の向きを定義する。

Step5 対象格子面の頂点とかかる格子面の周囲の格子点との物理座標を用いて、格子面の凹凸を定義する。

Step6 **Step4** で定義した 4 面体の向きと、**Step5** で定義した凹凸から、4 面体の 2 面をえらび物体表面とする。つまりここで 1 つの格子面を 2 つの 3 角形に分割する。

Step7 座標を投影面座標系に変換する。

Step8 3 角形の法線ベクトルを求め、これと視点方向との関係から、見える 3 角形を選択する。

Step9 見える3角形についてイメージデータとZ値の計算を行い、Zバッファ処理に必要なデータを格納する。

Step10 全ての物体上の格子面について **Step2** から **Step9** の処理を繰り返す。

4.2.2 等高線図

3次元$\xi\eta\zeta$格子中の格子面(ξ = 一定またはη = 一定またはζ = 一定)上に等高線を描く場合を考える。一般に実時間可視化を行なう場合、4.2節で述べた可視化制御情報が頻繁に変更されることはあまり無いと考えられる。このとき、連続する複数の時刻ステップを通じて、1つのピクセルPは格子面上の同一点$\boldsymbol{X}$を常に表示し、しかもそこに当たる光の強さIは常に一定である(図3)。点$\boldsymbol{X}$におけるデータ値は、この点を含むセルを囲む隣合う格子点上のデータ値D_lから次のよう

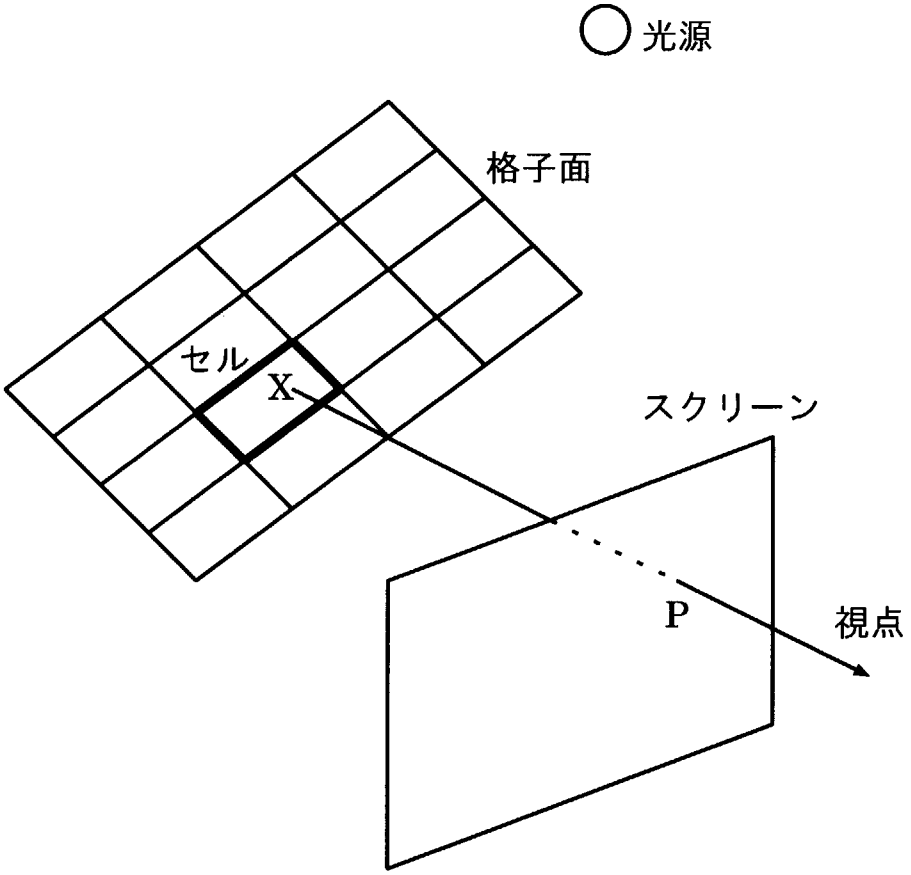


図 3: スクリーン、格子面、光源の位置関係

に求められる。

$$D = \sum_{l=1}^{4} w_l D_l \tag{1}$$

ここで、補間係数w_lは、一般に対応する格子点からの距離が小さくなるほど大きくとられ、上記条件の下では一定に保たれる。光の強さIは、

$$I = I_0 \max((\boldsymbol{n}, \boldsymbol{l}), 0), \qquad \boldsymbol{n} = \sum_{l=1}^{4} w_l \frac{\boldsymbol{e}_{\xi l} \times \boldsymbol{e}_{\eta l}}{|\boldsymbol{e}_{\xi l} \times \boldsymbol{e}_{\eta l}|} \tag{2}$$

で求められる。ここで、I_0は光源の強さ、$\boldsymbol{l}$は点$\boldsymbol{X}$から光源方向への単位ベクトル、$\boldsymbol{e}_{\xi l}, \boldsymbol{e}_{\eta l}, \boldsymbol{e}_{\zeta l}$は格子点$l$における局所基底ベクトルである。

こうして、等高線図作成のための以下のアルゴリズムが得られる。

Step1 可視化制御情報が変更されていれば、各ピクセルに対して、表示すべき格子面上の点を含むセルの番号と補間係数を求め、パーマネント領域に格納する。

Step2 可視化制御情報が変更されていれば、格子面上の各点に当たる光の強さを (2) 式より計算し、パーマネント領域に格納する。

Step3 格子面上の各点におけるデータ値を (1) 式より求め、これに対応する色とパーマネント領域中に格納された光の強さとを掛け合わせ、ピクセルのもつ RGB 値とする。

ここで、特に **Step3** において画面上のピクセルによるベクトル化が可能である。並列計算機の場合には、指定された格子面が通る領域を担当しているプロセッサにおいて、上記 **Step1-Step3** が独立に実行される。

4.2.3 ベクトル図

本機能は、解析結果のベクトルデータからベクトル図を描画する。色は RGB により指定する。表示アルゴリズムは以下の通りである。

Step1 描画が指定された格子面を投影面座標系に変換する。

Step2 ベクトルデータと視点方向との関係から、投影面座標系でのベクトル線の大きさと方向を計算する。

Step3 投影面座標系の各格子点で、指定された色でベクトル線を描画する。線分の描画手法には、Bresenham のアルゴリズムを用いる。

step4 描画が指定された全ての格子面に対して、**Step1** から **Step3** を繰り返す。

4.2.4 パーティクルトレース

パーティクルトレースとは、計算された速度場から仮想粒子 (パーティクル) の運動を計算し、その軌跡を表示することにより流体の運動を把握する手法である。パーティクルの移動計算は、物理座標系を正規直交座標系に変換し、写像空間で行なう。これにより包含セルの探索が不要になる。ここでセルとは 8 個の隣合う格子点で囲まれた空間を表す (図 4)。

パーティクルの移動計算には、単段型の陽公式の 1 つである 2 次の Runge-Kutta(1/2) 公式を採用する。

$$\boldsymbol{X}_1 = \boldsymbol{X}(t) + \boldsymbol{V}(\boldsymbol{X}(t), t) \cdot \Delta t/2, \quad \boldsymbol{V}_1 = \boldsymbol{V}(\boldsymbol{X}_1, t), \quad \boldsymbol{X}(t + \Delta t) = \boldsymbol{X}(t) + \boldsymbol{V}_1 \cdot \Delta t \qquad (3)$$

ここで$\boldsymbol{X}$はパーティクルの写像空間中の座標値、$\boldsymbol{V}$はその点での速度ベクトルを表している。

これらの定式化を用いて、表示アルゴリズムは以下のようになる。

Step1 パーティクルを包含する格子セルを調べる。

Step2 パーティクルの現在位置を写像座標に変換する。

Step3 パーティクルの現在位置における速度を求める。

Step4 Δt 後のパーティクルの位置を (3) 式から算出する。

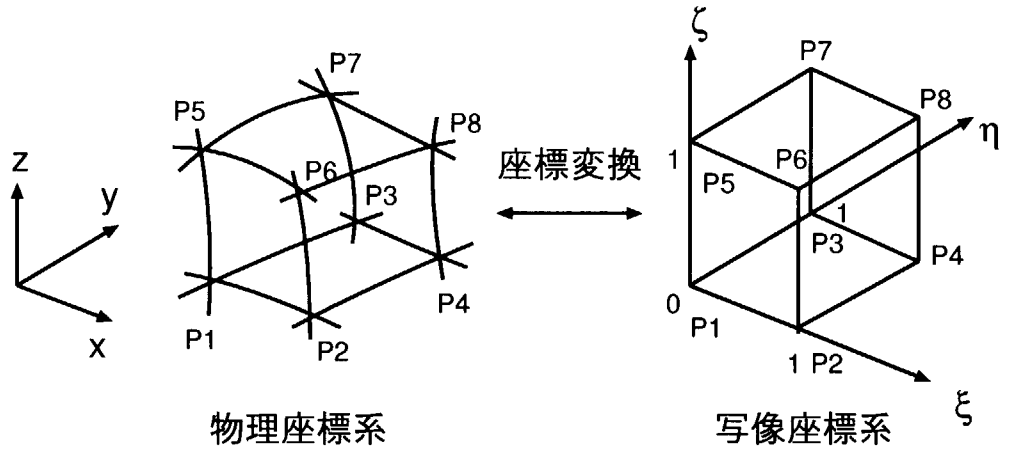


図 4: 座標変換とセル

Step5 パーティクルの位置を物理座標に変換する。

Step6 全てのパーティクルに対して **Step1** から **Step5** を繰り返す。

4.2.5 等値面

等値面は、MC(Marching Cube) 法により描画される。MC 法とは、等値面を構成する面をボリュームデータ内の各セルごとに計算する方法である。セルを処理単位として順番に面を計算していくことから立方体行進 (Marching Cube) 法と呼ばれている。基本的な処理の流れは、順番にボリュームデータ内のセルに注目し、セルの 8 つの頂点の物理量の状態から、等値面を構成する面を生成するというものである。

この等値面を構成する面は、セルの 8 つの頂点の物理量の状態で決めることができる。具体的には、あるセルに注目し、その 8 つの頂点の物理量 $f(i,j,k)$ と等値面のレベル (以下、しきい値と呼ぶ)c の関係は、$f(i,j,k)-c$ の正負をもとに考えると 2^8通りのパターンが存在する。ただし回転などの操作により等価なパターンを除くと、15 通りのパターンに落ち着くことがわかる。この 15 通りのパターンを MC パターンと呼ぶ。図 5 に代表的なパターンを示す。図 5 中の●印の格子点とそれが無い格子点の違いは、物理量としきい値の差の正負が異なることを示している。

図 5 から明らかなように、パターンによってはセル内に複数の面を求める必要がある。これは、面の頂点は必ずセルの 12 本のエッジ上のいずれかに存在するものとし、面の頂点の座標をセルの 8 つの格子点の線形補間により求める。また、面は全て 3 角形ポリゴンとなるように表示する。

以上をまとめると、等値面の表示アルゴリズムは以下のようになる。

Step1 セルごとに MC パターンを求める。

Step2 求めた MC パターンから面の頂点の座標を計算する。

Step3 求めた面を 3 角形ポリゴンとして描画する。

Step4 可視化制御情報に基づいて、ピクセルデータと Z 値の計算を行ない、Z バッファ処理に必要なデータを格納する。

ここに示した等値面表示ルーチンの処理は、それぞれのセルごとに 8 個の格子点における物理

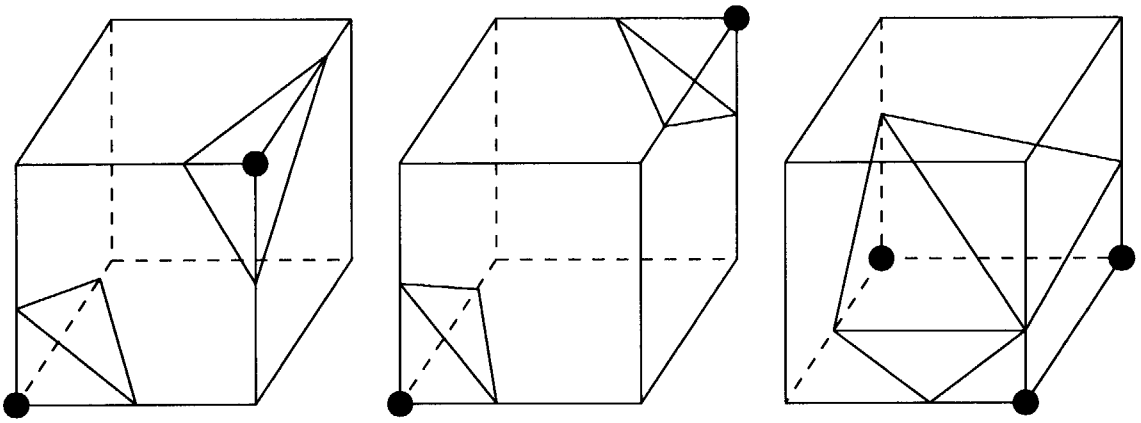


図 5: MC パターン

量から、面を求めることを行なっている。それゆえ、それぞれのセルごとに独立に実行することができるので、領域分割法に基づいた並列化は容易である。

4.2.6 ボリュームレンダリング

本システムで採用しているアルゴリズムの基本となるのはレイキャスティング法である [18]。この方法では、視線上のいくつかの点でデータ値をサンプリングし、その積み重ねの結果をこの視線に対応するピクセルの色として出力する。通常はデータ値のサンプリング操作を視線方向に行っていくが、ここでは全サンプリング点のうち、スクリーンに平行な面上に存在するサンプリング点を一まとまりとして考え、これらの点に関するサンプリング計算をベクトル化する方法を採用する (図 6)[19]。なおここでは、直交座標系を用いた解析結果を対象としているが、BFC(Boundary Fitted Curvilinear) 格子への拡張は容易である。

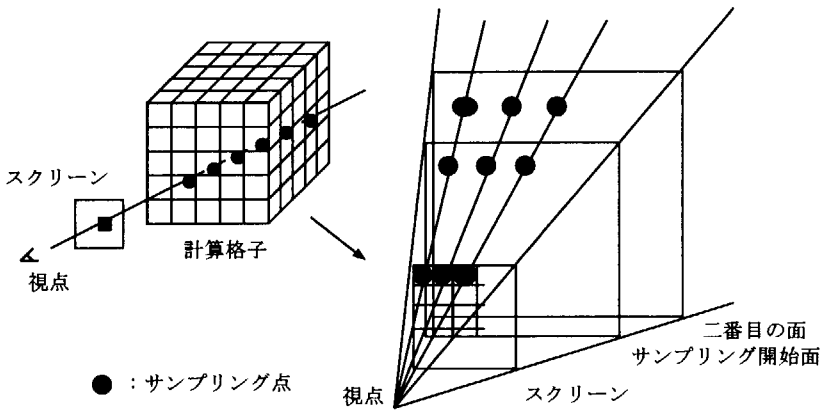


図 6: ボリュームレンダリングのベクトル化

計算アルゴリズムの詳細は以下の通りである。

Step1 各ピクセルに対する色を0に、光の透過率を1に初期化する。すなわち、

$$C_l^{(0)} = 0, \qquad \alpha_l^{(0)} = 1, \qquad (l = 1, \cdots, \text{画面サイズ}) \tag{4}$$

Step2 スクリーンに平行な面上のサンプリング点のうち、解析領域内に存在し、視点までの透過率が0でないものをリストアップする。次にこれらの点におけるデータ値を双一次補間により求める。

Step3 サンプリング点においてライティング計算を行なう際に必要な法線ベクトルを、データ値の勾配方向にとる。ここでは、まず次式により、格子点におけるデータ値の勾配方向を求める。

$$\begin{aligned}
(\frac{\partial D}{\partial x})_{i,j,k} &= \frac{D_{i+1,j,k} - D_{i-1,j,k}}{x_{i+1} - x_{i-1}}, \\
(\frac{\partial D}{\partial y})_{i,j,k} &= \frac{D_{i,j+1,k} - D_{i,j-1,k}}{y_{j+1} - y_{j-1}}, \\
(\frac{\partial D}{\partial z})_{i,j,k} &= \frac{D_{i,j,k+1} - D_{i,j,k-1}}{z_{k+1} - z_{k-1}}
\end{aligned} \tag{5}$$

これを双一次補間することにより、サンプリング点におけるデータ値の勾配方向を求める。

Step4 各サンプリング点におけるデータ値に対応する色及び透過率 A_l と、**Step3** で求めた法線ベクトルを用いてライティング計算を行なう。これにはフォンの式を用いる [20]。こうして得られたサンプリング点の輝度 I_l は、次のように対応するピクセルの色に加算する。

$$C_l^{(n+1)} = C_l^{(n)} + I_l(1 - A_l)\alpha_l^{(n)} \tag{6}$$

最後に各ピクセルに対する透過率を更新する。

$$\alpha_l^{(n+1)} = \alpha_l^{(n)} A_l \tag{7}$$

Step5 全てのピクセルに対する透過率が0になったら処理を終了する。

Step6 **Step2-Step5** を、スクリーンに平行な面ごとに、手前の面から奥に向かって繰り返す。

なお並列計算機の場合には、各プロセッサがそれぞれの担当領域に対して上記 **Step1-Step6** を独立に実行する。最後に以下の処理を行なう。

Step7 各ピクセルに対する色を0に、光の透過率を1に初期化する ((4) 式)。

Step8 各プロセッサより得られた色 C_l 及び光の透過率 α_l から、次式を用いて各ピクセルの色及び透過率を更新する。

$$C_l^{(n+1)} = C_l^{(n)} + C_l(1 - \alpha_l)\alpha_l^{(n)}, \qquad \alpha_l^{(n+1)} = \alpha_l^{(n)}\alpha_l \tag{8}$$

Step9 全てのピクセルに対する透過率が0になったら処理を終了する。

Step10 **Step8-Step9** を、手前のプロセッサから奥に向かって繰り返す。

4.3 ピクセルデータ合成部

合成PEでは、以下の処理手順に従ってピクセルデータを合成している。

Step 1 各解析PEからRGB値配列データ (ピクセルデータ) とZバッファ値配列データを合成PEにおいて受信する。

Step 2 各解析PEから送信された複数のRGB値配列をそれぞれ対応するZバッファ値配列の値を参照して隠面処理し、1フレームのピクセルデータを合成する。

4.4 ピクセルデータ圧縮部

前節で得られた1フレームのピクセルデータを、前画面とのデータ差分をHuffmanコードで符号化することにより圧縮する。これにより、ピクセルデータは約1/10に圧縮される。

4.5 画像データ中継部

並列計算サーバで生成された圧縮ピクセルデータは、Java言語の持つソケット通信機能を利用して、Webサーバに転送される。クライアントでは、Webサーバから画像データを読み込み、画像解凍・表示を行なう。

4.6 制御パラメータ中継部

GUIにて変更された解析・可視化のための制御パラメータは、Javaアプレットのソケット通信機能により、Webサーバを経由して並列計算サーバに転送される。転送された制御パラメータは、並列計算サーバの磁気ディスクに書き込まれる。そして並列計算サーバの解析PEの中のPE0が制御パラメータを読み込み、他の解析PEにブロードキャストする。各PEは変更されたパラメータに基づいて解析計算・画像作成を行なう。

5 システムの特徴

5.1 ライブラリ形式

本システムはユーザの解析プログラムからのサブルーチンコールによって呼び出される。解析プログラムがBFC格子の場合、基本的に以下の4つのサブルーチンを呼び出すことによって実時間可視化が実現される(図7)。

RVS_INIT 可視化機能の初期化、および画像合成用モジュールの起動。このルーチンは、CFDの反復計算の直前に呼び出す。

RVS_BFC BFC格子における、格子データおよび解析データの**RVS_MAIN**への引き渡し。

RVS_MAIN 可視化処理および表示の制御を行なう。このルーチンは、**RVS_BFC**の直後に呼び出される。

RVS_TERM 可視化機能の終了化。このルーチンは、CFDの反復計算の後に呼び出す。

これらの制御ルーチン以外に、ユーザ定義ルーチンが用意されている。これは、実時間可視化機能が一般のソルバに容易に組み込めるようにするためのものである。これらのルーチンをユーザがプログラミングすることにより、必要なデータや処理内容を可視化制御ルーチンに供給することができる。具体的には、以下の5つのユーザ定義ルーチンがある。

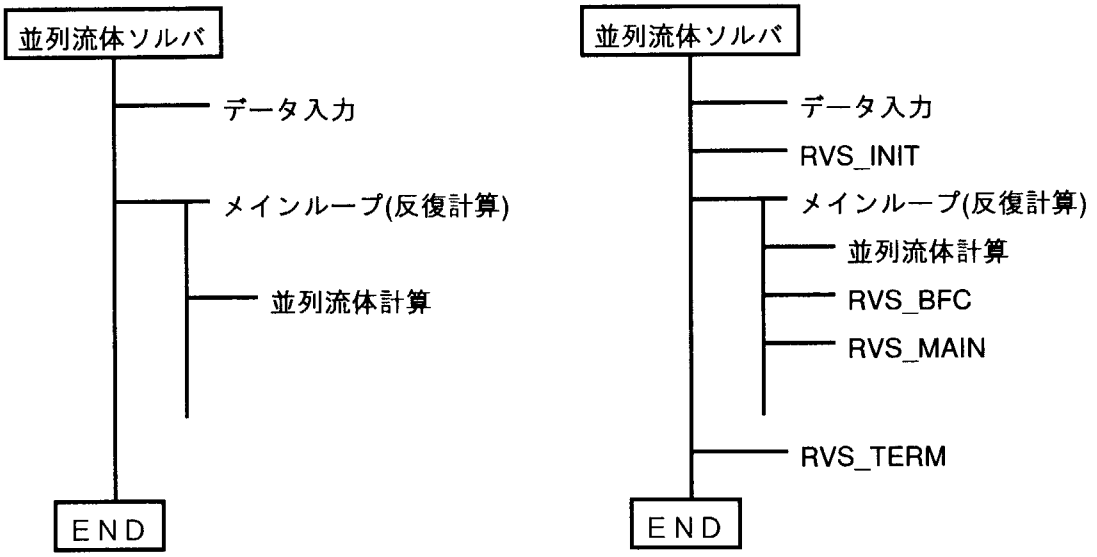
RVS_USER_INIT GUIおよび画像データ中継モジュールの起動用。

RVS_USER_BFC BFC格子における、並列化のための領域分割データ供給。

RVS_USER_OBJECT オブジェクト表示のためのオブジェクトデータ供給。

RVS_USER_TRACER トレーサ表示のための境界適合格子における切断面の位置データ供給。

RVS_USER_TIME 時間刻み幅供給。



(a) 可視化ルーチン呼び出し前　　(b) 可視化ルーチン呼び出し後

図 7: 可視化ルーチンの呼び出し構造

計算サーバ側の処理とクライアント側の処理は非同期で行なわれる。また可視化に必要なデータは整合配列として渡されるため、実際に大規模なデータのコピーなどを行なわずにイメージの生成がなされる。

具体的な実時間機能を組み込んだシステムの作成手順は、ユーザの CFD ソルバに **RVS_INIT**、**RVS_BFC**、**RVS_MAIN**、**RVS_TERM** などを組み込み、さらにユーザ定義ルーチンを作成する。これらのファイルをコンパイルし、他の可視化ルーチンのオブジェクト・ファイルとリンクすれば良い。

なおこのライブラリ形式は、BFC 格子だけではなく、非構造格子への設計も既に行なわれており、今後実装する予定である。非構造格子まで含めたライブラリ仕様は、付録 A で詳述される。

5.2 動作モード

これまで述べてきた実時間可視化以外に、ユーザは解析を途中で中断して解析結果の詳細を検討したい場合もある。あるいは初めから可視化を行なう必要が無く、解析をある程度まで進めた後で可視化を行ないたいということもありうる。そのため本システムでは、以下の 3 つの動作モードが用意されている。

実時間モード (Real-Time mode) 流体解析と可視化表示を同時に実行するモードである。解析と同時に結果を可視化することが可能である (トラッキング)。さらに、ライブラリモジュールはユーザインタフェースモジュールとソケット通信を行い、流体解析および可視化計算の実行を制御する。可視化された解析結果を見ながら、解析のためのパラメータを途中で変更するといったことが可能になる (ステアリング)。

解析中断モード (Halt_Solver mode) 流体解析を中断し、可視化表示とその制御のみを行なうモードである。ある時刻ステップの解析結果を詳細に調べたい場合に有用である。

可視化表示中断モード (Halt_Visualizer mode) 可視化表示を中断し、流体解析とその制御のみを行なうモードである。可視化表示を行なう必要の無い時刻ステップが続く場合に有用である。

5.3 グラフィカルユーザインターフェイス (GUI)

画像表示、ソフトウェアおよびユーザプログラムに対する諸制御はクライアントのGUIから行なう。本システムにおけるGUIは、OS非依存のWebベースで実現するために、Javaアプレットとして作成されている。アプレットは、以下に示すようなJavaのクラスで構成されている。

- **rvs.class**
  アプレットの中核クラス。初期画面のGUIボタン等の配置、画像表示部の機能、画像データと計算パラメータ通信のためのソケット生成から実際の通信を行なう機能を含む。
- **cgiexec.class**
  サーバ上のプログラムを実行するための通信 (CGI機能) を実現するクラス。**rvs.class** にある、Executionボタンが押された場合のコールバック関数から利用されている。
- **フレームクラス (Frame*.class)**
  パラメータ設定用のダイアログボックスを実現するクラス。それぞれの計算パラメータごとに、それぞれ別クラスとして実現されている。なお、それぞれのダイアログボックスにはSendボタンがあり、そのボタンが押された場合に入力されたパラメータはWebサーバへと送られる。Sendボタンが押されると、**rvs.class** で生成したソケット（これはStatic関数になっているため、他のクラスから利用可能）にデータを書き込む処理を行なう。

これらのJavaアプレットはネットワーク上のWebサーバに保管されている。クライアント上のNetscape Navigator、Internet ExplorerなどのWebブラウザから、このサーバをアクセスすることによってJavaアプレットをロードし、クライアント上のGUIを起動する (図8)。また、Webシステムの持つCGI(Common Gateway Interface)機能を用いることにより、Webブラウザから計算サーバにおける解析ソルバ・可視化モジュールおよびWebサーバ上のパラメータ中継モジュール・画像データ中継モジュールの起動も可能にしている。なお実際の可視化データの描画は、Javaの2次元グラフィクスライブラリを利用している

実際のGUIの描画イメージを図9に示す。

5.4 計算サーバのマルチプラットフォーム対応

可視化モジュールは単一PEのベクトル計算機、共有メモリ型のベクトル計算機、分散メモリ型並列計算機の3階層のプラットフォームを1種類のソフトウェアで対応し、かつ高性能を発揮することを、筆者らは目指している。

一般に、分散メモリ型並列計算機上での解析ソルバの並列化手法としては、領域分割法が広く用いられている。本システムも、解析ソルバと同様の領域分割法で並列化されている（これに関しては、次章で詳述)。並列化のための通信ライブラリとしてはMPIを用いている。

共有メモリ型並列計算機でMPIがサポートされていない場合には、MPIをダミールーチンとして扱うことで、可視化モジュールを実装している。このような工夫により、計算科学技術推進センターの複合並列計算機システムにおける、NEC SX-4/FUJITSU VPP300/HITACHI SR2201/IBM SP/CRAY T94の各並列計算機で実時間可視化が可能になっている。

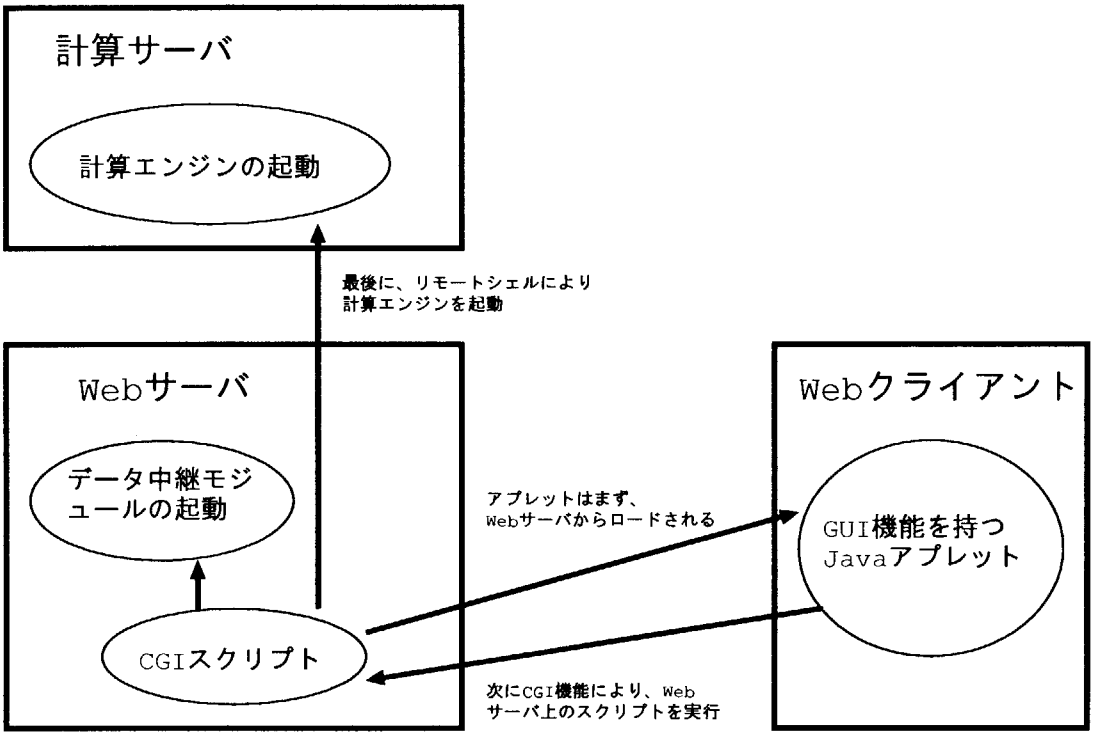


図 8: GUI の起動イメージ

6 Owner Computation Rule に基づく可視化処理の並列化

この節では、分散メモリ型並列計算機上での処理の分割・割当の問題を考える。一般に、第 2 章のアプローチ A では、並列計算サーバ上では CFD 計算のみが実行されるので、CFD 計算におけるデータ (配列) のみを分割すればよい。アプローチ B の場合、さらにポリゴンの生成があるが、その処理は基本的に格子に付随した処理と考えられるので、その分割は計算格子の分割に準じればよい。この分割法を以後、Owner Computation Rule と呼ぶことにする。

さてアプローチ C の場合、状況は少し異なり、ピクセルイメージ化された画像処理の並列化においては、画面分割 (ラスタ分割など) を行なうのが普通である。さらに、画面内の可視化対象が極存している場合、動的画面分割を行なうことによる PE 間の負荷の均等化が図られる (例えば [9])。具体的には、並列計算機の PE を CFD 計算用と可視化計算用に分別して、それぞれに適当な分割 (例えば、計算分割には領域分割、可視化分割にはラスタ分割) を行なう方法が思い浮かぶ。しかし、これは並列計算機内で上記 B を行なっていることになり、並列計算機内でのデータ転送がネックになる可能性がある。

以上の議論は計算分割と可視化分割の間の “協調化” の問題が、アプローチ C においては新たに生ずることを意味している。筆者らは、これまでの RVSLIB での経験から、可視化処理の負荷は計算処理に比べて比較的少ないと考え、アプローチ B で述べた Owner Computation Rule をこのアプローチ C にも適用することにした。すなわちピクセルイメージ上での各ピクセルの処理は、対応する計算格子が割り当てられている PE に割り当てるというものである。現在実現されているシステムでは、3 次元 BFC 系の CFD ソルバにのみ対応している。この場合について、上述の方法を説明する。

簡単化のために、2 次元の障害物回りの流れを考える (図 10)。図 10 の網掛け部分がユーザが可

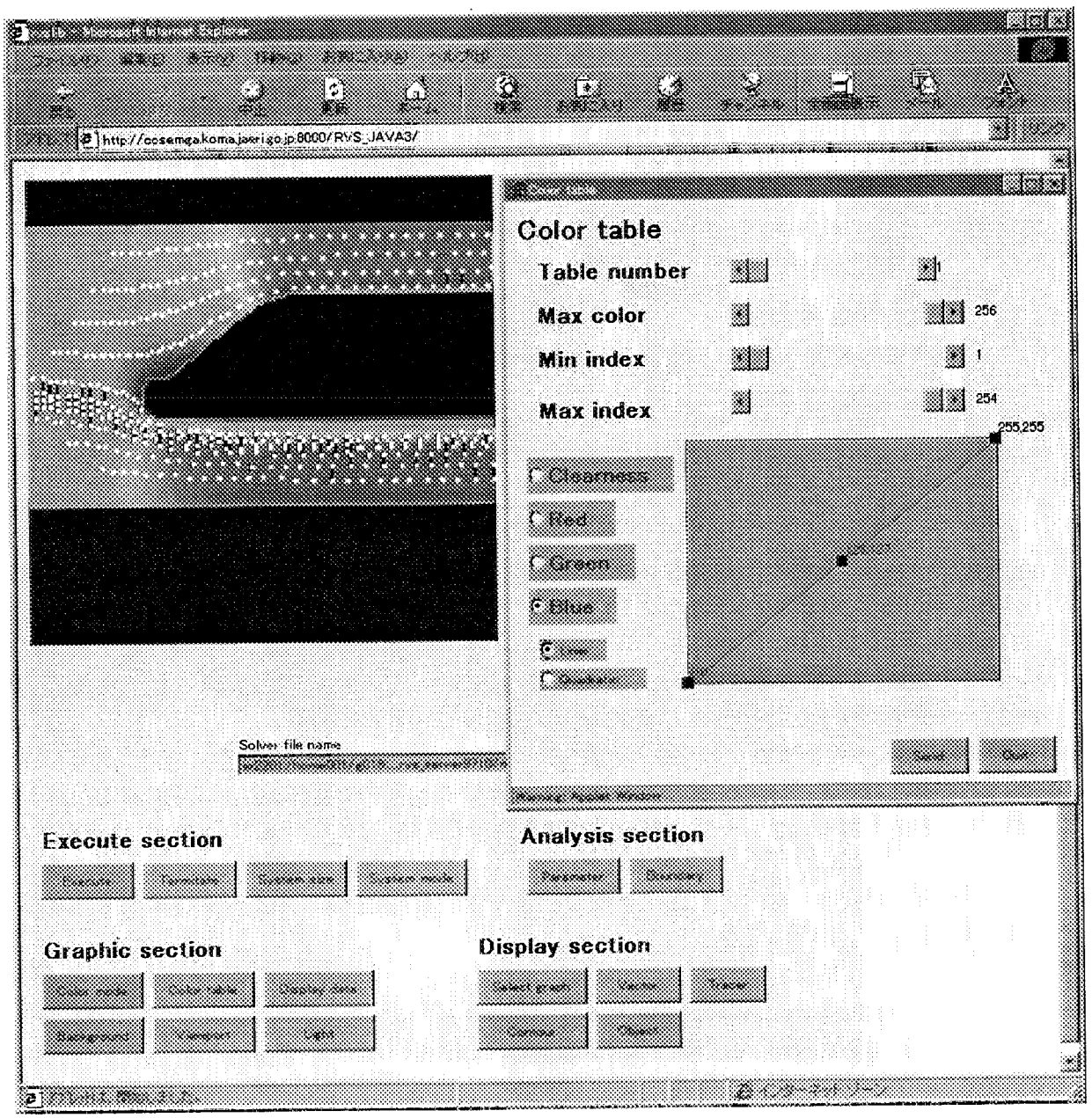


図 9: Web ベースの GUI

視化したい領域(クライアント上の表示画面)だとする。この時各PEは、図10に示されるようにそれぞれの計算領域の画面に対応する部分についてのみ可視化処理を行なう。図10から明らかなように、可視化領域を計算領域のある特定部分に限定(ズームアップ)した場合、その領域に対応するPEに可視化処理が集中することになり、PE内の負荷の不均等が生ずることになる。しかし前述のように、解析計算に比較して可視化処理の負荷は少ないと考えられる。それゆえ、解析計算と可視化処理を合わせた処理全体としては、その不均等は小さいと予想される。

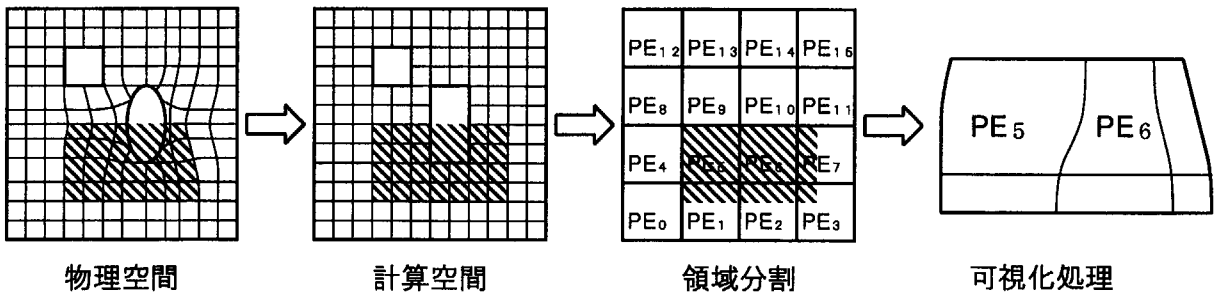


図 10: 可視化処理の並列化手法

7 SR2201を用いたシステムの性能評価

並列計算サーバとしてHITACHI SR2201(解析・画像データ生成に3PE、画像合成用に1PE)、Webサーバとして SGI POWER Onyx を用いた場合の性能評価を行なった。具体的には、計算サーバ側のプログラムにタイマールーチンを埋め込み、経過時間を計測した。簡単な2次元の流体モデルのため、CFD計算にはほとんど時間がかからず、1ステップ当たり0.1秒である。ピクセルデータ生成時間は、128×128で0.08秒、256×256で0.3秒である。

1フレーム当たりの画像データの転送時間、すなわち計算サーバからWebサーバへの転送、Webクライアントにおける画像データの読み込み・画像解凍・表示に要するトータルの時間が全実行時間のうちの大部分を占めている。ネットワーク環境は、図11に示すようなインターネットと、図12に示すような計算科学技術センター内イントラネットである。当然のことであるが、ネットワークは時間に依存して混んでいたり混んでいなかったりするので、計測結果はあくまでおおよその目安であることに注意されたい。

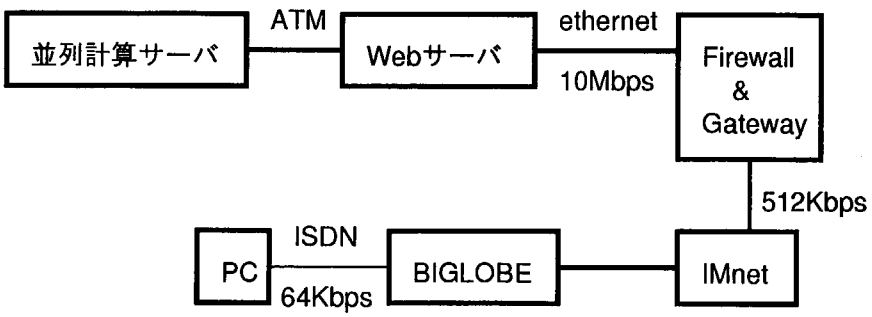


図 11: インターネット評価環境

図 12: 計算科学技術センター内イントラネット評価環境

表 3: インターネットにおける画像データ転送時間 (Internet Explorer)

ケース	転送時間（秒）
128×128、圧縮なし	6.51
128×128、圧縮あり	1.02
256×256、圧縮なし	27.55
256×256、圧縮あり	3.20

表 4: イントラネットにおける画像データ転送時間 (Internet Explorer)

ケース	転送時間（秒）
128×128、圧縮なし	0.46
128×128、圧縮あり	0.50
256×256、圧縮なし	1.46
256×256、圧縮あり	1.07

表 5: イントラネットにおける画像データ転送時間 (Netscape Navigator)

ケース	転送時間（秒）
128×128、圧縮なし	1.21
128×128、圧縮あり	3.49
256×256、圧縮なし	3.24
256×256、圧縮あり	15.84

クライアント側はPC-9821を用い、ピクセルサイズは128×128と256×256、画像圧縮を施した場合と施さない場合で比較した。またイントラネットでは、Internet ExplorerとNetscape Navigatorの両方のWebブラウザを比較した。

インターネット環境では、表3から明らかなように圧縮の効果が良く現われている。一方イントラネット環境では、表4から明らかなように圧縮の効果が現われていない。これはネットワークのデータ転送速度が十分に速いために、圧縮・非圧縮でその転送時間はほとんど変わらず、逆に圧縮の場合、解凍に要する時間の比率が大きくなり、圧縮した場合の方が転送時間がかかる。

またWebブラウザとしてInternet ExplorerとNetscape Navigatorを用いた場合と比較すると、Netscape Navigatorの方が転送時間が長くなる。これは、Internet Explorerの方がOS(Windows95)依存性が高いために、CPUの性能がより発揮されるからであると考えられる。

8 今後の課題

本システムは、現在のところ解析ソルバがシングルブロックのBFC格子にのみ対応している。しかし今後、より汎用性を高めるために、FEM/FVMなどの非構造格子やマルチブロック格子への拡張が必要である。

また前述のように、可視化処理は解析計算に比べてその負荷は比較的小さいと考え、Owner Computation Ruleに基づいて並列化を行なった。しかし高精細の画像表示すなわちピクセル数が多い場合や、計算格子数が小さい場合には状況が異なり、可視化処理と解析計算との負荷が同程度になる。このとき、視点や可視化領域に依存して、負荷のアンバランスが生ずる。このアンバランスの解消も課題の1つである。

最後に、現状の画像データ転送速度は256×256のピクセルサイズで1フレーム当たり3秒である。より高速な実時間可視化を実現するには、数値シミュレーション画像の特質を生かし、画質・圧縮率・速度的により高性能な画像圧縮のアルゴリズムの検討が必要である。

参考文献

[1] http://www.netlib.org/benchmark/top500/top500.list.html

[2] Chen, J.X. et al. ; Advancing Interactive Visualization and Coputational Steering, Computational Science & Engineering, Vol3,No.4, pp.13-17 (1996).

[3] Mulder, J.D. and van Wijk, J.J. ; 3D Computational Steering with Parametrized Geometric Objects, Proc. Visualization'95, IEEE Computer Soc. Press, pp.304-311 (1995).

[4] Haimes, R. and Giles, M. ; VISUAL3: Interactive Unsteady Unstructured 3D Visualization, AIAA Paper 91-0794 (1991).

[5] Haimes, R. ; pV3: A Distributed System for Large-Scale Unsteady CFD Visualization, AIAA Paper 94-0321 (1994).

[6] Jordan, K.E. et al. ; Parallel Interactive Visualization of 3D Mantle Convection, Computational Science & Engineering, Vol3,No.4, pp.29-37 (1996).

[7] Crockett, T.W. ; Design Consideration for Parallel Graphics Libraries, NASA Constractor Report 194935 (1994).

[8] Crockett, T.W. ; Parallel Rendering, in Encyclopedia of Computer Science and Technology, Kent, A. and Williams, J.G. eds., Vol.34,Suppl.19, A., Marcel Dekker, pp.335-371 (1996).

[9] Palmer, M.E. and Taylor, S. ; Rotation Invariant Partitioning for Concurrent Scientific Visualization, Proc. Parallel CFD'94, North-Holland, pp.409-416 (1995).

[10] 武井利文 ほか ; 流体解析のためのリアルタイムビジュアリゼーションシステム, 第7回数値流体力学シンポジウム講演論文集, pp.701-704 (1993).

[11] Doi, S. et al. ; A Real-time Visualization System for Computational Fluid Dynamics, NEC Research & Development, Vol.37,No.1, pp.114-123 (1996).

[12] 土肥俊 ほか；ハイパフォーマンスコンピュータのためのリアルタイム可視化, 情報処理学会研究報告, Vol.96,No.50, pp.21-26 (1996).

[13] 土肥俊 ほか；インターネット上でのCFDのためのリアルタイム可視化の可能性について, 第10回数値流体力学シンポジウム講演論文集, pp.22-23 (1996).

[14] 土肥俊 ほか；モバイル・ビジュアル・シミュレーション -PHSによるシミュレーション可視化-, 第11回数値流体力学シンポジウム講演論文集, pp.39-40 (1997).

[15] 村松一弘 ほか ； Cenju-3 上での流体解析のための実時間可視化システム, NEC 技報, Vol.48,No.12, pp.142-148 (1995).

[16] 村松一弘 ほか；Cenju-3 上での流体解析のための実時間可視化システム, 情報処理学会研究報告, Vol.96,No.50, pp.27-32 (1996).

[17] 村松一弘 ほか；並列計算機上での流体解析のための実時間可視化システム, 計算工学講演会論文集, Vol.2,No.1, pp.109-112 (1997).

[18] Levoy, M. ; Display of Surfaces from Volume Data, IEEE Computer Graphics, Vol.22,No.4, pp.355-356 (1991).

[19] 武井；ベクトル処理によるボリューム・レンダリングの高速化の研究, 第42回情報処理学会全国大会論文集 (2), pp.355-356 (1991).

[20] Phong, B.T. ; Illumination for Computer Generated Pictures, Comm. ACM, Vol.18,No.6, pp.311-317 (1975).

付録 A　　ライブラリ仕様

A.1　マルチブロック対応制御ルーチンの仕様

本システムは基本的に、初期化、格子タイプ対応、メイン、終了化の4ルーチンを呼び出すことによって実時間可視化が実現できる。これら4ルーチンを並列流体解析を行なっている各PEで呼び出す必要がある。並列流体ソルバの処理と制御ルーチン呼び出し位置の関係は図13のようになる。

A.1.1　RVS_INIT

1. 機能
   可視化機能の初期化、および画像合成用モジュールの起動
2. 呼び出し形式

```
CALL RVS_INIT(NS,NV,LABEL,IERR)
```

3. 引数

引数名	サイズ	入出力	内容
NS	I4	入力	スカラデータ数
NV	I4	入力	ベクトルデータ数
LABEL	A*15*(NS+NV)	入力	物理量種別名
IERR	I4	出力	エラーコード

4. 補足説明
   LABELの内容によりシステムで物理量種が区別されるため、LABELは空白であってはならない。

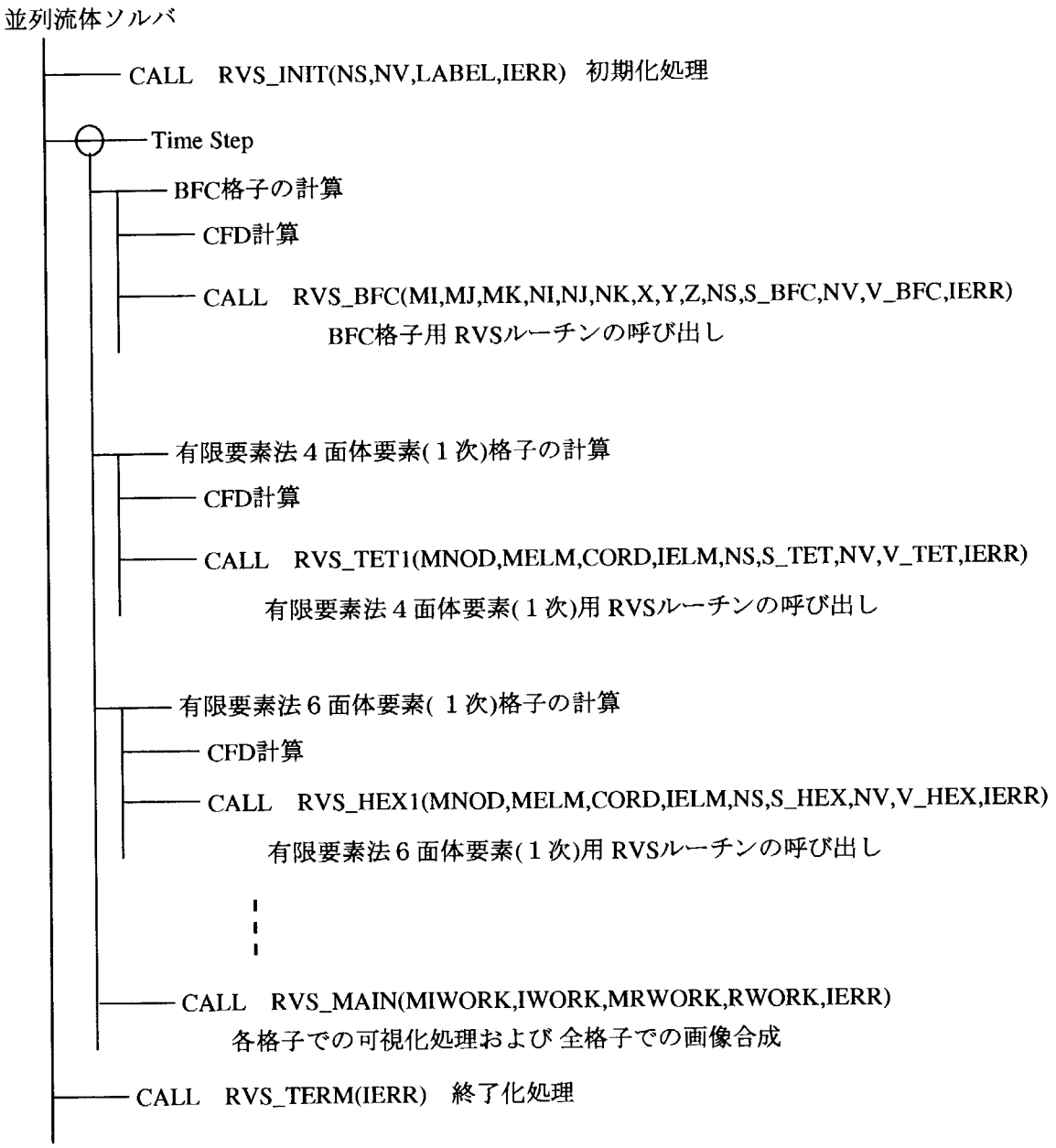


図 13: 並列版マルチブロック対応制御ルーチン呼び出し構造

A.1.2 RVS_BFC

1. 機能

BFC 格子における、格子データおよび解析データの **RVS_MAIN** への引き渡し

2. 呼び出し形式

```
CALL RVS_BFC(MI,MJ,MK,NI,NJ,NK,X,Y,Z,NS,S_BFC,NV,V_BFC,IERR)
```

3. 引数

引数名	サイズ	入出力	内容
MI	I4	入力	各 PE ごとの、I 軸方向整合寸法
MJ	I4	入力	各 PE ごとの、J 軸方向整合寸法
MK	I4	入力	各 PE ごとの、K 軸方向整合寸法
NI	I4	入力	各 PE ごとの、I 軸方向の格子点数
NJ	I4	入力	各 PE ごとの、J 軸方向の格子点数
NK	I4	入力	各 PE ごとの、K 軸方向の格子点数
X	R8*MI*MJ*MK	入力	格子点の x 座標値
Y	R8*MI*MJ*MK	入力	格子点の y 座標値
Z	R8*MI*MJ*MK	入力	格子点の z 座標値
NS	I4	入力	スカラデータ数
S_BFC	R8*(MI*MJ*MK,NS)	入力	スカラデータ
NV	I4	入力	ベクトルデータ数
V_BFC	R8*(MI*MJ*MK,3,NV)	入力	ベクトルデータ
IERR	I4	出力	エラーコード

4. 補足説明

- X,Y,Z,S_BFC,V_BFC
  最初に I 軸方向が、次に J 軸方向が、最後に K 軸方向が変わる順番に連続して値を格納する。また並列化はセルに対する領域分割法に基づいており、PE 間の分割領域の境界上の格子点データは、双方の PE で共有している。
- 解析ソルバでの配列使用方法が **RVS_BFC** と違う場合は、**RVS_BFC** を呼ぶ前にデータの並び換えを行なう必要がある。

A.1.3 RVS_TET1

1. 機能
   有限要素法格子（4面体1次要素）における、格子データおよび解析データの**RVS_MAIN**への引き渡し
2. 呼び出し形式

```
CALL RVS_TET1(MNOD,MELM,CORD,IELM,NS,S_TET1,NV,V_TET1,IERR)
```

3. 引数

引数名	サイズ	入出力	内容
MNOD	I4	入力	各PEごとの、節点データの配列サイズ
MELM	I4	入力	各PEごとの、要素データの配列サイズ
CORD	I4*MNOD*3	入力	節点のx,y,z座標値
IELM	I4*MELM*4	入力	各要素の構成節点番号
NS	I4	入力	スカラデータ数
S_TET1	R8*MNOD*NS	入力	スカラデータ
NV	I4	入力	ベクトルデータ数
V_TET1	R8*MNOD*NV*3	入力	ベクトルデータ
IERR	I4	出力	エラーコード

4. 補足説明
   - CORD,S_TET1,V_TET1
     各PEごとの節点番号順に値を格納する。節点番号は各PEごとに独立に割り当てる。
   - IELM
     IELMは各要素の構成節点番号であり、要素番号順に要素を構成する節点番号を、4個づつ格納する。要素番号は各PEごとに独立に割り当てる。
   - 解析ソルバでの配列使用方法が**RVS_TET1**と違う場合は、**RVS_TET1**を呼ぶ前にデータの並び換えを行なう必要がある。

A.1.4 RVS_HEX1

1. 機能

有限要素法格子（6面体1次要素）における、格子データおよび解析データの**RVS_MAIN**への引き渡し

2. 呼び出し形式

```
CALL RVS_HEX1(IG,IDD,NCOMM,
              MNOD,MELM,CORD,IELM,NS,S_HEX1,NV,V_HEX1,IERR)
```

3. 引数

引数名	サイズ	入出力	内容
MNOD	I4	入力	各PEごとの、節点データの配列サイズ
MELM	I4	入力	各PEごとの、要素データの配列サイズ
CORD	I4*MNOD*3	入力	節点のx,y,z座標値
IELM	I4*MELM*6	入力	各要素の構成節点番号
NS	I4	入力	スカラデータ数
S_HEX1	R8*MNOD*NS	入力	スカラデータ
NV	I4	入力	ベクトルデータ数
V_HEX1	R8*MNOD*NV*3	入力	ベクトルデータ
IERR	I4	出力	エラーコード

4. 補足説明

- CORD,S_HEX1,V_HEX1
  各PEごとの節点番号順に値を格納する。節点番号は各PEごとに独立に割り当てる。
- IELM
  IELMは各要素の構成節点番号であり、要素番号順に要素を構成する節点番号を、6個づつ格納する。要素番号は各PEごとに独立に割り当てる。
- 解析ソルバでの配列使用方法が**RVS_HEX1**と違う場合は、**RVS_HEX1**を呼ぶ前にデータの並び換えを行なう必要がある。

A.1.5 RVS_MAIN

1. 機能
   可視化処理および表示の制御を行なう
2. 呼び出し形式

```
CALL RVS_MAIN(NIWORK,IWORK,NRWORK,RWORK,IERR)
```

3. 引数

引数名	サイズ	入出力	内容
NIWORK	I4	入力	整作業領域配列サイズ
IWORK	I4*NIWORK	出力	整作業領域
NRWORK	I4	入力	実作業領域配列サイズ
RWORK	R8*NRWORK	出力	実作業領域
IERR	I4	出力	エラーコード

4. 補足説明

A.1.6 RVS_TERM

1. 機能
   可視化機能の終了化
2. 呼び出し形式

```
CALL RVS_TERM(IERR)
```

3. 引数

引数名	サイズ	入出力	内容
IERR	I4	出力	エラーコード

4. 補足説明

A.2 マルチブロック対応ユーザ定義ルーチンの仕様

ユーザ定義ルーチンは、実時間可視化機能が一般のソルバに容易に組み込めるようにするために用意されているものである。これらのルーチンをユーザがプログラミングすることにより、必要なデータや処理内容を制御ルーチンに供給することができる。

具体的な開発手順は、プログラミングしたユーザ定義ルーチンを含むファイルをコンパイルし、他のオブジェクトファイルとリンクすれば良い。

A.2.1 RVS_USER_INIT

1. 機能
   GUIおよび画像データ中継モジュールの起動用
2. 呼び出し形式

```
CALL RVS_USER_INIT()
```

3. 引数
4. 利用方法
   ユーザの利用環境に合わせて起動コマンドを記述する。
5. 補足説明
   このルーチンを使用しない場合は、ユーザのソルバで**RVS_INIT**を呼び出す前に、GUIおよび画像データ中継モジュールを起動させておく。

A.2.2 RVS_USER_BFC

1. 機能
   BFC格子における、領域分割データ供給
2. 呼び出し形式

```
CALL RVS_USER_BFC(MPI,MPJ,MPK,NPI,NPJ,NPK,NCOMM)
```

3. 引数

引数名	サイズ	入出力	内容
MPI	I4	入力	I軸方向のPE数 ($\geq$1)
MPJ	I4	入力	J軸方向のPE数 ($\geq$1)
MPK	I4	入力	K軸方向のPE数 ($\geq$1)
NPI	I4	入力	I軸方向のPEの位置 (0$\leq$NPI$\leq$MPI-1)
NPJ	I4	入力	J軸方向のPEの位置 (0$\leq$NPJ$\leq$MPJ-1)
NPK	I4	入力	K軸方向のPEの位置 (0$\leq$NPK$\leq$MPK-1)
NCOMM	I4	入力	本ブロックを構成するPE群のコミュニケータ

4. 利用方法
   (a) NPI,NPJ,NPKは、図14のように格納する。
5. 補足説明

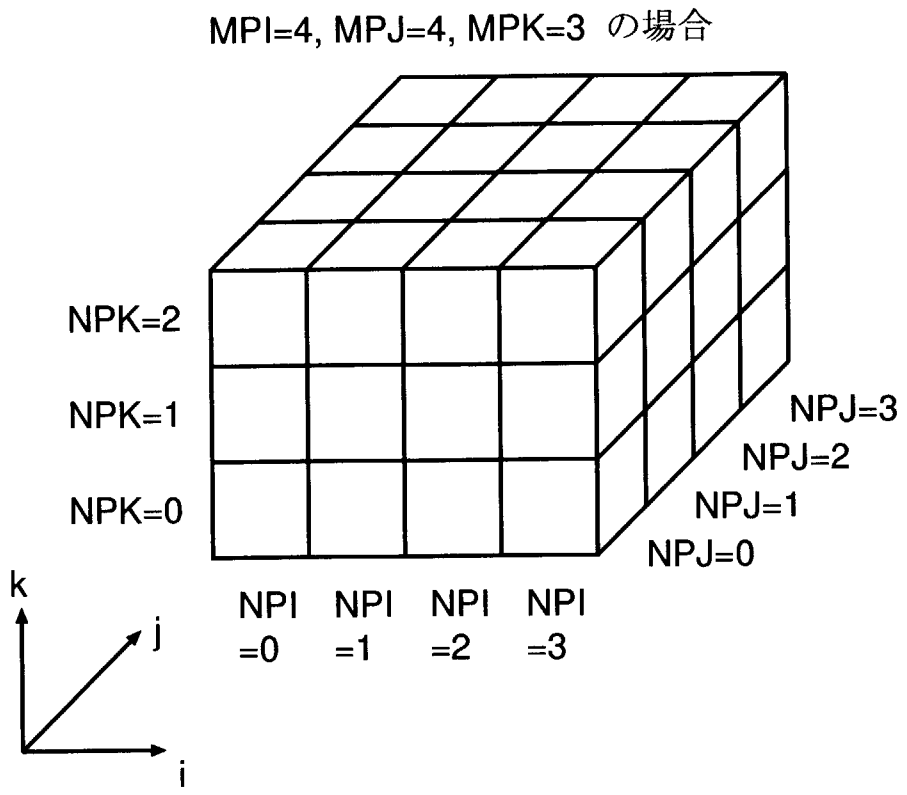


図 14: I,J,K の各軸方向の PE の位置 NPI,NPJ,NPK の格納方法

A.2.3 RVS_USER_TET1

1. 機能

有限要素法格子 (4 面体 1 次要素) における、領域分割データ供給

2. 呼び出し形式

```
CALL RVS_USER_TET1(MNOD,MNPE,NODDD,NUMDD,NPEDD,NODPDD,NCOMM)
```

3. 引数

引数名	サイズ	入出力	内容
MNOD	I4	入力	NODDD,NUMDD の整合寸法
MNPE	I4	入力	NPEDD,NODPDD の整合寸法
NODDD	I4*MNOD	入力	分割領域間の境界上の節点番号テーブル
NUMDD	I4*MNOD	入力	分割領域間の境界上の節点を共有する PE 数-1 ($\geq$1)
NPEDD	I4*MNPE	入力	分割領域間の境界上の節点を共有する PE 群のランク
NODPDD	I4*MNPE	入力	共有する各 PE での節点番号
NCOMM	I4	入力	本ブロックを構成する PE 群のコミュニケータ

4. 利用方法

(a) NODDD,NUMDD,NPEDD,NODPDD は、図 15 のような 2 次元の場合、以下のように格納する。

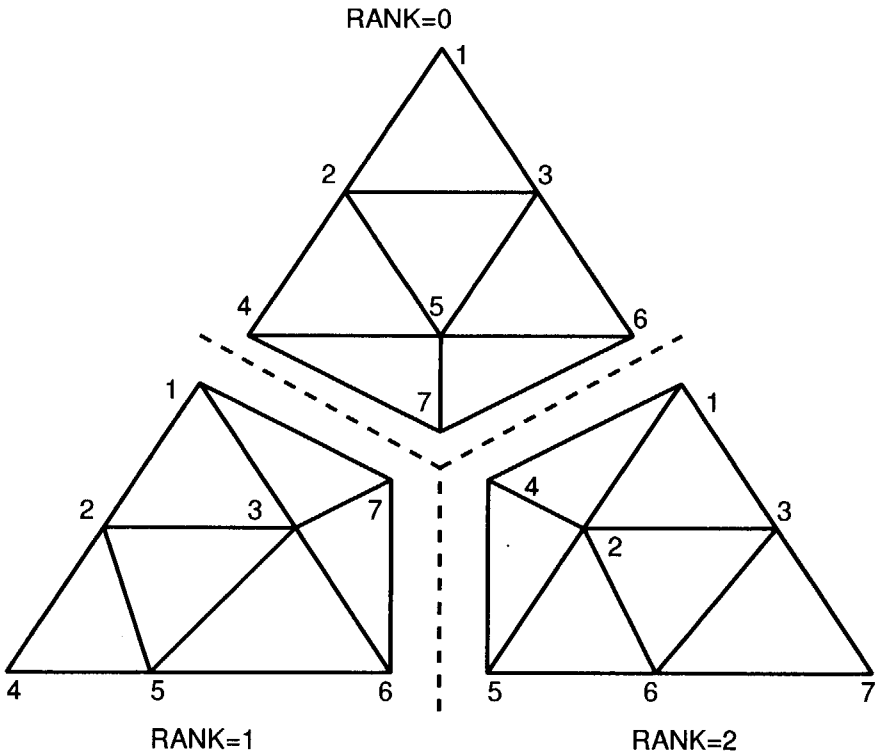


図 15: 有限要素法 3 角形要素 (1 次) の領域分割

RANK=0 では、

NODDD	4	6	7	
NUMDD	1	1	2	
NPEDD	1	1	1	2
NODPDD	1	1	7	4

RANK=1 では、

NODDD	1	6	7	
NUMDD	1	1	2	
NPEDD	0	2	0	1
NODPDD	4	5	7	4

RANK=2 では、

NODDD	1	4	5	
NUMDD	1	2	1	
NPEDD	0	0	1	1
NODPDD	6	7	7	6

5. 補足説明

A.2.4 RVS_USER_HEX1

1. 機能
   有限要素法格子 (6 面体 1 次要素) における、領域分割データ供給
2. 呼び出し形式

```
CALL RVS_USER_HEX1(MNOD,MNPE,NODDD,NUMDD,NPEDD,NODPDD,NCOMM)
```

3. 引数

引数名	サイズ	入出力	内容
MNOD	I4	入力	NODDD,NUMDD の整合寸法
MNPE	I4	入力	NPEDD,NODPDD の整合寸法
NODDD	I4*MNOD	入力	分割領域間の境界上の節点番号テーブル
NUMDD	I4*MNOD	入力	分割領域間の境界上の節点を共有する PE 数-1 ($geq1$)
NPEDD	I4*MNPE	入力	分割領域間の境界上の節点を共有する PE 群のランク
NODPDD	I4*MNPE	入力	共有する各 PE での節点番号
NCOMM	I4	入力	本ブロックを構成する PE 群のコミュニケータ

4. 利用方法
   (a) NODDD,NUMDD,NPEDD,NODPDD のデータ格納方法は、**RVS_USER_TET1** と同様である。
5. 補足説明

A.2.5 RVS_USER_OBJECT

1. 機能
   オブジェクト表示のためのオブジェクトデータ供給
2. 呼び出し形式

```
CALL RVS_USER_OBJECT(MFLAG,NO,LO)
```

3. 引数

引数名	サイズ	入出力	内容
MFLAG	I4	出力	データ変更フラグ (0:変更なし、1: 変更あり)
NO	I4	出力	オブジェクト数
LO	I4	出力	オブジェクトを表す格子点の値

4. 利用方法
   (a) 流体ソルバ側でオブジェクトに関する変数および配列を COMMON 化する。
   (b) **RVS_USER_OBJECT** 内でも上の COMMON 変数を宣言する。
   (c) **RVS_USER_OBJECT** 内で上記の引数内容に合わせてオブジェクトデータを変換する。
5. 補足説明
   - オブジェクトの表示機能を使用しない場合はプログラミングしなくても良い。

A.2.6 RVS_USER_TRACER

1. 機能
   トレーサ表示のための BFC 格子における切断面の位置データ供給
2. 呼び出し形式

```
CALL RVS_USER_TRACER(AREA1,AREA2)
```

3. 引数

引数名	サイズ	入出力	内容
AREA1	I4*6	出力	切断面の位置 (IS1,JS1,KS1,IE1,JE1,KE1)
AREA2	I4*6	出力	切断面の位置 (IS2,JS2,KS2,IE2,JE2,KE2)

4. 利用方法
   (a) 流体ソルバ側で切断面の位置に関する配列を COMMON 化する。
   (b) **RVS_USER_TRACER** 内でも上の COMMON 変数を宣言する。
   (c) **RVS_USER_TRACER** 内で上記の引数内容に合わせて切断面の位置データを変換する。
5. 補足説明
   - 配列 AREA1 では、IS1≤IE1,JS1≤JE1,KS1≤KE1 を満たすようにする。
   - 配列 AREA2 では、(IS1,JS1,KS1) と (IS2,JS2,KS2) が物理空間で一致するようにする。(IE1,JE1,KE1) と (IE2,JE2,KE2) も同様。
   - 切断面が存在しない場合、全てのデータに 0 を入れておく。

A.2.7 RVS_USER_TIME

1. 機能
   時間刻み幅供給
2. 呼び出し形式

```
CALL RVS_USER_TIME(DT)
```

3. 引数

引数名	サイズ	入出力	内容
DT	R8	出力	時間刻み幅Δ

4. 利用方法
   (a) 流体ソルバ側で時間刻み幅に関する配列を COMMON 化する。
   (b) **RVS_USER_TIME** 内でも上の COMMON 変数を宣言する。
   (c) **RVS_USER_TIME** 内で上記の引数内容に合わせて時間刻み幅データを変換する。
5. 補足説明

This is a blank page.

国際単位系 (SI) と換算表

表1　SI基本単位および補助単位

量	名称	記号
長さ	メートル	m
質量	キログラム	kg
時間	秒	s
電流	アンペア	A
熱力学温度	ケルビン	K
物質量	モル	mol
光度	カンデラ	cd
平面角	ラジアン	rad
立体角	ステラジアン	sr

表2　SIと併用される単位

名称	記号
分，時，日	min，h，d
度，分，秒	°，′，″
リットル	l，L
トン	t
電子ボルト	eV
原子質量単位	u

1 eV = 1.60218 × 10^{-19} J

1 u = 1.66054 × 10^{-27} kg

表3　固有の名称をもつSI組立単位

量	名称	記号	他のSI単位による表現
周波数	ヘルツ	Hz	s^{-1}
力	ニュートン	N	$m \cdot kg/s^2$
圧力，応力	パスカル	Pa	N/m^2
エネルギー，仕事，熱量	ジュール	J	N·m
工率，放射束	ワット	W	J/s
電気量，電荷	クーロン	C	A·s
電位，電圧，起電力	ボルト	V	W/A
静電容量	ファラド	F	C/V
電気抵抗	オーム	Ω	V/A
コンダクタンス	ジーメンス	S	A/V
磁束	ウェーバ	Wb	V·s
磁束密度	テスラ	T	Wb/m^2
インダクタンス	ヘンリー	H	Wb/A
セルシウス温度	セルシウス度	℃	
光束	ルーメン	lm	cd·sr
照度	ルクス	lx	lm/m^2
放射能	ベクレル	Bq	s^{-1}
吸収線量	グレイ	Gy	J/kg
線量当量	シーベルト	Sv	J/kg

表4　SIと共に暫定的に維持される単位

名称	記号
オングストローム	Å
バーン	b
バール	bar
ガル	Gal
キュリー	Ci
レントゲン	R
ラド	rad
レム	rem

1 Å = 0.1 nm = 10^{-10} m

1 b = 100 fm^2 = 10^{-28} m^2

1 bar = 0.1 MPa = 10^5 Pa

1 Gal = 1 cm/s^2 = 10^{-2} m/s^2

1 Ci = 3.7 × 10^{10} Bq

1 R = 2.58 × 10^{-4} C/kg

1 rad = 1 cGy = 10^{-2} Gy

1 rem = 1 cSv = 10^{-2} Sv

表5　SI接頭語

倍数	接頭語	記号
10^{18}	エクサ	E
10^{15}	ペタ	P
10^{12}	テラ	T
10^{9}	ギガ	G
10^{6}	メガ	M
10^{3}	キロ	k
10^{2}	ヘクト	h
10^{1}	デカ	da
10^{-1}	デシ	d
10^{-2}	センチ	c
10^{-3}	ミリ	m
10^{-6}	マイクロ	μ
10^{-9}	ナノ	n
10^{-12}	ピコ	p
10^{-15}	フェムト	f
10^{-18}	アト	a

（注）

1. 表1－5は「国際単位系」第5版，国際度量衡局 1985年刊行による。ただし，1 eVおよび1 uの値はCODATAの1986年推奨値によった。
2. 表4には海里，ノット，アール，ヘクタールも含まれているが日常の単位なのでここでは省略した。
3. barは，JISでは流体の圧力を表わす場合に限り表2のカテゴリーに分類されている。
4. EC閣僚理事会指令ではbar，barnおよび「血圧の単位」mmHgを表2のカテゴリーに入れている。

換算表

力	N(=10^5 dyn)	kgf	lbf
	1	0.101972	0.224809
	9.80665	1	2.20462
	4.44822	0.453592	1

粘度　1 Pa·s(N·s/m^2) = 10 P(ポアズ)(g/(cm·s))

動粘度　1 m^2/s = 10^4 St(ストークス)(cm^2/s)

圧力	MPa(=10 bar)	kgf/cm^2	atm	mmHg(Torr)	lbf/in^2(psi)
	1	10.1972	9.86923	7.50062 × 10^3	145.038
	0.0980665	1	0.967841	735.559	14.2233
	0.101325	1.03323	1	760	14.6959
	1.33322 × 10^{-4}	1.35951 × 10^{-3}	1.31579 × 10^{-3}	1	1.93368 × 10^{-2}
	6.89476 × 10^{-3}	7.03070 × 10^{-2}	6.80460 × 10^{-2}	51.7149	1

エネルギー・仕事・熱量	J(=10^7 erg)	kgf·m	kW·h	cal(計量法)	Btu	ft·lbf	eV
	1	0.101972	2.77778 × 10^{-7}	0.238889	9.47813 × 10^{-4}	0.737562	6.24150 × 10^{18}
	9.80665	1	2.72407 × 10^{-6}	2.34270	9.29487 × 10^{-3}	7.23301	6.12082 × 10^{19}
	3.6 × 10^6	3.67098 × 10^5	1	8.59999 × 10^5	3412.13	2.65522 × 10^6	2.24694 × 10^{25}
	4.18605	0.426858	1.16279 × 10^{-6}	1	3.96759 × 10^{-3}	3.08747	2.61272 × 10^{19}
	1055.06	107.586	2.93072 × 10^{-4}	252.042	1	778.172	6.58515 × 10^{21}
	1.35582	0.138255	3.76616 × 10^{-7}	0.323890	1.28506 × 10^{-3}	1	8.46233 × 10^{18}
	1.60218 × 10^{-19}	1.63377 × 10^{-20}	4.45050 × 10^{-26}	3.82743 × 10^{-20}	1.51857 × 10^{-22}	1.18171 × 10^{-19}	1

1 cal = 4.18605 J (計量法)
= 4.184 J (熱化学)
= 4.1855 J (15 °C)
= 4.1868 J (国際蒸気表)

仕事率　1 PS (仏馬力)
= 75 kgf·m/s
= 735.499 W

放射能	Bq	Ci
	1	2.70270 × 10^{-11}
	3.7 × 10^{10}	1

吸収線量	Gy	rad
	1	100
	0.01	1

照射線量	C/kg	R
	1	3876
	2.58 × 10^{-4}	1

線量当量	Sv	rem
	1	100
	0.01	1

(86年12月26日現在)

並列計算機上での流体解析のための実時間可視化システムの開発