卒業論文

PC クラスタ上での有限要素法による カルマン渦の並列化

氏名　　：　林　雅樹
学籍番号　：　2210990182-6
指導教員　：　山崎　勝弘　教授

提出日　　：　2003 年 2 月 21 日

立命館大学　理工学部　情報学科

内容梗概

汎用のワークステーション・パーソナルコンピュータをネットワーク結合したＰＣクラスタが急速に広まっている。ＰＣクラスタは、スーパーコンピュータ並の高速計算処理能力を有し、最先端情報処理技術に欠かせない重要な基盤技術となっており、今までスーパコンピュータを必要とした応用分野以外の新しい応用分野が開拓されている。

近年では、共有メモリ計算機の普及に伴い、並列プログラミングも分散メモリ環境から、共有メモリ環境へと移行しつつある。その共有メモリ用のプログラミングモデルとして現在注目を集めているのがOpenMP である。

本論文では、有限要素法によるカルマン渦の流体解析プログラムを作成し、そのプログラムを用いてSCore型ＰＣクラスタ上による並列化を行ったことについて述べる。有限要素法によるカルマン渦の並列化は、並列処理部分の速度向上が得られたが、理想的な速度向上ではなかった。また、全体処理の時間は速度向上と程遠い結果となった。その原因に、圧力計算のICCG法の逐次性が高いことにあった。

目次

図目次

表目次

1. はじめに

並列処理の研究の応用分野として気象予測、環境問題、流体計算、デジタル画像処理、遺伝子の解明、データマイニングの並列化などが挙げられる。特に 2002 年 4 月に完成した地球シミュレータによって、地球全体の異常気象のシミュレーション、地球変動の予測を並列処理で研究がされており、一層の並列処理の研究が進められることであろう。[12]

流体計算は気象予測、航空機設計・エンジニアリング、自動車設計・エンジニアリングなどに用いられている。流体計算は計算時間が大きいため並列化すべき対象である。地球シミュレータでも流体計算を問題とした実験も行われている。

並列計算機の種類として、複数のプロセッサがメモリバス/スイッチ経由で主記憶に接続された共有メモリモデル、プロセッサと主記憶から構成されたシステムが複数個互いに接続された形態で、プロセッサはほかのプロセッサの主記憶に読み書きができる共有分散メモリモデル、プロセッサと主記憶から構成されたシステムが複数個互いに接続された形態で、プロセッサは他のプロセッサの主記憶の読み書きができない、分散メモリモデルの３タイプある。近年、汎用の PC を高速のネットワークで結合した PC クラスタが急速に広まっている。PC クラスタには、米国 NASA のゴダード研究所で開発された Beowulf 型 PC クラスタと、日本の新情報処理開発機構（RWCP）並列分散システムソフトウェアつくば研究所で開発された SCore 型クラスタがある。PC クラスタの長所として、プロセッサの数に比例して実行速度が向上する、大規模な計算が可能、コストパフォーマンスが良い点が挙げられる。[1]

本研究では、16 台の PC を Myrinet（Myrinet2000）と Ethernet（100Base-TX）で接続し、PC クラスタシステムソフトウェアに SCore を用いて構築された PC クラスタを用いて、カルマン渦を対象とした流体計算の並列化を行い考察する。

流体とは空気や水など微小な力により形が変形するものを言う。流体力学はこの流体を空間的、時間的スケールで観察し、流体がどのように運動しているか解析することである。本研究で解析対象とした流体はカルマン渦である。カルマン渦とは、流れの中にある円柱構造物の周りにできる渦のことを言う。現象例として、風になびく旗や、水を張った水槽に棒を入れ水平に移動させることに起きる振動などが挙げられる。

流体解析において用いられる方程式がナビエ・ストークス方程式である。この方程式は、ニュートン力学における運動方程式の第 2 方程式を表していて、流体に加わる力によって流体がどのように運動するかを記述した式である。また、ナビエ・ストークス方程式は非線形性を持っているため、解析的に解くことは非常に限られた場合を除いては一般に難しい。よってナビエ・ストークス方程式を数値的アプローチによって解く有限要素法を用いて近似的に解析する。

有限要素法は、解析領域を有限要素と呼ばれる部分領域に分割し、要素ごとに対象の式から剛性方程式と呼ばれる連立 1 次方程式を導き出し、全要素の剛性方程式を重ね合わせ

ることによって最終的に解くべき連立 1 次方程式を組み立てる。この連立 1 次方程式を解くことで求める解を近似解として得られることができる。

有限要素法によるカルマン渦の解析の並列化を行う。流体解析は空間的、時間的スケールで解析する。よって空間的スケールが並列処理させる部分に当たる。並列化アルゴリズムとして、全要素を計算ステージとして複数に分割し、各プロセッサを割り当て処理させるパイプライン処理で行う。

本論文では、2 章で PC クラスタと並列プログラミグ、OpenMP についての概念を説明し、3 章で有限要素法によるカルマン渦の流体解析の説明、4 章で有限要素法の並列化について述べる。

2. 並列処理と OpenMP

2.1. PC クラスタ

PC クラスタとは、PC 単体では性能的にはスーパーコンピュータに及ばないものの、複数の PC をネットワークで接続し、仮想的に 1 台の並列コンピュータとして利用することでパフォーマンスを向上させた PC 群を言う。PC クラスタは 2～8 台の小規模な PC クラスタから、数千台規模でスーパーコンピュータの性能を発揮する PC クラスタを構成することが可能である。また近年の PC の高性能化、低価格化に伴うコストパフォーマンスの向上により PC クラスタの構成もコストパフォーマンスの向上が図られた。ネットワークにおいても 100M Ethernet から Gigabit Ethernet や Myrinet などの高速ネットワークの登場で PC の高性能化に対するバランスが取れるようになり、大規模な PC クラスタで発生するネットワークのオーバヘッドを最小限に抑えることが可能となった。第 20 回 TOP500 において Linux NetworX の開発した PC クラスタが第 5 位に入り PC クラスタが今まで以上に注目をされている。[1][7][8][9]

2.2. 研究室の PC クラスタ

本研究室の PC クラスタは Server Host 1 台、Compute Host 16 台で構成されている。図 2 に本研究室の PC クラスタの構成図を示す。PC を結合するネットワークは、Myrinet（Myrinet2000）と Ethernet（100Base-TX）である。

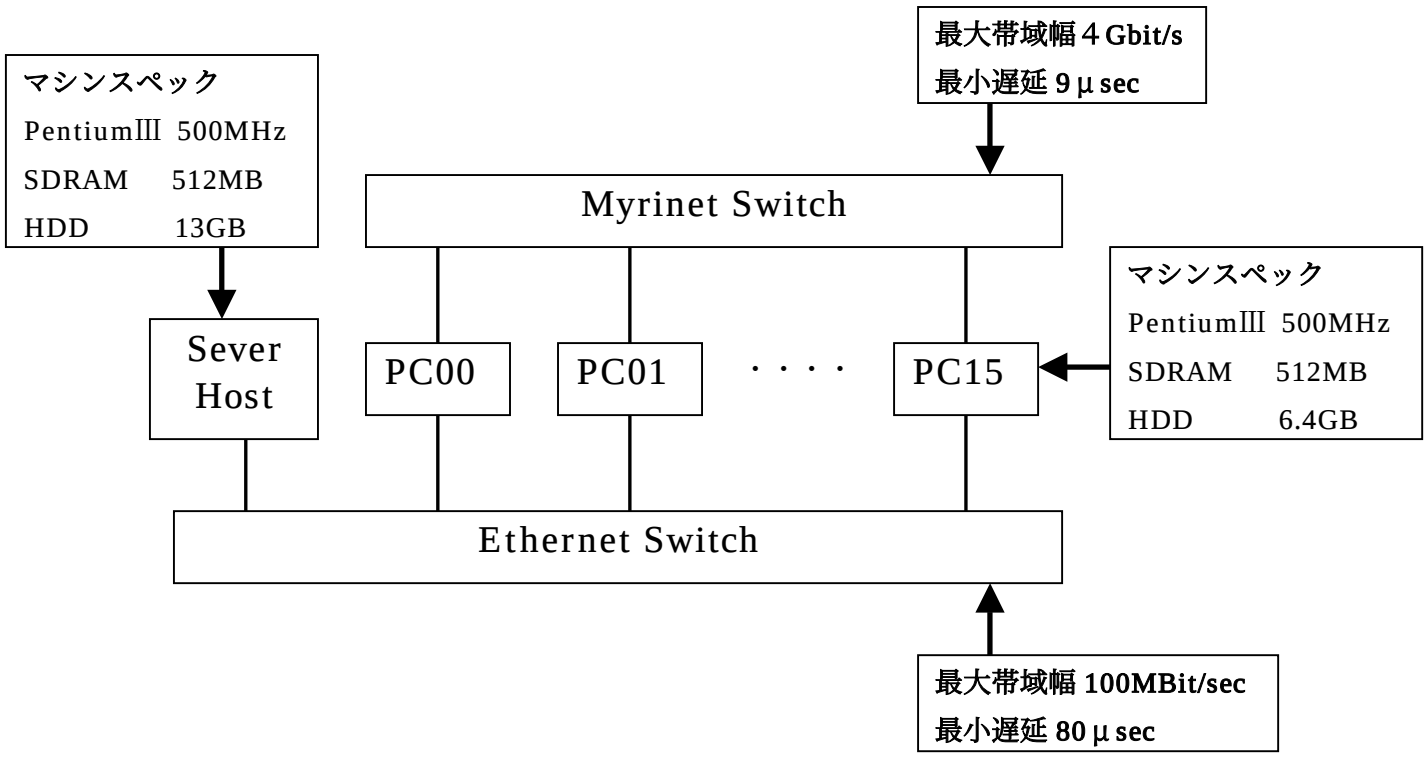


図 1　PC クラスタの構成

各 PC は OS に Redhat Linux 7.3 がインストールされ、PC クラスタシステムソフトウェアに SCore 5.2.0 がインストールされている。

PC クラスタシステムソフトウェアとして代表的な Beowulf は 1994 年に NASA の Earth and space science プロジェクトのため CESDIS（Center of Excellense in Space Data and Informantion Scienses）で開発された PC クラスタシステムソフトウェアとして初めて登場したソフトウェアである。Beowulf は TCP/IP プロトコル上のフリーウェアの組み合わせでクラスタシステムを構築する。しかし、TCP/IP プロトコルを用いると、何階層ものプロトコル変換が行われるため応用プログラムの多様な資源の割り当て要求に対応することが難しい。このオーバーヘッドを減らすためにユーザーレベルから直接ネットワークハードウェアが制御できるゼロコピー通信ソフトウェアを搭載した PC クラスタシステムソフトウェアの SCore が新情報処理開発機構（RWCP）並列分散システムソフトウェアつくば研究所で開発された。SCore クラスタシステムソフトウェアの階層図を図１に示す。

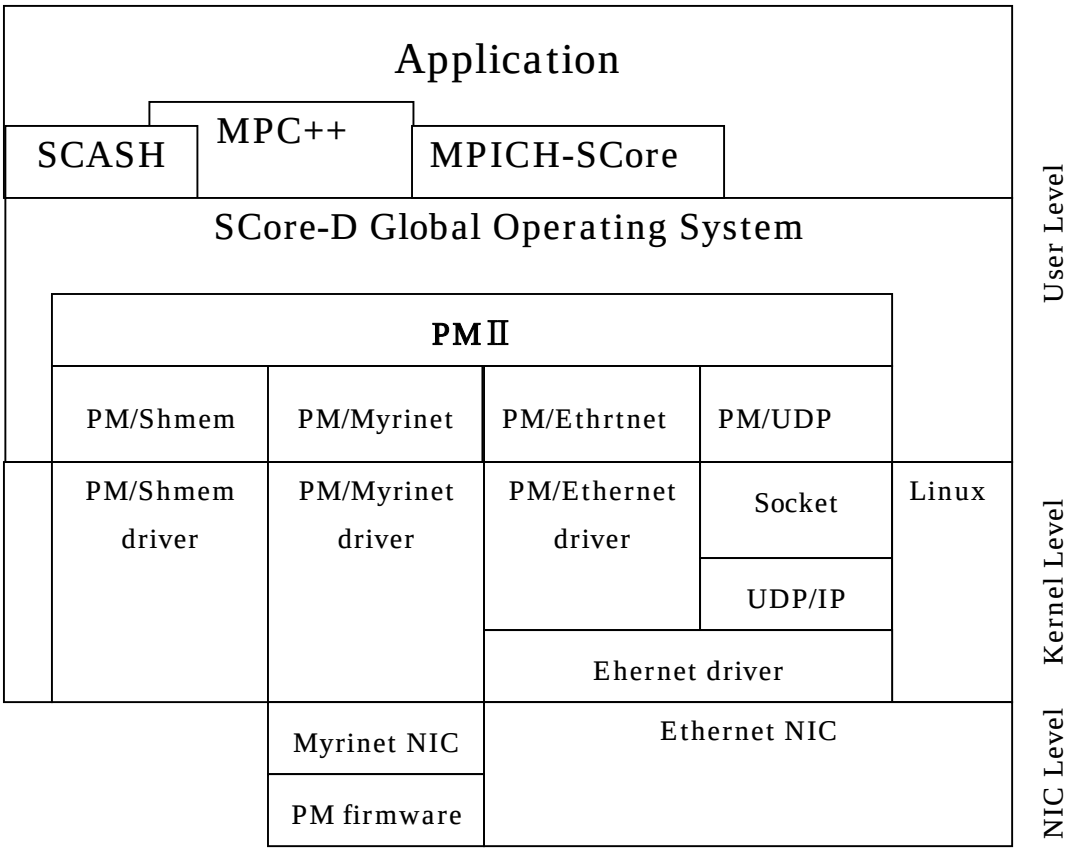


図 2　SCore のソフトウェア階層

SCore を構成している幾つかのコンポーネントについて説明する。

PMⅡはクラスタコンピューティング用低レベル通信ライブラリ。PMⅡAPI はクラスタコンピューティングに おいて多くの種類のネットワークや共有メモリに同一の方法でアクセスできるように設計されている。 Myrinet、Ethernet、UDP、Shmem 用の PMⅡ ドライバが実装済みである。SCore-D はプロセッサやネットワークのような、クラスタのリソースを利用するユーザ レベルのグローバル オペレーティング システムである。プロセッサやネットワークのような、クラスタのリソースを利用するユーザ レベルのグローバルオペレーティング システムである。種々の PM ネットワークデバイスは SCore-D によって管理され利用される。SMP クラスタや異種混在クラスタも単一種クラスタと同様にサポートされる。SCASH は分散メモリの PC クラスタ上で共有メモリプログラミングの環境を実現するためのソフトウェア DSM（Distributed Shared Memory）システムである。

2.3. 並列プログラミング

並列プログラミグをするにあたって、並列プログラミグを記述する方法として大きく２つに分けることができる。１つは分散メモリプログラミグ、もう１つは共有メモリプログラミグである。分散メモリプログラミングとして挙げられるのは PVM、MPI のメッセージパッシング型のプログラミグである。共有メモリプログラミグとして挙げられるのは OpenMP の従来のプログラミング言語にコンパイラ指示文として並列処理の指示を挿入するプログラミングである。

（1）PVM（Parallel Virtual Machine）

PVM は米国のオークリッジ国立研究所を中心に開発された、メッセージパッシングによる並列計算を行うためのソフトウェアであり、動作するマシンの種類が多く、フリーで手に入るソフトウェアで広く利用されるようになってきている。PVM は Parallel Virtual Machine つまり仮想並列計算機を意味している。

PVM は、ネットワークに接続された異機種 UNIX コンピュータ群を、単一の並列コンピュータとして利用することを可能にするソフトウェアである。これにより、多数のコンピュータの持つ計算パワーを１つの大規模計算問題に結集して処理を行うことができる。

PVM は、アプリケーション、マシン及びネットワークレベルでの異機種間利用をサポートする。言い替えると、PVM の下では、アプリケーションを構成するタスクは、問題に最も適したアーキテクチャを利用することができる。PVM は、異なるコンピュータ間の整数あるいは浮動小数点の表現の違いを吸収するための、データ変換を扱うことができる。そして、PVM は多様なネットワークで接続されたバーチャルマシンを実現する。

PVM ソフトウェアシステムの構成は、大きく２つに分けられる。１つはデーモンである。デーモンは、バーチャルマシンを構成するすべてのコンピュータ上に常駐する。ユーザは、ログイン可能でさえあればどんなコンピュータにも、PVM をインストールすることができ

る。PVM アプリケーションを実行する場合、最初にユーザはどれか１つのコンピュータで PVM を起動する。次にユーザが定義したバーチャルマシンを構成するコンピュータそれぞれにおいて順次 PVM デーモンを起動する。最後にどれか１つのコンピュータに表示された UNIX プロンプトに対してコマンドを入力することにより PVM アプリケーションを実行する。

PVM ソフトウェアシステムを構成するもう１つは、PVM インターフェース関数のライブラリである。ライブラリはメッセージパッシング、プロセスの生成、タスクの協調、及びバーチャルマシンの再構成のための関数を提供する。PVM を利用するためには、アプリケーションプログラムと、このライブラリを必ずリンクする必要がある。[3]

（2）MPI（Message Passing Interface）

MPI は Message Passing Interface を意味し、1992 年に米国の企業、研究所、大学などが中心となって設立された MPI Forum により、メッセージ通信のプログラムを記述するために広く使われる標準化を目指して作られたメッセージ通信の API 仕様である。仕様は 1994 年に MPI1.0 がまとめられえ、その後、並列 I/O やプロセスの動的生成・消滅、1 方向通信などの機能をさらに追加した MPI2.0 の言語仕様が 1997 年に定められた。MPI はネットワーク上のプロセス間の効率の良い通信が可能であること、異なるプラットフォーム上での移植のし易さの保証をすること、信頼できる通信インターフェースの提供、PVM など既存のものと同様の使いやすさとより高い融通性を実現することを目的に現在も開発が進められている。[4][11]

2.3. OpenMP

2.3.1. OpenMP とは

OpenMP は、1997 年に米国の OpenMP Arehiteture Review Board が、Fortran をベースとして開発された共有メモリ並列プログラミングのための API 仕様である。その後 C/C++においても API 仕様が開発された。OpenMP は Fortran、C/C++をベースにコンパイラ指示文、ライボラリ、環境変数によって拡張したものである。よって、プログラムソースの中に逐次処理部分と並列処理部分が存在し、１つのプログラムソースで管理することが可能である。

OpenMP の実行モデル

OpenMP の実行モデルは PVM や MPI とは違い fork-join という並列実行モデルを持っている。OpenMP で記述されたプログラムソースは逐次処理部分と並列処理部分が混在したプログラムである。OpenMP のプログラムを実行すると Master スレッドと呼ばれる単一のスレッドのみで実行が開始される。マスタスレッドは並列指示文まで達成すると Slave スレッドと呼ばれる複数のスレッドを生成し、ワークシェアリング構文に対応する処理を全スレッドにて実行される。この全スレッドで並列処理を行っている部分を並列リージョン（パラレルリージョン）

という。[2]

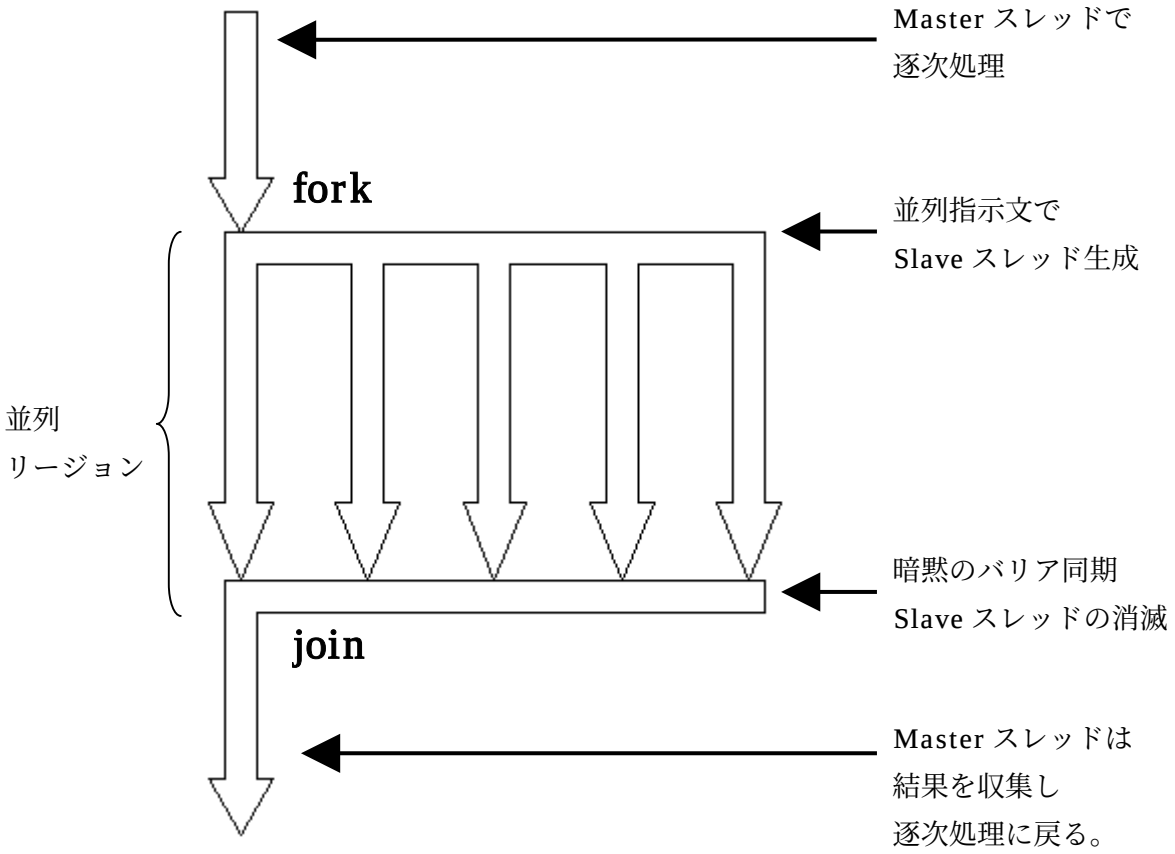


図 3　fork-join モデル

3. 有限要素法によるカルマン渦の流体解析

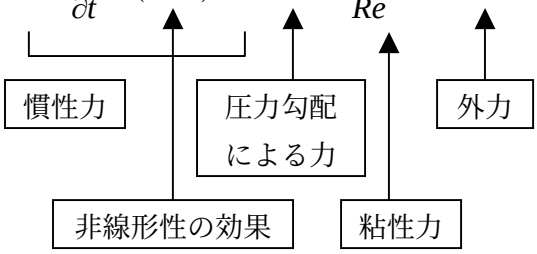
3.1. ナビエ・ストークス方程式

流体とは空気や水など微小な力により形が変形するものを言う。流体力学はこの流体を空間的、時間的スケールで観察し、流体がどのように運動しているか解析することである。

流体の性質として粘性と圧縮性がある。粘性とは流体に働く摩擦力によって流体の運動をできるだけ均一にしようとする性質をいう。例えば水をボールのようなそこが丸い容器に入れるとする。その水を棒などでかき混ぜるとそのうち容器も一緒に回転し始める。また、水をかき混ぜることを止め、しばらく時間を置けば容器の水の回転は収まるだろう。つまり、この現象は水が持つ粘性によって作用して起きる現象である。次に圧縮性とは流体の密度が変化する性質をいう。例えば、注射器に空気を入れ空気が逃げないように密閉し、ピストンを動かしてみると注射器の中の空気の体積が変化することが分かる。この現象は空気の持つ圧縮性が作用して起きている現象である。流体計算をする上で流体の粘性、圧縮性などの性質を考えて計算を行う必要がある。

流体力学において用いられる方程式がナビエ・ストークス方程式である。ナビエ・ストークス方程式は、ニュートン力学における運動の第 2 法則を表していて、流体に加わる力によって流体がどのように運動をするかを記述した式である。流体が慣性によってこれまでの運動をそのまま続けようとする力（慣性力）と、圧力の分布が空間的に均一ではなく変化があることによって生じる力、粘性力、及び外力（重力など）をすべて考慮した、力のつり合いを表した方程式である。

$$\frac{\partial u}{\partial t} + (u \cdot \nabla)u = -\nabla p + \frac{1}{Re}\nabla u + f \qquad (3.1)$$



u ： 流速
p ： 圧力
Re ： レイノズル数
f ： 外力

図 4　ナビエ・ストークス方程式

式（3.1）が非圧縮性流体に対するナビエ・ストークス方程式である。

レイノズル数とは、流体の運動における慣性の降下と、粘性の降下の比を表す量である。比であることから物理量を比較した量であり、慣性による力と粘性による力とを比べた、単位を持たない無次元量である。レイノズル数 Re は次式で計算できる。

$$\mathrm{Re} = \frac{\rho L U}{\mu} \qquad \cdots (3.2)$$

ここで、ρ は流体の密度、L は流れにおける代表的な長さ、U は流れの代表的な速度、μ は粘性係数と呼ばれる流体ごとに固有の値をとる物性値である。[10]

3.2. 有限要素法

式（3.1）を見て分かるように、ナビエ・ストークス方程式は非線形性を持っている。そのため、解析的に解くことは非常に限られた場合を除いては、一般には難しい。よってナビエ・ストークス方程式を数値的アプローチによって解くことが盛んに行われている。そのアプローチの中に有限要素法がある。[10]

有限要素法は

1．幾何学的柔軟性を持っており、任意形状領域の問題に適応できる。
2．自然境界条件の取り扱いが容易である。
3．手法の数学的正当化と誤差解析の研究が進んでいる。
4．計算機向きの手法であり、汎用プログラムの作成が可能である。

などの特徴をもつナビエ・ストークス方程式の有力な解析法である。

有限要素法で解くべき支配方程式を一般的な表現で次のようにする。

$$Lu = f \qquad (3.3)$$

ここで、L はさまざまな物理現象を記述するための微分演算を含んでいる。u は求めるべき未知変数、f は既知の量とする。次に式（3.3）を解くにあたって、その解 x を次のように近似して考える。

$$u \approx \sum_{i=1}^{M} N_i u_i = N_1 u_1 + N_2 u_2 + \cdots + N_M u_M \qquad (3.4)$$

ここで、問題が対象とする空間には離散点として置かれているとし、添え字の i は離散点を表す指標である。N_i は点 i について定義された関数、u_i は点 i について定義された未知量、M は点の総数である。N_i は基本関数と呼ばれ、その関数形は解析手法によって異なるが、近似の仕方によって決まる既知のものである。つまり、解 u は方程式の種類、すなわち、演算子 L にどのような微分操作が含まれているかによってさまざまな形になる。

支配方程式において式（3.4）の近似形を仮定し、各解法のアルゴリズムに従って処理すると、式（3.3）は次式のような $u_1, u_2, \cdots, u_M$ に関する連立 1 次方程式に変換され、この連立 1 次方程式を解くことで最終的に解が求められる。

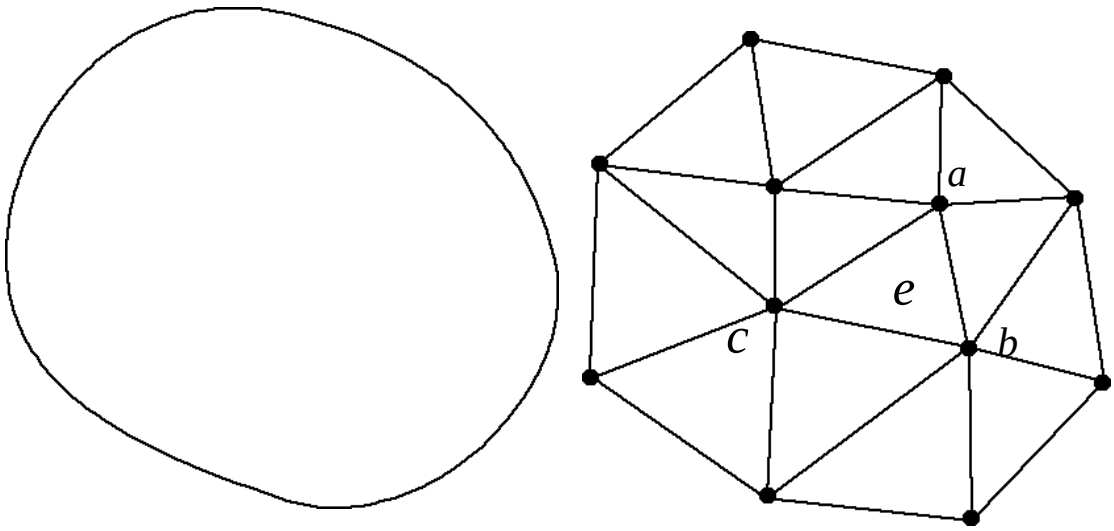
$$[K_{ij}] \cdot \{u_j\} = \{f_i\} \qquad i, j = 1,2,\ldots,M \qquad (3.5)$$

式（3.5）のことを有限要素法では剛性方程式と呼ぶ。

有限要素法では、解析形状を図 5 のように要素（element）と呼ばれる部分領域に分割し、要素の点に節点（node）を置く。要素や節点にはそれぞれ要素番号、節点番号がつけられ、節点には式（3.4）のu_iが定義される。したがって、要素を多く用いて分割を細かくするほど節点の数も多くなり解の精度が向上する。

いま、ある 1 つの要素eに注目し、その要素での剛性方程式を考える。要素eは 3 個の節点a、b、cから構成され、要素eに関する剛性方程式はこれらの 3 個の節点に定義されたu_a、u_b、u_cだけを用いて表すことができる。また、要素eに関する剛性方程式も 3×3 の行列方程式になる。解析形状は要素の集まりでできているから、全要素について剛性方程式を作り、それらを重ね合わせることによって本来の最終的に解くべき連立 1 次方程式（式（3.5））を組み立てる。

要素は 2 次元空間では三角形、四角形に分割するのが一般的で、3 次元空間においては四面体、六面体に分割するのが一般的である。



（a）解析形状　　（b）有限要素分割

図 5　解析形状の有限要素分割

3.3. 有限要素法による流体解析

本節では流れを 2 次元非定常流れ、取り扱う流体を非圧縮性粘性流体、つまり自然の空気中の流れの変化を問題として有限要素法による流体解析の説明をする。[5]

3.3.1. 基礎方程式

流れの基礎方程式は次式のようなナビエ・ストークス方程式と非圧縮性連続式となる。

$$\frac{\partial u_i}{\partial t}+u_i\frac{\partial u_i}{\partial x_j}=-\frac{\partial p}{\partial x_i}+\frac{1}{Re}\frac{\partial}{\partial x_j}\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right) \tag{3.6}$$

$$\frac{\partial u_i}{\partial x_i}=0 \tag{3.7}$$

ここで、i は有限要素を構成する節点の節点番号、u は流体の速度（ベクトル量）、p は圧力、x_i は空間に関する座標$\left(x_j, j=1,2\right)$、t は時間とする。また Re はレイノルズ数である。

3.3.2. 流速修正法

流速修正法とは、流れ場の未知数である流速と圧力を分離して解くといういわゆる分離型手法のひとつである。特長として、流速場を陽的に処理する準陽的解法であるため陰的解法にくらべてメモリ容量が節約できるという点がある。

まず、基礎方程式に対する時間方向の離散化を以下のように行う。

$$\frac{u_i^{n+1}-u_i^n}{\Delta t}+u_j^n\frac{\partial u_i^n}{\partial x_j}=-\frac{\partial p^{n+1}}{\partial x_i}+\frac{1}{Re}\frac{\partial}{\partial x_j}\left(\frac{\partial u_i^n}{\partial x_j}+\frac{\partial u_j^n}{\partial x_i}\right) \tag{3.8}$$

$$\frac{\partial u_i^{n+1}}{\partial x_i}=0 \tag{3.9}$$

次に流速の予測子として中間流速 $\tilde{u}_i$ を以下のように定義する。

$$\tilde{u}_i=u_i^n-\Delta t\left\{u_j^n\frac{\partial u_i^n}{\partial x_j}-\frac{1}{Re}\frac{\partial}{\partial x_j}\left(\frac{\partial u_i^n}{\partial x_j}+\frac{\partial u_j^n}{\partial x_i}\right)\right\} \tag{3.10}$$

境界条件は

$$\tilde{u}_i=\tilde{u}_i、\qquad \left\{\frac{1}{Re}\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right)\right\}\cdot n_j=\tilde{t}_i \tag{3.11}$$

であり、n_j は境界上の外向き法線ベクトルである。運動方程式 （3.8）の両方の発散をとり、非圧縮性の連続条件（3.9）を代入して整理すると、次の圧力に関するポアソン方程式

が導かれる。

$$\frac{\partial^2 p^{n+1}}{\partial x_i^2} = \frac{1}{\Delta t}\frac{\partial \tilde{u}}{\partial x_i} \tag{3.12}$$

このポアソン方程式に対する境界条件は次のように与えられる

$$p^{n+1} = \hat{p}、 \qquad \frac{\partial p^{n+}}{\partial n_i} = \hat{q}_i \tag{3.13}$$

また式（3,8）から式（3.10）を引くと速度場u_i^{n+1}に関する

$$u_i^{n+1} = \tilde{u}_i - \Delta t \frac{\partial p^{n+1}}{\partial x_i} \tag{3.14}$$

が得られる。これに式（3.12）で得られたp^{n+1}と式（3.10）から得られた$\tilde{u}_i$を代入して速度場が計算される。

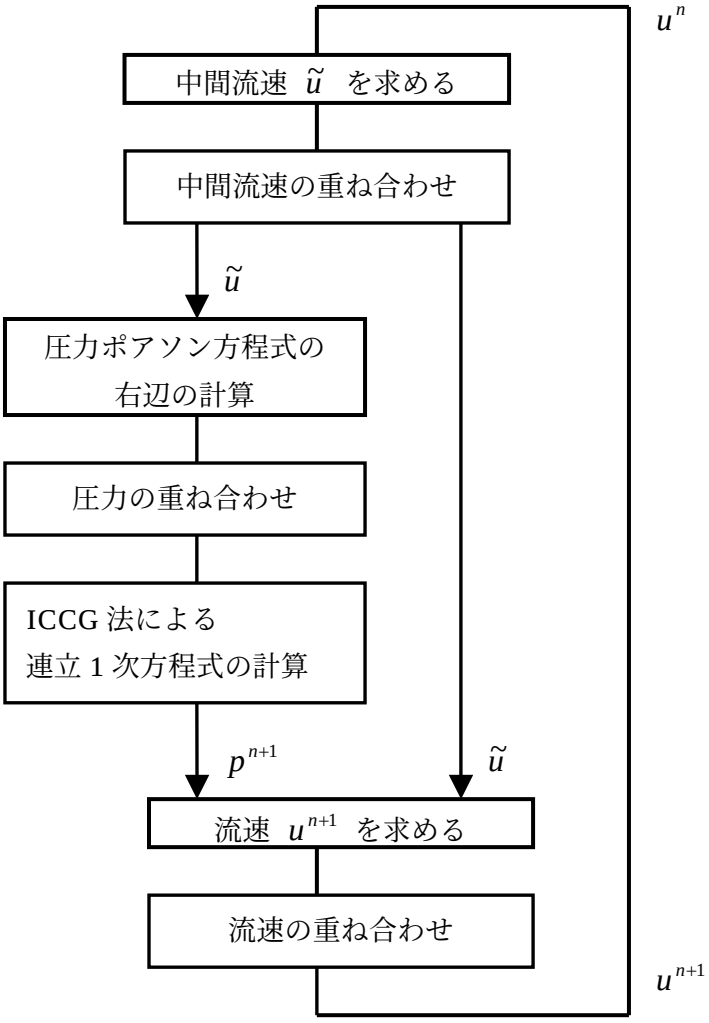


図 6　流速修正法における計算の流れ

流速修正法における計算の流れを図 6 に示す。
解くべき式は、(3.10)、(3.12)、(3.14) の 3 つである。そこで、これらの式の空間微分の項を三角形有限要素を用いて定式化する。(付録 A) 三角形有限要素の節点と座標を図 7 のようにする。

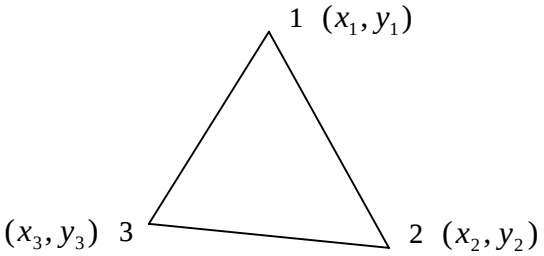


図 7　要素節点と未知変数

定式化したものが以下の剛性方程式である。ただし、

$$\Delta = \frac{1}{2}\left[x_i\left(y_i - y_k\right) + x_j\left(y_k - y_i\right) + x_k\left(y_i - y_j\right)\right] \tag{3.15}$$

$$\left.\begin{aligned} b_i &= \frac{1}{2\Delta}\left(y_j - y_k\right) \\ a_i &= \frac{1}{2\Delta}\left(x_k - x_j\right) \end{aligned}\right\} \quad (i,k,j = 1,2,3) \tag{3.16}$$

・中間流速の式（3.10）
＜i＝1（x 方向）の場合＞

$$\frac{\Delta}{3}\begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}\begin{Bmatrix} \tilde{u}_1 \\ \tilde{u}_2 \\ \tilde{u}_3 \end{Bmatrix} = \frac{\Delta}{3}\begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}\begin{Bmatrix} u_1^n \\ u_2^n \\ u_3^n \end{Bmatrix}$$

$$-\Delta t\left\{\frac{\Delta}{12}\left(b_1u_1 + b_2u_2 + b_3u_3\right)\cdot\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}\begin{Bmatrix} u_1^n \\ u_2^n \\ u_3^n \end{Bmatrix} + \frac{\Delta}{12}\left(c_1u_1 + c_2u_2 + c_3u_3\right)\cdot\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}\begin{Bmatrix} v_1^n \\ v_2^n \\ v_3^n \end{Bmatrix}\right.$$

$$\left. + \frac{1}{Re}\left(2\Delta\begin{bmatrix} b_1^2 & b_1b_2 & b_1b_3 \\ b_2b_1 & b_2^2 & b_2b_3 \\ b_3b_1 & b_3b_2 & b_3^2 \end{bmatrix}\begin{Bmatrix} u_1^n \\ u_2^n \\ u_3^n \end{Bmatrix} + \Delta\begin{bmatrix} b_1c_1 & b_1c_2 & b_1c_3 \\ b_2c_1 & b_2c_2 & b_2c_3 \\ b_3c_1 & b_3c_2 & b_3c_3 \end{bmatrix}\begin{Bmatrix} v_1^n \\ v_2^n \\ v_3^n \end{Bmatrix}\right)\right\} \tag{3.17}$$

＜i＝2（y 方向）の場合＞

$$\frac{\Delta}{3}\begin{bmatrix}1 & & \\ & 1 & \\ & & 1\end{bmatrix}\begin{Bmatrix}\tilde{v}_1\\ \tilde{v}_2\\ \tilde{v}_3\end{Bmatrix}=\frac{\Delta}{3}\begin{bmatrix}1 & & \\ & 1 & \\ & & 1\end{bmatrix}\begin{Bmatrix}v_1^n\\ v_2^n\\ v_3^n\end{Bmatrix}$$

$$-\Delta t\left\{\frac{\Delta}{12}\left(b_1u_1+b_2u_2+b_3u_3\right)\cdot\begin{bmatrix}2 & 1 & 1\\ 1 & 2 & 1\\ 1 & 1 & 2\end{bmatrix}\begin{Bmatrix}u_1^n\\ u_2^n\\ u_3^n\end{Bmatrix}+\frac{\Delta}{12}\left(c_1u_1+c_2u_2+c_3u_3\right)\cdot\begin{bmatrix}2 & 1 & 1\\ 1 & 2 & 1\\ 1 & 1 & 2\end{bmatrix}\begin{Bmatrix}v_1^n\\ v_2^n\\ v_3^n\end{Bmatrix}\right.$$

$$\left.+\frac{1}{Re}\left(\Delta\begin{bmatrix}c_1b_1 & c_1b_2 & c_1b_3\\ c_2b_1 & c_2b_2 & c_2b_3\\ c_3b_1 & c_3b_2 & c_3b_3\end{bmatrix}\begin{Bmatrix}u_1^n\\ u_2^n\\ u_3^n\end{Bmatrix}+2\Delta\begin{bmatrix}c_1^2 & c_1c_2 & c_1c_3\\ c_2c_1 & c_2^2 & c_2c_3\\ c_3c_1 & c_3c_2 & c_3^2\end{bmatrix}\begin{Bmatrix}v_1^n\\ v_2^n\\ v_3^n\end{Bmatrix}\right)\right\} \quad (3.18)$$

・圧力ポアソン方程式（3.12）

$$\Delta\begin{bmatrix}b_1^2+c_1^2 & b_1b_2+c_1c_2 & b_1b_3+c_1c_3\\ b_2b_1+c_2c_1 & b_2^2+c_2^2 & b_2b_3+c_2c_3\\ b_3b_1+c_3c_1 & b_3b_2+c_3c_2 & b_3^2+c_3^2\end{bmatrix}\begin{Bmatrix}p_1^{n+1}\\ p_2^{n+1}\\ p_3^{n+1}\end{Bmatrix}$$

$$=-\frac{1}{\Delta t}\frac{\Delta}{3}\left(\begin{bmatrix}b_1 & b_2 & b_3\\ b_1 & b_2 & b_3\\ b_1 & b_2 & b_3\end{bmatrix}\begin{Bmatrix}\tilde{u}_1\\ \tilde{u}_2\\ \tilde{u}_3\end{Bmatrix}+\begin{bmatrix}c_1 & c_2 & c_3\\ c_1 & c_2 & c_3\\ c_1 & c_2 & c_3\end{bmatrix}\begin{Bmatrix}\tilde{v}_1\\ \tilde{v}_2\\ \tilde{v}_3\end{Bmatrix}\right) \quad (3.19)$$

・流速を求める式（3.14）

＜i＝1（x 方向）の場合＞

$$\frac{\Delta}{3}\begin{bmatrix}1 & & \\ & 1 & \\ & & 1\end{bmatrix}\begin{Bmatrix}u_1^{n+1}\\ u_2^{n+1}\\ u_3^{n+1}\end{Bmatrix}=\frac{\Delta}{3}\begin{bmatrix}1 & & \\ & 1 & \\ & & 1\end{bmatrix}\begin{Bmatrix}\tilde{u}_1\\ \tilde{u}_2\\ \tilde{u}_3\end{Bmatrix}-\Delta t\frac{\Delta}{3}\begin{bmatrix}b_1 & b_2 & b_3\\ b_1 & b_2 & b_3\\ b_1 & b_2 & b_3\end{bmatrix}\begin{Bmatrix}p_1^{n+1}\\ p_2^{n+1}\\ p_3^{n+1}\end{Bmatrix} \quad (3.20)$$

＜i＝2（y 方向）の場合＞

$$\frac{\Delta}{3}\begin{bmatrix}1 & & \\ & 1 & \\ & & 1\end{bmatrix}\begin{Bmatrix}v_1^{n+1}\\ v_2^{n+1}\\ v_3^{n+1}\end{Bmatrix}=\frac{\Delta}{3}\begin{bmatrix}1 & & \\ & 1 & \\ & & 1\end{bmatrix}\begin{Bmatrix}\tilde{v}_1\\ \tilde{v}_2\\ \tilde{v}_3\end{Bmatrix}-\Delta t\frac{\Delta}{3}\begin{bmatrix}c_1 & c_2 & c_3\\ c_1 & c_2 & c_3\\ c_1 & c_2 & c_3\end{bmatrix}\begin{Bmatrix}p_1^{n+1}\\ p_2^{n+1}\\ p_3^{n+1}\end{Bmatrix} \quad (3.21)$$

以上の式が剛性方程式と呼ばれる方程式になる。

3.3.3. 剛性方程式の全体系への重ね合わせ

前節で求めた要素ごとの剛性方程式を全体領域で重ねあわせる手順を解説する。

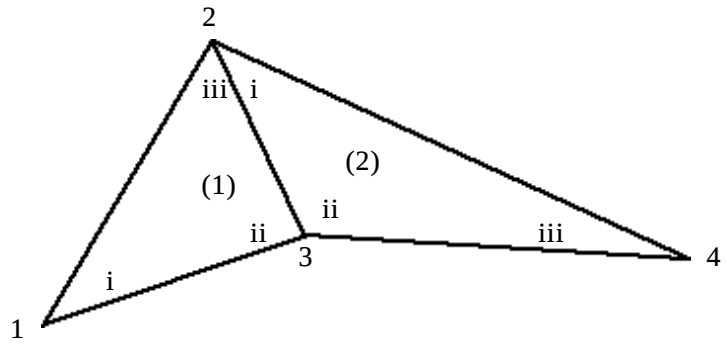


図 8　有限要素分割の例

表 1　要素と節点の結合状態

要素番号	節点 i	節点 ii	節点 iii
1	1	3	2
2	2	3	4

全体系への剛性方程式の重ね合わせは､図 8 のように示すような有限要素分割例における､表 1 に示した要素と節点の結合状態を参照して行われる。いま、図 8 のような 4 節点 2 要素の有限要素分割において、図のようにそれぞれの要素の頂点の節点番号を i ～ iii のように与えておく。ここで簡単のために、要素内の剛性方程式を次のようなものとする。

$$\mathbf{K}^{(\mathbf{e})}\mathbf{u}^{(\mathbf{e})} = \mathbf{F}^{(\mathbf{e})}$$

$$\Updownarrow$$

$$\begin{bmatrix} K_{11}^{(e)} & K_{12}^{(e)} & K_{13}^{(e)} \\ K_{21}^{(e)} & K_{22}^{(e)} & K_{23}^{(e)} \\ K_{31}^{(e)} & K_{32}^{(e)} & K_{33}^{(e)} \end{bmatrix} \begin{Bmatrix} u_{\mathrm{i}}^{(\mathrm{e})} \\ u_{\mathrm{ii}}^{(\mathrm{e})} \\ u_{\mathrm{iii}}^{(\mathrm{e})} \end{Bmatrix} = \begin{Bmatrix} F_{\mathrm{i}}^{(\mathrm{e})} \\ F_{\mathrm{ii}}^{(\mathrm{e})} \\ F_{\mathrm{iii}}^{(\mathrm{e})} \end{Bmatrix} \qquad (3.22)$$

ここに、$\mathbf{u}^{(\mathbf{e})}$を要素領域内の節点未知ベクトル、$\mathbf{F}^{(\mathbf{e})}$を要素領域内の節点での荷重（既知）ベクトル、$\mathbf{K}^{(\mathbf{e})}$を未知ベクトル$\mathbf{u}^{(\mathbf{e})}$の係数行列とし、添字の“ e ”は要素番号を表すものとする。ここで、この式を要素ごとに行列表示すれば、次のようになる。

$$\begin{bmatrix} K_{11}^{(1)} & K_{12}^{(1)} & K_{13}^{(1)} & \vdots & & & \\ K_{21}^{(1)} & K_{22}^{(1)} & K_{23}^{(1)} & \vdots & & & \\ K_{31}^{(1)} & K_{32}^{(1)} & K_{33}^{(1)} & \vdots & & & \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ & & & \vdots & K_{11}^{(2)} & K_{12}^{(2)} & K_{13}^{(2)} \\ & & & \vdots & K_{21}^{(2)} & K_{22}^{(2)} & K_{23}^{(2)} \\ & & & \vdots & K_{31}^{(2)} & K_{32}^{(2)} & K_{33}^{(2)} \end{bmatrix} \begin{Bmatrix} u_{\mathrm{i}}^{(1)} \\ u_{\mathrm{ii}}^{(1)} \\ u_{\mathrm{iii}}^{(1)} \\ \cdots \\ u_{\mathrm{i}}^{(2)} \\ u_{\mathrm{ii}}^{(2)} \\ u_{\mathrm{iii}}^{(2)} \end{Bmatrix} = \begin{Bmatrix} F_{\mathrm{i}}^{(1)} \\ F_{\mathrm{ii}}^{(1)} \\ F_{\mathrm{iii}}^{(1)} \\ \\ F_{\mathrm{i}}^{(2)} \\ F_{\mathrm{ii}}^{(2)} \\ F_{\mathrm{iii}}^{(2)} \end{Bmatrix} \qquad (3.23)$$

図 8 の要素分割の場合、要素内の節点の未知数と全体系の節点における未知数には、表 1 に対応し、次のような関係があることが分かる。

$$\left.\begin{array}{lll} u_{\mathrm{i}}^{(1)} = u_1 & , & u_{\mathrm{i}}^{(2)} = u_1 \\ u_{\mathrm{ii}}^{(1)} = u_2 & , & u_{\mathrm{ii}}^{(2)} = u_2 \\ u_{\mathrm{iii}}^{(1)} = u_3 & , & u_{\mathrm{iii}}^{(2)} = u_3 \end{array}\right\} \qquad (3.24)$$

また、要素節点における荷重ベクトル**F**を要素が重なり合った全体系の節点において単純に合計したものが全体系の節点の荷重値になることから、次のような関係が得られる。

$$\left.\begin{array}{lll} F_{\mathrm{i}}^{(1)} & = & F_1 \\ F_{\mathrm{iii}}^{(1)} + F_{\mathrm{i}}^{(2)} & = & F_2 \\ F_{\mathrm{ii}}^{(1)} + F_{\mathrm{ii}}^{(2)} & = & F_3 \\ F_{\mathrm{iii}}^{(2)} & = & F_4 \end{array}\right\} \qquad (3.25)$$

ここで、式（3.24）と式（3.25）関係を行列により表すと次のようになる。

$$\begin{Bmatrix} u_{\mathrm{i}}^{(1)} \\ u_{\mathrm{ii}}^{(1)} \\ u_{\mathrm{iii}}^{(1)} \\ \cdots \\ u_{\mathrm{i}}^{(2)} \\ u_{\mathrm{ii}}^{(2)} \\ u_{\mathrm{iii}}^{(2)} \end{Bmatrix} = \begin{bmatrix} 1 & & & \\ & & 1 & \\ & 1 & & \\ \cdots & \cdots & \cdots & \cdots \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{Bmatrix} \qquad (3.26)$$

$$\begin{bmatrix} 1 & & & \vdots & & & \\ & & 1 & \vdots & 1 & & \\ & 1 & & \vdots & & 1 & \\ & & & \vdots & & & 1 \end{bmatrix} \begin{Bmatrix} F_{\mathrm{i}}^{(1)} \\ F_{\mathrm{ii}}^{(1)} \\ F_{\mathrm{iii}}^{(1)} \\ \cdots \\ F_{\mathrm{i}}^{(2)} \\ F_{\mathrm{ii}}^{(2)} \\ F_{\mathrm{iii}}^{(2)} \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{Bmatrix} \tag{3.27}$$

そこで、この式（3.26）と式（3.27）の関係を式（3.23）へ代入し整理すると、最終的に次のような全体系の剛性方程式が導かれる。

$$\begin{bmatrix} K_{11}^{(1)} & K_{13}^{(1)} & K_{12}^{(1)} & \\ K_{31}^{(1)} & K_{33}^{(1)} + K_{11}^{(2)} & K_{32}^{(1)} + K_{12}^{(2)} & K_{13}^{(2)} \\ K_{21}^{(1)} & K_{23}^{(1)} + K_{21}^{(2)} & K_{22}^{(1)} + K_{22}^{(2)} & K_{23}^{(2)} \\ & K_{31}^{(2)} & K_{32}^{(2)} & K_{33}^{(2)} \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{Bmatrix} \tag{3.28}$$

以上のように、ひとつひとつの要素における剛性方程式から、全体系への剛性方程式へと作り替える作業を“全体系への重ね合わせ”と呼んでいる。

3.3.4. 連立 1 次方程式の解法

全体系への重ね合わせが終了し、未知ベクトルに対する剛性方程式（連立 1 次方程式）として定式化された。これらのうち運動方程式については、未知数ベクトルの係数行列が対角集中化されていることから逆行列の計算を必要とせず、代入計算だけで計算を進めていくことができるが、圧力ポアソン方程式は係数行列は対角集中化されておらず、なんらかの連立 1 次方程式の解法を用いて計算を進めなければならない。この、連立 1 次方程式の解法には実に多くのものがあるが、大別をすれば直説法と反復法の 2 つに分類される。本節では、連立 1 次方程式の反復解法としてよく知られている不完全コレスキー分解を用いて前処理付き共役勾配法（ICCG 法）に関する解説を行う。

3.3.5. 共役勾配法

現在、反復法の中でよく用いられるものは共役勾配法（以下 CG 法）である。そこで本節では、共役勾配法をもとにした連立 1 次方程式の反復解法について簡単に説明する。

いま、与えられた連立方程式を

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{3.29}$$

とする。ただし、**A** は係数行列、**x** は未知量ベクトル、**b** は荷重ベクトルである。CG 法の基本原理は最適化法に由来しており、この方法では式（3.56）に関する評価関数を次のように設定し、その最小化を考える。ただし、 **A** を n 次対称正定値行列とする。

$$F(\mathbf{x}) = (\mathbf{x}, \mathbf{Ax}) - 2(\mathbf{x}, \mathbf{b}) \tag{3.30}$$

ここで、$(\mathbf{x},\mathbf{b})$は$\mathbf{x},\mathbf{b}$の内積を表すものとする。また$\mathbf{A}$として正定値を仮定していることから、このn変数の2次関数は最小値を持ち、$\mathbf{A}$の対称性から

$$\frac{\partial F}{\partial x_k} = \sum_{j=1}^{n} \left\{ (a_{kj} + a_{jk}) x_j - 2b_k \right\} = 2\sum_{j=1}^{n} (a_{kj} - b_k) = 0 \qquad (k = 1,2,\ldots,n) \tag{3.31}$$

を満たすことになる。これは与えられた方程式（3.29）そのものであり、したがって$F(\mathbf{x})$を最小にする$\mathbf{x}$は与えられた連立方程式の解となる。そこで、連立1次方程式（3.56）を解くために、適当な初期値$\mathbf{x}_0$から出発して順次修正を加え近似ベクトル$\mathbf{x}_{k+1}$を求めるアルゴリズムを以下のように構築する。

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k \tag{3.32}$$

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

$$\begin{aligned} F(\mathbf{x}+\mathbf{h}) &= (\mathbf{x}+\mathbf{h}, \mathbf{A}(\mathbf{x}+\mathbf{h})) - 2(\mathbf{x}+\mathbf{h}, \mathbf{b}) \\ &= F(\mathbf{x}) + (\mathbf{h}, \mathbf{Ah}) - 2(\mathbf{h}, \mathbf{b} - \mathbf{Ax}) \end{aligned} \tag{3.33}$$

であることから、式（3.32）を式（3.30）へ代入し、$F(\mathbf{x})$を最小にするα_kを決定する。$\mathbf{p}_k \neq \mathbf{0}$のとき、

$$\begin{aligned} F(\mathbf{x}_{k+1}) &= (\mathbf{x}_k + \alpha_k \mathbf{p}_k, \mathbf{A}(\mathbf{x}_k + \alpha_k \mathbf{p}_k)) - 2(\mathbf{x}_k + \alpha_k \mathbf{p}_k, \mathbf{b}) \\ &= F(\mathbf{x}_k) + \alpha_k^2 (\mathbf{p}_k, \mathbf{Ap}_k) - 2\alpha_k (\mathbf{p}_k, \mathbf{r}_k) \end{aligned} \tag{3.34}$$

ただし、$\mathbf{r}_k$は近似ベクトル$\mathbf{x}_k$の残差

$$\mathbf{r}_k = \mathbf{b} - \mathbf{Ax}_k \tag{3.35}$$

である。ここで最小化の手順に従い、$\partial F / \partial \alpha_k = 0$として$\alpha_k$を求めると次のようになる。

$$\alpha_k = \frac{(\mathbf{p}_k, \mathbf{r}_k)}{(\mathbf{p}_k, \mathbf{Ap}_k)} \tag{3.36}$$

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

$$\mathbf{p}_k = \mathbf{r}_k + \beta_{k-1} \mathbf{p}_{k-1} \tag{3.37}$$

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

$$\beta_k = -\frac{(\mathbf{r}_{k+1}, \mathbf{Ap}_k)}{(\mathbf{p}_k, \mathbf{Ap}_k)} \tag{3.38}$$

以上の手順をまとめると、次のようになる。

i. $k = 0$とし、適当な初期値$\mathbf{x}_0$を与え以下を計算する。

$$\mathbf{r}_0 = \mathbf{b} - \mathbf{Ax}_0 、 \qquad \mathbf{p}_0 = \mathbf{r}_0$$

ii. 以下の計算を行う。

$$\alpha_k = \frac{(\mathbf{p}_k, \mathbf{r}_k)}{(\mathbf{p}_k, \mathbf{Ap}_k)} 、 \quad \mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k 、 \quad \mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{Ap}_k$$

iii. $\|\mathbf{r}_{k+1}\|_2$ / $\|\mathbf{b}\|_2 \le \varepsilon$ ならば終了する。そうでなければ以下を計算する。

$$\beta_k = -\frac{(\mathbf{r}_{k+1}, \mathbf{A}\mathbf{p}_k)}{(\mathbf{p}_k, \mathbf{A}\mathbf{p}_k)}、 \quad \mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$$

iv. $k = k+1$として ii へ戻る。

ただし、$\|\ \|_2$ はベクトルのニークリッドノルムであり、

$$\|\mathbf{u}\|_2 = \left(\sum_{j=1}^{n} u_j^2 \right)^{\frac{1}{2}} \tag{3.39}$$

である。

以上の方法によって連立 1 次方程式の解を求める方法を、共役勾配法（CG 法）と言う。CG 法は、限界的には n 回の探索で解に到達するが、丸め誤差の影響で n 回の探索では必ずしも解に到達しない場合もある。このことを避けるために、一般に次節で述べるような前処理が施され、効率の良い計算が行われる。

3.3.5. LU 分解と $\mathbf{LDL}^T$ 分解

連立方程式の解法や固有値の問題で行列の分解に用いられるものとして **LU** 分解や $\mathbf{LDL}^T$ 分解と呼ばれる分解がある。この分解は、CG 法の前処理を行う際に必要となる重要なものであるため、この概略について説明する。

いま、式（3.56）の行列 **A** を下三角行列 **L** と上三角行列 **U** の積として分解する。

$$\mathbf{A} = \mathbf{LU}、 \quad \mathbf{L} = \left(l_{ij} \right)、 \quad \mathbf{U} = \left(u_{ij} \right) \tag{3.40}$$

ここで、下三角行列および上三角行列の要素 l_{ij}、u_{ij} は次式の手順で求められる。

$$\begin{aligned} l_{ij} &= a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj} \qquad (i \ge j) \\ u_{ij} &= \frac{\left(a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj} \right)}{l_{ii}} \qquad (i < j) \end{aligned} \tag{3.41}$$

A を非対称行列としたとき **LU** 分解はクラウト分解と呼ばれる。一方、**A** を対称行列としたとき **LU** 分解はコレスキー分解と呼ばれ、一般に対称行列に対しては $\mathbf{LDL}^T$ 分解（改訂コレスキー分解）と呼ばれる分解を行う。

$\mathbf{LDL}^T$ 分解のときには、**LU** 分解のときに使用した下三角行列 **L** から作った対角行列 **D** を上三角行列 **U** すなわち $\mathbf{L}^T$ に掛ければよい。このときの各行列の成分は

$$d_{ii} = a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2 d_{kk} \qquad (i = 1,2,\ldots,n)$$

$$l_{ij} = \frac{\left(a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk} d_{ii} \right)}{d_{jj}} \tag{3.42}$$

となる。結局、$\mathbf{LDL}^T$ 分解した後の連立方程式の解法は

$$\mathbf{Lz} = \mathbf{b}$$
$$\mathbf{DL^T x} = \mathbf{z} \tag{3.43}$$

の 2 式を解くことに帰着する。

3.3.6. 不完全コレスキー分解

CG 法は、行列 $\mathbf{A}$ の固有値の分布に縮退があるほど少ない探索回数で真の解に到達できる。極端な場合は $\mathbf{A}$ が単位行列のときであり、このとき固有値は 1 に縮退しており探索回数は 1 である。このように、係数行列が単位行列に近いような連立 1 次方程式の場合には CG 方の収束が早められることが知られており、係数行列に対して何らかの前処理を施し係数行列を単位行列に近づけておいてから CG 法を適用するという方法が取られている。不完全コレスキー分解と言う分解も、この CG 法の収束を早めるための前処理としてよく使用されている分解法のひとつである。不完全コレスキー分解では、連立方程式（3.29）の係数行列 $\mathbf{A}$ を以下のようにして“不完全”に $\mathbf{LDL}^T$ 分解する。

$$\mathbf{A} \cong \mathbf{LDL}^T \tag{3.44}$$

本来の改訂コレスキー分解ではすべての成分 l_{ij} は正確に計算されず、ある場所の成分を強制的に 0 にする。例えば、行列 $\mathbf{A}$ の成分が 0 であった部分に対応する l_{ij} の成分を強制的に 0 にし計算をしない。このようにして、改訂コレスキー分解を不完全に実行したのち、$\mathbf{D}$ の成分の平方根を対角成分に持つ行列を $\mathbf{D}^{1/2}$ とし、

$$\mathbf{LDL}^T = \mathbf{LD}^{\frac{1}{2}} \left(\mathbf{LD}^{\frac{1}{2}} \right)^T \tag{3.45}$$

と書き直す。$\mathbf{A}$ が性定値であれば $\mathbf{LD}^{1/2}$ は特異ではないから、

$$\left[\left(\mathbf{LD}^{\frac{1}{2}} \right)^{-1} \mathbf{A} \left(\left(\mathbf{LD}^{\frac{1}{2}} \right)^T \right)^{-1} \right] \left(\mathbf{LD}^{\frac{1}{2}} \right)^T \mathbf{x} = \left(\mathbf{LD}^{\frac{1}{2}} \right)^{-1} \mathbf{b} \tag{3.46}$$

なる連立方程式の解は与えられた方程式の解と等しくなる。もし、$\mathbf{LDL}^T$ が $\mathbf{A}$ の正しい改訂コレスキー分解であるならば、

$$\mathbf{B}=\left(\mathbf{L}\mathbf{D}^{\frac{1}{2}}\right)^{-1}\mathbf{A}\left(\left(\mathbf{L}\mathbf{D}^{\frac{1}{2}}\right)^{T}\right)^{-1} \qquad (3.47)$$

は単位行列となるが、不完全分解であるため単位行列とはならない。しかしながら、ほぼ改訂コレスキー分解に近い操作を行っているため、**B** は単位行列に近い行列となっている。したがって、この連立 1 次方程式に CG 法を適用すれば、わずかな探索回数で解を得ることが期待できることになる。

この不完全コレスキー分解を適用して連立 1 次方程式を変換し、CG 法を適用したものが ICCG 法である。

3.3.7. ICCG 法

不完全コレスキー分解を適応した連立方程式に CG 法を適用した ICCG 法の解法のアルゴリズムは以下のようになる。

i. $k=0$ とし、適当な初期値 $\mathbf{x}_0$ を与え以下を計算する。

$$\mathbf{r}_0=\mathbf{b}-\mathbf{A}\mathbf{x}_0、\quad \mathbf{q}=\left(\mathbf{L}\mathbf{D}\mathbf{L}^T\right)^{-1}\mathbf{r}_0、\quad \mathbf{p}_0=\mathbf{q}$$

ii. 以下の計算を行う。

$$z_1=\left(\mathbf{q},\mathbf{r}_k\right)、\quad \mathbf{q}=\mathbf{A}\mathbf{p}_k$$

$$\alpha_k=\frac{\left(\mathbf{r}_k,\left(\mathbf{L}\mathbf{D}\mathbf{L}^T\right)^{-1}\mathbf{r}_k\right)}{\left(\mathbf{p}_k,\mathbf{A}\mathbf{p}_k\right)}=\frac{z_1}{\left(\mathbf{p}_k,\mathbf{q}\right)}$$

$$\mathbf{x}_{k+1}=\mathbf{x}_k+\alpha_k\mathbf{p}_k、\quad \mathbf{r}_{k+1}=\mathbf{r}_k-\alpha_k\mathbf{A}\mathbf{p}_k$$

iii. $\left\|\mathbf{r}_{k+1}\right\|_2/\left\|\mathbf{b}\right\|_2\le\varepsilon$ ならば終了する。そうでなければ以下を計算する。

$$\mathbf{q}=\left(\mathbf{L}\mathbf{D}\mathbf{L}^T\right)^{-1}\mathbf{r}_{k+1}、\quad z_2=\left(\mathbf{r}_{k+1},\mathbf{q}\right)$$

$$\beta_k=-\frac{\left(\mathbf{r}_{k+1},\left(\mathbf{L}\mathbf{D}\mathbf{L}^T\right)^{-1}\mathbf{r}_{k+1}\right)}{\left(\mathbf{p}_k,\left(\mathbf{L}\mathbf{D}\mathbf{L}^T\right)^{-1}\mathbf{r}_k\right)}=\frac{z_2}{z_1}$$

$$\mathbf{p}_{k+1}=\left(\mathbf{L}\mathbf{D}\mathbf{L}^T\right)^{-1}\mathbf{r}_{k+1}+\beta_k\mathbf{p}_k=\mathbf{q}+\beta_k\mathbf{p}_k$$

iv. $k=k+1$ として ii へ戻る。

ここに $\mathbf{L}\mathbf{D}\mathbf{L}^T$ は **A** の不完全コレスキー分解であるが、$\mathbf{q}=\left(\mathbf{L}\mathbf{D}\mathbf{L}^T\right)^{-1}\mathbf{r}_k$ の計算は、以下の方程式を解くことによって求められる。

$$\mathbf{L}\mathbf{y} = \mathbf{r}_k \qquad (3.48)$$

$$\mathbf{D}\mathbf{L}^{\mathrm{T}}\mathbf{q}_k = \mathbf{y} \qquad (3.49)$$

有限要素法流れ解析問題においては、ICCG 法を用いた計算は CG 法を用いた場合に比して 3～10 倍程度の計算の高速化が図られる

3.4. カルマン渦の問題定義

3.4.1 カルマン渦

カルマン渦とは、流れの中に構造物がある場合に、その構造物後方に生じる規則的な渦列のことを言う。現象例として風に吹かれて旗がはためく、風で電線が揺れうなるなどがある。もっとも簡単なカルマン渦の再現方法として棒と水槽を使った方法である。丸い棒を用意し、この棒の一端を持ち、水の張った水槽にもう一方の端をつけて手を水平に動かしてみる。棒を動かす速さをいろいろ変えると、手に揺れが伝わるポイントがあることに気づくはずだ。このとき、おがくずなど目印になるようなものを水面に浮かべておけば、棒が動いたうしろの水面に渦が並んでいるのが観察できる。以上のような現象を図で表したのが図 8 である。渦の発生直後は、図 8 のような渦列ではなく上下対称な双子渦という 2 つの渦が発生し、時間の経過とともに 2 つの渦は後方に移動する。この状態では旗がはためく現象や電線が風で起きる音が鳴る現象は起きない。双子渦のような対称な渦列は不安定な渦列であることが知られている。渦は安定した渦列の状態になるため、双子渦の後方から非対称な流れが発生し、その非対称性が徐々に前方へと伝わっていき安定したカルマン渦へと発達していく。このときに、旗がはためく現象や電線が風で起きる音が鳴る現象が起きるのである。渦列は最終的にある条件に達したとき、渦は安定し周期的なカルマン渦を発生するようになる。その条件は、上下の渦列の間隔を h、同じ渦列で隣り合う渦の間隔を l としたとき、h と l の比（h/l）が 0.281 になるときである。その後は流れが止まらない限り流れは条件のもとにカルマン渦を生じし続ける。

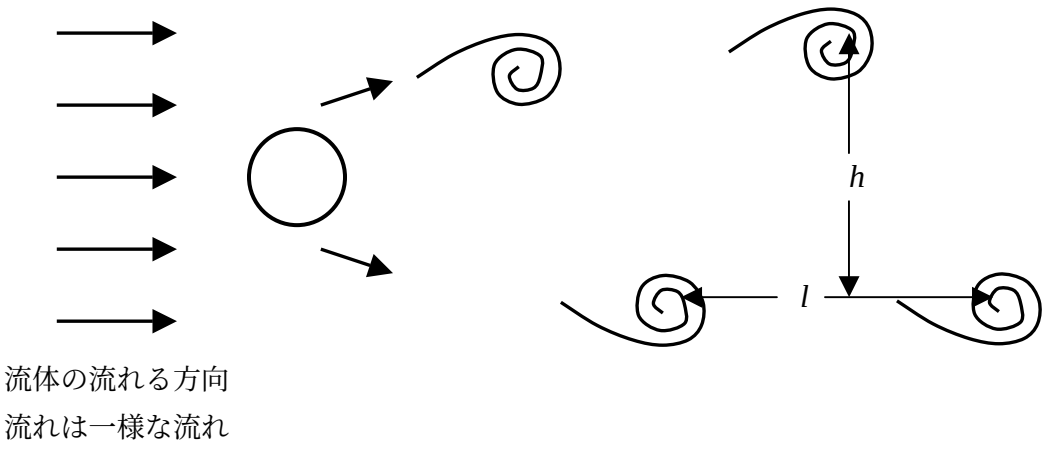


図 9　カルマン渦の渦列の様子

3.4.2. 問題定義

本研究では、2 次元空間上でカルマン渦の数値流体解析を行う。カルマン渦を発生させる構造物に円柱状の構造物をおき、2 次元空間での円柱を真円として扱う。円の中心を xy 座標で原点(0,0)に、円の半径を 40 とする円を配置する。ある点における流速を x 座標成分と y 座標成分に分解し値を与える。

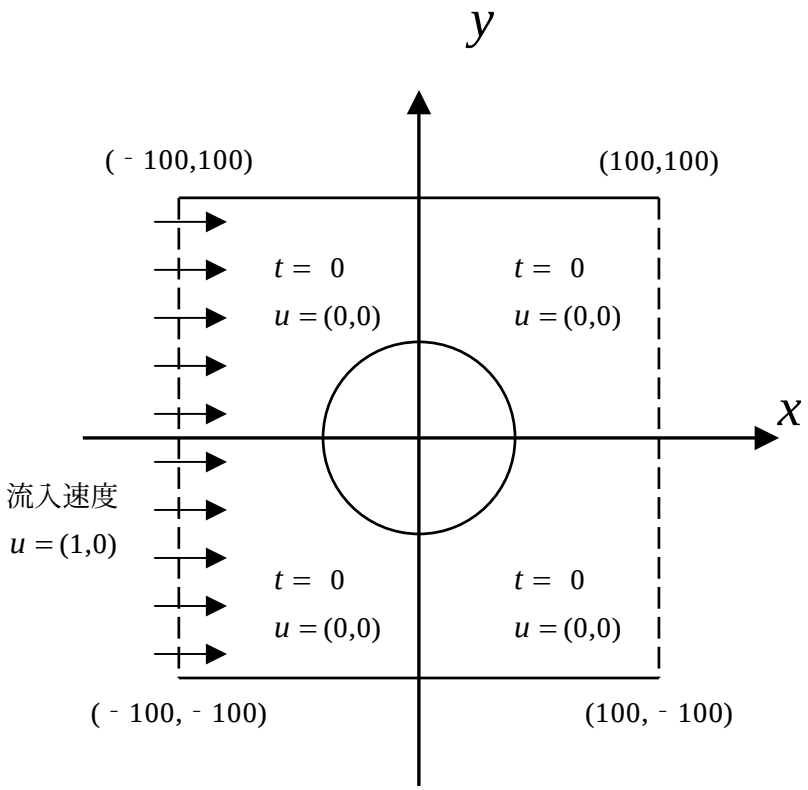
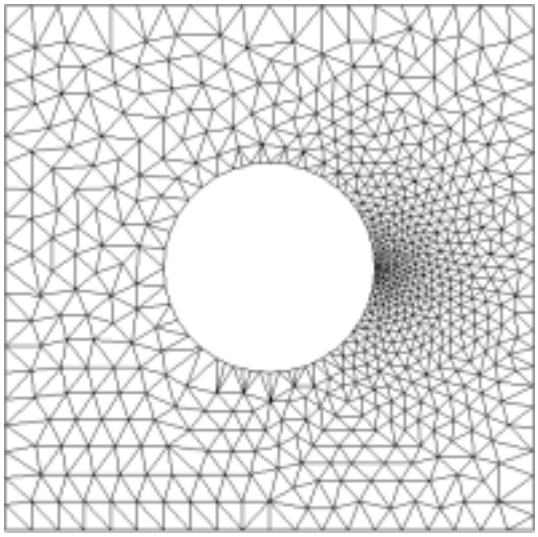


図 10　流路の形状と 座標

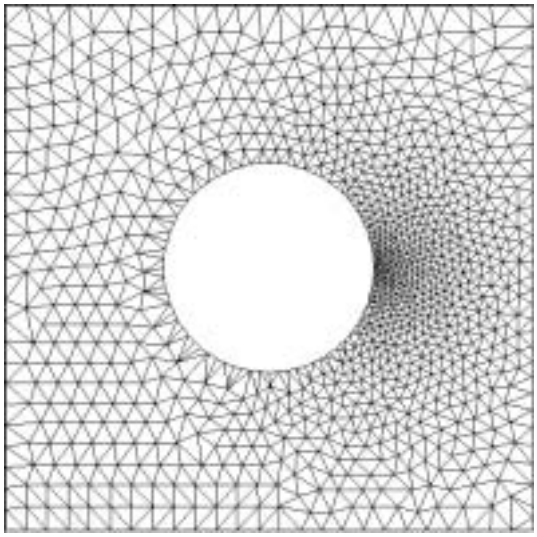
図 9 のように解析領域を定義した。点線の部分を流体が出入りする開放された部分とし、実線の部分を壁と円柱部分とした。壁と円柱の表面と、その表面に接する流体の間には摩擦が発生すると考える。流体は x 軸と平行に左から右方向へ流れ、流入時の速度値に $(1,0)$ を与える。時間 $t=0$ のとき、解析領域内の流速値は $(0,0)$ で流れが無いとする。

流体を解析するの時間を 5 秒とし、時間を 0.01 の微小時間をとり、500 回のループ（時間積分）させる。

流体解析に有限要素法を用いるため、解析領域を図 10 のようなメッシュに区切る。$x>0$ のときの領域を細かくメッシュを作ったのは、構造物の後方にカルマン渦ができるため、カルマン渦の発生を詳しく解析するためである。



（1）節点 771 個 要素 1381 個



（2）節点 1116 個 要素 2031 個

図 11　メッシュ図

3.5. 実行結果

PC 環境は、２つの環境で行った。① CPU：Intel(R) Pentium(R)4　2.54GHz、メモリ：DirectRDRAM 1.5GB、OS：Redhat7.3、② CPU：Intel(R) Pentium(R)Ⅲ　450MHz、メモリ：SDRAM 512GB、OS：Redhat7.3 である。レイノルズ数を 100 として実行を行った結果、表 2 のような実行結果が得られた。

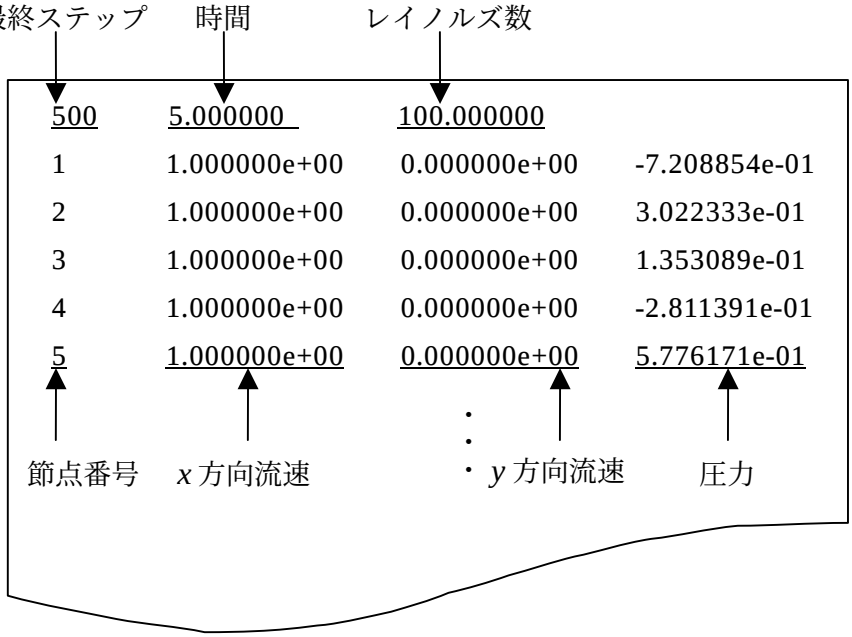


図 12　出力結果（節点 771 個 要素 1381 個）

表 2　実行時間

①のマシン環境の場合

メッシュ図	実行時間
1	31.259
2	40.191

②のマシン環境の場合

メッシュ図	実行時間
1	60.222
2	81.971

（単位：秒）

3.6. 考察

図 9 のような出力結果が得られ、表 2 のような実行時間となった。表 2 からメッシュを細かく取った場合、その分計算が増えるため実行時間がかかった。実行を行ったハイスペックの PC 環境においても実行に時間がかかることが分かる。出力結果は可視化して確認することで正しい結果か確認するべきだが、可視化部分が実現されていないためこの結果を正しい結果と仮定する。

4. 有限要素法によるカルマン渦の並列化

4.1. 問題定義

3.で有限要素法によるカルマン渦の流体解析のプログラミングを行い、実行結果、実行時間を得た。その結果、本研究室のハイスペックなマシン環境でもかなりの時間がかかることが分かった。よって、有限要素法の並列化アルゴリズムを検討し、プログラミングを行い処理速度の向上を実現する。

4.2. 並列化アルゴリズム

処理に時間がかかる問題を並列処理するときに、はじめに考えるのは全体ループの並列化である。ループの処理範囲を各プロセッサに割り当て、それぞれの処理が終われば結果を収集し最終的な結果を出すということで並列効果が得ることができる。しかし、流体解析は時間積分によって処理を行う。つまり、流体解析における全体ループは時間によるループであり、時間 i+1 のときの処理は時間 i の結果を利用して処理を行うため全体ループでの並列化は不可能であることが分かる。よって、流体解析の並列化を行うには処理する領域を分割し、各プロセッサに分割領域ごとに処理させることで並列化をさせる。

図 11 が本研究で作成したカルマン渦の流体解析プログラムのフローチャートである。図 11 中の右の図は、有限要素法による流速、圧力の計算処理と、データの流れを示している。本プログラムで並列処理させる部分は有限要素法による流速計算の部分である。プログラムの性質上、圧力 p^{n+1} の計算は逐次処理で計算をさせなければ、正常な結果が得られないため圧力計算の並列化は難しい。

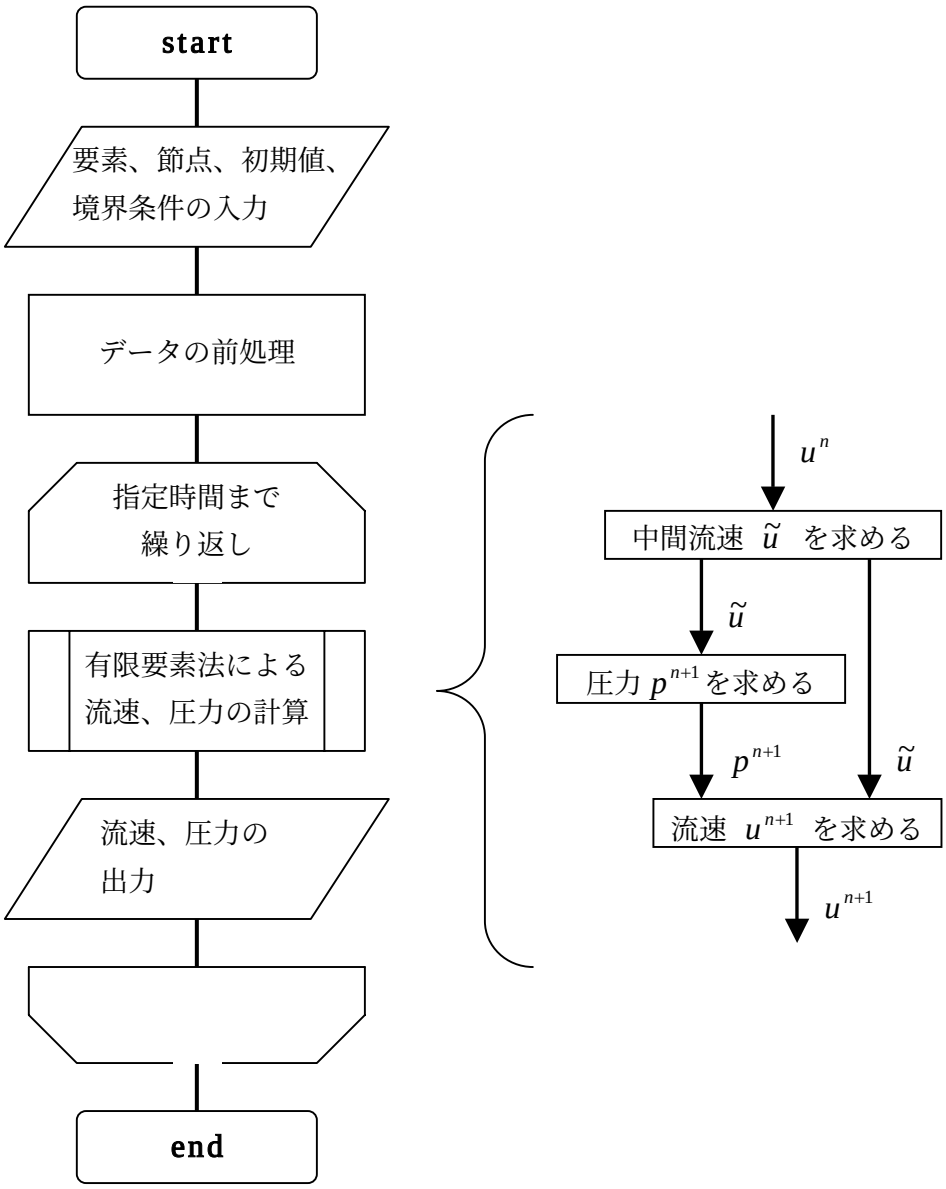


図 13　フローチャート

流速の計算は各メッシュで独立に計算が行えるため、プロセスネットワークで並列処理が可能である。その方法は、メッシュデータに対して、ある範囲で複数に分割する。その範囲の中のメッシュデータを使って、各 PC に流速計算を行わせる。

解析領域はデータでメッシュに区切ってある。図 11 は解析領域をメッシュに区切った様子を示している。そのメッシュデータをある範囲に基づいて分割する。つまり、プロセスネットワークの計算ステージは、有限要素法のメッシュデータにあたる。分割したメッシュデータを、各 PC に割り当て処理をさせる。図 11 の場合、メッシュデータを 5 つの範囲

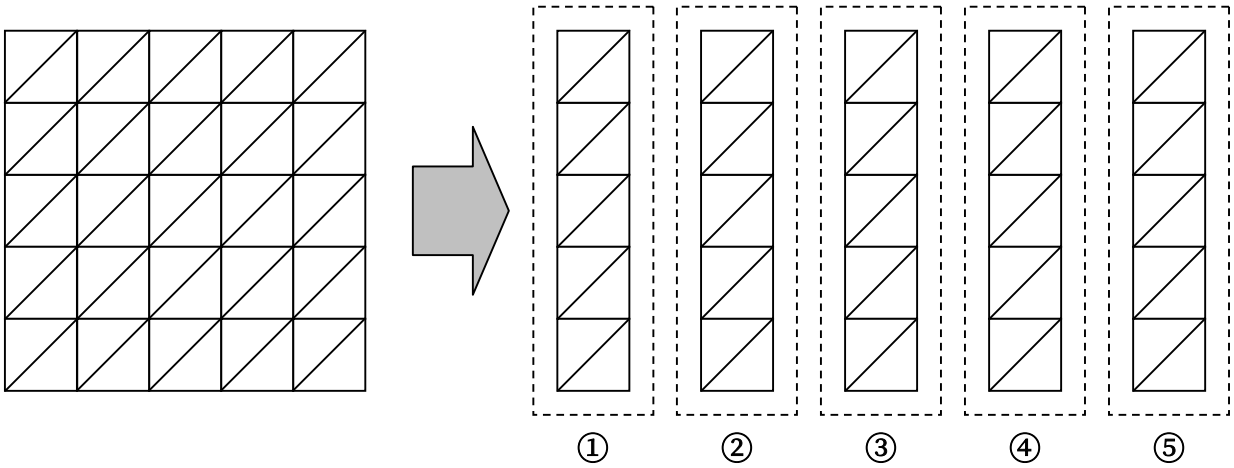


図 14　領域分割

に分割してあるため、PC4 台で実行する場合 1 つないしは 2 つの範囲を計算することになり、PC5 台では１つの範囲を計算する。本研究では、このようにして並列処理をさせる。

OpenMP によるプログラミングにおいて、全プロセッサに変数を共有する共有変数と、各プロセッサに占有させる変数がそれぞれ宣言できる。OpenMP プログラムを SCore 型 PC クラスタで実行させる。SCore は各 PC に分散しているメモリをソフトウェア的に共有メモリに見せて動作している。そのため、プログラム中の共有変数の量が多くなると、データの送受信が増えるため並列効果が得られない可能性がある。プログラム中の共有変数を、流速u^n、u^{n+1}、中間流速 $\tilde{u}$、メッシュデータ（節点、要素）とし、各 PC が占有する変数を 4.2.で述べた各要素で計算で用いる変数を指定する。有限要素法の並列化は図 12 のようになる。本並列アルゴリズムは解析領域をいくつかの領域に分解し、その部分領域のメッシュデータを編集したのち、図 12 のように並列処理を行う。また、本アルゴリズムの拡張として、メッシュの分割を PC の台数に合わせて分割するのではなく、PC の台数の 2 倍や 3 倍に分割をすることで､負荷分散を行うサイクリック分割をすることも可能である。

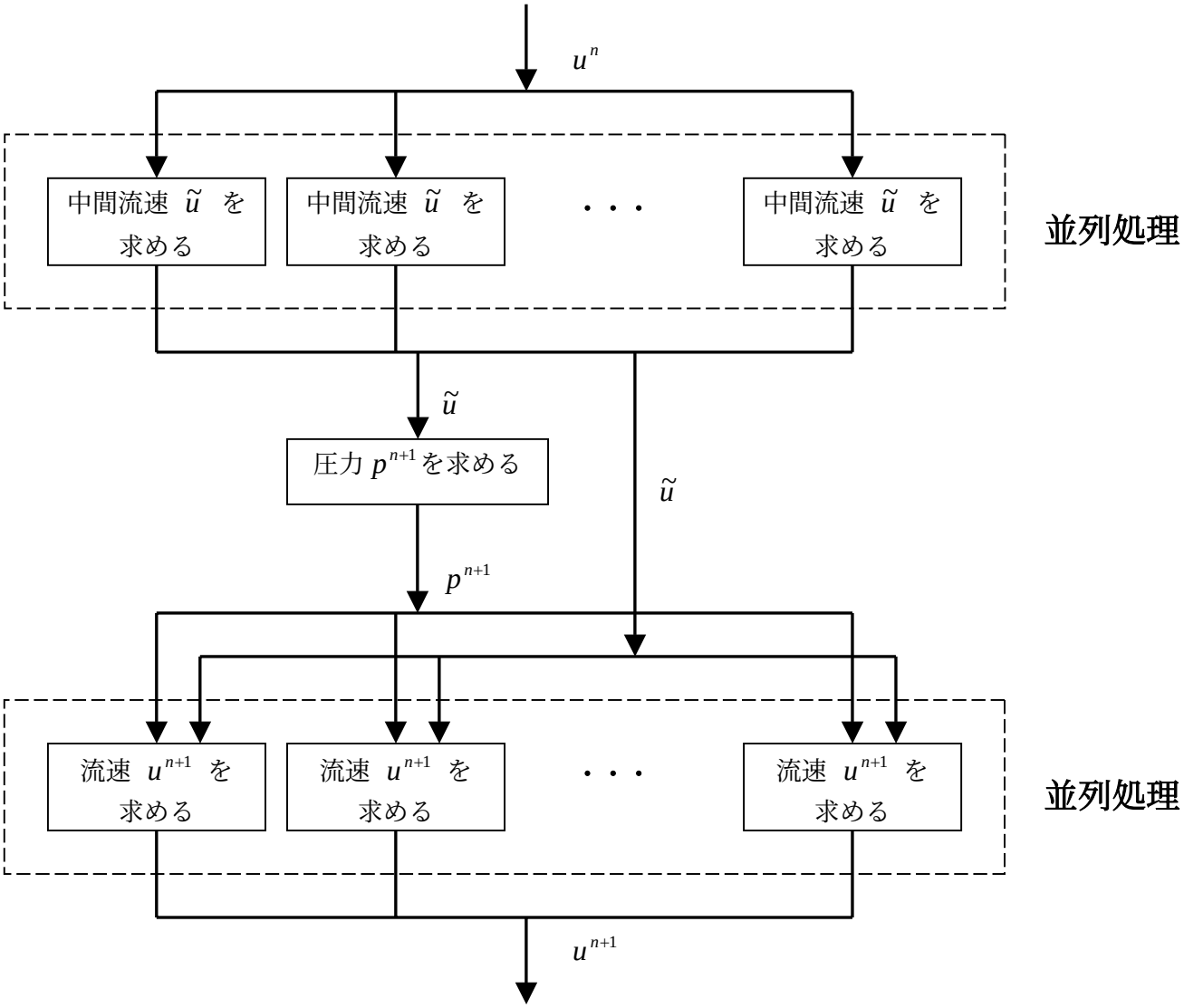


図 15　有限要素法の並列化

4.3. 実行結果

メッシュデータを 1：『1375 節点 2530 要素』、2：『2343 節点 4404 要素』、3：『3372 節点 6399 要素』、4：『5335 節点 10241 要素』、5：『10243 節点 19885 要素』のデータを用いた。微小時間を 0.01、1 回ループを行った。またクラスタの台数を１、２，４、８、16 の場合に分けてデータを得た。実測値は実行時間を 5 つ取りその平均を出したものとする。その実測値を以下の表 3 に示す。計測区間は並列処理させた部分のみとする。

今回並列化した部分は、中間流速の計算処理である。表 3 は#pragma omp for を並列指示文とし、コンパイラ上で自動並列をさせた場合の実行結果である。

表 3　並列処理の実行時間

	1	2	4	8	16
1375節点 2530要素	0.00686	0.00660	0.00485	0.00480	0.00506
2343節点 4404要素	0.01199	0.01076	0.00721	0.00623	0.00590
3372節点 6399要素	0.01757	0.01340	0.00906	0.00681	0.00623
5335節点 10241要素	0.02818	0.01627	0.01014	0.00750	0.00671
10243節点 19885要素	0.05470	0.02921	0.01609	0.01066	0.00907

（単位：秒）

表 4　全体処理の実行時間

	1	2	4	8	16
1375節点 2530要素	0.3269	0.3283	0.3272	0.3382	0.3478
2343節点 4404要素	0.6533	0.6537	0.6657	0.6806	0.6883
3372節点 6399要素	0.9772	0.9853	1.0009	1.0061	1.0214
5335節点 10241要素	1.6125	1.6303	1.6611	1.6833	4.4454
10243節点 19885要素	3.2885	3.3480	3.3860	3.4007	3.4078

（単位：秒）

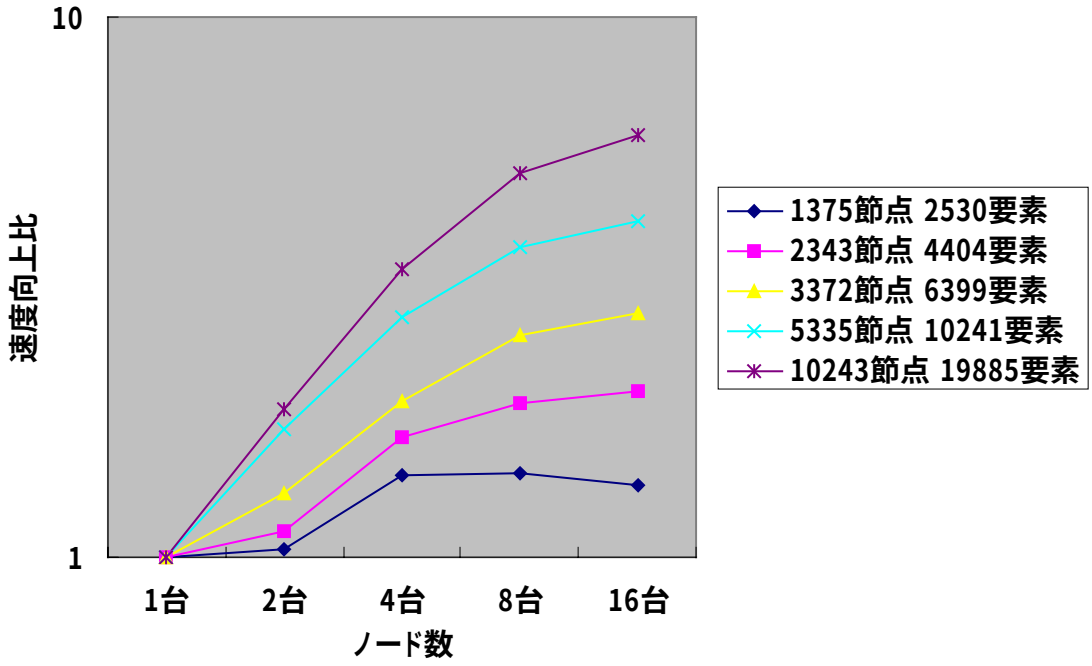


図 16　速度向上比

4.4 考察

実行結果から、並列処理部分の理想的な速度向上ではないものの速度向上は得られた。データに負担が少ないとき、バリア同期による遅延が発生し安定した速度向上は得られなかったと考えられる。データが大きい場合に、ノード数を増やすにつれて速度向上比の低下が見られるのは、先ほどと同様に、分割データの負担が少なくなるためバリア同期による遅延が発生したと考えられる。また、全体処理の時間は明らかに上がっていない。それは、圧力計算の ICCG が逐次処理でなければ正確な解が求められないため、ICCG 法を逐次処理で行わなければならないことが挙げられる。

5. おわりに

本研究では、ＰＣクラスタ上における有限要素法の並列化を行ってきた。本プログラムでは、圧力計算の処理部分が並列化に向いていなかったため、流速計算の処理を並列処理をさせた。並列処理部分の理想的な並列効果は得られなかったが計算時間は速くなった。しかし、全体処理の時間は、速度が上がったとは言えない結果であった。その原因として、圧力計算の ICCG 法の逐次性が高いことが挙げられる。有限要素法での並列化アルゴリズムとして、並列計算機が分散環境上で考えられた領域分割法という方法がある。このアルゴリズムは、領域を分割し、分割した境界部分にお互いの境界条件を与え、全体の処理を可能にしたアルゴリズムである。[13]今後は、この領域分割法のアルゴリズムを OpenMP 環境で検討したい。その他の課題として、流体解析から得られたデータを可視化すること、有限要素法による流体解析のほかに、流体を粒子として考え解析をする粒子法（MPS 法）の検討が挙げれる。

謝辞

本研究の機会を与えてくださり、数々の助言を頂きました山崎勝弘教授、小柳滋教授に心より感謝いたします。また、本研究にあたり、励ましの言葉や様々な貴重なご意見を頂き、また質疑に答えていただくなどした本研究室の皆様に心より感謝いたします。

参考文献

[1]PC Cluster Consortium http://www.pccluster.org/
[2] R. Chandra, L. Fagum, D. Kohr, D. Maydan, J. McDonald, R. Menon:"Parallel Programming in OpenMP", MORGAN KAUFMANN PUBLISHERS, 2000.
[3] 湯浅太一,安村通晃,中田登志之 編:”はじめての並列プログラミング”bit別冊,共立出版,1999
[4]石川祐,他:”Linuxで並列処理をしよう”,共立出版,2002.
[5]日本数値流体力学学会有限要素法研究委員会 編:”有限要素法による流れのシミュレーション”,シュプリンガー・ファアラーク東京,1998.
[6]森 正武:”Frotran77 数値計算プログラミング 増補版”,岩波書店,1987.
[7]三浦 誉大：PCクラスタ上での並列プログラミング環境の構築：立命館大学理工学部情報学科卒業論文、2002
[8]大村 浩文：PCクラスタの動作テストとOpenMP並列プログラミング：立命館大学理工学部情報学科卒業論文、2002
[9]内田 大介：OpenMPによる並列プログラミング：立命館大学理工学部情報学科卒業論文、2000
[10]矢川 元基：“パソコンで見る流れの科学 数値流体力学入門”, ブルーバックス, 2001
[11]三木 光範 他：“PC クラスタ超入門 2000”、PC クラスタ型並列計算機の構築と利用、超並列計算研究会、2000
[12]横川 三津夫：“地球シミュレータの完成を向かえて”、並列処理シンポジウム JSPP2002、2002
[13]矢川元基、奥田洋司共編：“計算力学 7 計算力学における超並列計算法”、養賢堂、2002

付録 A：剛性方程式の導出と、3 節点の剛性方程式の行列化

A.1. 重み付き残差法

重み付き残差法とは、偏微分方程式と境界条件から，体積積分方程式を導出する方法のひとつで、もとの方程式に重み関数と呼ばれる関数をかけて積分したものを 0 と置く方法である。

式（3.10)、(3.12)、(3.14）を空間微分の項を有限要素を用いて定式化する。まず、重み関数u_i^*、 p^*を用いて次の重み付き残差方程式を誘導する。

$$\int_\Omega u_i^* \tilde{u}_i d\Omega = \int_\Omega u_i^* u_i^n d\Omega - \Delta t \left[\int_\Omega u_i^* u_i^n \frac{\partial u_i^n}{\partial x_j} d\Omega - \frac{1}{Re} \underbrace{\int_\Omega u_i^* \frac{\partial}{\partial x_j} \left(\frac{\partial u_i^n}{\partial x_j} + \frac{\partial u_j^n}{\partial x_i} \right) d\Omega}_{\text{部分積分する}} \right] \quad \text{(A.1)}$$

$$\underbrace{\int_\Omega p^* \frac{\partial^2 p^{n+1}}{\partial x_i^2} d\Omega}_{\text{部分積分する}} = \frac{1}{\Delta t} \int_\Omega p^* \frac{\partial \tilde{u}_i}{\partial x_i} d\Omega \quad \text{(A.2)}$$

$$\int_\Omega u_i^* u_i^{n+1} d\Omega = \int_\Omega u_i^* \tilde{u}_i d\Omega - \Delta t \int_\Omega u_i^* \frac{\partial p^{n+1}}{\partial x_i} d\Omega \quad \text{(A.3)}$$

次に、式（A.1）及び式（A.2）中に示したように、2 次式の微係数の項を部分積分し、発散定理を適用して次のように変形する。

$$\int_\Omega u_i^* \tilde{u}_i d\Omega = \int_\Omega u_i^* u_i^n d\Omega - \Delta t \left[\int_\Omega u_i^* u_j^n \frac{\partial u_i^n}{\partial x_j} d\Omega - \frac{1}{Re} \underbrace{\int_\Gamma u_i^* \left(\frac{\partial u_i^n}{\partial x_j} + \frac{\partial u_j^n}{\partial x_i} \right) \cdot n_j d\Gamma}_{=0 \text{ (自然境界条件)}} + \frac{1}{Re} \int_\Omega \frac{\partial u_i^*}{\partial x_j} \left(\frac{\partial u_i^n}{\partial x_j} + \frac{\partial u_j^n}{\partial x_i} \right) d\Omega \right] \quad \text{(A.4)}$$

$$\underbrace{\int_\Gamma p^* \frac{\partial p^{n+1}}{\partial x_i} \cdot n_i d\Gamma}_{=0 \text{ (自然境界条件)}} - \int_\Omega \frac{\partial p^*}{\partial x_i} \frac{\partial p^{n+1}}{\partial x_i} d\Omega = \frac{1}{\Delta t} \int_\Omega p^* \frac{\partial \tilde{u}_i}{\partial x_i} d\Omega \quad \text{(A.5)}$$

これらの式中の$\int_\Gamma ( ) d\Gamma$ の積分は解析領域Ωの境界Γ上で“ ( ) ”内の自然境界条件の積分

することを表しており、特別な場合を除いてゼロとして取り扱われることが多い。

以上から、求めるべき方程式の重み付き残差方程式をまとめると次のようになる。

$$\int_\Omega u_i^* \tilde{u}_i d\Omega = \int_\Omega u_i^* u_i^n d\Omega - \Delta t \left[\int_\Omega u_i^* u_j^n \frac{\partial u_i^n}{\partial x_j} d\Omega + \frac{1}{Re} \int_\Omega \frac{\partial u_i^*}{\partial x_j} \left(\frac{\partial u_i^n}{\partial x_j} + \frac{\partial u_j^n}{\partial x_i} \right) d\Omega \right] \quad \text{(A.6)}$$

$$\int_\Omega \frac{\partial p^*}{\partial x_i} \frac{\partial p^{n+1}}{\partial x_i} d\Omega = -\frac{1}{\Delta t} \int_\Omega p^* \frac{\partial \tilde{u}_i}{\partial x_i} d\Omega \quad \text{(A.7)}$$

$$\int_\Omega u_i^* u_i^{n+1} d\Omega = \int_\Omega u_i^* \tilde{u}_i d\Omega - \Delta t \int_\Omega u_i^* \frac{\partial p^{n+1}}{\partial x_i} d\Omega \quad \text{(A.8)}$$

A.2. 補間関数

有限要素法では有限要素と呼ばれる部分領域Ω_eで解析領域Ωを分割する。有限要素法では未知変数をこの分割領域Ω_eごとにある種の関数（補間関数）によって近似している。補間関数の取り方には非常に多くの方法があるが、ここでは、最も簡単な三角形 1 次有限要素による補間法を解説する。

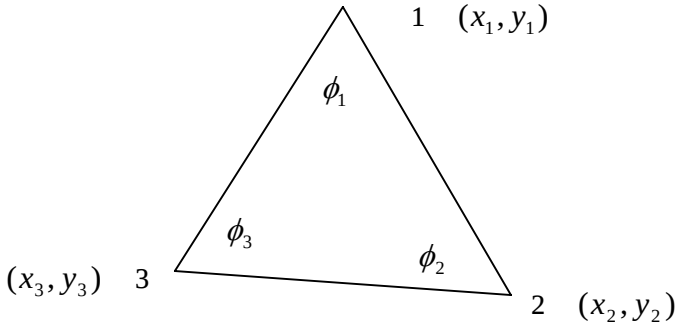


図 A1：要素節点と未知変数

いま、図 5 に示すような三角形領域があり、この領域内の任意の点の未知関数を 3 節点での値より補完することを考える。そこで、要素内における未知関数ϕの分布を 2 次元座標x、yを用いて 1 次式で表すものと仮定する。すなわち、

$$\phi = \eta_1 + \eta_2 x + \eta_3 y \quad \text{(A.9)}$$

ここに、η_i（$i = 1 \sim 3$）は未定定数である。この関数は 3 節点の関数値$\phi_1 \sim \phi_3$に関しても成立するはずであり、

$$\left.\begin{aligned} \phi_1 &= \eta_1 + \eta_2 x_1 + \eta_3 y_1 \\ \phi_2 &= \eta_1 + \eta_2 x_2 + \eta_3 y_2 \\ \phi_3 &= \eta_1 + \eta_2 x_3 + \eta_3 y_3 \end{aligned}\right\} \quad \text{(A.10)}$$

なる関係が得られる。これをη_i（$i = 1 \sim 3$）について解くと、

$$\left.\begin{aligned}\eta_1 &= a_1\phi_1 + a_2\phi_2 + a_3\phi_3\\ \eta_2 &= b_1\phi_1 + b_2\phi_2 + b_3\phi_3\\ \eta_3 &= c_1\phi_1 + c_2\phi_2 + c_3\phi_3\end{aligned}\right\} \qquad (A.11)$$

が得られる。ただし、

$$\left.\begin{aligned}a_i &= \frac{1}{2\Delta}\left(x_j y_k - x_k y_i\right)\\ b_i &= \frac{1}{2\Delta}\left(y_j - y_k\right) \qquad (i,k,j = 1,2,3)\\ a_i &= \frac{1}{2\Delta}\left(x_k - x_j\right)\end{aligned}\right\} \qquad (A.12)$$

であり、Δは三角形要素の面積で、

$$\Delta = \frac{1}{2}\left[x_i\left(y_i - y_k\right) + x_j\left(y_k - y_i\right) + x_k\left(y_i - y_j\right)\right] \qquad (A.13)$$

である。以上の関係を整理して、先の補間関数は次のように表される。

$$\phi = N_1\phi_1 + N_2\phi_2 + N_3\phi_3 \qquad (A.14)$$

ここに、N_iは

$$\left.\begin{aligned}N_1 &= a_1 + b_1 x + c_1 y\\ N_2 &= a_2 + b_2 x + c_2 y\\ N_3 &= a_3 + b_3 x + c_3 y\end{aligned}\right\} \qquad (A.15)$$

であり、形状関数と呼ばれる。

A.3. 剛性方程式の誘導

前節の重み付き残差方程式（A.6）～（A.8）において、未知変数であったものは流速$\tilde{u}_i$、u_iおよび圧力pである。そこで、三角形 1 次有限要素の形状関数を用いて未知変数の補間関数を次のように定める。

$$\left.\begin{aligned}\tilde{u}_i &= N_\alpha \tilde{u}_{\alpha i}\\ u_i &= N_\alpha u_{\alpha i} \qquad (i = 1,2 \text{ 、 } \alpha = 1\sim 3)\\ p &= N_\alpha p_\alpha\end{aligned}\right\} \qquad (A.16)$$

ここで、$\tilde{u}_i$、u_i、pは三角形要素内に分布する未知関数であり、$\tilde{u}_{\alpha i}$、$u_{\alpha i}$、p_αは 3 節点における未知関数の値である。また、添字iは 2 次元ベクトルの添字であり、これと混同することをさけるため、三角形要素の 3 節点を表す記号としてαを用いた。次に、それぞれの重み関数u_i^*、p^*も補間関数を先と同様に定義する。

$$\left.\begin{aligned} u_i^* &= N_\alpha u_{\alpha i}^* \\ p^* &= N_\alpha p_\alpha^* \end{aligned} \qquad (i=1,2\ 、\ \alpha=1\sim3) \right\} \tag{A.17}$$

このように未知関数と重み付き関数に同じ形の補間関数を用いることをガラーキン法という。さて、補間関数を用いて未知関数と重み付き関数の近似式が得られたことから、各変数の偏微分の計算があらかじめ行えることになる。例えば、

$$\frac{\partial u}{\partial x}=\frac{\partial N_\alpha}{\partial x}u_\alpha=\frac{\partial\left(a_\alpha+b_\alpha x+c_\alpha y\right)}{\partial x}u_\alpha=b_\alpha u_\alpha=\left\langle b_1 b_2 b_3\right\rangle\begin{Bmatrix}u_1\\u_2\\u_3\end{Bmatrix} \quad 、$$

$$\frac{\partial v}{\partial x}=\frac{\partial N_\alpha}{\partial y}v_\alpha=\frac{\partial\left(a_\alpha+b_\alpha x+c_\alpha y\right)}{\partial y}v_\alpha=c_\alpha v_\alpha=\left\langle c_1 c_2 c_3\right\rangle\begin{Bmatrix}v_1\\v_2\\v_3\end{Bmatrix} \tag{A.18}$$

のようになる。未知変数の空間微分が、補間関数を用いることにより形状関数の微分に置き換えられる。すなわち、未知変数の微分は形状関数の導関数と三角形要素の節点値との積で表されるということである。以上を用い、三角形要素内で近似された未知変数の補間関数を用いて先の重み付き残差方程式を書き直すと、以下のようになる。

$$\int_{\Omega_e}N_\alpha N_\beta d\Omega\tilde{u}_{\beta i}=\int_{\Omega_e}N_\alpha N_\beta d\Omega u_{\beta i}^n-\Delta t\left[\int_{\Omega_e}N_\alpha N_\beta\frac{\partial N_\gamma}{\partial x_\gamma}d\Omega u_{\beta j}^n u_{\gamma i}^n\right.$$

$$\left.+\frac{1}{Re}\left\{\int_{\Omega_e}\frac{\partial N_\alpha}{\partial x_j}\frac{\partial N_\beta}{\partial x_j}d\Omega u_{\beta i}^n+\int_{\Omega_e}\frac{\partial N_\alpha}{\partial x_j}\frac{\partial N_\beta}{\partial x_i}d\Omega u_{\beta j}^n\right\}\right] \tag{A.19}$$

$$\int_{\Omega_e}\frac{\partial N_\alpha}{\partial x_i}\frac{\partial N_\beta}{\partial x_i}d\Omega p_{\beta i}^{n+1}=-\frac{1}{\Delta t}\int_{\Omega_e}N_\alpha\frac{\partial N_\beta}{\partial x_i}d\Omega\tilde{u}_{\beta j} \tag{A.20}$$

$$\int_{\Omega_e}N_\alpha N_\beta d\Omega u_\beta^{n+1}=\int_{\Omega_e}N_\alpha N_\beta d\Omega\tilde{u}_{\beta i}-\Delta t\int_{\Omega_e}N_\alpha\frac{\partial N_\beta}{\partial x_i}d\Omega p_\beta^{n+1} \tag{A.21}$$

また、これらを行列の形に書き直すと以下のようになる。

$$\overline{M}_{\alpha\beta}=\overline{M}_{\alpha\beta}u_{\beta i}^n-\Delta t\left[K_{\alpha\beta\gamma j}u_{\beta j}^n+\frac{1}{Re}S_{\alpha i\beta j}u_{\beta j}^n\right] \tag{A.22}$$

$$A_{\alpha i\beta i}p_\beta^{n+1}=-\frac{1}{\Delta t}H_{\alpha\beta i}u_{\beta i} \tag{A.23}$$

$$\overline{M}_{\alpha\beta}u_\beta^{n+1}=\overline{M}_{\alpha\beta}\tilde{u}_{\beta i}-\Delta t H_{\alpha\beta i}p_\beta^{n+1} \tag{A.24}$$

ただし、

$$\overline{M}_{\alpha\beta} = \int_{\Omega_e} N_\alpha N_\beta d\Omega \quad 、 \qquad K_{\alpha\beta\gamma j} = \int_{\Omega_e} N_\alpha N_\beta \frac{\partial N_\gamma}{\partial x_\gamma} d\Omega u^n_{\gamma i}$$

$$S_{\alpha i\beta j} = \int_{\Omega_e} \frac{\partial N_\alpha}{\partial x_k}\frac{\partial N_\beta}{\partial x_k} d\Omega \cdot \delta_{ij} + \int_{\Omega_e} \frac{\partial N_\alpha}{\partial x_j}\frac{\partial N_\beta}{\partial x_i} d\Omega$$

$$A_{\alpha i\beta i} = \int_{\Omega_e} \frac{\partial N_\alpha}{\partial x_i}\frac{\partial N_\beta}{\partial x_i} d\Omega \quad 、 \qquad H_{\alpha\beta i} = \int_{\Omega_e} N_\alpha \frac{\partial N_\beta}{\partial x_i} d\Omega$$

である。一般に式（3.36）～（3.38）を剛性方程式と呼び、式中の行列は係数行列と呼ばれる。また、式中の $\overline{M}_{\alpha\beta}$ は $M_{\alpha\beta}$ を対角集中化した集中質量行列である。上記の係数行列は次節で説明する。

A.4. 係数行列の積分

有限要素が三角形要素である場合には、係数行列を求める時に面積座標を用いるのが便利である。本節では、その方法について説明する。いま、図 A2 に示した三角要素内部の任意の 1 点（x、y）をとり、この点を頂点とした 3 つの三角形 Δ_1～Δ_3 を考える。

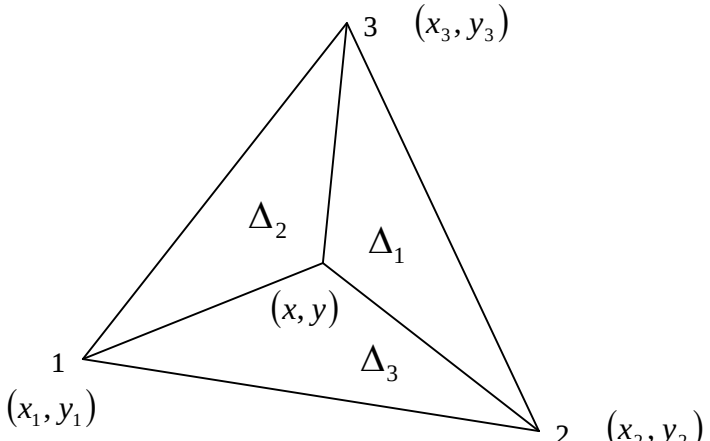


図 A2：要素節点と未知変

このとき、元の三角形要素の面積を Δ とすると次のような面積座標と呼ばれる座標が定義できる。

$$\xi_1 = \frac{\Delta_1}{\Delta}、 \qquad \xi_2 = \frac{\Delta_2}{\Delta}、 \qquad \xi_3 = \frac{\Delta_3}{\Delta} \qquad (A.25)$$

この面積座標を先の点（x、y）を用いて実際に計算すると、

$$\xi_1 = \frac{\Delta_1}{\Delta} = \frac{1}{2\Delta}\{x_2y_2 - x_3y_2 + x(y_2 - y_3) + y(x_3 - x_2)\}$$
$$= a_1 + b_1x + c_1y$$

となる。同様に求めると

$$\left.\begin{aligned}\xi_1 &= a_1 + b_2 x + c_1 y \\ \xi_2 &= a_2 + b_2 x + c_2 y \\ \xi_3 &= a_3 + b_3 x + c_3 y\end{aligned}\right\} \tag{A.26}$$

が得られる。ここにa_i、b_i、c_iは式（A.12）によって与えられるものである。この 3 座標はそれぞれが独立でなく、

$$\xi_1 + \xi_2 + \xi_3 = 1 \tag{A.27}$$

なる関係にある。面積座標ξ_iは先の形状関数N_iの式（3.29）と全く等しく、したがって形状関数の積分が面積関数の積分であることがわかる。また、形状関数の微係数も面積座標の微係数と等価である。このことから、面積座標を使って係数行列の積分が可能になる。

面積座標の積分には、次の面積座標積分公式が用いられる。

$$\iint \left(\xi_1^l \xi_2^m \xi_3^n\right) dxdy = 2\frac{l!m!n!}{(l+m+n+2)!}\Delta \tag{A.28}$$

ここに、積分は三角形要素内にて行うものである。また、形状関数の微係数も面積座標の微係数に置き換えられ、次のように計算することができる。

$$\frac{\partial}{\partial x}\begin{Bmatrix}\xi_1 \\ \xi_2 \\ \xi_3\end{Bmatrix} = \begin{Bmatrix}b_1 \\ b_2 \\ b_3\end{Bmatrix} \quad 、 \qquad \frac{\partial}{\partial y}\begin{Bmatrix}\xi_1 \\ \xi_2 \\ \xi_3\end{Bmatrix} = \begin{Bmatrix}c_1 \\ c_2 \\ c_3\end{Bmatrix} \tag{A.29}$$

以上から、係数行列のすべての計算が可能になった。そこで次に、各係数行列を実際に計算したものを以下に示す。

（1）質量行列$M_{\alpha\beta}$の計算

$$M_{\alpha\beta} = \int_{\Omega_e} N_\alpha N_\beta d\Omega = \int_{\Omega_e} \begin{Bmatrix}N_1 \\ N_2 \\ N_3\end{Bmatrix} \langle N_1 N_2 N_3 \rangle d\Omega = \int_{\Omega_e} \begin{Bmatrix}\xi_1 \\ \xi_2 \\ \xi_3\end{Bmatrix} \langle \xi_1 \xi_2 \xi_3 \rangle d\Omega$$

$$= \int_{\Omega_e} \begin{bmatrix}\xi_1^2 & \xi_1\xi_2 & \xi_1\xi_3 \\ \xi_2\xi_1 & \xi_2^2 & \xi_2\xi_3 \\ \xi_3\xi_1 & \xi_3\xi_2 & \xi_3^2\end{bmatrix} d\Omega = \frac{\Delta}{12}\begin{bmatrix}2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2\end{bmatrix}$$

（2）移流項の行列 $K_{\alpha\beta\gamma j}$ の計算

＜$i=1$（x 方向）の場合＞

$$K_{\alpha\beta\gamma 1}=\int_{\Omega_e} N_\alpha N_\beta \frac{\partial N_\gamma}{\partial x} d\Omega u_\gamma^n = \left(b_1u_1+b_2u_2+b_3u_3\right)\cdot\int_{\Omega_e}\begin{bmatrix} N_1^2 & N_1N_2 & N_1N_3 \\ N_2N_1 & N_2^2 & N_2N_3 \\ N_3N_1 & N_3N_2 & N_3^2 \end{bmatrix} d\Omega$$

$$=\frac{\Delta}{12}\left(b_1u_1+b_2u_2+b_3u_3\right)\cdot\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}$$

$$K_{\alpha\beta\gamma 2}=\int_{\Omega_e} N_\alpha N_\beta \frac{\partial N_\gamma}{\partial x} d\Omega u_\gamma^n = \frac{\Delta}{12}\left(c_1u_1+c_2u_2+c_3u_3\right)\cdot\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}$$

＜$i=2$（y 方向）の場合＞

$$K_{\alpha\beta\gamma 1}=\int_{\Omega_e} N_\alpha N_\beta \frac{\partial N_\gamma}{\partial x} d\Omega u_\gamma^n = \frac{\Delta}{12}\left(b_1v_1+b_2v_2+b_3v_3\right)\cdot\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}$$

$$K_{\alpha\beta\gamma 2}=\int_{\Omega_e} N_\alpha N_\beta \frac{\partial N_\gamma}{\partial x} d\Omega u_\gamma^n = \frac{\Delta}{12}\left(c_1v_1+c_2v_2+c_3v_3\right)\cdot\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}$$

（3）粘性項の行列 $S_{\alpha i\beta j}$ の計算

＜$i=1$（x 方向）の場合＞

$$S_{\alpha 1\beta 1}=\int_{\Omega_e}\frac{\partial N_\alpha}{\partial x}\frac{\partial N_\beta}{\partial x}d\Omega+\int_{\Omega_e}\frac{\partial N_\alpha}{\partial x}\frac{\partial N_\beta}{\partial x}d\Omega = 2\Delta\begin{bmatrix} b_1^2 & b_1b_2 & b_1b_3 \\ b_2b_1 & b_2^2 & b_2b_3 \\ b_3b_1 & b_3b_2 & b_3^2 \end{bmatrix}$$

$$S_{\alpha 1\beta 2}=\int_{\Omega_e}\frac{\partial N_\alpha}{\partial x}\frac{\partial N_\beta}{\partial y}d\Omega = \Delta\begin{bmatrix} b_1c_1 & b_1c_2 & b_1c_3 \\ b_2c_1 & b_2c_2 & b_2c_3 \\ b_3c_1 & b_3c_2 & b_3c_3 \end{bmatrix}$$

＜i＝2（y 方向）の場合＞

$$S_{\alpha 2\beta 1}=\int_{\Omega_e}\frac{\partial N_\alpha}{\partial y}\frac{\partial N_\beta}{\partial x}d\Omega=\Delta\begin{bmatrix}c_1b_1 & c_1b_2 & c_1b_3\\ c_2b_1 & c_2b_2 & c_2b_3\\ c_3b_1 & c_3b_2 & c_3b_3\end{bmatrix}$$

$$S_{\alpha 2\beta 2}=\int_{\Omega_e}\frac{\partial N_\alpha}{\partial y}\frac{\partial N_\beta}{\partial y}d\Omega+\int_{\Omega_e}\frac{\partial N_\alpha}{\partial y}\frac{\partial N_\beta}{\partial y}d\Omega=2\Delta\begin{bmatrix}c_1^2 & c_1c_2 & c_1c_3\\ c_2c_1 & c_2^2 & c_2c_3\\ c_3c_1 & c_3c_2 & c_3^2\end{bmatrix}$$

（4）ポアソン方程式左辺の行列 $A_{\alpha i\beta j}$ の計算

$$A_{\alpha i\beta j}=A_{\alpha 1\beta 1}+A_{\alpha 2\beta 2}$$

$$=\int_{\Omega_e}\frac{\partial N_\alpha}{\partial x}\frac{\partial N_\beta}{\partial x}d\Omega+\int_{\Omega_e}\frac{\partial N_\alpha}{\partial y}\frac{\partial N_\beta}{\partial y}d\Omega=\Delta\begin{bmatrix}b_1^2 & b_1b_2 & b_1b_3\\ b_2b_1 & b_2^2 & b_2b_3\\ b_3b_1 & b_3b_2 & b_3^2\end{bmatrix}+\Delta\begin{bmatrix}c_1^2 & c_1c_2 & c_1c_3\\ c_2c_1 & c_2^2 & c_2c_3\\ c_3c_1 & c_3c_2 & c_3^2\end{bmatrix}$$

（5）ポアソン方程式右辺の行列 $H_{\alpha\beta j}$ の計算

＜i＝1（x 方向）の場合＞

$$H_{\alpha\beta 1}=\int_{\Omega_e}N_\alpha\frac{\partial N_\beta}{\partial x}d\Omega=\frac{\Delta}{3}\begin{bmatrix}b_1 & b_2 & b_3\\ b_1 & b_2 & b_3\\ b_1 & b_2 & b_3\end{bmatrix}$$

＜i＝2（y 方向）の場合＞

$$H_{\alpha\beta 2}=\int_{\Omega_e}N_\alpha\frac{\partial N_\beta}{\partial y}d\Omega=\frac{\Delta}{3}\begin{bmatrix}c_1 & c_2 & c_3\\ c_1 & c_2 & c_3\\ c_1 & c_2 & c_3\end{bmatrix}$$

なお、質量行列 $M_{\alpha\beta}$ は、対角集中化された集中化行列 $\overline{M}_{\alpha\beta}$ に変換されることがある。この行列を用いることの利点は、実際に連立方程式を解く際に、未知数ベクトルに掛かる係数行列を対角集中化しておくことで逆行列の計算が、単に係数の逆数を右辺側へ掛けるだけとなる点である。このような方法を陽的解法と言う。$\overline{M}_{\alpha\beta}$ を計算すると、次のようになる。

質量行列の集中化行列 $\overline{M}_{\alpha\beta}$ の計算

$$\overline{M}_{\alpha\beta}=\frac{\Delta}{3}\begin{bmatrix}1 & & \\ & 1 & \\ & & 1\end{bmatrix}$$

ここで、行列中の空白部分は各要素における係数が 0 であることを意味している。

以上から、係数行列の積分ができたことになる。そこで、これらを用いて先の剛性方程式（A.22）～（A.24）をすべて書き下すと次のようになる。

・中間流速の式（A.22）

＜i＝1（x 方向）の場合＞

$$\overline{M}_{\alpha\beta}\tilde{u}_{\beta}=\overline{M}_{\alpha\beta}u_{\beta}^{n}-\Delta t\left\{K_{\alpha\beta\gamma1}u_{\beta}^{n}+K_{\alpha\beta\gamma2}v_{\beta}^{n}+\frac{1}{Re}\left(S_{\alpha1\beta1}u_{\beta}^{n}+S_{\alpha1\beta2}v_{\beta}^{n}\right)\right\}$$

$$\Downarrow$$

$$\frac{\Delta}{3}\begin{bmatrix}1 & & \\ & 1 & \\ & & 1\end{bmatrix}\begin{Bmatrix}\tilde{u}_1\\ \tilde{u}_2\\ \tilde{u}_3\end{Bmatrix}=\frac{\Delta}{3}\begin{bmatrix}1 & & \\ & 1 & \\ & & 1\end{bmatrix}\begin{Bmatrix}u_1^n\\ u_2^n\\ u_3^n\end{Bmatrix}$$

$$-\Delta t\left\{\frac{\Delta}{12}\left(b_1u_1+b_2u_2+b_3u_3\right)\cdot\begin{bmatrix}2&1&1\\1&2&1\\1&1&2\end{bmatrix}\begin{Bmatrix}u_1^n\\u_2^n\\u_3^n\end{Bmatrix}+\frac{\Delta}{12}\left(c_1u_1+c_2u_2+c_3u_3\right)\cdot\begin{bmatrix}2&1&1\\1&2&1\\1&1&2\end{bmatrix}\begin{Bmatrix}v_1^n\\v_2^n\\v_3^n\end{Bmatrix}\right.$$

$$\left.+\frac{1}{Re}\left(2\Delta\begin{bmatrix}b_1^2&b_1b_2&b_1b_3\\b_2b_1&b_2^2&b_2b_3\\b_3b_1&b_3b_2&b_3^2\end{bmatrix}\begin{Bmatrix}u_1^n\\u_2^n\\u_3^n\end{Bmatrix}+\Delta\begin{bmatrix}b_1c_1&b_1c_2&b_1c_3\\b_2c_1&b_2c_2&b_2c_3\\b_3c_1&b_3c_2&b_3c_3\end{bmatrix}\begin{Bmatrix}v_1^n\\v_2^n\\v_3^n\end{Bmatrix}\right)\right\}\qquad\text{（A.30）}$$

＜i＝2（y 方向）の場合＞

$$\overline{M}_{\alpha\beta}\tilde{v}_{\beta}=\overline{M}_{\alpha\beta}v_{\beta}^{n}-\Delta t\left\{K_{\alpha\beta\gamma1}u_{\beta}^{n}+K_{\alpha\beta\gamma2}v_{\beta}^{n}+\frac{1}{Re}\left(S_{\alpha2\beta1}u_{\beta}^{n}+S_{\alpha2\beta2}v_{\beta}^{n}\right)\right\}$$

$$\Downarrow$$

$$\frac{\Delta}{3}\begin{bmatrix}1 & & \\ & 1 & \\ & & 1\end{bmatrix}\begin{Bmatrix}\tilde{v}_1\\ \tilde{v}_2\\ \tilde{v}_3\end{Bmatrix}=\frac{\Delta}{3}\begin{bmatrix}1 & & \\ & 1 & \\ & & 1\end{bmatrix}\begin{Bmatrix}v_1^n\\ v_2^n\\ v_3^n\end{Bmatrix}$$

$$-\Delta t\left\{\frac{\Delta}{12}\left(b_1u_1+b_2u_2+b_3u_3\right)\cdot\begin{bmatrix}2&1&1\\1&2&1\\1&1&2\end{bmatrix}\begin{Bmatrix}u_1^n\\u_2^n\\u_3^n\end{Bmatrix}+\frac{\Delta}{12}\left(c_1u_1+c_2u_2+c_3u_3\right)\cdot\begin{bmatrix}2&1&1\\1&2&1\\1&1&2\end{bmatrix}\begin{Bmatrix}v_1^n\\v_2^n\\v_3^n\end{Bmatrix}\right.$$

$$\left.+\frac{1}{Re}\left(\Delta\begin{bmatrix}c_1b_1&c_1b_2&c_1b_3\\c_2b_1&c_2b_2&c_2b_3\\c_3b_1&c_3b_2&c_3b_3\end{bmatrix}\begin{Bmatrix}u_1^n\\u_2^n\\u_3^n\end{Bmatrix}+2\Delta\begin{bmatrix}c_1^2&c_1c_2&c_1c_3\\c_2c_1&c_2^2&c_2c_3\\c_3c_1&c_3c_2&c_3^2\end{bmatrix}\begin{Bmatrix}v_1^n\\v_2^n\\v_3^n\end{Bmatrix}\right)\right\} \quad \text{(A.31)}$$

・圧力ポアソン方程式（A.23）

$$\left(A_{\alpha1\beta1}+A_{\alpha2\beta2}\right)p_\beta^{n+1}=-\frac{1}{\Delta t}\left(H_{\alpha\beta1}\tilde{u}_\beta+H_{\alpha\beta2}\tilde{v}_\beta\right)$$

⇩

$$\Delta\begin{bmatrix}b_1^2+c_1^2&b_1b_2+c_1c_2&b_1b_3+c_1c_3\\b_2b_1+c_2c_1&b_2^2+c_2^2&b_2b_3+c_2c_3\\b_3b_1+c_3c_1&b_3b_2+c_3c_2&b_3^2+c_3^2\end{bmatrix}\begin{Bmatrix}p_1^{n+1}\\p_2^{n+1}\\p_3^{n+1}\end{Bmatrix}$$

$$=-\frac{1}{\Delta t}\frac{\Delta}{3}\left(\begin{bmatrix}b_1&b_2&b_3\\b_1&b_2&b_3\\b_1&b_2&b_3\end{bmatrix}\begin{Bmatrix}\tilde{u}_1\\\tilde{u}_2\\\tilde{u}_3\end{Bmatrix}+\begin{bmatrix}c_1&c_2&c_3\\c_1&c_2&c_3\\c_1&c_2&c_3\end{bmatrix}\begin{Bmatrix}\tilde{v}_1\\\tilde{v}_2\\\tilde{v}_3\end{Bmatrix}\right) \quad \text{(A.32)}$$

・流速を求める式（A.24）

＜i＝1（x方向）の場合＞

$$\overline{M}_{\alpha\beta}u_\beta^{n+1}=\overline{M}_{\alpha\beta}\tilde{u}_\beta-\Delta tH_{\alpha\beta1}p_\beta^{n+1}$$

⇩

$$\frac{\Delta}{3}\begin{bmatrix}1 & & \\ & 1 & \\ & & 1\end{bmatrix}\begin{Bmatrix}u_1^{n+1}\\ u_2^{n+1}\\ u_3^{n+1}\end{Bmatrix}=\frac{\Delta}{3}\begin{bmatrix}1 & & \\ & 1 & \\ & & 1\end{bmatrix}\begin{Bmatrix}\tilde{u}_1\\ \tilde{u}_2\\ \tilde{u}_3\end{Bmatrix}-\Delta t\frac{\Delta}{3}\begin{bmatrix}b_1&b_2&b_3\\b_1&b_2&b_3\\b_1&b_2&b_3\end{bmatrix}\begin{Bmatrix}p_1^{n+1}\\p_2^{n+1}\\p_3^{n+1}\end{Bmatrix} \quad \text{(A.33)}$$

＜i＝2（y方向）の場合＞

$$\overline{M}_{\alpha\beta} v_\beta^{n+1} = \overline{M}_{\alpha\beta} \tilde{v}_\beta - \Delta t H_{\alpha\beta 2} p_\beta^{n+1}$$

$$\frac{\Delta}{3}\begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}\begin{Bmatrix} v_1^{n+1} \\ v_2^{n+1} \\ v_3^{n+1} \end{Bmatrix} = \frac{\Delta}{3}\begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}\begin{Bmatrix} \tilde{v}_1 \\ \tilde{v}_2 \\ \tilde{v}_3 \end{Bmatrix} - \Delta t \frac{\Delta}{3}\begin{bmatrix} c_1 & c_2 & c_3 \\ c_1 & c_2 & c_3 \\ c_1 & c_2 & c_3 \end{bmatrix}\begin{Bmatrix} p_1^{n+1} \\ p_2^{n+1} \\ p_3^{n+1} \end{Bmatrix} \quad \text{（A.34）}$$

以上により、剛性方程式（A.22）～（A.24）の具体的内容が明らかになった

付録 B：有限要素法による流体解析並列プログラム

（1）自動並列

```
/*****************************************************************************
 * 中間流速の計算関数 step01()
 *
 * nx              : 総節点数
 * mx              : 総要素数
 * flow            : 流速 U, V (既知)
 * flow1           : 中間流速　(未知変数)
 * amb             : 質量行列の集中化行列
 * nodec           : 要素の適合条件
 * bb              : 補間関数の微係数(X)
 * cc              : 補間関数の微係数(Y)
 * area            : 要素の面積
 * area3, area12 : area/3, area/12
 * delt            : 微小時間増分量
 * reiv            : レイノズル数の逆数
 *****************************************************************************/

void step01(int nx, int mx, double delt, double reiv, double *result)
{
   int i,j,ie;
   double start, end;
   long double ss;

   for(i = 1; i <= nx; i++){
      flow2[i][1] = 0;
      flow2[i][2] = 0;
   }

#pragma omp parallel
   {
      int i, j, ii, i1, i2, i3;
      int ID = omp_get_thread_num();
      long double x1, x2, x3, y1, y2, y3;
```

```
      long double mmm1, mmm2, mmm3, nnn1, nnn2, nnn3;
      long double a;
      long double a01, a12, bb1, bb2, bb3, cc1, cc2, cc3;
      long double uu1, uu2, uu3, vv1, vv2, vv3;
      long double u123, v123, bu, bv, cu, cv;
      long double ukk1, ukk2, ukk3, vkk1, vkk2, vkk3;
      long double avis;
      long double cbuv, uss1, uss2, uss3, vss1, vss2, vss3;
      long double temp[16][MD1][2];

      if(ID == 0)
        start = seconds();
#pragma omp for
      for(ie = 1; ie <= mx; ie++){
        i1 = nodc[ie][1];
        i2 = nodc[ie][2];
        i3 = nodc[ie][3];
        x1 = (long double)element[i1][1];
        x2 = (long double)element[i2][1];
        x3 = (long double)element[i3][1];
        y1 = (long double)element[i1][2];
        y2 = (long double)element[i2][2];
        y3 = (long double)element[i3][2];

        a    = x1*(y2 - y3) + x2*(y3 - y1) + x3*(y1 - y2);

        a01 = a / 2;
        a12 = a01 / 12;
        a = 1.0 / a;
        bb1 = (y2 - y3)*a;
        bb2 = (y3 - y1)*a;
        bb3 = (y1 - y2)*a;
        cc1 = (x3 - x2)*a;
        cc2 = (x1 - x3)*a;
        cc3 = (x2 - x1)*a;
        uu1 = flow1[i1][1];
```

```
uu2 = flow1[i2][1];
uu3 = flow1[i3][1];
vv1 = flow1[i1][2];
vv2 = flow1[i2][2];
vv3 = flow1[i3][2];

/* 移流項の作成 */
u123 = uu1 + uu2 + uu3;
v123 = vv1 + vv2 + vv3;
bu = bb1*uu1 + bb2*uu2 + bb3*uu3;
bv = bb1*vv1 + bb2*vv2 + bb3*vv3;
cu = cc1*uu1 + cc2*uu2 + cc3*uu3;
cv = cc1*vv1 + cc2*vv2 + cc3*vv3;
mmm1 = u123 + uu1;
mmm2 = u123 + uu2;
mmm3 = u123 + uu3;
nnn1 = v123 + vv1;
nnn2 = v123 + vv2;
nnn3 = v123 + vv3;
ukk1 = (bu*(mmm1) + cu*(nnn1)) * a12;
ukk2 = (bu*(mmm2) + cu*(nnn2)) * a12;
ukk3 = (bu*(mmm3) + cu*(nnn3)) * a12;
vkk1 = (bv*(mmm1) + cv*(nnn1)) * a12;
vkk2 = (bv*(mmm2) + cv*(nnn2)) * a12;
vkk3 = (bv*(mmm3) + cv*(nnn3)) * a12;

/* 粘性項の作成 */
avis = a01 * reiv;
cbuv = cu + bv;
mmm1 = 2 * bb1 * bu + cc1 * cbuv;
mmm2 = 2 * bb2 * bu + cc2 * cbuv;
mmm3 = 2 * bb3 * bu + cc3 * cbuv;
nnn1 = 2 * cc1 * cv + bb1 * cbuv;
nnn2 = 2 * cc2 * cv + bb2 * cbuv;
nnn3 = 2 * cc3 * cv + bb3 * cbuv;
uss1 = mmm1 * avis;
```

```
            uss2 = mmm2 * avis;
            uss3 = mmm3 * avis;
            vss1 = nnn1 * avis;
            vss2 = nnn2 * avis;
            vss3 = nnn3 * avis;

            mmm1 = ukk1 + uss1;
            mmm2 = ukk2 + uss2;
            mmm3 = ukk3 + uss3;
            nnn1 = vkk1 + vss1;
            nnn2 = vkk2 + vss2;
            nnn3 = vkk3 + vss3;

            /* 既知項の重ね合わせ */
            temp[ID][i1][1] += mmm1;
            temp[ID][i2][1] += mmm2;
            temp[ID][i3][1] += mmm3;
            temp[ID][i1][2] += nnn1;
            temp[ID][i2][2] += nnn2;
            temp[ID][i3][2] += nnn3;
        }
        if(ID == 0)
            end = seconds();

#pragma omp for
        for(ie = 1; ie <= nx; ie++){
            flow2[ie][1] += temp[ID][ie][1];
            flow2[ie][2] += temp[ID][ie][2];
        }
    }

    *result += (end - start);
    /* 中間流速の計算 */
    for(i = 1; i <= nx; i++){
        ss = amb[i] * (long double)delt;
        flow2[i][1] = flow1[i][1] - flow2[i][1] * ss;
```

```
        flow2[i][2] = flow1[i][2] - flow2[i][2] * ss;
    }
}
```