修 士 論 文

オブジェクト指向に基づく
ナビエ・ストークス方程式の数値シミュレーション

指導教官　松澤 照男 教授

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

上田 隆宏

1997 年 2 月 14 日

目 次

第 1 章

はじめに

1.1 数値流体力学の現状

数値流体力学はコンピュータの発達とともに歩んで来た学問形態であり [1]、流れのシミュレーションを行なう場合に、差分法や有限要素法、境界要素法や各種乱流モデル等、様々な手法が開発されている。更にこれらの方法の計算効率向上を狙い種々の並列化技法の研究がなされている。しかし、これらの方法により、複雑形状や熱を伴う流れ等の複雑な流れの解析を行なうめには、プログラミングに多大な労力を要する。また既存のプログラムを少し変更して他の問題を解こうとする場合にも、同じ問題が伴う。加えてプログラムのバグ発見は実行時にわかる場合が多くデバッグが大変になることが多い。更に、プログラムがモデルの概念を無視している為、その信頼性についても疑問が残る。このような状況において、山のように存在する問題に対して簡単に解を求めるのは困難であり、得られた解がモデルの概念を正確に表現しているかを検証するのは困難である。

近年オブジェクト指向をシミュレーションに用いるということが行なわれてきているが、まだまだこれらの問題を解決するのに充分な成果はあげられていないように見受けられる。

1.2 本研究の位置づけ

現実の世界に近いシミュレーションを行なうために畠山ら [2] は、オブジェクトベースに基づくシミュレーションで分子モデルを用い希薄流れをシュミレーションする実行支援

環境を構築した。しかし、本研究の対象とする流体はニュートン流体であり分子モデルとは本質的に異なっている。畠山らは格子点をオブジェクトにした差分解法についても行ないオブジェクト指向解法による簡単な実行例を示した。[3]。これによりオブジェクト指向パラダイムを差分法にとりいれる理論的方法は確立された。しかし、オブジェクト指向を取り入れたことで、従来法に対して改善された点については示されていない。その上、複雑形状や熱を伴う流れのような複雑な流れに対してどの様な方針で対処して行くのかについて全く触れられていない。

そこで、本研究では、彼らの流れに従い、複雑な流れ、つまり熱移動を伴う流れや複雑形状流れ、移動境界の流れ、あるいはそれらの複合問題に対して柔軟かつ信頼性の高いシミュレーションを行なうことを目的とし、そのためオブジェクト指向の概念を数値シミュレーションに用いることにする。そのための第１段階として畠山らと同様に格子点をオブジェクトにとる。更に本研究ではMAC法による離散化式を用いて、物理量という関係で格子点クラスからサブクラスの導出を行ないモデル化を行なう。このモデルにより熱を加える問題への柔軟な拡張を試みる。更に格子点オブジェクトと集約関係にある領域をオブジェクトとするモデル化を行ない、領域オブジェクト間でメッセージ通信を行なわせることで、複雑形状流れへの適用及び解法の高速化のための並列化について検討する。そして提案したモデルが柔軟かつ信頼のおけるものであることを示すとともに、シミュレーション例とともに得られた成果を報告する。

第 2 章

計算工学におけるオブジェクト指向

2.1　オブジェクト指向のパラダイム

ソフトウエアはそれを構成する機能的要素をモジュール化し、他のモジュールとのインターフェースを単純化することによって大きなコードを比較的容易に設計、制作、保守できる。オブジェクト指向により、ソフトウエアの複雑性は「分割統治」という概念で軽減される。これはソフトウエア設計の基本となるからである。

次に重要な点は対象のモデル化である。人間の脳が行なう認知活動の最も重要な特徴の一つは、事物や実体が個々に異なっていても、ある種の概念的カテゴリーとして取り扱われるということである。従来のプログラミングはプログラムの流れに沿って分割されており、脳における概念またはモデル化という観点は必要ではなかった。しかしながら、オブジェクト指向では、対象をプログラムの観点からではなく、全体のシステムの中でどのように捉えるかが重要となっている。

人間が道具を用いて何かを行なう場合、人間が完全な知識を持っていなくても適切な行動が取れることが多い。これは道具が知識を持っていると考えられる。例えば、電話線にどのように声が流れているかを知らなくても、電話機はそれを知っていると考えることができる。こうしてすべての道具は知識を有していると考えられ、すべてのオブジェクトには知識が埋め込まれているといえる。この考えは、オブジェクト指向において、オブジェクトはデータとメソッドを持っているという考えにつながる。

オブジェクトは他からメッセージを受けた時だけ埋め込まれた知識を駆動させ、変化を起こす。メッセージはそれを受信したオブジェクトが理解する。このため、間違ったメッ

セージを送れば受信側で理解できずエラーとなる。この点は、サブルーチンを誤ってコールしても誤った処理が進んでしまう、従来のプログラミング言語と異なり、デバッグが容易になるだけでなく、コードの信頼性向上につながる。

オブジェクトのヒューマン・フレンドりネスは他の言語に比べてかなり高いといえる。それはオブジェクトに知識が埋め込まれていることと、その知識を起動するきっかけとなるメッセージがそのオブジェクトに固有のものとなり、複数のメッセージを同一のオブジェクトに与えたり、逆に同じメッセージを異なったオブジェクトに送信することにより異なった知識を駆動できる。これによって自然言語の持つ柔軟性をそのまま反映できることになる。

2.2 オブジェクト指向の基本概念

オブジェクト指向の基本概念 [6] は主に次のようにわけられる。

1. 抽象化
2. カプセル化 (情報隠蔽)
3. モジュール化
4. 階層構造

以下にこれらについて述べる。

現実世界の複雑性をいかにモデル化して制御可能なオブジェクトに展開できるかがオブジェクト指向によるモデル化の重要な点になる。これは抽象化と呼ばれ、対象の特徴を調べ、共通の特徴をまとめて一般化し、不必要な細部を捨てる操作を行なう。これによって本質的な議論が容易になる。モデル化したオブジェクトをクラスと呼び、そのクラスに属する具体的なオブジェクトをインスタンスあるいは、単にオブジェクトと呼ぶ。

オブジェクト指向においては何をオブジェクトとしその属性を何にするかを決定するのは容易でない。これは抽象化された概念の詳細化の度合が視点や専門的知識によって異なるからである。この場合、正しいモデル化を判断する厳密な基準はない。関係する専門家が納得し、想定している仕事に役立つ最小限の単純さを持てばいいと言える。

すでに述べたように分割統治によりオブジェクトのインターフェースは単純となる。そして内部の複雑性は外部からは見えないようにする。このような性質はカプセル化、ある

いは情報隠蔽と呼ばれている。情報隠蔽では、公開部分を変更しない限り、非公開部分を変更しても他に影響を与えないという長所がある。

情報隠蔽された抽象データ型を使用する側では型名の参照と操作の呼び出ししか許されない。このためデータ構造そのものを他からアクセスされることはなく、オブジェクトが誤って使用される危険性を軽減できシステムの信頼性向上につながる。

一般に、あるオブジェクトは別の複数のオブジェクトから構成される。このようにオブジェクトが複数のオブジェクトに分解されることをモジュール化と呼ぶ。例えば、自動車は車体、エンジン、変速機、タイヤ等から構成され、こうした関係がオブジェクトの集合関係を形成する。

モジュール化の利点は計算機コードの部品化である。再利用可能な部分を多く所有すれば、新規コードの作成は単に多くの部品を組み合わせるだけの仕事になる。再利用性は、上述の抽象化と情報隠蔽に大いに依存し、モジュールの汎用性と特殊性が決まる。

分割して統治せよという原則は再帰的、つまり分割された要素がさらに分割されることも意味的に含んでいる。こうしてオブジェクトは再帰的に分割され、これが階層構造を形成する。階層構造は抽象化の度合の異なるレベルが多層となることを意味している。工学におけるオブジェクトの階層構造は、抽象化の考え方が人によって、分野によって異なるため、目的によって異ならざるを得ないと考えられている。

階層構造において重要な概念は継承である。これは上位クラスの性質を下位クラスが受け継ぐことを意味している。上位クラスをスーパークラス、下位クラスをサブクラスと呼ぶ。この性質からサブクラスではスーパークラスと異なる部分だけを記述すればよく、プログラミングが容易になる。

2.3 オブジェクト指向差分解法の概念

差分方程式は空間でΔx 時間でΔt だけ離れた位置における物理量 u の代数的関係式で表現されている。

これらの差分関係をオブジェクト指向的な観点からみれば、ノードである格子点をオブジェクトに設定し、アークである格子点間の関係を相互作用とみると、差分関係式は各該当オブジェクトの相互作用のメカニズムを表現している「相互作用方程式」であるとみなせる。そして、格子点の数だけインスタンスを生成すると、このオブジェクトの集団は、相互作用をしながら未知決定量 u が時間的に変化していくオブジェクトの集団とみなす

ことが可能となる [3]。ただしノードである格子点をオブジェクトと設定したが、モデルとしてのオブジェクトは格子点を中心としΔx の微小幅を持った微小体積をオブジェクトと設定したことになる。従って、相互作用方程式は微小体積間の力学的関係をそのまま表現している。

このようにオブジェクトとそれらの相互作用の関係を設定することでオブジェクト指向に特有の駆動機構、つまりメッセージ通信に基づく情報交換と自己情報処理を利用して以下のような解法が設定される。

1 メッシュ点を一般的な意味でのオブジェクトと設定

2 持っている属性は、その点における空間座標、物理量 u

3 メソッドにはその点を中心とした差分方程式

4 関係オブジェクトへ相互作用リンクの生成

5 各オブジェクトは自己のメソッドにより周囲のオブジェクトから自己の u の値を更新

6 これを全てのオブジェクトについて実行

このように対象としているすべての領域に対し代数的関係を導出し連立方程式系が解けることになる。

見方を変えれば、連立方程式の各方程式が表す元になっている実体をオブジェクトに設定し、各方程式を相互作用方程式と考え、方程式内に現れる変数名を相互作用リンクを張るべき相手と見ることができる。

第 3 章

基礎方程式

3.1 熱伝導方程式

オブジェクト指向に基づいて、偏微分方程式をシミュレーションする簡単な例として、まずはじめに定常熱伝導問題でよく用いられる次のポアソン方程式 [8] を考える。

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = Q(x, y) \tag{3.1}$$

熱伝導問題の場合、T は温度、Q はデカルト座標 (x、y) での発熱量となる。時間項は含まれないので必ず定常解が得られる。

3.1.1 差分法による離散化

ポアソン方程式を中心差分で離散化すると以下のようになる。

$$\frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{\Delta x^2} + \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{\Delta y^2} = -Q(i, j) \tag{3.2}$$

$\Delta x = \Delta y$とし、$T_{i,j}$で表すと、

$$T_{i,j} = \frac{1}{4}(T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1} + Q(i, j)) \tag{3.3}$$

となる。上式により位置 i、j での温度が各々求まり対象領域全体で収束させることにより温度分布が求められる。

3.2　２次元粘性非圧縮性流れ

本研究では、応力と歪み速度が線形関係にあるニュートン流体において、粘性非圧縮性流体を仮定する。以上の仮定より、一般的な速度と圧力を流れの未知量とする方程式系 [1] は、デカルト座標系 (x-y 座標系) において以下のように表される。

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{3.4}$$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{\partial p}{\partial x} + \frac{1}{Re}(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}) \tag{3.5}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{\partial p}{\partial y} + \frac{1}{Re}(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}) \tag{3.6}$$

ここで u、v はそれぞれ x、y 方向の速度成分 p は圧力、Re はレイノルズ数である。Re 数は無次元量であり慣性力／粘性力であたえられ、慣性力の大きい流れでは Re が大きくなり、逆に粘性の強いストークス流れのような場合には Re 数は小さくなる。

3.2.1　MAC 法による離散化

速度と圧力表示の N-S 方程式に対する差分法として最も広く用いられる古典的な方法は、Harlow と Welch によって提案された MAC 法 [1](Harlow-Welch、1965;Welch ら、1966) である。この方法は当初、自由表面を含む非定常流れを取り扱うために開発された。自由表面の位置を時間の関数として決定するために流れ場中に質量の無いマーカー粒子が導入された。マーカー粒子は速度場に従って移動するが、流れ場には何の影響も及ぼさないものとされた。これが MAC 法と呼ばれる所以である。計算に際してはまず対象となる領域を間隔Δx、Δyの格子 (セル) に分割する。この方法の特徴は、速度成分と圧力が図 3.1 に示されるように異なる点で定義される、いわゆる食い違い格子 (Staggerd grid) を用いることである。ここで、i,j は各々x、y 方向のセル番号とする。

基礎方程式は FTCS 差分により以下のように離散化される。

$$\frac{u_{i+1/2,j}^{n+1} - u_{i-1/2,j}^{n+1}}{\Delta x} + \frac{v_{i,j+1/2}^{n+1} - v_{i,j-1/2}^{n+1}}{\Delta y} = 0 \tag{3.7}$$

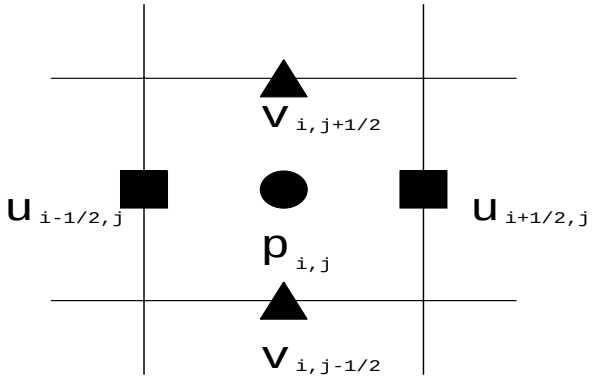

図 3.1: Staggerd Grid

$$u_{i+1/2,j}^{n+1} = F_{i+1/2,j}^{n} - \frac{\Delta t}{\Delta x}(p_{i+1,j}^{n} - p_{i,j}^{n}) \tag{3.8}$$

$$v_{i,j+1/2}^{n+1} = G_{i,j+1/2}^{n} - \frac{\Delta t}{\Delta y}(p_{i,j+1}^{n} - p_{i,j}^{n}) \tag{3.9}$$

ここで、

$$\begin{aligned} F_{i+1/2,j}^{n} &= u_{i+1/2,j}^{n} + \Delta t(-\frac{u_{i+1/2,j}^{2} - u_{i,j}^{2}}{\Delta x} \\ &\quad -\frac{(uv)_{i+1/2,j+1/2} - (uv)_{i+1/2,j-1/2}}{\Delta y} \\ &\quad +\frac{u_{i+3/2,j} - 2u_{i+1/2,j} + u_{i-1/2,j}}{Re\Delta x^2} \\ &\quad +\frac{u_{i+1/2,j+1} - 2u_{i+1/2,j} + u_{i+1/2,j-1}}{Re\Delta y^2})^n \end{aligned} \tag{3.10}$$

$$\begin{aligned} G_{i,j+1/2}^{n} &= v_{i,j+1/2}^{n} + \Delta t(-\frac{v_{i,j+1}^{2} - v_{i,j}^{2}}{\Delta y} \\ &\quad -\frac{(uv)_{i+1/2,j+1/2} - (uv)_{i-1/2,j+1/2}}{\Delta x} \end{aligned}$$

$$+\frac{v_{i,j+3/2}-2v_{i,j+1/2}+v_{i,j-1/2}}{Re\Delta y^2}$$

$$+\frac{v_{i+1,j+1/2}-2v_{i,j+1/2}+v_{i-1,j+1/2}}{Re\Delta x^2})^n \tag{3.11}$$

である。なお、連続の式はセル中心での中心差分、ナビエ・ストークス方程式は空間微分を中心差分で、時間微分を前進差分で離散化している。定義点以外の位置における速度が必要になる場合は周囲の定義点上の速度の平均により求める。例えば、位置 i+1/2,j+1/2 での v は、

$$v_{i+1/2,j+1/2}=\frac{v_{i,j+1/2}+v_{i+1,j+1/2}}{2} \tag{3.12}$$

で与えられる。式 (3.8), (3.9) を連続の式 (3.7) に代入すると圧力に対するポアソン方程式を得る。

$$\frac{p_{i+1,j}-2p_{i,j}+p_{i-1,j}}{\Delta x^2}+\frac{p_{i,j+1}-2p_{i,j}+p_{i,j-1}}{\Delta y^2}$$

$$=\frac{1}{\Delta t}(\frac{F^n_{i+1/2,j}-F^n_{i-1/2,j}}{\Delta x}+\frac{G^n_{i,j+1/2}-G^n_{i,j-1/2}}{\Delta y}) \tag{3.13}$$

(3.13) 式を $p_{i,j}$について陽的に表すと、

$$\begin{aligned} p_{i,j} &= \frac{1}{2(\Delta x^2+\Delta y^2)}\{\Delta y^2(p_{i+1,j}+p_{i-1,j}) \\ &+\Delta x^2(p_{i,j+1}+p_{i,j-1})-\frac{\Delta x^2\Delta y^2}{\Delta t}(\frac{F^n_{i+1/2,j}-F^n_{i-1/2,j}}{\Delta x} \\ &+\frac{G^n_{i,j+1/2}-G^n_{i,j-1/2}}{\Delta y})\} \end{aligned} \tag{3.14}$$

となる。

MAC 法の計算過程は、新しい時刻 n+1 での速度を時刻 n での値から式 (3.8),(3.9) より求め、次に圧力を式 (3.13) について陽的に解いた式 (3.14) を反復計算より収束値として時刻 n+1 での各セルの圧力を得る。収束した圧力場は時刻 n+1 において連続の式を満足する。

境界条件の取り扱い

今回は固体表面についての境界条件を考えるので、自由表面の境界条件については省略する。固体表面での境界条件は固着 (no-slip) にするか、自由すべり (free-slip) にするか、どちらの設定も可能である。MAC 法では計算領域の境界に隣接する外側に仮想セル (fictitious cell) を設置する [9]。壁面上では、壁面に対して垂直方向の速度成分はゼロである。しかし仮想壁面上での速度成分はゼロにはならず、壁に対して垂直速度成分がゼロであることから、壁内部と仮想壁でそれぞれ連続の式を満たすように求められる。

i) 壁面が自由すべりの場合

垂直壁に対して $u_0 = 0$、$u_1^{'} = -u_1$、$v_1^{'} = v_1$

水平壁に対して $v_0 = 0$、$u_1^{'} = u_1$、$v_1^{'} = -v_1$

ii) 壁面が固着の場合

垂直壁に対して $u_0 = 0$、$u_1^{'} = u_1$、$v_1^{'} = -v_1$

水平壁に対して $v_0 = 0$、$u_1^{'} = -u_1$、$v_1^{'} = v_1$

ここで、添字 0 は壁面上、添字 1 は壁面に隣接するセルにおける量、$^{'}$は仮想壁内を表している。

境界条件は圧力の決定についても必要であり、壁内部のセルで連続の式を満たすように運動方程式から求められる。

i) 壁面が自由すべりの場合

垂直壁に対して $p^{'} = p_1 \pm \Delta x g_x$(右側境界)

水平壁に関して $p^{'} = p_1 \pm \Delta y g_y$(上側境界)

ii) 壁面が固着の場合

垂直壁に対して $p^{'} = p_1 \pm \Delta x g_x \pm 2u_1/(Re\Delta x)$

水平壁に対して $p^{'} = p_1 \pm \Delta y g_y \pm 2v_1/(Re\Delta y)$

ここで、垂直壁の場合には、i(実セル)<i+1(仮想セル) のときに正の符号を取り、水平壁の場合は、j(実セル)<j+1(仮想セル) のときに正の符号をとる。

3.3 熱移動を伴う２次元粘性非圧縮性流れ

熱移動を伴う問題へ拡張する場合、連続の式、運動量保存の式に加えて新たにエネルギー保存式が必要になる。これらの問題を取り扱う場合に次の仮定を置く。熱を加えない

場合と同様に非圧縮性ニュートン流体を対象とするが浮力の原因となる密度変化は考慮する。つまり密度一定の非圧縮性流れとし、温度の変化による密度差の効果は浮力項にのみ働くとするブジネ近似 [7] を用いて解析する。物性値は一定とする。熱エネルギ保存のみを考え、運動エネルギから熱エネルギへの不可逆的な散逸を無視する。以上の仮定よりエネルギ方程式は以下のように表される。

$$\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} + v\frac{\partial T}{\partial y} = \frac{\lambda}{\rho C_p}(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}) + \frac{Q}{\rho C_p} \tag{3.15}$$

また、y 方向に浮力項を追加した運動方程式は次のようになる。

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{\partial p}{\partial y} + \frac{1}{Re}(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}) - g\beta(T - T^*) \tag{3.16}$$

ここで、T^*は断面における平均温度であり、βは体膨張係数と呼ばれ$-(\partial ln\rho/\partial T)$ で定義される。 今回は高温壁温度と低温壁温度の平均温度の逆数 [10] つまり次式を用いた。

$$\beta = \frac{2}{T_{high} + T_{low}} \tag{3.17}$$

熱に関する問題では、無次元量として Re 数の他に Gr 数等が用いられる。Gr 数は次式で定義される。

$$Gr = \frac{g\beta\Delta T L^3}{\nu^2} \tag{3.18}$$

Gr 数は浮力と粘性力の大きさの比を表している。

3.3.1 離散化

エネルギ方程式は以下のように通常の FTCS 差分を用いて離散化した。

$$\begin{aligned} T_{i,j}^{n+1} &= T_{i,j}^n + \Delta T(-\frac{(Tu)_{i+1,j} - (Tu)_{i-1,j}}{2\Delta x} - \frac{(Tv)_{i.j+1} - (Tv)_{i,j-1}}{2\Delta y} \\ &+ \frac{\lambda}{\rho C_p}(\frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{\Delta x^2} + \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{\Delta y^2}) \\ &+ \frac{Q}{\rho C_p})^n \end{aligned} \tag{3.19}$$

格子点の定義位置はセルの中心とし圧力の定義点と同じ位置に設定した。

第 4 章

モデリング方法

流れのシミュレーションにオブジェクト指向を適用しようとする場合、まず、対象世界 (流れ) を頭脳内で認識 (認識モデリング段階) しなければならない、次にそれをなんらかの形で表現し (概念モデリング段階)、それらが充分に写像されたなら重要な本質的な部分を抜きだして、より詳細にかつ論理的に理論や式を用いて表現される段階に入る (論理モデリング段階)。それから対象世界の静的な構造部分と動的な機構部分とに分けて、再構成を行なう段階 (再構成モデリング段階) に入り、最後に実装される (再現モデリング段階)[4]。概念モデルから始めるのがオブジェクト指向に一貫したモデリング過程論ということになるが、流れの計算を行なう場合に、概念モデルから始めてモデリングを構築することは容易なことではない。偏微分方程式というのは元来、流れというものを認識し、それを概念化した後に論理的な式にモデル化したものであるので、オブジェクト指向一貫モデリングの論理モデリングの段階まではすでになされていると考えられる。

そこで本研究では、数値流体の分野で現在まで蓄積されてきた流れを定式化した偏微分方程式のシミュレーションを考える。そのため、まず格子点 (微小体積) をオブジェクトに取り、差分法により離散化された式を格子点間の関係に対応させることからモデリングを行なう。更に格子点オブジェクトを含む領域を新たなオブジェクトと考え、領域オブジェクト間の関係をオブジェクトを引数に持つオブジェクトの関係そのものを表すメソッドとしてモデリングを行なった。

4.1 格子点オブジェクトの設定

本研究では連続体を対象とし、連続体というのは各位置の持つ物理量が連続的に連なってある状態を決定していると考えられるが、計算機を使って流れをシミュレートする場合には、何らかの方法で近似しなければならない。一方、オブジェクト指向では何をオブジェクトとするかが重要であり、明確な境界と意味を持つ何ものかとして定義する [5]。そこで、本研究では対象領域を微小体積を持つものに分割し、格子点オブジェクトと呼ぶことにする。格子点オブジェクトは、その属性として自己の位置、物理量を持ち、周囲の他のオブジェクトの状態より、自己の持つ属性値を変更する。オブジェクトはインターフェースを通してのみ自己の状態を変化させることができるので、周囲のオブジェクトの持つ物理量を引数とするインターフェース関数が必要になる。インターフェース関数としては、周囲のオブジェクトと自己のオブジェクトの関係を正確に表現したものであれば、どの様な関数を用いてもよい。しかし、差分法を離散化した式以外を考え出すのは困難であり、離散化式をインターフェース関数に用いることにする。以上よりオブジェクト及びそれらの関係を表すインターフェース関数が設定されたので、各オブジェクトインスタンスごとに周囲のインスタンスより自己の状態を変化させるということを、各々全てのインスタンスごとに行なうことでオブジェクト指向による解法が駆動される。図 4.1 に格子点オブジェクトの設定概念図を示す。従来法との実装段階での大きな違いは、オブジェクトクラスを設定することにより、そのクラスが有する属性とインターフェース関数 (メソッド) を一体にまとめること、つまりデータ構造と手続きが同じオブジェクトへ格納されるということである。しかし、各格子点オブジェクトインスタンスごとに周囲のオブジェクトから自己の状態を変化させるという操作を行ない、対象領域全体で収束させる作業には、何らかの反復が必要になり、本研究では収束を満たす解法に SOR 法を用いる。

4.1.1 ポアソン方程式への適用

ここでは、ポアソン方程式の２次元平板定常熱伝導問題についてオブジェクト指向を用いた解法について説明する。この問題の解法に格子点オブジェクトクラスを設定し、更に格子点クラスをスーパークラスとして、境界上の格子点オブジェクトクラスと境界を含まない格子点クラスに汎化させた。スーパークラスでは属性に物理量、この場合には温度 T と、位置 (x,y) を持っており、これらの性質はサブクラスに継承される。境界上のクラ

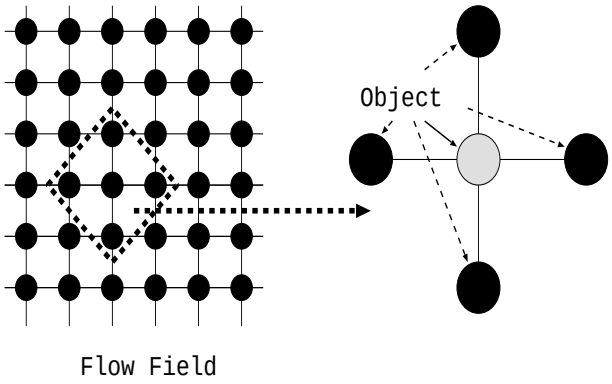

図 4.1: オブジェクト設定の概念

スでは、メソッドに Dirichlet 境界条件、Neuman 境界条件を満たす離散化式が格納され、境界内クラスでは、周囲のオブジェクトと自己のオブジェクトとの関係を表す差分法で離散化された式がメソッドに格納される。オブジェクト図を図 4.2 に示す。

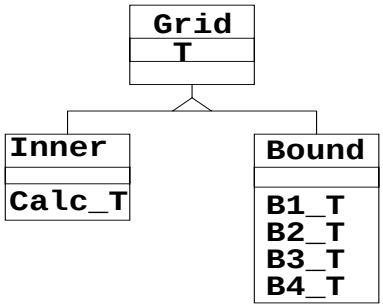

図 4.2: 熱伝導問題におけるオブジェクト図

4.1.2 MAC 法を用いたオブジェクト指向モデリング

格子点をオブジェクトに取る場合、各格子点は各々の物理量を属性として持つ。ナビエ・ストークス方程式は、圧力及び粘性による力が微小体積の持つ加速度による力に等しいという運動方程式の関係を定式化したものである。一方 MAC 法は図 3.1 にすでに示したようにスタッガード格子を用い、圧力差により速度が得られるというナビエ・ストークス方程式の物理的な意味を比較的うまく表現している。そこで、オブジェクト指向による

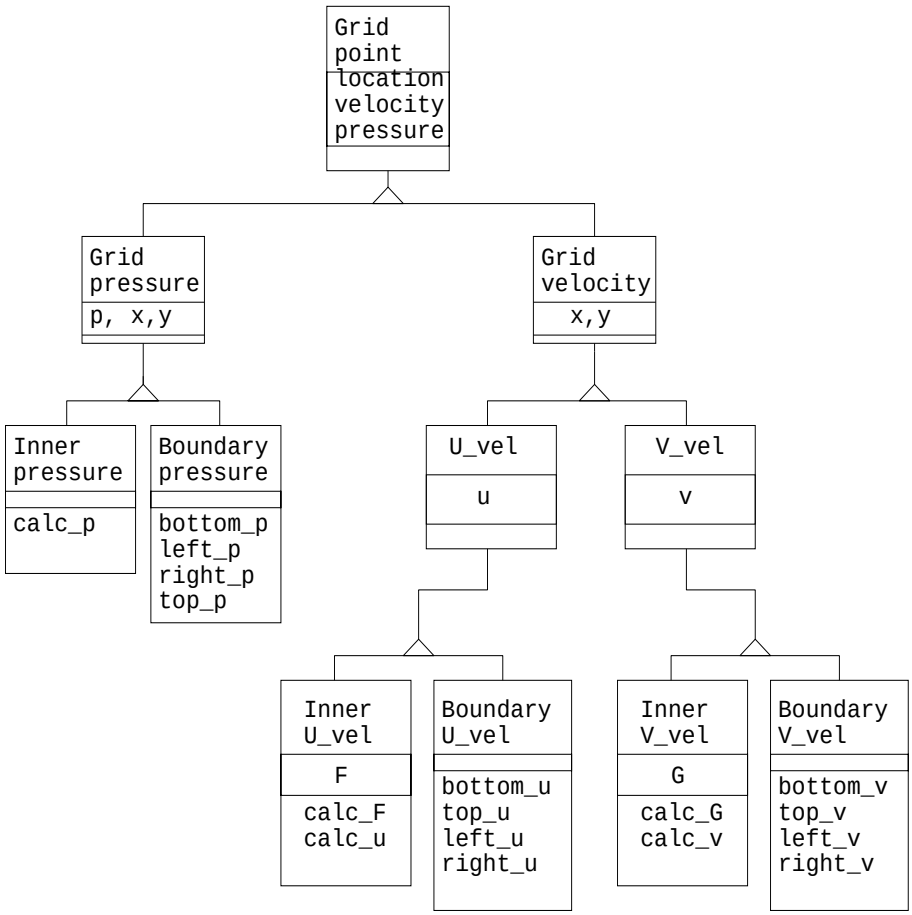

図 4.3: MAC 法におけるオブジェクト図

MAC 差分解法を行なう為に、格子点オブジェクトクラスから更に、物理量という関係で速度を定義する格子点クラスと圧力を定義する格子点クラスを派生させる [11]。スーパークラスは属性として位置、速度、圧力を持っているが、サブクラスである圧力格子点オブジェクトクラスは計算で扱う圧力 P を属性として持っているとし、属性の重複定義を行なっている。速度はベクトル量であり、x 方向の速度、y 方向の速度があり、速度格子点オブジェクトをスーパークラスとして、x 方向の速度定義点クラスと y 方向の速度定義点クラスが派生される。ここでも計算で扱う物理量 u,v という意味で属性の重複定義が行なわれている。また境界内部の x、y 方向オブジェクトはそれぞれ計算で用いる F,G の式をインターフェース関数として持ち、F,G の計算値もそれぞれ属性として持っているものとする。これは速度 u,v を求める為の途中段階での物理量の値を保持するということを意味しており、速度の物理量を属性に持つことと相違はない。 更に圧力格子点オブジェクト、

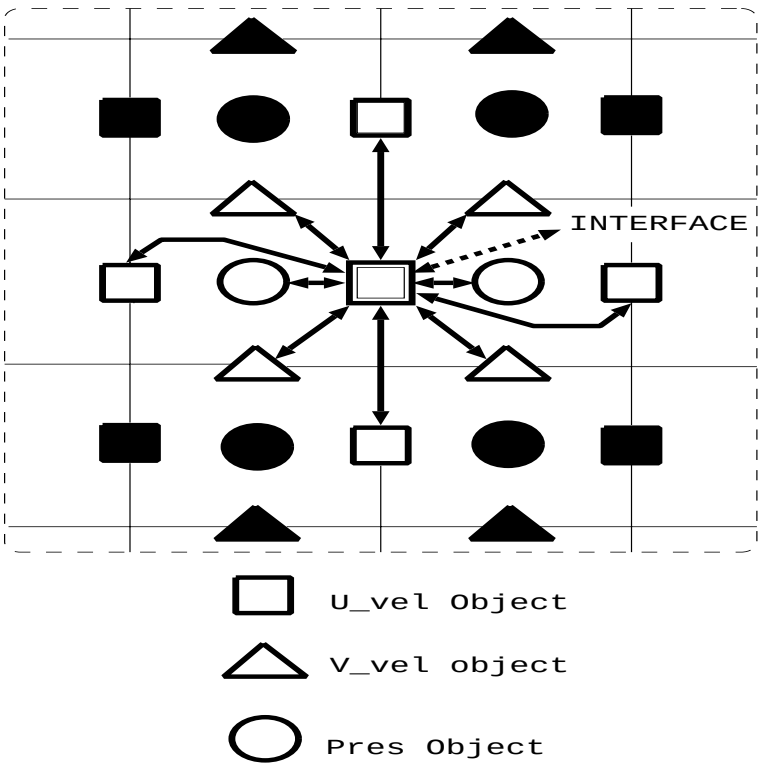

図 4.4: x 方向速度オブジェクトのメッセージ通信

速度格子点オブジェクトのサブクラスは各々境界上のオブジェクトクラスと境界を含まないオブジェクトクラスに分けられる。そして分けられたそれぞれのクラスでは境界上のクラスであれば境界上での式や、境界内であれば周囲との関係式が格納される。MAC 法を離散化式として用いる場合のオブジェクト図を図 4.3 に示す。

図 4.4 は対象領域内部の x 方向の速度格子点オブジェクトインスタンスが自己のインターフェースを通して周囲の x、y 方向速度格子点オブジェクトインスタンス及び圧力格子点オブジェクトインスタンスとメッセージ通信により自己の状態を変化させる概念図である。図の２重の四角で表されたインスタンスが、周囲のどのオブジェクトと関連するかを示している。

同様に y 方向速度格子点オブジェクトのメッセージ通信の概念を図 4.5 に圧力に関する概念を図 4.6 に示す。速度に関するインスタンスはメッセージ通信の引数に物理量 u,v,p を必要とするが、圧力に関するインスタンスは u,v を直接必要とせず、F,G を必要とする。

境界でのオブジェクトにおけるインターフェース関数は境界条件により与えられる離散

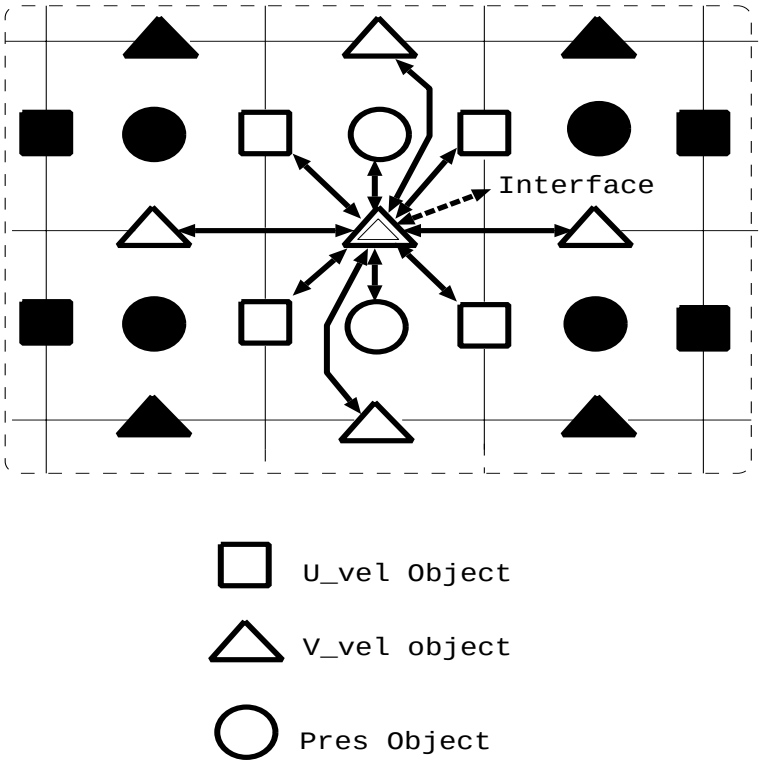

図 4.5: y 方向速度オブジェクトのメッセージ通信

化式や、境界でのオブジェクトに直接値を代入する関数に相当する。例えば、x=0 に壁面がある場合には、$p_{i-1,j} = p_{i,j} - 2u_{i+1/2,j}/(Re\Delta x)$ がメソッドとして用いられる。$p_{i-1,j}$ は仮想セル内で定義されるが、属性に圧力を持つ必要のある格子点オブジェクトであり、これらのオブジェクトは圧力の境界上オブジェクトクラスのインスタンスとしてまとめられる。速度オブジェクトについては壁面で速度がゼロになるように直接値を代入する関数と、仮想壁面で仮定される速度から導かれる関数をメソッドに格納している。

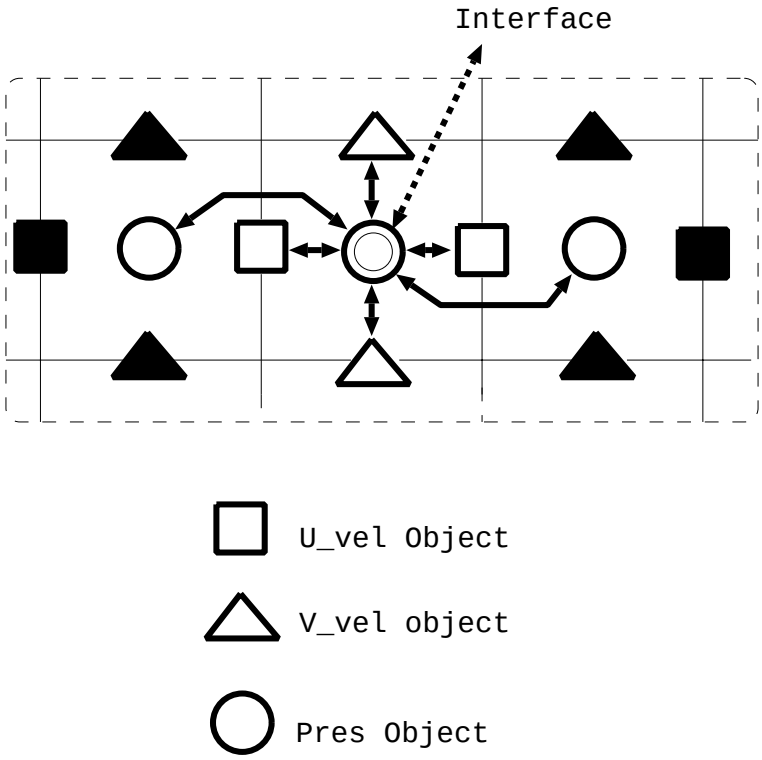

図 4.6: 圧力オブジェクトのメッセージ通信

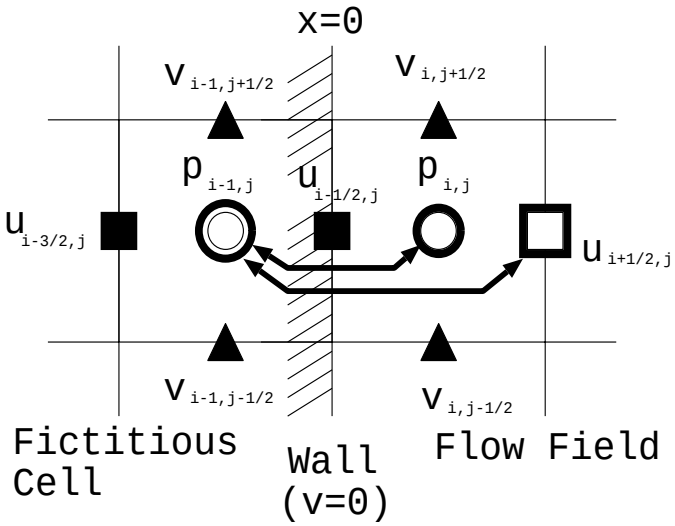

図 4.7: 境界でのメッセージ通信

4.1.3 熱問題への拡張

ここでは、オブジェクト指向による解法が柔軟であることを示す例として、オブジェクト指向による MAC 解法を熱と流れの連成問題へ拡張することを考える。熱を含んだ問題を取り扱う場合に、新たな物理量として温度が必要となる。このため新たに格子点オブジェクトに属性として温度を持たせる必要がある。格子点オブジェクトというスーパークラスから物理量という関係で、速度定義オブジェクトクラス、圧力定義オブジェクトクラスをすでに導出させてきたが、温度定義オブジェクトクラスを格子点クラスのサブクラスに新たに加えることにする。

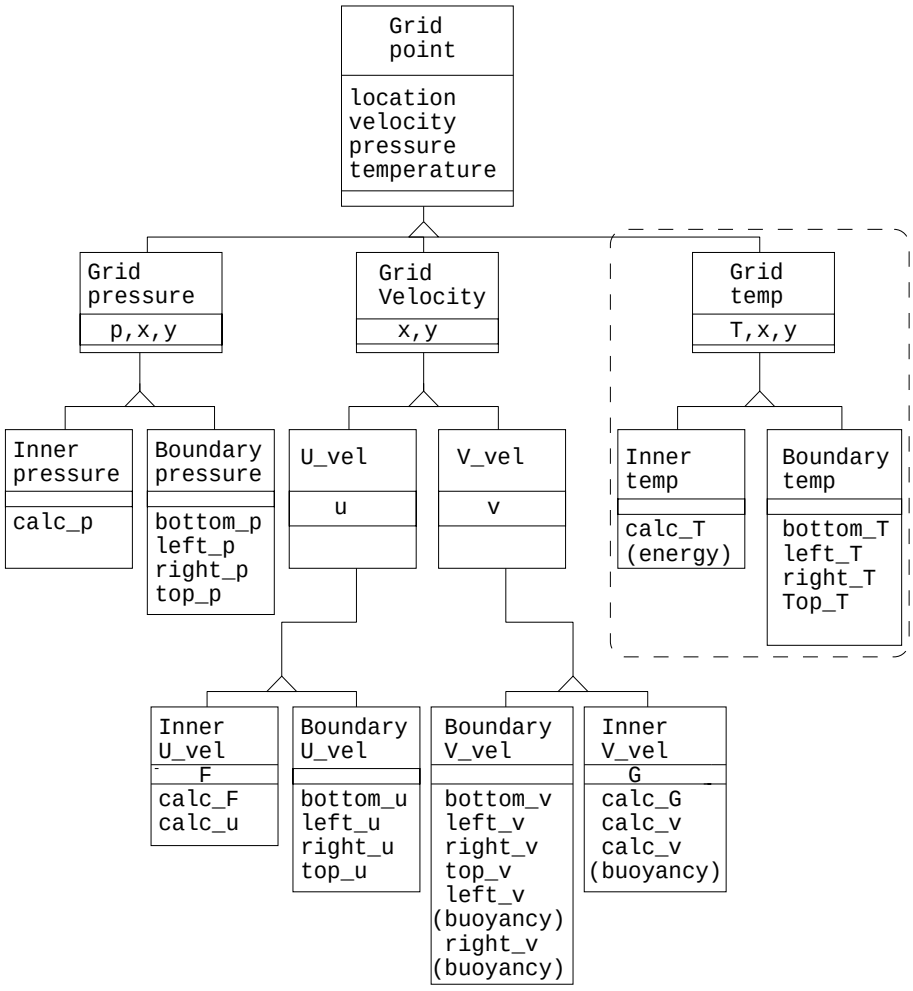

図 4.8: 拡張されたオブジェクト図

この場合のオブジェクト図を図 4.8に示す [11]。図の点線で囲まれた部分は新たに追加

されたクラスである。温度オブジェクトの定義される位置は、スタッガード格子の中央、つまり圧力定義の格子点オブジェクトと同じ位置で定義した。温度定義のオブジェクトクラスでは、属性に温度 T を持つ。更に、温度クラスをスーパークラスとして境界上に位置するクラスと境界内に位置するクラスが導出される。境界上のクラスでは、境界条件を表す式が格納される。今回は、境界上で温度の値を直接与える Dirichilet 条件を満たす式を用いた。境界内のクラスでは、(3.15) 式に示されたエネルギ方程式を離散化した式が格納される。これらは温度格子点オブジェクトインスタンスと周囲のオブジェクトインスタンスとの関係を記述したものである。y 方向速度オブジェクトクラスでは、浮力の影響を考慮した運動方程式が新たなメソッドに追加されている。また既存の y 方向の運動方程式を表すメソッドもそのままの形で確保されているために、浮力の影響を考慮する複合対流を解析する場合と考慮しない強制対流を解析する場合を容易に使いわけることが可能になる。加えて既存のクラス、属性、メソッド、は変更点を除いて再利用され、プログラミングが比較的容易になることが予想される。

4.2 領域オブジェクトの設定

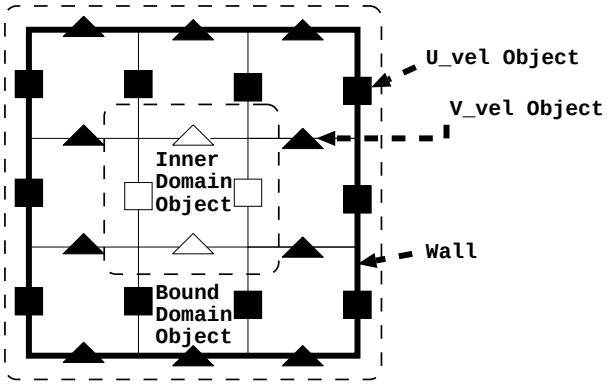

図 4.9: 速度に関する領域オブジェクトの設定

これまで述べてきた格子点をオブジェクトに取る方法は、流れの微子的な状態を自然なモデル化により実現してきたといえる。格子点オブジェクトをとることで熱問題への拡張も容易に行なわれるということを示したが、複雑形状の流れに以上の方法を適用するとなると、メソッドに差分式以外の何かを用いなければならず、これは困難である。そこで、もう少し別の視点からオブジェクトの取り方を考え直すことにする。

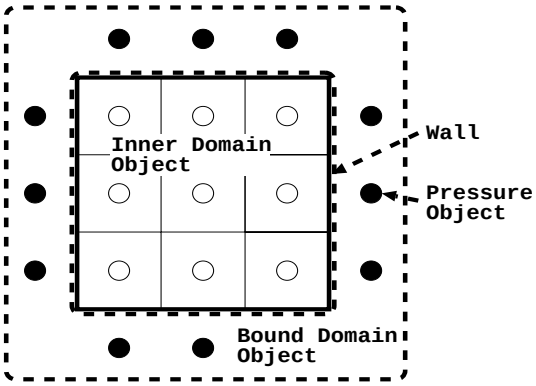

図 4.10: 圧力に関するオブジェクトの設定

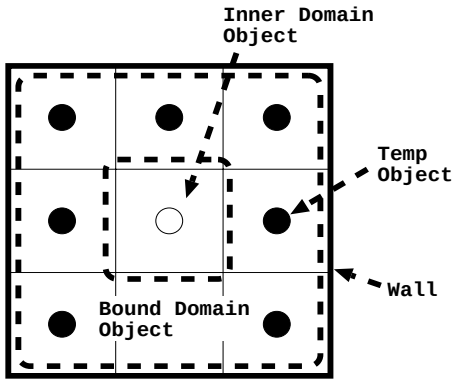

図 4.11: 温度に関するオブジェクトの設定

計算における領域というのは、格子点オブジェクトが複数集まって作られると考えられる。そこで、領域オブジェクトを新たに考え、格子点オブジェクトと集約関係にあるとする。このようにモデル化すると、以前に用いられた格子点クラスやそれらの関係を表す式などがそのままの形で再利用され、領域オブジェクトのインスタンスを複数組み合わせるだけで、比較的容易に複雑形状の流れにも適用され得ることが期待される。更に領域をオブジェクトに取れば領域単位での並列化によるモデルの高速化が可能になり、非常に多くの格子点オブジェクトを取り扱うことで、より精度の良い近似が行なわれ得ることが期待される。領域クラスは更に境界に影響される領域境界クラスと領域内部のクラスに汎化される。

どの格子点オブジェクトが、領域の境界上クラスのオブジェクトと領域内部のクラス

のオブジェクトに定義されるかを速度に関して図 4.9 、圧力に関して図 4.10 、温度に関して図 4.11 に示す。領域間の境界は通常の境界条件により与えられる境界とは性質の異なったものであり、境界オブジェクトクラスは更に通常境界のクラスと領域の分割により新たに生じる分割境界クラスを導出する。これらのクラスではメソッドに各々離散化式を領域内でまとめた式を格納している。また、計算手順を管理するクラス (Brain クラス) を設け、このクラスでは離散化式をまとめたメソッドを順に呼び出すメソッドを持っており、計算手順を管理する働きを行なっている。

以上のオブジェクト図を図 4.12 に示す。

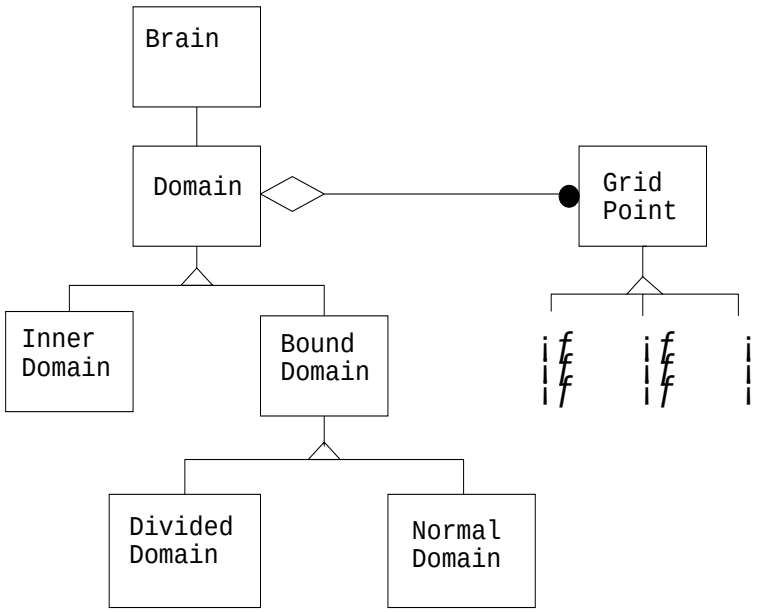

図 4.12: 領域をオブジェクトに取る場合のオブジェクト図

オブジェクト図には、メソッドをすべて記述しきれないので、以下に各々のクラスの主要なメソッドを示す。データをファイルに落す処理を行なうような本質と関係ないメソッドについては省略する。

各クラスに対して

○メソッド名：役割：引数：返り値：格子点クラスでのメソッド; の順に記述する

クラス Inner Domain

○ Calc_UV：速度 u、v の計算：自己の領域オブジェクト (クラス Domain) ：なし：calc_u(ク

ラス Inner U_vel),calc_v(クラス Inner V_vel);
○ Calc_FG：F、G の計算：自己の領域オブジェクト (クラス Domain) ：なし：calc_F(クラス Inner U_vel)、calc_G(クラス InnerV_vel);
○ Calc_P：圧力 p の計算:自己の領域オブジェクト (クラス Domain) ：なし：calc_P(クラス Inner Pressure);

クラス Normal Domain
○ Bound_TU：領域オブジェクトの上側境界の処理:境界条件により決まる：なし：top_u(クラス Boundary U_vel);
○ Bound_TV、Bound_TP、Bound_TT:上と同様;
以下領域オブジェクトに対して左側境界、右側境界、下側境界ともに同様の方法による。

クラス Divided Domain
○ BoundU：隣接する領域オブジェクト間の境界での平均値 u の計算：自己の領域オブジェクト (クラス Domain)、領域オブジェクト (境界に対して左側)、領域オブジェクト (境界に対して右側)：なし：なし
○ BoundV：隣接する領域オブジェクト間の境界での平均値 v の計算：自己の領域オブジェクト (クラス Domain)：領域オブジェクト (境界に対して上側)、領域オブジェクト (境界に対して下側)：なし：なし
これらの場合のオブジェクト及び使用されるメソッドについて図 4.13 に概要を示す。
○ RPasFG：対象とする領域オブジェクトの右側境界オブジェクトでのメッセージ通信と自己情報処理による FG の計算：自己の領域オブジェクト (クラス Domain)：領域オブジェクト (自己オブジェクトに対し右側)：なし：calc_F(クラス Inner U_vel)、calc_G(クラス Inner V_vel);
以下 LPasFG、TPasFG、BPasFG メソッドについても同様である。
○ RBPasFG：対象とする領域オブジェクトの右下 (角部分) でのメッセージ通信：自己の領域オブジェクト (クラス Domain)：領域オブジェクト (自己オブジェクトに対して右側)、領域オブジェクト (自己領域に対して右下)、領域オブジェクト (自己領域に対して下側)：なし：calc_F(クラス Inner U_vel)、calc_G(クラス Inner V_vel);
以下 TRPasFG,LBPasFG,LTPasFG メソッドについても同様である。

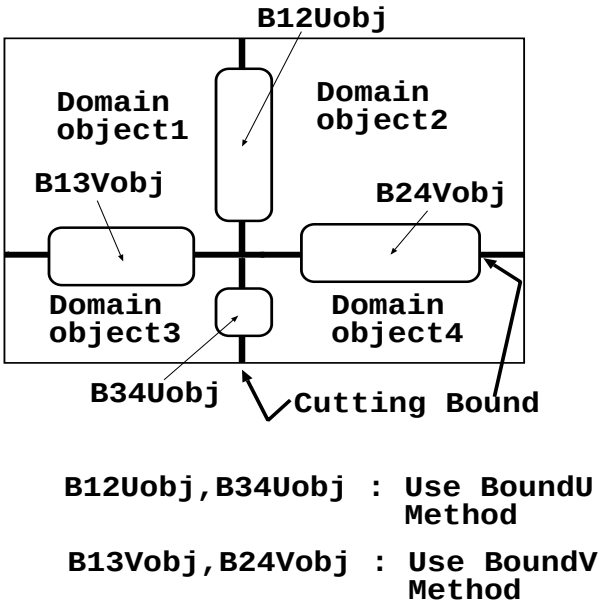

図 4.13: 分割境界オブジェクトでの速度 u,v の扱い

FG に関するメソッド及びオブジェクトとの関係を図 4.14 に示す。図では１つの領域オブジェクトインスタンス (Domain x) を考えており周囲に領域インスタンスが隣接していると考えている。

○ RPasUV：対象とする領域オブジェクトの右側境界オブジェクトでのメッセージ通信と自己情報処理による速度 U、V の計算：自己の領域オブジェクト (クラス Domain)：領域オブジェクト (自己オブジェクトに対して右側)：なし：calc_u(クラス Inner U_vel)、calc_v(クラス Inner V_vel);

以下 LPasUV、TPasUV、BPasUV メソッドについても同様である。

UV に関するメソッド及びオブジェクトとの関係は、図 4.14 の場合と同様の考え方でメソッドを F、G から U、V に置き換えて考えればよい。ただし、メソッド内で記述される内容は全く異なる。

○ RBPasUV：右下領域オブジェクトでの周囲オブジェクトとメッセージ通信による速度 U、V の計算：自己の領域オブジェクト (クラス Domain)：領域オブジェクト (自己領域に対して右側)、領域オブジェクト (自己領域に対して右下)、領域オブジェクト (自己領域に対して下側)：なし：calc_u(クラス Inner U_vel)、calc_v(クラス Inner V_vel);

以下 LBPasUV、TRPasUV、LTPasUV メソッド共に F、G の場合と同様である。

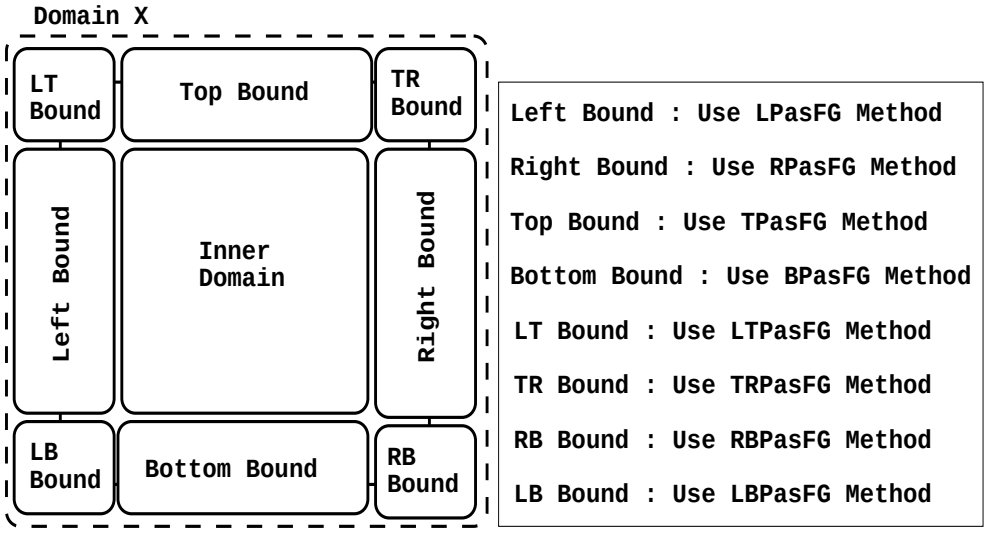

図 4.14: オブジェクトと F、G に関するメソッドの対応

○ RPasP：対象とする領域オブジェクトの右側境界オブジェクトでのメッセージ交換と自己情報処理による圧力 p の計算：自己の領域オブジェクト (クラス Domain)：領域オブジェクト (自己オブジェクトに対して右側)：なし：calc_p(クラス Inner_pressure);
LPasP、BPasP、TPasP メソッドについても同様の考え方による。
○ RPasT：対象とする領域オブジェクトの右側境界オブジェクトでのメッセージ交換と自己情報処理による温度 T の計算：自己の領域オブジェクト (クラス Domain)：領域オブジェクト (自己に対して右側)：なし：calc_T(クラス Inner temp);
LPasT、BPasT、TPasT についても同様の考え方による。
○ RBPasT：右下領域オブジェクトでの周囲オブジェクトとメッセージ通信による温度 T の計算：自己の領域オブジェクト (クラス Domain)：領域オブジェクト (自己に対して右側)、領域オブジェクト (自己に対して下側)、領域オブジェクト (自己に対して右下)：なし：calc_T(クラス Inner temp);
LBPasT、LTPasT、TRPasT メソッドについても同様の考え方による。
○ BoundP：境界での領域オブジェクトに対して圧力場の収束判定値を返す：自己の領域オブジェクト (クラス Domain) 領域オブジェクト (隣接境界に対して相手側のオブジェクト)：Rmax(収束判定値)：なし;

以上のメソッドが各々のクラスのインターフェースとして各クラス内へ格納される。各クラスでオブジェクトインスタンスを複数生成させ、以上のメソッドを各インスタンスに対して用いることで、容易に複雑な領域をシミュレーションできるということを示唆している。図 4.15 は分割境界オブジェクトと通常の境界オブジェクトを組み合わせることで、格子点で離散化された領域を、領域オブジェクト単位で表現している一例である。図で、Bottom Bound オブジェクト、Right Bound オブジェクト、RB Bound オブジェクトは、他の領域オブジェクトとのメッセージ通信を要する。

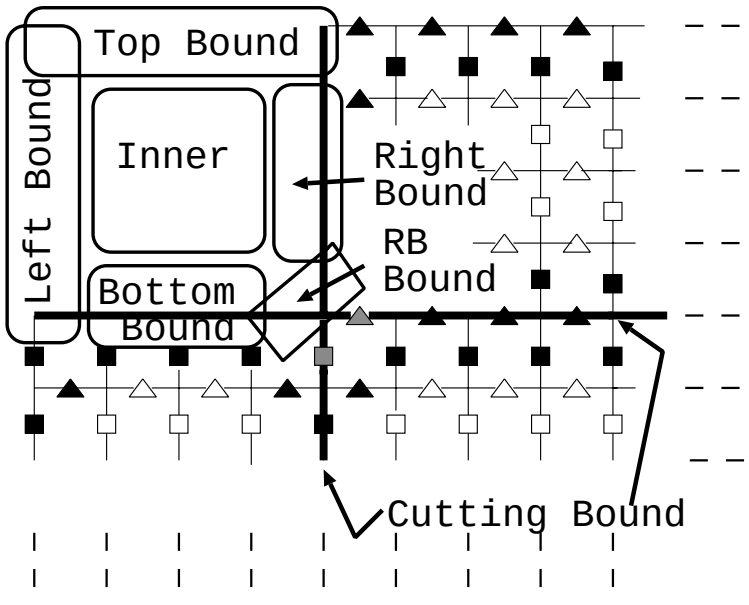

図 4.15: インスタンス設定の一例

各領域オブジェクトではその内部に格子点クラスのオブジェクトや格子点クラスのメソッドが各々含まれていると考えられる。領域クラスのオブジェクトでのメソッドにおける引数は、上に述べたように境界クラスのオブジェクトであれば自己のいる領域クラスのオブジェクトとメッセージ通信を必要とする領域クラスのオブジェクトを取るのみでよく、境界内クラスのオブジェクトでは計算 (速度、圧力等) を行なうために引数に自己のいる領域クラスのオブジェクトを用いるメソッドを境界内クラスのオブジェクトが呼び出すだけでよい。このため領域クラスオブジェクトのメソッド内では、格子点インスタンスのリンクに関する処理や、周囲領域オブジェクトにおける格子点インスタンスとのメッセージ通信のための処理が含まれていて、やや複雑なものとなっている。しかし、このよ

うな複雑な処理は領域オブジェクトインスタンスレベルでは考慮しなくてよく、つまり領域オブジェクトレベルではブラックボックスと考えてよく、周囲領域とメッセージ通信が必要なら、単に相手の領域オブジェクトを指定してやるだけでよい。このようにオブジェクトとメソッドを組み合わせるだけで極めて容易にシミュレーションが行なわれる。

以上のオブジェクトがメソッドを呼び出す順序をまとめるのがBrainクラスであり、このクラスではこれらの処理を正しい順序で呼び出す為のメソッドを一つ有している。

第 5 章

実験

5.1 ポアソン方程式の熱伝導問題

ここでは格子点をオブジェクトとするオブジェクト指向差分解法の最初の実験としてポアソン方程式の熱伝導問題の結果を示す。

図 5.1 のように平行平板を一様に Q で加熱した問題を考える。境界上では、2 種類の境界条件が与えられる。物理量を直接与える Dirichlet 境界条件と、物理量の勾配で与える Neumann 境界条件である。この例では、向かいあう 2 境界を Dirichlet、他方を Neumann で与えることにした。

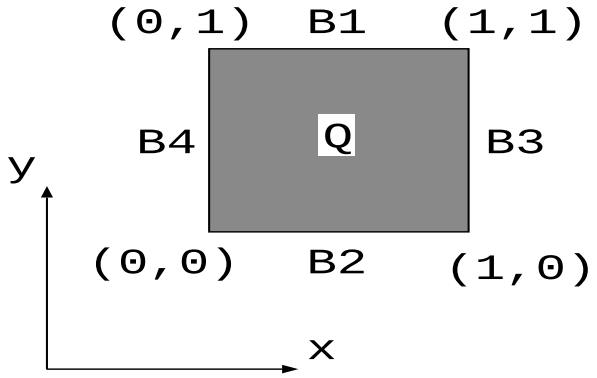

図 5.1: 平板での熱伝導問題

つまり

y=1.0(B1) : T=673.0;

y=0.0(B2) : T=233.15;

x=1.0(B3) : $\partial T/\partial x = 0.0$;

x=0.0(B4) : $\partial T/\partial x = 0.0$;

である。また、発熱 Q(x,y)=1000、$\Delta x = \Delta y = 0.02$、である。

図 5.2 は平板表面上での温度分布の結果を示している。結果は妥当であり、オブジェク

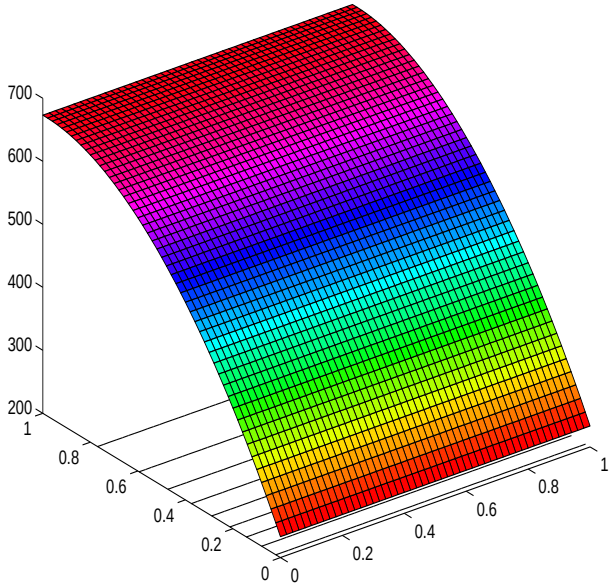

図 5.2: 平板上の温度分布

ト指向による解法が正しい数値解をもたらすことを確認した。オブジェクト指向を用いた本解法では、格子点オブジェクトが周囲のオブジェクトから自己の属性値 (温度) を更新するというモデリング概念を実装段階においてもある程度実現でき、定量的な評価はできないが、通常のプログラムに比べて理解しやすくなっていると考えられる。また各オブジェクトは自己のクラスのメソッドしか呼び出すことができない。これには次のようなメリットがある。例えば、通常のプログラムであれば、オブジェクトを設定するかわりに、物理量を確保する変数を設け、その変数に対してはどのような処理もできるので、気がつかない間に変数に対して無関係な値が入っていたとしてもシミュレーションは実行され得る危険性が大きい。それに対してオブジェクト指向によるモデリングでは、このように不用意なミスから生じる誤ったシミュレーションの実行可能性が低くなり信頼性が向上する

といえる。これはプログラムのデバッグにおいても、オブジェクト指向を用いた解法が容易に行なえることを明示している。しかしながら、オブジェクト間のリンクにおける手間は従来法で行なうのと本質的に変わるところはなく、メッセージ通信を行なう相手と自己のオブジェクトの正確なリンクに注意を要する。

ここではオブジェクト指向を用いたことによる信頼性の向上がなされ、理論的に統一された方法により、オブジェクト指向差分解法が確かに正しい解をもたらすことを確認した。

次のセクションでは、非線形方程式であるナビエ・ストークス方程式に格子点オブジェクトによるモデリング概念を適用し、本格的に流れ問題へオブジェクト指向の概念を用いてシミュレーションを行ない, モデリングの柔軟性や拡張性についても検討していくことにする。

5.2 オブジェクト指向による MAC 解法

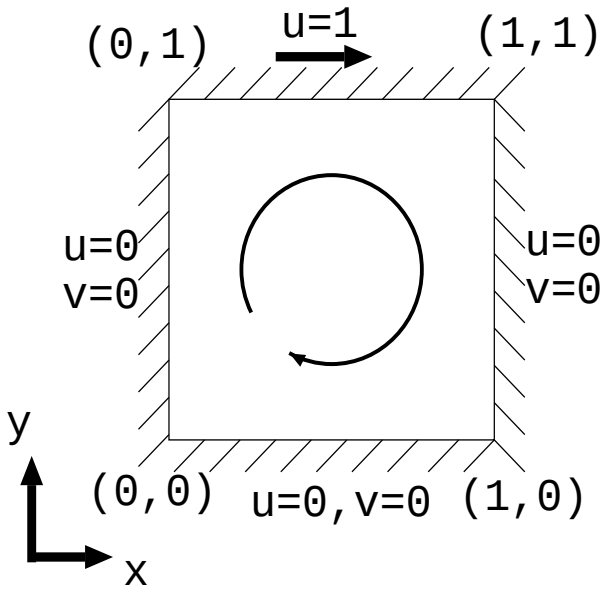

図 5.3: キャビティ流れの概要

ここではナビエ・ストークス方程式のシミュレーションをオブジェクト指向に基づく MAC 差分解法により行なった結果を示す。

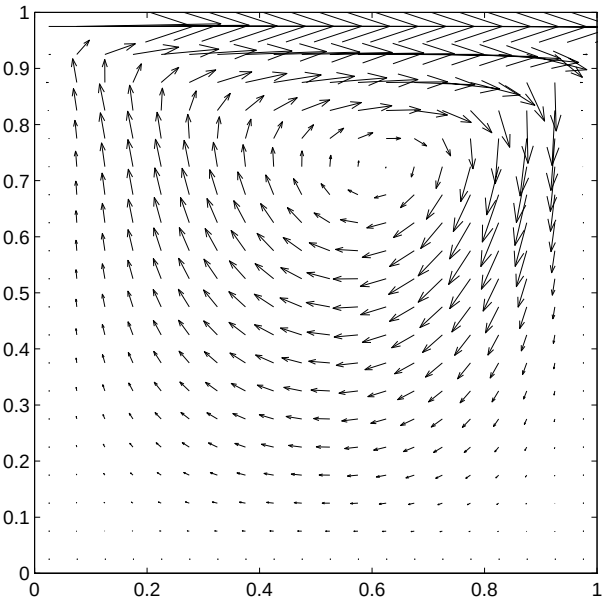

図 5.4: キャビティ流れの速度分布 (Re=100、$\Delta t = 0.001$)

オブジェクト指向による MAC 差分解法によるモデリングの検定問題として、通常よく用いられる正方キャビティ流れのシミュレーションを行なった。図 5.3 にその概要図を示す。図のように y=1 の面で速度 u=1 を与えている。

図 5.4 は Re 数が 100、$\Delta t = 0.001$、$\Delta x = \Delta y = 0.05$、step=40000 の場合の正方キャビティ流れの速度分布の結果である。(x,y)=(0.6,0.75) 付近に渦が留まり、従来法による場合の結果と定性的によく一致することが確認された。また図 5.5 は圧力分布についての結果であり、これも通常の差分法による結果と非常によく一致することが示された。

提案したモデルが確かに正しい解をもたらすことを確認したが、従来法との比較において何が優れているのか等について以下に述べていく。

図 4.3でメソッドに MAC 法による離散化式を用いた場合のオブジェクト図を既に示したが、オブジェクト図では物理量という関係で格子点オブジェクトクラスから圧力に関するクラス、速度に関するクラスに分けてモデリングを行なった。このことは次の点において重要であると考えられる。格子点クラスから圧力クラス、速度クラスを導出しない場合には、格子点オブジェクトに物理量の属性すべてを持たせることになる。この場合、格子点オブジェクトクラスでは、圧力に関するインスタンス、速度に関するインスタンスが区別されず、全ての物理量の属性を持ったオブジェクトを複数生成させることになる。格子

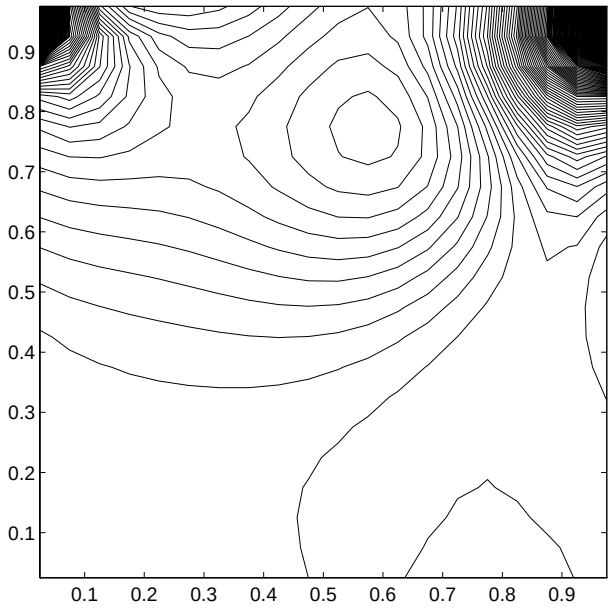

図 5.5: キャビティ流れの圧力分布 (Re=100、$\Delta t = 0.001$)

点クラスには、圧力の属性を変更するメソッドと速度の属性を変更するメソッドが当然ながら格納されなければならない。このように格子点クラスからサブクラスを派生させないと、そのクラス内のメソッドの数は派生させる場合に比べて多くなる。加えて、圧力の属性を変更させたい場合に、誤って速度の属性を変更してしまうという予想しないような危険性を秘めている。それに対して物理量という関係でサブクラスを導出すれば、このような矛盾は生じないだけでなく、他の物理量クラスを格子点クラスから導出すれば容易に連成問題へ拡張される可能性を持っている。このことについては次のセクションで述べる。

また、オブジェクト図では速度オブジェクトクラスを x、y 方向に分けて考えたが、このことも上と同様の効果を持つ。更に z 方向のクラスを加えれば３次元にも柔軟に拡張されるはずである。

以上のここで示した MAC 法を用いたオブジェクト指向モデリングと通常の差分解法の比較を行なう。前のセクションでも少し述べたが、従来法では、オブジェクトを設定する代わりに、オブジェクトモデルでいうと属性に相当する物理量を確保する変数を宣言し、その変数に対して差分方程式を適用することによりシミュレーションを行なう。しかし、その変数に対してはどの様な差分式が用いられてもプログラム的には間違いではなく、速度の変数に対して圧力の離散化式が誤って用いられるということも起こり得るはずであ

る。先述では、オブジェクトの属性が温度だけであったが、ここでのオブジェクトモデルでは、u、v、p の３つの属性が用いられている。更に他の物理量を増加させれば、属性の数が多くなる。このことは通常の解法において変数の数が増えるということと等価である。変数の数が増えると当然のように、変数に対し予期しない値を代入したり、その変数に関係のない関数を使うということが増え、間違った処理を継続する可能性が多くなり、モデル全体で見た場合の信頼性は低いものとなるはずである。更にプログラムのバグ、特に実行時のエラー検出が困難になる。

ここで提案した MAC 解法によるオブジェクトモデルでは、以上のことをふまえた上で、非常に信頼性の高いモデリングであり、モデリングが実装段階においても実現されているという意味で精度のよいシミュレーションの実行が保証されていると考えられる。

5.3 熱問題への拡張

MAC 法を用いた上述のオブジェクトモデルが異なった属性を持つ格子点オブジェクトに対して柔軟に拡張されるということを示すために、つまり上述のモデルが熱問題に対して柔軟に対処できるであろうことを証明する為に、ここでは、オブジェクトモデルに温度定義の格子点オブジェクトを加えた、拡張されたモデルにより正方キャビティ複合対流や自然対流のシミュレーションを行なった。

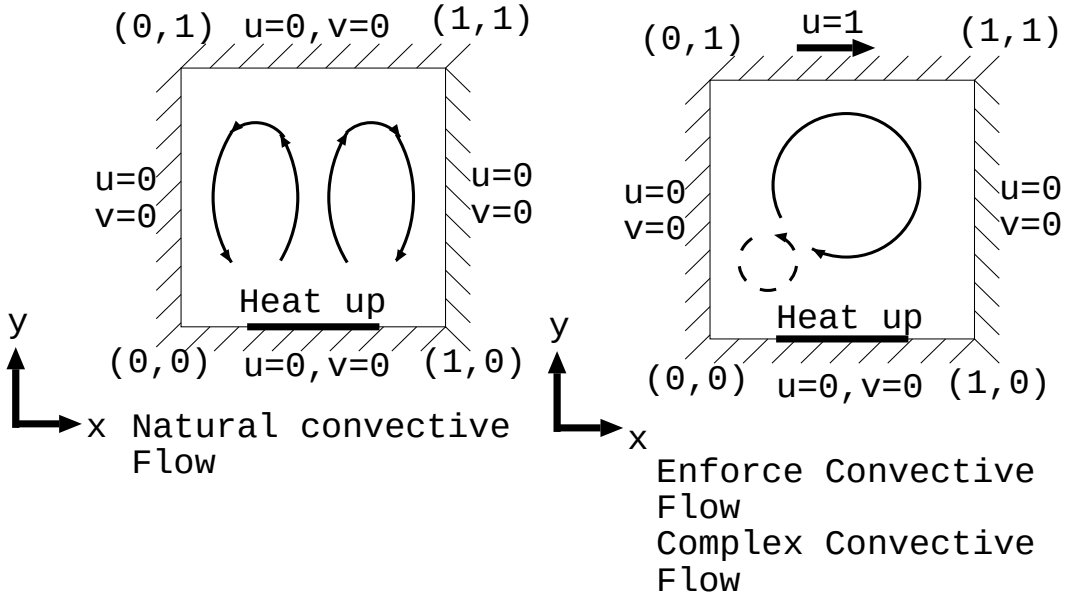

図 5.6: 熱を加えた流れの概要

実行例として、図 5.6 のように下壁を加熱することにより、強制対流、自然対流、複合対流の３つをシミュレーションする。複合対流と強制対流の違いは、複合対流の場合には、流れが温度により影響される熱と流れが連成するような流れを示すのに対し、強制対流の場合には、流れ場が温度による浮力の影響を受けない程小さいとし考慮しないということである。図 5.7 は Re=100、T_{high}(加熱壁)=503.3、$T_{else} = 273.3$、Gr=10^5、$\beta = 2.58 \times 10^3$、$\Delta t = 0.001$、$\Delta x = \Delta y = 0.05$、step=40000 で下壁 (0.25-0.75) を加熱した場合の強制対流の温度分布図の結果である。加熱壁面上で温度勾配が急激に変化しているといえる。この結果も当然ながら通常の差分法により得られる結果と一致することを示している。強制対流の速度分布と圧力分布についての結果は、図 5.4 、図 5.5 に同じである。

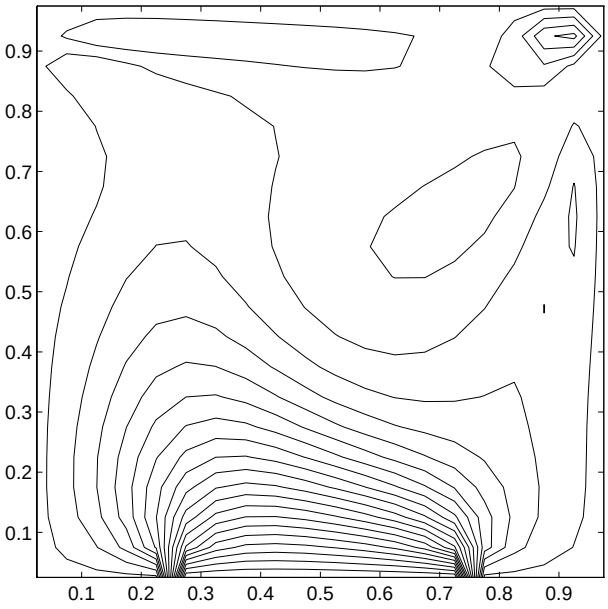

図 5.7: 強制対流の温度分布 (Re=100、Δt = 0.001、T_{high}=503.3、Gr=10^5)

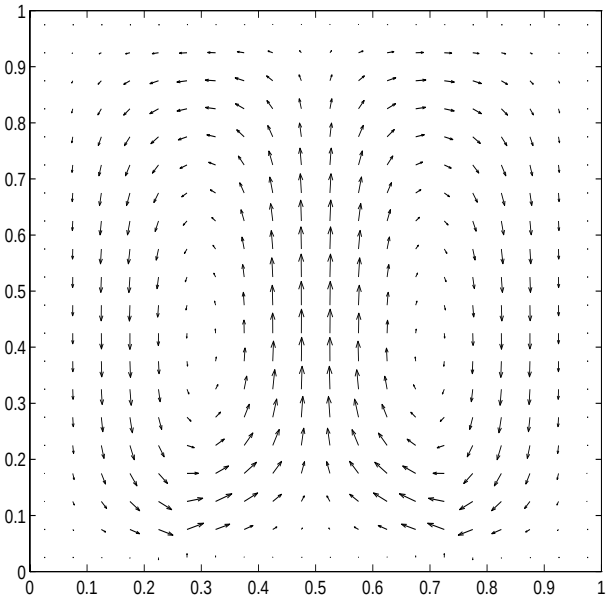

図 5.8: 自然対流の速度分布 (Re=100、Δt = 0.001、T_{high}=503.3、Gr=10^5)

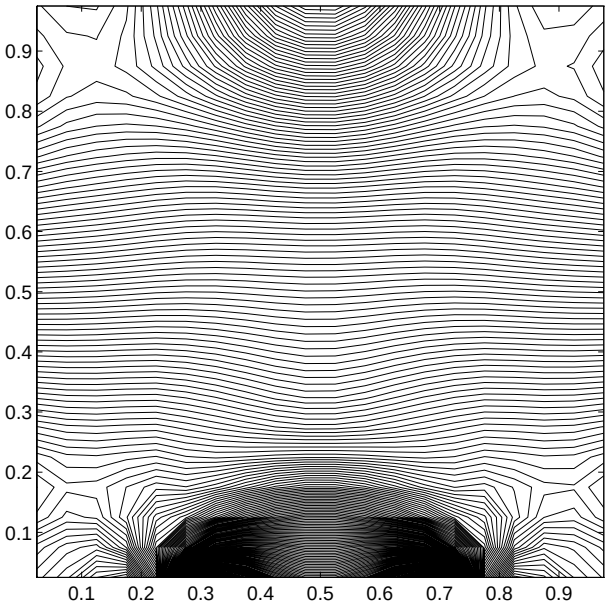

図 5.9: 自然対流の圧力分布 (Re=100、Δt = 0.001、T_{high}=503.3、Gr=10^5)

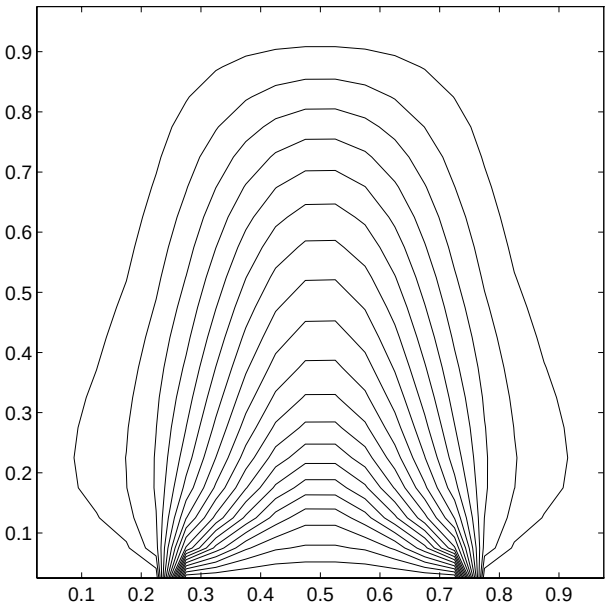

図 5.10: 自然対流の温度分布 (Re=100、Δt = 0.001、T_{high}=503.3、Gr=10^5)

図 5.8 は Re=100、T_{high}(加熱壁)=503.3、$T_{else} = 273.3$、Gr=10^5、$\beta = 2.58 \times 10^3$、$\Delta t = 0.001$、$\Delta x = \Delta y = 0.05$、step=40000 で下壁 (0.25-0.75) を加熱した場合の自然対流の速度分布図の結果である。予想されるような流れが得られている。図 5.10 はこの場合の温度分布図、図 5.9 は圧力分布図である。

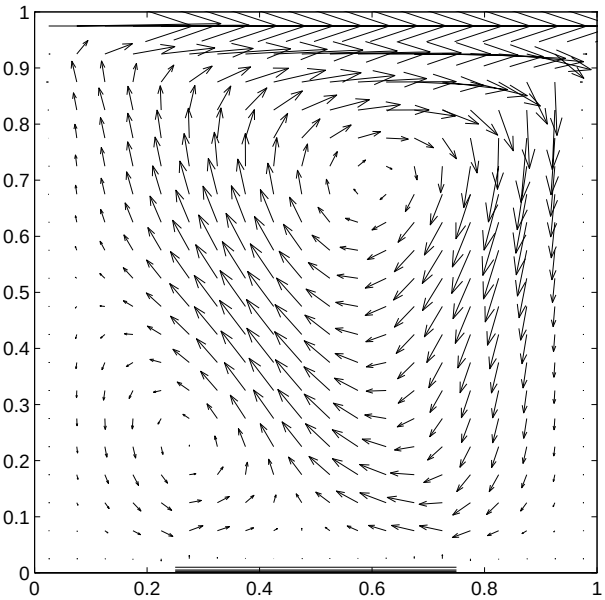

図 5.11: 複合対流の速度分布 (Re=100、$\Delta t = 0.001$、T_{high}=503.3、Gr=10^5)

図 5.11 は Re=100、T_{high}(加熱壁)=503.3、$T_{else} = 273.3$、Gr=10^5、$\beta = 2.58 \times 10^3$、$\Delta t = 0.001$、$\Delta x = \Delta y = 0.05$、step=40000 で下壁 (0.25-0.75) を加熱した場合の複合対流の速度分布図の結果である。図のように強制対流の場合と異なり左下方で新たな渦が発生するのをうまくシミュレーションできている。当然ながらこの結果も従来法の結果と一致する。

以上のように、オブジェクト指向によるモデリングの結果は、正しい解をもたらすということを、ここでも確認した。問題はオブジェクト指向を用いたことによりどのくらい容易かつ信頼できるシミュレーションが行なわれたかにある。

図 4.8 に熱を加えた問題に対するオブジェクトモデルを既出したが、本モデリングでは、新たに温度オブジェクトクラスを加えることで、熱への連成問題へ容易に拡張されることを示した。この場合、既存の MAC 法を用いたオブジェクト指向モデリングから、新たに温度オブジェクトの取り扱いをモデリング概念に加えたものとみなせ、既存クラスの

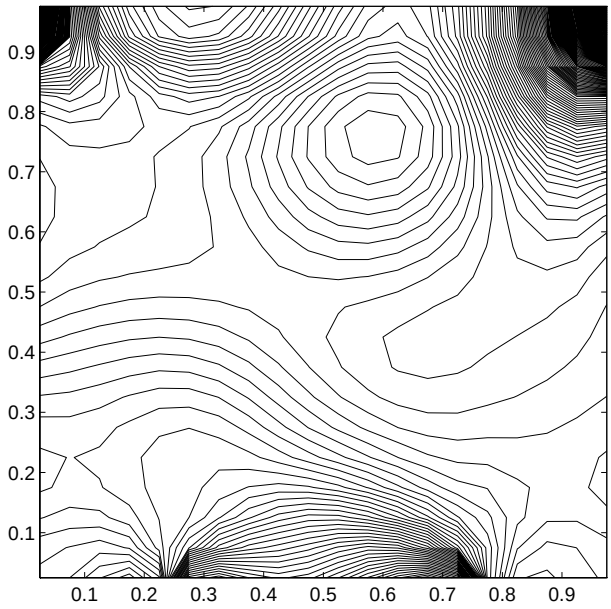

図 5.12: 複合対流の圧力分布 (Re=100、$\Delta t = 0.001$、T_{high}=503.3、Gr=10^5)

属性やメソッドがほぼそのまま再利用できることを意味している。このことは熱問題へのオブジェクトモデルの移行が柔軟に行なわれることを示している。この際、浮力の影響を考慮したメソッドを v 速度オブジェクトクラスに加える必要があるが、既存の浮力の影響を加えないメソッドも依然として、そのクラス内に格納されたままであり、従って v 速度クラスのオブジェクトがメソッド呼び出しに浮力を考慮したメソッド、考慮しないメソッドを自由に選択できる。このため、浮力の影響を考慮した複合対流、浮力を無視する強制対流の両方を容易にシミュレーションできる。

従来法では、オブジェクトに属性を持たせる代わりに物理量の値を保持する変数を宣言することによる問題、つまり圧力に関する変数が速度に関する関数を呼び出す等の問題が生じることは、先に述べたが、ここでも同じ問題が伴う。更にここでの問題のように熱を考慮するとなると境界条件の数も増大し、誤った処理をさせる危険性が大きい。従って、従来法では拡張に対して不要な努力を必要とすると考えられる。これは概念から実装までのモデリングが一貫しておらず、実装においてデータ (変数) と手続き (関数) が無秩序に定義されている為に起こる。従来法ではこのように誰がプログラムを書いてもこの問題からは逃げられない。オブジェクト指向を用いた解法では、概念から実装までモデリングが一貫しているので、このような問題は生じない。ここでの拡張されたオブジェクトモ

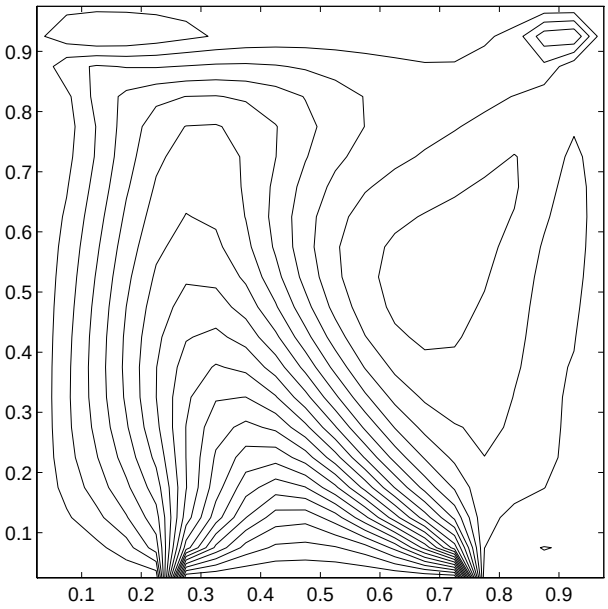

図 5.13: 複合対流の温度分布 (Re=100、$\Delta t = 0.001$、T_{high}=503.3、Gr=10^5)

デルでは、温度クラスの追加、そして各物理量クラスに対し境界クラス、境界内クラスを導出させることで、各オブジェクトが誤った処理を行なわれる危険性をある程度低くしている。

以上のことから、ここでのモデリングは従来法によるものに比べて信頼の高いものであることを実証するとともに、格子点オブジェクトから物理量の関係でサブクラスを派生させることで、前セクションで述べた基となるオブジェクトモデルが柔軟に熱への連成問題へモデリングの精度を下げることなく移行されるということを確認した。

5.4 領域オブジェクトによる解法

ここでは、格子点オブジェクトを集約した領域オブジェクトによるオブジェクト指向解法における実行例とその有効性を示す。すでに、領域をオブジェクトに取ることにより、単に領域を組み合わせるだけで複雑形状の流れをシミュレーションできるであろうことを述べてきたが、実際に確認するとともにモデリングの問題点等について検討することにする。まず簡単な分割による流れのシミュレーションをいくつか考えることにする。複雑形状への拡張は組合せの数が多くなるだけで比較的容易に実行されると考えられる。

図 5.14 は２つの領域オブジェクトを用いる場合の正方キャビティ問題の概要図を示している。１つのエリアに５つのインスタンスがあり、分割境界でも境界領域オブジェクトを設定している。このように領域ごとに境界オブジェクト、境界内オブジェクトを設定し、領域オブジェクト間の分割境界オブジェクトを設定する。そして領域オブジェクトが自己の持つメソッドを呼び出し、自己のメソッド内で格子点オブジェクトが自己のメソッドを更に起動させることで、モデリングが可能となる。領域オブジェクトがメソッドを呼び出し、内部オブジェクトを起動させる概念を図 5.15 に示す。

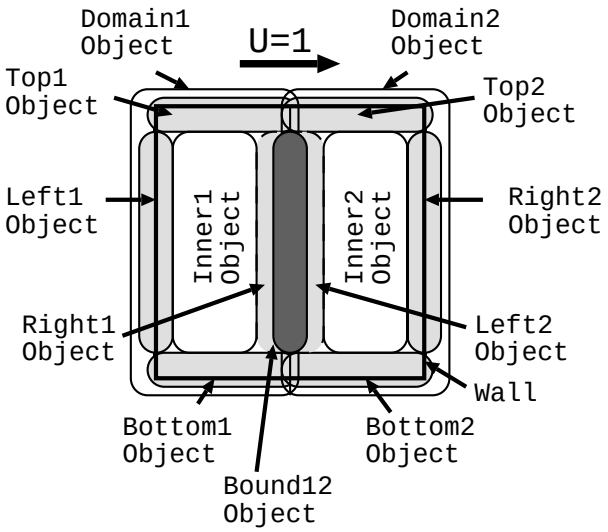

図 5.14: ２領域オブジェクトによるキャビティ問題

外側の枠が領域オブジェクトのメソッドが行なう処理、内側の枠が格子点オブジェクトのメソッドが行なう処理を示している。領域オブジェクトが自己のメソッドを呼び出すだ

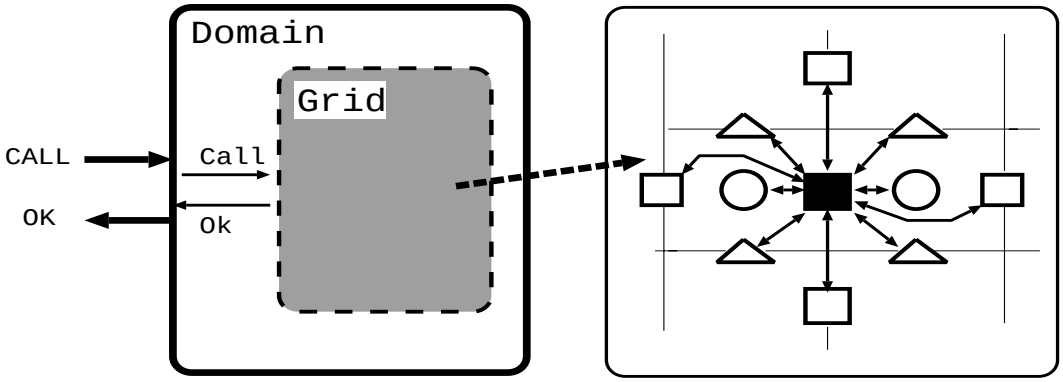

図 5.15: オブジェクト起動概念

けで、自動的に格子点オブジェクトが自己の状態を周囲のオブジェクトから変化させるということを領域オブジェクト内のすべての格子点オブジェクトについて行なうことを意味している。

図 5.14 では各領域境界オブジェクトを Right、Left、Top、Bottom に分割したオブジェクトを設定しているが、分割境界オブジェクト１、２、通常境界オブジェクト１、２というようなオブジェクトのとりかたも可能である。こちらの方がオブジェクトの数が少なくてすむという利点はある。この場合について図 5.16 に示す。ただ、説明の都合上、図 5.14 のように書いた方が、ある計算段階で各オブジェクトが各々一つのメソッド呼出しに対応しているという意味で、わかりやすいと考えて前者のオブジェクトの取り方をしている。

今回はキャビティ流れの計算を上の領域２分割の場合、４分割の場合について行なった。結果はいずれも先の分割をしない場合と同じであることが示されたが、ここで改めて示すまでもないので省略する。以上からオブジェクトを領域にとる場合における解の信頼性は保証されるということが確認された。

次に、キャビティ流れの中に角柱を置くことによるやや複雑な流れの解析を取り扱った。図 5.17 は８分割領域オブジェクトによる角柱を置いたモデルの概要図である。図では各々の領域オブジェクトサイズを全て同じ大きさにしているが、オブジェクト間の境界が一致する範囲において大きさを変えることができる。

シミュレーション結果を図 5.18 から図 5.20 に示す。なお、図 5.18 から図 5.20 は Re=100、T_{high}(加熱壁)=503.3、$T_{else} = 273.3$、Gr=10^5、$\beta = 2.58 \times 10^3$、$\Delta t = 0.001$、

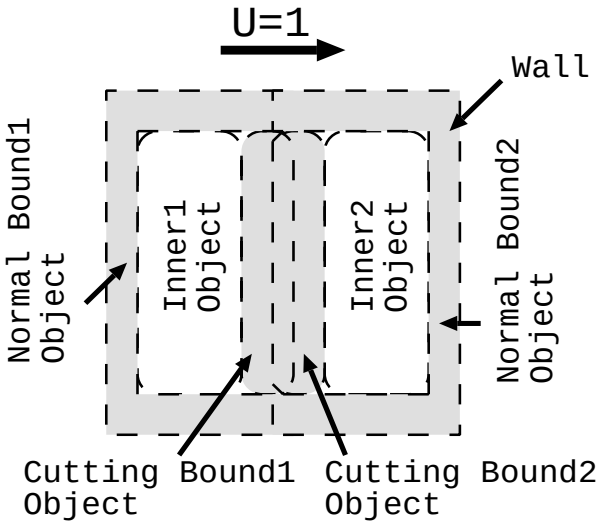

図 5.16: キャビティ問題に対する別のオブジェクトの設定方法

$\Delta x = \Delta y = 0.05$、step=40000 でありシミュレーションでは、強制対流を取り扱い角柱表面上を加熱している。今回の実験では角柱表面全体を通して一様な加熱を与えたが、例えば角柱の左側の面だけ局部的に加熱するということも容易にできる。また、角柱以外の壁面を局所的に加熱することも同様に容易に行なわれる。つまり、加熱領域が領域オブジェクトの各々の境界上でのオブジェクトの大きさに依存する。この為、小範囲を加熱する場合にはオブジェクトの大きさを小さくとり、反対に広範囲を加熱するときにはオブジェクトの大きさを大きくとることで達成される。

これを通常の差分法で実行しようとすると、どの格子点が角柱表面の点であるかを決めてそれらの点に対して、角柱表面での境界条件を満たすように領域全体で差分式を作成し連立方程式系を解く必要があり、かなり面倒であるといえる。それに加えて角柱の上部を外すというような変更を行ないたいときにも新たに格子点レベルでリンクする必要性が生じる。このように問題が少し変えられるだけで、格子点レベルでのリンクずけをしていたのでは、手間がかかる上に間違えた処理をさせてしまうことが多くなる。それに対して領域をオブジェクトに取った場合には、一度格子点オブジェクト間の関係を決定すれば、領域オブジェクトレベルでは、格子点間の関係をいちいち考えなくてよく、自己の領域が他のどの領域と関係しているかを明示するだけでよい。

このように領域をオブジェクトに取る場合には、ある程度複雑な形状を有する流れに対

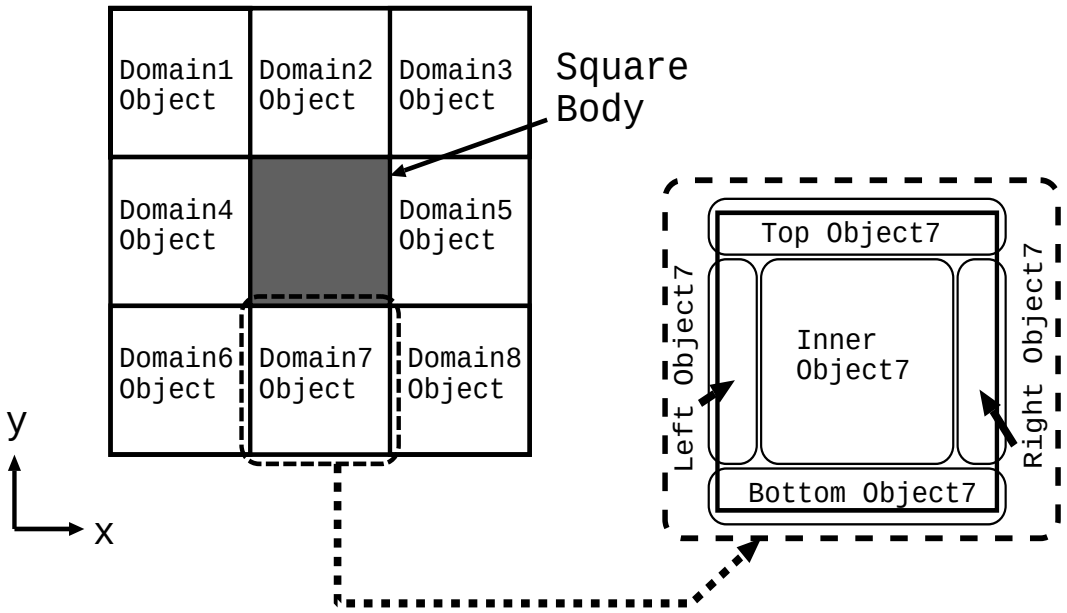

図 5.17: ８分割領域オブジェクトによる角柱まわりキャビティの概要

して拡張が容易にシミュレーションできるということを示すとともに、局部的な加熱による熱の影響も容易に加えることができるとういことを示した。本領域オブジェクトモデリングによる解法の柔軟性が示されたといえる。

次の例は、角柱のかわりに垂直平板を置いた場合のシミュレーションである。角柱の場合の領域に加えて新たに第９のオブジェクトが角柱のあった部分に追加される。これは、角柱としていた部分を対象領域に置き換える為である。オブジェクトの取り方は既述の通りであるが、第９のオブジェクトが加えられることによるメソッド呼び出しに加え、図 5.17 の領域オブジェクト２の下側境界、領域オブジェクト７の上側境界での呼び出すメソッドが変わることになる。更に領域１、３、６、８の平板に近い角でのメソッドを新たに呼び出す必要がある。しかし、変更は従来法と比較して非常に簡単であると考えられる。

図 5.21 から図 5.23 に速度分布、圧力分布、温度分布の結果を示している。

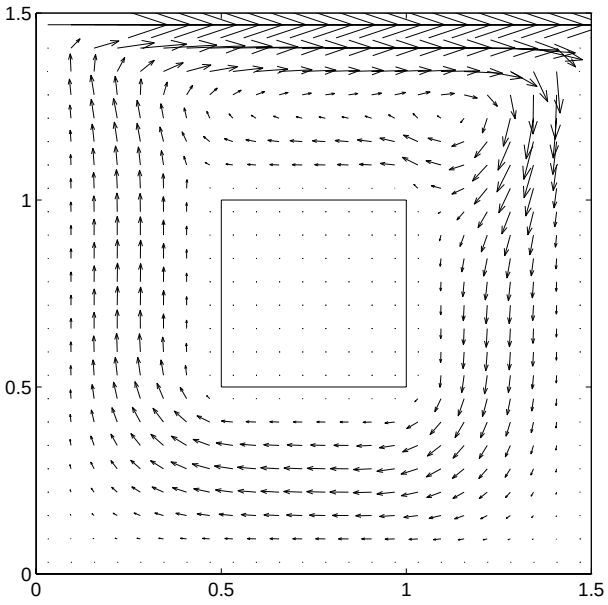

図 5.18: 角柱問題の速度分布 (Re=100、Δt=0.001、T_{high}=503.3、Gr=10^5)

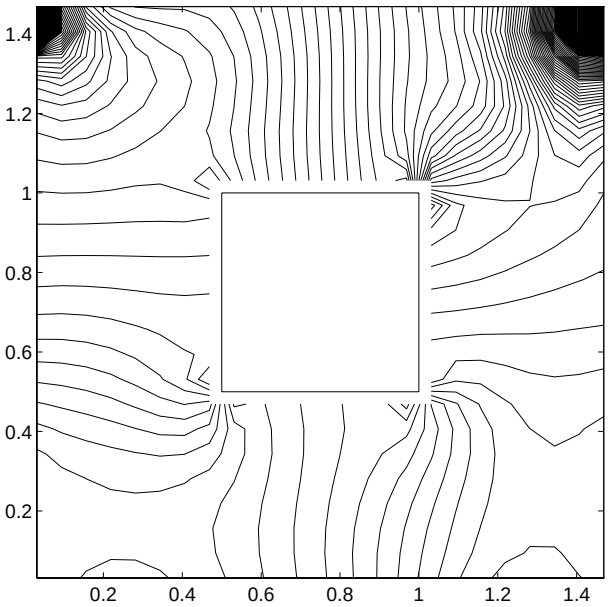

図 5.19: 角柱問題の圧力分布 (Re=100、Δt=0.001、T_{high}=503.3、Gr=10^5)

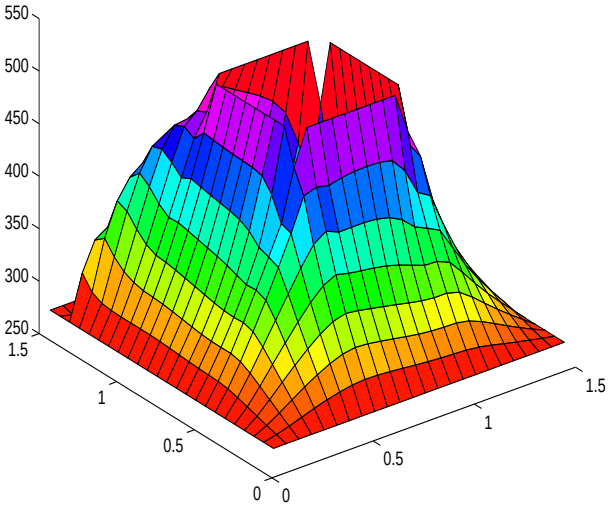

図 5.20: 角柱問題の温度分布 (Re=100、Δt=0.001、T_{high}=503.3、Gr=10^5)

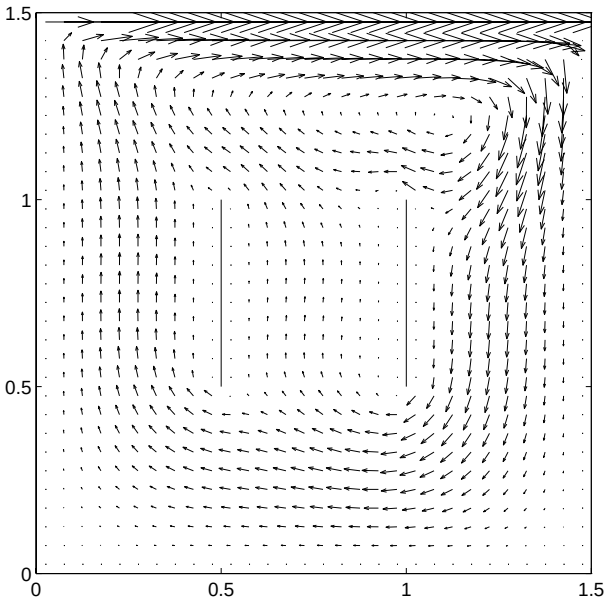

図 5.21: 平板問題の速度分布 (Re=100、Δt=0.001、T_{high}=503.3、Gr=10^5)

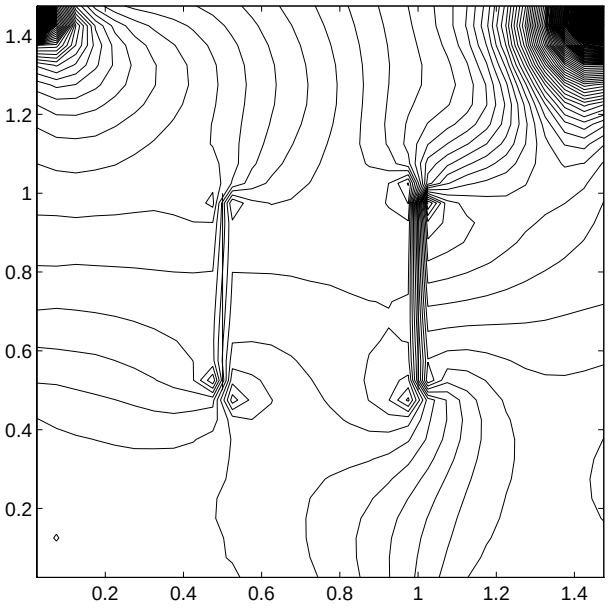

図 5.22: 平板問題の圧力分布 (Re=100、Δt=0.001、T_{high}、=503.3、Gr=10^5

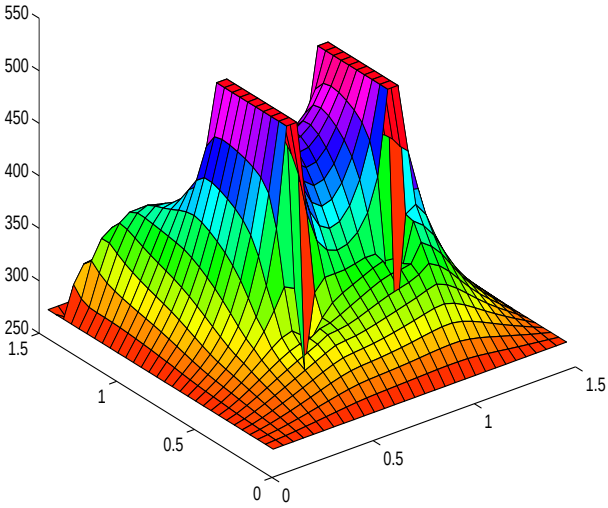

図 5.23: 平板問題の温度分布 (Re=100、Δt=0.001、T_{high}=503.3、Gr=10^5)

領域をオブジェクトにとる解法では、領域オブジェクトが格子点オブジェクトと集約関係にある。このため、既存のMAC法による格子点オブジェクトモデリング概念及び熱問題へ拡張されたモデリング概念が、領域オブジェクト内に包含され、一度、各格子点オブジェクトを自己と関係のあるオブジェクトとリンクしてしまえば、その後の変更は必要とされないということが示された。つまり領域オブジェクトがメソッドを呼び出せば、領域内における格子点オブジェクトのリンクや各格子点オブジェクトが周囲の物理量から自己の物理量を変容させるとう動作が自動的に行なわるということを意味している。この結果として領域オブジェクトを単に組み合わせるというだけで、熱移動を伴う流れや、複雑形状流れを比較的容易にシミュレートできることを実証した。しかし非常に複雑な形状を有する問題の解析において、オブジェクトの数の増加が予想される為に面倒な作業を行なう必要があると考えられる。この問題を解決するのに、例えばある領域オブジェクトの集団を更にまとめて一つの領域オブジェクトとするという処理等が必要になると考えられるが課題として残される。解法の高速化のために並列化を行なうということが考えられるが、並列化を行なう場合には各プロセッサの処理が独立している必要がある。領域オブジェクトモデルでは、領域オブジェクト単位での処理に少し独立でない処理が含まれている。考察で述べるが、各領域オブジェクトでの独立な処理への変更は容易に行なわれ得る。そのため解法の並列化への移行も比較的容易に行なわれると期待される。

ここでは、領域をオブジェクトに設定することで、比較的複雑形状を有する流れや複雑形状を有する流れに熱の影響を加えた流れを容易かつモデリングの精度を落すことなくシミュレーションできることを示した。そして、領域オブジェクトが格子点オブジェクトを集約するという形でオブジェクト指向モデリングを行なうことで、従来法と比較して、信頼性があり、しかも柔軟なモデリング方法を提案したと考えられる。

第 6 章

考察

6.1 モデリングのメリット

本研究ではオブジェクトを格子点とすることによりモデリングを始めた。格子点をオブジェクトとする概念は、微小体積の物理的な関係を表現していて、自然なモデリングが行なわれるということを示した。特にモデリングの概念が実装段階にまで確立されているという点で、従来法よりも高精度なモデリングであると考えられる。更に格子点クラスから境界クラスと境界内クラスを導出した。今回は固定壁面の境界条件式しか用いなかったが、自由表面の境界条件などを表す別の境界条件メソッドを境界上クラスに数種格納することで、対象とする問題に対して柔軟に対応できるということを示唆している。このことは従来法でも関数を複数作っておけば可能であるが、オブジェクトモデルでは境界クラスと境界内クラスを分けたことで、境界内クラスのオブジェクトが境界条件のメソッドを呼び出すのを防いでいる。従来法であれば境界内と境界の区別がなされていないので、境界内の格子点変数が境界条件式を表す関数を不正に呼び出すことができる。この為、プログラミングのデバッグにおける手間が増加すると考えられる。ポアソン方程式に関するオブジェクト図、図 4.2 におけるモデリングでは、従来法と比較して不正呼び出しが行なわれにくいというメリットがある。しかし、格子点オブジェクト間の関係をリンクするには、従来法と同様の苦労を強いられる。その他の MAC 法を用いたモデリング等は以上のメリットに加え以降述べていくメリットを持っている。

次に格子点クラスから速度に関するクラス、圧力に関するクラス、更に温度に関するクラスを導出させてナビエ・ストークスのシミュレーションを MAC 法をメソッドに用いた

オブジェクトモデルで行なった。各物理量クラスで格子点オブジェクトを生成させることで、圧力オブジェクトが速度に関するメソッドを呼び出すようなモデリングではありえないような操作を実行前に防いでくれることを述べた。このことは従来法が物理量を表す変数に対して、どんな不正な操作も許すのに対して本オブジェクトモデルが概念から実装段階まで一貫して信頼できるモデリングであることを示している。オブジェクト指向のメリットである情報隠蔽による効果であると考えられる。畠山らは一つの格子点クラスに属性として全ての物理量を持たせたモデルを用いたが、彼らのモデルでは従来の方法と同様に速度のオブジェクトが圧力に関するメソッドを呼び出すような危険性を持っている。この点において、格子点クラスから物理量という関係でサブクラスを導出したモデリングは優れており、より信頼性が高く精度の良いモデルであると考えられる。

物理量の関係で温度オブジェクトクラスを新たに格子点クラスから導出することにより、流れ場に熱が加わった連成した流れを比較的容易にシミュレーションできるといういうことをすでに述べた。熱問題を考慮する場合のオブジェクト図は図 4.8 のとおりであるが、この拡張されたモデルを用いることにより、既存のクラス、属性、メソッドが再利用され、図 4.3 の MAC 法による離散化式をメソッドに用いた格子点オブジェクトモデルの熱問題に対しての変更容易性が示された。

拡張されたオブジェクトモデルの属性やメソッドが、従来法の物理量を示す変数や差分式を表す関数に相当するといえる。このため、オブジェクトにおける属性の増加が変数の増加を意味し、メソッドの増加が差分式の増加を意味する。オブジェクトモデルでは、何度も述べるが自己のクラスのオブジェクトは自己のクラスのメソッドしか呼び出せず、圧力オブジェクトが速度オブジェクトのメソッドを呼べばエラーとなり、このような処理はできない。従来法ではこれと同じようなことが変数と関数の関係で容易になされてしまう。変数や関数が増えるとミスを犯す危険が大きい。従って、実装段階で間違った処理を実行することもありうる。拡張されたオブジェクトモデルを用いた解法では、依然として、このような誤った処理が行なわれない。よってオブジェクトモデルによる本解法はこの点において、信頼性が高いといえる。このことは、モデルが複雑になればなるほど従来法との信頼性における差がますます大きくなるであろうことを示唆していると考えられる。別の見方をすると提案したオブジェクトモデルが変更に対して強く、従来法が変更に対して弱いということを暗示している。オブジェクト指向が持つ柔軟性の結果であると考えられる。

格子点をオブジェクトとする提案したオブジェクトモデリングが信頼性があり、熱を加えるような問題に対して変更が柔軟に行なわれ、従来法によるモデリングよりも精度の良いシミュレーションが行なわれることを示してきた。しかし、格子点オブジェクトが周囲の格子点オブジェクトから自己の状態を変容させ、対象領域全体で繰り返す処理を行なうには従来法と同様に連立方程式を解くのと同じような操作を必要とする。このため、複雑形状の流れを解析しようとする場合には、格子点オブジェクトが、メッセージ通信を行なう相手を変更しなければならないということを必然的に必要とする。これをオブジェクトごとに、いちいちリンクしていたのでは、非常に面倒な処理を強いられることになる。

本研究では領域をオブジェクトに取り、格子点オブジェクトを集約する形で、以上の問題解決を狙った。

領域クラスと格子点クラスが集約関係にあるということで、格子点オブジェクトレベルでのリンクは一度行なうだけで良く、複雑形状流れであっても領域オブジェクトを設定し、領域レベルでのメッセージ通信を行なうことで、容易にシミュレーションが行なわれることを示した。つまり、格子点オブジェクトが周囲のオブジェクトから自己の状態を変容し、これを全てのオブジェクトに対して繰り返すという操作を、領域オブジェクト内に隠蔽することにより、面倒なオブジェクト間のリンクずけが、最小限ですむようになった。オブジェクト指向のモジュール化を領域モデルと格子点モデルにおいて行なった結果であると考えられる。

従来法でモデリングを行なう場合は、格子点をリンクずける面倒な操作を何度も強いられる。加えて上述してきたように、物理量を表す変数が不正な関数を呼ぶ危険性も存在し、これらのことに注意してプログラムするのは容易でないことが予想される。その上、誤った実行をしていても気が付かないかもしれないし、プログラムにおいて、どこが間違ってるかを探すのは大変であると考えられる。

領域オブジェクトによるモデリングでは、既存の格子点オブジェクトモデルがそのまま利用でき、信頼性が損なわれることはなく、モデリングの精度は依然保証されているといえる。問題は最初の格子点オブジェクトを周囲のオブジェクトと関連ずけるときに誤りがあると、領域オブジェクトレベルでは発見できなくなるので、この点には十分気を付ける必要がある。しかし、そのことを加味したとしても従来法に比べては、精度が良いモデリングであるといえるであろう。領域をオブジェクトに取ることである程度の複雑な流れは容易に解けるようになったが、非常に複雑となると、インスタンスの数が膨大になり、領

域間の関連ずけが面倒になることが考えられる。しかし、この問題は複雑に入り組んだ領域だけを小さな領域オブジェクトにして、そうでない領域を大きな領域オブジェクトを割り当てるようにすれば解決され得ると考えられるが、今後の課題として残される。

6.2 信頼性の評価

ここでは提案したモデルの信頼が高いことを定量的に評価することにする。格子点をオブジェクトに取った差分解法、図 4.3 のオブジェクト図による解法と従来法との比較を行う。従来法との主な違いはクラス階層を用いるか否かであり、他の格子点間のリンクにかかる手間等は同様であり、これらの部分で誤った処理を行う確率は同じであるとし無視することにする。ここでは物理量を表す変数に対して正しい処理が行われる確率をもって、信頼性が高いとする。従ってモデル全体での信頼性ということにはならないと言える。提案した解法における x 方向の属性値 u と従来法の物理変数 U について取り上げる。

まず、提案したオブジェクトモデルでは境界クラスと境界内のクラスに分けられ、属性値の変更は自己のクラスのメソッドに対してのみ変更されるので、属性値 u に対して正しいメソッドが呼ばれる確率は、境界上、境界内で各々

$$\text{境界上 } u \text{ の確率} = \frac{1}{\text{クラス } Inner \quad U_{vel} \text{メソッド数}} = \frac{1}{4} = 0.25 \quad (6.1)$$

$$\text{境界内 } u \text{ の確率} = \frac{1}{\text{クラス } Outer \quad U_{vel} \text{メソッド数}} = \frac{1}{2} = 0.5 \quad (6.2)$$

となる。これに対して従来法ではオブジェクトモデルのようにクラス分けはなく、どのような式を用いても物理変数は変更され得る。ここで物理変数 U を変更する式の数を図 4.3 に示したオブジェクトモデルのメソッドの数に等しいと仮定する。従来法の物理変数 U に対して正しい関数が呼ばれる確率は

$$U\text{の確率} = \frac{1}{\text{全てのメソッド数}} = = \frac{1}{6+6+5} = \frac{1}{17} = 0.059 \quad (6.3)$$

となる。

以上より、提案したモデルにおいて、物理量を表す属性値を正しく変更する確率が、従来法の物理量を表す物理変数を正しく変更する確率に比べて高くなることを示された。オ

ブジェクトモデルが境界上の u に関して約 4.2 倍、境界内に関して約 8.5 倍信頼できることを示していると考えられる。このことは他の変数 v、p についても同様に導かれる。ここで述べているモデルに温度を追加したモデル図 4.8 では更に従来法に比較して信頼できることを示すと容易に予想される。

ここで述べた評価方法は、メソッドあるいは関数を同じ重みで選択することを仮定しており、実際のプログラミングでは何らかの思考が作用するのでこの方法は必ずしも正しいとは言えない。しかし、従来法において物理変数を変更する関数や式がオブジェクトモデルのメソッドの数とする仮定より、多くなるであろうことを加味し、オブジェクトモデルが自己のクラスのメソッドでのみ物理量を表す属性を変更することを考えると、誤った処理をする確率はオブジェクトモデルを用いた解法で少なくなるはずであると考えられる。ここでは既述の仮定に基づき、定量的に信頼性について評価を行った。

領域をオブジェクトに取ったモデルでは、従来法で、領域境界上で格子点レベルでのリンクずけを要するのに対し、オブジェクトモデルでは領域間のリンクであらわされるところが容易であり、従って誤る可能性が少なくより信頼できると考えられるが、定量的な評価は困難である。

以上、現段階では必ずしも定量的な評価はできていないが、提案したモデル (格子点オブジェクト、領域オブジェクト共に) が従来法に比べて信頼性が高いとしても問題はないと考えられる。

6.3 並列化への指標

格子点をオブジェクトにとるモデリングと領域をオブジェクトにとるモデリングを行なったが、並列化を行なおうとする場合には、各プロセッサでの処理が独立している必要があり、領域オブジェクトごとに各プロセッサに割り当てるのが、現実的であると考えられる。そこで、領域オブジェクトを取った場合についてどのように並列化が行なわれるかを検討する。

今回の領域オブジェクトを取ったモデリングでは、領域オブジェクト内の処理で、各格子点オブジェクトが最新の周囲の格子点の持つ物理量を用いて自己の状態を変容させることを行なっているが、この方法では、境界上のオブジェクトで領域間の処理が独立されず並列化ができない。解決法として、分割境界上のオブジェクトでは、メッセージ通信相手の１ステップ前の段階の物理量を用いることにすれば解決され得ると考えている。この変

更は、オブジェクトの属性に１ステップ前の物理量を持たせて、それを呼び出す関数を自己のクラスに持つことで容易に解決される。今回のモデリングでは収束性を考慮して最新の物理量を使うようにしている。この問題を解決すると、残された問題は、対象領域全体の収束を何ステップ間隔で行うかについてである。つまり、各領域ごとに収束させると、メッセージ通信の回数は減るが、領域全体での収束にかかるステップが多くなるであろうし、１ステップごとに収束を見れば通信量が増える代わりに全体での収束にかかるステップが減少するであろう問題をどのように扱うかである。これらの問題は課題として残される。

以上、並列化により、どの程度高速化が実現されるかは未知であるが、オブジェクトを領域に取ることで、比較的容易な並列化への移行がなされるであろうことを考察した。

第 7 章

あとがき

7.1 本研究で得られた成果

本研究で得られた成果としては、流れのシミュレーションを行なうためのオブジェクト指向に基づいた柔軟なモデルを提案したことである。以下の知見が得られた。

(1) 格子点オブジェクトを取ることで、概念から実装まで自然なモデル化の実現

(2) 格子点クラスから物理量の関係でサブクラスを導出することによる信頼性の高いモデルの構築

(3) 上記のモデルの熱を考慮した問題への柔軟な拡張

(4) 拡張されたモデルは、上記のモデルから再利用の恩恵を受け、従来法との比較において信頼性が向上

(5) 領域クラスと格子点クラスのモジュール化による複雑な処理を隠蔽

(6) 複数の領域オブジェクトによる複雑形状流れへ柔軟に対応

(7) 上記に関して局部的な加熱も可能

(8) 領域オブジェクトをプロセッサに割り当てれば並列化が容易に行なわれる可能性

(1) から (4) は格子点をオブジェクトに取ったことに関連している。

(1) は格子点オブジェクトが周囲のオブジェクトから、自己の物理量を変化させる概念がモデリングの実装段階においても実現されていることを示している。

(2) は MAC 法による離散化式をメソッドに持たせた場合のオブジェクトモデルによるモデリングが、物理量単位で細分化されているために不正な処理を行なわれる確立を低下させていることを述べている。従来法が物理量を表す変数の自由な変更による不正な処理を行なう危険性が高いのに対して、本モデルの信頼性が高いということを示している。

更に、畠山らの提案した格子点クラスに物理量すべての属性を持たせるモデルよりも、本モデルが信頼性において優れていることを同様に示した。同様に、速度オブジェクトから x、y 方向のサブクラスを導出することで、３次元問題への拡張容易性についても示唆した。

(3) は上述モデルが熱問題へ拡張する場合にも、柔軟に拡張でき、モデルの信頼性は損なわれないということを示している。従来法では、物理量の数と方程式の数が増えると信頼性が低下する。このため上述のオブジェクトモデルの派生クラスが多くなればなるほど、従来法に比べて信頼性が増加するということを述べてきた。このことは同時にプログラミングが容易に行なわれることも意味していると考えられる。

(4) は (3) に関連して、オブジェクトモデルが拡張に際して、オブジェクト指向のメリットである再利用の恩恵を拡張モデルが受けるということを意味している。

(5) 以降は領域オブジェクトを考える上でのメリットに関する知見である。

(5) は、領域クラスから格子点クラスを集約としてモデル化することで、複雑な処理、つまり格子点オブジェクト間の関係をリンクするという操作にかかる手間が軽減されるということを意味している。(6) は、領域オブジェクトから内部の格子点オブジェクトが行なう処理が隠蔽される為に、格子点オブジェクトに関するリンクを一度行なうだけでよく、後は領域間のメッセージ交換により、比較的複雑形状の流れであっても容易にシミュレーションできるということを意味している。

(7) については、(5)、(6) の場合と同様に、領域オブジェクトモデリングに拡張された格子点オブジェクトモデリングの概念が含まれている為に、流れに対して柔軟に、熱の影響を加えることができるということである。

(8) の並列化については、領域オブジェクトごとの処理を各プロセッサに割り当てることで比較的容易に達成されると考えられる。プロセッサには一つの領域を割り当てると決まっているわけではないので、負荷を均等にするために複数の領域オブジェクトを割り当

てるようなことも考慮しなけばならないと考えられる。課題である。ここでは並列化のための一つの指針を示したといえる。

以上から、本研究では、従来法で問題であった複雑流れの解析、プログラム構築、変更に伴う問題が改善され、既存のモデルによる問題解析から、別の問題の解析への移行が柔軟に行なわれる一つのアプローチを提案したといえる。

7.2 課題

今後の課題として、以下のことがあげられる。

本研究では一つの柔軟なオブジェクトモデルを提案したが、より複雑で動的に変化するような移動境界問題や圧縮性流体のような流れに対しては、シミュレーションを行なうのは困難であり、他のオブジェクトを取ることによる異なったモデリングについて検討する必要があると考えられる。これらの問題を解決できれば、ほとんどの問題が容易に解析を行なえるということになり、更にオブジェクト指向に基づく実行支援環境が構築できれば、より柔軟なシミュレーションを容易に実行できると期待される。

また、解法の高速化のため、並列化についても考慮する必要があると考えられる。

謝辞

本研究を進めるにあたり、貴重な御助言、御指導を賜わりました松澤照男教授に深く感謝致します。最後に２年間お世話になった研究室の皆様、その他御迷惑をお掛けした皆様に深く感謝致します。

参考文献

[1] 保原充、大宮司久明偏、数値流体力学、東京大学出版会、pp1-6、pp16-20、1992.

[2] 畠山正行、金子勇、オブジェクトベース機構に基づく数値シミュレーション情報処理学会第 51 回ハイパフォーマンスコンピューティング研究会、pp5-13、vol.1、No.2、1996

[3] 畠山正行、横澤譲二、オブジェクト指向に基づく差分解析法とその実現例第７回数値流体力学シンポジウム講演論文集、pp515-518、1993.

[4] 畠山正行、金子勇、オブジェクトベース機構：オブジェクト指向一貫モデリング過程論に基づくシミュレーションの実現、情報処理学会第 17 回情報処理学会プログラミング研究会、pp33-44、vol.1、No.2、1994

[5] J. ランボー他著、羽生田栄一監訳、オブジェクト指向方法論-モデル化と設計、トッパン、pp25、1992.

[6] 三木光範、中井正一、オブジェクト指向とエージェント指向、計算工学、日本計算工学会、pp5-13、vol.1、No.2、1996

[7] 荒川忠一著、数値流体工学、東京大学出版会、pp78,1994.

[8] 高橋亮一著、応用数値解析、朝倉書店、pp31、1993.

[9] 八田夏夫著、流れの計算、森北出版株式会社、pp94-97、1994.

[10] 坂元美定、功刀資彰、一宮浩一、下壁加熱・上壁冷却の水平正方形管内複合対流熱伝達に関する３次元数値解析、第９回数値流体力学シンポジウム、pp309-310、1995.

[11] 上田 隆宏、松澤照男、非圧縮性流れ解析におけるオブジェクト指向の応用、第１０回数値流体力学シンポジウム、pp368-369、1996.