オブジェクト指向の数値流体力学への応用

Application of Object Oriented Programming to a Computational Fluid Dynamics

猪瀬　貴茂，茨城大学大学院理工学研究科，茨城県日立市中成沢町 4-12-1，inose@aries.dse.ibaraki.ac.jp
石黒　美佐子，茨城大学工学部，茨城県日立市中成沢町 4-12-1，ishiguro@ipc.ibaraki.ac.jp
Takashige Inose, Graduate School of Science and Engineering, Ibaraki University, 316-8511 Japan
Misako Ishiguro, Faculty of Engineering, Ibaraki University, 316-8511 Japan

Object oriented programming is applied to a simulation of computational fluid dynamics. A flow domain which characterizes each part of flow field is treated as an object and calculated independently having the relation with neighbor objects. We can compose a total flow field by combining the flow domains as if combining parts of puzzles. The CFD system provides for users to generate various flow fields easily and to analyze the over all flow automatically.

１．はじめに

近年コンピュータの性能の向上にはめざましいものがあり，数年前のスーパーコンピュータの性能を凌ぐ計算機環境をパーソナルユーザが持てるようになっている．また，マルチ CPU のコンピュータや PC クラスタ(1)などにより，安価に並列計算環境などを構築することが可能になってきた．それに伴い，CFD（計算流体力学）などに代表される科学技術計算などの CPU パワーを使うような数値シミュレーションを限られた人だけでなく，一般的なユーザが行うことが可能になってきた．しかしながら，現在科学技術計算のシミュレーションは専門性が強く少数のユーザしか利用することが出来ていない．

従来，科学技術計算の分野では，主に手続き型言語である Fortran や C などを用いてシミュレーションが行われてきた．しかし，近年ではオブジェクト指向言語である C++や Java などによる科学技術計算のシミュレーションも散見されるようになってきた(2) (3)．シミュレーションは現実世界の現象をコンピュータ上でモデル化し，数値計算の手法を用いてその現象に関する結果を推定するものである．オブジェクト指向プログラミング(OOP)もまた，現実世界の現象をオブジェクトとしてコンピュータ上に直接表現する．したがって，両者の考え方が非常に似ているのでシミュレーションに OOP は向いていると考えられる．

さらに，OOP は従来のプログラミングスタイルに比べて、複雑さを押さえ開発効率が高く，動的結合などによる優れた柔軟性などの特徴をもっている．また，従来の並列計算と異なり，オブジェクトにより並列化を行う手法も考えられている．

そこで，本研究では CFD のシミュレーションに OOP を適用する．そして，ユーザが CFD のシミュレーションを簡単に行えるようなシステムを開発する．

２．システムの概要

本研究では，従来のようなに解析を行いたい流れ場を領域分割法などにより計算を行うのではない．OOP を適用することで，流れ場を構成するための部品として，流入，流出，扇形や矩形などの流れ場領域を作成する．そして，解析を行いたい流れ場の形状に流れ場オブジェクトを組み合わせることで構築しシミュレーションを行う．

さらに，Java を用いてシステム開発することで，アプレットや GUI を利用し，流れ場構成領域上に流入，流出，扇形や矩形などの流れ場オブジェクトを置き，簡単に解析を行う流れ場を構成出来るようにし，シミュレーションを行えるようにする．(Fig.1)

１つの流れ場領域は流速 u, v 圧力 p というデータ，流れ場の計算というメソッドを持つオブジェクトと定義する．流れ場領域をオブジェクトとして扱うことによって，それぞれの独立性を高めることができる．そして，流れ場領域を一つの部品のように扱うことが可能になる．

流れ場の解析には HSMAC 法(4) (5)を使用する．予測速度を計算後，連続の式を満たすように反復計算によって実速度を計算する．反復後に流れ場領域オブジェクトの境界条件を設定する．

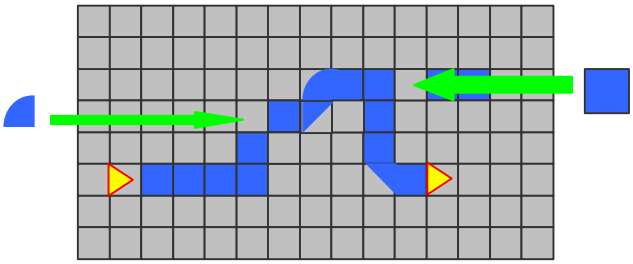


Fig. 1 Flow field.

流れ場領域オブジェクト(FlowParts；Fig. 2)は，拡張性と柔軟性をもたせるために流れ場の計算部分と境界条件をべつに Calculation オブジェクトと BoundaryCondition オブジェクトとして定義する．これらを別のオブジェクトとすることで，流れ場領域オブジェクトを異なる形の流れ場に拡張する場合，それぞれの親クラスを継承して，新しい Calculation オブジェクトと BoundaryCondition オブジェクトを作ることができる．そして，流れ場領域オブジェクトに組み込むことによって今までと異なる新しい流れ場オブジェクトにすることが出来る．

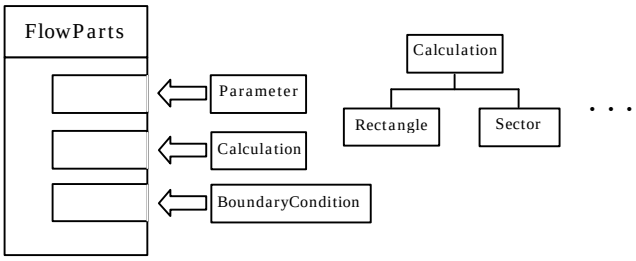


Fig. 2 Flow parts object .

ユーザはGUIを用いて流れ場領域オブジェクトを生成し，それらを流れ場設定領域に配置し，組み合わせ流れ場を構成する事が出来る．これにより，流れ場を簡単に構成できる．また，この流れ場領域オブジェクトは，自分が設置された位置から隣接する領域の状態を知ることができる，そこから，オブジェクト自身が境界条件の設定をすることが出来る．従って，ユーザはCFDについて詳しい知識を有していなくても，流れ場領域オブジェクトをただ置くだけで流れ場のシミュレーションを行うことも可能になる．

３．計算モデルの概要

流れ場の計算には，基礎方程式として連続の式とナビエ・ストークス方程式を用いた．

矩形の流れ場の計算には，

連続の式

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0 \tag{1}$$

ナビエ・ストークス方程式

$$\frac{\partial u}{\partial t}+\frac{\partial u^2}{\partial x}+\frac{\partial uv}{\partial y}=X-\frac{1}{r}\frac{\partial p}{\partial x}+n\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right) \tag{2}$$

$$\frac{\partial v}{\partial t}+\frac{\partial uv}{\partial x}+\frac{\partial v^2}{\partial y}=Y-\frac{1}{r}\frac{\partial p}{\partial y}+n\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right) \quad . \tag{3}$$

扇形の流れ場の計算には，ｒ-θ座標系で記述された連続の式

$$\frac{\partial (v_r r)}{\partial r}+\frac{\partial v_q}{\partial q}=0 \tag{4}$$

ｒ-θ座標系で記述されたナビエ・ストークス方程式

$$\frac{\partial v_q}{\partial t}+\frac{1}{r}\frac{\partial v_q^2}{\partial q}+\frac{\partial v_q v_r}{\partial r}+\frac{2v_r v_q}{r}$$
$$=\Theta-\frac{1}{r}\frac{\partial p}{r\partial q}+n\left(\frac{\partial^2 v_q}{\partial r^2}+\frac{1}{r}\frac{\partial v_q}{\partial r}+\frac{1}{r^2}\frac{\partial^2 v_q}{\partial q^2}+\frac{2}{r^2}\frac{\partial v_r}{\partial q}-\frac{v_q}{r^2}\right) \tag{5}$$

$$\frac{\partial v_r}{\partial t}+\frac{1}{r}\frac{\partial v_r v_q}{\partial q}+\frac{\partial v_r^2}{\partial r}+\frac{v_r^2-v_q^2}{r}$$
$$=R-\frac{1}{r}\frac{\partial p}{\partial r}+n\left(\frac{1}{r^2}\frac{\partial^2 v_r}{\partial q^2}+\frac{\partial^2 v_r}{\partial r^2}+\frac{1}{r}\frac{\partial v_r}{\partial r}-\frac{2}{r^2}\frac{\partial v_q}{\partial q}-\frac{v_r}{r^2}\right) \tag{6}$$

流速u, vや圧力pはスタガードメッシュを用いて定義する．これらの基礎方程式を差分法[6]により離散化する．圧力勾配項と拡散項は中心差分，移流項は風上差分を用いた．HSMAC法で解くことで，連続の式を満たすように速度と圧力を修正し流れ場を求める．

４．オブジェクト指向とJava

４．１　オブジェクト指向[7]

オブジェクト指向（Object Oriented）とは，実世界の事柄や仮想的な事象の1つを対象にしてモジュール化するパラダイムである．このパラダイムに基づけばソフトウェアの再利用性や可読性を高めることが出来る．

オブジェクト指向では，オブジェクトとそれに対して送られるメッセージだけで世の中の動きを表現する．

オブジェクトは，実体を特徴づけるいくつかの属性と，その実体が役割を果たすためにいくつかの操作の定義によって成り立つ．メッセージとは，オブジェクトが持つ機能を動作させるための要求のことである．

１つのシステムは複数のオブジェクトから構成され，メッセージが次々に送られていくことにより相互に作用し計算が行われる．

オブジェクト指向は，オブジェクト間で役割を分担することによるモジュール化を基礎とし，データ抽象化，継承，動的結合などを備えている．

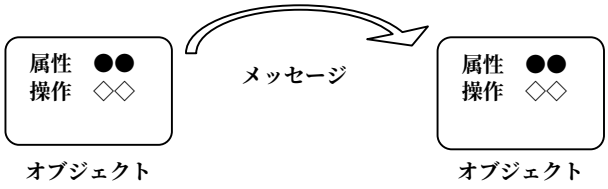


Fig. 3 Object and message.

・データ抽象化

プログラムとして表現する対象の中からまとまりのある実体をみつけ，その実体に本質的に備わるべきインターフェースを定めることである．本質的な機能ではないインターフェースをあえて除外することによって，プログラム構造が単純化され扱いやすくなる．

複雑になりがちなプログラムの可読性や再利用性を高めるために有効な手法である．うまくデータ抽象化を行えば生産性も向上するし，保守性も向上する．

・継承

すでに定義したクラスの機能を受け継いで新しいクラスを定義することである．継承の機能を使うと過去に作ったモジュールの機能を追加したい場合や，部分的に修正したい場合に，変更部分を書き換えるだけで新たなモジュールを作ることができる．再利用できる可能性が格段に高くなる．

・動的結合

プログラム中の識別子に対応するクラスを実行時に決めることである．特に動的なメソッド結合とは，メッセージを受け取るオブジェクトが，そのメッセージに応じて起動するメソッドを決めることである．動的結合を利用することでプログラムに柔軟性を持たせることができる．

４．２　Java

本システムは，Javaを用いて構築した．以下にJava言語の特徴[8]と用いた理由を挙げる．

- オブジェクト指向を完璧にサポート
- コンパイラ＋インタプリタ型言語
- コンピュータ（プラットフォーム）に非依存
- アプレットによるブラウザ上での実行
- C / C++言語に似た構文を採用
- 強力なガーベッジ・コレクション機能
- 強力な例外処理機能
- 多重継承はインターフェース機能でサポート
- マルチスレッドのサポート
- ２次元グラフィクスライブラリを持つ
- 最も急速に成長している言語

プラットフォームに依存せず，アプレットなどによりブラウザ上で実行できることは，一般的なユーザがCFDのシミュレーションを行ううえで非常に便利である．また，マルチスレッドによる並行処理のサポートは，将来のオブジェクト並列化による並列計算に拡張しやすい．

５．オブジェクト指向プログラミングのCFDへの適用

５．１　流れ場領域の構築

本研究では，OOPをCFDに応用することによって，流れ解析のシミュレーションを容易に行えるようなシステムを

構築する．まず，流入，流出，扇形や矩形などといった特徴を持つ流れ場領域をオブジェクト化する．それぞれの流れ場オブジェクトは流速 u, v 圧力 p というデータ，流れ場の計算というメソッドを持つオブジェクトと定義する．流れ場領域をオブジェクトとして扱うことによって，それぞれの独立性を高めることができ，流れ場という特徴を持ったパズルの破片のように扱うことができる．オブジェクトを Fig.4 のような流れ場構成領域に配置することによって，解析を行いたい流れ場をパズルを組み合わせるように構築することができるようになる．

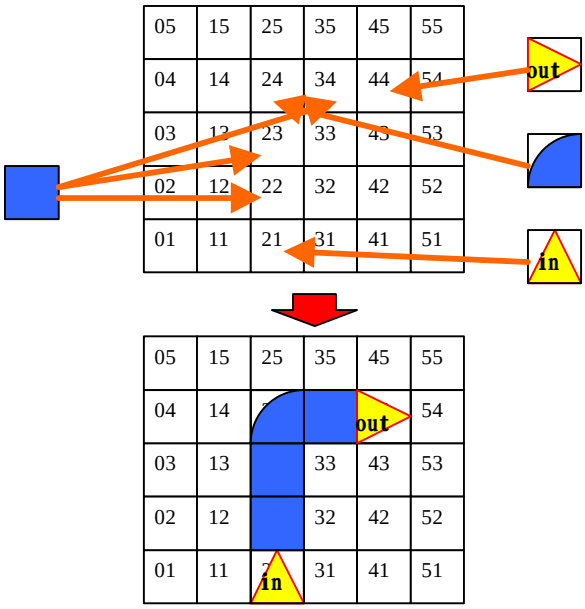


Fig. 4 Flow set field.

また，この流れ場パーツオブジェクトは，自分が置かれた位置を流れ場構成領域から番号を取得することで判別する．そして，隣接する領域にどのような流れ場が設定されているのかを判断する．そこから，流れ場オブジェクトは上下左右の境界条件の設定を自分で行う．従って，CFD について詳しい知識を有していなくても，パーツオブジェクトをただ置くだけで流れ場のシミュレーションを行うことが可能になる．

さらにオブジェクト化の利点として，計算を行いたいような流れ場のパーツが存在しない場合は，現在ある流れ場クラスを継承して新しいクラスを簡単に作ることができ，異なる流れ場への拡張もし易い．

５．２　クラスの設計

上記のようなシステムを構築するため，以下のようなクラスを設計する．

・FlowParts クラス

流れ場を構成する流れ場オブジェクトを生成するクラス．流れ場オブジェクトは，計算や境界条件の設定などの機能を内部に別のオブジェクトを生成することにより定義する．

異なる形状に流れ場を変更する場合，FlowPats の中で定義された機能を変更する必要がある．このとき，機能を内部の別のオブジェクトとして定義することによって，動的に変更することができ簡単化することができる．また，新しい形状の流れ場を追加したい場合，初めから FlowParts を定義する必要はない．FlowPats オブジェクトの機能を定義するクラスを継承し，新しい形状に対応した内部のオブジェクトを定義し直すだけで良い．FlowParts クラスは Fig. 5 のように構成される．また， Table 1 のようなメソッドを持つ．

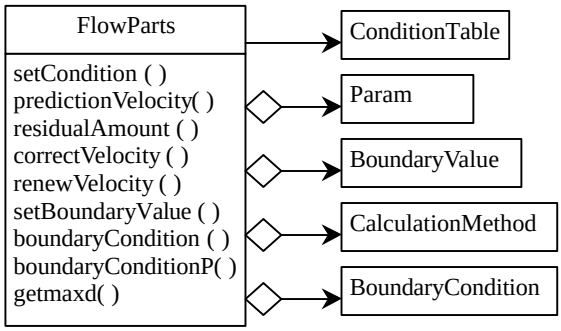


Fig. 5　FlowParts class.

Table 1 FlowParts method.

setCondition( )	流れ場の形状を決めて，それに応じた CalculationMethod と BoundaryCondition を設定する．
prediction Velocity( )	CalculationMethod オブジェクトから predictionVelocity( )メソッドを呼び出し，予測速度を計算する．
residual Amount( )	CalculationMethod オブジェクトから residualAmount ( )メソッドを呼び出し，残差を計算する．
correct Velocity( )	CalculationMethod オブジェクトから correctVelocity( )メソッドを呼び出し，速度の修正をする．
renew Velocity( )	CalculationMethod オブジェクトから renewVelocity ( )メソッドを呼び出し，速度を更新する．
setBoundary Value( )	BoundaryCondition オブジェクトから setBoundaryValue( )メソッドを呼び出し，Boundary Value に境界値の値を渡す．
boundary Condition ( )	BoundaryCondition オブジェクトから boundaryCondition ( )メソッドを呼び出し，境界条件を設定する．
boundary ConditionP( )	BoundaryCondition オブジェクトから boundaryConditionP( )メソッドを呼び出し境界条件を設定する．
getmaxd( )	残差の値を渡す．

・ConditionTable クラス

隣接する流れ場オブジェクトの状態を保存しておくデータクラス．今回のシステムでは，流れ場オブジェクトは流れ場構成領域に配置された後自分自身で境界条件を設定する．そのためには，周囲に配置されたオブジェクトが何かを知る必要がある．そこで，流れ場構成領域のどこに，どのような流れ場オブジェクトが配置されているかという情報を ConditionTable オブジェクトに格納しておく．それらを流れ場オブジェクトが参照することで周囲の状態を把握し境界条件を設定することができる．

・Parameter クラス

計算に使うパラメータの設定を行うクラス．数値計算上に必要な情報を定義する．流れ場オブジェクトで共通の刻み時間Δt，空間分割幅Δx，Δy，Δr，Δθ，流体の密度ρ，流

体の動粘性係数υ，収束判定値 dmin，などの物理パラメータを定義する．また，流速 u, v, 予測流速 pu, pv，圧力 p などの物理量の配列を定義する．

・CalculationMethod クラス

流れ場オブジェクトでの流れの計算を行う部分を CalculationMethod クラスによって定義する．流れ場オブジェクトは形状によって計算式が異なるため，それぞれ異なるメソッドが必要となる．流れ場の形状を増やすとメソッドも増えプログラムが複雑になってしまうため，CalculationMethod クラスを継承して矩形や扇形などの流れ場に対応した計算手法を持つ RectangleCalculation クラスや，SectorCalculation クラス(Fig.6)など計算手法クラスを作る．これにより，メソッドの種類を増やすことなく計算手法クラスを動的に変更することによって異なる形状の流れ場の計算が行える．CalculationMethod クラスは Table 2 のようなメソッドを持つ．

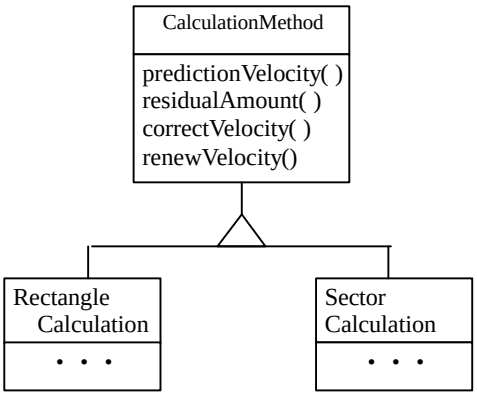


Fig. 6 CalculationMethod class.

Table 2 Calculation method

predictionVelocity ( )	予測流速の計算を行う．
residualAmount ( )	残差の計算を行う
correctVelocity ( )	速度の修正し実流速を求める．
renewVelocity ( )	速度の更新を行う．

・BoundaryValue クラス

流れ場オブジェクト間のデータの受け渡しを行う上でのバッファの役目をするクラス．流れ場オブジェクトは内部で差分法により流速や圧力の計算をしているため，流れ場オブジェクトの境界部分では周囲の流速 u, v, 圧力 p の値が必要である．そこで，流れ場オブジェクトは隣接する境界部分の流速と圧力の値を受け渡ししなくてはならない．

矩形の流れ場オブジェクトでは，流速や圧力の配列の取り方は同一なので流れ場オブジェクト間での境界値の受け渡しは容易に行うことが可能である．しかし，矩形と扇形，または，その他の形状をした流れ場オブジェクトのように異なる形状を組み合わせる場合，計算の仕方が異なるため配列の取り方などが異なっている．この場合，流れ場オブジェクトを組み合わせて計算することが非常に面倒になる．そこで，計算手法や，配列の取り方などに関係しないような値の受け渡しが必要になる．

流れ場オブジェクトで受け渡しが必要な値を BoundaryValue オブジェクトに決まった形で一度格納する．BoundaryValue オブジェクトどうしを受け渡しすることで，異なる形状の流れ場オブジェクト間でのデータの受け渡しもすべて同じ方法で行うことが可能になる(Fig.9)．BoundaryValue クラスは，Table 3 のようなメソッドを持つ．

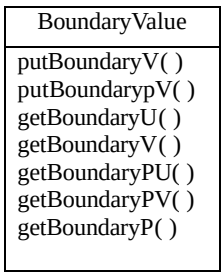


Fig. 7 BoundaryValue Class.

Table 3 BoundaryValue method.

putBoundaryV( )	境界値 u, v, p を流れ場オブジェクトに渡す．
putBoundarypV( )	予測速度の境界値 pu, pv, p を流れ場オブジェクト渡す．
getBoundaryU( )	流れ場オブジェクトから流速 u を受け取る．
getBoundaryV( )	流れ場オブジェクトから流速 v を受け取る．
getBoundaryPU( )	流れ場オブジェクトから予測速度 pu を受け取る．
getBoundaryPV( )	流れ場オブジェクトから予測速度 pv を受け取る．
getBoundaryP( )	流れ場オブジェクトから圧力 p を受け取る．

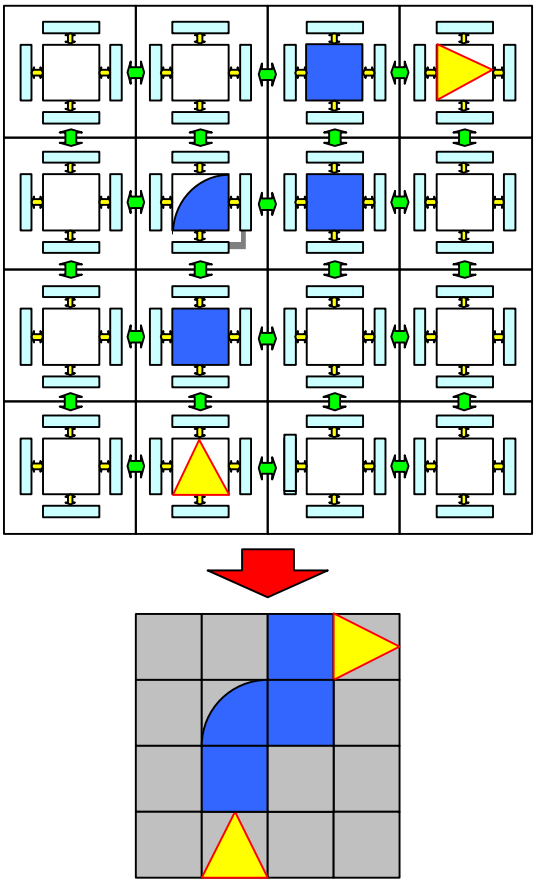


Fig. 9 BoundaryValue image.

・BoundaryCondition クラス

流れ場オブジェクトに境界条件を設定するクラス．オブジェクトの上下左右には壁面，流入，流出，オブジェクトの接続という境界条件を考える．

壁面の境界条件はすべりなし条件とする．壁面に平行な流れおよび垂直な流れの双方ともゼロ（u=0, v=0）とする．圧力の境界条件は，法線方向の流速に変化を与えないことから $\partial P/\partial y = 0$ または $\partial P/\partial x = 0$ となる．

流入の境界条件は，境界上の流速成分を指定したい値にする（u=a, v=b）．圧力は，壁面同様 $\partial P/\partial y = 0$ または $\partial P/\partial x = 0$ とする．

流出の境界条件は，流速は $\partial u/\partial x = 0$， $\partial v/\partial y = 0$ とする．圧力は， $\partial P/\partial y = 0$， $\partial P/\partial x = 0$ とする．

流れ場境界オブジェクトどうしを接続するものとして接続条件を設定する．接続は BoundaryValue クラスで生成したオブジェクトを流れ場オブジェクト間でやりとりする（Fig. 10）．

境界条件の設定には BoundaryCondition クラスを継承し，流れ場の形状に合わせた境界条件クラスを作る（Fig. 11）．BoundaryCondition クラスは，Table 3 のようなメソッドを持つ．

流れ場オブジェクトの接続時には，オブジェクト間で境界値を統一的な方法で受け渡しができるようにするため，境界値を BoundaryValue オブジェクトわたす．このとき流れ場の形状によって配列の取り方が異なっているため setBoundaryValue メソッドを流れ場ごとに変え BoundaryValue オブジェクトにわたす．

流れ場オブジェクトの境界値は，boundaryCondition メソッドによって，隣接する流れ場オブジェクトの BoundaryValue オブジェクトをもらうことによって設定する．

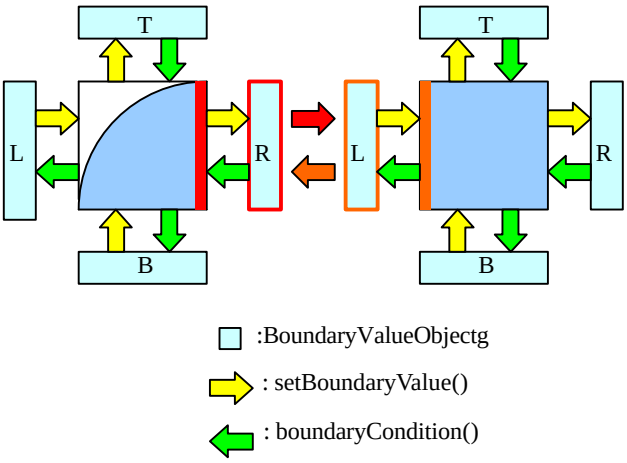
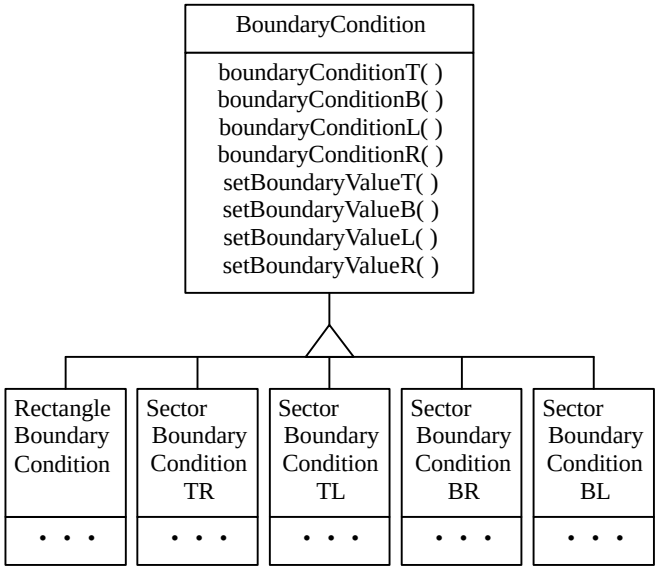


Fig. 10 Exchange of boundary value.



矩形：RectangleBoundaryCondition
右上の４分の１円：SectorBoundaryConditionTR
左上の４分の１円：SectorBoundaryConditionTL
右下の４分の１円：SectorBoundaryConditionBR
左下の４分の１円：SectorBoundaryConditionBL

Fig. 11 BoundaryCondition class.

Table 3 BoundaryValue method

boundaryCondition( ) (T, B, L, R)	流れ場オブジェクトの境界条件を設定する．このとき，条件が壁面，流入，流出，接続という条件は ConditionTable オブジェクトからメッセージを受け取り判断する．
setBoundaryValue( ) (T, B, L, R)	流れ場オブジェクトの境界値を BoundaryValue オブジェクトに渡す．

・Simulation クラス

流れ場オブジェクトを生成し，流れ場オブジェクト間の同期をとりながら計算を行うクラス．HSMAC 法を用いた計算の流れを Fig. 12 に示す．

boundaryCondtion()メソッドで流れ場オブジェクト間での境界値を BoundaryValue オブジェクトを用いて受け渡しをしている．計算の結果，流速ベクトルの可視化も行う．

```
public void main() {
  int i, m, fl;
  // 初期条件の設定
  for( i = 0; i < pmax; i++)
    f[i].setBoundaryValue();
  for( i = 0; i < pmax; i++)
    f[i].boundaryCondition( f);
```

```
  // タイムステップループ
  for ( int n = 0; n < Nstep; n++) {
    // 予測流速の計算
    for( i = 0; i < pmax; i++) {
      f[i].predictionVelocity();
      f[i].setBoundaryValue();
    }
    // 圧力の修正反復
    for( m = 1; m <= Mstep; m++) {
      // 残差の計算
      for( i = 0; i < pmax; i++) {
        f[i].boundaryConditionP( f);
        f[i].residualAmount();
      }
      // 連続の式の判定
      fl = 0;
      for( i = 0; i < pmax; i++)
        if( f[i].getmaxd() < DMIN) fl++;
      // 速度・圧力の更新
      if( fl == Omax || m == Mstep) {
        for( i = 0; i < pmax; i++)
          f[i].renewVelocity();
        break;
      }
      // 速度・圧力の修正
      else {
        for( i = 0; i < pmax; i++) {
          f[i].boundaryConditionP( f);
          f[i].correctVelocity();
          f[i].setBoundaryValue();
        }
      }
      for( i = 0; i < pmax; i++)
        f[i].setBoundaryValue();
      for( i = 0; i < pmax; i++)
        f[i].boundaryCondition( f);
} } }
```

Fig. 12 .Main part of Simulation class.

6．ユーザ・インターフェース

本システムではユーザが簡単にシミュレーションを行えるようにするためにユーザ・インタフェースには GUI（Graphical User Interface）を用いる．GUI では，ユーザに対する情報の表示をボタンやラベル，メニュー項目，リストボックス，ツールバーなどグラフィックを多用する．それにより，大半の基礎的な操作をマウスなどのポインティングデバイスによって行うことができる． GUI を用いることで，解析したい流れ場を視覚的に構成することができ，簡単にシミュレーションの準備を行うことができる．

Java では，ユーザ･インターフェースとして．AWT（Abstract Window Toolkit）[9]や JFC（Java Foundation Class）に含まれる Swing[10]などにより GUI を強力にサポートしている．

Fig. 13 に GUI よる流れ場の設定を示す．図の右側のチョイスコントロールから配置したい流れ場オブジェクトを選び，左側にある流れ場構成領域に配置することで流れ場を構成し，スタートボタンを押すことでシミュレーションを開始する．

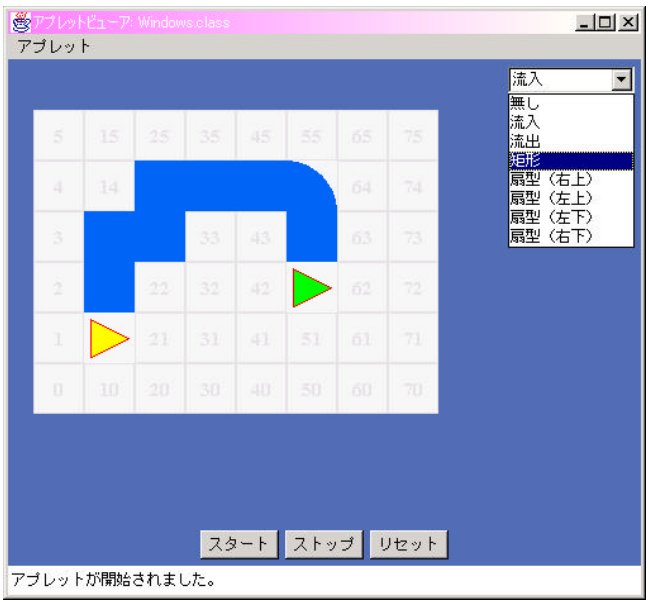


Fig. 13 Graphical user interface.

7．解析事例

解析事例を Fig.14 に８つのオブジェクトで構成されたパイプ内の流れを示す．Fig.15 に４つのオブジェクトで構成されたキャビティー流れを示す．このとき，レイノルズ数 100．

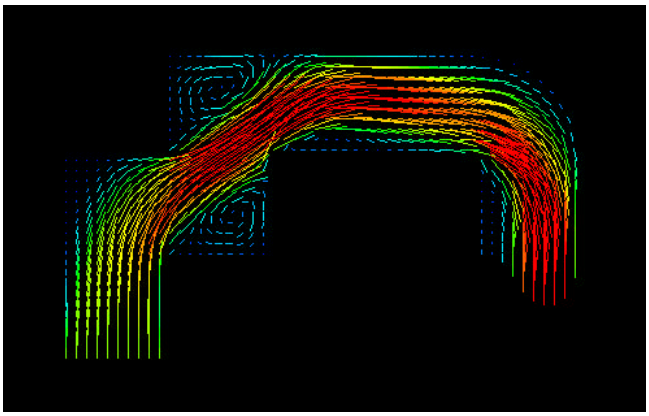


Fig. 14 Example 1.

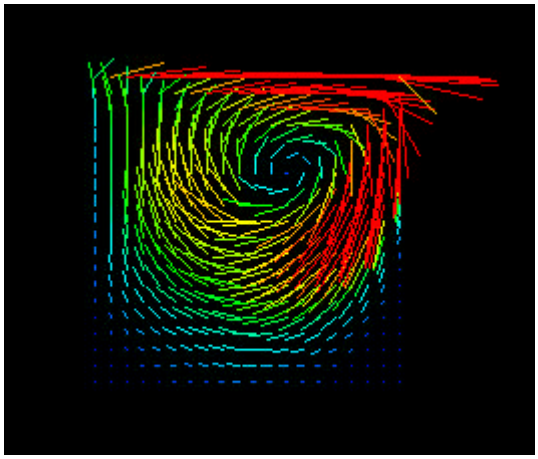


Fig. 15 Example 2.

８．まとめ

オブジェクト指向を CFD に応用するために，流れ場を矩形，扇形といった形状ごとにオブジェクトとして設計した．また，それらを組み合わせることで，CFD の知識を持たない一般的なユーザが簡単にシミュレーションを行えるシステムを提案した．

流れ場オブジェクトは，基礎方程式の計算や境界条件の設定などの機能ごとにモジュール化し，それらをインスタンス変数として持たせた．流れ場の形状の変化を，内部オブジェクトを動的に変更することにより，流れ場オブジェクト自身の性質を変えられるようにした．　異なった形状の流れ場オブジェクト間でも境界値の受け渡しを簡単に行えるように，BoundaryValue オブジェクトを定義し，それを受け渡しすることにより，統一的な方法で行えるようにした．GUI を用いることにより，流れ場オブジェクトを配置するだけでシミュレーションができるようにした．今後は，他の計算方法にも応用できるように，FlowParts オブジェクトのインターフェースを拡張したい．

参考文献

(1) 野澤 恵, “PC Cluster とは何か？”, 第 13 回数値流体力学講演論文集, 特別企画 1(1999), pp. 4-6.

(2) 上原 均, “オブジェクト指向シミュレーションシステムのアプリケーション・パターン”, 電子情報通信学会論文誌 Vol.J82-D-I(1999), pp1-13.

(3) 城之内 忠正, 千葉 賢, “流れ計算 Java コンポーネント：flowBeans, ”日本流体力学年会’98 講演論文集(1999), pp. 419-422

(4) 標 宣男, 鈴木 正昭, 石黒 美佐子, 寺坂 晴夫, “数値流体力学－複雑流れモデルと数値解析－”, 朝倉書店,1994.

(5) 棚橋 隆彦, “非圧縮粘性流体の過渡流れ(1)”, 機械の研究, 第 37 巻，第３号および第４号, (1985), pp.383-388.

(6) 高橋 亮一, “応用数値解析”, 朝倉書店,1993.

(7) 戸松 豊和, “Java プログラムデザイン”, ソフトバンク,1989.

(8) Yoo Hong Jun, ”図解 Java 流オブジェクト指向入門”, 技術評論社,1997.

(9) Steven Holzner, “テクニック・ライブラリシリーズ Java2”, IDG コミュニケーションズ,1999.

(10) Steven Holzner, “Java Swing プログラミング Black Book”, インプレスコミュニケーションズ,2000.