OpenFOAM

オープンソース CFD ツールボックス

ユーザガイド和訳

Version 2.3.0

2014 年 3 月 14 日

一般社団法人 オープン CAE 学会

このユーザガイド和訳は，一般社団法人 オープン CAE 学会の責任のもとで公開しております．本書に関するご意見等がございましたら，当学会事務局 (office@opencae.jp) までご連絡ください．

一般社団法人 オープン CAE 学会

Typeset in pLaTeX.

原文著作権表示

License

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED. BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. TO THE EXTENT THIS LICENSE MAY BE CONSIDERED TO BE A CONTRACT, THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. Definitions

a. "Adaptation" means a work based upon the Work, or upon the Work and other preexisting works, such as a translation, adaptation, derivative work, arrangement of music or other alterations of a literary or artistic work, or phonogram or performance and includes cinematographic adaptations or any other form in which the Work may be recast, transformed, or adapted including in any form recognizably derived from the original, except that a work that constitutes a Collection will not be considered an Adaptation for the purpose of this License. For the avoidance of doubt, where the Work is a musical work, performance or phonogram, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered an Adaptation for the purpose of this License.

b. "Collection" means a collection of literary or artistic works, such as encyclopedias and anthologies, or performances, phonograms or broadcasts, or other works or subject matter other than works listed in Section 1(f) below, which, by reason of the selection and arrangement of their contents, constitute intellectual creations, in which the Work is included in its entirety in unmodified form along with one or more other contributions, each constituting separate and independent works in themselves, which together are assembled into a collective whole. A work that constitutes a Collection will not be considered an Adaptation (as defined above) for the purposes of this License.

c. "Distribute" means to make available to the public the original and copies of the Work through sale or other transfer of ownership.

d. "Licensor" means the individual, individuals, entity or entities that offer(s) the Work

under the terms of this License.

e. “Original Author” means, in the case of a literary or artistic work, the individual, individuals, entity or entities who created the Work or if no individual or entity can be identified, the publisher; and in addition (i) in the case of a performance the actors, singers, musicians, dancers, and other persons who act, sing, deliver, declaim, play in, interpret or otherwise perform literary or artistic works or expressions of folklore; (ii) in the case of a phonogram the producer being the person or legal entity who first fixes the sounds of a performance or other sounds; and, (iii) in the case of broadcasts, the organization that transmits the broadcast.

f. “Work” means the literary and/or artistic work offered under the terms of this License including without limitation any production in the literary, scientific and artistic domain, whatever may be the mode or form of its expression including digital form, such as a book, pamphlet and other writing; a lecture, address, sermon or other work of the same nature; a dramatic or dramatico-musical work; a choreographic work or entertainment in dumb show; a musical composition with or without words; a cinematographic work to which are assimilated works expressed by a process analogous to cinematography; a work of drawing, painting, architecture, sculpture, engraving or lithography; a photographic work to which are assimilated works expressed by a process analogous to photography; a work of applied art; an illustration, map, plan, sketch or three-dimensional work relative to geography, topography, architecture or science; a performance; a broadcast; a phonogram; a compilation of data to the extent it is protected as a copyrightable work; or a work performed by a variety or circus performer to the extent it is not otherwise considered a literary or artistic work.

g. “You” means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.

h. “Publicly Perform” means to perform public recitations of the Work and to communicate to the public those public recitations, by any means or process, including by wire or wireless means or public digital performances; to make available to the public Works in such a way that members of the public may access these Works from a place and at a place individually chosen by them; to perform the Work to the public by any means or process and the communication to the public of the performances of the Work, including by public digital performance; to broadcast and rebroadcast the Work by any means including signs, sounds or images.

i. “Reproduce” means to make copies of the Work by any means including without limitation by sound or visual recordings and the right of fixation and reproducing fixations of the Work, including storage of a protected performance or phonogram in digital form or other electronic medium.

2. Fair Dealing Rights.

Nothing in this License is intended to reduce, limit, or restrict any uses free from copyright or rights arising from limitations or exceptions that are provided for in connection with the copyright protection under copyright law or other applicable laws.

3. License Grant.

Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

a. to Reproduce the Work, to incorporate the Work into one or more Collections, and to Reproduce the Work as incorporated in the Collections;
b. and, to Distribute and Publicly Perform the Work including as incorporated in Collections. The above rights may be exercised in all media and formats whether now known or hereafter devised.

The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats, but otherwise you have no rights to make Adaptations. Subject to 8(f), all rights not expressly granted by Licensor are hereby reserved, including but not limited to the rights set forth in Section 4(d).

4. Restrictions.

The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

a. You may Distribute or Publicly Perform the Work only under the terms of this License. You must include a copy of, or the Uniform Resource Identifier (URI) for, this License with every copy of the Work You Distribute or Publicly Perform. You may not offer or impose any terms on the Work that restrict the terms of this License or the ability of the recipient of the Work to exercise the rights granted to that recipient under the terms of the License. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties with every copy of the Work You Distribute or Publicly Perform. When You Distribute or Publicly Perform the Work, You may not impose any effective technological measures on the Work that restrict the ability of a recipient of the Work from You to exercise the rights granted to that recipient under the terms of the License. This Section 4(a) applies to the Work as incorporated in a Collection, but this does not require the Collection apart from the Work itself to be made subject to the terms of this License. If You create a Collection, upon notice from any Licensor You must, to the extent practicable, remove from the Collection any credit as required by Section 4(c), as requested.
b. You may not exercise any of the rights granted to You in Section 3 above in any manner

that is primarily intended for or directed toward commercial advantage or private monetary compensation. The exchange of the Work for other copyrighted works by means of digital file-sharing or otherwise shall not be considered to be intended for or directed toward commercial advantage or private monetary compensation, provided there is no payment of any monetary compensation in connection with the exchange of copyrighted works.

c. If You Distribute, or Publicly Perform the Work or Collections, You must, unless a request has been made pursuant to Section 4(a), keep intact all copyright notices for the Work and provide, reasonable to the medium or means You are utilizing: (i) the name of the Original Author (or pseudonym, if applicable) if supplied, and/or if the Original Author and/or Licensor designate another party or parties (e.g., a sponsor institute, publishing entity, journal) for attribution ("Attribution Parties") in Licensor' s copyright notice, terms of service or by other reasonable means, the name of such party or parties; (ii) the title of the Work if supplied; (iii) to the extent reasonably practicable, the URI, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work. The credit required by this Section 4(c) may be implemented in any reasonable manner; provided, however, that in the case of a Collection, at a minimum such credit will appear, if a credit for all contributing authors of Collection appears, then as part of these credits and in a manner at least as prominent as the credits for the other contributing authors. For the avoidance of doubt, You may only use the credit required by this Section for the purpose of attribution in the manner set out above and, by exercising Your rights under this License, You may not implicitly or explicitly assert or imply any connection with, sponsorship or endorsement by the Original Author, Licensor and/or Attribution Parties, as appropriate, of You or Your use of the Work, without the separate, express prior written permission of the Original Author, Licensor and/or Attribution Parties.

d. For the avoidance of doubt:

   (a) **Non-waivable Compulsory License Schemes.** In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme cannot be waived, the Licensor reserves the exclusive right to collect such royalties for any exercise by You of the rights granted under this License;

   (b) **Waivable Compulsory License Schemes.** In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme can be waived, the Licensor reserves the exclusive right to collect such royalties for any exercise by You of the rights granted under this License if Your exercise of such rights is for a purpose or use which is otherwise than noncommercial as permitted under Section 4(b) and otherwise waives the right to collect royalties through any statutory or compulsory licensing scheme; and,

   (c) **Voluntary License Schemes.** The Licensor reserves the right to collect royalties, whether individually or, in the event that the Licensor is a member of a collecting

society that administers voluntary licensing schemes, via that society, from any exercise by You of the rights granted under this License that is for a purpose or use which is otherwise than noncommercial as permitted under Section 4(b).

e. Except as otherwise agreed in writing by the Licensor or as may be otherwise permitted by applicable law, if You Reproduce, Distribute or Publicly Perform the Work either by itself or as part of any Collections, You must not distort, mutilate, modify or take other derogatory action in relation to the Work which would be prejudicial to the Original Author' s honor or reputation.

5. Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTIBILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability.

EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

a. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Collections from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.

b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force

and effect unless terminated as stated above.

8. Miscellaneous

a. Each time You Distribute or Publicly Perform the Work or a Collection, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.

b. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

c. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.

d. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You.

e. This License may not be modified without the mutual written agreement of the Licensor and You. The rights granted under, and the subject matter referenced, in this License were drafted utilizing the terminology of the Berne Convention for the Protection of Literary and Artistic Works (as amended on September 28, 1979), the Rome Convention of 1961, the WIPO Copyright Treaty of 1996, the WIPO Performances and Phonograms Treaty of 1996 and the Universal Copyright Convention (as revised on July 24, 1971). These rights and subject matter take effect in the relevant jurisdiction in which the License terms are sought to be enforced according to the corresponding provisions of the implementation of those treaty provisions in the applicable national law. If the standard suite of rights granted under applicable copyright law includes additional rights not granted under this License, such additional rights are deemed to be included in the License; this License is not intended to restrict the license of any rights under applicable law.

Trademarks

ANSYS is a registered trademark of ANSYS Inc.
CFX is a registered trademark of Ansys Inc.
CHEMKIN is a registered trademark of Reaction Design Corporation
EnSight is a registered trademark of Computational Engineering International Ltd.
Fieldview is a registered trademark of Intelligent Light
Fluent is a registered trademark of Ansys Inc.
GAMBIT is a registered trademark of Ansys Inc.
Icem-CFD is a registered trademark of Ansys Inc.
I-DEAS is a registered trademark of Structural Dynamics Research Corporation
JAVA is a registered trademark of Sun Microsystems Inc.
Linux is a registered trademark of Linus Torvalds
OpenFOAM is a registered trademark of ESI Group.
ParaView is a registered trademark of Kitware
STAR-CD is a registered trademark of Computational Dynamics Ltd.
UNIX is a registered trademark of The Open Group

目次

第1章
はじめに

本ガイドは，Open Source Field Operation and Manipulation (OpenFOAM) C++ ライブラリ，バージョン 2.3.0 リリースに付属するものです．本ガイドでは，まず第 2 章のチュートリアル演習を通して，またそれ以降は OpenFOAM を構成する個々のコンポーネントに関するより詳細な記述によって，OpenFOAM の基礎的な操作法に関して説明しています．

OpenFOAM に関して最も重要なことは，主にアプリケーションとよばれる実行ファイルを作成するために使われる C++ ライブラリであるということです．このアプリケーションは，連続体力学における特定の問題を解くためのソルバ，およびデータに対する各種の操作を行うためのユーティリティの二つのカテゴリに大別されます．OpenFOAM の配布物は第 3 章に述べるように，多岐にわたる問題を扱うための多数のソルバおよびユーティリティを含んでいます．

OpenFOAM の特長の一つは，背景となる手法，物理学，関連するプログラミング技術に関する知識があれば，新しいソルバやユーティリティをユーザ自身が作成可能であることです．これらに関する情報はプログラマズガイドに掲載しています．

OpenFOAM には前処理・後処理の環境も含まれています．前処理・後処理へのインタフェースもまた OpenFOAM のユーティリティですから，OpenFOAM 内の全ての環境にわたってデータの取扱いの一貫性が保たれています．OpenFOAM の全体的な構造を図 1.1 に示します．

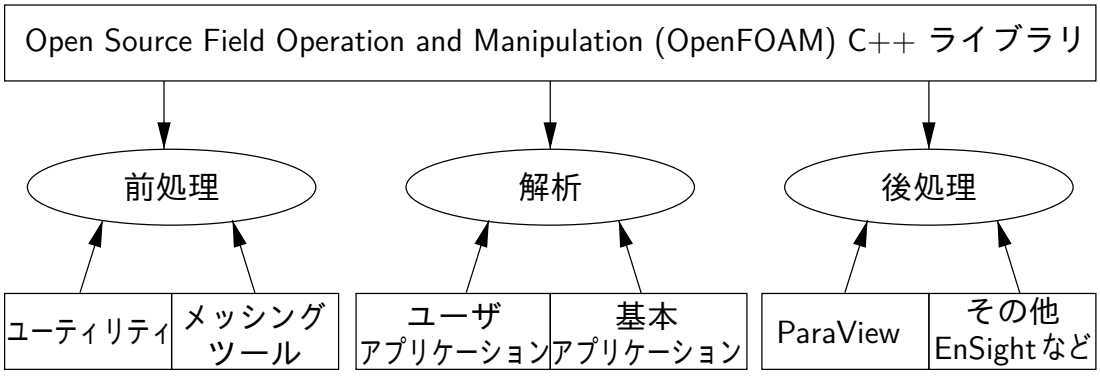


図 1.1　OpenFOAM の全体的な構造

前処理や OpenFOAM のケースの実行方法については，第 4 章で説明します．第 5 章では，OpenFOAM に付属するメッシュ・ジェネレータを使ってメッシュを生成する方法や，サードパーティ製品で生成したメッシュを変換する方法も説明します．後処理については第 6 章で説明します．

第2章
チュートリアル

この章では OpenFOAM を動かす基本的な手順をユーザに説明することを主な目標として，OpenFOAM のいくつかのテストケースで，設定，シミュレーション，および後処理のプロセスを詳しく記述します．*$FOAM_TUTORIALS* のディレクトリには OpenFOAM が提供するすべてのソルバと多くのユーティリティを使い方を示す数多くのケースがあります．チュートリアルを始める前にユーザは最初に OpenFOAM が正しくインストールされていることを確かめなければなりません．

チュートリアルのケースは blockMesh の前処理ツールを使用して記述し，OpenFOAM のソルバで動かし，paraFoam を使用して後処理を行います．OpenFOAM のサポートするサードパーティの後処理ツールにアクセスするユーザには次の選択肢があります．paraFoam を使用しチュートリアルを進めるか，または後処理が必要な際に第 6 章で述べるサードパーティ製品の使い方を参照するかです．

すべてのチュートリアルのコピーは OpenFOAM をインストールしたチュートリアルのディレクトリから利用できます．チュートリアルは，流れのタイプによるディレクトリとソルバのサブディレクトリにまとめられています．例えば icoFoam のケースはすべて *incompressible/icoFoam* サブディレクトリの中に置かれています．ここで incompressible が流れのタイプを表しています．ユーザがさまざまな例題を実施するときには，*tutorials* ディレクトリをローカルの実行ディレクトリにコピーすることをお勧めします．そのためには，次のようにタイプすることで容易にコピーすることができます．

```
mkdir -p $FOAM_RUN
cp -r $FOAM_TUTORIALS $FOAM_RUN
```

2.1 天井駆動のキャビティ流れ

このチュートリアルは 2 次元正方形領域の等温非圧縮性流れに関して，前処理，計算，後処理する方法を解説します．図 2.1 に正方形のすべての境界が壁面境界とした形状を示します．上の壁面境界は x 軸方向に 1 m/s の速度ではたらき，他の三つの壁面境界は静止しています．チュートリアルにおいてはこれを解くにあたって，まず層流を仮定し，層流等温非圧縮性流れのための icoFoam ソルバを使用し均一メッシュ上で解きます．チュートリアルでは，メッシュの解像度の増加や壁方向への勾配の影響を調べます．これにより流れの Reynolds 数を増加させ，pisoFoam ソルバを乱流，等温，非圧縮性流れに使用します．

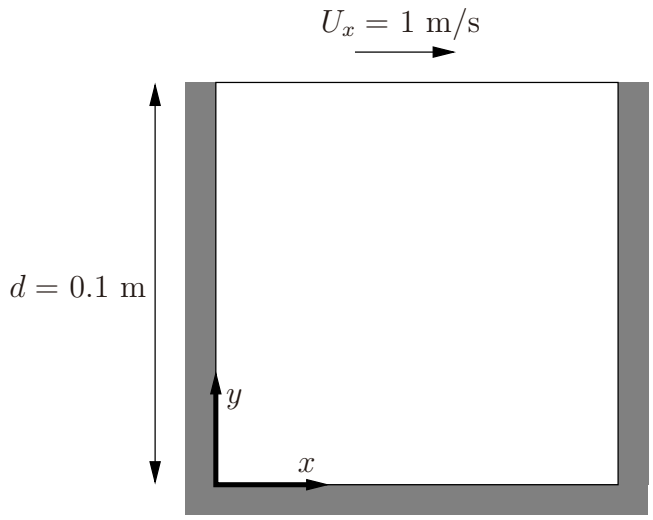


図 2.1　天井駆動キャビティのジオメトリ

2.1.1　前処理

ケースは OpenFOAM でケースファイルを編集することで設定します．ケースファイルは emacs や vi，gedit，kate，nedit などのテキストエディタで作成・編集します．OpenFOAM では，初心者でも理解できるようなわかりやすいキーワードをもつディクショナリ形式を用いて入出力を行うため，ファイルを直接編集することが可能です．

解析ケースはメッシュ，物理量，物性，制御パラメータなどの要素を含んでいますが 4.1 節において示すように，多くの CFD ソフトが一つのファイルにこれらのデータを格納するのに対し，OpenFOAM は一連のファイルセットとして解析ケースディレクトリに格納します．解析ケースのディレクトリには，(最初のチュートリアルの例題が単純に cavity であるように) わかりやすい名前を与えます．解析ケースを編集・実行する前の準備として，まず解析対象のディレクトリに移動します．

```
cd $FOAM_RUN/tutorials/incompressible/icoFoam/cavity
```

2.1.1.1　メッシュ生成

OpenFOAM は常に 3 次元 Cartesian 座標系で動くため，全てのジオメトリを 3 次元で生成します．OpenFOAM はデフォルトの設定において問題を 3 次元として解きますが，2 次元を解く場合は，解決が必要でない（第 3）次元方向に垂直な境界に特別な empty という境界条件を指定します．

x–y 平面上の一辺の長さの正方形からなるキャビティの領域に，まず 20×20 セルの均一なメッシュを設定します．このブロック構造を図 2.2 に示します．

OpenFOAM で提供されるメッシュ・ジェネレータ blockMesh は *constant/polyMesh* ディレクトリにある入力ディクショナリ *blockMeshDict* で指定された記述からメッシュを生成します．このケースの *blockMeshDict* は，以下のとおりです．

```
1  /*--------------------------------*- C++ -*----------------------------------*\
2  | =========                 |                                                 |
3  | \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
```

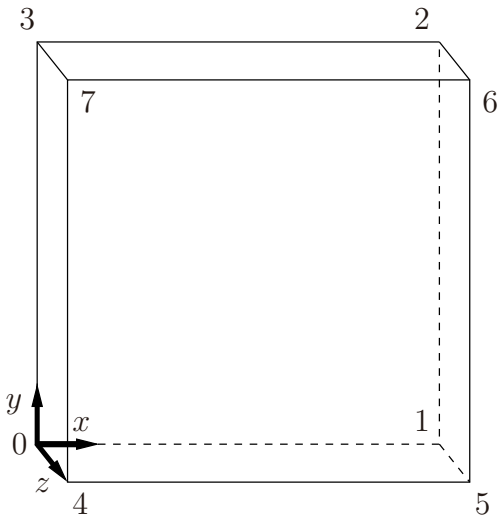


図 2.2 キャビティのメッシュのブロック構造

```
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 0.1;

vertices
(
    (0 0 0)
    (1 0 0)
    (1 1 0)
    (0 1 0)
    (0 0 0.1)
    (1 0 0.1)
    (1 1 0.1)
    (0 1 0.1)
);

blocks
(
    hex (0 1 2 3 4 5 6 7) (20 20 1) simpleGrading (1 1 1)
);

edges
(
);

boundary
(
    movingWall
    {
        type wall;
        faces
        (
            (3 7 6 2)
        );
    }
    fixedWalls
    {
```

```
52          type wall;
53          faces
54          (
55              (0 4 7 3)
56              (2 6 5 1)
57              (1 5 4 0)
58          );
59      }
60      frontAndBack
61      {
62          type empty;
63          faces
64          (
65              (0 3 2 1)
66              (4 5 6 7)
67          );
68      }
69  );
70
71  mergePatchPairs
72  (
73  );
74
75  // ************************************************************************* //
```

ファイルの最初（1–7 行目）にはバナー形式のヘッダ情報があり，それから波括弧 ({...}) で囲まれる *FoamFile* サブディクショナリの中に，ファイルの情報が記述されています．

簡便化とスペースの都合上，今後ケースファイルを引用する際は，バナーと *FoamFile* サブディクショナリを含むファイルヘッダは省きます．

ファイルの最初には，ブロックの頂点の座標 `vertices` を指定します．それから，頂点名とセル番号から `blocks`（ここでは一つのみ）を定義します．そして最後に境界パッチを定義します．*blockMeshDict* ファイルの記述の詳細を理解するには 5.3 節を参照してください．

メッシュは *blockMeshDict* ファイル上で blockMesh を実行すると生成されます．ケースディレクトリ内から以下をターミナルに入力するだけで実行されます．

```
blockMesh
```

blockMesh の実行状況はターミナルウィンドウに表示されます．*blockMeshDict* ファイルに誤りがあった場合，エラーメッセージが表示され，ファイルのどの行に問題があるかを教えてくれます．今この段階でエラーメッセージが出ることはないでしょう．

2.1.1.2　境界条件と初期条件

メッシュの生成が完了すると，物理的条件の初期状態を確認することができます．このケースは開始時刻がに設定されているので解析領域の初期状態のデータは *cavity* ディレクトリの *0* というサブディレクトリに格納されています．*0* には *p* と *U* の二つのファイルがあり，圧力 (p) と速度 (U) の初期値と境界条件を設定する必要があります．*p* のファイルを例に説明します．

```
17  dimensions      [0 2 -2 0 0 0 0];
18
19  internalField   uniform 0;
20
21  boundaryField
22  {
23      movingWall
24      {
25          type            zeroGradient;
```

```
26      }
27
28      fixedWalls
29      {
30          type            zeroGradient;
31      }
32
33      frontAndBack
34      {
35          type            empty;
36      }
37  }
38
39  // ************************************************************************* //
```

物理的条件のデータファイルには三つの主要な項目があります．

dimensions　物理量の次元を指定します．ここでは動圧，つまり $\mathrm{m^2s^{-2}}$（4.2.6 項に詳述）とします．

internalField　内部の物理量は単一の値で記述すれば一様となり，一様でない場合はすべての値を指定する必要があります（4.2.8 項に詳述）．

boundaryField　境界面の物理量は境界条件と境界パッチに与えるデータを記述します（4.2.8 項に詳述）．

このキャビティ流れの解析ケースでは境界は壁面のみですが，二つのパッチに分けて以下のように名付けます．(1) キャビティの固定された側面と底面用の `fixedwall` と，(2) キャビティの駆動天井面用の `movingwall` です．どちらも p が `zeroGradient` ですが，これは圧力の境界に垂直な方向の勾配が 0 であるということです．`frontAndBack` は 2 次元の問題の場合の表裏の平面を示していて，本ケースでは当然 `empty` となっています．

このケースでは，もっともよく目にするものでありますが，物理量の初期条件が `uniform`（一様）になっています．ここでは圧力は動圧のみの非圧縮ケースであるため，絶対値は解析と関係ないので便宜上 `uniform 0` としています．

0/U の速度のファイルにおいても同様です．`dimensions` は速度であり，内部の初期条件はベクトル量で 3 成分とも 0 を意味する `uniform (0 0 0)` になっています（4.2.5 項に詳述）．

速度の境界条件は `frontAndBack` パッチと同じ条件です．その他の境界は壁ですが，`fixedWall` ではすべりなし (no-slip) 条件を仮定するため，`value` が `uniform (0 0 0)` の `fixedValue` とします．上面は x 方向に $1\,\mathrm{m/s}$ で移動するので，こちらも `fixedValue` ですが，`value` は `uniform (1 0 0)` とします．

2.1.1.3 物性値

ケースの物理量は，名前に*...Properties* という語尾を与えられてディクショナリに保存され，*Dictionaries* ディレクトリツリーに置かれます．icoFoam ケースでは，*transportProperties* ディクショナリに保存される動粘性係数を指定するだけです．*transportProperties* ディクショナリを開いてその中身を確認したり，編集することができますので，動粘性係数が正しくセットされることを確かめてください．動粘性係数は，`nu` (数式中で ν と書かれるギリシア文字の音声ラベル) というキーワードになります．まず最初に，このケースは Reynolds 数を 10 で計算し

ます．Reynolds 数は次のように定義されます．

$$Re = \frac{d|\boldsymbol{U}|}{\nu} \tag{2.1}$$

d と $\boldsymbol{U}$ はそれぞれ特性長さと速度を表し，ν は動粘性係数を表します．ここで，$d = 0.1\,\mathrm{m}$，$|\boldsymbol{U}| = 1\,\mathrm{m\,s^{-1}}$，$Re = 10$ とすると，$\nu = 0.01\,\mathrm{m^2 s^{-1}}$ となります．動粘性係数の適切な設定は以下のようになります．

```
17
18 nu              nu [0 2 -1 0 0 0 0] 0.01;
19
20
21 // ************************************************************************* //
```

2.1.1.4　制御

計算時間の制御，解のデータの読み書きに関する入力データは，*controlDict* ディクショナリから読み取られます．これは *system* ディレクトリにありますので，ケースを制御するファイルとして参照してください．

まず最初にスタート・停止時刻と時間ステップを設定しなければなりません．OpenFOAM は，柔軟性の高い時間制御を提供しますが，詳しくは 4.3 節で述べます．このチュートリアルでは，時刻 $t = 0$ から実行を始めたいと思います．つまり，OpenFOAM は *0* というディレクトリから場のデータを読む必要があることになります（ケースファイル構造の詳しい情報に関しては 4.1 節を見てください）．したがって，`startFrom` キーワードを `startTime` に設定して，次に `startTime` キーワードを 0 に指定します．

終了時刻には，流れがキャビティ周りを循環している定常解に達することを目標にするわけですが，概して，流体は層流で定常状態に到達するために領域を 10 回通り抜けなければなりません．このケースでは，入口も出口もないので，流れが解析領域を通り抜けません．代わりに，ふたがキャビティを 10 回移動する時刻 (すなわち 1 s) を終了時刻としてセットしてもいいでしょう．実際は，後の知見により，0.5 s で十分であるとわかるので，この値を採用しましょう．この終了時刻を指定するために，`stopAt` キーワードとして `endTime` を指定して，`endTime` キーワードを 0.5 に設定しなければなりません．

次に，時間ステップを設定する必要がありますが，これはキーワード `deltaT` によって表されます．icoFoam を動かすとき，時間の精度と安定性を達成するために，1 未満の Courant 数が必要です．Courant 数は以下のように定義されます．

$$Co = \frac{\Delta t|\boldsymbol{U}|}{\Delta x} \tag{2.2}$$

Δt は時間ステップ，$|\boldsymbol{U}|$ はセルを通る流速の大きさ，そして Δx は流速方向のセルサイズです．流速が領域内で変化しても必ず $Co < 1$ を成り立たせる必要があります．だから，最も悪い場合（つまり，大きな流速と小さなセルサイズの組合わせによる最大の Co）を元に Δt を決定します．ここでは，セルサイズは解析領域中全域で固定されているので，最大 Co はふた付近に生じ，$1\,\mathrm{m\,s^{-1}}$ に近い流速になるでしょう．

$$\Delta x = \frac{d}{n} = \frac{0.1}{20} = 0.005\,\mathrm{m} \tag{2.3}$$

したがって，領域中で 1 以下の Courant 数を達成するために，時間ステップ deltaT を次のように設定しなくてはいけません．

$$\Delta t = \frac{Co\Delta x}{|\boldsymbol{U}|} = \frac{1 \times 0.005}{1} = 0.005\,\mathrm{s} \tag{2.4}$$

シミュレーションが進行するとき，後処理パッケージで後から見ることができるように，ある一定の時間間隔での結果の書き出しをもとめるため，writeControl キーワードは結果が書かれる時刻を決めるためのいくつかのオプションを提示します．timeStep オプションは，結果が n 回の時間ステップごとに結果を書き出すということを意味し，そのときの値は writeInterval キーワードで指定されます．$0.1, 0.2, \ldots, 0.5\,\mathrm{s}$ で結果を書きたいとしましょう．したがって，$0.005\,\mathrm{s}$ の時間ステップなので，時間ステップ 20 回ごとに結果を出力する必要があります．よって writeInterval に 20 を設定します．

OpenFOAM は 4.1 節で議論するデータセットを書き込むごとに例えば $0.1\,\mathrm{s}$ という現在時刻にちなんで名付けられた新しいディレクトリを作成します．icoFoam ソルバでは，U や p の各項目ごとに結果を時刻ディレクトリに書き込みます．このケースでは，*controlDict* の記述内容は以下のとおりです．

```
17
18 application     icoFoam;
19
20 startFrom       startTime;
21
22 startTime       0;
23
24 stopAt          endTime;
25
26 endTime         0.5;
27
28 deltaT          0.005;
29
30 writeControl    timeStep;
31
32 writeInterval   20;
33
34 purgeWrite      0;
35
36 writeFormat     ascii;
37
38 writePrecision  6;
39
40 writeCompression off;
41
42 timeFormat      general;
43
44 timePrecision   6;
45
46 runTimeModifiable true;
47
48
49 // ************************************************************************* //
```

2.1.1.5 離散化と線形ソルバの設定

ユーザは *fvSchemes* ディクショナリ (*system* ディレクトリ) 内で有限体積離散化法を選択するかどうか指定します．線形方程式ソルバと許容値および他のアルゴリズムコントロールの指定は *fvSolution* ディクショナリ内に作られています．ユーザは自由にこれらのディクショナリを見ることができますが，*fvSolution* ディクショナリの *PISO* サブディクショナリの pRefCell

と pRefValue を除いて，現在のところ，それらすべての項について議論する必要はありません．キャビティのような閉じた非圧縮系では，圧力は相対的であり，重要なのは（絶対値ではなく）圧力範囲です．このような場合では，ソルバはセル pRefCell に pRefValue による参照レベルをセットします．この例では，両方が 0 に設定されます．しかし，これらの値のどちらかを変えると絶対圧力（速度と相対圧力ではなく）が変化します．

2.1.2　メッシュの確認

解析を実行する前に正しくメッシュができているか確認しましょう．メッシュは OpenFoam が提供する後処理ソフトの paraFoam で確認します．paraFoam は解析ケースのディレクトリ上でターミナルから起動します．

```
paraFoam
```

あるいは，オプションに-case をつけることで他のディレクトリからでも起動することができます．

```
paraFoam -case $FOAM_RUN/tutorials/incompressible/icoFoam/cavity
```

図 6.1 に示すように ParaView のウィンドウが開きます．Pipeline Browser を見ると，ParaView が cavity.OpenFOAM，つまりキャビティケースのモデュールを開いていることが確認できます．Apply ボタンをクリックする前に Mesh Parts パネルから表示する要素を選択する必要があります．解析ケースが単純なので Mesh Parts パネルのチェックボックスで全てのデータを選択することが簡単です．パネル内の全要素を自動的にチェックすることができます．ParaView でジオメトリを読込むためには Apply ボタンをクリックします．

Display パネルを開き選択したモジュールの表示形式を調整します．図 2.3 に示すように，(1) Color by を Solid Color に設定し，(2) Set Solid Color をクリックし適当な色（背景が白の場合は黒など）を選択，(3) Style パネルでは Representation メニューから Wireframe を選択します．背景色はトップメニューパネルで Edit から View Settings... を選択して設定します．

ParaView を使うのがはじめてならば，6.1.5 項で述べるように視点操作を試してみることをお勧めします．特に本ケースは 2 次元なので Edit メニューの View Settings の General パネルで Use Parallel Projection を選択するのがよいでしょう．軸の方向は，Annotation ウィンドウの Orientation Axes をオン・オフするか，マウスのドラッグ&ドロップによって操作することができます．

2.1.3　アプリケーションの実行

あらゆる UNIX/Linux の実行ファイルと同様に，OpenFOAM アプリケーションは二つの方法で実行することができます．一つ目はフォアグラウンドのプロセスで，コマンドプロンプトを与えるのにシェルが命令終了まで待つものです．二つ目はバックグラウンドプロセスで，シェルがさらなる命令を受け入れるのに命令完了の必要がないものです．

ここでは，フォアグランドで icoFoam を動かしましょう．icoFoam ソルバはケースディレク

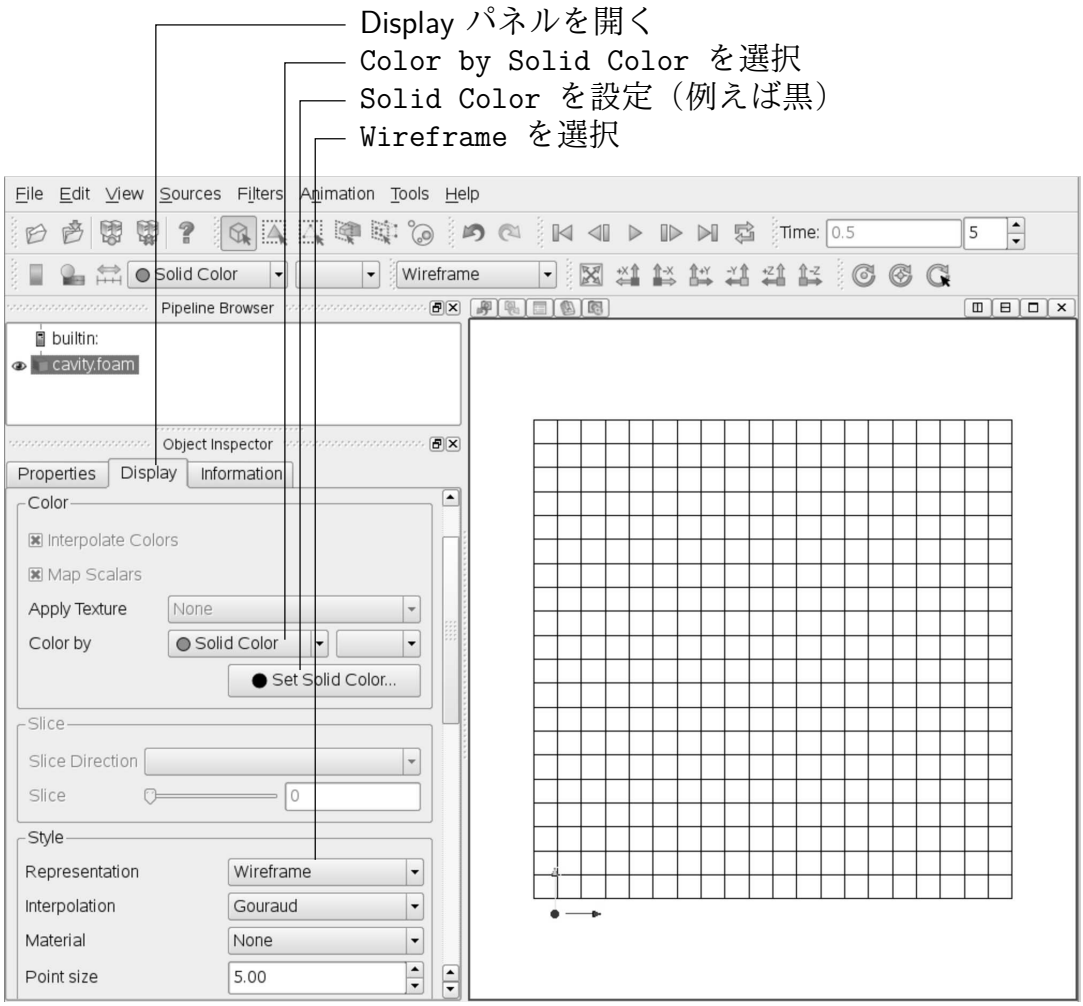


図 2.3 paraFoam でのメッシュの表示

トリ内に入って，コマンドプロンプト上で

```
icoFoam
```

と入力することで実行できますが，

あるいはオプションに-case をつけることで他のディレクトリからでも起動することができます．

```
icoFoam -case $FOAM_RUN/tutorials/incompressible/icoFoam/cavity
```

ジョブの進行過程は，ターミナルウィンドウに表示されます．現在の時刻，最大 Courant 数，全てのフィールドの初期値と最終的結果を表示します．

2.1.4 後処理

結果が時刻ディレクトリに書かれるとすぐに，paraFoam を使って見ることができます．paraFoam ウィンドウに戻って，cavity.OpenFOAM ケースモジュールの Properties パネルを選択してください．ケースモジュールのパネルが存在していないようならば，cavity.OpenFOAM が青

くハイライトされているか，それと並んだ目のボタンは表示が有効であることを示しているか，を確認してください．

見たいデータを表示する paraFoam を準備するには，最初に必須の実行時間として 0.5 s 分のデータを読込まなければなりません．ケースが実行中で一方 ParaView を開いている場合，時間ディレクトリの出力データは ParaView に自動的にロードはされません．データをロードするためには，Properties ウィンドウで Refresh Times をクリックします．これで各時刻のデータが ParaView にロードされます．

2.1.4.1　等値面とコンタプロット

圧力を見るには Display パネルを開き，選択したモジュールの表示形式を調整します．圧力分布を見るには図 2.4 に示すように Style パネルの Representation メニューを surface にして ColorPanel の Set Color by を ◦p，そして Rescale to Data Range ボタンをクリックし，メニューバーの下のツールバーにある `VCR Controls` または `Current Time Controls` で現在時刻を 0.5 にして $t = 0.5\,\mathrm{s}$ における解析結果を表示します．それらのパネルは図 6.4 に示すように ParaView ウィンドウのトップメニューの下にあります．圧力場の解析結果は図 2.5 のように左上が低く，右上が高い圧力分布になるはずです．

圧力分布を作成するには図 2.4 に示すように StylePanel で Representation メニューから Surface を選択し，Color パネルで ◦p，そして Rescale to Data Range ボタンによって Color を選択します．メニューバーの下のツールバーにある VCR Controls または Current time を 0.5 にして $t = 0.5\,\mathrm{s}$ における解析結果を表示します．

◦p のアイコンで圧力分布をセル間を補完した連続分布を表示します．もし Color by メニューからセルアイコン p を選択しなければ各々のセルが等級づけなしで一つの色によって意味されるように，圧力のための一つの値は各々のセルに起因しています．

Active Variable Controls ツールバーの Toggle Color Legend Visibility ボタンをクリックするか `View` メニューから `Show Color Legend` を選択することで，カラーバーを表示させることができます．Active Variable Controls toolbar か Display ウィンドウの Color panel にある Edit Color Map button をクリックするとフォントの大きさや種類，スケールの番号付けの形式など，カラーバーの設定を変更することができます．カラーバーはドラッグアンドドロップにより image ウィンドウに置くことも可能です．

最近のバージョンの ParaView では，よく使われる青・緑・赤という（虹色の）カラースケールではなく，青から白そして赤へと変化するカラースケールがデフォルトになっています．そこで，はじめて ParaView を使うユーザはこのカラースケールを変えたいと思うでしょう．これは，Color Scale Editor で Choose Preset を選び，Blue to Red Rainbow を選択することで変更できます．OK ボタンで確定したあとに，Make Default を押せば ParaView はいつもこのタイプのカラーバーを使うようになります．

イメージを回転をさせるとすべての表面に圧力分布で色づけされていることが確認できます．正しいコンタ図を得るために断面を作成するか，6.1.6.1 に示す slice フィルタを用いてジオメトリをスライスします．6.1.6.1 に示す slice フィルタを用います．断面の中心座標は $(0.05, 0.05, 0.005)$，基準点は $(0, 0, 1)$ とします (Z Normal ボタンをクリックします)．断面を作成後，6.1.6 項に示す contour フィルタによってコンタを描画します．

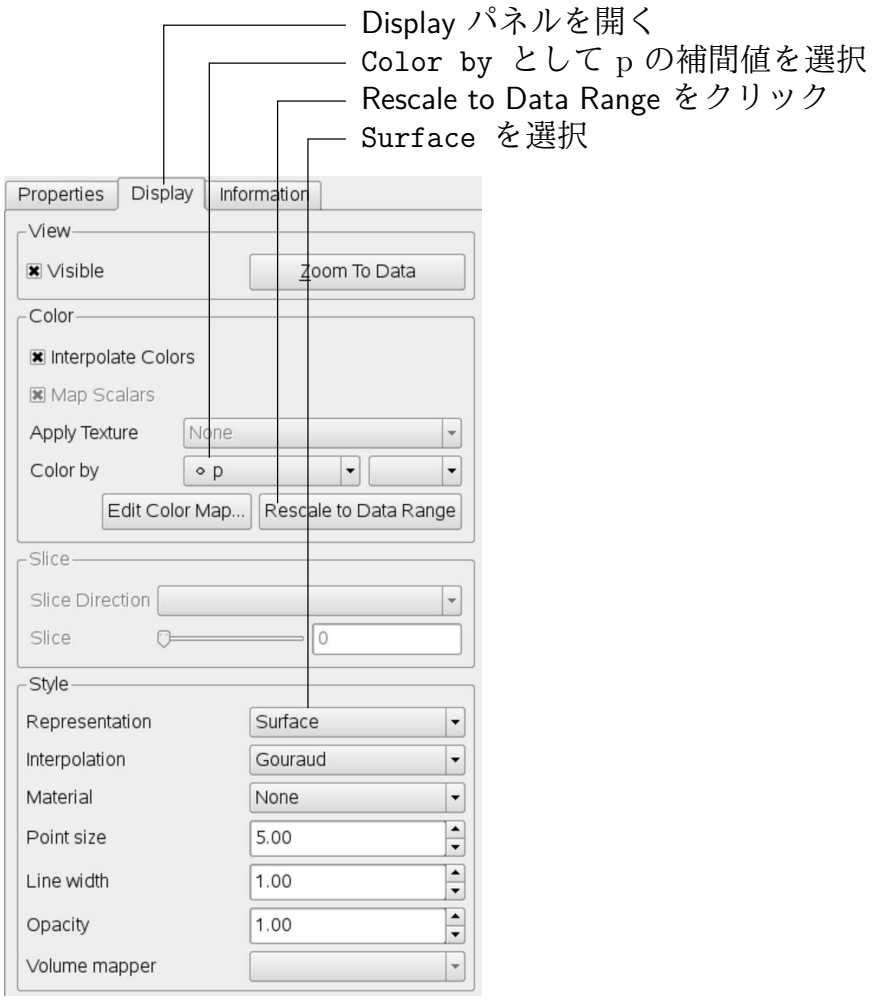


図 2.4　キャビティケースでの圧力等圧線の描画

2.1.4.2　ベクトルプロット

流速ベクトルを描画する前に，先に作成した断面やコンタなどの他のモジュールは不要なので取り除きましょう．Pipeline Browser でそれらのモジュールを選択し，Properties Panel の Delete をクリックして削除するか，Pipeline Browser で目の形のボタンをクリックしてそれらのモジュールを非表示にします．

各格子の中心におけるベクトルグラフを作成することにしましょう．まず，6.1.7.1 に述べるように格子の中心のデータのみに絞り込みます．Pipeline Browser 上で強調表示されている cavity.OpenFOAM のモジュールを選択し，Filter → Alphabetical メニューから Cell Centers を選択して Apply をクリックします．

Pipeline Browser で Centers が強調表示された状態で，Filter → Alphabetical メニューから Glyph を選択します．図 2.6 のような Properties ウィンドウが表示されます．この Properties パネルの `vectors` メニューでは，ベクトル場は速度のみなので，速度場 `U` が自動的に選択されています．`Scale Mode` は速度の `Vector Magnitude` が初期値として選択されていますが，領域全体を通る速度の様子を見るために，`off` を選択し，Set Scale Factor に 0.005 をにします．Apply をクリックするとベクトルが表示されますが，単色，例えば白になっているでしょう．通常は Display パネルで `Color by U` を選択して速度に応じた色付けをします．`Edit Color Map` の中で Show Color Legend を選択し，速度の凡例を表示させましょう．出力結果は図 2.7 のよ

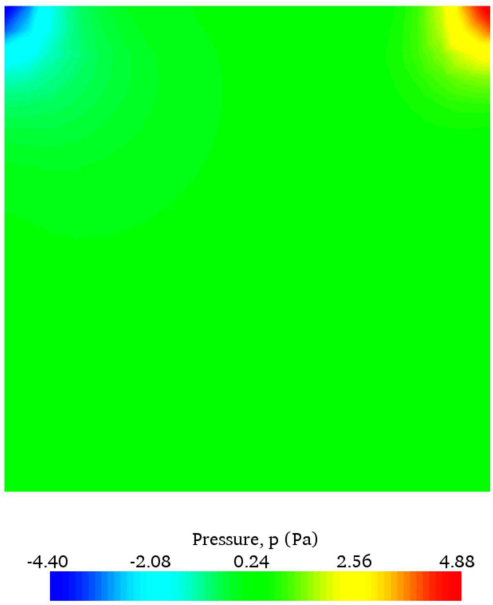


図 2.5　キャビティケースでの圧力

うになります．Color Legend（凡例）には Times Roman フォントが使用され，Automatic Label Format を解除して Label Format テキストボックスに `%-#6.2f` を入力することで二つの有効数字でラベルを固定しています．背景色は，6.1.5.1 で述べるように，`View Settings` の General パネルで白に設定されています．

左右の壁において，ベトルが壁面を通り抜けるように見えていることに注意してください．しかし，さらによく調べると，この壁に垂直な方向を向いている速度は 0 であることがわかります．この少し混乱する状態は，スケーリングが `off` で速度が 0 のとき，ParaView は x 方向のベクトルで表示するということに起因します．

2.1.4.3　流線プロット

ParaView で後処理を続ける前に，上述のベクトルプロットのモジュールは不要なので削除しましょう．そうしたら，6.1.8 項の記述のように流速の流線をプロットしましょう．

Pipeline Browser で `cavity.OpenFOAM` モジュールをハイライトした状態で，Filter メニューから `Stream Tracer` を選択し，Apply をクリックします．そうすると，図 2.8 に示すように Propaties ウィンドウが現れます．Seed の点は，ジオメトリの中心を垂直に通って，`Line Sourse` に沿うように (例えば $(0.05, 0, 0.005)$ から $(0.05, 0.1, 0.005)$ まで) 指定しましょう．このガイドに掲載した図では Point Resolution を 21 に，Max Propagation を `Length` で 0.5 に，Initial Step Length を `Cell Length` で 0.01 に，Integration Direction を `BOTH` という設定を行いました．また，`Runge-Kutta 2` Integrator Type はデフォルトパラメータで使いました．

Apply をクリックすると，トレーサが生成されます．そこで Filter メニューから `Tubes` を選択することで，高品質の流線図を作ることができます．このレポートでは，次の設定を使いました．Num. sides を 20，Radius を 0.003，Radius factor を 10 にしました．Accept を押すことで，図 2.9 ができます．

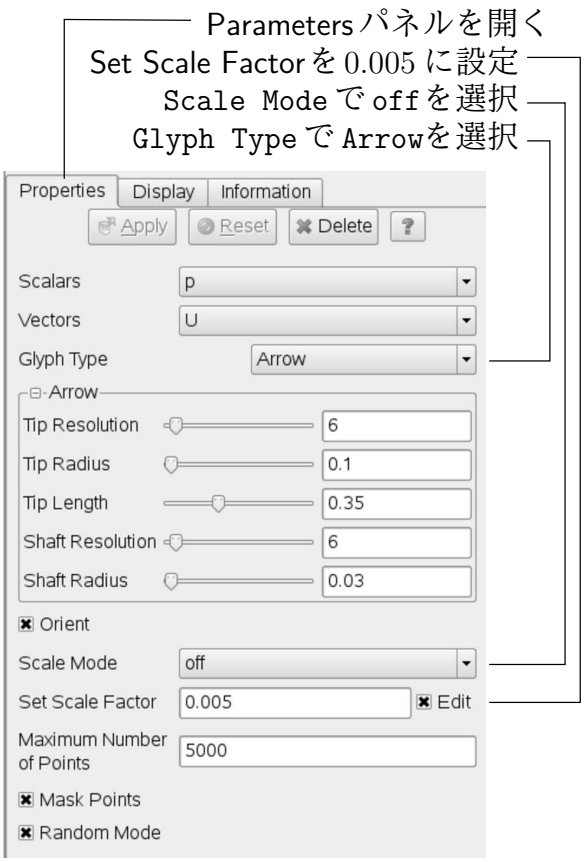


図 2.6　Glyph フィルタのパラメータパネル

2.1.5　メッシュの解像度を増やす

メッシュの解像度を各々の方向で 2 倍に増やします．問題の初期条件として使うために，粗いメッシュでの結果を，細かいメッシュ上に写像します．そして，細かいメッシュの解を粗いメッシュの解と比較します．

2.1.5.1　既存ケースを用いた新しいケースの作成

cavity をコピーし，修正することで解析ケース cavityFine を作成します．まず *cavity* と同じ階層に新しいディレクトリを作成します．

```
cd $FOAM_RUN/tutorials/incompressible/icoFoam
mkdir cavityFine
```

基本となる解析ケース *cavity* の内容を解析ケース *cavityFine* にコピーし，*cavityFine* に移動します．

```
cp -r cavity/constant cavityFine
cp -r cavity/system cavityFine
cd cavityFine
```

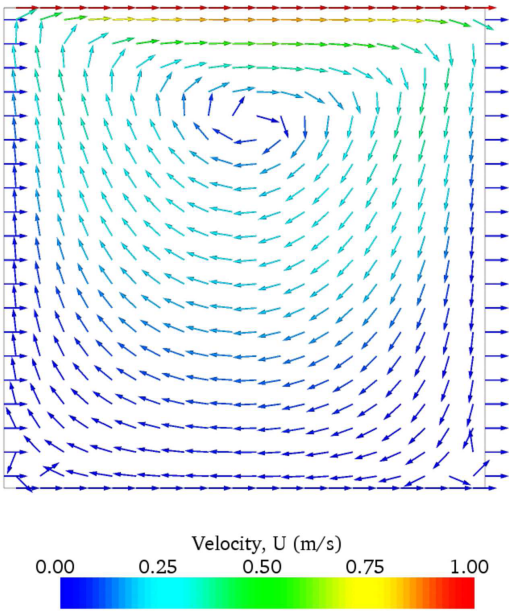


図 2.7　キャビティケースの速度

2.1.5.2　細かいメッシュの作成

blockMesh を使って計算格子数を増やしましょう．*blockMeshDict* ファイルをエディタで開き，ブロックに関する記述を修正します．ブロックを特定するには `blocks` というキーワードを用いましょう．ブロック定義の対称性に関しては 5.3.1.3 で詳しく述べるので，ここでは hex が最初の頂点リストで，各方向の計算格子の番号リストがあることを知ればよいでしょう．これは，先の cavity ケースでは (20 20 1) になっています．これを (40 40 1) に変え，保存します．ここで blockMesh を実効することで新しい，より細かいメッシュを生成することができます．

2.1.5.3　粗いメッシュの結果を細かなメッシュにマッピングする

mapFields ユーティリティは，他のジオメトリの対応するフィールドの上へ与えられたジオメトリに関した一つ以上のフィールドをマッピングします．本チュートリアルの例では，入力フィールドと求める結果のフィールド両方のジオメトリ・境界の種類・境界条件が同一であるので，フィールドは『首尾一貫している』と考えられます．この例で mapFields を実行するとき，`-consistent` コマンドラインオプションを使います．

mapFields maps のフィールドデータは，目的ケース（すなわち結果が図にされている）の *controlDict* 内の startFrom/startTime で指定される時間ディレクトリから読まれます．この例では，cavityFine ケースの細かいメッシュ上に cavity ケースから粗いメッシュの最終結果をマッピングしましょう．これらの結果が *cavity* の *0.5* のディレクトリに格納されているので，startTime を *controlDict* ディクショナリで 0.5 s に，startFrom を startTime にセットします．これらの変更を保存しましょう．

mapFields を実行する準備ができました．`mapFields -help` と打ち込むと mapFields の実行には入力ケースのディレクトリを指定する必要があることがわかります．`-consistent` オプションを使うので，次のようにユーティリティは *cavityFine* ディレクトリから実行される．

```
mapFields ../cavity -consistent
```

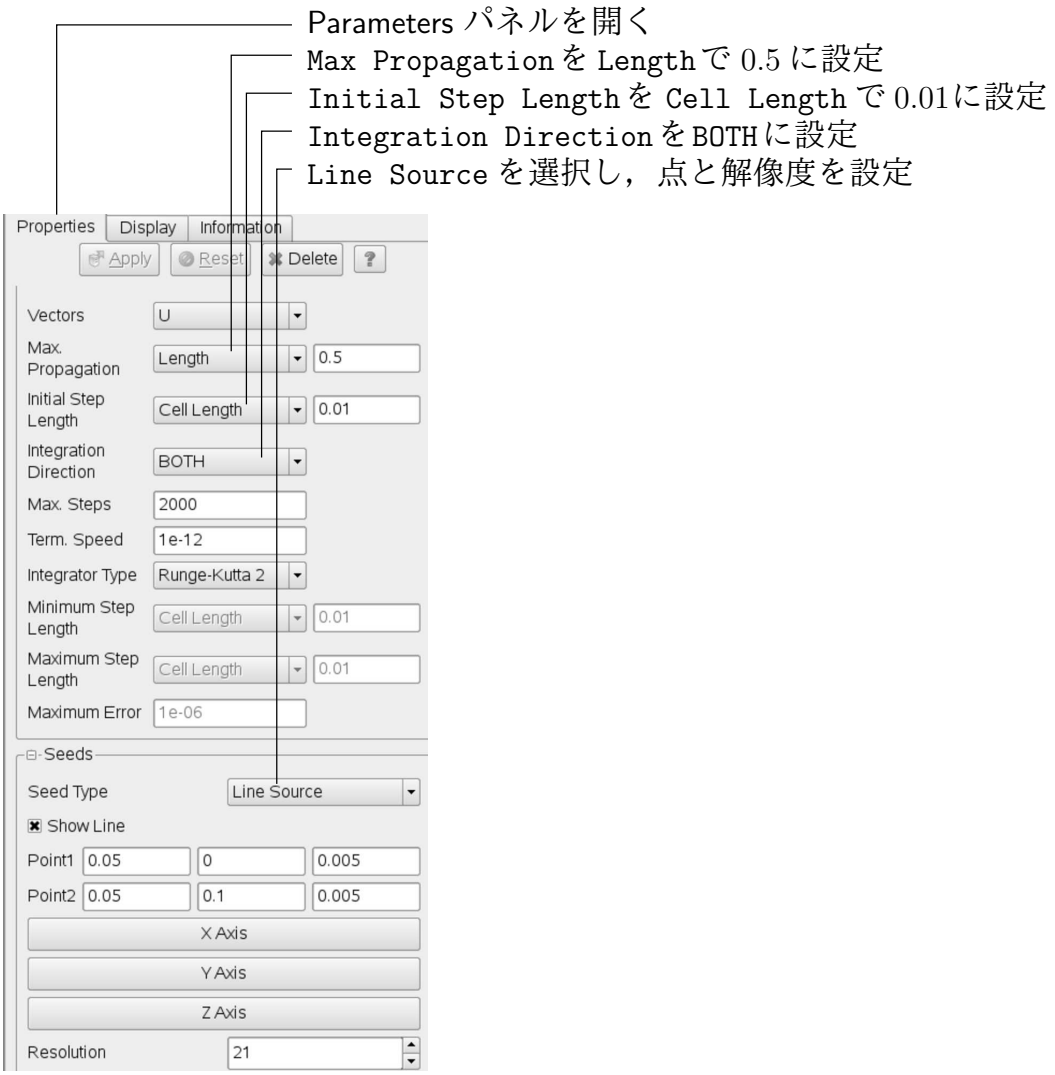


図 2.8　Stream Tracer フィルタのパラメータパネル

mapFields が実行され次のように出力されるでしょう．

```
Source: ".." "cavity"
Target: "." "cavityFine"

Create databases as time

Source time: 0.5
Target time: 0.5
Create meshes

Source mesh size: 400   Target mesh size: 1600

Consistently creating and mapping fields for time 0.5

    interpolating p
    interpolating U

End
```

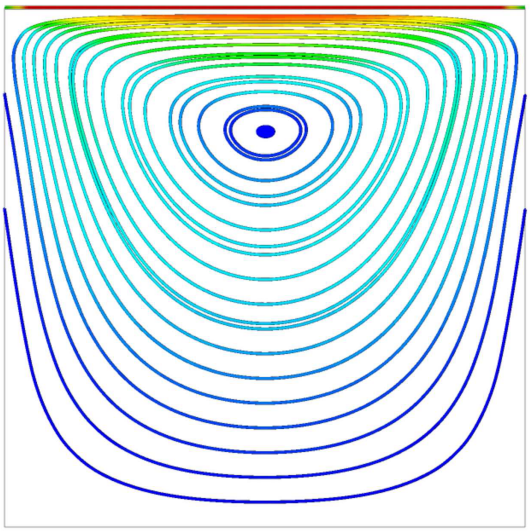


図 2.9　キャビティケースの流線

2.1.5.4　設定の調整

さて，全てのセルの寸法が半分になったので，1 より小さい Courant 数を維持するためには 2.1.1.4 で述べるように時間ステップを半分にしなければいけません．deltaT を *controlDict* ディクショナリにて 0.0025 s に設定しましょう．いままでは，フィールドデータを固定のステップ回数のもとでの時間間隔で出力する方法を示してきましたが，今回は固定の計算時間でデータ出力を指定する方法を示してみましょう．*controlDict* の `writeControl` キーワード下において，`timeStep` エントリで固定のステップ回数で出力する代わりに，`runTime` を使って固定の計算時間を指定して結果を出力することができます．

このケースでは 0.1 ごとの出力を指定します．したがって，`writeControl` を `runTime` に，`writeInterval` を 0.1 に設定しましょう．このようにすることで，ケースは粗いメッシュでの解を入力条件として計算をはじめるので，定常状態に収束するには適切な短い時間だけ動かせばよいのです．したがって，`endTime` は 0.7 s でよいでしょう．これらの設定が正しいことを確認し，ケースを保存しましょう．

2.1.5.5　バックグラウンドプロセスとしてコードを動かす

icoFoam をバックグラウンドプロセスとして動かしてみて，最終的な結果を後で見ることができるように *log* ファイルに出力しましょう．*cavitiyFine* ディレクトリにおいて次のコマンドを実行してください．

```
icoFoam > log &
cat log
```

2.1.5.6　精密なメッシュによるベクトルプロット

各々の新しいケースは本質的には単なる Pipeline Browser に現れる他のモジュールであるので，ParaView で同時に複数のケースを開くことができます．若干不便なことには，ParaView で新しいケースを開けるときには，選ばれたデータが拡張子を含むファイル名である必要があります．しかし，OpenFOAM において，各々のケースは特定のディレクトリ構造の中に拡張子なしで複数のファイルに保存されます．解決方法として，paraFoam スクリプトが自動的に拡張子

.OpenFOAM が付いたダミーファイルを作成することになっています．それゆえに，cavity ケースモジュールは `cavity.OpenFOAM` と名づけられます．

ParaView 内から他のケースディレクトリを開けたいならば，そのようなダミーファイルを作成する必要があります．たとえば，cavityFine ケースを読み込むには，コマンドプロンプトで次のようにタイプしてファイルを作成します．

```
cd $FOAM_RUN/tutorials/incompressible/icoFoam
touch cavityFine/cavityFine.OpenFOAM
```

こうして File メニューから Open Data を選んでディレクトリツリーをたどり，`cavityFine.OpenFOAM` を選ぶことで，cavityFine ケースを ParaView に読み込めるようになりました．さて，ParaView で精密なメッシュの結果のベクトルプロットを作ることができます．同時に両方のケースの glyph を見られるようににすることによって，cavityFine ケースのプロットを cavity ケースと比較することができます．

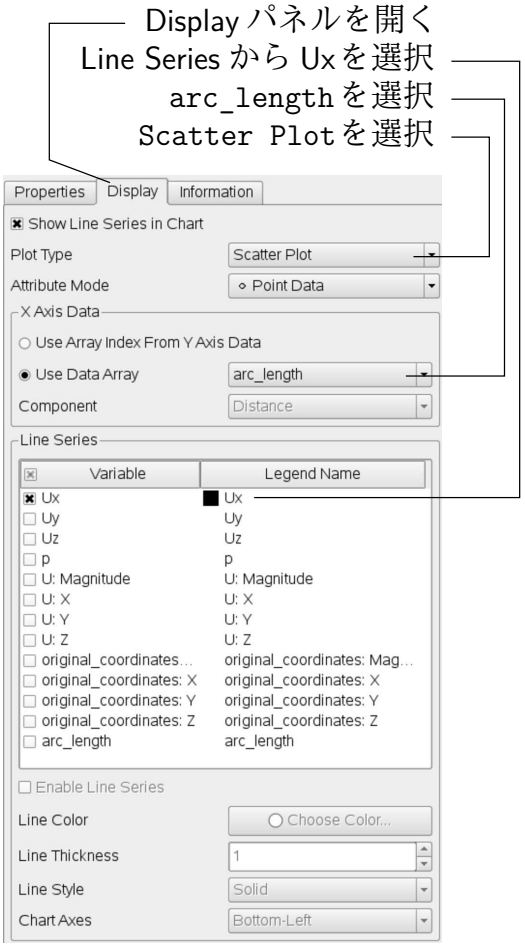


図 2.10　グラフ作図のためのフィールド選択

2.1.5.7　グラフを描く

OpenFOAM は，速度のスカラ値を抽出して 2 次元のグラフに描画したい場合のデータの取り扱いに長けています．データを操作するための特別なユーティリティが多数あり，単純な計算を foamCalc によって組み合わせることができます．次のようにユーティリティを指定して実

行します．

```
foamCalc <calcType> <fieldName1 ... fieldNameN>
```

処理を規定する<calcType>には addSubtract，randomise，div，components，mag，magGrad，magSqr，interpolate を指定することができます．<calcType>のリストを見るには，意図的に無効な処理を要求することでエラーメッセージとともに見ることができます．

```
>> foamCalc xxxx
Selecting calcType xxxx
unknown calcType type xxxx, constructor not in hash table
Valid calcType selections are:

8
(
randomise
magSqr
magGrad
addSubtract
div
mag
interpolate
components
)
```

components および mag の calcType はスカラ速度を計測するのに有用です．ケースにて“foamCalc components U”を動かすと，各時刻のディレクトリから速度のベクトル場を読み込み，各ディレクトリに各軸方向成分のスカラ場 Ux，Uy，Uz を書き出します．同様に“foamCalc mag U”とは各時刻のディレクトリにスカラ場 magU を書き込みます．

foamCalc は cavity と cavityFine のどちらに対しても実行することができます．例えば cavity に対しては，以下のように *cavity* ディレクトリに移動して foamCalc を実行します．

```
cd $FOAM_RUN/tutorials/incompressible/icoFoam/cavity
foamCalc components U
```

それぞれの成分が ParaView 内でグラフとして描画されます．簡単に，早く，しかもラベル付けや形式化の調整ができるので，とても高性能な出力を表示ができます．しかしながら，出版用にグラフを作成するならば gnuplot や Grace/xmgr などの専用のグラフ描画ソフトを使って生データから作画するのがよいでしょう．これを行うには，6.5 節や 2.2.3 項で述べる sample ユーティリティを使うとよいでしょう．

描画をする前に，新しく生成された Ux，Uy，Uz のデータを ParaView に読み込ませる必要があります．これには，cavity.OpenFOAM モジュールの Properties パネルの上部にある Refresh Times をクリックします．これにより，ParaView に新しいフィールドが読み込まれ，Volume Fields ウィンドウに現れます．新しいフィールドを選択し，変更が適用されたことを確認します．つまり，必要なら Apply を再度クリックします．また，Mesh Parts パネルで境界領域が選択されているならば，境界部分のデータ補間が不適切に行われています．したがって，Mesh Parts パネルで，movingwall や fixedwall，frontAndBack といったパッチの選択を解除して，変更を適用します．

さて，ParaView でグラフを表示してみましょう．まずは描画したいモジュールを選択し，

Plot Over Line フィルタを Filter → Data Analisys から選択します．3D View ウィンドウの下または横に新しい XY Plot ウィンドウが開きます．Properties ウィンドウで線の終点を指定すると Plobeline モジュールが作成されます．この例では Point1 を $(0.05, 0, 0.005)$，Point2 を $(0.05, 0.1, 0.005)$ と指定して線を領域の中心の真上におきます．Resolution は 100 まで設定できます．

Apply をクリックすると XY Plot ウィンドウにグラフが描画されます．Display パネルで，Attribute Mode を Point Data に設定します．Use Data Array オプションを X Axis Data にし，arc_length オプションを加えて，グラフの x 軸データがキャビティの底からの距離になるようにできます．

Display ウィンドウの Line Series パネルから表示するデータを選択することができます．表示されているスカラ場のリストから，ベクトルの大きさや成分を初期値とすることもできます．つまり，Ux を foamCalc から計算する必要はありません．それでも，Ux 以外の系列の選択はすべて解除しましょう．選択した系列の上の四角形の色が線の色です．この上でダブルクリックをすれば簡単に変更することができます．

グラフの体裁を整えるには，Line Series パネルの下にある設定，Line Color，Line Thickness，Line Style，Marker Style，そして Chart Axes を変更します．

また，XY Plot の左上にあるボタンをクリックすることもできます．例えば，3 番目のボタンでは，それぞれの軸のタイトルや凡例などを設定する View Settings を制御することができます．また，軸のタイトルのフォント，色，配置，値の範囲や線形・対数表示など，様々な設定を行うことができます．

図 2.11 は ParaView によって作画された図です．望みどおりのグラフが作成できます．図 2.11 は軸のオプションとして Standard type of Notation，Specify Axis Range を選択し，フォントは Sans Serif の 12 ポイントです．このグラフは点で表示していますが，Display ウィンドウで Enable Line Series ボタンを有効にすれば線で表示できます．注：もしこのボタンが，グレー表示で無効の状態になっていたら，Line Series パネルでどれか変数を選択すれば有効になります．Enable Line Series ボタンを選択しておけば，Line Style や Marker Style もユーザの好みで調整できます．

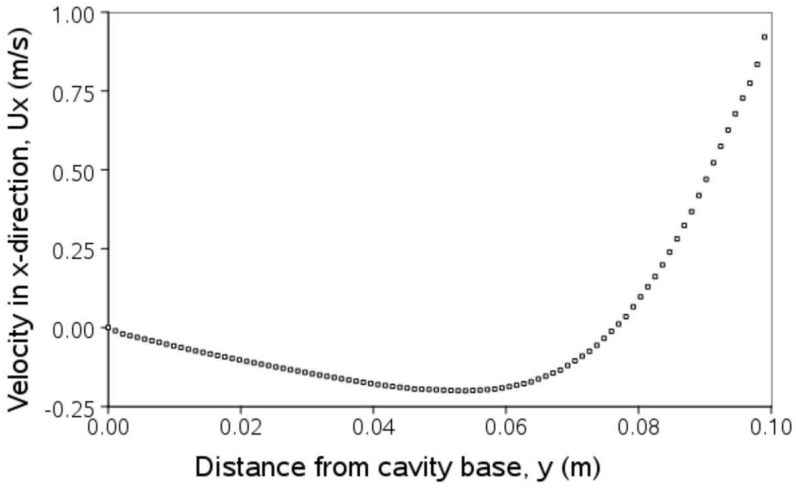


図 2.11 paraFoam でのグラフ作図

2.1.6　勾配メッシュ

解の誤差は，正しい解の形と選択した数値スキームで想定される形とが大きく異なる領域で出ます．例えば，数セルにわたる変数の線形変化に基づく数値スキームは，正しい解自体が線形の場合にしか正確な解を導くことができません．例えば勾配の変化が最も大きいところのような正しい解が線形から一番大きく外れる領域で誤差は最も大きくなります．セルの大きさに従って，誤差は減少します．

どんな問題も取りかかる前に解の概形の直感的予測ができるといいです．次に，誤差が最も大きくなるところを予測し，メッシュ幅に勾配をつけ，最も小さいセルがこれらの領域にくるようにします．キャビティの場合，壁の近くで速度の大きい変化があることを予想できるので，チュートリアルのこの部分では，メッシュがこの領域で，より小さくなるように勾配付けします．同じ数のセルを使用することによって，コンピュータの負荷をあまり増加させずに，より精度を上げられます．

lid-driven キャビティ問題のために壁に向かって勾配を付けた 20×20 セルのメッシュを作り，2.1.5.2 の細かいメッシュの結果を初期条件として勾配付けされたメッシュに適用しましょう．そして，勾配付けされたメッシュの結果を前のメッシュの結果と比較してみましょう．*blockMeshDict* ディクショナリの書換えはとても重要であるので，チュートリアルのこの部分を使ったケース (*cavityGrade*) は*$FOAM_RUN/tutorials/incompressible/icoFoam* ディレクトリに入れておきました．

2.1.6.1　勾配メッシュの作成

ここで，四つの異なるメッシュ間隔の計算メッシュが計算領域の上下左右のブロックに必要となります．このメッシュのブロック構造を図 2.12 に示します．

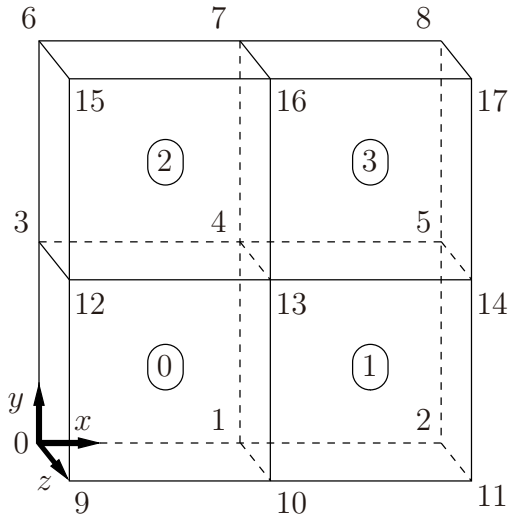


図 2.12　キャビティケースの勾配メッシュのブロック構造（ブロック番号）

cavityGrade の *constant/polyMesh* サブディレクトリで *blockMeshDict* ファイルを見ることができます．念のため *blockMeshDict* の重要な要素を以下に述べます．それぞれのブロックは x 方向，y 方向に 10 セルを有し，もっとも大きなセルともっとも小さなセルとの大きさの比は 2 です．

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 0.1;

vertices
(
    (0 0 0)
    (0.5 0 0)
    (1 0 0)
    (0 0.5 0)
    (0.5 0.5 0)
    (1 0.5 0)
    (0 1 0)
    (0.5 1 0)
    (1 1 0)
    (0 0 0.1)
    (0.5 0 0.1)
    (1 0 0.1)
    (0 0.5 0.1)
    (0.5 0.5 0.1)
    (1 0.5 0.1)
    (0 1 0.1)
    (0.5 1 0.1)
    (1 1 0.1)
);

blocks
(
    hex (0 1 4 3 9 10 13 12) (10 10 1) simpleGrading (2 2 1)
    hex (1 2 5 4 10 11 14 13) (10 10 1) simpleGrading (0.5 2 1)
    hex (3 4 7 6 12 13 16 15) (10 10 1) simpleGrading (2 0.5 1)
    hex (4 5 8 7 13 14 17 16) (10 10 1) simpleGrading (0.5 0.5 1)
);

edges
(
);

boundary
(
    movingWall
    {
        type wall;
        faces
        (
            (6 15 16 7)
            (7 16 17 8)
        );
    }
    fixedWalls
    {
        type wall;
        faces
        (
            (3 12 15 6)
            (0 9 12 3)
            (0 1 10 9)
```

```
            (1 2 11 10)
            (2 5 14 11)
            (5 8 17 14)
        );
    }
    frontAndBack
    {
        type empty;
        faces
        (
            (0 3 4 1)
            (1 4 5 2)
            (3 6 7 4)
            (4 7 8 5)
            (9 10 13 12)
            (10 11 14 13)
            (12 13 16 15)
            (13 14 17 16)
        );
    }
);

mergePatchPairs
(
);

// ************************************************************************* //
```

いったんこのケースの *blockMeshDict* ファイルを理解しておけば，後はコマンドラインから blockMesh を実行できます．2.1.2 項に示した paraFoam を使用することで勾配付けされたメッシュを見ることができます．

2.1.6.2　計算時間，時間ステップの変更

最も速度が速くて最もサイズの小さいセルが上面に隣接するセルであり，2.1.1.4 で示したように，この上面のセルにおいて Courant 数が最大となります．このようなことから上面に隣接するセルの大きさを見積もることは，本ケースにて適当な時間ステップを計算する上で有効です．

一様でないメッシュ勾配を使用している場合，blockMesh は形状に関する数列をもちいてセルの大きさを算出します．長さ l に沿って，最初と最後のセルとの間に，比 R の n 個の計算セルが必要であるならば，もっとも小さいセルの大きさは，次のように与えられます．

$$\Delta x_{\mathrm{s}} = l\frac{r-1}{\alpha r - 1} \tag{2.5}$$

ここで，r はあるセルの大きさとその隣のセルの大きさとの比であり，次式で表されます．

$$r = R^{\frac{1}{n-1}} \tag{2.6}$$

そして，

$$\alpha = \begin{cases} R & \text{for } R > 1, \\ 1 - r^{-1} + r^{-1} & \text{for } R < 1. \end{cases} \tag{2.7}$$

cavityGrade ケースにおいては，各方向のセルの数は 10 であり，最大セルと最小セルとの比は 2，ブロックの縦横は 0.05 m です．したがって，最小セルサイズは 3.45 mm となります．式 (2.2) から時間ステップは，クラーン数を 1 以下に抑えるために 3.45 ms 以下にしなければなりません．有意な解析結果を得るためには，時間ステップ `deltaT` を 2.5 ms まで小さくし，`writeInterval` を 40 とします．これより解析結果は 0.1 s ごとに書き出されることとなります．

このように，各設定に対応したファイルを編集することにより，ケースディクショナリの各種条件を変更することができます．ここで時間ないし計算経過の書き出しを操作したいならば，*/cavityGrade/system/controlDict* ファイル内にそれらのパラメータは納められており，任意のエディタでこのファイルを開くことができます．先に述べたように，計算を収束させるための保証として，このケースでは時間ステップ `deltaT` は `0.25e-3` に，`writeInterval` は 40 とします．

`startTime` はその cavityFine ケースの最終的な条件，すなわち 0.7 に設定される必要があります．cavity と cavityFine が規定された実行時間の中でよく収束させるためには，cavityGrade ケースのための実行時間を 0.1 s に設定，すなわち `endTime` を 0.8 とします．

2.1.6.3 解析場のマッピング

2.1.5.3 にあるように mapFields を使用して，cavityFine ケースの最終的な結果を cavityGrade ケースのメッシュにマッピングします．以下のように *cavityGrade* ディレクトリに入り，mapFields を実行してください．

```
cd $FOAM_RUN/tutorials/incompressible/icoFoam/cavityGrade
mapFields ../cavityFine -consistent
```

今度は，ケースディレクトリから icoFoam を実行して，計算実行時の情報をモニタリングします．そして，2.1.5.6 と 2.1.5.7 で説明した後処理ツールを使って，収束した結果を見たり，他の結果と比較します．

2.1.7 Reynolds 数の増大

これまで解いたケースは Reynolds 数が 10 でした．これは大変に低い条件であり，したがってキャビティの底部中央に小さな二次渦を伴うのみで，迅速に安定解を導くことができました．しかし，ここで Reynolds 数を 100 に上げると，収束解を得るのにより長い時間を要することになります．そこで cavity ケースのメッシュを初期条件として使用することとします．*cavity* ケースディレクトリを *cavityHighRe* という名前でコピーします．

```
cd $FOAM_RUN/tutorials/incompressible/icoFoam
cp -r cavity cavityHighRe
```

2.1.7.1 後処理

cavityHighRe ケースに入り，*transportProperties* ディクショナリを編集します．Reynolds 数を 10 倍に増加させるためには，動粘性係数を 10 分の 1，すなわち $1 \times 10^{-3}\,\mathrm{m^2 s^{-1}}$ まで減らす必要があります．これで cavity ケースの実行結果からリスタートして，このケースを実行できます．これを実行するために，`startFrom` キーワードを `latestTime` にオプションを切り替えることにより，icoFoam は，最新の時間ディレクトリを初期データとして使用します（例えば 0.5）．`endTime` は 2 s に設定し，本ケースを保存します．

2.1.7.2　コードの実行

まずはケースディレクトリから icoFoam を実行し，ランタイム情報を見ます．バックグラウンドでジョブを実行するときには，以下の UNIX コマンドが便利です．

`nohup`　ユーザがログアウト後も稼働し続けるコマンド

`nice`　カーネル・スケジューラのジョブの優先順位を変えるコマンド．−20 が最優先で，19 は最も低い優先度．

これらのコマンドは，例えば，ユーザがリモートマシンでケースを実行できるよう設定し，頻繁にモニタしなくてもいいような場合，リモートマシンではケース実行をあまり優先させたくないでしょうが，そのような場合に便利です．その場合，ユーザは `nohup` コマンドで稼働しているリモートマシンをログアウトしてジョブを実行し続けることができます．一方，`nice` は優先度を 19 に設定します．試しに，以下のようにコマンドを実行してみましょう．

```
cd $FOAM_RUN/tutorials/incompressible/icoFoam
nohup nice -n 19 icoFoam > log &
cat log
```

お気づきかもしれませんが，前述の解析方法では icoFoam は，速度 U の計算が止まっても，それよりもずっと長い間もしくは解析が終わるまで圧力 p の計算をし続けていました．実際には，icoFoam がいったん U の計算をやめ，p の初期残差が *fvSolution* ディクショナリで設定された許容値（通常は 10^{-6}）を下回ると結果が効率的に収束するので，フィールド・データをいったん時間ディレクトリに書き出して計算を止めることができます．例として，cavityHighRen ケースの収束の *log* ファイルを以下に示します．示したとおり，1.62 s 後に速度はすでに収束し，初期の圧力残差は小さくなります．*log* において `No Iterations 0` は，U の計算が止まったことを示しています．

```
Time = 1.43

Courant Number mean: 0.221921 max: 0.839902
smoothSolver: Solving for Ux, Initial residual = 8.73381e-06, Final residual = 8.73381e-06, No Iterations 0
smoothSolver: Solving for Uy, Initial residual = 9.89679e-06, Final residual = 9.89679e-06, No Iterations 0
DICPCG: Solving for p, Initial residual = 3.67506e-06, Final residual = 8.62986e-07, No Iterations 4
time step continuity errors : sum local = 6.57947e-09, global = -6.6679e-19, cumulative = -6.2539e-18
DICPCG: Solving for p, Initial residual = 2.60898e-06, Final residual = 7.92532e-07, No Iterations 3
time step continuity errors : sum local = 6.26199e-09, global = -1.02984e-18, cumulative = -7.28374e-18
ExecutionTime = 0.37 s ClockTime = 0 s

Time = 1.435

Courant Number mean: 0.221923 max: 0.839903
smoothSolver: Solving for Ux, Initial residual = 8.53935e-06, Final residual = 8.53935e-06, No Iterations 0
smoothSolver: Solving for Uy, Initial residual = 9.71405e-06, Final residual = 9.71405e-06, No Iterations 0
DICPCG: Solving for p, Initial residual = 4.0223e-06, Final residual = 9.89693e-07, No Iterations 3
time step continuity errors : sum local = 8.15199e-09, global = 5.33614e-19, cumulative = -6.75012e-18
DICPCG: Solving for p, Initial residual = 2.38807e-06, Final residual = 8.44595e-07, No Iterations 3
time step continuity errors : sum local = 7.48751e-09, global = -4.42707e-19, cumulative = -7.19283e-18
ExecutionTime = 0.37 s ClockTime = 0 s
```

2.1.8　高 Reynolds 数流れ

では，paraFoam による結果を確認し，速度ベクトルを表示してください．計算領域の角における二次渦が幾分増大していることがわかります．このようなとき，ユーザは粘性係数を下げることにより Reynolds 数を増大させた計算ケースを再度実行できます．渦の数が増加するに

ともない，より複雑な流れを解くために当該領域でのメッシュ解像度を上げる必要がでてきます．さらに，Reynolds 数は収束に要する時間を増加させます．このような場合，残差をモニタし，解を収束させるために `endTime` を延長したほうがよいでしょう．

空間および時間解像度の増加を要することは，流れが乱流域に移行するという非現実的な状態となり，解法の安定性の問題が生じることとなります．もちろん，多くの工学的な問題は極めて高い Reynolds 数条件となっており，したがって，乱流挙動を直接解くのに多くのコストを負担することとなり，実行不可能であります．そのかわりに，Reynolds 平均シミューレション (RAS) 乱流モデルが平均流れの挙動を解くのに用いられ，乱れの統計値が計算されています．壁関数を伴う標準 k–ε モデルが本チュートリアルの上面が移動するキャビティケース（Reynolds 数 10^4）を解くのに用いられています．二つの追加変数が解かれています．それは，乱流エネルギ k，乱流消散速度 ε です．乱流のための追加の方程式およびモデルは pisoFoam と呼ばれる OpenFOAM ソルバにおいて実行されます．

2.1.8.1 前処理

$FOAM_RUN/tutorials/incompressible/pisoFoam/ras ディレクトリの *cavity* ケースに移動します．これまでと同様に，blockMesh を走らせ，メッシュを生成します．壁関数付き標準 k–ε モデルを用いる場合は，壁近傍のセルにおける流れがモデル化されることにより，壁方向へのメッシュ勾配は必ずしも必要ではありません．

OpenFOAM では，様々な壁関数モデルを利用することができ，それぞれのパッチの境界条件として設定します．これにより，壁面ごとに異なる壁関数モデルを適用することが可能になります．壁関数の選択は，乱流粘性係数 ν_t のファイル *0/nut* で指定します．

```
17
18 dimensions      [0 2 -1 0 0 0 0];
19
20 internalField   uniform 0;
21
22 boundaryField
23 {
24     movingWall
25     {
26         type            nutkWallFunction;
27         value           uniform 0;
28     }
29     fixedWalls
30     {
31         type            nutkWallFunction;
32         value           uniform 0;
33     }
34     frontAndBack
35     {
36         type            empty;
37     }
38 }
39
40
41 // ************************************************************************* //
```

このケースでは標準的な壁関数を採用し，`movingWall` と `fixedWalls` のパッチに対して `nutWallFunction` タイプを指定しています．これ以外の壁関数モデルとしては，粗壁面の壁関数 `nutRoughWallFunction` などがあります．

次に，k と ε のファイル (*0/k* と *0/epsilon*) を開き，境界条件を確かめます．壁タイプの境界条

件の選択には，ε については epsilonWallFunction 境界条件を，k については kqRWallFunction を指定します．後者は乱流運動エネルギの表現 k，q，あるいは Reynolds 応力 R のいずれにも適用できる一般的な壁関数です．k，ε の初期条件には，速度変動 $\boldsymbol{U}'$ と乱流長さスケール l から推測した値を設定します．k と ε は，これらのパラメタを用いて次式で表されます．

$$k = \frac{1}{2}\overline{\boldsymbol{U}' \cdot \boldsymbol{U}'} \tag{2.8}$$

$$\varepsilon = \frac{C_\mu^{0.75} k^{1.5}}{l} \tag{2.9}$$

ここで C_μ は k–ε モデルの定数であり，その値は 0.09 です．Cartesian 座標系では k は，

$$k = \frac{1}{2}({U'_x}^2 + {U'_y}^2 + {U'_z}^2) \tag{2.10}$$

で表されます．各項は x，y，z 方向速度乱れ成分です．ここで，初期乱流が等方的であると仮定します．例えば，${U'_x}^2 = {U'_y}^2 = {U'_z}^2$ となり，これら速度は上面速度の 5 % に等しく，また，乱流長さスケール l はボックス幅 0.1 m の 20 % に等しいとすると，次のように表されます．

$$U'_x = U'_y = U'_z = \frac{5}{100} 1\,\mathrm{m\,s^{-1}} \tag{2.11}$$

$$\Rightarrow k = \frac{3}{2}\left(\frac{5}{100}\right)^2\ \mathrm{m^2 s^{-2}} = 3.75 \times 10^{-3}\,\mathrm{m^2 s^{-2}} \tag{2.12}$$

$$\varepsilon = \frac{C_\mu^{0.75} k^{1.5}}{l} \approx 7.65 \times 10^{-4}\,\mathrm{m^2 s^{-3}} \tag{2.13}$$

上記のとおり k，ε を設定してください．U と p に対する初期条件は前と同じように，それぞれ $(0, 0, 0)$ と 0 です．

OpenFOAM で提供されている乱流モデルには，例えば RAS や large-edy simulation (LES) のような，さまざまな手法があります．ほとんどの非定常ソルバでは，乱流のモデリング手法は実行時に *turbulenceProperties* ディクショナリの simulationType キーワードで選択できます．このファイルは *constant* ディレクトリの中に見つかります．

```
17
18  simulationType  RASModel;
19
20
21  // ************************************************************************* //
```

simulationType の選択肢は laminar，RASModel，そして LESModel です．このケースで選択されている RASModel の場合，RAS モデリングの選択は *RASProperties* ファイルに記述します．このファイルも同じく *constant* ディレクトリにあります．乱流モデルは表 3.9 に示されている多くの使用可能なモデルから，RASModel エントリで選択します．ここでは，標準 k–ε モデルである kEpsilon を選択します．turbulence のスイッチが on になっていることも確認します．乱流モデルに必要な係数には，それぞれのコードの中でデフォルト値が与えられています．printCoeffs というオプションのスイッチを on にすると，実行時に乱流モデルが呼ばれたときに，これらのデフォルト値が標準出力，すなわちターミナルに出力されるようになります．これらの係数は，モデル名に Coeffs をつけた名前（たとえば kEpsilon モデルなら kEpsilonCoeffs）のサブディクショナリとして表示されます．モデルの係数は，必要に応じて *RASProperties* ディ

クショナリにサブディクショナリを追加（コピー&ペースト）し，値を適宜調整することで変更することができます．

次いで，*transportProperties* ディクショナリの層流動粘性係数を設定します．Reynolds 数 10^4 を実現するために，式 (2.1) の Reynolds 数の定義式に示されるように，動粘性係数を $10^{-5}\,\mathrm{m^2 s^{-1}}$ にする必要があります．

最後に，*controlDict* の `startTime`，`stopTime`，`deltaT`，そして `writeInterval` を設定します．Courant 数の制限を満たすために `deltaT` を 0.005 s に設定し，`endTime` は 10 s とします．

2.1.8.2 コードの実行

ケースディレクトリに入り，ターミナルで `pisoFoam` とタイプすることで pisoFoam を実行します．粘性が小さいこの計算ケースでは，移動している上面近傍の境界層は極めて薄く，そして，上面に面するセルは比較的大きいことから，上面速度よりもそれらセル中心の流体速度は極めて小さいです．事実，100 時間ステップ後，上面に隣接したセルにおける速度は，上限である $0.2\,\mathrm{m\,s^{-1}}$ 程度です．したがって最大 Courant 数は 0.2 以上にはなりません．Courant 数がより 1 に近接するように時間ステップを大きくし，解析時間を増やすことは理にかなっています．したがって，`deltaT` を 0.02 s にセットしなおし，これに伴い，`startFrom` を `latestTime` にセットします．本操作は，pisoFoam が最新のディレクトリ，例えば *10.0*，からスタートデータを読み込むように指示するものです．`endTime` は層流条件よりも収束に時間を要するため，20 s にセットします．従来どおり計算をリスタートし，解析の収束をモニタします．解析が進行したら，連続した時間における結果を見てください．そして解析が安定状態に収束するか，もしくは周期的に振動しているか確認してください．後者の場合には，収束は決して起こりませんが，結果が不正確であるという意味ではありません．

2.1.9 ケース形状の変更

計算ケースの形状を変更し，新たな解析を行いたい場合，新たな解析のスタート条件としてオリジナルの解析の全てないし一部を保持しておくことは有効でしょう．しかし，これは少し複雑になります．なぜなら，オリジナルの解析の物理量が，新しい解析ケースの物理量と一致しないからです．しかし，mapFields ユーティリティは，形状や境界のタイプもしくはその両者が不一致な場を位置づけることができます．

例であるように，*icoFoam* ディレクトリ内にある *cavityClipped* ケースを開きます．このケースは，標準的な cavity ケースからなりますが，底部右側，長さ 0.04 m の正方形を除いたものであり，*blockMeshDict* は以下のようになっています．

```
17  convertToMeters 0.1;
18
19  vertices
20  (
21      (0 0 0)
22      (0.6 0 0)
23      (0 0.4 0)
24      (0.6 0.4 0)
25      (1 0.4 0)
26      (0 1 0)
27      (0.6 1 0)
28      (1 1 0)
29
30      (0 0 0.1)
```

```
31      (0.6 0 0.1)
32      (0 0.4 0.1)
33      (0.6 0.4 0.1)
34      (1 0.4 0.1)
35      (0 1 0.1)
36      (0.6 1 0.1)
37      (1 1 0.1)
38
39  );
40
41  blocks
42  (
43      hex (0 1 3 2 8 9 11 10) (12 8 1) simpleGrading (1 1 1)
44      hex (2 3 6 5 10 11 14 13) (12 12 1) simpleGrading (1 1 1)
45      hex (3 4 7 6 11 12 15 14) (8 12 1) simpleGrading (1 1 1)
46  );
47
48  edges
49  (
50  );
51
52  boundary
53  (
54      lid
55      {
56          type wall;
57          faces
58          (
59              (5 13 14 6)
60              (6 14 15 7)
61          );
62      }
63      fixedWalls
64      {
65          type wall;
66          faces
67          (
68              (0 8 10 2)
69              (2 10 13 5)
70              (7 15 12 4)
71              (4 12 11 3)
72              (3 11 9 1)
73              (1 9 8 0)
74          );
75      }
76      frontAndBack
77      {
78          type empty;
79          faces
80          (
81              (0 2 3 1)
82              (2 5 6 3)
83              (3 6 7 4)
84              (8 9 11 10)
85              (10 11 14 13)
86              (11 12 15 14)
87          );
88      }
89  );
90
91  mergePatchPairs
92  (
93  );
94
95  // ************************************************************************* //
```

blockMesh を実行してメッシュを生成します．パッチは cavity ケースと同様に設定されています．物理量の適用の過程を明確にするために，元となるケース cavity で `movingWall` であった上側の壁は `lid` という名前に変更されています．

パッチが一致しない場合，すべての物理量のデータが元のケースからマップされるという保証はありません．残っているデータは元のケースと同一であるべきです．したがってマッピングする前に時間のディレクトリに物理量のデータが存在している必要があります．*controlDict* の `startTime` が 0.5 s に設定されているので cavityClipped ケースにおけるマッピングは時刻 0.5 s に予定されています．したがって初期状態の物理量のデータ，たとえば時刻 0 からをコピーする必要があります．

```
cd $FOAM_RUN/tutorials/incompressible/icoFoam/cavityClipped
cp -r 0 0.5
```

データをマッピングする前に 0.5 s における形状と物理量の状況をみておきましょう．

速度場と圧力場を cavity から cavityClipped にマップしようとしています．パッチが一致しないため，*system* ディレクトリの *mapFieldsDict* を編集する必要があります．`patchMap` と `cuttingPatches` という二つの入力項目があります．`patchMap` リストは元となる物理量のパッチとマッピング対象となる物理量のパッチを含みます．対象物理量のパッチに元となる物理量のパッチの値を引き継ぎたいときに利用します．cavityClipped において `lid` の境界値を cavity の `movingWall` から引き継ぎたいので次のように *patchMap* に記述します．

```
patchMap
(
   lid movingWall
);
```

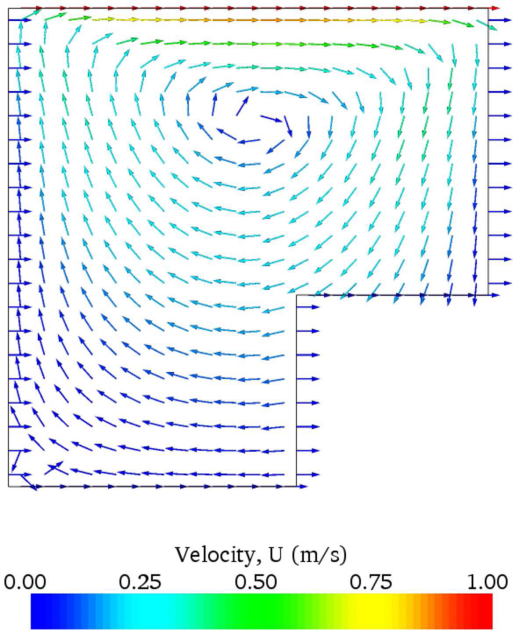


図 2.13　cavity ケースで解いた速度場を cavityClipped 上にマッピングした図

`cuttingPatches` リストは，対象パッチを削除した，元の場の内部の値を写像した対象のパッチを含みます．本ケースでは，`fixedWalls` を内挿プロセスの実例説明に用いることとします．

```
cuttingPatches
(
```

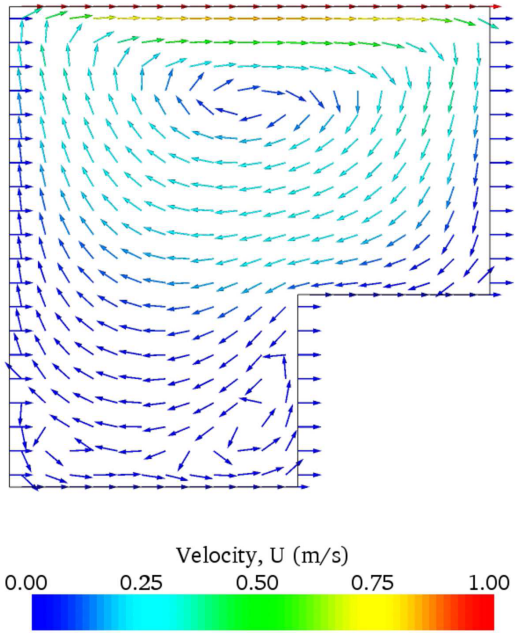


図 2.14　速度場の cavityClipped の解法

```
    fixedWalls
);
```

ここで，mapFields を次のコマンドから実行することができます．

```
mapFields ../cavity
```

図 2.13 に示すような場を確認することができます．境界パッチは，期待したように元のケースからの値が引き継がれています．この実例において，`fixedWalls` パッチの速度を $(0,0,0)$ にリセットしたい場合があります．このときは，*U* をエディタで開き，`fixedWalls` を `nonuniform` から `uniform (0,0,0)` に変更します．そして，icoFoam を実行しましょう．

2.1.10　修正した形状の後処理

最初と最後の解析の比較のために，この解析ケースのベクトル図を，最初の時刻は 0.5 s，次いで 0.6 s のように作成することができます．さらに，幾何形状のアウトラインも示しますが，これは 2 次元のケースでは少し注意が必要です．Filter メニューから `Extract Parts` を選択し，Parameter パネルで，興味のあるパッチ，つまり `lid` と `fixedWalls` をハイライトします．Apply ボタンをクリックし，Display パネルで `Wireframe` の選択すれば，ジオメトリのうち選択したものを表示することができます．図 2.14 は，パッチを黒で表示し，修正した形状の底部角部分において形成される渦を示しています．

2.2　穴あき板の応力解析

本チュートリアルでは，中央に円形の穴を有する正方形板の線形弾性定常応力解析における前処理，実行および後処理の方法を述べます．板の大きさは，辺長 4 m および穴の半径 0.5 m です．さらに図 2.15 に示すように，板の左右端には $\sigma = 10\,\mathrm{kPa}$ の一様表面力が負荷されてい

ます．本形状においては二つの対称面が存在するため，解析領域は図 2.15 のグレーで示した板全体の 4 分の 1 の部分のみをカバーすれば十分です．

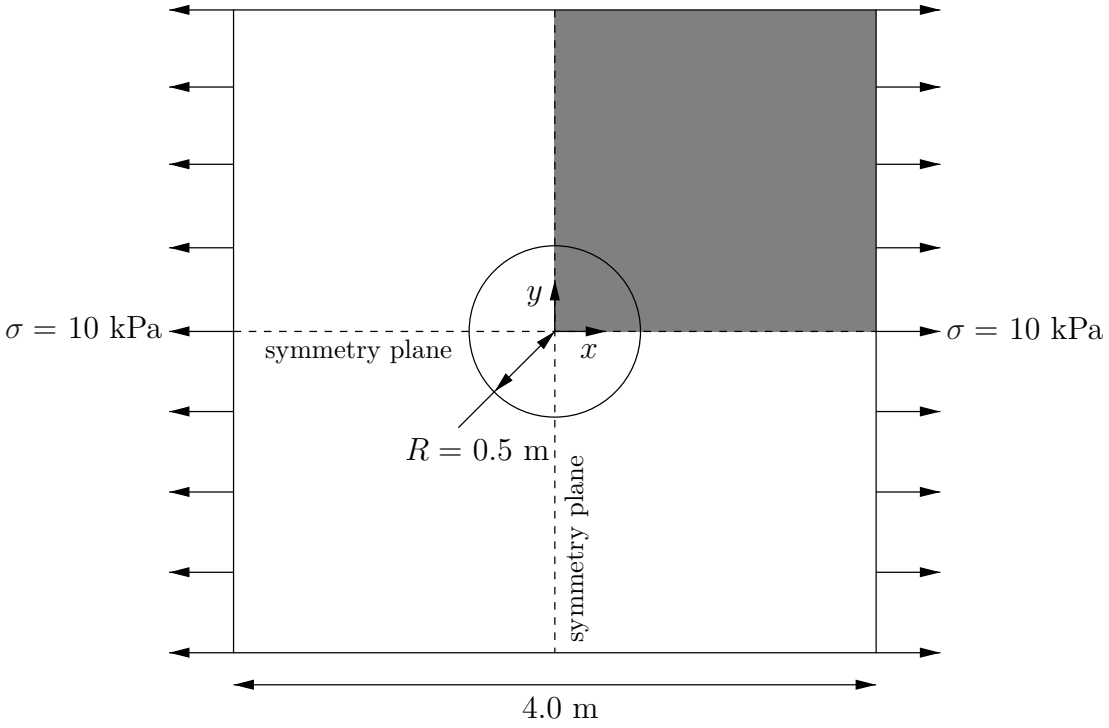


図 2.15 穴あき板の形状

本問題では，板の面内に応力が負荷されるため，2 次元問題として近似することができます．Cartesian 座標系においては，この構造の 3 番目の次元についての振る舞いを考察するにあたって，以下の二つの仮定をすることができます．(1) 平面応力条件：2 次元平面内以外の方向にはたらく応力成分を無視できるものと仮定します．(2) 平面ひずみ条件：2 次元平面内以外の方向にはたらくひずみ成分を無視できるものと仮定します．本ケースのように 3 次元方向に薄い固体に対しては，平面応力条件が適当です．なお平面ひずみ条件は，3 次元方向に厚い固体に対して適用されます．

円形の穴を有する無限大に大きく薄い板への負荷に対しては，解析解が存在します．垂直な対称面における法線方向応力の解は以下となります．

$$(\sigma_{xx})_{x=0} = \begin{cases} \sigma\left(1 + \dfrac{R^2}{2y^2} + \dfrac{3R^4}{2y^4}\right) & \text{for } |y| \geq R \\ 0 & \text{for } |y| < R \end{cases} \tag{2.14}$$

シミュレーションの実行結果をこの解析解と比較することとしましょう．チュートリアルの最後に，メッシュの解像度および非等間隔化に対する解の感度を調べ，また，穴に対する板の大きさを大きくすることで無限大板に対する解析解と有限板に対する本問題の解を比較して誤差を見積もることができるように演習問題を用意しています．

2.2.1 メッシュ生成

解析領域は 4 ブロックからなり，そのうちのいくつかは円弧形の端部を有します．x–y 平面におけるメッシュブロックの構造を図 2.16 に示します．2.1.1.1 で述べたように，2 次元として

扱われるようなケースであっても，OpenFOAM では全てのジオメトリが 3 次元で生成されます．したがって z 方向のブロックの大きさを設定しなければなりませんので，ここでは 0.5 m とします．表面力境界条件は力でなく応力で指定されますので，断面積すなわち z 方向の大きさは解に影響を与えません．

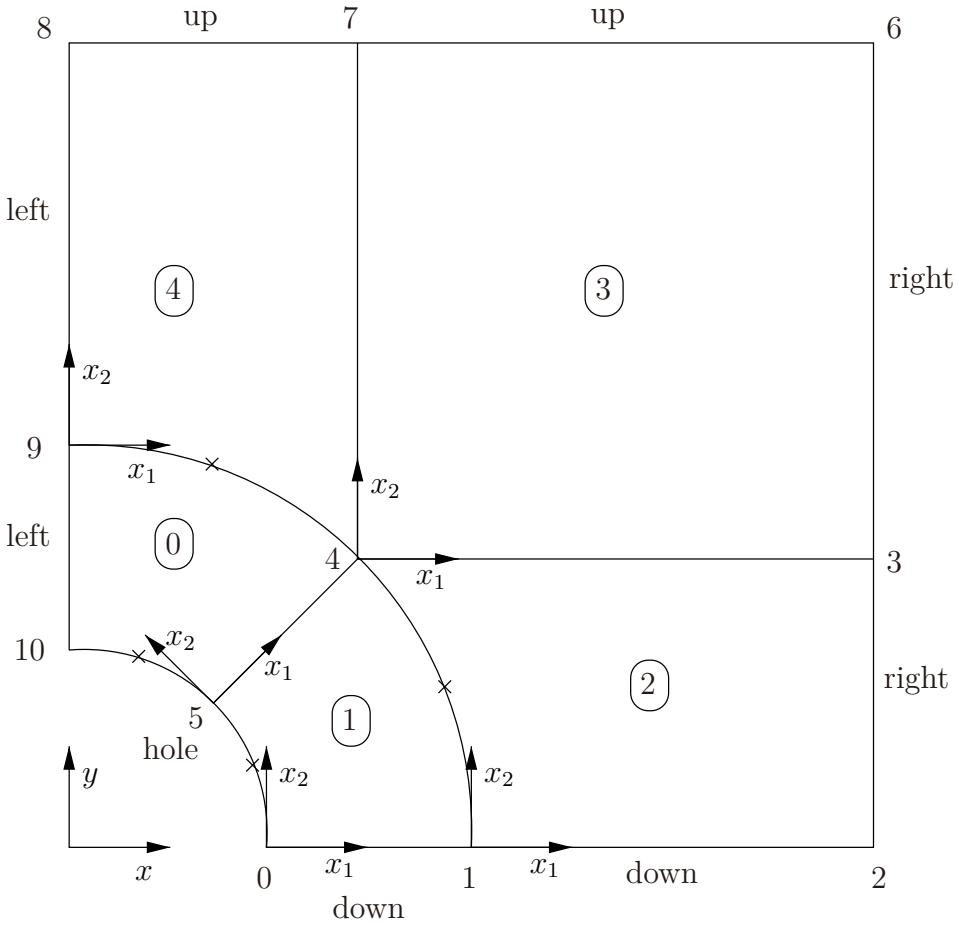


図 2.16　穴あき板解析のためのメッシュのブロック構造

$FOAM_RUN/tutorials/stressAnalysis/solidDisplacementFoam ディレクトリの *plateHole* ケースに移動し，*plateHole* ケースの *constant/polyMesh/blockMeshDict* をエディタで開きます．*blockMeshDict* ディクショナリのエントリを以下に示します．

```
17  convertToMeters 1;
18
19  vertices
20  (
21      (0.5 0 0)
22      (1 0 0)
23      (2 0 0)
24      (2 0.707107 0)
25      (0.707107 0.707107 0)
26      (0.353553 0.353553 0)
27      (2 2 0)
28      (0.707107 2 0)
29      (0 2 0)
30      (0 1 0)
31      (0 0.5 0)
32      (0.5 0 0.5)
33      (1 0 0.5)
34      (2 0 0.5)
```

```
35      (2 0.707107 0.5)
36      (0.707107 0.707107 0.5)
37      (0.353553 0.353553 0.5)
38      (2 2 0.5)
39      (0.707107 2 0.5)
40      (0 2 0.5)
41      (0 1 0.5)
42      (0 0.5 0.5)
43  );
44
45  blocks
46  (
47      hex (5 4 9 10 16 15 20 21) (10 10 1) simpleGrading (1 1 1)
48      hex (0 1 4 5 11 12 15 16) (10 10 1) simpleGrading (1 1 1)
49      hex (1 2 3 4 12 13 14 15) (20 10 1) simpleGrading (1 1 1)
50      hex (4 3 6 7 15 14 17 18) (20 20 1) simpleGrading (1 1 1)
51      hex (9 4 7 8 20 15 18 19) (10 20 1) simpleGrading (1 1 1)
52  );
53
54  edges
55  (
56      arc 0 5 (0.469846 0.17101 0)
57      arc 5 10 (0.17101 0.469846 0)
58      arc 1 4 (0.939693 0.34202 0)
59      arc 4 9 (0.34202 0.939693 0)
60      arc 11 16 (0.469846 0.17101 0.5)
61      arc 16 21 (0.17101 0.469846 0.5)
62      arc 12 15 (0.939693 0.34202 0.5)
63      arc 15 20 (0.34202 0.939693 0.5)
64  );
65
66  boundary
67  (
68      left
69      {
70          type symmetryPlane;
71          faces
72          (
73              (8 9 20 19)
74              (9 10 21 20)
75          );
76      }
77      right
78      {
79          type patch;
80          faces
81          (
82              (2 3 14 13)
83              (3 6 17 14)
84          );
85      }
86      down
87      {
88          type symmetryPlane;
89          faces
90          (
91              (0 1 12 11)
92              (1 2 13 12)
93          );
94      }
95      up
96      {
97          type patch;
98          faces
99          (
100             (7 8 19 18)
101             (6 7 18 17)
102         );
103     }
104     hole
105     {
```

```
106             type patch;
107             faces
108             (
109                 (10 5 16 21)
110                 (5 0 11 16)
111             );
112         }
113         frontAndBack
114         {
115             type empty;
116             faces
117             (
118                 (10 9 4 5)
119                 (5 4 1 0)
120                 (1 4 3 2)
121                 (4 7 6 3)
122                 (4 9 8 7)
123                 (21 16 15 20)
124                 (16 11 12 15)
125                 (12 13 14 15)
126                 (15 14 17 18)
127                 (15 18 19 20)
128             );
129         }
130     );
131
132     mergePatchPairs
133     (
134     );
135
136     // ************************************************************************* //
```

ここまで前のチュートリアルのように直線的なエッジの形状を対象としてきましたが，本チュートリアルでは曲線のエッジについて定義する必要があります．edges のキーワードエントリ（曲線エッジのリスト）内で曲線エッジが定義されています．それらのリストの構文では，最初に arc，simpleSpline，polyLine などの曲線タイプが示されていますが，さらに詳しくは 5.3.1 項を参照してください．この例題ではすべてのエッジが円弧なので arc を使用します．曲線を arc で定義する際は始点と終点および円弧上の点の 3 点によって指定します．

この *blockMeshDict* に含まれるブロック全てが同じ方向を向いているわけではありません．図 2.16 に示すように，ブロック 0 の x_2 方向はブロック 4 の $-x_1$ 方向と同じになっています．このためブロック界面でセルが矛盾なく合うよう，それぞれのブロックにおけるセルの番号および順序を決定する際には注意を払わねばなりません．

プレートの全側面，穴の面，前後面の六つのパッチが定義されます．そのうち左の面 (left) と下の面 (down) は対称面です．このようなことはジオメトリ上の制限であるため，ただの場の境界条件とするよりはメッシュの定義の中に組み込んで作ります．よって，このパッチは *blockMeshDict* 内の特別な SymmetryPlane タイプを使って定義するとよいでしょう．frontAndBack パッチは 2 次元問題の場合は無視される面を示しています．これは先ほども言ったようにジオメトリ上の制限なので，*blockMeshDict* 内の empty タイプを使って定義しましょう．境界条件に関してさらに詳しくは 5.2.1 項を参照してください．そのほかのパッチは通常の patch タイプです．メッシュは blockMesh を使って生成し，2.1.2 項に述べたようにして paraFoam で見ることができます．メッシュは図 2.17 のようになります．

2.2.1.1　境界および初期条件

メッシュの生成ができたら初期条件と境界条件を設定します．熱抵抗を考慮しない応力解析では，変位 D のみ設定する必要があります．*0/D* のファイルは以下のようになります．

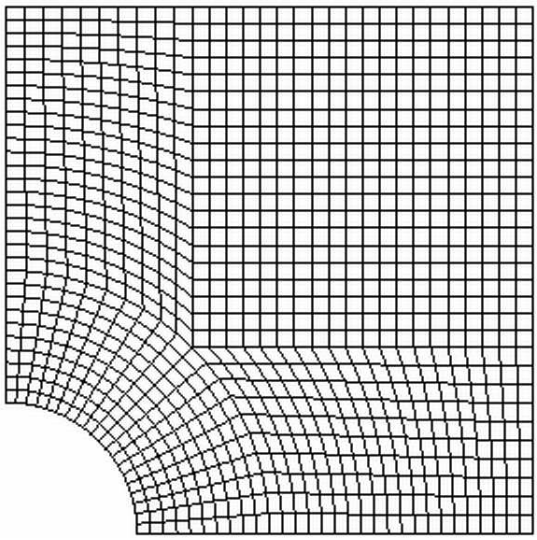


図 2.17 穴あき板問題のための解析メッシュ

```
17  dimensions      [0 1 0 0 0 0 0];
18
19  internalField   uniform (0 0 0);
20
21  boundaryField
22  {
23      left
24      {
25          type            symmetryPlane;
26      }
27      right
28      {
29          type            tractionDisplacement;
30          traction        uniform ( 10000 0 0 );
31          pressure        uniform 0;
32          value           uniform (0 0 0);
33      }
34      down
35      {
36          type            symmetryPlane;
37      }
38      up
39      {
40          type            tractionDisplacement;
41          traction        uniform ( 0 0 0 );
42          pressure        uniform 0;
43          value           uniform (0 0 0);
44      }
45      hole
46      {
47          type            tractionDisplacement;
48          traction        uniform ( 0 0 0 );
49          pressure        uniform 0;
50          value           uniform (0 0 0);
51      }
52      frontAndBack
53      {
54          type            empty;
55      }
56  }
57
58  // ************************************************************************* //
```

まず，変位の初期条件が $(0, 0, 0)$ m になっています．*Constant/polyMesh/boundaries* のメッシュの記述にあるように，`left` と `down` のパッチは `type` が `symmetryPlane` である必要があります．

同様に frontAndBack は empty になります．

その他のパッチは表面力境界条件です．表面力境界条件は，(1) キーワード traction で表される，境界面における表面力ベクトル，(2) キーワード pressure で表される，境界面の法線方向に働く表面力となる（外向きの場合は負値となる）圧力，の線形結合で指定されます．up および hole パッチは表面力ゼロであるため，表面力ベクトルおよび圧力ともにゼロが設定されます．right パッチについては，図 2.24 に示すように，表面力ベクトルは $(1e4, 0, 0)$ Pa，圧力は 0 Pa が設定されます．変位の初期条件は全て $(0, 0, 0)$ m が設定されます．

2.2.1.2　機械的性質

本ケースにおける物性値は *mechanicalProperties* ディクショナリによって設定します．本問題においては，表 2.1 に示す鋼の機械的性質を指定する必要があります．さらに本ディクショナリで planeStress を yes に設定しなければなりません．

物性	単位	キーワード	値
密度	$\mathrm{kg\,m^{-3}}$	rho	7854
Young 率	Pa	E	2×10^{11}
Poisson 比	—	nu	0.3

表 2.1　鋼の機械的性質

2.2.1.3　熱的性質

運動によって発生する熱応力によって運動方程式と連成した熱方程式を解くことができるよう，solidDisplacementFoam ソルバには温度場を表す変数 T が存在します．*thermalProperties* ディクショナリの thermalStress スイッチによって，OpenFOAM が熱方程式を解くべきかどうかを実行時に指定します．また本ディクショナリによって，本ケースすなわち表 2.2 に示す鋼の熱的性質を指定します．

物性	単位	キーワード	値
比熱容量	$\mathrm{J\,kg^{-1}K^{-1}}$	C	434
熱伝導率	$\mathrm{W\,m^{-1}K^{-1}}$	k	60.5
熱膨張率	$\mathrm{K^{-1}}$	alpha	1.1×10^{-5}

表 2.2　鋼の熱的性質

本ケースにおいては熱の方程式は解きません．したがって *thermalProperties* ディクショナリにおける thermalStress キーワードエントリは no に設定します．

2.2.1.4　制御

通常どおり，解法の制御に関する情報は *controlDict* ディクショナリから読み込まれます．本ケースでは，startTime は 0 です．本ケースは定常状態ですので，時間刻みは重要ではありません．このような状況では，定常状態のケースにおける反復回数カウンタとして働くよう，時間刻み deltaT を 1 に設定するのが最善です．このようにした場合，本ケースで 100 に設定した endTime は反復回数の上限として働きます．writeInterval は 20 に設定します．

controlDict のエントリは以下のようになります．

```
17
18  application     solidDisplacementFoam;
19
20  startFrom       startTime;
```

```
21
22 startTime       0;
23
24 stopAt          endTime;
25
26 endTime         100;
27
28 deltaT          1;
29
30 writeControl    timeStep;
31
32 writeInterval   20;
33
34 purgeWrite      0;
35
36 writeFormat     ascii;
37
38 writePrecision  6;
39
40 writeCompression off;
41
42 timeFormat      general;
43
44 timePrecision   6;
45
46 graphFormat     raw;
47
48 runTimeModifiable true;
49
50
51 // ************************************************************************* //
```

2.2.1.5 離散化スキームおよび線形方程式ソルバ制御

次は *fvSchemes* ディクショナリについて見てみましょう．まず，この問題は定常状態ですので，timeScheme における時間微分としては steadyState を選択します．これによって時間微分項がオフの状態になります．全てのソルバが定常状態および過渡的状態の双方に対して適用可能な訳ではありませんが，solidDisplacementFoam は基本的なアルゴリズムが双方のシミュレーションともに共通であるため，双方に適用可能となっています．

線形弾性応力解析における運動方程式には，変位の勾配を含む陽な項がいくつか存在します．この勾配を正確かつ滑らかに評価できれば，良い計算結果が得られます．通常，有限体積法における離散化は，Gauss の定理に基づいています．Gauss 法は大抵の目的においては十分に正確ですが，本ケースにおいては最小二乗法を使用することとします．したがって *fvSchemes* ディクショナリを開き，grad(U) 勾配離散化スキームとして leastSquares を選択してください．

```
17
18 d2dt2Schemes
19 {
20     default         steadyState;
21 }
22
23 ddtSchemes
24 {
25     default         Euler;
26 }
27
28 gradSchemes
29 {
30     default         leastSquares;
31     grad(D)         leastSquares;
32     grad(T)         leastSquares;
33 }
```

```
34
35  divSchemes
36  {
37      default         none;
38      div(sigmaD)     Gauss linear;
39  }
40
41  laplacianSchemes
42  {
43      default         none;
44      laplacian(DD,D) Gauss linear corrected;
45      laplacian(DT,T) Gauss linear corrected;
46  }
47
48  interpolationSchemes
49  {
50      default         linear;
51  }
52
53  snGradSchemes
54  {
55      default         none;
56  }
57
58  fluxRequired
59  {
60      default         no;
61      D               yes;
62      T               no;
63  }
64
65
66  // ************************************************************************* //
```

system ディレクトリにある *fvSolution* ディクショナリでは，求解に使用される線形方程式のソルバおよびアルゴリズムを設定します．まず *solvers* サブディクショナリを見ると，D の solver が GAMG になっていることがわかります．tolerance で表されるソルバ許容値（ソルバ名の次の数値）は，本問題では 10^{-6} を設定します．relTol で表されるソルバの相対許容値（さらにその次の数値）には各反復ごとの残差の所要低減量を設定します．本問題においては多くの項が陽であり，また個別の反復的手順の一部としてアップデートされるため，各反復において厳しい相対許容値を設定することは非効率的です．したがって相対許容値として合理的な値は 0.01，もしくはさらに高めの 0.1，あるいはせいぜいこのケースのように 0.9 程度にしておきます．

```
17
18  solvers
19  {
20      "(D|T)"
21      {
22          solver          GAMG;
23          tolerance       1e-06;
24          relTol          0.9;
25          smoother        GaussSeidel;
26          cacheAgglomeration true;
27          nCellsInCoarsestLevel 20;
28          agglomerator    faceAreaPair;
29          mergeLevels     1;
30      }
31  }
32
33  stressAnalysis
34  {
35      compactNormalStress yes;
36      nCorrectors     1;
37      D               1e-06;
```

```
38 }
39
40
41 // ************************************************************************* //
```

fvSolution ディクショナリは，アプリケーションソルバに特有の制御パラメータを含む *stressAnalysis* サブディクショナリを含みます．まず，各時刻ステップ内での表面力境界条件処理を含めた，全方程式系に関する外側ループの数を指定する `nCorrectors` があります．本問題は定常状態を扱いますので，「時刻ステップ」を反復回数カウンタとして使い収束解へと向かう反復を実行することになります．したがって `nCorrectors` を 1 に設定します．

`D` キーワードには外側反復ループにおける収束許容値，すなわち初期残差に対して反復計算によって消去されるべきレベルを設定します．本問題では前述において設定したソルバ許容値の 10^{-6} に設定します．

2.2.2 コードの実行

以下に示すようなコマンドによって，実行後にログファイルに記録された収束状況を見ることができるよう，バックグラウンドでコードを実行します．

```
cd $FOAM_RUN/tutorials/stressAnalysis/solidDisplacementFoam/plateHole
solidDisplacementFoam > log &
```

実行後には生成されたログファイルを見て，反復回数および解を求める各方向変位の初期・最終残差などの収束状況を確認できます．本ケースの反復許容回数設定では，最終残差は必ず初期残差の 0.1 倍以下となるはずです．いったん両初期残差ともに 10^{-6} の収束許容残差以下となれば，その計算は収束したとみなしバッチジョブを `kill` することによって止めることができます．

2.2.3 後処理

後処理は 2.1.4 項と同様に行うことができます．solidDisplacementFoam ソルバは，応力場 σ を対称テンソル場として出力します．したがって例えば，σ_{xx} を paraFoam で見ることができます．OpenFOAM ソルバにおける変数名は通常，それらを表す数学記号にならって名付けられることは，ここで再度述べるに値するでしょう．ギリシア記号の場合は，変数は発音どおりに名付けられます．例えば，σ_{xx} は `sigmaxx` と名付けられます．

独立したスカラ成分の σ_{xx} や σ_{xy} などは，2.1.5.7 で述べた foamCalc を `sigma` に関して実行して求めます．

```
foamCalc components sigma
```

`sigmaxx` や `sigmaxy` などと名づけられた成分のファイルが時間のディレクトリに生成されます．図 2.18 のように応力を paraFoam で見ることができます．

式 (2.14) の解析解とここで得られた数値解を比較しましょう．そのためには，解析領域左端の対称面に沿ってのデータを出力しなければなりません．このようなグラフのために必要な

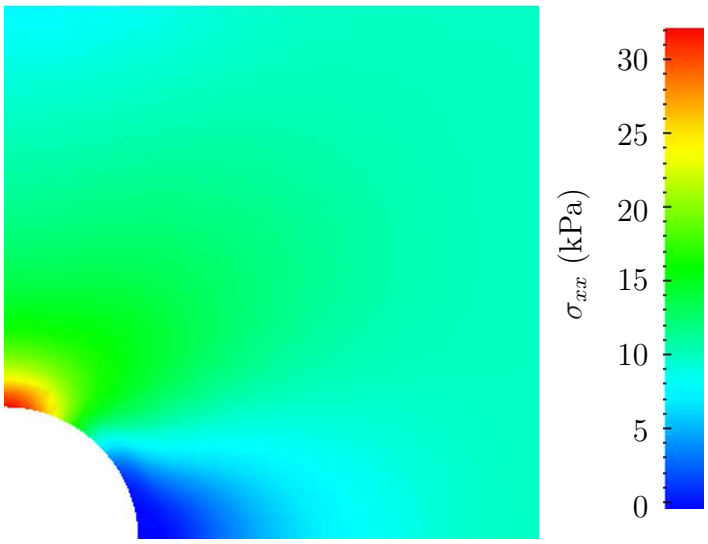


図 2.18　穴あき板における応力場

データは，sample ユーティリティによって作成することができます．sample の設定は *system* ディレクトリ内の *sampleDict* で行い詳細は表 6.3 に要約しています．データのサンプリングを行う座標区間は，`sets` によって $(0.0, 0.5, 0.25)$ から $(0.0, 2.0, 0.25)$ に指定されています．物理量は `fields` に指定します．

```
17
18  interpolationScheme cellPoint;
19
20  setFormat       raw;
21
22  sets
23  (
24      leftPatch
25      {
26          type    uniform;
27          axis    y;
28          start   ( 0 0.5 0.25 );
29          end     ( 0 2 0.25 );
30          nPoints 100;
31      }
32  );
33
34  fields          ( sigmaxx );
35
36
37  // ************************************************************************* //
```

通常通り sample を実行してください．`writeFormat` は `raw` 形式で 2 列のフォーマットとなっています．データは *postProcessing/sets* ディレクトリの時刻サブディレクトリの中のファイルに書き込まれます．たとえば，時刻 $t = 100\,\mathrm{s}$ のデータは *postProcessing/sets/100/leftPatch_sigmaxx.xy* に書き込まれます．GnuPlot のようなアプリケーションでは，コマンドプロンプトで以下を入力することで，数値解および解析解の両方をプロットすることができます．

```
plot [0.5:2] [0:] 'postProcessing/sets/100/leftPatch_sigmaxx.xy',
     1e4*(1+(0.125/(x**2))+(0.09375/(x**4)))
```

プロット例を図 2.19 に示します．

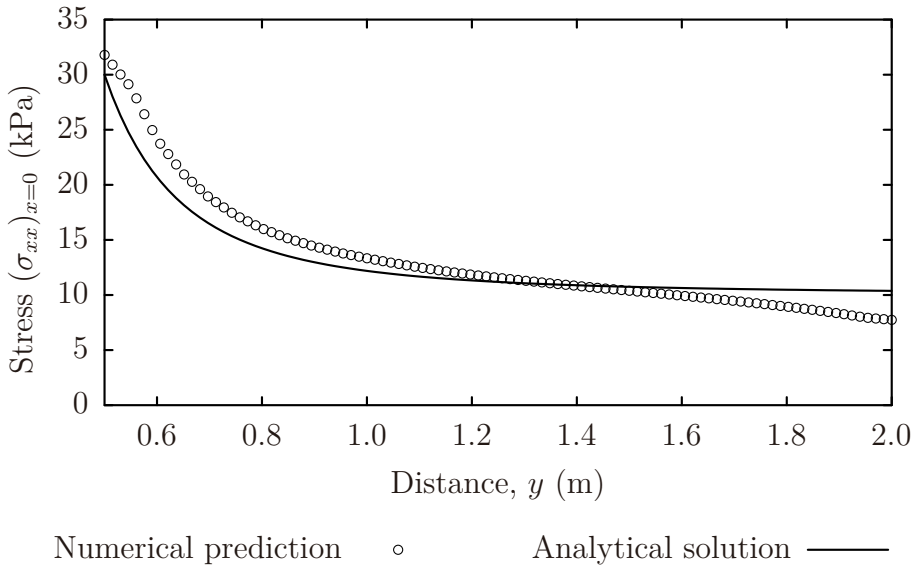


図 2.19 垂直方向対称面における法線方向応力

2.2.4 演習

以下は solidDisplacementFoam に習熟していただくための演習課題です．

2.2.4.1 メッシュ解像度の増加

x，y 方向それぞれのメッシュ解像度を増やしてみましょう．2.2.3 項の最終的な粗メッシュの結果を，mapFields を使って密メッシュの初期条件にマッピングしてください．

2.2.4.2 非等間隔メッシュの導入

穴に近いセルが遠いセルより密になるよう，メッシュ幅を変化させてください．隣接するセルの大きさの比率が 1.1 以上にならないように，またブロック間のセルの大きさの比率がブロック内の比率と同様となるようメッシュを作成してください．メッシュの非等間隔化については 2.1.6 項で述べました．ここでも，2.2.3 項の最終的な粗メッシュでの結果を，mapFields を使って非等間隔メッシュの初期条件としてマッピングします．結果を解析解および非等間隔化する前の結果と比較してみましょう．等間隔メッシュと同一のセル数を使用した場合，解の精度は改善されるでしょうか？

2.2.4.3 板の大きさの変更

ここで示されている解析解は，有限な大きさの穴を有する無限大板におけるものです．したがって有限な大きさの板においては，この解析解は必ずしも正確ではありません．誤差を見積もるために，穴の大きさを同一に保ったまま板を大きくしてみましょう．

2.3 ダムの決壊

このチュートリアルでは，interFoam を用いて，単純化したダム決壊の 2 次元問題を解くことにします．この問題の特徴は，くっきりとした界面や自由表面によって隔てられている二つの流体による非定常の流れ場であることです．interFoam における 2 相流体を解くアルゴリズムは，Volume of fluid (VOF) 法によるものであり，ここでは特別な輸送方程式を解いて，計

算格子における 2 相の体積分率，もしくは相比率 α を決定します．各物理量は，この相比率に（各流体の）密度をかけた平均的な値として算出されます．個々の物質の界面は，VOF 法ではその性質上明示的には解かれず，相比率場の特性として浮き上がってくるということになります．相比率は 0 から 1 の間の任意の値をとり得るため，界面は決してくっきりと定義されませんので，本来のくっきりとした界面が存在するべき領域の周辺を，（計算上の）界面がぼんやりと占めることになります．

計算条件では，貯水池の左側に，膜で仕切られた水柱が最初存在します．時刻 $t = 0\,\mathrm{s}$ に，膜が取り除かれて，水柱が崩れだします．崩壊しながら，水流は貯水槽の底にある出っ張りにぶつかり，いくつかの気泡を含む，複雑な流れ場の様相を呈します．計算形状と初期条件は図 2.20 に示しました．

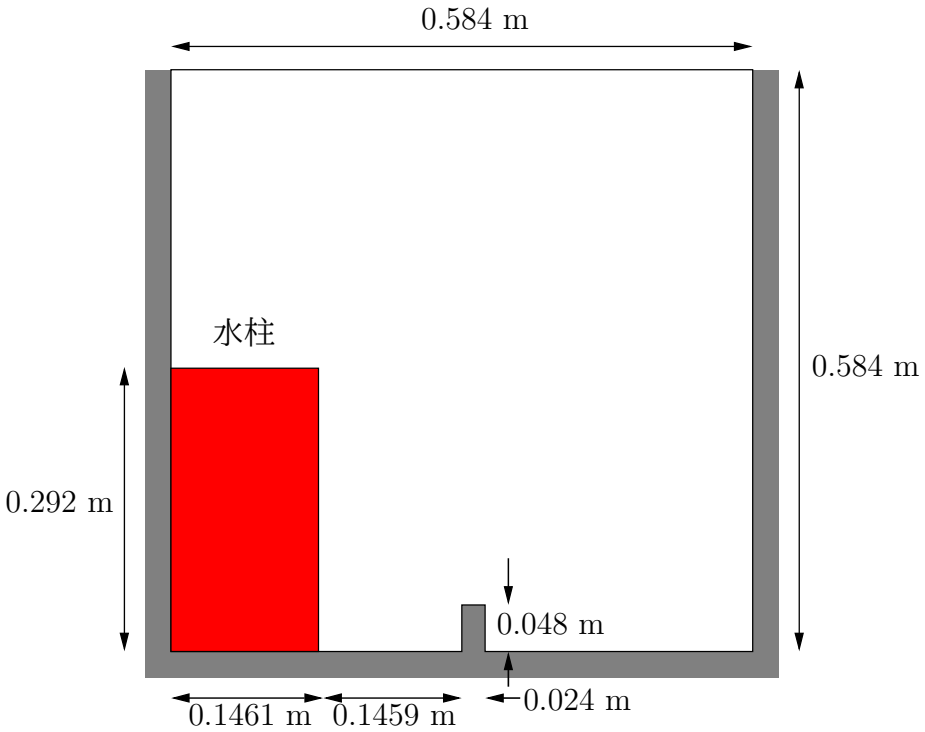


図 2.20　ダム決壊の計算形状

2.3.1　格子の生成

$FOAM_RUN/tutorials/multiphase/interFoam/laminar にある *damBreak* のケースディレクトリに移動しましょう．前述した方法で blockMesh を実行して格子を生成してください．この damBreak のメッシュは五つのブロックで構成されます．*blockMeshDict* の中身を以下に示します．

```
17
18  convertToMeters 0.146;
19
20  vertices
21  (
22      (0 0 0)
23      (2 0 0)
24      (2.16438 0 0)
25      (4 0 0)
```

```
26      (0 0.32876 0)
27      (2 0.32876 0)
28      (2.16438 0.32876 0)
29      (4 0.32876 0)
30      (0 4 0)
31      (2 4 0)
32      (2.16438 4 0)
33      (4 4 0)
34      (0 0 0.1)
35      (2 0 0.1)
36      (2.16438 0 0.1)
37      (4 0 0.1)
38      (0 0.32876 0.1)
39      (2 0.32876 0.1)
40      (2.16438 0.32876 0.1)
41      (4 0.32876 0.1)
42      (0 4 0.1)
43      (2 4 0.1)
44      (2.16438 4 0.1)
45      (4 4 0.1)
46  );
47
48  blocks
49  (
50      hex (0 1 5 4 12 13 17 16) (23 8 1) simpleGrading (1 1 1)
51      hex (2 3 7 6 14 15 19 18) (19 8 1) simpleGrading (1 1 1)
52      hex (4 5 9 8 16 17 21 20) (23 42 1) simpleGrading (1 1 1)
53      hex (5 6 10 9 17 18 22 21) (4 42 1) simpleGrading (1 1 1)
54      hex (6 7 11 10 18 19 23 22) (19 42 1) simpleGrading (1 1 1)
55  );
56
57  edges
58  (
59  );
60
61  boundary
62  (
63      leftWall
64      {
65          type wall;
66          faces
67          (
68              (0 12 16 4)
69              (4 16 20 8)
70          );
71      }
72      rightWall
73      {
74          type wall;
75          faces
76          (
77              (7 19 15 3)
78              (11 23 19 7)
79          );
80      }
81      lowerWall
82      {
83          type wall;
84          faces
85          (
86              (0 1 13 12)
87              (1 5 17 13)
88              (5 6 18 17)
89              (2 14 18 6)
90              (2 3 15 14)
91          );
92      }
93      atmosphere
94      {
95          type patch;
96          faces
```

```
97          (
98              (8 20 21 9)
99              (9 21 22 10)
100             (10 22 23 11)
101         );
102     }
103 );
104
105 mergePatchPairs
106 (
107 );
108
109 // ************************************************************************* //
```

2.3.2　境界条件

constant/polyMesh ディレクトリの *boundary* ファイルを見ることで blockMesh で生成された境界の形状を確認しましょう．`leftWall`，`rightWall`，`lowerWall`，`atmosphere`，`defaultFaces` の五つの境界パッチがあります．パッチの種類について理解しておきましょう．`atmosphere` は何の属性もなく，単に境界条件によって規定される標準の patch です．`defaultFaces` は，本ケースでは 2 次元であるためパッチの法線方向を解析の対象としないため，empty とします．`leftWall`，`rightWall`，`lowerWall` はそれぞれ wall です．ただの patch と同様に wall もメッシュについて形状や位相の情報をもちませんが，壁として識別することができるので，アプリケーションに特殊な壁表面のモデリングを明示するために wall と定義しています．

interFoam のソルバが，界面と壁面との接点における表面張力に対するモデルを含んでいる，というのがよい例です．このモデルは `alpha` (α) 場の alphaContactAngle の境界条件と関連付けられています．その場合，静的な接触角度 `theta0` θ_0 や，前縁や後縁における動的な接触角度である `thetaA` θ_A と `thetaR` θ_R，そして，動的な接触角度において速度に比例する係数 `uTheta` を指定する必要があります．

このチュートリアルでは，壁面と界面間の表面張力による効果を無視することにしたいと思います．それは，静的な接触角度を $\theta_0 = 90°$ に，速度比例係数を 0 と設定することで可能です．しかしながら，壁の境界条件として，通常の `wall` タイプの境界条件を指定する別なやり方もあります．この場合，`alpha` に対して `alphaContactAngle` の境界条件を設定する代わりに，`zeroGradient` に設定します．

`top` の境界は大気に対して開放されていることから，内部流れに応じて流出・流入のいずれも可能にしておく必要があります．したがって，以下のような，安定性を維持しながらこれを可能にするような圧力と速度に対する境界条件の組み合わせを用いることになります．

- totalPressure は，与えられた全圧 `p0` と局所速度 `U` から計算される fixedValue 条件です．
- pressureInletOutletVelocity は全成分に zeroGradient を適用しますが，流入がある場合は例外として fixedValue が接線成分に適用されます
- inletOutlet は，流れが外向きならば zeroGradient であり，流れが内向きならば fixedValue です

すべての壁の境界においては，圧力場には fixedFluxPressure 境界条件を適用しますが，これは境界のフラックスが速度の境界条件にマッチするように圧力勾配を調整するものです．

2 次元問題における前後の面を表している `defaultFaces` パッチは，通常通り `empty` タイプ

にします．

2.3.3 初期条件の設定

これまでのケースと異なり，ここでは相比率 α_{water} に対して，以下のような非一様な初期条件を与えます．

$$\alpha_{\text{water}} = \begin{cases} 1 & \text{水の相について} \\ 0 & \text{空気の相について} \end{cases} \tag{2.15}$$

これは，setFields ユーティリティを実行することによって行います．この実行には *system* ディレクトリ内の *setFieldsDict* を必要とします．このケースにおける *setFieldsDict* ファイルの内容を以下に示します．

```
17
18  defaultFieldValues
19  (
20      volScalarFieldValue alpha.water 0
21  );
22
23  regions
24  (
25      boxToCell
26      {
27          box (0 0 -1) (0.1461 0.292 1);
28          fieldValues
29          (
30              volScalarFieldValue alpha.water 1
31          );
32      }
33  );
34
35
36  // ************************************************************************* //
```

ここで，`defaultFieldValues` は場の規定値を設定するものであり，*regions* のサブディクショナリにおいて別途指定されない場合に場に与えられる値です．*regions* のサブディクショナリは，指定された領域において，規定値を上書きする `fieldValues` を含んだサブディクショナリのリストを含んでいます．領域の定義は，ある位相幾何学的な制約に基づいて，点や格子，界面等の集合を生成する `topoSetSource` によって行います．ここでは，`boxToCell` を使って，大きいほうと小さいほうの二つの座標点で定義されるバウンディング・ボックスを生成し，水の領域となるセルの集合を定義しています．また，この領域における相比率 α_{water} を 1 と指定しています．

setFields ユーティリティはファイルから場を読み込み，再計算したうえで，再びファイルに書き込みます．そのファイルは上書きされてしまうので，setFields を実行する前にバックアップをとることをお勧めします．この damBreak チュートリアルには，`alpha.water` 場は最初は `alpha.water.org` という名前のバックアップしか置いてありません．setFields を実行する前に，まず以下のようにタイプして `alpha.water.org` を `alpha.water` にコピーする必要があります．

```
cp 0/alpha.water.org 0/alpha.water
```

water の物性			
動粘性率	m^2s^{-1}	nu	1.0×10^6
密度	$kg\,m^{-3}$	rho	1.0×10^3
air の物性			
動粘性率	m^2s^{-1}	nu	1.48×10^{-5}
密度	$kg\,m^{-3}$	rho	1.0
両相の物性			
表面張力	$N\,m^{1}$	sigma	0.07

表 2.3　damBreak チュートリアルにおける流体物性

さて，それでは setFields を他のプログラムと同様に起動してください．そうしたら，paraFoam を用いて，初期の alpha.water 場が図 2.21 のように望むような分布になっているかどうか確かめてください．

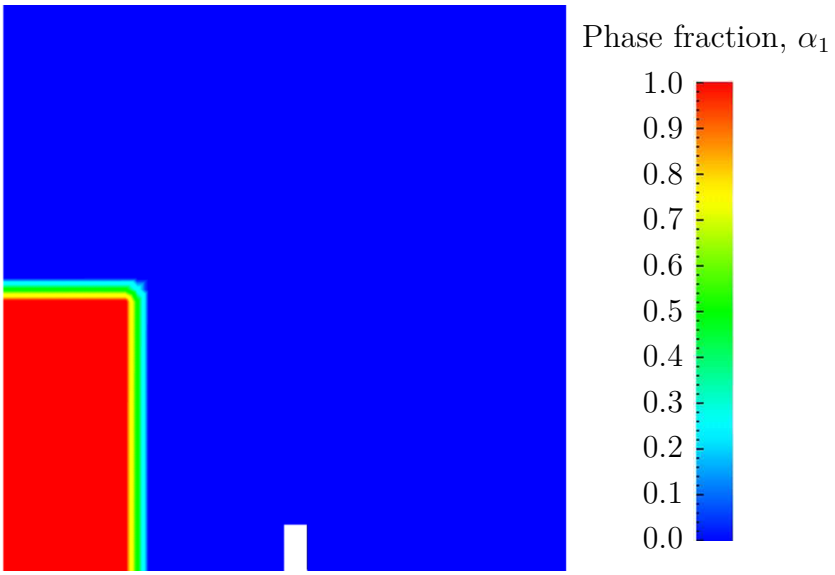


図 2.21　相比率 alpha.water の初期条件

2.3.4　流体の物性値

constant ディレクトリの *transportProperties* ファイルを確認しましょう．このディクショナリは各流体の物性値を含んでおり，二つのディクショナリ *water* と *air* に分かれています．各相の輸送モデルは，transportModel によって選択されます．ここで，動粘性係数が nu というキーワードで指定され，一定値である Newtonian モデルを選んでください．

CrossPowerLaw といったその他のモデルにおける粘性係数の指定は，この例における *CrossPowerLawCoeffs* といったように，*<model>Coeffs* という名のサブディクショナリの中で行います．

密度の指定は，rho キーワードで行います．

二つの相の間の表面張力は，sigma キーワードで指定します．

このチュートリアルで用いた値を表 2.3 に挙げます．

重力加速度は全領域にわたって一様で，*constant* ディレクトリの *g* ファイルで指定されます．*U* や *p* のような通常のフィールドのファイルと異なり，*g* は uniformDimensionedVectorField であ

り，単に dimensions と value の組だけを含みます．このチュートリアルでは $(0, 9.81, 0)\,\mathrm{m\,s^{-2}}$ です．

```
17
18 dimensions      [0 1 -2 0 0 0 0];
19 value           ( 0 -9.81 0 );
20
21
22 // ************************************************************************* //
```

2.3.5 乱流モデル

キャビティの例題のように，*turbulenceProperties* ディクショナリの simulationType キーワードで乱流のモデリング手法を選択することができます．この例題は乱流モデルを使わずに実行したいので laminar と指定します．

```
17
18 simulationType  laminar;
19
20
21 // ************************************************************************* //
```

2.3.6 時間ステップの制御

自由界面の捕捉においては，時間ステップの制御は重要です．というのも，界面捕捉のアルゴリズムは，通常の流体計算に比べ，Courant 数 Co に対してかなり鋭敏だからです．理想的には，界面がある領域において，上限値として $Co \approx 0.5$ を超えないようにすべきです．伝播速度の予測が容易であるようなケースでは，Co の制限を守るような固定した時間ステップを指定することができますが，より複雑なケースの場合時間ステップの指定はずっと困難になります．そこで，interFoam では，*controlDict* において，時間ステップの自動修正を指定することをお勧めします．adjustTimeStep を on にして，Co の最大値 (相の場については maxAlphaCo，その他の場については maxCo) を 1.0 にしましょう．時間ステップの上限 maxDeltaT はこのシミュレーションでは超えようのない値，たとえば 1.0 等に設定すればよいでしょう．

ただし，自動時間ステップ制御を用いると，その時間ステップは必ずしも使いやすい値に丸められるとは限りません．したがって，固定の時間ステップ間隔で OpenFOAM に結果を出力させた場合，その時刻はきりの良い値になりません．ところがこの自動時間ステップ制御を用いていても，OpenFOAM では決まった時刻に結果を出力するように指定することが可能です．この場合，OpenFOAM は，結果の出力に指定された時刻ぴったりに合うように時間刻みを補正しつつ，自動時間刻みの制御を行います．これを行うには，*controlDict* ディクショナリにおける writeControl に対して，adjustableRunTime オプションを選んでください．*controlDict* ディクショナリの中身は以下のようになります．

```
17
18 application     interFoam;
19
20 startFrom       startTime;
21
22 startTime       0;
```

```
23
24  stopAt          endTime;
25
26  endTime         1;
27
28  deltaT          0.001;
29
30  writeControl    adjustableRunTime;
31
32  writeInterval   0.05;
33
34  purgeWrite      0;
35
36  writeFormat     ascii;
37
38  writePrecision  6;
39
40  writeCompression uncompressed;
41
42  timeFormat      general;
43
44  timePrecision   6;
45
46  runTimeModifiable yes;
47
48  adjustTimeStep  yes;
49
50  maxCo           1.0;
51  maxAlphaCo      1.0;
52
53  maxDeltaT       1;
54
55
56  // ************************************************************************* //
```

2.3.7　離散化スキーム

この interFoam ソルバは，OpenCFD によって開発された Multidimensional Universal Limiter for Explicit Solution (MULES) 法を用いており，基礎を成す数値的スキームやメッシュ構造から独立な段階分数の有界性を保存するために使います．したがって，対流項に対するスキームの選択は，風上差分のように，安定性や有界性の強いものに限定されません．

対流項のスキームは，*fvSchemes* ディクショナリの *divSchemes* サブディクショナリで設定します．この例題では，運動量方程式における対流項 $\nabla \cdot (\rho \boldsymbol{U}\boldsymbol{U})$ に対しては，`div(rho*phi,U)` キーワードにて，`Gauss linearUpwind grad(U)` を使えば良い精度が得られます．リミッタ付きの線形なスキームでは，4.4.1 項に記述されるように係数 ϕ を必要とします．ここでは最も安定性が高くなるように $\phi = 1.0$ とします*．`div(phi,alpha)` キーワードで表される $\nabla \cdot (\boldsymbol{U}\alpha)$ 項には `vanLeer` を使用します．`div(phirb,alpha)` キーワードで表される $\nabla \cdot (\boldsymbol{U}_{\mathrm{rb}}\alpha)$ 項には 2 次精度の `linear`（中心）差分を使うと，MULES アルゴリズムによって有界性が保証されます．

その他の離散化項は一般に決ったスキームを使用します．以上から *fvSchemes* ディクショナリのエントリは以下のようになるでしょう．

```
17
18  ddtSchemes
19  {
20      default         Euler;
```

*訳注：旧バージョンで `Gauss limitedLinearV 1.0` が使われていたときの説明が残っているが，バージョン 2.3.0 から `Gauss linearUpwind grad(U)` に変更されたので $\phi = 1.0$ は使われない．

```
21  }
22
23  gradSchemes
24  {
25      default         Gauss linear;
26  }
27
28  divSchemes
29  {
30      div(rhoPhi,U)  Gauss linearUpwind grad(U);
31      div(phi,alpha)  Gauss vanLeer;
32      div(phirb,alpha) Gauss linear;
33      div((muEff*dev(T(grad(U))))) Gauss linear;
34  }
35
36  laplacianSchemes
37  {
38      default         Gauss linear corrected;
39  }
40
41  interpolationSchemes
42  {
43      default         linear;
44  }
45
46  snGradSchemes
47  {
48      default         corrected;
49  }
50
51  fluxRequired
52  {
53      default         no;
54      p_rgh;
55      pcorr;
56      alpha.water;
57  }
58
59
60  // ************************************************************************* //
```

2.3.8 線形ソルバの制御

fvSolution では，*PISO* サブディクショナリが interFoam に特化した要素を含んでいます．ここには，通常と同じく運動量方程式に対する反復数だけでなく，α 相方程式の PISO ループに対する反復数も指定します．特に重要なものは `nAlphaSubCycles` と `cAlpha` キーワードです．`nAlphaSubCycles` は α 方程式内の内側反復の数を表してあり，ここで，内側反復は与えられた時間ステップ内での方程式に対する付加的な解の点数です．それは，時間ステップや計算時間の莫大な増加なしで解を安定させることができるようにするものです．ここでは，二つの sub-cycle を指定しており，α 方程式は実際の各時間ステップ内で 2 分の 1 の幅の時間スッテプで 2 回解かれていることを意味します．

`cAlpha` キーワードは界面の圧縮を制御する要素です．つまり，0 は無圧縮に対応し，1 は保存的な圧縮に対応し，1 以上は拡張された界面の圧縮を意味します．通常はこの例題で用いられている 1.0 の値が推奨されます．

2.3.9　コードの実行

コードの実行方法については，前述のチュートリアルに詳細に記述しています．以下のようにして，標準出力とファイルの両方への書き込みを可能にする tee コマンドを試してみてください．

```
cd $FOAM_RUN/tutorials/multiphase/interFoam/laminar/damBreak
interFoam | tee log
```

このコードは，対話的に実行されつつ，出力のコピーを *log* ファイルに記録してくれます．

2.3.10　後処理

結果の後処理は，通常の方法で行えます．ユーザは参照時間の経過に伴う相比率 alpha.water の発達を見ることができます．例えば図 2.22 をみてください．

2.3.11　並列計算

前述の例の結果はかなり目の粗い格子を使って得られました．ここでは格子の解像度を増やして再計算します．新しいケースは，一般的に一つのプロセッサでは計算するのに数時間を要するので，複数のプロセッサにアクセスしているのであれば，OpenFOAM の並列計算機能を試してみてもよいでしょう．

まず初めに，damBreak ケースのコピーをしてください．

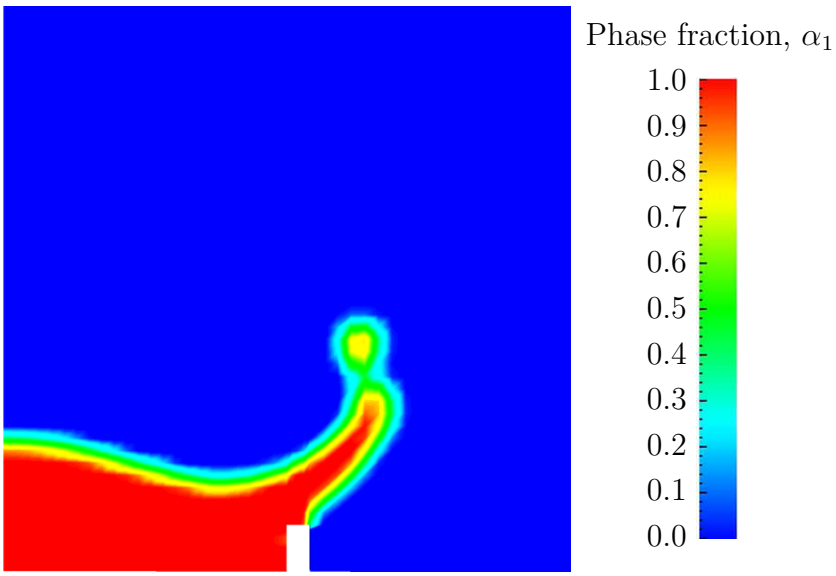
```
cd $FOAM_RUN/tutorials/interFoam
mkdir damBreakFine
cp -r damBreak/0 damBreakFine
cp -r damBreak/system damBreakFine
cp -r damBreak/constant damBreakFine
```

新しいケースは damBreakFine と名づけてください．新しいケースディレクトリを開いて *blockMeshDict* ファイル内の blocks の記述を以下のように変更してください．

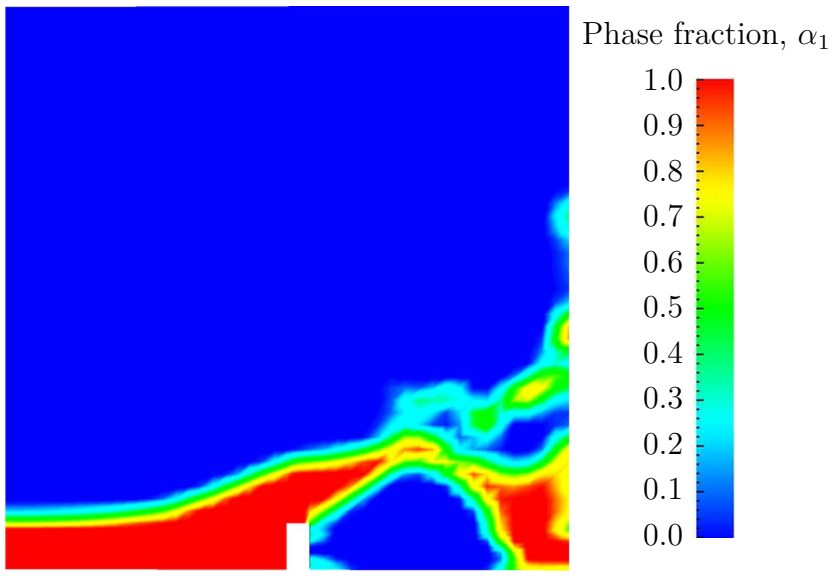
```
blocks
(
    hex (0 1 5 4 12 13 17 16) (46 10 1) simpleGrading (1 1 1)
    hex (2 3 7 6 14 15 19 18) (40 10 1) simpleGrading (1 1 1)
    hex (4 5 9 8 16 17 21 20) (46 76 1) simpleGrading (1 2 1)
    hex (5 6 10 9 17 18 22 21) (4 76 1) simpleGrading (1 2 1)
    hex (6 7 11 10 18 19 23 22) (40 76 1) simpleGrading (1 2 1)
);
```

上記で，入力は *blockMeshDict* ファイルで表示されているように，つまりは，格子の密度を変更しなければなりません．例えば 46 10 1 という入力や 1 2 1 という格子幅の勾配の入力のようにです．正しく入力できたら，メッシュを生成します．

ここで格子が damBreak の例から変更されると，時刻 *0* のディレクトリ内の *alpha.water* という相の場を再初期化しなければなりません．というのも alpha.water は新しい格子とは合致しないいくつかの要素を含んでいるからです．ここで，U や p_rgh という場は変更する必要がな



(a) $t = 0.25\,\mathrm{s}$ のとき



(b) $t = 0.50\,\mathrm{s}$ のとき

図 2.22 α 相のスナップショット

いことに注意しましょう．それらは `uniform` として明記されておりフィールド内の要素の数と独立だからです．フィールドの初期化はシャープな界面をもつように行いたいものです．つまり，その要素が $\alpha = 1$ または $\alpha = 0$ となるようにです．mapFields によりフィールドを更新すると，界面に補間された $0 < \alpha < 1$ となる値が生成される可能性があるので，setFields ユーティリティを再実行したほうがよいでしょう．その前に初期条件の一様な α のバックアップファイル *0/alpha.water.org* を *0/alpha.water* にコピーします．

```
cd $FOAM_RUN/tutorials/interFoam/laminar/damBreakFine
cp -r 0/alpha.water.org 0/alpha.water
setFields
```

OpenFOAM で用いられる並列計算の手法はいわゆる領域分割であり，幾何形状やそれに関連する場が領域ごとに分解されて，解析のため個々のプロセッサに割り当てられます．そのた

め，並列計算を実行するために必要な最初の段階は，decomposePar を用いて領域を分解することです．decomposePar の設定は *system* ディレクトリにある，*decomposeParDict* というファイルです．他のユーティリティ同様，初期状態のファイルがユーティリティのソースコードのディレクトリ (*$FOAM_UTILITIES/parallelProcessing/decomposePar*) にあります．

最初の入力の `numberOfSubdomains` において何個のサブ領域に分割するかを指定します．通常はこのケースに利用できるプロセッサの数と対応します．

このチュートリアルでは，分解の手法は `simple` で，対応する `simpleCoeffs` は以下の基準のように編集しましょう．領域は，x，y，z 方向で部分かサブ領域に分けられ，各方向へのサブ領域の数はベクトル $\boldsymbol{n}$ として与えられます．この幾何形状は 2 次元なので，3 番目の方向 z に分割されることはなく，それゆえ必ず n_z は 1 になります．n_x と n_y は x，y 方向の領域の分割数 $\boldsymbol{n}$ を構成し，n_x と n_y の積で表されるサブ領域の数が `numberOfSubdomain` に指定したものと等しくなる必要があります．隣接するサブ領域間のセル面の数を最小にしたほうがよいので，正方形の幾何形状では，x，y 方向を均等に分割するのが良いでしょう．`delta` キーワードは `0.001` に設定しましょう．

例として，四つのプロセッサで計算を実行するとします．`numberOfSubdomain` を 4 に，$\boldsymbol{n} = (2, 2, 1)$ に設定します．*decomposeParDict* を閉じて，decomposePar を実行します．decomposePar のスクリーンメッセージが確認でき，分解はプロセッサ間で均等に分配されたと表示されます．

3.4 節に並列計算の方法についての詳細があるので参照してください．このチュートリアルでは並列計算の一例を示しているにすぎません．openMPI を用いて標準のメッセージパッシングインターフェース (MPI) を実装しています．ここでは，テストとしてローカルホストのみの単独ノードで実行します．

```
mpirun -np 4 interFoam -parallel > log &
```

3.4.2 項に後述しますが，ケースが実行されるマシンのホストネームを列記したファイルを作っておけばネットワーク上のより多くのノードを使って計算することも可能です．ケースはバックグラウンドで実行し，進行状況を *log* ファイルで監視するのがよいでしょう．

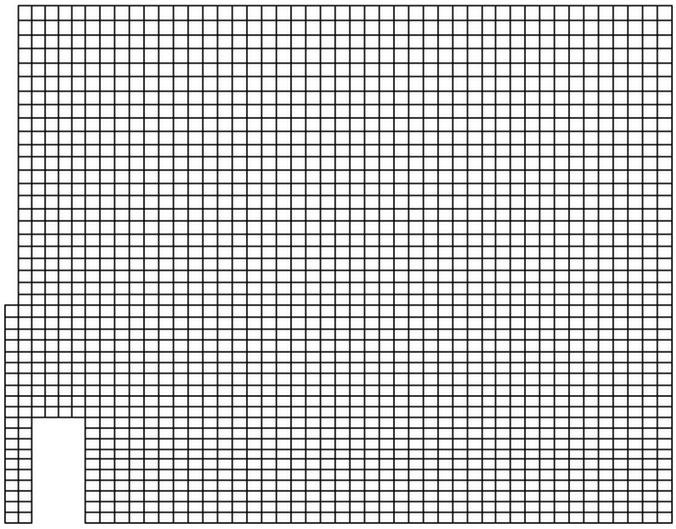
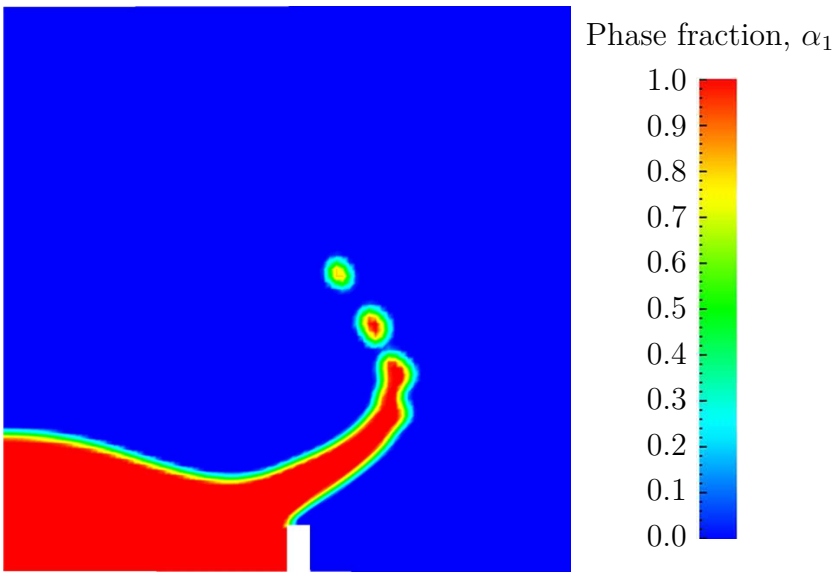
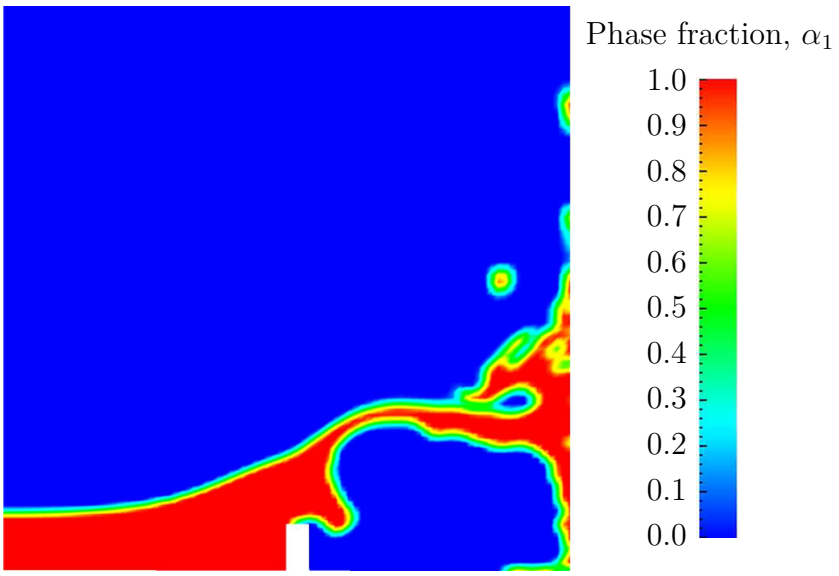


図 2.23　並列プロセスケースでのプロセッサ 2 のメッシュ



(a) $t = 0.25\,\mathrm{s}$ のとき



(b) $t = 0.50\,\mathrm{s}$ のとき

図 2.24 正確なメッシュの α 相のスナップショット

2.3.12 並列計算ケースの後処理

一度ケースの実行が完了したら，分解されたフィールドとメッシュは reconstructPar を実行して後処理のために再統合します．コマンドラインから容易に実行できます．細かい格子による結果は図 2.24 に表されます．インタフェースでの結果は粗い格子のものと比較して著しく改良されたことがみてとれます．

また，単に個々のプロセッサの領域を一つのケースと扱うことで，分解された領域の部分を個々に後処理することもできます．

例えば，paraFoam を以下のように起動します．

```
paraFoam -case processor1
```

すると processor1 が ParaView のケースモジュールとして表れます．図 2.23 に，simple 方式

による領域分割を行った際の `processor1` のメッシュを示します.

第3章
アプリケーションとライブラリ

繰り返しいいますが，OpenFOAM は実行のために，基本的に C++ のライブラリを用いています．OpenFOAM はプリコンパイル済みの数多くのアプリケーションで構成されていますがユーザが独自に作成したり従来のものを修正しても構いません．アプリケーションは大きく二つのカテゴリに分けられています．

ソルバ　数値連続体力学の特定の問題を解くためのもの
ユーティリティ　主にデータ操作や代数計算を主に行う前後処理を実行するもの

OpenFOAM は一連のプリコンパイル済ライブラリに分けられ，それらはソルバとユーティリティの集合体をダイナミックにリンクします．ユーザが適宜独自のモデルをライブラリに追加できるように物理的モデルのこのようなライブラリはソースコードとして与えられます．

この章ではソルバ，ユーティリティ，ライブラリの概説とこれらの作成，修正，編集，実行方法について述べます．

3.1　OpenFOAM のプログラミング言語

OpenFOAM ライブラリが動作する方法を理解するためには OpenFOAM の基本言語である C++ の予備知識がいくらか必要となります．そのために必要な知識を本章に述べます．その前に重要なことは，オブジェクト指向プログラミングの背景にある様々なアイデアと，我々が OpenFOAM のメインプログラミング言語として C++ を選択したことを説明するための，言語に関する一般的な概念を知っておくことです．

3.1.1　言語とは

話し言葉と数学が普及したのは，特に抽象的な概念を表現する際の，効率性によるものでした．その例として，流量について，「速度場 (velocity field)」という言葉を私たちが使うとき，流れの性質の言及もせず，何ら具体的な速度データがなくとも使っています．その言葉の中には，運動の向きと大きさやその他の物理的性質との関係の概念が要約されています．これを数学にすると，「速度場」を $\boldsymbol{U}$ などの簡易な記号で表し，また速度場の大きさを表したいときには $|\boldsymbol{U}|$ として表記します．数学は口話よりも効率性に優れ，複雑な概念を極めて明快に表現できます．

連続体力学の中で解析しようとしている問題は，固有の構成要素やタイプとして表現されたものでもなく，コンピュータの認識する，いわゆるビット，バイト，数値などの概念とも異な

ります．問題はいつも，まず口話で提起され，空間と時間の三次元での偏微分方程式として表現されます．それらの方程式はスカラ，ベクトル，テンソルそしてそれらの場，テンソル代数，テンソル解析，次元の単位などの概念を含んでおり，これらの解は離散化手法やマトリクス，ソルバそして解法アルゴリズムを必要とします．

3.1.2　オブジェクト指向と C++

C++ のようなオブジェクト指向のプログラミング言語は，宣言の型としてクラスという考え方を採用しており，口語の部分や科学計算や技術計算に用いられる数学的な言語を取り扱っています．先に紹介した速度場はプログラミングコードでは記号 `U` で表され，速度場の大きさは `mag(U)` で表されます．速度はベクトル場であり，オブジェクト指向コードでは `vectorField` クラスとなります．速度場 `U` は，オブジェクト指向の項であることから，この場合 `vectorField` クラスのインスタンス，あるいはオブジェクトとということになります．

プログラミングの中で，物理的なオブジェクトと抽象的な構成要素を表現するオブジェクト指向のもっている明瞭さを過小評価してはいけません．クラス構造は，クラス自身など，開発したコードの領域を包含する集合であるから，容易にコードを管理することができます．新しいクラスには，他のクラスからのプロパティを継承させることができることから，`vectorField` には `vector` クラスと `Field` クラスを継承させることができます．C++ はテンプレートクラスのメカニズムを備えています．例えばテンプレートクラス `Field<Type>` は `scalar`，`vector`，`tensor` などどんな`<Type>` の `Field` も表現できます．テンプレートクラスの一般的な特性はテンプレートから作成されるどんなクラスにも通じます．テンプレート化や継承はコードの重複を減らし，コードの全体構造を決めるクラスのヒエラルキを作ります．

3.1.3　方程式の説明

OpenFOAM の設計の中心的なテーマは，OpenFOAM のクラスを用いて書かれたソルバのアプリケーションであり，偏微分方程式の解法と非常に似た構造をもっています．例えば方程式

$$\frac{\partial \rho \boldsymbol{U}}{\partial t} + \nabla \cdot \phi \boldsymbol{U} - \nabla \cdot \mu \nabla \boldsymbol{U} = -\nabla p$$

はコード

```
solve
(
    fvm::ddt(rho, U)
  + fvm::div(phi, U)
  - fvm::laplacian(mu, U)
    ==
  - fvc::grad(p)
);
```

で表されます．

これらの必要条件として，OpenFOAM の主たるプログラミング言語が継承やテンプレートクラス，仮想関数，演算子の多重定義といったオブジェクト指向的特徴をもっていることが必要です．これらの特性は，Fortran 90 のようにオブジェクト指向と称しつつ実際には非常に限られたオブジェクト指向の能力しかもっていない多くの言語では十分に利用できません．しかし，C++ はこれらの特性をすべてもつうえに，効率性の良い実行ファイルを作り出す信頼性の

あるコンパイラを使えるように標準的な仕様が定められたうえで広く使われているというさらなる長所をもっています．ゆえに OpenFOAM の主要言語なのです．

3.1.4 ソルバコード

ソルバコードは，解法アルゴリズムと方程式の手続き上の説明のようなものなので当然のようにほとんど手続きです．オブジェクト指向やソルバを書くための C++ プログラミングへの深い知識は必要ありませんが，オブジェクト指向やクラスの原理やいくらか C++ コードの構文の基礎知識は知っておくべきでしょう．基礎的な方程式やモデルや解法の理解やアルゴリズムは非常に重要です．

ユーザはたいていの場合 OpenFOAM クラスのどんなコードでも深く考える必要はありません．オブジェクト指向の真髄はユーザが何もしなくてもよいところにあります．単にクラスの在り方と機能の知識だけでクラスを使うのに十分です．それぞれのクラスやその機能などの説明は，OpenFOAM の配布物の中に Doxygen で生成された HTML のドキュメントとして供給されており，*$WM_PROJECT_DIR/doc/Doxygen/html/index.html* にあります．

3.2 アプリケーションやライブラリのコンパイル

コンパイルはアプリケーションの開発には必要不可欠の部分であり，コードのあらゆる部分には，OpenFOAM ライブラリ中の依存コンポーネントへのアクセス方法を指定する必要があるため，細心の管理が必要となります．

多くの場合，これらの構築は UNIX/Linux システムでは標準の UNIX make ユーティリティを使ってコンパイルします．しかしながら，OpenFOAM はより用途が広く簡便性に優れている，wmake でのコンパイルスクリプトを提供しています．実際，wmake は OpenFOAM のライブラリだけでなく，どのコードにも使われています．コンパイルのプロセスを理解するために，最初に C++ のある側面とそのファイル構成について図 3.1 で説明します．クラスとは，オブジェクトの構築様式，データの格納およびクラスのメンバ関数のような命令文のセットで定義されるものです．クラスの定義を含むファイルは.*C* の拡張子をもっており，例えばクラス nc であればファイル *nc.C* と書かれます．このファイルは，他のコードとは独立にコンパイルして *nc.so* のような拡張子.*so* をもつオブジェクトライブラリとして知られるバイナリ実行ライブラリファイルとすることができます．コードの一部を，仮に *newApp.C* などとしてコンパイルするとき，ユーザーは nc クラスを使うことにより，*nc.C* を再コンパイルしなくてもランタイムとして *newApp.C* で *nc.so* を呼び出せばよいことになります．これがダイナミックリンクといわれるものです．

3.2.1 ヘッダ.*H* ファイル

エラーチェックをおこなうにあたって，コンパイルするコードの部分がどのクラスで用いられるか，また実際の操作でどのように振舞うかを認識しなければなりません．それゆえ，(例えば *nc.H* のような) .*H* ファイル拡張子をもつヘッダファイルによってクラス宣言が必要です．このようなヘッダファイルにはクラス名とその機能が記述されています．

このファイルは，クラスを用いるあらゆるコード（クラス宣言のためのコードも含め）の最初

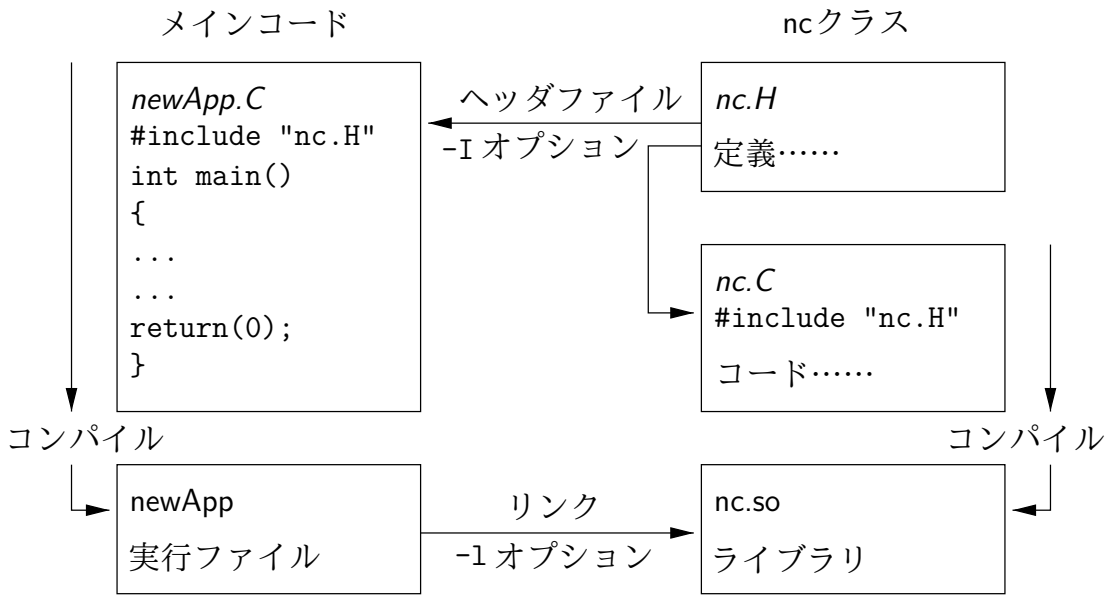


図 3.1　ヘッダファイル，ソースファイル，コンパイル，リンク

の部分に置きます．*.C* コードではどの部分でいくつのクラスを用いてもかまいませんが，かならずクラス宣言のために *.H* ファイルではじめる必要があります．クラスは他のクラスのリソースとして使うことができますが，その場合も関連付けた *.H* ファイルではじめます．クラスヒエラルキを再帰的に検索することで，結局，上位 *.C* コードが依存しているクラス (これらの *.H* ファイルは dependency とよばれる) で，すべてのクラスに関するヘッダファイルのリストをコンパイルすることができます．依存リストがあればコンパイラはソースファイルが最終コンパイル以来アップデートされているかどうかチェックでき，選択的に必要な部分だけコンパイルできます．

ヘッダファイルは，例えば

```
# include "otherHeader.H";
```

のような `# include` 命令文を使ったコードに含まれていますが，このようなコードはコンパイラに特定のファイルを読ませるために現在のファイルの読み込みを一時中断させます．コードの中の，あらゆる独立した部品はヘッダファイルに収めて，メインコードの関連箇所でインクルードすることで，コード可読性を高めることができます．例えば多くの OpenFOAM アプリケーションでは，作成フィールドや読み込みフィールドの入力データのコードはコードの始めに *createFields.H* と名づけられたファイルに含まれます．この方法では，ヘッダファイルは単独でクラスの宣言として使われるだけではありません．以下のようなその他の機能とともに依存リストファイルを維持管理するタスクを実行するのが wmake なのです．

- ソースファイルと，それらが依存しているファイルの依存関係リストの自動作成と管理
- マルチ・プラットフォームコンパイル適切なディレクトリ構造を通じてハンドルされたマルチプラットフォームでのコンパイルとリンク
- マルチ・ランゲージコンパイルと C や C++ や Java 等のリンケージ
- C や C++，Java のようなマルチ言語でのコンパイルとリンク
- デバッグや最適化，並列処理，分析といったマルチオプションでのコンパイルとリンク

- lex，yacc，IDL，MOC といった，ソースコードの作成プログラムのサポート
- ソースファイルリストの簡潔なシンタックス
- 新規のコードリストのソースファイルリストの自動生成
- 多重分割あるいは静的ライブラリの簡潔なハンドリング
- 新しいタイプのマシンへの拡張性
- `make`；`sh`，`ksh` または `csh`；`lex`，`cc` をもついかなるマシンでの作業に対する優れた移植性
- Apollo, SUN, SGI, HP (HPUX), Compaq (DEC), IBM (AIX), Cray, Ardent, Stardent, PC Linux，PPC Linux，NEC，SX4，Fujitsu VP1000 での動作確認

3.2.2 wmake によるコンパイル

OpenFOAM のアプリケーションは各アプリケーションのソースコードがそのアプリケーション名のディレクトリに置かれるという一般的決まりで編成されます．最上位ソースファイルはアプリケーション名に拡張子 *.C* をつけます．例えば，newApp というアプリケーションのソースコードは図 3.2 に示すように newApp のディレクトリに存在し，最上位ファイルは *newApp.C* となります．

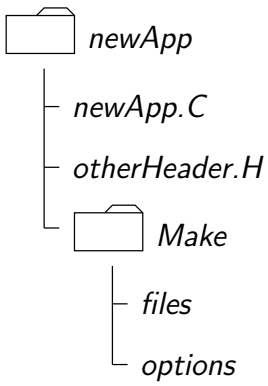


図 3.2 アプリケーションのディレクトリ構成

ディレクトリは *options* と *files* の二つのファイルを含んだ *Make* というサブディレクトリももっており，それについては次節で述べます．

3.2.2.1 ヘッダの読み込み

コンパイラは，以下の順で wmake で－ I オプションが指定されたヘッダファイルを検索します．

1. *$WM_PROJECT_DIR/src/OpenFOAM/lnInclude* ディレクトリ
2. *newApp/lnInclude* のようなローカルディレクトリ
3. *newApp* のようなローカルディレクトリ
4. */usr/X11/include* や *$(MPICH_ARCH_PATH)/include* のように，プラットフォームに依存する *$WM_PROJECT_DIR/wmake/rules/$WM_ARCH/* ディレクトリの中のファイルに設定されているパス
5. -I オプションをもつ *Make/options* ファイルの中で明確に指定されている他のディレクトリ

Make/options ファイルは構文を使っているヘッダファイルを配置するためのフルディレクトリパスを含みます．

```
EXE_INC = \
    -I<directoryPath1> \
    -I<directoryPath2> \
    ...                \
    -I<directoryPathN>
```

ディレクトリ名は頭に`-I` をつけ，各行では `EXE_INC` を続けるために構文は`\`を使い，最終記入後は`\`をつけないことに注意してください．

3.2.2.2　ライブラリへのリンク

コンパイラは，以下の wmake の`-L` オプションで指定されたディレクトリパスのオブジェクトライブラリファイルにリンクします．

1. *$FOAM_LIBBIN* ディレクトリ
2. *$WM_DIR/rules/$WM_ARCH/*ディレクトリの中に設定された機種に依存するパス，例えば，*/usr/X11/*や*$(MPICH_ARCH_PATH)/lib*
3. *Make/options* ファイルで指定された他のディレクトリ

リンクされる実際のライブラリファイルは`-l` オプションで指定し，接頭辞 *lib*，ライブラリファイルの拡張子*.so* を外さなければなりません．例えばライブラリ *libnew.so* はフラグ`-lnew` に含まれます．

デフォルトでは，wmake は以下のライブラリをロードするようになっています

1. *$FOAM_LIBBIN* ディレクトリからの *libOpenFOAM.so* ライブラリ
2. *$WM_DIR/rules/$WM_ARCH/*ディレクトリの中のファイルに設定された機種に依存するライブラリ，例えば，*/usr/X11/lib* における *libm.so* や，*$(LAM ARCH PATH)/lib* における *liblam.so*
3. *Make/options* ファイルで指定された他のライブラリ

Make/options ファイルは構文を使っているヘッダファイルをおくための全ディレクトリパスを含みます．

```
EXE_LIBS = \
    -L<libraryPath1> \
    -L<libraryPath2> \
    ...              \
    -L<libraryPathN> \
    -l<library1>     \
    -l<library2>     \
    ...              \
    -l<libraryN>
```

繰り返しになりますが，ディレクトリパスは頭に`-L` フラグを付け，ライブラリ名は頭に`-l` フラグをつけます．

3.2.2.3 コンパイルすべきソースファイル

コンパイラはコンパイルすべき.C ソースファイルのリストが必要です．リストはメインの.C ファイルだけではなく特定のアプリケーションのために生成されるがクラスライブラリの中に含まれない他のソースファイルも含まれなければなりません．例えば，新しいクラスを作成したり，特定のアプリケーション用のクラスに新しい機能をつけくわえることができます．.C ソースファイルのフルリストは *Make/files* ファイルに含む必要があります．当然，アプリケーションは多くなるので，フルリストには (例えば前述のアプリケーション例における *newApp.C* のような) メイン.C ファイルの名前だけを入れます．*Make/files* ファイルは `EXE =`構文によって指定されたコンパイル済み実行ファイルの名前とフルパスも含みます．一般的な決まりでは newApp のようにアプリケーション名をつけることが規定されています．OpenFOAM のリリースにはパスのために便利な二つの選択肢があります．標準的なリリースではアプリケーションは*$FOAM_APPBIN* に保存されますが，ユーザにより開発されたアプリケーションは*$FOAM_USER_APPBIN* に保存されます．

もしアプリケーションを開発したら，個人の OpenFOAM アプリケーションのためのソースコードを含む*$WM_PROJECT_USER_DIR* ディレクトリにアプリケーションサブディレクトリを作ることをお薦めします．スタンダードアプリケーションと同様に各 OpenFOAM アプリケーションのソースコードも各ディレクトリ内に保存しておいてください．ユーザーアプリケーションとスタンダードリリースのものの違いは *Make/files* ファイルが*$FOAM_USER_APPBIN* ディレクトリ内に書き込まれている実行可能ファイルを指定していることだけです．例としての *Make/files* を以下に記載します．

```
newApp.C

EXE = $(FOAM_USER_APPBIN)/newApp
```

3.2.2.4 wmake の実行

wmake のスクリプトは以下のように入力することで実行されます．

```
wmake <optionalArguments> <optionalDirectory>
```

`<optionalDirectory>` はコンパイルしようとしているアプリケーションのディレクトリパスです．通常，`<optionalDirectory>` が省略可能な場合には wmake はコンパイル中のアプリケーションのディレクトリ内から実行されます．

アプリケーションファイルを作成したい場合には `<optionalArguments>` は必要ありません．しかし `<optionalArguments>` は表 3.1 に示すようにライブラリ等の作成の際には指定されることになります．

Argument	コンパイルの種類
`lib`	静的にリンクされたライブラリの作成
`libso`	動的にリンクされたライブラリの作成
`libo`	静的にリンクされたオブジェクトファイルライブラリの作成
`jar`	JAVA アーカイブの作成
`exe`	特定のプロジェクトから独立したアプリケーションの作成

表 3.1 wmake のコンパイルオプション

3.2.2.5　wmake の環境変数

参考として，wmake で使われる環境変数の設定を表 3.2 に示します．

主なパス	
$WM_PROJECT_INST_DIR	インストールディレクトリへのフルパス，例：*$HOME/OpenFOAM*
$WM_PROJECT	コンパイルされたプロジェクトの名前：`OpenFOAM`
$WM_PROJECT_VERSION	コンパイルされたプロジェクトのバージョン：`2.3.0`
$WM_PROJECT_DIR	OpenFOAM のバイナリ実行ファイル置き場へのフルパス， 例：*$HOME/OpenFOAM/OpenFOAM-2.3.0*
$WM_PROJECT_USER_DIR	ユーザのバイナリ実行ファイル置き場へのフルパス， 例：*$HOME/OpenFOAM/$USER-2.3.0*
その他のパスと設定	
$WM_ARCH	マシン・アーキテクチャ：`Linux`, `SunOS`
$WM_ARCH_OPTION	32 あるいは 64 ビットのアーキテクチャ
$WM_COMPILER	使用するコンパイラ：`Gcc45` - gcc 4.5+, `ICC` - Intel
$WM_COMPILER_DIR	コンパイラインストールディレクトリ
$WM_COMPILER_BIN	コンパイラインストールバイナリ：*$WM_COMPILER_BIN/bin*
$WM_COMPILER_LIB	コンパイラインストールライブラリ：*$WM_COMPILER_BIN/lib*
$WM_COMPILE_OPTION	コンパイルオプション：`Debug` - debugging, `Opt` optimisation.
$WM_DIR	wmake ディレクトリのフルパス
$WM_MPLIB	並列通信ライブラリ：`LAM`，`MPI`，`MPICH`，`PVM`
$WM_OPTIONS	= WM_ARCHWM_COMPILER... ...$WM_COMPILE_OPTION$WM_MPLIB， 例：`linuxGcc3OptMPICH`
$WM_PRECISION_OPTION	コンパイルされたバイナリの精度，`SP` なら単精度，もしくは，`DP` なら倍精度

表 3.2　wmake の環境変数の設定

3.2.3　依存リストの削除：wclean と rmdepall

実行に際して，例題における *newApp.dep* のように，wmake は拡張子として *.dep* をもった依存関係のリストファイルを構築し，*Make/$WM_OPTIONS* ディレクトリの中にファイルのリストを格納します．コードを変更して make した後などこれらファイルを除去したい場合には，wclean を入力してスクリプトを実行します．

```
wclean <optionalArguments> <optionalDirectory>
```

さらに，`<optionalDirectory>` はコンパイルされるアプリケーションのディレクトリへのパスです．通常，パスが省略できる場合には wclean はアプリケーションのディレクトリ範囲内で実行されます．

依存ファイルと *Make* ディレクトリのファイルを削除したい場合には，`<optionalArguments>` は必要ありません．しかし，もし `lib` が `<optionalArguments>` に指定されていたらローカルの *lnInclude* ディレクトリも同時に削除されます．

追加のスクリプト，rmdepall は実行時に，再帰的にディレクトリツリー下の依存関係にあるすべての *.dep* ファイルを除去します．これは OpenFOAM のライブラリが更新されたときには有効な方法です．

3.2.4 コンパイルの例：pisoFoam アプリケーション

アプリケーション pisoFoam のソースコードは*$FOAM_APP/solvers/incompressible/pisoFoam* ディレクトリ内にあり，最上位ソースファイルは *pisoFoam.C* という名前です．*pisoFoam.C* ソースコードは

```
/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2013 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------------
License
    This file is part of OpenFOAM.

    OpenFOAM is free software: you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
    ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
    for more details.

    You should have received a copy of the GNU General Public License
    along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.

Application
    pisoFoam

Description
    Transient solver for incompressible flow.

    Turbulence modelling is generic, i.e. laminar, RAS or LES may be selected.

\*---------------------------------------------------------------------------*/

#include "fvCFD.H"
#include "singlePhaseTransportModel.H"
#include "turbulenceModel.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

int main(int argc, char *argv[])
{
    #include "setRootCase.H"

    #include "createTime.H"
    #include "createMesh.H"
    #include "createFields.H"
    #include "initContinuityErrs.H"

    // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

    Info<< "\nStarting time loop\n" << endl;

    while (runTime.loop())
    {
        Info<< "Time = " << runTime.timeName() << nl << endl;

        #include "readPISOControls.H"
        #include "CourantNo.H"

        // Pressure-velocity PISO corrector
        {
            // Momentum predictor
```

```
63
64              fvVectorMatrix UEqn
65              (
66                  fvm::ddt(U)
67                + fvm::div(phi, U)
68                + turbulence->divDevReff(U)
69              );
70
71              UEqn.relax();
72
73              if (momentumPredictor)
74              {
75                  solve(UEqn == -fvc::grad(p));
76              }
77
78              // --- PISO loop
79
80              for (int corr=0; corr<nCorr; corr++)
81              {
82                  volScalarField rAU(1.0/UEqn.A());
83
84                  volVectorField HbyA("HbyA", U);
85                  HbyA = rAU*UEqn.H();
86                  surfaceScalarField phiHbyA
87                  (
88                      "phiHbyA",
89                      (fvc::interpolate(HbyA) & mesh.Sf())
90                    + fvc::interpolate(rAU)*fvc::ddtCorr(U, phi)
91                  );
92
93                  adjustPhi(phiHbyA, U, p);
94
95                  // Non-orthogonal pressure corrector loop
96                  for (int nonOrth=0; nonOrth<=nNonOrthCorr; nonOrth++)
97                  {
98                      // Pressure corrector
99
100                     fvScalarMatrix pEqn
101                     (
102                         fvm::laplacian(rAU, p) == fvc::div(phiHbyA)
103                     );
104
105                     pEqn.setReference(pRefCell, pRefValue);
106
107                     if
108                     (
109                         corr == nCorr-1
110                      && nonOrth == nNonOrthCorr
111                     )
112                     {
113                         pEqn.solve(mesh.solver("pFinal"));
114                     }
115                     else
116                     {
117                         pEqn.solve();
118                     }
119
120                     if (nonOrth == nNonOrthCorr)
121                     {
122                         phi = phiHbyA - pEqn.flux();
123                     }
124                 }
125
126                 #include "continuityErrs.H"
127
128                 U = HbyA - rAU*fvc::grad(p);
129                 U.correctBoundaryConditions();
130             }
131         }
132
133         turbulence->correct();
```

```
134
135         runTime.write();
136
137         Info<< "ExecutionTime = " << runTime.elapsedCpuTime() << " s"
138             << "  ClockTime = " << runTime.elapsedClockTime() << " s"
139             << nl << endl;
140     }
141
142     Info<< "End\n" << endl;
143
144     return 0;
145 }
146
147
148 // ************************************************************************* //
```

コードはアプリケーションを説明している記述で始まり，この中で 1 行のコメントは `//` で，複数行にわたるコメントは `/*...*/` で記述されます．それに続き，コードはコンパイラに現在のファイルの読み込みを一時停止させ，*pisoFoam.C* に *fvCFD.H* を読み込ませるための例えば `#include "fvCFD.H"` のような様々な `# include` 命令文を含んでいます．

pisoFoam は incompressibleRASModels や incompressibleLESModels や incompressibleTransportModels ライブラリを提供し，それゆえ `EXE_INC = -I...` オプションとライブラリにリンクする `EXE_LIBS = -l...` オプションにより指定されるヘッダファイルが必要となります．*Make/options* はそれゆえ以下のようになります．

```
1  EXE_INC = \
2      -I$(LIB_SRC)/turbulenceModels/incompressible/turbulenceModel \
3      -I$(LIB_SRC)/transportModels \
4      -I$(LIB_SRC)/transportModels/incompressible/singlePhaseTransportModel \
5      -I$(LIB_SRC)/finiteVolume/lnInclude
6
7  EXE_LIBS = \
8      -lincompressibleTurbulenceModel \
9      -lincompressibleRASModels \
10     -lincompressibleLESModels \
11     -lincompressibleTransportModels \
12     -lfiniteVolume \
13     -lmeshTools
```

pisoFoam は *pisoFoam.C* ソースしか含まず，実行ファイルはすべての標準的なアプリケーションと同様に *$FOAM_APPBIN* に書き込まれます．*Make/files* はそれゆえ以下のようになります．

```
1  pisoFoam.C
2
3  EXE = $(FOAM_APPBIN)/pisoFoam
```

$FOAM_SOLVERS/incompressible/pisoFoam ディレクトリで `wmake` とタイプすれば pisoFoam をコンパイルできます．

コードはコンパイルされ以下のようなメッセージをが作成されます．

```
Making dependency list for source file pisoFoam.C
SOURCE DIR=.
SOURCE=pisoFoam.C ;
g++ -DFOAM EXCEPTION -Dlinux -DlinuxOptMPICH
-DscalarMachine -DoptSolvers -DPARALLEL -DUSEMPI -Wall -O2 -DNoRepository
-ftemplate-depth-17 -I.../OpenFOAM/OpenFOAM-2.3.0/src/OpenFOAM/lnInclude
-IlnInclude
-I.
......
```

```
    -lmpich -L/usr/X11/lib -lm
    -o .../OpenFOAM/OpenFOAM-2.3.0/applications/bin/linuxOptMPICH/pisoFoam
```

再コンパイルすることも可能ですが，実行ファイルが最新でコンパイルする必要がないというときには以下のようなメッセージが返ってきます．

```
make: Nothing to be done for ‘allFiles’.
make: ‘Make/linuxOptMPICH/dependencies’ is up to date.
make: ‘.../OpenFOAM/OpenFOAM-2.3.0/applications/bin/linuxOptMPICH/pisoFoam’
is up to date.
```

```
wclean
```

を使って依存リストを削除し，wmake を起動することでゼロからアプリケーションをコンパイルできます．

3.2.5　デバッグメッセージと最適化スイッチ

OpenFOAM は，実行時にメッセージを出力するシステムを提供しており，これらのメッセージの多くは，OpenFOAM のケースの実行時に遭遇する問題のデバッグに役立ちます．そのスイッチは*$WM_PROJECT_DIR/etc/controlDict* ファイルの中にあり，設定を変更したい場合には，*$HOME* ディレクトリに (例えば*$HOME/.OpenFOAM/2.3.0/controlDict* ファイルのように) コピーを作成します．スイッチが可能なリストは非常に多く，foamDebugSwitches アプリケーションを実行することにより閲覧できます．スイッチのほとんどは，クラスまたは機能性のレンジと一致しており，設定を 1 にすることにより，*controlDict* ファイルの中にあるそれ自身により変更できます．例えば，OpenFOAM では，`dimensionSet` スイッチを 1 に設定することにより，すべての計算におけるディメンションをチェックする機能があります．表 3.3 に示すものはより高機能にメッセージをコントロールできるスイッチです．

加えて，いくつかのオペレーションと最適化をコントロールするスイッチがあります．これらのスイッチについても表 3.3 に示します．特に重要なものは `fileModificationSkew` であり，OpenFOAM では，変更をチェックするためにデータファイルの書き込み時間をスキャンしています．異なるマシンでクロックの設定に不整合が生じた状態で NFS を実行すると，先駆けしてフィールドデータの修正が表示されます．このことは，OpenFOAM が新規に修正されたとしてファイルを閲覧する場合と，このデータを再読み込みしようとする場合には問題を引き起こすことになります．キーワード `fileModificationSkew` は秒単位の時間であり，OpenFOAM は，ファイルが新しく修正されたかどうか調べるときには，ファイルの書き込み時間から差し引きます．

3.2.6　現在のアプリケーションへの新しいユーザ定義ライブラリのリンク

タイトルのような状況は，新しいライブラリ (例えば new) を作成するとき，新しい宣言およびアプリケーションのレンジを越えてライブラリの中に入れ込みたい場合に生じることが考えられます．例えば，ユーザーが新規の境界条件を作成し，new の中でコンパイルし，ソルバのアプリケーションや，前および後処理用のユーティリティ，メッシュツール等々の範囲で認識

ハイレベルデバッグスイッチ - サブディクショナリ *DebugSwitches*	
`level`	OpenFOAM のデバッグメッセージの全体のレベル - - 0, 1, 2 の 3 レベル
`lduMatrix`	実行中のソルバの収束メッセージ - 0, 1, 2 の 3 レベル
最適化スイッチ - サブディクショナリ *OptimisationSwitches*	
`fileModificationSkew`	NFS の上で NFS のアップデートの最大遅れと OpenFOAM 実行のための差分クロックより長く設定すべき時間（秒）.
`fileModificationChecking`	シミュレーション中にファイルが変更されたかどうかをチェックする方法であり，`timeStamp` を読む，または `inotify`（通知しない），あるいはマスタ・ノードに存在するデータのみを読み込む `timeStampMaster`，`inotifyMaster` があります.
`commsType`	並列計算の通信方法．`nonBlocking`，`scheduled`，`blocking`.
`floatTransfer`	1 とすると，転送の前に数値を `float` の精度に丸めます．デフォルトは 0 です.
`nProcsSimpleSum`	並列処理のために全領域の和を最適化します．階層和は線形和（デフォルトで 16）よりよく機能し，プロセッサの数を設定します.

表 3.3 ランタイムメッセージスイッチ

させる必要があることがあります．通常の環境下では，ユーザはすべてのアプリケーションを，リンクさせるために new で再コンパイルする必要があります.

その代わりに OpenFOAM では，一つもしくは複数の共有オブジェクトライブラリを実行時に動的にリンクさせるメカニズムを採用しています．そのためには，ただ単に *controlDict* にオプションのキーワードである `libs` を追加し，ライブラリの完全なファイル名を引用符付き文字列のエントリとしてリストに入力すればすればよいだけです．たとえば new1 と new2 というライブラリを実行時にリンクしたいなら，*controlDict* に以下を書き加えます.

```
libs
(
    "libnew1.so"
    "libnew2.so"
);
```

3.3 アプリケーションの実行

各アプリケーションは，ターミナルのコマンドラインから実行されるようになっており，個別のケースに関連したデータファイルのセットの書き込みと読み込みが行われるようになっています．ケースに関するデータファイルは 4.1 節で述べているように，ケースの後に名前をつけられたディレクトリの中に格納されており，ここではフルパスをもつディレクトリ名は一般名<*caseDir*>としています.

どのアプリケーションにおいても，コマンドラインの入力フォームはコマンドラインでアプリーション名に`-help` オプションをつけて入力するだけで見つけられます．例えば

```
blockMesh -help
```

と入力すると以下を含むデータが返ってきます.

```
Usage: blockMesh [-region region name] [-case dir] [-blockTopology]
```

```
[-help] [-doc] [-srcDoc]
```

角括弧 [ ] 内の引数はオプショナルフラグです．アプリケーションがケースディレクトリ内で実行されると，そのケースを作動します．あるいは，-case <*caseDir*>オプションでは，直接ファイリングシステムでどこからでもアプリケーションを実行できるようにケースを指定することもできます．

すべての UNIX/Linux の実行方法と同様に，アプリケーションは，バックグラウンドのプロセスで実行しており，ユーザがシェルに追加コマンドを与える必要はありません．blockMesh のサンプルをバックグラウンドのプロセスで実行し，ケースの進捗をログファイルに出力したい場合には，以下のように入力します．

```
blockMesh > log &
```

3.4　アプリケーションの並列実行

この節では複数のプロセッサによる並列処理での OpenFOAM の実行方法について説明します．OpenFOAM による並列処理の方法はドメインの分割として知られており，ジオメトリと関連したフィールドを解析に用いるプロセッサに合わせてピースに分割します．並列処理には，メッシュとフィールドの分割と，並列でのアプリケーションの実行がありますが，分割したケースの前処理については以降の節で説明します．並列処理には，標準の MPI (message passing interface) の実装である openMPI というパブリックドメインを使用しています．

3.4.1　メッシュの分解と初期フィールド・データ

メッシュとフィールドは，decomposePar ユーティリティを用いて分割します．この根本的な目的は，最小限の労力でドメインを分割しつつ，解析の効率性を向上させようとするものです．ジオメトリとフィールドのデータは，*decomposeParDict* と名前のつけられたディクショナリの中で指定されたパラメータにより分割されますが，このディクショナリは対象とするケースの *system* ディレクトリの中におかれている必要があります．もしユーザが必要とする場合には，*interFoam/damBreak* チュートリアルから *decomposeParDict* ディクショナリをコピーすることができます．そして，ディクショナリ中のエントリを次のように置き換えます．

```
17 /*--------------------------------*- C++ -*----------------------------------*\
18 | =========                 |                                                 |
19 | \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
20 |  \\    /   O peration     | Version:  2.3.0                                 |
21 |   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
22 |    \\/     M anipulation  |                                                 |
23 \*---------------------------------------------------------------------------*/
24 FoamFile
25 {
26     version     2.0;
27     format      ascii;
28     class       dictionary;
29     location    "system";
30     object      decomposeParDict;
31 }
32 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
33
```

```
34  numberOfSubdomains 4;
35
36  method          simple;
37
38  simpleCoeffs
39  {
40      n               ( 2 2 1 );
41      delta           0.001;
42  }
43
44  hierarchicalCoeffs
45  {
46      n               ( 1 1 1 );
47      delta           0.001;
48      order           xyz;
49  }
50
51  manualCoeffs
52  {
53      dataFile        "";
54  }
55
56  distributed     no;
57
58  roots           ( );
59
60
61  // ************************************************************************* //
```

ユーザは，以下に述べる `method` キーワードにより指定できる四つの分割方法から選択します．

`simple` 簡単なジオメトリの分割：ドメインは x，y 方向に，例えば x 方向に二つに，y 方向一つにというように，ピースが分割されます．

`hierarchical` 階層的なジオメトリの分割方法：基本的には `simple` と同じですが，ユーザが，最初に y 方向を，次に x 方向を，というように，各方向の分割する順番を指定する点が異なっています．

`scotch` Scotch 分割はユーザからのジオメトリの入力を必要とせず，プロセッサの限界の数値を最小化するよう試みます．ユーザは，任意指定の `processorWeights` キーワードによりプロセッサ間の重み付けを行うことができるため，パフォーマンスの異なるマシン同士を有効に使うことができます．また，もう一つ `strategy` という任意のキーワードエントリがあり，複雑な文字列を Scotch に渡すことにより分割の戦略を制御できます．さらなる情報を得るには，ソースコードファイル *$FOAM_SRC/decompositionMethods/decompositionMethods/scotchDecomp/scotchDecomp.C* を読んでください．

`manual` マニュアルでの分割：個別のプロセッサに対して，各々のセルの割り当てを直接指定します．

これらの各 `method` については，ディクショナリのリストに示すように，`<method>coeffs` と名前の付けられた *decompositionDict* のサブディクショナリの中で指定された係数のセットがあります．*decompositionDict* ディクショナリの中にある入力のキーワードのフルセットの説明を，表 3.4 に示します．

decomposePar ユーティリティは以下のように入力することで正常に実行されます．

```
decomposePar
```

必須入力		
numberOfSubdomains	サブドメインの総数	N
method	分割方法	simple/ hierarchical/ scotch/ metis/ manual/
simpleCoeffs エントリ		
n	x，y，z のサブドメイン数	(n_x, n_y, n_z)
delta	セルのスキュー因数	一般的には，10^{-3}
hierarchicalCoeffs エントリ		
n	x，y，z のサブドメイン数	(n_x, n_y, n_z)
delta	セルのスキュー因数	一般的には，10^{-3}
order	分割の順序	xyz/xzy/yzx...
scotchCoeffs エントリ		
processorWeights （省略可）	プロセッサへのセルの割当の重み係数の一覧．例：<wt1>はプロセッサ 1 の重み係数．重みは規格化され，どんな範囲の値も取ることが可能．	(<wt1>...<wtN>)
strategy	分割の戦略（省略可）．デフォルトは"b"	
manualCoeffs エントリ		
dataFile	プロセッサへのセルの割当のデータを含むファイル名	"<fileName>"
分散型データの入力（省略可）—3.4.3 項参照		
distributed	データはいくつかのディスクのに分散しますか？	yes/no
roots	ケースディレクトリへのルートパス．例：<rt1> はノード 1 へのルートパス	(<rt1>...<rtN>)

表 3.4 *decompositionDict* ディクショナリのキーワード

最終的に，ケースディレクトリ内に各プロセッサに一つずつ一連のサブディレクトリが作成されるでしょう．そのディレクトリはプロセッサナンバを表す $N = 0, 1, \ldots$ を用いて *processorN* と名づけられ，そして分割されたフィールドの説明を含むタイムディレクトリや分解されたメッシュの説明を含む *constant/polyMesh* ディレクトリをもっています．

3.4.2 分解ケースの実行

分解された OpenFOAM のケースは MPI の openMPI を使って並列実行されます．

構成される LAM マルチコンピュータのホストマシンの名前があるファイルを作成する必要があります．ファイルには名前とパスを与えることができます．以下の記述では，フルパスを含んだ一般的な名前として<*machines*>としています．

この<*machines*>ファイルは，1 行ごとに 1 台のマシンのリストをもっています．これらの名前は，LAM のスタート時にマシンの*/etc/hosts* ファイルの中のホスト名と，完全に一致させる必要があります．リストには，openMPI を実行するマシンの名前をもたせる必要があります．ここに，マシンのノードは一つ以上のプロセッサをもっており，ノードの名称は cpu=n の登録に依存しますが，この n はノード上で openMPI が実行されるプロセッサの数です．

例として，aaa，二つのプロセッサをもつ bbb，ccc というマシン構成からマシン aaa をホストとして openMPI を実行させるものとします．<*machines*>は次のようにします．

```
aaa
bbb cpu=2
ccc
```

openMPI はそのとき以下の実行によって起動されます．
あるアプリケーションを mpirun を使って並列実行します．

```
mpirun --hostfile <machines> -np <nProcs>
    <foamExec> <otherArgs> -parallel > log &
```

ここにあげた<nProcs>はプロセッサーの数，<foamExec>は icoFoam のような実行可能なファイル名であり，アウトプットは *log* と名前の付けられたファイルに変更されています．例えば，*$FOAM_RUN/tutorials/incompressible/icoFoam* ディレクトリの中の *cavity* チュートリアルにおいて *icoFoam* を四つのノード上で走らせる場合には，以下のコマンドを実行させる必要があります

```
mpirun --hostfile machines -np 4 icoFoam -parallel > log &
```

3.4.3 複数のディスクへのデータの分配

例であげたように，ローカルのディスクのみのパフォーマンスを向上させるために，データファイルを分配する必要が生じる場合が考えられます．このようなケースでは，ユーザは異なるマシン間のケースディレクトリに対するパスを見つけなければなりません．その場合には，distributed と roots のキーワードを使って，パスを *decomposeParDict* ディクショナリの中に指定する必要があります．distributed のエントリが以下のように読み込まれなければなりません．

```
distributed  yes;
```

また，roots のエントリは，各々のノードである，<root0>，<root1>，...，のルートパスのリストとなっています．

```
roots
<nRoots>
(
   "<root0>"
   "<root1>"
   ...
);
```

<nRoots>はルートの数です．

各 *processorN* ディレクトリは，*decomposeParDict* ディクショナリの中で指定された各ルートパスにあるケースディレクトリの中に置かなければなりません．system ディレクトリや *constant* ディレクトリ中のファイルについてもまた，各々のケースディレクトリの中にある必要があります．*constant* ディレクトリの中のファイル類は必要となりますが，*polyMesh* ディレクトリは必要のないことに注意してください．

3.4.4　並列実行されたケースの後処理

並列実行されたケースの後処理時には，ユーザにふたつのオプションがあります．

完全なドメインとフィールドを再生するためにメッシュとフィールドの再構築を行う．ここではノーマルとして後処理を行うことができます．分割されたドメインを個別に引数で後処理を行う．

3.4.4.1　メッシュとデータの再構築

ケースが並列処理された後に，後処理によって再構築を行うことができます．ケースは，時刻ディレクトリの一つのセットの中にある各 *processorN* ディレクトリから，時刻ディレクトリのセットを合併操作することにより再構築されます．reconstructPar ユーティリティは，次のように，コマンドラインから実行することにより機能を発揮します

```
reconstructPar
```

データが異なるディスクに分散されるときには，最初に，再構築におけるローカルのケースディレクトリにコピーされる必要があります．

3.4.4.2　分解ケースの後処理

6.1 節に示すように paraFoam ポストプロセッサを使って分割された各ケースの後処理を行えます．シミュレーション全体はケースを再構築することで後処理できますし，またはその代わりに個々のプロセッサディレクトリをそれ自体でひとつのケースとして扱うことで個々に分解されたドメインのセグメントを後処理することもできます．

3.5　標準のソルバ

OpenFOAM のディストリビューションのソルバは*$FOAM_SOLVERS* ディレクトリの中にあり，コマンドラインから `sol` と入力すれば素早く到達できます．このディレクトリはさらに，非圧縮流体のような連続体力学，対流および固体応力解析等のカテゴリにより，いくつかのディレクトリに再分割されています．各ソルバには，非圧縮性・層流の icoFoam ソルバといったように分かりやすい名前がつけられています．この OpenFOAM で提供されているソルバのリストを表 3.5 に示します．

基礎的な CFD コード

laplacianFoam	固体の熱拡散のような単純な Laplace 方程式を解く
potentialFoam	単純なポテンシャル流のソルバ．完全な Navier–Stokes 用コードを解く際の初期値の生成にも使用できる
scalarTransportFoam	パッシブスカラの輸送方程式を解く

非圧縮性流れ

adjointShapeOptimizationFoam	随伴形式を用いて圧力損失を生じると推定された領域に，「ブロッケージ」を適用することで管路形状を最適化する，非圧縮性・乱流の非ニュートン流体用定常ソルバ
boundaryFoam	1 次元の非圧縮性・乱流用の定常状態ソルバで，通常，解析では流入口で境界層条件を発生させます．
icoFoam	ニュートン流体の非圧縮性，層流の速度-圧力ソルバ
nonNewtonianIcoFoam	非ニュートン流体の非圧縮性，層流の非定常ソルバ
pimpleDyMFoam	移動メッシュでのニュートン流体の非圧縮性・乱流の PIMPLE (SIMPLE と PISO の融合) アルゴリズムによる非定常ソルバ
pimpleFoam	PIMPLE (SIMPLE と PISO の融合) アルゴリズムによる非圧縮性・乱流の，大きな時間ステップの非定常ソルバ
pisoFoam	非圧縮性流れの非定常ソルバ
porousSimpleFoam	多孔性を陰的または陽的に扱う非圧縮性，乱流のための定常状態ソルバ
shallowWaterFoam	回転を伴う非粘性浅水方程式の非定常ソルバ
simpleFoam	非圧縮性・乱流の定常状態ソルバ
SRFSimpleFoam	一つの回転フレームにおける非ニュートン流体の，非圧縮性・乱流の定常状態ソルバ
SRFPimpleFoam	一つの回転フレームにおける非圧縮性流体に対する，PIMPLE (PISO と SIMPLE の融合) アルゴリズムを用いた大きな時間ステップの非定常ソルバ

圧縮性流れ

rhoCentralDyMFoam	移動メッシュおよび乱流モデルに対応した，Kurganov と Tadmor の中央風上スキームに基づいた密度ベースの圧縮性流ソルバ
rhoCentralFoam	Kurganov と Tadmor の中央風上スキームに基づいた密度ベースの圧縮性流ソルバ
rhoLTSPimpleFoam	実行時に MRF や多孔性などの有限体積法オプションを選択できる，圧縮性の層流および乱流用の定常状態ソルバ
rhoPimplecFoam	冷暖房やそれに似た問題のための圧縮性の層流および乱流用の非定常ソルバ
rhoPimpleFoam	冷暖房やそれに似た問題のための圧縮性の層流および乱流用の非定常ソルバ
rhoPorousSimpleFoam	RANS 乱流モデル，多孔性の陰的または陽的取り扱い，実行時に有限体積法のソース項を選択できる乱流の圧縮性流体のための定常ソルバ
rhoSimplecFoam	層流および RANS による乱流の圧縮性流体用定常状態 SIMPLEC ソルバ
rhoSimpleFoam	層流および RANS による乱流の圧縮性流体用定常状態 SIMPLE ソルバ
sonicDyMFoam	移動メッシュを伴う，遷音速または超音速用の，層流および乱流の圧縮性気体用非定常ソルバ
sonicFoam	遷音速または超音速用の，層流および乱流の圧縮性気体用非定常ソルバ
sonicLiquidFoam	遷音速または超音速用の，層流圧縮性液体用非定常ソルバ

多層流

cavitatingDyMFoam	均質な平衡モデルに基づいて，液体・蒸気の混合物の圧縮率を得る非定常のキャビテーション用コード．オプションとしてメッシュの移動や，適応再メッシングを含むメッシュのトポロジ変化をサポートする．
cavitatingFoam	均質な平衡モデルに基づいて，液体・蒸気の混合物の圧縮率を得る非定常のキャビテーション用コード

compressibleInterDyMFoam	VOF (volume of fluid) 相比率に基づいた界面捕獲法による不混和流体の圧縮性・非等温 2 相流用ソルバ．オプションとしてメッシュの移動や，適応再メッシングを含むメッシュのトポロジ変化をサポートする．
compressibleInterFoam	VOF (volume of fluid) 相比率に基づいた界面捕獲法による不混和流体の圧縮性・等温 2 相流用ソルバ
compressibleMultiphaseInterFoam	VOF (volume of fluid) 相比率に基づいた界面捕獲法による不混和流体の圧縮性・非等温 n 相流用ソルバ
interFoam	VOF (volume of fluid) 相比率に基づいた界面捕獲法による不混和流体の非圧縮性・等温 2 相流用ソルバ
interDyMFoam	任意でメッシュ移動や，アダプティブ再メッシングを含むメッシュのトポロジ変化を伴う，VOF (volume of fluid) 相比率に基づいた界面捕獲法による不混和流体の非圧縮性・等温 2 相流用ソルバ
interMixingFoam	界面捕獲のために VOF 法を用いた非圧縮性 3 相流（うち二つは混和性）用ソルバ
interPhaseChangeFoam	相変化（キャビテーションなど）を伴なう，不混和流体の非圧縮性・等温 2 相流用ソルバ．VOF (volume of fluid) 相比率に基づいた界面捕獲法を用いる．
interPhaseChangeDyMFoam	相変化（キャビテーションなど）を伴なう，不混和流体の非圧縮性・等温 2 相流用ソルバ．VOF (volume of fluid) 相比率に基づいた界面捕獲法を用いる．メッシュ移動や，アダプティブ再メッシングを含むメッシュのトポロジ変化も扱える．
LTSInterFoam	VOF (volume of fluid) 相比率に基づいた界面捕獲法による非圧縮性・等温・不混和な 2 相流の局所時間ステップ (LTS, 定常状態) ソルバ
MRFInterFoam	VOF (volume of fluid) 相比率に基づいた界面捕獲法による非圧縮性・等温・不混和な 2 相流の複合参照フレーム (MRF) ソルバ
MRFMultiphaseInterFoam	界面捕獲と，それぞれの相での接触角効果を考慮した非圧縮性 n 相流用の複合参照フレーム (MRF) ソルバ
multiphaseEulerFoam	熱伝達を含む圧縮性多相流体系のソルバ
multiphaseInterFoam	界面捕獲と，それぞれの相での接触角効果を考慮した非圧縮性 n 相流ソルバ
porousInterFoam	多孔質領域の陽的な扱いを備えた，VOF (volume of fluid) 相比率に基づいた界面捕獲法による非圧縮性・等温・不混和な 2 相流用ソルバ
potentialFreeSurfaceFoam	単相の自由表面近似を使えるように波高さのフィールドを含んだ，非圧縮性の Navier–Stokes 方程式ソルバ
settlingFoam	分散相の設定シミュレーション用の非圧縮 2 相流コード
twoLiquidMixingFoam	2 層の非圧縮性流れを混合したソルバ
twoPhaseEulerFoam	液体の中の気体の泡のように分散した状態の 2 層の非圧縮性流れのシステム

直接数値シミュレーション (DNS)

dnsFoam	直方体中の等方性乱流のための直接数値解法 (DNS) コード

燃焼

chemFoam	化学問題のためのソルバ．他の化学ソルバとの比較用に，単一セル上で使うように作られています．単一セルのメッシュはソルバ内でその場で作成され，場も初期条件からソルバ内でその場で作成されます．
coldEngineFoam	内燃機関のコールドフローのソルバ
engineFoam	エンジン内部の燃焼用ソルバ
fireFoam	火炎と乱流拡散炎のための非定常ソルバ

LTSReactingFoam	圧縮性の層流・乱流，化学反応あり・なしを扱える定常問題用の局所時間ステップ (LTS) ソルバ
PDRFoam	乱流モデルを伴う圧縮性予混合または部分予混合燃焼用ソルバ
reactingFoam	化学反応を伴う燃焼用ソルバ
rhoReactingBuoyantFoam	密度ベースの熱力学パッケージと，改良された浮力の処理を用いた，化学反応を伴う燃焼用ソルバ
rhoReactingFoam	密度ベースの熱力学パッケージによる化学反応を伴う燃焼用ソルバ
XiFoam	乱流モデルを伴う圧縮性予混合または部分予混合燃焼用コード

熱輸送と浮力駆動流れ

buoyantBoussinesqPimpleFoam	浮力を伴う非圧縮性乱流用非定常ソルバ
buoyantBoussinesqSimpleFoam	浮力を伴う非圧縮性乱流用定常状態ソルバ
buoyantPimpleFoam	換気・熱輸送のための，浮力を伴う圧縮性乱流用非定常ソルバ
buoyantSimpleFoam	浮力を伴う圧縮性乱流用定常状態ソルバ
chtMultiRegionFoam	個体領域と流体領域の間の熱輸送を連成するため，heatConductionFoam と buoyantFoam を融合させたもの
chtMultiRegionSimpleFoam	chtMultiRegionFoam の定常版

粒子追跡流

coalChemistryFoam	石炭・石灰石パーセルの噴射，エネルギ源，および燃焼を伴う圧縮性乱流用非定常ソルバ
DPMFoam	連続した相の体積率の影響を伴う単一の運動学的粒子群の練成した輸送の非定常ソルバ
icoUncoupledKinematicParcelDyMFoam	単一の運動学的粒子群の受動的輸送用の非定常ソルバ
icoUncoupledKinematicParcelFoam	単一の運動学的粒子群の受動的輸送用の非定常ソルバ
LTSReactingParcelFoam	質量，運動量，エネルギの陽的なソースを含む，多孔質媒体の多相 Lagrange 型パーセルの反応の有無を扱える圧縮性の層流または乱流で定常問題用の局所時間ステップ (LTS) ソルバ
reactingParcelFilmFoam	Lagrange 型パーセルの反応と表面膜のモデリングを伴う圧縮性層流・乱流用非定常 PISO ソルバ
reactingParcelFoam	Lagrange 型パーセルの反応を伴う圧縮性層流・乱流用非定常 PIMPLE ソルバ．実行時にソース項や制約条件といった有限体積法オプションを選択できる．
simpleReactingParcelFoam	Lagrange 型パーセルの反応を伴う圧縮性層流・乱流用定常状態 SIMPLE ソルバ．実行時にソース項や制約条件といった有限体積法オプションを選択できる．
sprayEngineFoam	噴霧パーセルを伴うエンジン流の，圧縮性層流・乱流の非定常 PIMPLE ソルバ
sprayFoam	噴霧パーセルを伴う，圧縮性層流・乱流の非定常 PIMPLE ソルバ
uncoupledKinematicParcelFoam	単一の運動学的粒子群の受動的輸送用の非定常ソルバ

分子動力学法

mdEquilibrationFoam	分子動力学系の平衡化や前処理を行う
mdFoam	流体力学のための分子動力学ソルバ

直接シミュレーション・モンテ・カルロ法

dsmcFoam	3 次元で非定常な多化学種流れ用の直接シミュレーション・モンテ・カルロ (DSMC) 法ソルバ

電磁流体

electrostaticFoam	静電方程式ソルバ

magneticFoam	永久磁石により印加される磁場のソルバ
mhdFoam	磁場の影響によって誘発される非圧縮性層流の電磁流体 (MHD) 用ソルバ
固体応力解析	
solidDisplacementFoam	線形弾性や固体の微小ひずみの非定常分離型有限体積ソルバ．熱拡散と熱応力も扱える．
solidEquilibriumDisplacementFoam	固体の線形弾性や微小ひずみの定常状態分離有限体積ソルバ．熱拡散と熱応力も扱える．
金融工学	
financialFoam	物価に対する Black–Scholes 方程式を解く

表 3.5　標準ライブラリソルバ

3.6　標準のユーティリティ

OpenFOAM で提供されているユーティリティは*$FOAM_UTILITIES* ディレクトリの中にあり，コマンドラインで `util` と打つことにより簡単にアクセスできます．名称は内容を記述するようになっており，例えば，ideasToFoam は I-DEAS のフォーマットで書かれたデータを OpenFOAM のフォーマットに変換します．OpenFOAM で配布されている最新のユーティリティリストを表 3.6 に示しておきます．

前処理	
applyBoundaryLayer	1/7 乗則に基づいて，速度場と乱流場に簡易的な境界層モデルを適用する．
applyWallFunctionBoundaryConditions	OpenFOAM の RAS ケースを，新しい（バージョン 1.6 の）壁関数を使うように更新する．
boxTurb	与えられたエネルギスペクトルに適合し，連続の式を満す乱流の box を生成する
changeDictionary	ディクショナリのエントリを変更するユーティリティ．たとえば，フィールドと *polyMesh/boundary* ファイルのパッチタイプを変更するときなどに使える．
createExternalCoupledPatchGeometry	externalCoupled 境界条件で使うためのパッチ形状（点および面）を生成する．
dsmcInitialise	初期化ディクショナリ *system/dsmcInitialise* に従って，dsmcFoam 用にケースを初期化する
engineSwirl	エンジン計算のために旋回流を発生させる
faceAgglomerate	形態係数放射モデルの解析用に境界面をグルーピングして，細かい格子から粗い格子へのマップを書き出す．
foamUpgradeCyclics	分離された周期境界のメッシュや場を更新するツール
foamUpgradeFvSolution	*system/fvSolution::solvers* の書式を更新する簡易ツール
mapFields	両ケースの時刻ディレクトリの全ての場を読み込み，補間し，体積場を一つのメッシュから他のメッシュにマップする．並列・非並列のどちらのケースでも再構築せずに実行可能
mdInitialise	分子動力学 (MD) シミュレーションのフィールドを初期化する．
setFields	ディクショナリによって，選択されたセル・パッチのセット上に値を設定する．
viewFactorGen	グルーピングされた面 (faceAgglomerate) に基づいて，形態係数放射モデルで使うための形態係数を計算する．
wallFunctionTable	乱流の壁関数で使用される表を生成する．

メッシュ生成

blockMesh	マルチブロック・メッシュのジェネレータ
extrudeMesh	既存のパッチやファイルから読み込んだパッチを（デフォルトでは面の外側へ，オプションで反転して）押し出す．
extrude2DMesh	2D メッシュ（すべての面が 2 点で，前後の面がない）を読み込み，与えられた厚さに押し出すことで 3D メッシュをつくる．
extrudeToRegionMesh	`faceZones` を個別のメッシュに（別の領域として）押し出す．例えば，液体の膜領域を作るために
foamyHexMesh	等角ボロノイ自動メッシュ生成ツール
foamyHexMeshBackgroundMesh	foamyHexMesh で構成されたように背景メッシュを書き出し，distanceSurface を構成する．
foamyHexMeshSurfaceSimplify	再抽出により面を単純化する．
foamyQuadMesh	等角ボロノイ 2 次元押し出しによる自動メッシャ
snappyHexMesh	自動分割六面体メッシャ．細分化して面にスナップする．

メッシュの変換

ansysToFoam	I-DEAS から出力した ANSYS インプットメッシュファイルを OpenFOAM 形式へ変換する
cfx4ToFoam	CFX 4 メッシュを OpenFOAM 形式へ変換する
datToFoam	datToFoam (.dat) メッシュファイル内を読み，*points* ファイルを出力する．blockMesh との結合に使われる．
fluent3DMeshToFoam	Fluent のメッシュを OpenFOAM 形式に変換する
fluentMeshToFoam	Fluent のメッシュを OpenFOAM 形式に変換する．複数の領域と，領域の境界の処理も扱える
foamMeshToFluent	OpenFOAM メッシュを Fluent メッシュ形式で出力する
foamToStarMesh	OpenFOAM メッシュを読み込み，PROSTAR (v4) の bnd/cel/vrt フォーマットに書き出す
foamToSurface	OpenFOAM のメッシュを読み込み，面のフォーマットで境界を書き出す．
gambitToFoam	GAMBIT メッシュを OpenFOAM 形式へ変換する
gmshToFoam	Gmsh によって書かれた*.msh* ファイルを読み込む
ideasUnvToFoam	I-DEAS *unv* フォーマットのメッシュ変換
kivaToFoam	KIVA グリッドを OpenFOAM 形式へ変換する
mshToFoam	Adventure システムによって作られた*.msh* 形式を読み込む
netgenNeutralToFoam	Netgen v4.4 によって書かれた Neutral ファイルフォーマットを変換する
plot3dToFoam	Plot3d メッシュ（アスキー形式）を OpenFOAM 形式に変換
sammToFoam	STAR-CD (v3) SAMM メッシュを OpenFOAM 形式へ変換する
star3ToFoam	STAR-CD (v3) PROSTAR メッシュを OpenFOAM 形式へ変換する
star4ToFoam	STAR-CD (v4) PROSTAR メッシュを OpenFOAM 形式へ変換する
tetgenToFoam	tetgen により書かれた .ele，.node，.face ファイルを変換する
vtkUnstructuredToFoam	VTK/ParaView により作られたアスキーの .vtk（旧形式）ファイルを変換する
writeMeshObj	メッシュのデバッグのため：たとえば javaview で見れる，三つの別々の OBJ ファイルとしてメッシュを書く

メッシュの操作

attachMesh	指定されたメッシュ修正ユーティリティによってトポロジ的に独立したメッシュを付加する
autoPatch	ユーザが指定した角度に基づいて外部面をパッチに分割する
checkMesh	メッシュの妥当性をチェックする

createBaffles	内部面を境界面にする．mergeOrSplitBaffles と異なり，点の複製はしない．
createPatch	選択した境界面の外部にパッチを作成する．面は既存のパッチか faceSet から選択する
deformedGeom	polyMesh を変位場 U と引数として与えられた尺度因子により変形させる
flattenMesh	2 次元デカルトメッシュの前後の面を平らにする
insideCells	面の内側に中心があるセルを抽出する．面は閉じていて単一である必要がある
mergeMeshes	二つのメッシュを合体させる
mergeOrSplitBaffles	同じ点を共有する面（バッフル）を探索し，それらの面をマージ，もしくは点を複製する．
mirrorMesh	与えられた面に対してメッシュの鏡映をつくる．
moveDynamicMesh	メッシュの移動とトポロジ変化のユーティリティ
moveEngineMesh	エンジン解析のためにメッシュを動かすソルバ
moveMesh	メッシュを動かすソルバ
objToVTK	obj 線（面ではない）のファイルを読み込み，vtk に変換する
orientFaceZone	faceZone の方向を修正する．
polyDualMesh	polyMesh の双対を求め，特徴線やパッチの辺を忠実に再現する．
refineMesh	複数の方向にセルを細分化する．
renumberMesh	行列の帯幅を狭くするためにセルの順番を付け直す．全ての時刻ディレクトリから全ての計算領域を読み込み，順番を付け直す
rotateMesh	メッシュおよび場を方向 $\boldsymbol{n}_1$ から方向 $\boldsymbol{n}_2$ へと回転させる
setSet	セル・面・点のセットやゾーンをインタラクティブに操作する
setsToZones	メッシュに pointZones/faceZones/cellZones を，同様に名づけられた pointSets/faceSets/cellSets から追加する
singleCellMesh	全てのフィールドを読み込み，内側の面が全て取り除かれたメッシュ (singleCellFvMesh) に変換して singleMesh 領域に書き出す．境界のみのデータとして使えるメッシュとフィールドを生成するのに使う．しかし不正なメッシュになりやすいので，ParaView で境界を見ることにのみ使う．
splitMesh	内部の面の外面を作ることでメッシュを分割する．attachDetach を用いる
splitMeshRegions	メッシュを複数の領域に分割する
stitchMesh	メッシュを縫う
subsetMesh	cellSet に基づいたメッシュ領域を選択する
topoSet	ディクショナリによって faceSets/cellSets/pointSets を操作する．
transformPoints	平行移動，回転，拡大・縮小のオプションにしたがって，*polyMesh* ディレクトリのメッシュの点を変形させる
zipUpMesh	有効な形をもった全ての多面体のセルが閉じていることを確実にするために，ぶら下がった頂点をもつメッシュを読み込み，セルを閉じる

その他のメッシュ・ツール

autoRefineMesh	境界面付近のセルを細分化するユーティリティ
collapseEdges	短い辺をつぶし，また複数の辺を結合して一つの線分にする
combinePatchFaces	同じセル内でパッチの重複した面をチェックし結合する．重複した面は，例えば細分化された隣接セルが削除されたり，同じセルに属する 4 面が取り残された結果として現れる．
modifyMesh	メッシュ要素を操作する
PDRMesh	PDR タイプのシミュレーションのためのメッシュおよび場の調整ユーティリティ
refineHexMesh	セルを $2 \times 2 \times 2$ に分割して六面体メッシュを細分化する

refinementLevel	細分化されたデカルト・メッシュの細分化レベルを判別する．スナップの前に実行すること
refineWallLayer	パッチに隣接するセルを細分化するユーティリティ
removeFaces	面を削除し両隣のセルを結合するユーティリティ
selectCells	面との関連でセルを選択する
splitCells	平面でセルを分割するユーティリティ

画像の後処理

ensightFoamReader	変換せずに OpenFOAM のデータを直接読むための EnSight のライブラリ・モジュール

データ変換の後処理

foamDataToFluent	OpenFOAM データを Fluent 形式へ変換する
foamToEnsight	OpenFOAM データを EnSight 形式へ変換する
foamToEnsightParts	OpenFOAM データを EnSight 形式へ変換する．それぞれのセル・ゾーンとパッチに対して Ensight パーツが作られる
foamToGMV	OpenFOAM の出力を GMV で読めるファイルに変換する．
foamToTecplot360	Tecplot バイナリファイル形式のライタ．
foamToVTK	レガシーな VTK ファイル形式のライタ．
smapToFoam	STAR-CD SMAP データファイルを OpenFOAM の計算領域の形式に変換する

速度場の後処理

Co	`phi` 場から Courant 数 Co を計算し，volScalarField として書き出す
enstrophy	速度場 `U` のエンストロフィを計算し，書き出す
flowType	速度場 `U` の flowType を計算し，書き出す
Lambda2	速度勾配テンソルの対称，非対称部分の正方形の合計のうち 2 番目に大きな固有値を計算し，書き出す
Mach	各時刻の速度場 `U` から局所 Mach 数を計算し，オプション指定により書き出す
Pe	`phi` 場から Peclet 数 Pe を計算し，surfaceScalarField として書き出す
Q	速度勾配テンソルの第 2 不変量を計算し，書き出す
streamFunction	各時刻の速度場 `U` の流れ機能を計算し，書き出す
uprime	`uprime` $(\sqrt{2k/3})$ のスカラ場を計算し，書き出す
vorticity	速度場 `U` の渦度を計算し，書き出す

応力場の後処理

stressComponents	各時刻の応力テンソル `sigma` の六つの要素のスカラ場を計算し，書き出す

スカラ場の後処理

pPrime2	各時刻の `pPrime2` $([p-\bar{p}]^2)$ のスカラ場を計算し，書き出す

壁の後処理

wallGradU	壁における `U` の勾配を計算し，書き出す
wallHeatFlux	`volScalarField` の境界面として全てのパッチに対する熱流束を計算し，書き出す．そして全ての壁について積分した熱流量も書き出す
wallShearStress	RAS 乱流モデルを使用しているとき，指定した時刻における壁面せん断応力を計算して書き出す
yPlusLES	LES 乱流モデルを使用しているとき，指定した時刻について，各壁面における `yPlus` を計算する
yPlusRAS	RAS 乱流モデルを使用しているとき，指定した時刻について，各壁面における `yPlus` を計算する

乱流の後処理

createTurbulenceFields	乱流場を表すすべての変数を生成する
R	現在の時間ステップについて，レイノルズ応力 R を計算して書き出す

パッチの後処理

patchAverage	指定したフィールドの指定したパッチにわたる平均を計算する
patchIntegrate	指定したフィールドの指定したパッチにわたる積分を計算する

ラグランジアン・シミュレーションの後処理

particleTracks	パーセル追跡タイプの雲を使って計算されたケースの粒子の飛跡を VTK ファイルに書き出す．
steadyParticleTracks	定常状態の雲を使って計算されたケースの粒子の飛跡を VTK ファイルに書き出す．注意：使う前に（並列で計算しているなら）ケースを再構築しておく必要がある．

サンプリングの後処理

probeLocations	位置での値を調べる
sample	選択した補間スキーム，サンプリング・オプション，書き出しフォーマットに従って，フィールドのデータをサンプリングする．

様々な後処理

dsmcFieldsCalc	DSMC 計算による広域的に平均化された場から，U や T といった集約的な場を計算する
engineCompRatio	幾何的な圧縮比を計算する．BDC と TCD で体積を計算するので，バルブと非有効体積があるかどうか注意すること
execFlowFunctionObjects	選択された時間セットに対して，選択されたディクショナリ (デフォルトでは *system/controlDict*) で指定された関数オブジェクトのセットを実行する．代わりのディクショナリは *system*/フォルダ以下に置く．
foamCalc	指定された時刻におけるフィールド計算の汎用ユーティリティ．
foamListTimes	timeSelector を使って時刻をリスト化する．
pdfPlot	確率密度関数のグラフを生成する．
postChannel	チャンネル流計算のポストプロセスデータ
ptot	時刻ごとに全圧を計算する
wdot	時刻ごとに wdot を計算し，書き出す
writeCellCentres	閾値制限してポストプロセスで使えるよう，格子中心の三つの座標値を volScalarField として書き出す

面メッシュ (例えば STL) ツール

surfaceAdd	二つの面を加える．点を幾何学的に同化させる．三角形の重複や交差のチェックは行わない．
surfaceAutoPatch	特性角度によって面をパッチにする．autoPatch と同様．
surfaceBooleanFeatures	二つの面についてのブーリアン演算のインタフェースのための `extendedFeatureEdgeMesh` を生成する．
surfaceCheck	幾何的・トポロジ的な面の品質をチェックする．
surfaceClean	- バッフルを除去，- 小さなエッジをつぶして三角形を除去，- 細長い三角形の頂点を底辺に射影してエッジに分解する．
surfaceCoarsen	'bunnylod' を使って面を粗くする．
surfaceConvert	ある面メッシュの書式を他のものに変換する．
surfaceFeatureConvert	edgeMesh の書式との変換を行う．
surfaceFeatureExtract	面要素を取り出し，ファイルに書き込む．
surfaceFind	近くの面と頂点を見つける．

surfaceHookUp	近接した開いたエッジを見つけて，それらに沿った面を縫い合わせる．
surfaceInertia	コマンドラインで指定された triSurface の慣性テンソル・慣性主軸・慣性モーメントを計算する．ソリッドもしくは薄い殻の慣性が算出可能．
surfaceLambdaMuSmooth	lambda/mu スムージングを用いて面を滑らかにする．ラプラシアンスムージングを取得 (以前の surfaceSmooth の挙動) する際に，lambda を緩和係数に，mu をゼロにセットする．
surfaceMeshConvert	オプションで coordinateSystem 上でのスケーリングや変形（回転・移動）を伴って，面のフォーマットを変換する．
surfaceMeshConvertTesting	ある面メッシュのフォーマットを他のものに変換する．ただし現時点では試験的な機能．
surfaceMeshExport	オプションで coordinateSystem 上でのスケーリングや変形（回転・移動）を伴って，surfMesh をさまざまなサードパーティの面フォーマットにエクスポートする．
surfaceMeshImport	オプションで coordinateSystem 上でのスケーリングや変形（回転・移動）を伴って，さまざまなサードパーティの面フォーマットから surfMesh にインポートする．
surfaceMeshInfo	面メッシュに関するさまざまな情報
surfaceMeshTriangulate	polyMesh から triSurface を取り出す．アウトプットの面フォーマットに依存して，面を三角形にする．三角形の領域番号は polyMesh のパッチ番号になる．オプションで名前のついたパッチのみを三角形にする．
surfaceOrient	ユーザが与えた「外側の」点に従って，法線を設定する．`-inside` を使うと，与えた点は内側とみなされる．
surfacePointMerge	面上で，絶対距離以内にある点をマージする．絶対距離であることに注意．
surfaceRedistributePar	triSurface を（再）配置する．分解されていない面またはすでに分解された面を，それぞれのプロセッサがそのメッシュにオーバラップする三角形全てをもつように再配置する．
surfaceRefineRedGreen	三角形の三辺全てを分割して精密化する (‘red’ 精密化)．(精密化するようマークされていない) 隣り合う三角形は半分にする (‘green’ 精密化)．(R. Verfuerth, “A review of a posteriori error estimation and adaptive mesh refinement techniques”, Wiley–Teubner, 1996)
surfaceSplitByPatch	triSurface の領域を個別のファイルに書きだす．
surfaceSplitByTopology	面の邪魔板部分を全てはぎ取る．
surfaceSplitNonManifolds	複雑に接続された面を，点を複製することで複雑に接続されたエッジにおいて分割しようとする．コンセプトを紹介すると，- borderEdge は四つの面が接続しているエッジ，- borderPointe はちょうど二つの borderEdge が接続している点，- borderLine は borderEdge の接続リスト．
surfaceSubset	triSurface の必要な部分だけを選択する面の解析ツール．subsetMesh に基づいている．
surfaceToPatch	面を読み込み，メッシュに面の領域を適用する．難しい作業には boundaryMesh を使う．
surfaceTransformPoints	面を変形（拡大縮小・回転）する．transformPoints と同様であるが対象は面．

並行処理

decomposePar	OpenFOAM の並列計算用にケースのメッシュと計算領域を自動的に分割する
redistributePar	*decomposeParDict* ファイルの設定に従って分割されたメッシュとフィールドを再分配する．
reconstructParMesh	幾何情報のみを使ってメッシュを再構築する

熱物理に関連したユーティリティ

adiabaticFlameT	与えられた燃料の種類・燃焼していない気体の温度と平衡定数に対して断熱状態の炎の温度を計算する
chemkinToFoam	CHEMKIN 3 の熱運動と反応のデータファイルを OpenFOAM のフォーマットに変換する
equilibriumCO	一酸化炭素の平衡状態を計算する
equilibriumFlameT	与えられた燃料の種類・燃焼していない気体の温度と平衡定数に対して酸素，水，二酸化炭素の分離の影響を考慮して平衡状態の炎の温度を計算する
mixtureAdiabaticFlameT	与えられた混合・温度に対して断熱状態の炎の温度を計算する

様々なユーティリティ

expandDictionary	引数として与えられたディクショナリを読み込み，マクロなどを展開した結果を標準出力に書き出す
foamDebugSwitches	すべてのライブラリのデバッグスイッチを書き出す
foamFormatConvert	*controlDict* に指定された書式に従って，ケースに関わる `IOobject` をすべて変換する
foamHelp	OpenFOAM のヘルプユーティリティまわりのトップレベルのラッパユーティリティ
foamInfoExec	ケースを調べ，標準出力に情報を書きだす．
patchSummary	指定された時刻について，各パッチに対するフィールドと境界条件を書き出す

表 3.6　標準ライブラリユーティリティ

3.7　標準のライブラリ

OpenFOAM 配布のライブラリは*$FOAM_LIB/$WM_OPTIONS* ディレクトリ内にあり，コマンド欄に `lib` と入力すればすぐに見つかります．一方，名前は `lib` を前につけて，例えば incompressibleTransportModels が非圧縮性の輸送モデルのライブラリを含むというように合理的でかつ説明的です．表現を簡単にするためにライブラリは二つのタイプに分けられます．

一般的ライブラリ　これらは一般的なクラスや表 3.7 に記載したような関連機能を備えています．

モデルライブラリ　これらは表 3.8，表 3.9，表 3.10 に記載した計算連続体力学で使われるモデルを定めます．

基本的な OpenFOAM ツールのライブラリ — OpenFOAM

algorithms	アルゴリズム
containers	コンテナクラス
db	データベースクラス
dimensionedTypes	dimensioned<Type> クラスと派生クラス
dimensionSet	dimensionSet クラス
fields	領域クラス
global	グローバルな設定
graph	graph クラス
interpolations	補間スキーム
matrices	行列クラス
memory	メモリ管理ツール
meshes	メッシュクラス
primitives	根本クラス

有限体積法ライブラリ — finiteVolume

cfdTools	CFD ツール
fields	ボリューム，サーフェス，そしてパッチのフィールドのクラス．境界条件も含む
finiteVolume	有限体積法による離散化
fvMatrices	有限体積法解析のための行列
fvMesh	有限体積法による離散化のためのメッシュ
interpolation	フィールドの補間とマッピング
surfaceMesh	有限体積法による離散化のためのメッシュのサーフェスデータ
volMesh	有限体積法による離散化のためのメッシュのボリューム（セル）データ

後処理ライブラリ

cloudFunctionObjects	ラグランジアン雲の情報をファイルに書き出す関数オブジェクト
fieldFunctionObjects	平均・最大・最小などを含むフィールド関数オブジェクト
foamCalcFunctions	foamCalc ユーティリティのための関数
forces	関数オブジェクトによる，力・揚力・抗力の後処理ツール
FVFunctionObjects	関数オブジェクトを用いて fvcDiv や fvcGrad などを計算するツール
jobControl	関数オブジェクトが使われている実行中のジョブを制御するツール
postCalc	後処理工程で関数オブジェクトの機能を利用するためのもの
sampling	領域における特定の場所での場のデータの抽出用ツール
systemCall	ケースの実行時にシステム・コールを行うための一般的な関数オブジェクト
utilityFunctionObjects	ユーティリティの関数オブジェクト

解法とメッシュ操作のライブラリ

autoMesh	snappyHexMesh ユーティリティの機能のためのライブラリ
blockMesh	blockMesh ユーティリティの機能のためのライブラリ
dynamicMesh	移動メッシュをもつシステムの解法
dynamicFvMesh	移動とトポロジ変化を伴う有限体積メッシュのためのライブラリ
edgeMesh	辺ベースのメッシュ記法の処理用
fvMotionSolvers	有限体積メッシュの移動のソルバ
ODE	常微分方程式のソルバ
meshTools	OpenFOAM メッシュ操作のためのツール
surfMesh	様々な書式のサーフェス・メッシュを扱うためのライブラリ
triSurface	標準的な三角形分割された面ベースのメッシュ記法の処理用
topoChangeFvMesh	トポロジ変化の機能（大部分は冗長）

Lagrange 型粒子追跡ライブラリ

coalCombustion	炭塵燃焼のモデリング
distributionModels	粒子分布関数のモデリング
dsmc	直接シミュレーション・モンテ・カルロ法のモデリング
lagrangian	基本 Lagrange 型もしくは粒子追跡解スキーム
lagrangianIntermediate	粒子追跡の動力学，熱力学，多種粒子反応，粒子力など
potential	分子動力学のための分子間ポテンシャル
molecule	分子動力学のための分子クラス
molecularMeasurements	分子動力学における測定を実行するためのもの
solidParticle	個体粒子の実装
spray	噴霧・噴射のモデリング
turbulence	乱流に基づく粒子の分散とブラウン運動

さまざまなライブラリ

conversion	メッシュとデータの変換のためのツール
decompositionMethods	領域分割のためのツール
engine	エンジンの計算のためのツール

fileFormats	いくつかのサードパーティフォーマットデータの読み込み・書き込みのためのコア・ルーチン
genericFvPatchField	一般的なパッチフィールド
MGridGenGAMGAgglomeration	`MGridGen` アルゴリズムを用いたセルの凝集のためのライブラリ
pairPatchAgglomeration	基礎的なペアのパッチのグルーピング手法
OSspecific	オペレーティング・システム固有の機能
randomProcesses	分析と生成のランダムプロセスのツール

並列ライブラリ

decompose	一般的なメッシュ・フィールド分解のライブラリ
distributed	分散した面の探索と入出力のツール
metisDecomp	Metis 領域分解のライブラリ
reconstruct	メッシュ・フィールドの再構築のライブラリ
scotchDecomp	Scotch 領域分割のライブラリ
ptscotchDecomp	PTScotch 領域分割のライブラリ

表 3.7　一般的使用のための共有オブジェクトライブラリ

基本熱物理モデル — basicThermophysicalModels

hePsiThermo	圧縮率 ψ に基づく一般熱物理モデル計算
heRhoThermo	密度 ρ に基づく一般熱物理モデル計算
pureMixture	不活性混合気体の一般熱物理モデル計算

化学反応モデル — reactionThermophysicalModels

psiReactionThermo	ψ に基づいて燃焼混合気のエンタルピを計算する
psiuReactionThermo	ψ_u に基づいて燃焼混合気のエンタルピを計算する
rhoReactionThermo	ρ に基づいて燃焼混合気のエンタルピを計算する
heheupsiReactionThermo	未燃ガスおよび燃焼混合気のエンタルピを計算する
homogeneousMixture	規格化燃料質量分率 b に基づく混合気燃焼
inhomogeneousMixture	b と総燃料質量分率 f_t に基づく混合気燃焼
veryInhomogeneousMixture	b，f_t と不燃燃料質量分率 f_u に基づく混合気燃焼
basicMultiComponentMixture	複数の要素に基づく基本的な混合気
multiComponentMixture	複数の要素に基づく派生混合気
reactingMixture	熱力学と反応スキームによる燃焼混合気
egrMixture	排気再循環の混合気
singleStepReactingMixture	素反応を伴う混合気

輻射モデル — radiationModels

P1	P1 モデル
fvDOM	有限体積離散座標法
opaqueSolid	不透明な固体の輻射．エネルギ式のソース項には何も影響しない（ゼロを返す）が，absorptionEmissionModel および scatterModel を作る．
viewFactor	形態係数の輻射モデル

層流火炎速度モデル — laminarFlameSpeedModels

constLaminarFlameSpeed	一定の層流火炎速度
GuldersLaminarFlameSpeed	Gulder の層流火炎速度モデル
GuldersEGRLaminarFlameSpeed	排気再循環モデルを伴う Gulder の層流火炎速度モデル
RaviPetersen	Ravi と Petersen の相互関係から層流火炎速度を得る．

バロトロピック圧縮性モデル — barotropicCompressibilityModels

linear	線形圧縮性モデル
Chung	Chung の圧縮性モデル
Wallis	Wallis の圧縮性モデル

ガス種の熱物理特性 — specie	
adiabaticPerfectFluid	断熱完全気体の状態方程式
icoPolynomial	液体などの非圧縮性流体に対する多項式の状態方程式
perfectFluid	完全気体の状態方程式
incompressiblePerfectGas	一定の参照圧力を用いた非圧縮性気体の状態方程式．密度は温度と組成によってのみ変化する．
rhoConst	密度を一定とした状態方程式
eConstThermo	内部エネルギ e とエントロピ s に関する一定比熱 c_p モデル
hConstThermo	エンタルピ h とエントロピ s に関する一定比熱 c_p モデル
hPolynomialThermo	h と s を評価する多項式の係数の関数により c_p が評価される
janafThermo	h や s のような JANAF 熱力学テーブルの係数をもつ関数によって評価した c_p
specieThermo	c_p，h そして/または s から派生するような熱物理特性
constTransport	一定の輸送特性
polynomialTransport	多項式に基づく温度依存輸送特性
sutherlandTransport	温度依存輸送特性のための Sutherland の公式

熱物理特性の関数/表 — thermophysicalFunctions	
NSRDSfunctions	標準参照データシステム (NSRDS) - 米国化学工学会 (AICHE) のデータ編集表
APIfunctions	蒸気拡散のための米国石油協会 (API) の関数

化学モデル — chemistryModel	
chemistryModel	化学反応モデル
chemistrySolver	化学反応ソルバ

その他のライブラリ	
liquidProperties	液体の熱物性
liquidMixtureProperties	混合液体の熱物性
basicSolidThermo	固体の熱物理モデル
hExponentialThermo	equationOfState 用にテンプレート化された指数関数物性の熱力学パッケージ
SLGThermo	固体・液体・気体の熱力学モデルのパッケージ
solidChemistryModel	熱分解を含む固体化学の熱力学モデル
solidProperties	固体の熱物性
solidMixtureProperties	混合固体の熱物性
solidSpecie	固体の反応速度と輸送のモデル
solidThermo	固体エネルギのモデリング

表 3.8 熱物理モデルのライブラリ

非圧縮性流れ用 RAS 乱流モデル — incompressibleRASModels	
laminar	層流用ダミー乱流モデル
kEpsilon	標準の高 Re k–ε モデル
kOmega	標準の高 Re k–ω モデル
kOmegaSST	k–ω–SST モデル
RNGkEpsilon	RNG k–ε モデル
NonlinearKEShih	非線形 Shih k–ε モデル
LienCubicKE	Lien cubic k–ε モデル
qZeta	q–ζ モデル
kkLOmega	非圧縮性流れ用の低レイノルズ数 k–k_l–ω 乱流モデル
LaunderSharmaKE	Launder–Sharma 低 Re k–ε モデル
LamBremhorstKE	Lam–Bremhorst 低 Re k–ε モデル

LienCubicKELowRe	Lien cubic 低 Re k–ε モデル
LienLeschzinerLowRe	Lien–Leschziner 低 Re k–ε モデル
LRR	Launder–Reece–Rodi RSTM
LaunderGibsonRSTM	壁反射項付き Launder–Gibson RSTM
realizableKE	Realizable k–ε モデル
SpalartAllmaras	Spalart–Allmaras 1 方程式混合距離モデル
v2f	Lien と Kalitzin の非圧縮性流れ用 v_2–f 乱流モデル

圧縮性流れ用 RAS 乱流モデル — compressibleRASModels

laminar	層流用のダミー乱流モデル
kEpsilon	標準 k–ε モデル
kOmegaSST	k–ω–SST モデル
RNGkEpsilon	RNG k–ε モデル
LaunderSharmaKE	Launder–Sharma 低 Re k–ε モデル
LRR	Launder–Reece–Rodi RSTM
LaunderGibsonRSTM	Launder–Gibson RSTM
realizableKE	Realizable k–ε モデル
SpalartAllmaras	Spalart–Allmaras 1 方程式混合距離モデル
v2f	Lien と Kalitzin の圧縮性流れ用 v_2–f 乱流モデル

Large-eddy シミュレーション (LES) フィルタ — LESfilters

laplaceFilter	Laplace フィルタ
simpleFilter	単純フィルタ
anisotropicFilter	異方性フィルタ

Large-eddy シミュレーションのデルタ — LESdeltas

PrandtlDelta	Prandtl のデルタ
cubeRootVolDelta	セル体積の立方根のデルタ
maxDeltaxyz	x，y，z の最大値，6 面体セルの構造格子に対してのみ
smoothDelta	スムージングのデルタ

非圧縮 LES モデル — incompressibleLESmodels

Smagorinsky	Smagorinsky モデル
Smagorinsky2	3 次元フィルタ付き Smagorinsky モデル
homogenousDynSmagorinsky	同次ダイナミック Smagorinsky モデル
dynLagrangian	Lagrange 型 2 方程式渦粘性モデル
scaleSimilarity	スケール相似モデル
mixedSmagorinsky	Smagorinsky とスケール相似の混合モデル
homogenousDynOneEqEddy	非圧縮性流れ用の 1 方程式渦粘性モデル
laminar	単純に層流の物理量を返す
kOmegaSSTSAS	k–ω–SST スケール適応シミュレーション (SAS) モデル
oneEqEddy	k 方程式渦粘性モデル
dynOneEqEddy	ダイナミック k 方程式渦粘性モデル
spectEddyVisc	スペクトル渦粘性モデル
LRDDiffStress	LRR 差分応力モデル
DeardorffDiffStress	Deardorff 差分応力モデル
SpalartAllmaras	Spalart–Allmaras モデル
SpalartAllmarasDDES	Spalart–Allmaras 遅延型分離渦シミュレーション (DDES) モデル
SpalartAllmarasIDDES	Spalart–Allmaras 改良 DDES モデル
vanDriestDelta	非圧縮 LES モデルで使われる Δ としてセル体積の単純な立方根が使われる．

圧縮性 LES モデル — compressibleLESmodels

Smagorinsky	Smagorinsky モデル
oneEqEddy	k 方程式渦粘性モデル
lowReOneEqEddy	低 Re k 方程式渦粘性モデル
homogenousDynOneEqEddy	非圧縮性流れ用の 1 方程式渦粘性モデル

DeardorffDiffStress	Deardorff 差分応力モデル
SpalartAllmaras	Spalart–Allmaras 1 方程式混合距離モデル
vanDriestDelta	圧縮 LES モデルで使われる Δ としてセル体積の単純な立方根が使われる.

表 3.9 乱流モデルと LES モデルのライブラリ

非圧縮性流れ用輸送モデル — incompressibleTransportModels	
Newtonian	線形粘性流れモデル
CrossPowerLaw	Cross Power 則非線形粘性モデル
BirdCarreau	Bird–Carreau 非線形粘性モデル
HerschelBulkley	Herschel–Bulkley 非線形粘性モデル
powerLaw	べき乗則非線形粘性モデル
interfaceProperties	多相流解析における接触角のようなインタフェースのモデル

その他の輸送モデルライブラリ	
interfaceProperties	界面の物性値の計算
twoPhaseProperties	2 相の物性値モデル，境界条件も含む.
surfaceFilmModels	表面フィルムモデル

表 3.10 移送モデルの共有オブジェクトライブラリ

第4章

OpenFOAMのケース

本章では，OpenFOAMのケースのファイル構造について説明します．通常，ユーザはケースに名前を割り当てます (例えばチュートリアルのキャビティ流れのケースは単純にcavityと名付けられています)．この名前は，すべてのケースファイルとサブディレクトリが収納されているディレクトリの名前になります．このケースディレクトリ自体はどこにでも置くことができますが，第2章の冒頭で述べたように，*$HOME/OpenFOAM/${USER}-2.3.0*のように，ユーザのプロジェクトのサブディレクトリ，*run*の中に置くことを推奨します．この利点の一つは，$FOAM_RUNの環境変数がデフォルトで*$HOME/OpenFOAM/${USER}-2.3.0/run*に設定されていることです．コマンドラインでプリセットエイリアス，*run*を実行することにより，素早くこのディレクトリに移動することができます．OpenFOAMをダウンロードする際に添付されているチュートリアルのケースは，ケースのディレクトリ構造の有用な例を提供しています．チュートリアルは*$FOAM_TUTORIALS*のディレクトリに置かれており，コマンドラインで`tut`エイリアスを実行することにより素早くたどり着けます．この章を読みながら，チュートリアルの例を参照してください．

4.1 OpenFOAMのケースのファイル構造

アプリケーションを実行するために必要な最小限のファイルを含む，OpenFOAMケースの基本的なディレクトリ構造を図4.1に示し，以下で説明します．

constantディレクトリ　そのケースのメッシュに関する全ての情報を含むサブディレクトリ*polyMesh*，および使おうとしているアプリケーションのための物性値を定めるファイル (例えば*transportProperties*) が格納されています．

systemディレクトリ　解析の手順に関連するパラメータ設定のためのディレクトリです．少なくとも以下の三つのファイルを含みます．パラメータが開始/終了時間や時間ステップおよびデータのアウトプットのためのパラメータの設定を行う*controlDict*，実行時に選択される解析に使われるスキームを記述している*fvSchemes*，そして実行のために方程式のソルバ，許容誤差およびその他のアルゴリズム制御を設定する*fvSolution*です．

時刻ディレクトリ　特定のフィールドに関するデータの個別のファイルを含んでいます．データは，問題を定義するためにユーザが指定する初期値と境界条件，または書き込まれたOpenFOAMのファイルの結果が存在します．OpenFOAMのフィールドは，定常状態の問題のように厳密に解く必要のない場合であっても，常に初期化する必要があることに留意してください．各時刻ディレクトリの名称は，データが書き込まれた時点のシミュレーションが行われた時刻に基づいており，詳細については4.3節に記述されています．

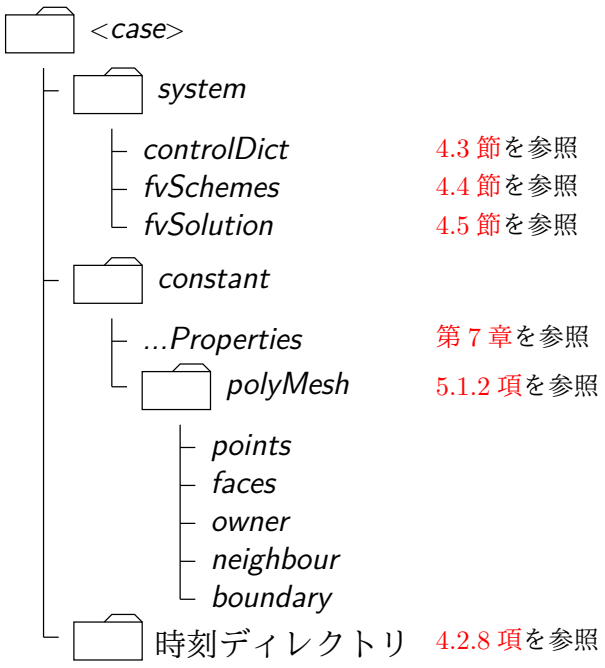


図 4.1　ケースディレクトリの構造

私達は通常時間 $t = 0$ でシミュレーションを始めて，初期条件は指定された名前のフォーマットに応じて *0* または *0.000000e+00* と名付けられたディレクトリの中に通常収納されるため，十分といえます．例えば，cavity のチュートリアルで，速度場の $\boldsymbol{U}$ と圧力場の p それぞれファイル *0/U* と *0/p* から初期化されます．

4.2　基本的な入出力ファイルのフォーマット

OpenFOAM は，文字列，スカラ，ベクトル，テンソル，リスト，およびフィールド等のデータ構造の範囲を読み込む必要があります．入出力 (I/O) ファイルのフォーマットはユーザが OpenFOAM のアプリケーションをできる限り容易に修正できるよう，非常に柔軟に設計されています．この I/O は，ファイルの作成が非常に簡単で理解しやすい単純なルールに従っているものであり，ファイルのフォーマットが直観的に理解しづらいばかりかどこにも公開されていないような，多くのソフトウェアパッケージとは対照的です．OpenFOAM のファイルフォーマットについては次節で説明します．

4.2.1　一般的な構文規則

フォーマットは以下の C++ ソースコードのいくつかの一般的な原理に従います．

- ファイルは，列によって特定の意味が割り当てられることもなく，継続行を明示する必要もない，自由形式となっています．
- 行は特に意味をもちませんが，コメント・デリミタ `//` があれば OpenFOAM は行の最後までテキストを無視します．
- 複数行にわたるコメントは，`/*` と `*/` で囲みます．

4.2.2 ディクショナリ

OpenFOAM においてデータを指定する最も一般的な手段としては**ディクショナリ**を使います．ディクショナリには，**キーワード**に応じて I/O から読み出すことのできるデータ項目が含まれています．キーワード・エントリは以下のような一般的な書式に従います．

```
<keyword>  <dataEntry1> ... <dataEntryN>;
```

ほとんどの入力項目は単一のデータ入力の書式になっています

```
<keyword>  <dataEntry>;
```

ほとんどの OpenFOAM のデータファイルはそれ自体 1 セットのキーワード入力を含むディクショナリです．ディクショナリは論理的なカテゴリにエントリを構成するための手段を提供しており，階層的に指定できるので，どんなディクショナリもそれ自体が一つ以上のディクショナリエントリを含んでいます．ディクショナリの書式は，以下のようにディクショナリ名を指定し，その後に波括弧 `{ }` で囲まれたキーワード・エントリが続きます．

```
<dictionaryName>
{
    ... keyword entries ...
}
```

4.2.3 データファイルヘッダ

OpenFOAM によって読み書きされるすべてのデータファイルは，表 4.1 に記載されており，キーワード入力の標準セットを含む `FoamFile` と名付けられたディクショナリから始まります．

キーワード	説明	入力
`version`	入出力形式のバージョン	2.0
`format`	データ形式	`ascii` / `binary`
`location`	"..." ファイルへのパス	（オプション）
`class`	関連するデータファイルから構成された OpenFOAM のクラス	一般的に `dictionary` もしくは `volVectorField` などのフィールド
`object`	ファイル名	例：*controlDict*

表 4.1 データファイルのためのヘッダのキーワード入力

この表には各エントリの簡単な説明を載せています．これは `class` については多くの例外があるものの，おそらくほとんどのエントリについては十分な内容でしょう．`class` エントリはファイル内のデータから構成される OpenFOAM ライブラリでの C++ クラスの名前です．おそらくユーザは，読み込まれるファイルを呼び出す基礎的なコードの知識や OpenFOAM クラスの知識なしに，`class` の入力を正確に推測することはできません．しかし，ほとんどのデータファイルは単純なキーワードエントリをもち内部の dictionary クラスの中に読み込まれます．それゆえ，それらの場合では `class` エントリは `dictionary` となります．

以下の例はこれまで説明してきたエントリのタイプを使ったケースへのデータ供給のキーワードの使い方を示しています．*fvSolution* ディクショナリファイルを分解すると，*solvers* と *PISO* という二つのディクショナリが含まれています．*solvers* ディクショナリは圧力方程式と速度方

程式に対してそれぞれ計算用と収束用に複数のデータ入力があり，それぞれ p と U のキーワードによって記述されます．*PISO* ディクショナリはアルゴリズムの制御パラメータを含みます．

```
17
18  solvers
19  {
20      p
21      {
22          solver           PCG;
23          preconditioner   DIC;
24          tolerance        1e-06;
25          relTol           0;
26      }
27
28      U
29      {
30          solver           smoothSolver;
31          smoother         symGaussSeidel;
32          tolerance        1e-05;
33          relTol           0;
34      }
35  }
36
37  PISO
38  {
39      nCorrectors     2;
40      nNonOrthogonalCorrectors 0;
41      pRefCell        0;
42      pRefValue       0;
43  }
44
45
46  // ************************************************************************* //
```

4.2.4 リスト

OpenFOAM アプリケーションはリストを含んでいます．例えば，メッシュ記述のための頂点リストがあります．リストは一般的に I/O にあり独自のフォーマットをもっていて，入力は丸括弧（ ）内にされます．また，丸括弧の前のフォーマットの選択もあります．

simple キーワードに続いてすぐに丸括弧がくる．

```
<listName>
(
    ... entries ...
);
```

numbered キーワードに続いてリスト内の要素数 <n> がくる．

```
<listName>
<n>
(
    ... entries ...
);
```

token identifier キーワードに続いてクラス名の識別子ラベル <Type> がくる．<Type> はリストに何が入っているかを記載したもので，例えばスカラ要素のリストであれば次のようになる．

```
<listName>
List<scalar>
<n>        // optional
(
    ... entries ...
);
```

ここで留意すべきはリスト <scalar> での <scalar> は一般名ではなく入力された実際の文字列です．単純なフォーマットは，リストを書くときの便利な方法です．その他のフォーマットはリストのサイズがデータを読み込む前にメモリに割り当てられるのでコードがより早くデータを読み込めます．それゆえ単純なフォーマットは読み込み時間が最小の短いリストに適しており，その他のフォーマットは長いリストに適しています．

4.2.5 スカラとベクトル，テンソル

スカラは，データファイルでは一つの数字として記述されます．vector は，ランク 1 で 3 次元の VectorSpace であり，要素数はいつも 3 に決まっているので単純なリストフォーマットで使われます．それゆえ，ベクトル $(1.0,\ 1.1,\ 1.2)$ は次のように書かれます．

```
(1.0 1.1 1.2)
```

OpenFOAM では，テンソルはランク 2 で 3 次元の VectorSpace であり，それゆえデータ入力はいつも九つの実数と決まっています．それゆえ単位テンソルは以下のように書かれます．

```
(
    1 0 0
    0 1 0
    0 0 1
)
```

この例は入力が複数の行に上書きできるように OpenFOAM がその行に戻るのを無視する方法を示しています．一行に数字を羅列することと扱いは違いません．

```
( 1 0 0 0 1 0 0 0 1 )
```

4.2.6 次元の単位

連続体力学では，物理量はある決められた単位で表現されます．例えば，質量ならキログラム (kg)，体積なら立法メートル (m^3)，圧力ならパスカル $(\mathrm{kg\,m^{-1}\,s^{-2}})$ というように．代数の演算は統一した測量単位を用いて実行されなければなりません．特に，足し算，引き算，および等式は同じ次元の単位の物理的特性においてのみ意味があります．無意味な操作を実行することへの安全装置として，OpenFOAM はフィールドデータと物理的特性に次元を付けて，どのようなテンソル操作についても次元をチェックして実行します．dimensionSet の入出力形式は角括弧内の七つのスカラ量です．例えば，

```
[0 2 -1 0 0 0 0]
```

表 4.2 に記載するように各値は計測基準単位のそれぞれの物理量に対応しています．表は国際単

No.	物理量	SI 単位	USCS 単位
1	質量	キログラム (kg)	質量ポンド (lbm)
2	長さ	メートル (m)	フィート (ft)
3	時間	秒 (s)	
4	温度	ケルビン (K)	ランキン温度 (°R)
5	物質量	モル (mol)*	ポンドモル (lbmol)
6	電流	アンペア (A)	
7	光度	カンデラ (cd)	

表 4.2 SI と USCS の基本単位

位系 (SI) と the United States Customary System (USCS) の基本単位ですが OpenFOAM はどの単位系も使えます．要求されることは入力データが選択した単位に合っているということです．特に重要なのは，OpenFOAM がいくつかの次元化された物理定数を必要とするということを知っておくことです．例えば熱力学のモデル化したある計算のための一般気体定数 R などがいい例です．これらの次元定数は OpenFOAM インストール (*$WM_PROJECT_DIR/etc/controlDict*) のメイン *controlDict* ファイルの *DimensionedConstant* サブディクショナリで指定されます．デフォルトでは，これらの定数は SI で設定されます．USCS もしくはその他の単位系を使用したい場合は，選択した単位系に合わせてこれらの定数を変更してください．

4.2.7 次元付きの型

物理量は一般に，それらの関連する次元によって特定されます．これらの入力は，`dimensionedScalar` の以下の例が示すフォーマットをもっています．

```
nu              nu  [0 2 -1 0 0 0 0]  1;
```

最初の `nu` はキーワード，2 番目の `nu` はクラスの word の名前で，通常キーワードと同じものが選ばれる．その次の入力は dimensionSet で最終的な入力はスカラ値です．

4.2.8 フィールド

OpenFOAM の入出力データの多くはテンソル場，例えば速度や圧力のデータにあり，時刻ディレクトリから読み込み時刻ディレクトリに書き込まれます．表 4.3 で説明されるように，キーワード入力を使って，OpenFOAM はフィールドデータを書きこみます．

キーワード	説明	例
`dimensions`	領域の次元	`[1 1 -2 0 0 0 0]`
`internalField`	内部領域の値	`uniform (1 0 0)`
`boundaryField`	境界領域	4.2.8 項のファイル参照

表 4.3 フィールドディクショナリで使われる主なキーワード

データは `dimensions` エントリから始まります．その後に続くのは，以下のいずれかの方法で記述される `internalField` です．

一様フィールド ただひとつの値にそのフィールド内で全ての要素が対応していて，以下のような書式をとります．

*訳注：原文では kgmol とされているが，これは誤り．SI における物質量の基本単位は mol である．

```
internalField uniform <entry>;
```

非一様フィールド 各フィールドの要素は，固有の値を割り当てられ，リストの識別子トークンフォームにある以下の書式をとることが推奨されます．

```
internalField nonuniform <List>;
```

boundaryField は *polyMesh* ディレクトリ内の *boundary* ファイルにある境界パッチのそれぞれの名前に対応する名前の一連の入力を含んだディクショナリである．各パッチの入力はそれ自体がキーワード入力のリストを含むディクショナリとなります．必須エントリである `type` には，そのフィールドに指定すべきパッチ・フィールド条件を記述します．残りの入力は選択されたパッチ・フィールド条件のタイプに対応し，一般的にはパッチフェイスで初期条件を分類するフィールドデータを含みます．OpenFOAM で使えるパッチ・フィールド条件の選択肢は，その説明と指定しなければならないデータと併せて，表 5.3 と表 5.4 に記載してあります．速度 `U` のフィールドのディクショナリ入力の例を以下に示します．

```
17 dimensions      [0 1 -1 0 0 0 0];
18
19 internalField   uniform (0 0 0);
20
21 boundaryField
22 {
23     movingWall
24     {
25         type            fixedValue;
26         value           uniform (1 0 0);
27     }
28
29     fixedWalls
30     {
31         type            fixedValue;
32         value           uniform (0 0 0);
33     }
34
35     frontAndBack
36     {
37         type            empty;
38     }
39 }
40
41 // ************************************************************************* //
```

4.2.9 ディレクティブとマクロ置換

OpenFOAM のケースファイルを柔軟に設定するためのディレクティブや代替マクロといったオプションのファイル構文があります．ディレクティブはケースファイル内で `#` から始まるコマンドとして記述されます．代替マクロは `$` から始まります．

OpenFOAM では現在 4 種類のディレクティブが利用可能できます．

`#include "<fileName>"` または **`#includeIfPresent "<fileName>"`** *<fileName>*という名前のファイルを読み込む

`#inputMode` 二つのオプションがある．`merge` は連続するディクショナリのキーワードのエン

トリを統合する．つまりある場所で指定されたキーワードのエントリを継承して以後の同一キーワードのエントリが指定される．overwrite はディクショナリ全体を上書きする．通常は merge を使う．

#remove <keywordEntry> インクルードされた全てのキーワードエントリを削除する．単語または正規表現で指定できる．

#codeStream 続けて C++ ソースコードを書くと，そのコードを即席でコンパイル・ロード・実行し，エントリを生成する．

4.2.10 #include および #inputMode ディレクティブ

一度使われた圧力の初期値を，内部フィールドと境界の初期値に設定する例を示します．以下の記述を含む *initialConditions* というファイルを作成していたとします．

```
pressure 1e+05;
#inputMode merge
```

この圧力をフィールド内部と境界に用いるために，以下の代替マクロを圧力場のファイル *p* に記述します．

```
#include "initialConditions"
internalField uniform $pressure;
boundaryField
{
    patch1
    {
        type fixedValue;
        value $internalField;
    }
}
```

あくまでもこれは，この機能がどのように働くかを示しただけの単純な例です．しかしこの機能は，ケースデータをユーザのニーズに合わせて一般化する手段として，より便利な使い方で多く用いることができます．例えば同一の RAS 乱流モデルの設定を用いるケースがいくつかある場合，その設定を記述したファイルを一つ作成し，各ケースの *RSAProperties* ファイルに include によって組み込めばよいのです．代替マクロは単独の値にとどまりません．例えば，単独のマクロで境界条件のまとまりを事前に定義して，それをよびだすことができます．この機能はほぼどこでも使えます．

4.2.11 #codeStream ディレクティブ

#codeStream ディレクティブは C++ コードをコンパイル・実行して，ディクショナリのエントリを生成します．コードとコンパイル方法は以下のキーワードで指定します．

- code: コードを指定します．これは OStream& os および dictionary& dict を引数とし，ユーザはコードの中でこれらの引数を使うことができます．例えば，当該ケースのディクショナリ（ファイル）からキーワード・エントリを取り出すことができます．
- codeInclude(オプション): OpenFOAM ファイルを読み込むため，追加の C++ #include 文を指定できます．

- codeOptions（オプション）：*Make/options* の中の EXE_INC に加えて，追加のコンパイル・フラグを指定できます．
- codeLibs（オプション）：*Make/options* の中の LIB_LIBS に加えて，追加のコンパイル・フラグを指定できます．

コードは，ハッシュ・ブラケット記号，すなわち #{...#} で囲むことで，通常の文字列と同じように複数行にわたって書くことができます．この二つの記号の間のあらゆるものは，全ての改行・引用符などの予約語とともに，一つの文字列となります．

以下に #codeStream の例を示します．このコードは *controlDict* ファイル内に書かれており，ディクショナリ・エントリを取り出し，出力間隔を決めるための簡単な計算を施しています．

```
startTime       0;
endTime         100;
...
writeInterval   #codeStream
{
    code
    #{
        scalar start = readScalar(dict.lookup("startTime"));
        scalar end = readScalar(dict.lookup("endTime"));
        label nDumps = 5;
        os << ((end - start)/nDumps);
    #};
};
```

4.3 時間とデータの入出力制御

OpenFOAM のソルバは全て，データベースをセットアップすることによって，動き始めます．データベースは入出力を制御し，またデータの出力は通常実行中，時間ごとに要求されるので，時間はデータベースにとって不可避の要素です．*controlDict* ディクショナリはデータベースの作成に不可欠な入力パラメータを設定します．*controlDict* のキーワード入力項目は表 4.4 に記載されています．時間制御方式と writeInterval 入力だけは完全に強制的で，省略できる任意の項目には表 4.4 で示されたデフォルト値のデータベースが用いられます．

時間制御	
startFrom	解析の開始時刻の制御
- firstTime	存在する時刻ディレクトリのうちで最初の時刻
- startTime	startTime の項目の入力により定める時刻
- latestTime	存在する時刻ディレクトリのうちで最近の時刻
startTime	startFrom の startTime を用いた解析の開始時刻
stopAt	解析の終了時刻の制御
- endTime	endTime の項目の入力により定める時刻
- writeNow	現在の時間ステップで解析を止めデータを書き出す
- noWriteNow	現在の時間ステップで解析を止めデータは書き出さない
- nextWrite	writeControl で指定した次のデータ書き出しの時間ステップで解析を止める
endTime	stopAt の endTime で指定した解析の終了時刻
deltaT	解析の時間ステップ

データの書き出し	
writeControl	ファイルへのデータの書き出しのタイミングの制御
- timeStep†	時間ステップの writeInterval ごとにデータを書き出す

- runTime	解析時間 writeInterval 秒ごとにデータを書き出す
- adjustableRunTime	解析時間 writeInterval 秒ごとにデータを書き出す，必要なら時間ステップを writeInterval と一致するように調整する（自動時間ステップ調整を行う場合に使用する）.
- cpuTime	CPU 時間 writeInterval 秒ごとにデータを書き出す
- clockTime	実時間 writeInterval 秒ごとにデータを書き出す
writeInterval	上記の writeControl と関連して用いられるスカラ
purgeWrite	周期的に時刻ディレクトリを上書きすることによって保存される時刻ディレクトリの数の限界を表す整数．たとえば $t_0 = 5\,\mathrm{s}$, $\Delta t = 1\,\mathrm{s}$, purgeWrite 2; のとき，*6* と *7*，二つのディレクトリにデータが書き込まれた後，8 s のデータが *6* に上書きされ，9 s のデータが *7* に上書きされる．時間ディレクトリ限界を無効にするには，purgeWrite 0; とする．†定常状態解析では，以前の反復計算の結果を purgeWrite 1; とすることで連続して上書きできる.
writeFormat	データファイルのフォーマットを指定する
- ascii†	ASCII フォーマット，writePrecision の有効桁まで書かれる
- binary	バイナリー・フォーマット
writePrecision	上記の writeFormat に関連して使用される整数，デフォルトでは 6†.
writeCompression	データファイルの圧縮を指定する
- uncompressed	非圧縮†
- compressed	gzip 圧縮
timeFormat	時刻ディレクトリのネーミングのフォーマットの選択
- fixed	$\pm m.dddddd$ の d の数が timePrecision で決められる
- scientific	$\pm m.dddddde\pm xx$ の d の数が timePrecision で決められる
- general†	指数が -4 未満もしくは timePrecision で指定された指数以上のとき scientific のフォーマットを指定する
timePrecision	上記の timeFormat に関連して使用される整数，デフォルトでは 6†
graphFormat	アプリケーションによって描かれるグラフデータのフォーマット
- raw†	横書きの生の ASCII 形式
- gnuplot	gnuplot 形式のデータ
- xmgr	Grace/xmgr 形式のデータ
- jplot	jPlot 形式のデータ
可変時間ステップ	
adjustTimeStep	時間ステップをシミュレーション実行中に調整するかどうかを決める yes†/no スイッチ．通常は以下に従う.
maxCo	Courant 数の最大値．例えば 0.5
データの読み込み	
runTimeModifiable	*controlDict* などのディクショナリを各時間ステップの始めに再読み込みするかどうかを決める yes†/no スイッチ
実行時にロード可能な機能	
libs	実行時にロードする (*$LD_LIBRARY_PATH* 上の) 追加ライブラリのリスト. 例えば ("libUser1.so" "libUser2.so")
functions	関数のリスト．実行時に例えば probes をロードする．*$FOAM_TUTORIALS* の例を参照.

† 関連キーワードが省略されるなら，デフォルト入力を表示します.

表 4.4 controlDict ディクショナリのキーワード項目

以下に *controlDict* ディクショナリの入力例を示します.

```
17
18  application     icoFoam;
```

```
19
20 startFrom       startTime;
21
22 startTime       0;
23
24 stopAt          endTime;
25
26 endTime         0.5;
27
28 deltaT          0.005;
29
30 writeControl    timeStep;
31
32 writeInterval   20;
33
34 purgeWrite      0;
35
36 writeFormat     ascii;
37
38 writePrecision  6;
39
40 writeCompression off;
41
42 timeFormat      general;
43
44 timePrecision   6;
45
46 runTimeModifiable true;
47
48
49 // ************************************************************************* //
```

4.4 数値スキーム

system ディレクトリにある *fvSchemes* ディクショナリは，アプリケーションの実行時に現われる，方程式における導関数等の項に対する数値スキームを設定します．この節では，*fvSchemes* ディクショナリにおいてどのように，これらのスキームを指定するかを説明します．

fvSchemes において数値スキームを割りあてなければならない典型的な項は，例えば空間勾配といった導関数項や，一つの点集合から他の集合へと値を補間する項等です．OpenFOAM では，ユーザに制限無くスキームを選択できるようにしたいと思っています．例えば，線形補間は多くのケースで効率的ですが，OpenFOAM では，全ての補間項に対して幅広い補間スキームの中から自由に選択ができるようになっています．

導関数の項は，このような選択の自由のさらなる好例となります．ユーザは，まず離散化手法を選択することができますが，ここでは Gauss による有限体積積分を用いるのが一般的です．Gauss 積分は格子の界面における値を足していくことで実現されますが，界面での値は格子中心での値から補間しなければなりません．この補間スキームにおいてもユーザは自由に選ぶことができ，特定の導関数項，特に対流項に用いる発散項には，特別に設計されたいくつかのスキームが用意されています．数値スキームを指定しなければならない項はいろいろありますが，それらは *fvSchemes* ディクショナリにおいて表 4.5 に示すカテゴリに分類されます．表 4.5 における各キーワードはサブディクショナリの名前ですが，それらは各々特定のタイプの項を持っているわけです．例えば，`gradSchemes` には `grad(p)`（と表現される）といった全ての勾配項があります．その他の例は，以下に示した *fvSchemes* ディクショナリの抜粋をご覧ください．

```
17
```

キーワード	数学的タームのカテゴリ
interpolationSchemes	2 点間の値の補間
snGradSchemes	格子界面の法線方向勾配の各要素
gradSchemes	勾配 ∇
divSchemes	発散 $\nabla \cdot$
laplacianSchemes	Laplacian ∇^2
timeScheme	1 次と 2 次の時間導関数 $\partial/\partial t$，$\partial^2/\partial^2 t$
fluxRequired	フラックスの生成が必要な物理量

表 4.5　fvSchemes で使用する主なキーワード

```
18  ddtSchemes
19  {
20      default         Euler;
21  }
22
23  gradSchemes
24  {
25      default         Gauss linear;
26      grad(p)         Gauss linear;
27  }
28
29  divSchemes
30  {
31      default         none;
32      div(phi,U)      Gauss linear;
33  }
34
35  laplacianSchemes
36  {
37      default         Gauss linear orthogonal;
38  }
39
40  interpolationSchemes
41  {
42      default         linear;
43  }
44
45  snGradSchemes
46  {
47      default         orthogonal;
48  }
49
50  fluxRequired
51  {
52      default         no;
53      p               ;
54  }
55
56
57  // ************************************************************************* //
```

この例を見ると *fvSchemes* ディクショナリは以下の要素から成り立っていることがわかります．

- 六つの*...Schemes* のサブディクショナリには，指定した各項に対するキーワードが書いてあり，default のキーワードも指定できますが，その他にも，例えば ∇p については grad(p) というように，特定の項に対して名前を書くことで，それに対応するキーワードを指定することができます．
- *fluxRequired* のサブディクショナリには，例えば p のように，アプリケーションの中でフラックスが生成される場が書かれています．

もし，default のスキームが特定の*...Schemes* のサブディクショナリで指定された場合には，サブディクショナリが参照している全ての項にそのスキームが適用されます．例えば，*gradSchemes*

において default が指定されている場合には，そのアプリケーションにおける，∇p，$\nabla \boldsymbol{U}$ といった全ての勾配項に対して，その default のスキームが適用されるわけです．default が指定されているときには，そのサブディクショナリにおいて各項のスキームをいちいち指定する必要がなくなります．この例では，grad(p)，grad(U) の行がそれです．しかしながら，特定の項の行が挿入された場合，その項に対しては，指定されたスキームが default より優先されます．

かわりに，ユーザは none エントリにより，あえて default スキームを使わないようにもできます．この場合には，ユーザはそのサブディクショナリの中の全ての項を個々に指定しなければなりません．default は上書きすることができるのですから，default に none を設定することはやりすぎかもしれません．しかしながら，none を指定することは，ユーザが全ての項を個別に指定しなければならないことから，そのアプリケーションに実際にどの項が存在するかを認識するという点では有用です．

次の節では，表 4.5 に示したそれぞれのカテゴリの項について，選択できるスキームを述べます．

4.4.1 補間スキーム

interpolationSchemes サブディクショナリには，通常，セル中心から界面中心へ値を内挿する項があります．OpenFOAM での内挿スキームの選択肢を表 4.6 に示しますが，これは四つのカテゴリに分けられます．一つのカテゴリは一般的なスキームが，そして他の三つのカテゴリは，4.4.5 項で説明するように，主に流体での対流（発散）項の Gauss の離散化と一緒に使われるものです．ユーザが *interpolationSchemes* サブディクショナリにおいて，対流特有のスキームを一般的なフィールドの内挿に適用することは，「ほとんどない」のですが，有効な内挿スキームとして 4.4.5 項よりもむしろここで説明しておきます．なお，UMIST のようなスキームも OpenFOAM では利用可能なことに注意すべきですが，一般的に推奨されるスキームのみを表 4.6 に示します．

普通のスキームは，単にキーワードとエントリのみを記すことで指定でき，例えば linear スキームを default として指定するには以下のようにします．

```
default linear;
```

対流特有のスキームは，流れの速度による流束に基づいて内挿を行います．これらのスキームを指定する場合には，内挿のベースとなる流束場の名前が必要ですが，ほとんどの OpenFOAM のアプリケーションでは，これは phi となっており，この名前は，通常，surfaceScalarField の速度の流束に対応するものです．このガイドの中では，対流特有のスキームの三つのカテゴリは，general convection，normalised variable (NV)，そして，total variation diminishing (TVD) と記述されます．blended スキームを除いて，general convection と TVD スキームは，そのスキーム名と流束場によって指定され，例えば流束 phi に基づく upwind スキームを default として指定するには以下のようにします．

```
default upwind phi;
```

いくつかの TVD/NVD スキームには，$0 \leq \psi \leq 1$ の範囲の係数 ψ が必要ですが，$\psi = 1$ は TVD

条件に従うことに対応し，通常最も良い収束性を示すのに対し，$\psi = 0$ は最も良い精度を与えます．通常 $\psi = 1$ での実行がお勧めです．流束 phi に基づく $\psi = 1.0$ での limitedLinear スキームを，default として指定するには以下のようにします．

```
default limitedLinear 1.0 phi;
```

4.4.1.1　厳密に範囲が限定されるスカラ量に対するスキーム

厳密に範囲が限定される必要のあるスカラ量のために，いくつかの制限付きスキームという拡張版があります．ユーザが指定した範囲に限定するためには，スキームの名前には limited という語が頭に付けられ，下限と上限それぞれを続けて指定します．例えば，vanLeer スキームを -2 と 3 の間で厳密に制限するためには，次のように指定します．

```
default limitedVanLeer -2.0 3.0;
```

よく使われる 0 と 1 の間で限定されるスカラ場のために特化された版もあります．それらは，スキームの名前に 01 を付けることで選択できます．例えば，vanLeer スキームを 0 と 1 の間で厳密に限定するためには，以下のように指定します．

```
default vanLeer01;
```

厳密に範囲が限定する拡張版は，limitedLinear，vanLeer，Gamma，limitedCubic，MUSCL，SuperBee のスキームで利用することができます．

4.4.1.2　ベクトル場に対するスキーム

ベクトル場に対する制限付きスキームについては，場の方向を考慮にいれて構成された改良版のリミッタがあります．これらのスキームは，通常のスキームの名前に V を加えることで選択することができ，limitedLinear に対しては limitedLinearV といった具合です．これら V 版は limitedLinearV，vanLeerV，GammaV，limitedCubicV，SFCDV といったスキームで利用することができます．

4.4.2　表面法線方向勾配スキーム

snGradSchemes サブディクショナリは，表面法線方向勾配の項によるものです．表面法線方向勾配は，格子の界面で計算されますが，それは，界面が接続している二つの格子の中心における値の勾配の，界面の法線方向の成分です．表面法線方向勾配は，それ自体を使うためにも指定されますが，Gauss 積分を使って Laplacian を評価する際にも必要となります．

利用可能なスキームを表 4.7 に示しますが，これらは単にキーワードとエントリを記述することで指定できます．ただ，limited は例外で，$0 \leq \psi \leq 1$ の範囲の係数 ψ を必要とします．

中心スキーム	
linear	線形補間（中心差分）
cubicCorrection	キュービックスキーム
midPoint	均等重み付け線形補間
風上対流スキーム	
upwind	風上差分
linearUpwind	線形風上差分
skewLinear	ひずみ補正付き線形スキーム
filteredLinear2	高周波の雑音のフィルタリングを伴う線形スキーム
TVD スキーム	
limitedLinear	有限線形差分
vanLeer	van Leer リミッタ
MUSCL	MUSCL リミッタ
limitedCubic	キュービックリミッタ
NVD スキーム	
SFCD	自動フィルタ中心差分
Gamma ψ	ガンマ差分

表 4.6 補間スキーム

ここで，

$$\psi = \begin{cases} 0 & \texttt{uncorrected} \text{ に対応}, \\ 0.333 & \text{非直交補正} \leq 0.5 \times \text{直交部分}, \\ 0.5 & \text{非直交補正} \leq \text{直交部分}, \\ 1.0 & \texttt{corrected} \text{ に対応}. \end{cases} \tag{4.1}$$

です．

よって，$\psi = 0.5$ の limited スキームを default として指定するには次のようにします．

```
default limited 0.5;
```

スキーム	説明
corrected	陽的非直交補正
uncorrected	非直交補正なし
limited ψ	有限非直交補正
bounded	ポジティブスカラの有界補正
fourth	4 次

表 4.7 表面法線方向勾配スキーム

4.4.3 勾配スキーム

gradSchemes サブディクショナリには勾配項を記述します．各項の離散化スキームは，表 4.8 の中から選択することができます．

leastSquares と fourth の場合には，離散化スキームの指定は次のようにそのスキーム名を指定するだけで十分です．

```
grad(p) leastSqueares;
```

離散化スキーム	説明
Gauss <interpolationScheme>	2 次の Gauss 積分
leastSquares	2 次の最小二乗法
fourth	4 次の最小二乗法
cellLimited <gradScheme>	上記のスキームのセル制限バージョン
faceLimited <gradScheme>	上記のスキームの面制限バージョン

表 4.8 *gradSchemes* において使用できる離散化スキーム

Gauss キーワードは，Gauss 積分による標準的な有限体積法の離散化を指定するもので，これは，格子の中心から界面の中心への値の内挿を必要とします．このため，Gauss エントリでは，表 4.6 のような内挿スキームを続けて指定する必要があります．一般的な内挿スキーム以外を選択することはほとんどなく，ほとんどのケースでは次のように linear スキームを選ぶのが効率的です．

```
grad(p) Gauss linear;
```

三つの基本的な勾配スキーム (Gauss，leastSquares，fourth) の範囲限定版は，離散化スキームの前に cellLimited (または faceLimited) を付けることで選択できます．例えば，セルで制限された Gauss スキームは以下のようになります．

```
grad(p) cellLimited Gauss linear 1;
```

4.4.4 Laplacian スキーム

laplacianSchemes サブディクショナリには Laplacian 項を記述します．流体力学の中で見られる $\nabla \cdot (\rho \nabla \boldsymbol{U})$ といった典型的な Laplacian 項をどのようにエントリに記述するかというと，laplacian(nu, U) といった word 識別子で与えます．離散化手法として選べるのは Gauss スキームだけですが，さらに拡散係数（この例では ν）の内挿スキームや，$\nabla \boldsymbol{U}$ に対する表面法線方向勾配スキームの両方を選択する必要があります．つまり，このエントリは以下のようになります．

```
Gauss <interpolationScheme> <snGradScheme>
```

内挿スキームは表 4.6 から選択しますが，通常は一般的なスキームから選択され，ほとんどの場合 linear にします．表面法線方向勾配スキームは表 4.7 から選択し，表 4.9 に書かれているようにスキームの選択は数値的性質を決定します．先の例での Laplacian 項の典型的なエントリは以下のようになります．

```
laplacian(nu, U) Gauss linear corrected;
```

4.4.5 発散スキーム

divSchemes サブディクショナリには発散項を記述します．流体力学の中で見られる典型的な対流項 $\nabla \cdot (\rho \boldsymbol{U} \boldsymbol{U})$ はどうように記述するかというと，OpenFOAM のアプリケーションでは通常 div(phi, U) という識別子で与えます．ここで phi はフラックス $\phi = \rho \boldsymbol{U}$ です．

スキーム	数値的性質
corrected	無制限，2 次，保存性
uncorrected	制限，1 次，非保存性
limited ψ	corrected と uncorrected の混合
bounded	制限スカラの 1 次
fourth	無制限，4 次，保存性

表 4.9 *laplacianSchemes* における表面法線方向スキームの性質

離散化手法として選べるのは Gauss スキームだけですが，さらに対象の場（この例では $\boldsymbol{U}$）の内挿スキームを選択する必要があります．つまり，このエントリは以下のようになります．

```
Gauss <interpolationScheme>
```

内挿スキームは，一般的なものや対流特有のものも含め，表 4.6 の中から選択します．この選択は，表 4.10 に示すように，数値的性質を大きく決定づけます．対流特有の内挿スキームを指定する場合でも，流束は特定の項として既に値がわかっているものとし，流束の内挿スキームは記述しません．つまり，例えば div(phi, U) の場合では，流束は phi として既知ですので，さらにその内挿スキームを指定すると矛盾が生じるだけです．よって，先の例での風上型内挿スキームの指定は次のようになります．

```
div(phi, U) Gauss upwind;
```

スキーム	数値的性質
linear	2 次，無制限
skewLinear	2 次，(より) 無制限，ひずみ補正
cubicCorrected	4 次，無制限
upwind	1 次，制限
linearUpwind	1 次/2 次，制限
QUICK	1 次/2 次，制限
TVD schemes	1 次/2 次，制限
SFCD	2 次，制限
NVD schemes	1 次/2 次，制限

表 4.10 *divSchemes* において使用される補間スキームの性質

4.4.6 時間スキーム

一次の時間微分項 $(\partial/\partial t)$ は，*ddtSchemes* サブディクショナリで指定します．各項に対する離散化スキームは表 4.11 から選ぶことができます．

CrankNicholson スキームでは，Eular スキームと混合させる割合を決める係数 ψ を用います．$\psi = 1$ の場合には純粋な CrankNicholson，$\psi = 0$ の場合は純粋な Eular に対応します．純粋な CrankNicholson では不安定なケースにおいては，混合係数をいじることで計算を安定化させることができることがあります．

時間スキームを指定するときは，非定常問題用のアプリケーションは定常状態で実行する必要はなく，またその逆も同じであることに注意してください．例えば，非定常の層流非圧縮流れのコードである icoFoam を実行するときに，steadyState（定常状態）を指定したら，おそらく解は収束しないので，定常の非圧縮流れのためには simpleFoam を使うべきです．

スキーム	説明
Euler	1 次，制限，陰的
localEuler	局所時間ステップ，1 次，制限，陰的
CrankNicholson ψ	2 次，制限，陰的
backward	2 次，陰的
steadyState	時間導関数について解かない

表 4.11　*ddtSchemes* において使用可能な離散化スキーム

2 次時間微分項（$\partial^2/\partial t^2$）は，*d2dt2Schemes* サブディクショナリの中で指定します．*d2dt2Schemes* としては，Euler スキームのみが利用可能です．

4.4.7　流束の算出

fluxRequired サブディクショナリには，アプリケーションの中で流束を生成する場を書き出します．例えば，多くの液体力学アプリケーションでは，圧力の方程式を解くと流束が生成するので，そのようなケースでは *fluxRequired* サブディクショナリには単に圧力のための word 識別子である p を記載します．

```
fluxRequired
{
    p;
}
```

4.5　解法とアルゴリズム制御

方程式のソルバ（求解機），公差，およびアルゴリズムは *system* ディレクトリの *fvSolution* ディクショナリから制御されます．以下に示すのは，icoFoam ソルバに必要な *fvSolution* ディクショナリからの入力例です．

```
17
18  solvers
19  {
20      p
21      {
22          solver           PCG;
23          preconditioner   DIC;
24          tolerance        1e-06;
25          relTol           0;
26      }
27
28      U
29      {
30          solver           smoothSolver;
31          smoother         symGaussSeidel;
32          tolerance        1e-05;
33          relTol           0;
34      }
35  }
36
37  PISO
38  {
39      nCorrectors     2;
40      nNonOrthogonalCorrectors 0;
41      pRefCell        0;
42      pRefValue       0;
43  }
44
```

```
45
46 // ************************************************************************* //
```

fvSolution は実行されるソルバ特有のサブディクショナリを含んでいます．しかしながら，標準のソルバに使われる *fvSolution* の大部分は標準的なサブディクショナリの小さなセットが占めています．これらのサブディクショナリは本節の後半で説明する *solvers*，*relaxationFactors*，*PISO*，および *SIMPLE* を含んでいます．

4.5.1 線形ソルバ制御

例題の最初のサブディクショナリであり，すべてのソルバのアプリケーションに現れるサブディクショナリは `solvers` です．ここには各離散化方程式に使用されるそれぞれの線形ソルバを指定します．つまり強調すると，特定の問題を解くための一連の方程式やアルゴリズムを意味するアプリケーションとしてのソルバとは対照的に，線形ソルバという用語は一連の線形方程式の解くための数値演算方法のことを指します．以下では，「線形ソルバ」という用語は「ソルバ」と省略されることが多くありますが，その文脈によって曖昧さは避けられると思われます．

solvers 内の各エントリの構文には，その方程式で解くべき変数を表す word がキーワードとして用いられます．例えば icoFoam は，速度 $\boldsymbol{U}$ と圧力 p の方程式を解くので，`U` および `p` に対するエントリを書きます．このキーワードの後には，ソルバのタイプとこのソルバが使うパラメータを含むディクショナリが続きます．ソルバは，表 4.12 に示す OpenFOAM での選択肢から，`solver` キーワードで指定します．`tolerance`，`relTol`，`preconditioner` などのパラメータは次の節で説明します．

ソルバ	キーワード
前処理付き（双）共役勾配	`PCG`/`PBiCG`†
スムーサを使ったソルバ	`smoothSolver`
汎用幾何学的代数マルチグリッド	`GAMG`
陽的な系のための対角ソルバ	`diagonal`

† `PCG` は対称用，`PBiCG` は非対称用

表 4.12 線形ソルバ

ソルバは対称行列と非対称行列を区別します．行列の対称性は解かれている方程式の構造に依存し，ユーザがこれを決定することも可能ですが，例えば OpenFOAM が不適当なソルバが選ばれているかどうかをユーザにアドバイスするためにエラーメッセージを出すので，それは必須ではありません．

```
--> FOAM FATAL IO ERROR : Unknown asymmetric matrix solver PCG
Valid asymmetric matrix solvers are :

(
PBiCG
smoothSolver
GAMG
)
```

4.5.1.1 解の許容範囲

疎行列ソルバは反復計算，すなわち解の連続性により方程式残差を減少させることに基づいています．残差は表面上，解の誤差の尺度なので小さければ小さいほど，より正確な解となります．より正確にいえば，残差は，現在の解を方程式に代入して左右両辺の差をとった大きさにより評価されたものです．また，解析されている問題のスケールに依存しないように正規化されます．特定のフィールドで方程式を解く前に，初期の残差はそのフィールドの現在値に基づいて値を決めます．それぞれのソルバの反復計算の後に，残差は再評価されます．以下の条件のどちらかを満たせばソルバは停止します．

- 残差が**ソルバの許容値**以下に減少する，`tolerance`;
- 初期残差比率がソルバの**相対的な許容値**以下に減少する，`relTol`;
- 反復回数が**最大反復回数**を超える．`maxIter`;

ソルバの許容値は，その解が十分正確とみなせるくらい残差が十分小さくなるようなレベルにしておくべきです．ソルバの相対的な許容値は，初期値から最終的な解までの相対的な改善幅に制限をかけます．非定常解析においては，各時刻ごとに解をソルバの許容値にしっかり収束させるために，ソルバの相対的な許容値を 0 に設定するのが一般的です．`tolerance` および `relTol` といった許容値は全てのソルバに対してディクショナリで指定しなければなりませんが，`maxIter` はオプションです．

4.5.1.2 前処理付き共役勾配ソルバ

共役勾配ソルバには，さまざまな行列の前処理方法があり，ソルバディクショナリの `preconditioner` キーワードで指定します．それらの前処理手法を表 4.13 に記載します．

前提条件	キーワード
対角不完全 Cholesky 分解 (対称)	`DIC`
高速対角不完全 Cholesky 分解 (キャッシング付き `DIC`)	`FDIC`
対角不完全 LU (非対称)	`DILU`
対角	`diagonal`
幾何学的代数マルチグリッド	`GAMG`
前処理なし	`none`

表 4.13 前提条件オプション

4.5.1.3 緩和法ソルバ

緩和法を使うソルバにおいては，緩和法を指定する必要があります．緩和法オプションを表 4.14 に記載します．一般に `GaussSeidel` は最も信頼できるオプションですが，行列がおかしい場合でも，`DIC` であればより収束しやすくなります．場合によっては `GaussSeidel` による緩和も追加した，いわゆる `DICGaussSeidel` と呼ばれる方法がさらに有用です．

緩和法	キーワード
Gauss–Seidel	`GaussSeidel`
対角不完全 Cholesky 分解（対称）	`DIC`
対角不完全 Cholesky 分解（対称）と Gauss–Seidel	`DICGaussSeidel`

表 4.14 緩和法オプション

また，許容値パラメータに従って，残差が再計算される前に `nSweeps` というキーワードによっ

てスイープの数も定めなければなりません．

4.5.1.4 代数幾何マルチグリッドソルバ

代数幾何マルチグリッド (GAMG) の一般化された手法は以下の原則に従います．セル数が少ないメッシュで素早く解を導きます．そして，この解をより細かいメッシュに写します．正確な解を出すのに細かいメッシュ上に初期の推測としてその値を使います．最初により粗いメッシュを解くときの速度の増加がメッシュ改良とフィールド・データに関するマッピングによる負荷の増加より重いときに，GAMG は標準の方法より速くなります．実際には，GAMG は指定されたメッシュから計算を始め，徐々にメッシュを粗くもしくは細かくしていきます．ユーザはセルの `nCoarsestCells` の数に関して最も粗いレベルにおける大体のメッシュサイズを指定するだけで構いません．セルの統合は `agglomerator` キーワードによって指定されたアルゴリズムで実行されます．今のところ，`faceAreaPair` 法を薦めます．`MGridGen` の共有されたオブジェクト・ライブラリを指定する追加入力が必要な `MGridGen` オプションがあることに注意する必要があります．

```
geometricGamgAgglomerationLibs ("libMGridGenGamgAgglomeration.so");
```

OpenCFD の経験によれば，MGridGen メソッドよりも `faceAreaPair` メソッドの方が優れています．すべての方法において，`cacheAgglomeration` スイッチによって統合を任意にキャッシュできます．緩和法は 4.5.1.3 で説明したように `smoother` によって指定されます．その緩和法が各メッシュ密度レベルにおいて実行されるスイープ回数は `nPreSweeps` や `nPostSweeps`，`nFinestSweeps` のキーワードによって指定されます．`nPreSweepsh` への入力はアルゴリズムがメッシュを粗くするときに使われ，`nPostSweeps` への入力はアルゴリズムがメッシュを細分割するときに使われ，`nFinestSweeps` は解が最も細かいレベルにあるときに使われます．

`mergeLevels` キーワードは，レベルを粗く，もしくは細かくするスピードを制御します．多くの場合は 1 レベルずつ，すなわち `mergeLevels 1` のように設定するのが最適です．場合によって，特に簡単なメッシュに関しては，例えば `mergeLevels 2` のように一度に 2 レベル粗くまたは細かくすることによって，解析を確実に早くできます．

4.5.2 不足緩和解析

OpenFOAM でよく使われる *fvSolution* の 2 番目のサブディクショナリは緩和して制御する *relaxationFactors* で，計算の安定性を改良するのに使用されるテクニックなのですが，特に定常状態問題を解析する際に使われます．緩和は，領域の解析の前に解のマトリックスとソースを変更するか，または直接領域を変更することによって，反復計算時の変数の変化を制限することで行われます．緩和係数 α $(0 < \alpha \leq 1)$ は緩和の量を指定し，0 から $\alpha = 1$ まで変化し，強さは $\alpha \to 0$ に従って増加します．$\alpha = 0$ は連続した反復計算で変数を全く変化させない場合の解であり，極端なケースです．最適な α の選択は安定した計算を確実にすることができるくらい小さく，また反復計算をスムーズに進められる程度大きくしなければなりません．α の値が 0.9 程度であれば安定性を確保できることがありますが，著しく低い値，例えば 0.2 といった値は，反復計算を遅くするためほとんど使われることはありません．ユーザは，その場と係数に関連している `word` を最初に指定することによって，特定の場に対して緩和係数を指定できま

す．以下に，非圧縮定常状態層流の simpleFoam のチュートリアルの例で使われている緩和係数を示します．

```
17
18  solvers
19  {
20      p
21      {
22          solver          GAMG;
23          tolerance       1e-06;
24          relTol          0.1;
25          smoother        GaussSeidel;
26          nPreSweeps      0;
27          nPostSweeps     2;
28          cacheAgglomeration on;
29          agglomerator    faceAreaPair;
30          nCellsInCoarsestLevel 10;
31          mergeLevels     1;
32      }
33
34      "(U|k|epsilon|R|nuTilda)"
35      {
36          solver          smoothSolver;
37          smoother        symGaussSeidel;
38          tolerance       1e-05;
39          relTol          0.1;
40      }
41  }
42
43  SIMPLE
44  {
45      nNonOrthogonalCorrectors 0;
46
47      residualControl
48      {
49          p               1e-2;
50          U               1e-3;
51          "(k|epsilon|omega)" 1e-3;
52      }
53  }
54
55  relaxationFactors
56  {
57      fields
58      {
59          p               0.3;
60      }
61      equations
62      {
63          U               0.7;
64          k               0.7;
65          epsilon         0.7;
66          R               0.7;
67          nuTilda         0.7;
68      }
69  }
70
71
72  // ************************************************************************* //
```

4.5.3　PISO と SIMPLE アルゴリズム

OpenFOAM のほとんどの流体力学ソルバアプリケーションは，pressure-implicit split-operator (PISO) もしくは semi-implicit method for pressure-linked equations (SIMPLE) アルゴリズムを使用します．これらのアルゴリズムは，速度と圧力の方程式を解くための反復法で，PISO は

非定常状態の問題に，SIMPLE は定常状態の問題に使います．

両アルゴリズムはいくつかの初期解を求め，次に，それらを修正するという方法をとります．SIMPLE は 1 段階の修正しかしませんが，PISO は 1 段階以上で，大概は 4 段階以下の修正をします．したがって，U-125 ページの入力例に示したように `nCorrectors` キーワードで *PISO* ディクショナリの補正数を定めます．

非直交性メッシュからなる追加補正は標準の OpenFOAM ソルバアプリケーションの SIMPLE と PISO の両方で利用できます．例えば面が直行座標系に並べられる 6 面体のセルのメッシュのように，メッシュ内の各面において隣接するセルの中心間のベクトルに面が平行であるなら，メッシュは直交しています．非直交の補正数は U-125 ページの入力例に示すように `nNonOrthogonalCorrectors` キーワードによって定めます．例えば，直交メッシュを 0 として非直交性の度合いによって最大で 20 まで増加するようにするなど非直交の補正数は解くケースのメッシュに対応させます．

4.5.3.1 圧力の参照

非圧縮の閉鎖系では圧力は相対的で，重要なのは絶対値ではなく範囲です．この場合，ソルバはセル内の `pRefValue` の基準面を，`p` が圧力の変数解の名前の場合，`pRefCell` に設定します．圧力が `p_rgh` であるところでは，名前はそれぞれ `p_rghRefValue` と `p_rghRefCell` です．これらの入力は，一般に *PISO*/*SIMPLE* サブディクショナリに格納されて，ケースに応じてソルバがそれらを必要としたときに使われます．もしこれを忘れるとソルバは実行されずに，エラーメッセージが出ます．

4.5.4 その他のパラメタ

標準の OpenFOAM ソルバアプリケーションの多くの *fvSolutions* ディクショナリには，これまで本節で説明した以外の項目はありません．しかし，一般に，*fvSolution* ディクショナリはソルバ，アルゴリズム，または実際の何かを制御するどんなパラメータをもっていてもおかしくありません．どんなソルバでも，必要なパラメタを把握するためにソースコードを見ることができます．結局，何かパラメータやサブディクショナリがなければ，ソルバが実行されるとき，詳細なエラーメッセージが印字されて終了するでしょう．そのとき，それに応じて不足のパラメータを加えてください．

第5章
メッシュの生成と変換

本章では，OpenFOAM におけるメッシュの生成に関する話題について述べます．5.1 節では OpenFOAM においてメッシュがどのように記述されるか概説します．5.3 節では六面体格子ブロックのメッシュの生成を行う blockMesh ユーティリティについて説明します．5.4 節では三角表面形状から自動的に六面体格子や分割六面体格子の複雑なメッシュを生成する snappyHexMesh ユーティリティについて説明します．5.5 節ではサードパーティの製品で生成したメッシュを，OpenFOAM で読み込むことができるフォーマットに変換する手法もあることを述べます．

5.1　メッシュの記法

この節では，OpenFOAM の C++ のクラスがどのようにメッシュを扱うか，その仕様について説明します．メッシュは数値解析において不可欠のものであり，妥当で精密な解を得るためには一定の条件を満している必要があります．OpenFOAM は，実行時，メッシュが妥当かどうかの一連の条件を満しているか厳しくチェックし，もしその条件を満していない場合には，実行を止めます．このため OpenFOAM が実行する前に，サードパーティ製のメッシャで生成した大規模なメッシュを修正することに疲れてしまうかもしれません．OpenFOAM 上で受けいれられるようにするために，根気良く修正する羽目に陥いることがあります．それは残念なことではありますが，メッシュの妥当性のチェックを行わなかったら，計算が始まる前に解は発散してしまうこともあるわけですから，OpenFOAM がメッシュの妥当性を常にチェックすることは決して悪いことではありません．

通常，OpenFOAM は，任意の多角形の面に囲まれた 3 次元で定義される任意の多面体セルによってメッシュを定義しますので，セルの面の数は無制限であり，その面についても，辺の数は無制限で配列についても何の制約もありません．このような汎用性が高いメッシュを OpenFOAM では polyMesh と定義しています．このような形式のメッシュを用いていると，特に計算領域の幾何形状が複雑であったり，それらが何度も変更されるときに，メッシュの生成やその操作においてとても大きな自由度があります．しかしながら，このようにメッシュが無条件の汎用性をもった代償として，従来のツールによって生成されたメッシュを変換するのは難しいこともあります．そのため，OpenFOAM のライブラリは，既存のセル形状セットを元にした従来のメッシュのフォーマットを上手く扱う cellShape ツールを提供しています．

5.1.1　メッシュの仕様と妥当性の制約

OpenFOAM のメッシュのフォーマットである polyMesh や cellShape ツールを説明する前に，まず，OpenFOAM におけるメッシュの妥当性の制約について述べたいと思います．メッシュが満していなければならない条件とは以下の通りです．

5.1.1.1　点

点というのは，3 次元空間における位置であり，メートル (m) 単位のベクトルによって定義されます．点の集まりはリストに蓄積され，個々の点はリストにおける位置を表わし，0 から始まるラベルにより参照されます．この点のリストには，別々の点でありながら位置が全く同一である点や，一つの面にも属さない点が含まれることはありません．

5.1.1.2　面

面は点を順番に並べたものであり，ひとつひとつの点はラベルによって参照されます．面における点のラベル順は，隣接した二つの点が一つの辺によって接続されるように付けられるため，面の周囲をぐるっと廻るように点の番号を追うことになります．点と同様に，面の集まりはリストで管理され，個々の面は，リストにおける位置を表わすラベルによって参照されます．面の法線方向ベクトルの向きは右手の法則により決まります．すなわち，図 5.1 のように，面に向って見たとき，点の順序が反時計廻りであったら，法線方向ベクトルはこちらを向いていることになります．

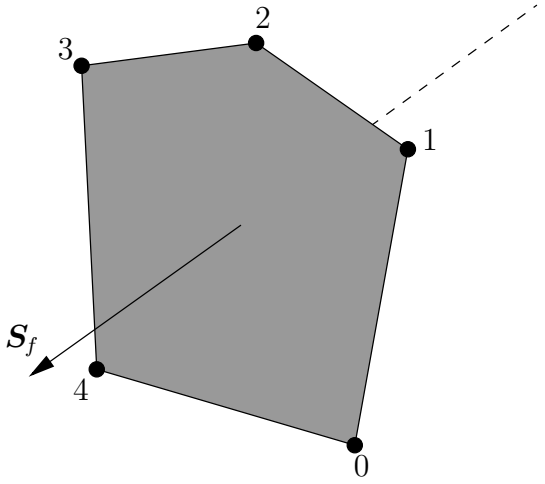


図 5.1　面における点の順序から決まる面領域ベクトル

面には 2 種類あります．

内部の面　これらの面は必ず二つのセルに接続されており，その数が 2 を超えることはありません．また，内部の面において，その法線方向ベクトルが，より大きなラベルをもつセルに向くように，点のラベルの番号付けがなされます．つまり，セル 2 とセル 5 を接続している面だったら，その法線はセル 5 を向くわけです．

境界の面　これらは領域の境界にあるので，一つのセルにしか属しません．したがって，ある境界の面を参照するのは，一つのセルと境界パッチだけです．点ラベルの番号付けは，面の法線が計算領域の外側に向くように設定されます．

5.1.1.3 セル

セルは，面を任意の順序で並べたものです．セルは以下に示す性質が必ず必要です．

切れ目なく連続である セル群は計算領域全体を完全にカバーしており，かつ，お互いに重複してはなりません．

凸である 全てのセルは凸で，かつ，セル中心はセルの内側にある必要があります．

閉じている 全てのセルは幾何的にも位相的（トポロジ的）にも閉じていなければなりません．ここで，セルが幾何的に閉じているためには，全ての面領域ベクトルがセルの外側を向いているとして，それらのベクトル和が，正確にゼロ・ベクトルとなる必要があります．また，セルが位相的に閉じているためには，問題において，セル中の全ての辺が，二つの面により使用されている必要があります．

直交性がある メッシュ内部の全ての面に対し，中心間ベクトルというのを，隣接する二つのセルの中心間を，小さいほうのラベルのセル中心から大きいほうのラベルのセル中心への向きで結んだベクトルとして定義することができます．直交性の制約というのは，内部の全ての面に対し，先に述べた面の面積ベクトルと中心間ベクトルのなす角が，常に $90°$ 未満であることをいいます．

5.1.1.4 境界

境界というのはパッチのリスト（集合）であり，これら一つ一つは，ある境界条件が割り当てられています．ここで，パッチというのは面のラベルのリストであり，境界の面のみで形成され，内部の面を含みません．この境界は閉じていることが条件であるので，境界における全面領域ベクトルの和は，数値計算上ゼロ・ベクトルになります．

5.1.2 polyMesh の記述

constant ディレクトリのサブディレクトリである *polyMesh* には，そのケースの polyMesh データが全て収められています．この polyMesh の記述は面ベースであり，既に述べたように，内部の面は二つのセルと接続し，境界面はセルと境界のパッチを指定します．各面には「保有」セルと「隣接」セルが割り当てられ，面を通じた接続は，保有セルと隣接セルのラベルによって簡潔に記述することができます．境界の場合には，面に接続されたセルがその面の保有者であり，隣接セルには -1 のラベルが割り充てられます．以上を踏まえた上で，以下のファイルで構成される入出力の詳細をご覧ください．

points セルの頂点を記述するベクトルのリストです．ここで，リストにおける最初のベクトルは頂点 0，次のベクトルの頂点 1 という風に番号付けします．

faces 面のリストです．各面は点中の頂点の番号のリストで成り立ってます．ここで，先程と同様に，リスト中の最初の面の番号は 0 です．

owner 保有セルのラベルのリストです．面のリストと同じ順番に並んでますので，リストの最初のラベルは 0 番の面の保有セルのラベル，次のラベルは 1 番の面の保有セルのラベルということになります．

neighbour 隣接セルのラベルのリストです．

boundary パッチのリストです．以下のように，パッチ名の宣言で始まる各パッチに対するディ

クショナリで構成されます．

```
movingWall
{
    type patch;
    nFaces 20;
    startFace 760;
}
```

startFace はそのパッチにおける最初の面のラベル番号です．また nFaces は，そのパッチ中の面数です．

備考：計算対象にいくつセルがあるか知りたい場合には，*owner* ファイルの FoamFile ヘッダにおける nCells を見てください．

5.1.3　cellShape ツール

標準的（でより単純）なメッシュ形式を，OpenFOAM のライブラリで扱えるように変換する際に，特に必要となるであろう cellShape というツールについても説明しておきたいと思います．

多くのメッシュ・ジェネレータや後処理システムは，実際にあり得る多面体セルの形状種類に対し，その一部だけをサポートするものがほとんどです．それらは，メッシュをセル形状セットといった，3 次元のセル幾何形状の限られた組み合わせで定義します．OpenFOAM のライブラリには，これらの一般的な形状集の定義がありますので，上記のようなメッシュを先の節で述べた polyMesh 形式に変換することができます．

OpenFOAM によってサポートされる cellShape モデルを表 5.1 に示します．形状は，形状モデルにおける番号付けスキームに従って付けれらた頂点ラベルの順序によって定義されます．点や面，辺に対する番号付けスキームも表 5.1 に書いてあります．点の番号付けは，形状がねじれたり，他の形状に変化することがないようにしなければならないので，同じ点番号は複数回使用できないことになります．さらに，重複した点は OpenFOAM では使う必要はありません．なぜなら，OpenFOAM で使用可能な形状は，六面体の変種を全てカバーしているからです．

セルの記述は，セルモデルの名前と，ラベルの順序リストという二つの部分より行います．例えば，以下の点のリストを使うと，

```
8
(
    (0 0 0)
    (1 0 0)
    (1 1 0)
    (0 1 0)
    (0 0 0.5)
    (1 0 0.5)
    (1 1 0.5)
    (0 1 0.5)
)
```

六面体セルは以下のように書けます．

```
(hex 8(0 1 2 3 4 5 6 7))
```

ここで，六面体セルの形状は hex というキーワードで記述しましたが，他の形状については，

表 5.1 に示したキーワードを使って記述できます．

5.1.4 1 次元や 2 次元，軸対称問題

OpenFOAM は 3 次元の空間用に設計されており，全てのメッシュもそのように定義します．しかしながら，OpenFOAM では，1 次元や 2 次元そして軸対称問題も解くことができ，それには，法線方向が意図する方向であるパッチに対して，特殊な境界条件を適用します．具体的には，1 次元や 2 次元問題では empty のパッチタイプを使い，軸対称問題では wedge タイプを使います．両者の使用法については 5.2.2 項で触れ，軸対称問題用の wedge 幾何形状の生成法については 5.3.3 項において述べます．

5.2 境界

本節では境界について述べます．境界はやや複雑です．なぜなら，形状の構成によって規定される単純なものではなく，境界条件や境界間の接続を通して解法を規定する不可欠の部分であるためです．境界はメッシュ，物理量，離散化，計算法といった多くの要素に関連しており，便宜上この章で扱います．

まず考えるべきことは，境界条件の適用のために，境界は分割されてパッチの組み合わせになるということです．一つのパッチは一つ以上の境界面に閉じられた領域をもち，それらが物理的に接続している必要はありません．

下に階層を示すように，パッチに関する性質は 3 種類あり，図 5.2 では各レベルにおけるさまざまなパッチの名前を挙げています．下で示す階層は OpenFOAM ライブラリの階層構造と類似しています．

Base type（基底型） 形状や情報の伝達を規定
Primitive type （基本型） 物理量の境界条件を規定
Derived type（派生型） Primitive type から派生した，複雑な境界条件を規定

5.2.1 パッチの形式の類型化

パッチの種類はメッシュと物理量のファイルに規定されます．もう少し正確にいえば，

- 基底型は *constant/polyMesh* ディレクトリにある *boundary* ファイル内の各パッチに対応する `type` キーワードに従って記述されます．
- 数値パッチ型は，基本型または派生型となり，フィールドファイルの各パッチに対応する `type` キーワードに従って記述されます．

例として sonicFoam のケースにおける *boundary* ファイルと *p* ファイル（圧力物理量ファイル）を示します．

```
17
18  6
19  (
20      inlet
21      {
```

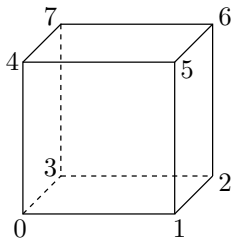
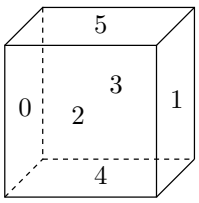
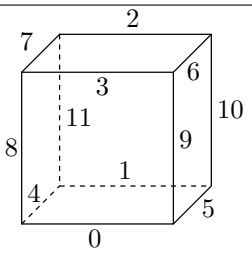
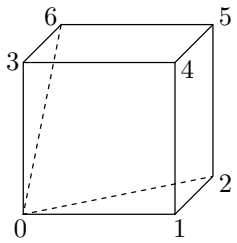
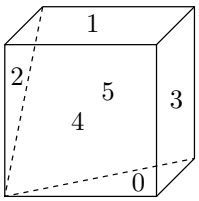
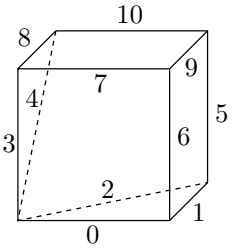
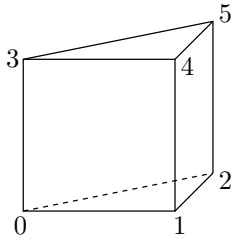
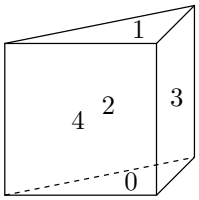
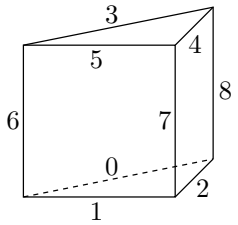
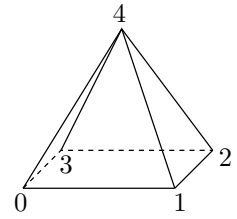
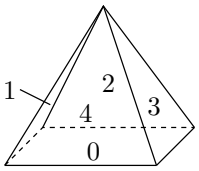
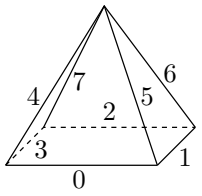
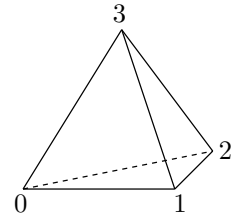
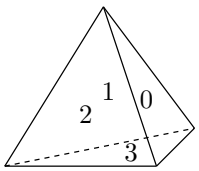
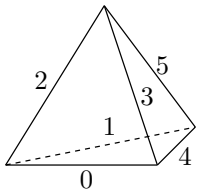
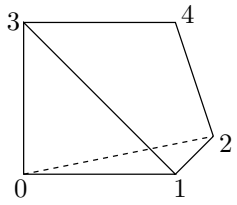
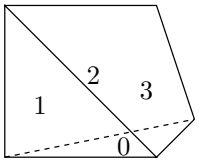
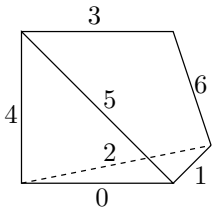
セルタイプ	キーワード	点の番号付け	面の番号付け	辺の番号付け
六面体	hex			
くさび形	wedge			
三角柱	prism			
四角錐	pyr			
四面体	tet			
くさび状四面体	tetWedge			

表 5.1　cellShapes における頂点，面，辺の番号付け

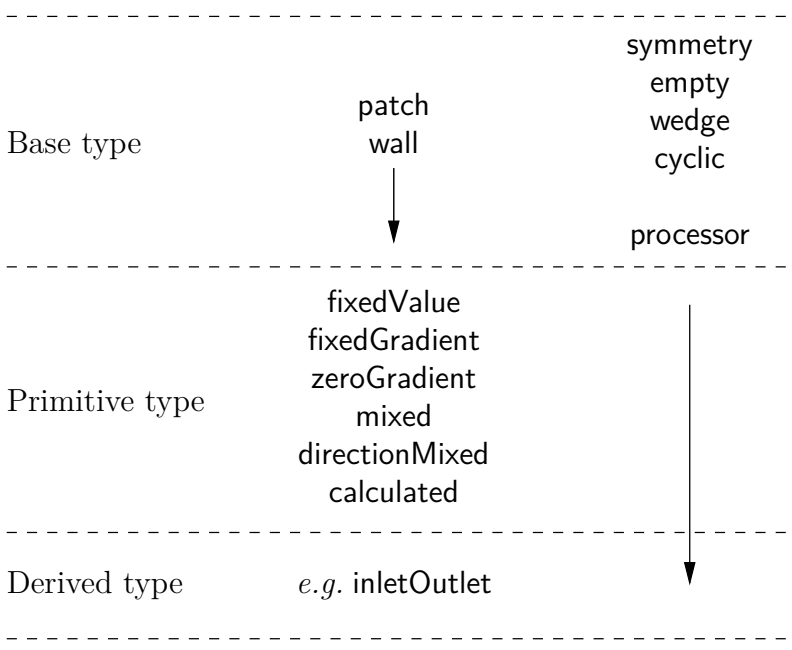


図 5.2 境界タイプの階層

```
22          type            patch;
23          nFaces          50;
24          startFace       10325;
25      }
26      outlet
27      {
28          type            patch;
29          nFaces          40;
30          startFace       10375;
31      }
32      bottom
33      {
34          type            symmetryPlane;
35          inGroups        1(symmetryPlane);
36          nFaces          25;
37          startFace       10415;
38      }
39      top
40      {
41          type            symmetryPlane;
42          inGroups        1(symmetryPlane);
43          nFaces          125;
44          startFace       10440;
45      }
46      obstacle
47      {
48          type            patch;
49          nFaces          110;
50          startFace       10565;
51      }
52      defaultFaces
53      {
54          type            empty;
55          inGroups        1(empty);
56          nFaces          10500;
57          startFace       10675;
58      }
59  )
60
61  // ************************************************************************* //
```

```
17  dimensions      [1 -1 -2 0 0 0 0];
```

```
18
19  internalField   uniform 1;
20
21  boundaryField
22  {
23      inlet
24      {
25          type            fixedValue;
26          value           uniform 1;
27      }
28
29      outlet
30      {
31          type            waveTransmissive;
32          field           p;
33          phi             phi;
34          rho             rho;
35          psi             thermo:psi;
36          gamma           1.4;
37          fieldInf        1;
38          lInf            3;
39          value           uniform 1;
40      }
41
42      bottom
43      {
44          type            symmetryPlane;
45      }
46
47      top
48      {
49          type            symmetryPlane;
50      }
51
52      obstacle
53      {
54          type            zeroGradient;
55      }
56
57      defaultFaces
58      {
59          type            empty;
60      }
61  }
62
63  // ************************************************************************* //
```

boundary ファイルにおける type は，symmetryPlane や empty といった幾何学的制約を受けるパッチを除くすべてのパッチに対して patch となっています．*p* ファイルには inlet や bottom といった面に適用される基本型と outlet に適用される複雑な派生型が記述されています．二つのファイルを比較すると，単純な patch ではなく，symmetryPlane や empty である場合，基底型及び数値型で一致していることがわかります．

5.2.2　基底型

以下に基底型の種類を挙げます．これらを規定するキーワードは表 5.2 にまとめてあります．

patch　メッシュに対する形状的，位相的情報をなにももたないパッチ条件のための基礎的なパッチ (wall の場合を除く)．流入口や流出口など．

wall　特に専門家が壁の境界を規定するときに，壁に適合するパッチが以下のように特定可能である必要がある場合があります．良い例としては，壁が wall パッチの型で特定されなければならない壁乱流モデルがあり，壁に隣接するセルの中心からの距離がパッチの一

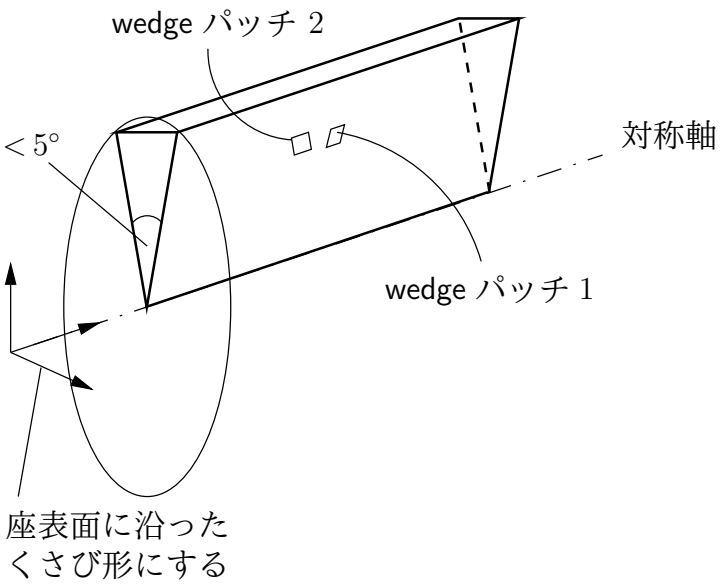


図 5.3 wedge パッチを利用した軸対象形状

種類	意味
patch	一般的なパッチ
symmetryPlane	対称面
empty	2 次元形状の前後の面
wedge	軸対称形状のための，くさび型の前後
cyclic	周期境界面
wall	壁面（乱流の壁関数に使用）
processor	並列計算時のプロセッサ間の境界

表 5.2 基底型の境界の種類

部として格納されます．

symmetryPlane 対称面

empty OpenFOAM が常に 3 次元で形状を生成する一方で，2 次元（1 次元）を解くことも可能です．そのためには，解が必要とされない 3 番目（2 番目）の次元に法線が向いている各パッチに特別な empty 条件を当てはめます．

wedge シリンダのような 2 次元の軸対称問題では，図 5.3 で示すように，小さい角度 (例えば $< 5°$) のくさびで，座標面の一つにまたがる対称面に沿って伸びている一つのセルとして形状が記述されます．軸対称くさび面は wedge 型という独自のパッチである必要があります．blockMesh を使ったくさびの形状の生成に関する詳細は 5.3.3 項に述べられています．

cyclic 熱交換管のような繰り返しの多い形状では，二つのパッチをあたかも一つのように扱うことができるようにする場合があります．ある cyclic パッチは *boundary* ファイル内の neighbourPatch キーワードでもう一つのパッチと結び付けられます．接続される面の各ペアは，*boundary* ファイル内の matchTolerance キーワードで与えられる許容誤差に収まるような，ほぼ等しい面積をもっている必要があります．面の方向が一致している必要はありません．

processor 多数のプロセッサで計算を並列実行する際には，各プロセッサがほぼ同数のセルを計算するようにメッシュを分割する必要があります．別々のメッシュの間の境界は

processor 境界とよばれます．

5.2.3　基本型

表 5.3 に基本型の種類を挙げます．

種類	物理量 ϕ に対して与える条件	与えるデータ
fixedValue	ϕ の値を指定	value
fixedGradient	ϕ の勾配を指定	gradient
zeroGradient	ϕ の勾配が 0	–
calculated	ϕ の境界条件が他の物理量から決まる	–
mixed	fixedValue と fixedGradient の組み合わせ．valueFraction に依存する条件	refValue, refGradient, valueFraction, value
directionMixed	テンソル型の valueFraction を用いる mixed 条件．例えば法線方向と接線方向で異なる取り扱いを行う場合	refValue, refGradient, valueFraction, value

表 5.3　基本型のパッチの種類

5.2.4　派生型

OpenFOAM には多数の派生型境界条件があり，ここには掲載しきれません．かわりに，ごく一部を表 5.4 に紹介します．利用できる全てのモデルの一覧を得たければ，OpenFOAM のソースコードを参照してください．派生型境界条件のソースコードは以下のような場所にあります．

- *$FOAM_SRC/finiteVolume/fields/fvPatchFields/derived* の中
- 特定のモデルライブラリの中．ターミナルで以下のようなコマンドを実行することで探せます．

```
find $FOAM_SRC -name "*derivedFvPatch*"
```

- 特定のソルバの中．ターミナルで以下のようなコマンドを実行することで探せます．

```
find $FOAM_SOLVERS -name "*fvPatch*"
```

5.3　blockMesh ユーティリティを使ったメッシュ生成

本節では，OpenFOAM 付属のメッシュ生成ユーティリティの blockMesh について説明します．blockMesh ユーティリティは，勾配付けや曲がった辺を使ったパラメトリックなメッシュを作成します．

メッシュはケースの *constant/polyMesh* ディレクトリに位置する *blockMeshDict* というディ

fixedValue から派生	意味	指定するデータ
movingWallVelocity	パッチでのフラックスが 0 となるよう，パッチでの値の法線方向を置き換える．	value
pressureInletVelocity	流入口の p が分かっているとき，$\boldsymbol{U}$ の値はフラックスから，法線方向はパッチから評価される．	value
pressureDirectedInletVelocity	流入口の p が分かっているとき，$\boldsymbol{U}$ の値はフラックスから計算され，inletDirection の方向となる．	value, inletDirection
surfaceNormalFixedValue	パッチの法線方向にベクトル型境界条件をその大きさによって指定する．領域の外側を示す方向が正．	value
totalPressure	全圧 $p_0 = p + \frac{1}{2}\rho\|\boldsymbol{U}\|^2$ は固定．$\boldsymbol{U}$ が変わるとそれに従い p も調整される．	p0
turbulentInlet	平均値のスケールに基づく変動変数について計算する	referenceField, fluctuationScale
fixedGradient/zeroGradient から派生		
fluxCorrectedVelocity	フラックスから流入口の $\boldsymbol{U}$ の法線成分を計算する	value
buoyantPressure	気圧勾配に基づく圧力に fixedGradient 設定する	—
mixed から派生		
inletOutlet	$\boldsymbol{U}$ の向きによって fixedValue と zeroGradient の間で $\boldsymbol{U}$ と p を切り替える	inletValue, value
outletInlet	$\boldsymbol{U}$ の向きによって fixedValue と zeroGradient の間で $\boldsymbol{U}$ と p を切り替える	outletValue, value
pressureInletOutletVelocity	pressureInletVelocity と inletOutlet の組み合わせ	value
pressureDirectedInletOutletVelocity	pressureDirectedInletVelocity と inletOutlet の組み合わせ	value, inletDirection
pressureTransmissive	周囲の圧力 p_∞ に超音速圧縮波を伝える	pInf
supersonicFreeStream	斜めの衝撃を p_∞，T_∞，U_∞ の環境に伝える	pInf, TInf, UInf
その他		
slip	ϕ がスカラなら zeroGradient, ϕ がベクトルなら法線成分は fixedValue 0 で, 接線成分は zeroGradient	—
partialSlip	混合 zeroGradient/slip 条件は valueFraction による．0 ならば slip.	valueFraction

Note: p は圧力, $\boldsymbol{U}$ は速度

表 5.4　派生型の種類

クショナリファイルから生成します．blockMesh はこのディクショナリを読み込んでメッシュを生成し，同じディレクトリの *points*，*faces*，*cells* および *boundary* ファイルにメッシュ・データを書き出します．

blockMesh がよりどころとする原則は，一つあるいは複数の 3 次元の六面体のブロックに領域を分割することです．ブロックの辺は，直線，円弧またはスプラインであるかもしれません．メッシュは，ブロックの各方向の多くのセルとして表面上指定され，これは blockMesh がメッシュ・データを生成するのに必要な情報です．

各ブロックの幾何形状は八つの頂点，六面体の各隅のひとつによって定義されます．頂点はリストの中に書かれ，各頂点にはそのラベルでアクセスできるようになっています．OpenFOAM は常に C++ の慣習に従って，リストの最初の要素をラベル '0' とします．リストに従って，各頂点に番号付けがされているブロックの例を図 5.4 に示します．頂点 1 と 5 を接続する辺をみてわかるように，blockMesh では曲線を作ることもできます．

5.3.3 項で説明するように，1 組以上の頂点をお互いに重ねることで八つ未満の頂点をもつブロックを生成することも可能です．

各ブロックは，右手系である局所座標系 (x_1, x_2, x_3) をもちます．右手系の軸群は，Oz 軸を見下ろしたとき，Ox 軸上の点から Oy 軸上への円弧が時計回りとなるように定義されます．局所座標系は以下に従ってブロックの定義で提示された頂点の順序に従って定義されます．

- 軸の原点はブロックの定義における最初の入力です．例では頂点 0 です．
- x_1 の方向は，頂点 0 から頂点 1 まで動くことによって示されます．
- x_2 の方向は，頂点 1 から頂点 2 まで動くことによって示されます．
- 頂点 0，1，2，3 は $x_3 = 0$ の平面を定義します．
- 頂点 4 は頂点 0 から x_3 方向へ辿っていくと見つかります．
- 頂点 5，6，および 7 は，頂点 1，2，および 3 からそれぞれ x_3 の方向へ辿っていくことで，同様に見つかります．

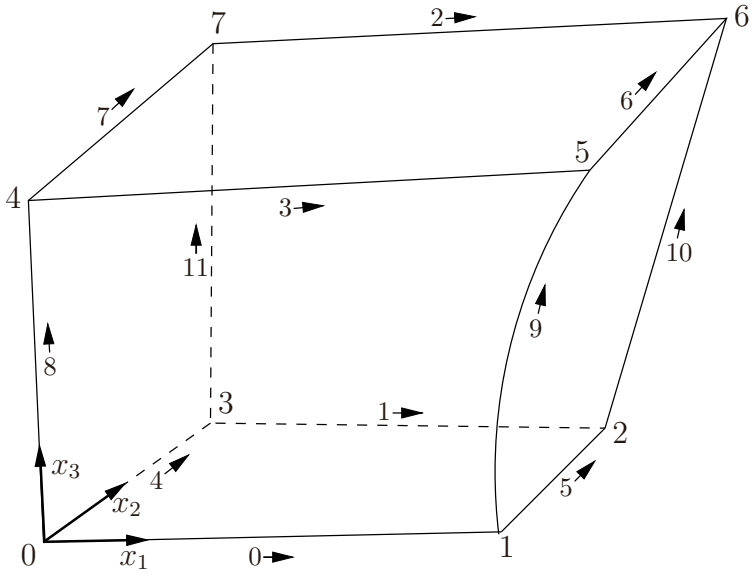


図 5.4　ひとつのブロック

キーワード	説明	指定するデータ
convertToMeters	頂点座標の倍率	0.001 とすれば mm
vertices	頂点座標のリスト	(0 0 0)
edges	arc もしくは spline の辺を書くために使用	arc 1 4 (0.939 0.342 -0.5)
block	頂点ラベルとメッシュサイズの順序リスト	hex (0 1 2 3 4 5 6 7) (10 10 1) simpleGrading (1.0 1.0 1.0)
patches	パッチのリスト	symmetryPlane base ((0 1 2 3))
mergePatchPairs	マージするパッチのリスト	5.3.2 項参照

表 5.5 *blockMeshDict* に使用するキーワード

5.3.1 *blockMeshDict* ファイルの記述

blockMeshDict ファイルは，表 5.5 で説明されているキーワードを使用するディクショナリです．convertToMeters キーワードは，メッシュ記述におけるすべての頂点の座標にかけられる尺度因子を指定します．例えば，

```
convertToMeters 0.001;
```

は，すべての座標に 0.001 をかけることを意味します．すなわち，*blockMeshDict* ファイルで引用された値が mm になります．

5.3.1.1 頂点

メッシュのブロックの頂点は，vertices と名づけられた標準のリストとして以下のように与えられます．例えば図 5.4 での私たちの例のブロックに関しては，頂点は以下のとおりです．

```
vertices
(
    ( 0 0 0 ) // vertex number 0
    ( 1 0 0.1) // vertex number 1
    ( 1.1 1 0.1) // vertex number 2
    ( 0 1 0.1) // vertex number 3
    (-0.1 -0.1 1 ) // vertex number 4
    ( 1.3 0 1.2) // vertex number 5
    ( 1.4 1.1 1.3) // vertex number 6
    ( 0 1 1.1) // vertex number 7
);
```

5.3.1.2 辺

2 頂点をつなぐ各辺はデフォルトで直線とみなされます．ただし，edges というリスト内のエントリで，いずれの辺も曲線として指定することができます．このリストはオプションです．ジオメトリ内に曲線が一つもなければ省略できます．

曲線の各エントリは，表 5.6 に挙げられているものからカーブのタイプを指定するキーワードとともに始まります．

そして，キーワードの後には辺が接続する二つの頂点のラベルが続きます．それに続いて，辺が通り過ぎる内挿点を指定しなければなりません．arc には，円弧が横切ることになる一つの内挿点が必要です．simpleSpline，polyLine，および polySpline に関しては，内挿点のリストが必要です．line 辺は，デフォルトとして実行されるオプションと全く同等であり，内挿点

キーワード選択	説明	追加するエントリ
arc	円弧	1 点の補間点
simpleSpline	スプライン曲線	補間点リスト
polyLine	線群	補間点リスト
polySpline	スプライン群	補間点リスト
line	直線	—

表 5.6　*blockMeshDict* ディクショナリで使用可能なエッジタイプ

を全く必要としません．line 辺を使用する必要は全くありませんが，それが完全性のために含まれていることに注意してください．図 5.5 に示した例題のブロックでは，内挿点 $(1.1, 0.0, 0.5)$ を通して以下のように頂点 1 と 5 をつなぐ arc 辺を指定します．

```
edges
(
    arc 1 5 (1.1 0.0 0.5)
);
```

5.3.1.3　ブロック

ブロックの定義は blocks と名づけられたリストに含まれています．各ブロックの定義は，5.3 節で示された順序をもつ頂点ラベルのリストからなる複合入力です．ベクトルが各方向に必要なセルの数，タイプ，および各方向のセル拡大比のリストを与えます．

そして，ブロックは以下のとおり定義されます．

```
blocks
(
    hex (0 1 2 3 4 5 6 7) // vertex numbers
    (10 10 10) // numbers of cells in each direction
    simpleGrading (1 2 3) // cell expansion ratios
);
```

それぞれのブロックの定義は以下のとおりです．

Vertex numbering　*OpenFOAM-2.0.0/cellModels* ファイルに定義されているように，最初の入力がブロックの形状識別子です．いつもブロックが六面体であるので，いつも形は hex です．U-142 ページで説明された方法で並べられた頂点番号のリストが従います．

Number of cells　2 番目の入力はそのブロックの x_1，x_2，x_3 とそれぞれの方向のセルの数を与えます．

Cell expansion ratios　3 番目の入力はブロックにおける各方向へのセルの拡大比を与えます．拡大比は，メッシュが指定された方向に段階的なものにするか，または精製されるのを可能にします．比率 δ_e は図 5.5 に示すように，ブロックのひとつの辺に沿った終わりのセルの幅の，辺に沿った始めのセル幅 δ_s への比です．以下のキーワードのそれぞれは blockMesh で利用可能な勾配付けの仕様の二つのタイプの一つを指定します．

simpleGrading　簡単な記述で，局所的な x_1，x_2 と x_3 方向それぞれに一様な拡大比を，三つの拡大比だけで指定します．例えば

```
simpleGrading (1 2 3)
```

edgeGrading　完全なセルの拡大比の記述は，図 5.4 に矢印で「最初のセルから最後のセル」の方向を表したスキームに従って番号付けられたブロックの各辺に比率を与えます．例えば，このようなものです．

```
edgeGrading (1 1 1 1 2 2 2 2 3 3 3 3)
```

これは，辺0–3に沿ったセル幅の比率が1，辺4–7に沿った比率が2であり，辺8–11に沿った比率が3であるということであることを意味しており，上述した simpleGrading の例にまったく同等です．

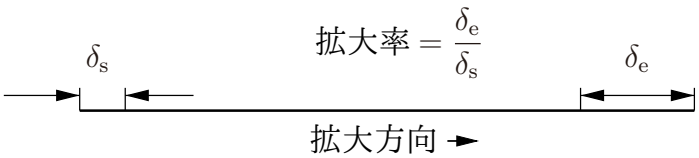


図 5.5 ブロックの辺に沿って段階わけされたメッシュ

5.3.1.4 境界

boundary というリストでメッシュの境界を与えます．境界はパッチ（領域）に分解され，リストされた各パッチは名前をキーワードとしてもっています．この名前はユーザが決めますが，例えば inlet のようにそのパッチの意味がわかりやすいような名前にすることをお勧めします．フィールドのデータファイルで境界条件を設定するときの識別にも，この名前が使われます．パッチの情報は以下のサブディクショナリ内に収められます．

- type: パッチタイプ．適用できる境界条件がいくつかある一般的な patch か，あるいは表 5.1 にリストアップされ 5.2.2 項で説明している特殊な幾何的条件のどちらか．
- faces: パッチを作るブロックの面のリスト．名前はユーザの選択に任されますが例えば inlet のように，パッチの特定に便利な名前を推奨します．この名前は，境界条件をフィールドのデータファイルに設定する際の識別に使用されます．

blockMesh は boundary リストに含まれない全ての面を集めて，それらに defaultFaces という名前で empty タイプのデフォルトパッチを割り当てます．これは，2 次元の幾何形状において，それらが必要に応じて empty パッチに集められるのを知りながら，ユーザは 2 次元平面にあるブロック面を省略する選択ができることを意味します．

図 5.4 での例のブロックに戻って，もし左面に流入があり，右面における流出があり，他の四つの表面が壁であるならば，以下のとおりパッチは定義できるでしょう．

```
boundary                // keyword
(
    inlet               // patch name
    {
        type patch;     // patch type for patch 0
        faces
        (
            (0 4 7 3)   // block face in this patch
        );
    }                   // end of 0th patch definition

    outlet              // patch name
    {
        type patch;     // patch type for patch 1
        faces
        (
            (1 2 6 5)
        );
    }
```

```
        walls
        {
            type wall;
            faces
            (
                (0 1 5 4)
                (0 3 2 1)
                (3 7 6 2)
                (4 5 6 7)
            );
        }
    );
```

それぞれのブロック面は四つの頂点番号のリストによって定義されます．頂点が与えられる順序は，ブロックの中から見て，どの頂点からも始めても，他の頂点を定義するために時計回りに面を回るようなものにならなければなりません．

blockMesh で cyclic パッチを指定するには，結びつけるパッチの名前を neighbourPatch キーワードで指定しなければなりません．例えば，ある一組の周期境界パッチは以下のように指定します．

```
    left
    {
        type            cyclic;
        neighbourPatch  right;
        faces           ((0 4 7 3));
    }
    right
    {
        type            cyclic;
        neighbourPatch  left;
        faces           ((1 5 6 2));
    }
```

5.3.2　複数のブロック

メッシュは一つ以上のブロックから作成されます．このため，前述したようにメッシュを作成することができますが，一つだけ追記しておくべきことはブロックどうしの接続です．これには 2 通りの可能性があります．

face matching（面の一致）　一方のブロックのパッチを構成する面の組が，他方のブロックのパッチを構成する面の組と同一の頂点の組からなる場合です．

face merging（面の合併）　一方のブロックのパッチをなす面のあるグループが，他方のブロックのパッチをなす面の別のグループに結合され，二つのブロックに接続された新しい内部面の組が作成されます．

face matching で 2 ブロックをつなげるためには，接続を形成する二つのパッチを patches リスト内から単に除外します．blockMesh は，面が外部の境界を形成せず，同じところに位置する各組を，2 ブロックからのセルを接続する，ひとつの内部面に結合するのを特定します．

もうひとつの face marging は，併合されるブロックパッチがまず patches リストで定義されることを必要とします．面が併合されるパッチのそれぞれの組が，mergePatchPairs というオプションリストに含まなければなりません．mergePatchPairs の形式は以下のとおりです．

```
mergePatchPairs
(
    ( <masterPatch> <slavePatch> ) // merge patch pair 0
    ( <masterPatch> <slavePatch> ) // merge patch pair 1
    ...
)
```

パッチの組は，最初のパッチはマスタになり，2 番目はスレイブになると解釈されます．併合するための規則は以下のとおりです

- マスタパッチの面は元々定義されているままで，すべての頂点は元の位置にあります．
- スレイブパッチの面は，スレイブとは多少異なるマスタパッチに投影されます．
- スレイブ面のどんな頂点の位置も，面の最小許容値より短いあらゆる辺を除去するために，blockMesh によって調整されるかもしれません．
- パッチが図 5.6 に示されるように重なるなら，併合しない各面が，境界条件を適用しなければならない，元のパッチの外部面として残ります．
- パッチのすべての面が併合されているなら，パッチ自体は表面を全く含まないので，除去されます．

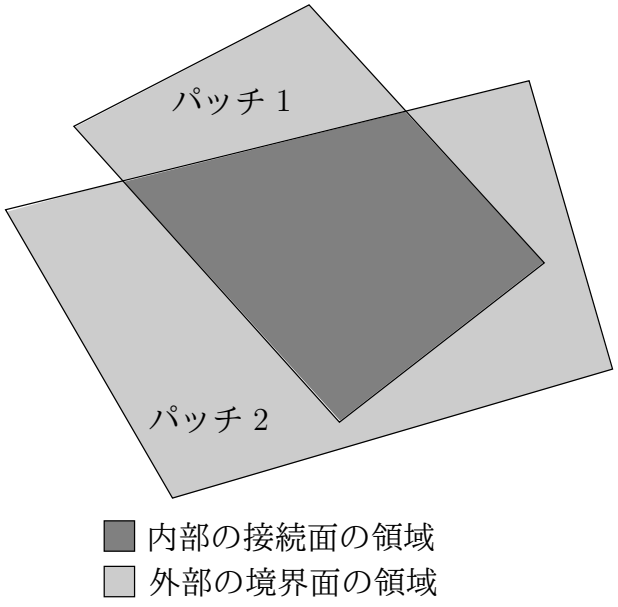


図 5.6 重なったパッチのマージ

結果的に，スレイブパッチのオリジナルの幾何形状が，併合の間必ずしも，完全に保存されるというわけではないということです．したがって，たとえば，円筒状のブロックが，より大きいブロックにつなげられている場合では，円筒状の形が正しく保存されるように，マスタパッチを円筒状のブロックにするのが賢いでしょう．併合手順を確実に成功させるためのいくつかの追加の推奨策があります．

- 2 次元の幾何学形状では，2 次元平面の外での 3 次元目のセルサイズは，2 次元平面でのセルの幅・高さと同様であるべきです．
- 二度パッチを併合すること，すなわち，mergePatchPairs で二度それを含めるのは勧められません．

- 併合されるべきパッチが，他の併合されるパッチと共通の辺を共有するところでは，両方がマスタパッチとして宣言されるべきです．

5.3.3　8 頂点未満のブロックの作成

八つ未満の頂点でブロックを作成するために，1 組以上の頂点をお互いの上で潰すことが可能です．頂点を潰す最も一般的な例としては，5.2.2 項で説明した wedge パッチタイプを使用する 2 次元の軸対称問題のための 6 面のくさび型ブロックを作成するときがあります．図 5.7 に示す私たちの例における，ブロックの簡易型のバージョンを使用することで，過程をわかりやすく例証します．頂点 7 を頂点 4 に，頂点 6 を頂点 5 に置いて潰すことによって，くさび型ブロックを作成したいということです．これは，ブロック番号 7 を 4 で，6 を 5 でそれぞれ交換することによって簡単にできます．するとブロック番号はこのようになります．

```
hex (0 1 2 3 4 5 5 4)
```

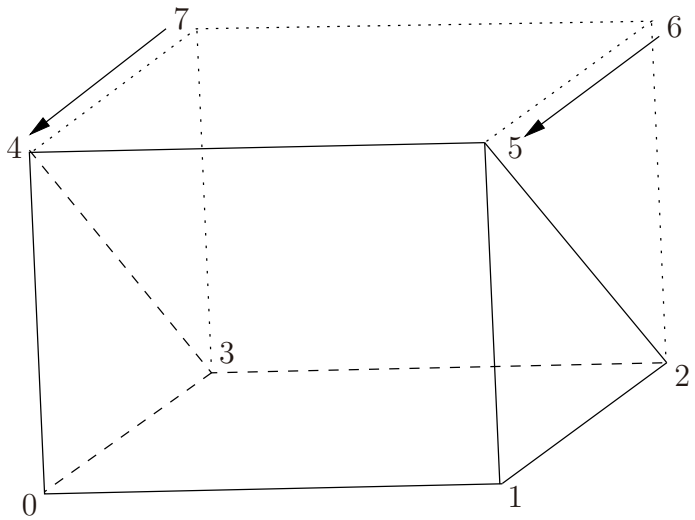


図 5.7　くさび形をしたブロックを六つの接点で作る

潰れている頂点を含むブロック面を考えることで，同じことがパッチにも適用でき，以前 (4 5 6 7) だったものが，(4 5 5 4) になります．これは面積をもたないブロック面で，polyMesh で面のないパッチを生成します．これは *boundary* ファイルにおいて同様の場合でも見ることができることと同じです．パッチは *blockMeshDict* で，empty として指定されるべきです．そしてどんなフィールドの境界条件も結果的に empty であるはずです．

5.3.4　blockMesh の実行

3.3 節で説明されたように，<*case*>ディレクトリのケースに対して blockMesh を実行するためには，以下のようにすればコマンドラインで実行できます．

```
blockMesh -case <case>
```

blockMeshDict ファイルは，サブディレクトリ *constant/polyMesh* に存在しなければなりません．

5.4 snappyHexMesh ユーティリティを使ったメッシュ生成

OpenFOAM のメッシュ生成ユーティリティ snappyHexMesh について解説します．snappyHexMesh は STL 形式の三角の表面形状から六面体と分割六面体の 3 次元メッシュを自動的に生成します．初期メッシュから細分化を繰り返し，できた六面体メッシュを表面に合わせて変形することで，徐々に表面形状を形成していきます．オプションとして，できたメッシュを縮小させ，セルレイヤを挿入することができます．メッシュの細分化のレベルは非常に柔軟性が高く，表面の処理はあらかじめ定義したメッシュの水準に適合します．snappyHexMesh は毎回負荷を平均化して並列動作をします．

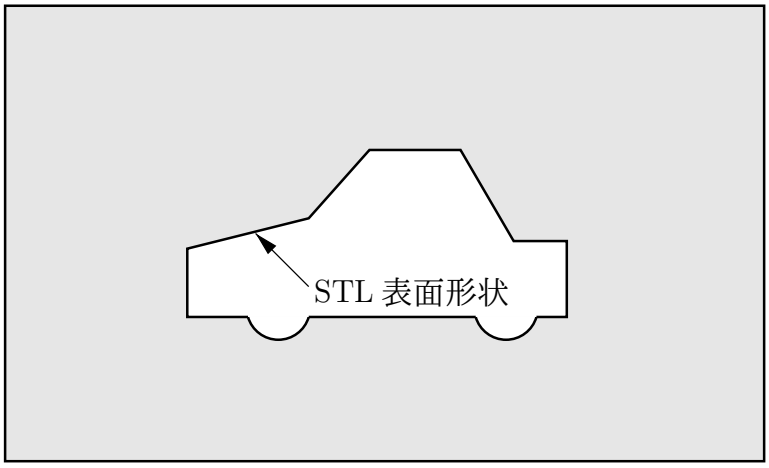


図 5.8 snappyHexMesh における 2 次元メッシュ問題の概略図

5.4.1 snappyHexMesh によるメッシュ生成の過程

図 5.8 に示す概略図を用いてメッシュを snappyHexMesh によって生成する流れを説明します．Stereolithography (STL) 形式の表面形状で描かれた対象を囲む長方形の部分（図中のグレーの部分）にメッシュを作成することを目的とします．これは外部の空気力学のシミュレーションにおいて典型的な手法です．あくまでも snappyHexMesh は 3 次元メッシュの生成ツールですが，簡単のためここでは 2 次元の図を使用しています．

snappyHexMesh を実行するには以下の準備が必要です．

- 2 進法または ASCII で表された STL 形式による表面形状データをケースディレクトリの *constant/triSurface* サブディレクトリに置く．
- 5.4.2 項で述べる blockMesh を使用して，解析領域の範囲とメッシュ密度の基準を決めるために六角形の基礎メッシュを作成しておく．
- ケースの *system* ディレクトリにある *snappyHexMeshDict* ディクショナリに，適切な内容を入力する．

snappyHexMeshDict ディクショナリには，メッシュ生成の様々な段階を管理する最上位での変更や，各過程における個々のサブディレクトリがあります．入力例を表 5.7 に示します．

snappyHexMesh で読み込む形状は *snappyHexMeshDict* 内の `geometory` の部分に記述します．形状は，STL による表面形状，または OpenFOAM による境界形状エントリによって指定でき

キーワード	意味	例
castellatedMesh	階段状のメッシュを作成するか	true
snap	表面への適合操作をするか	true
doLayers	レイヤの追加をするか	true
mergeTolerance	マージする許容値．初期メッシュのバウンディングボックスに対する比	1e-06
debug	中間メッシュと画面出力の制御	
	最終メッシュのみ出力	0
	中間メッシュの出力	1
	後処理のため cellLevel を付けた volScalarField を出力	2
	.obj ファイルとして中間時の交線を出力	4
geometry	使用する全表面形状のサブディクショナリ	
castellatedMeshControls	階段状メッシュ制御のサブディクショナリ	
snapControls	表面スナップ制御のサブディクショナリ	
addLayersControls	レイヤ追加制御のサブディクショナリ	
meshQualityControls	メッシュ品質制御のサブディクショナリ	

表 5.7　snappyHexMeshDict の最上位のキーワード

ます．

形状は STL 形状または OpenFOAM における幾何実体によって指定されます．以下に例を示します．

```
geometry
  {
      sphere.stl // STL filename
      {
          type triSurfaceMesh;
          regions
          {
              secondSolid             // Named region in the STL file
              {
                  name mySecondPatch; // User-defined patch name
              }                       // otherwise given sphere.stl_secondSolid
          }
      }

      box1x1x1  // User defined region name
      {
          type   searchableBox;       // region defined by bounding box
          min    (1.5 1 -0.5);
          max    (3.5 2 0.5);
      }

      sphere2  // User defined region name
      {
          type   searchableSphere;    // region defined by bounding sphere
          centre (1.5 1.5 1.5);
          radius 1.03;
      }
  };
```

5.4.2　六面体基礎メッシュの作成

snappyHexMesh を実行する前に blockMesh を使用して図 5.9 が示すように，解析領域をカバーする六面体セルの基礎メッシュを作成します．基礎メッシュの生成時は以下の点に注意しなければなりません．

- メッシュは六面体のみで構成されていること

- セルのアスペクト比がほぼ 1 であること．少なくとも連続したスナップが行われる表面近傍でそうでなければスナップの収束に時間がかかり，不良の原因となる．
- STL の表面とセルのエッジが最低でも一箇所は交差すること．つまり，一つのセルだけのメッシュでは機能しない．

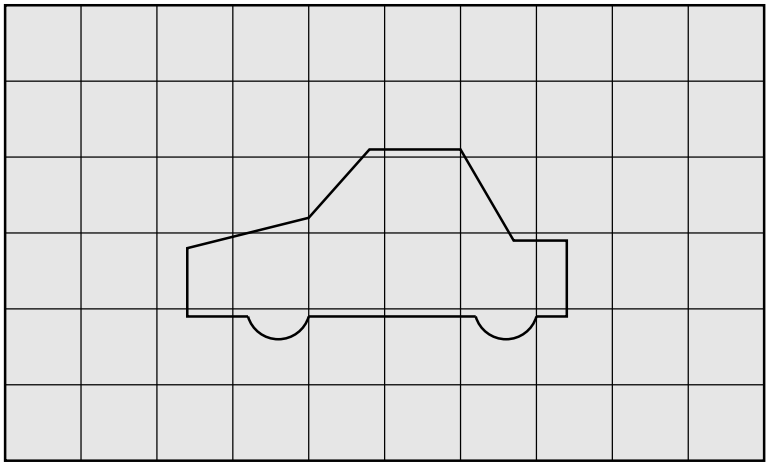


図 5.9 snappyHexMesh 実行前の基礎メッシュの生成

5.4.3 面と輪郭に合わせたセルの分割

セルの分割は，*snappyHexMeshDict* の *castellatedMeshControls* サブディクショナリにおいて設定して実行します．*castellatedMeshControls* の入力の例を表 5.8 に示します．

キーワード	意味	例
`locationInMesh`	メッシュが作成される領域内の位置ベクトル 位置ベクトルが細分化の前または最中にセルの面と一致してはいけない	`(5 0 0)`
`maxLocalCells`	細分化中におけるプロセッサあたりのセルの数の最大値	`1e+06`
`maxGlobalCells`	細分化中におけるセルの数の総数 (i.e. 除去の前)	`2e+06`
`minRefinementCells`	細分化すべきセルの数の最低値．この値以下だと停止	`0`
`nCellsBetweenLevels`	異なる細分化レベル間のセルの緩衝レイヤーの数	`1`
`resolveFeatureAngle`	角度がこの値を超えている交点をもつセルに最高レベルの細分化を行う	`30`
`features`	細分化する特徴辺のリスト	
`refinementSurfaces`	細分化する表面のディクショナリ	
`refinementRegions`	細分化する領域のディクショナリ	

表 5.8 *snappyHexMeshDict* の *castellatedMeshControls* サブディクショナリのキーワード

分割のプロセスは，図 5.10 に示したように，まず領域内で指定された輪郭に従って選択されたセルから始まります．*castellatedMeshControls* サブディクショナリの `features` リストには，edgeMesh のファイル名および細分化の `level` を含むディクショナリ・エントリを記述します．

```
features
(
    {
        file "features.eMesh"; // file containing edge mesh
        level 2;               // level of refinement
    }
```

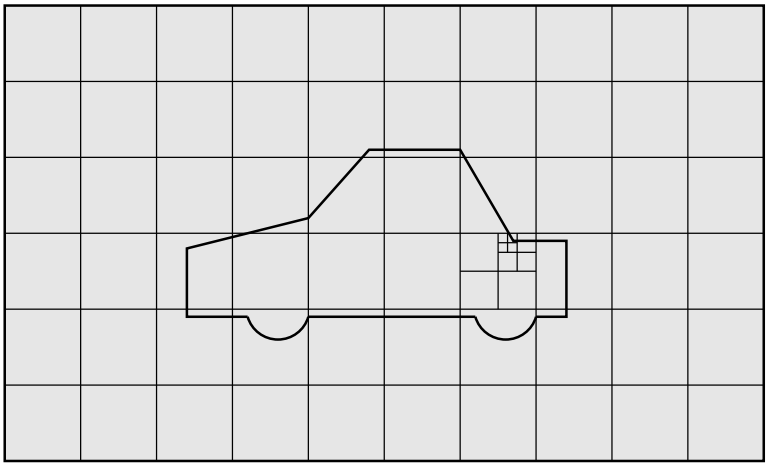


図 5.10　snappyHexMesh メッシングプロセスにおける輪郭によるセル分割

```
);
```

edgeMesh に含まれる輪郭データは，以下のように surfaceFeatureExtract を使って STL 形状ファイルから取り出すことができます．

```
surfaceFeatureExtract -includeAngle 150 surface.stl features
```

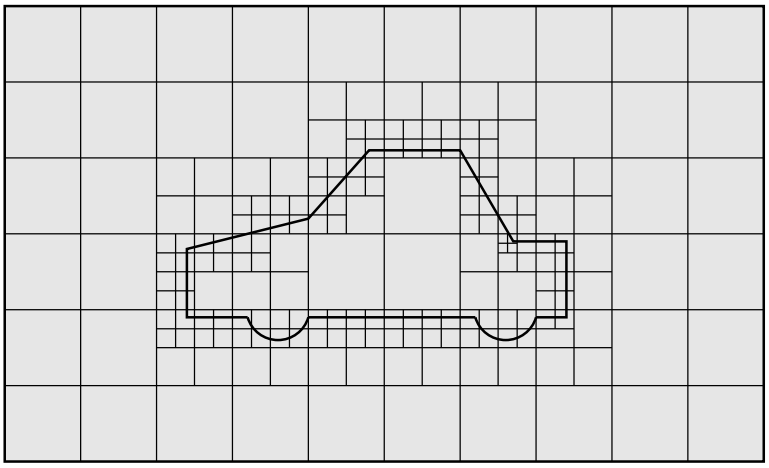


図 5.11　snappyHexMesh メッシングプロセスにおける表面によるセル分割

輪郭の細分化に続き，図 5.11 に示すように，指定された表面における分割のためにセルが選択されます．`castellatedMeshControls` の *refinementSurface* ディクショナリで，各 STL 表面のディクショナリ入力と，型の最小，最大細分化のデフォルトレベルの指定を行います．(`<min> <max>`) 最小レベルは表面のいたるところに適用され，最大レベルは `resolveFeatureAngle` に規定される角度を超過する交点をもつセルに適用されます．

細分化は STL 表面の特定領域に対して複数回行うことができます．領域の入力は *regions* サブディクショナリに収められています．各領域の入力に対するキーワードは領域の名前そのものであり，細分化のレベルはさらにサブのディクショナリに含まれます．以下の入力例を参考にしてください．

```
    refinementSurfaces
      {
          sphere.stl
          {
              level (2 2); // default (min max) refinement for whole surface
              regions
              {
                  secondSolid
                  {
                      level (3 3); // optional refinement for secondSolid region
                  }
              }
          }
      }
```

5.4.4 セルの除去

輪郭と表面の分割が完了するとセルの除去が始まります．セルの除去には領域内の有界表面によって完全に囲まれる一つ以上の範囲が必要です．セルが保持される領域は，`castellatedMeshControls` の `locationInMesh` キーワードに指定される領域内の位置ベクトルによって特定されます．セルの体積のほぼ 50 % 以上が領域内に存在する場合保持されます．残りのセルは図 5.12 に示すように除去されます．

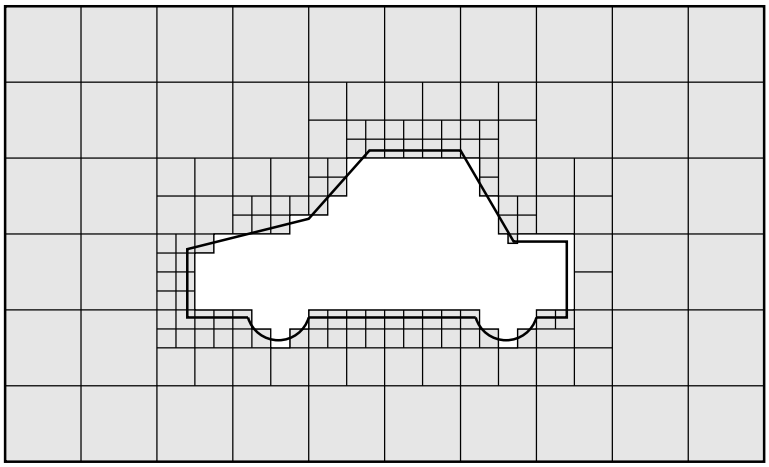


図 5.12 snappyHexMesh メッシングプロセスにおけるセルの除去

5.4.5 特定領域内のセルの分割

特定領域に含まれるセルはさらに細分化されます．図 5.13 では長方形の濃いグレーの領域が該当します．*castellatedMeshControls* 内の *refinementRegions* サブディクショナリでは，*geometry* サブディクショナリにおいて指定された領域の細分化の入力を行います．細分化の `mode` と対象領域は以下のとおりです．

`inside` 領域の内部を細分化します．
`outside` 領域の外部を細分化します．
`distance` 表面からの距離にしたがって細分化します．レベルキーワードを用いることで複数の距離にある異なるレベルにも適用できます．

refinementRegions では，細分化のレベルを levels 入力リストによって (<距離> <レベル>) のように記述します．inside と outside の細分化の場合，<distance>は不要で無視されますが，指定する必要があります．以下に入力例を示します．

```
refinementRegions
{
    box1x1x1
    {
        mode inside;
        levels ((1.0 4));          // refinement level 4 (1.0 entry ignored)
    }

    sphere.stl
    {                              // refinement level 5 within 1.0 m
        mode distance;             // refinement level 3 within 2.0 m
        levels ((1.0 5) (2.0 3)); // levels must be ordered nearest first
    }
}
```

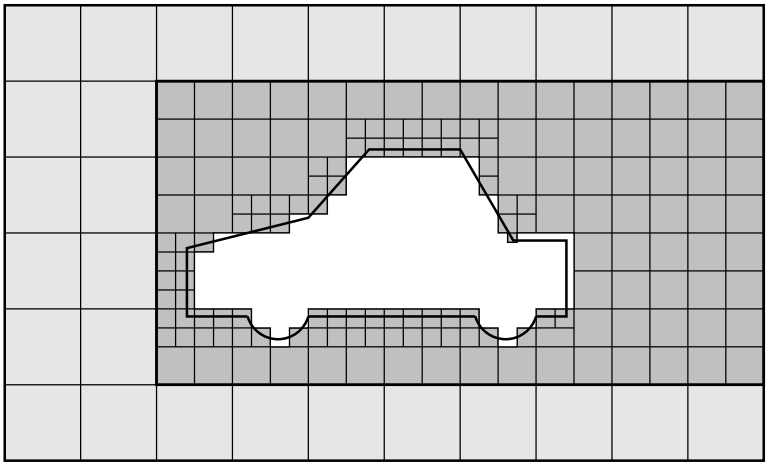


図 5.13　snappyHexMesh メッシングプロセスにおける領域によるセル分割

5.4.6　面へのスナップ

メッシュを生成する次の段階として，メッシュを平滑化するためにセルの頂点を表面に移動します．その手順は以下の通りです．

1. ギザギザの境界面の頂点を STL 表面上に移動する
2. 最後に移動した境界の頂点を用いて内部メッシュの緩和を求める
3. メッシュの水準に影響をもたらす頂点を探す
4. 最初の数値 (1) での頂点の移動を減らし，2 からメッシュの質が満足できるレベルに達するまで繰り返す．

表 5.9 に示す *snappyHexMeshDict* の *snapControls* サブディクショナリにおいて設定をします．
図 5.14 に概略図に例を示します．（メッシュの動きは多少現実と異なるように見えています．）

キーワード	意味	例
nSmoothPatch	表面との対応検索の前に行うパッチの平滑化の回数	3
tolerance	表面の特徴点や辺に引き寄せられる点に対する距離．その付近の辺の最大長に対する比	4.0
nSolveIter	メッシュの移動時の緩和反復回数	30
nRelaxIter	メッシュのスナップ時の緩和の最大反復回数	5

表 5.9 snapControls のキーワード

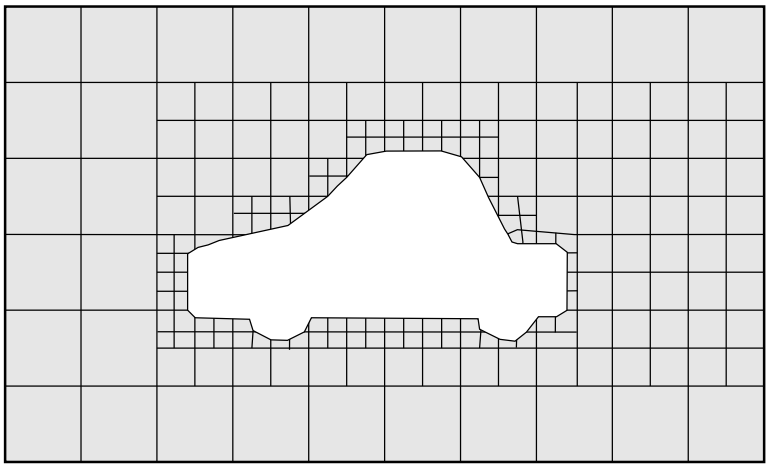


図 5.14 snappyHexMesh メッシングプロセスにおける表面のスナップ

5.4.7 メッシュレイヤ

境界面に沿った不規則なセルを作りもしますが，スナップによるメッシュの生成は目的に合致するでしょう．メッシュをかける過程にはさらにオプションがあり，図 5.15 の暗く影のついた部分が示すように，境界面に沿って並べられた六面体のセルのレイヤを追加します．

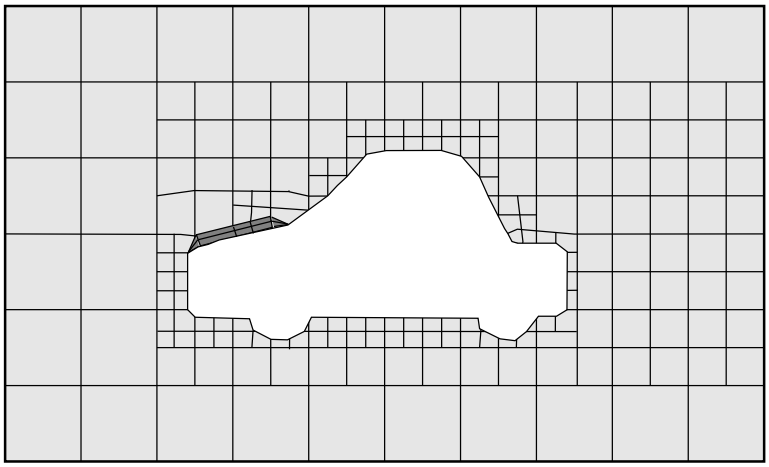


図 5.15 snappyHexMesh メッシングプロセスにおけるレイヤの挿入

メッシュのレイヤの追加は，以下の手順のように既存のメッシュを境界から縮小させ，レイヤを挿入することで行われます．

1. 表面に対して法線方向の厚み分だけメッシュを投影させる．
2. 最後に移動した境界面の頂点をもとに内部メッシュの緩和を計算する
3. 有効性を確認し，満足されていない場合は投影された厚みを減らし，2 からやり直す．い

かなる厚みでも有効性が満足できない場合はレイヤを挿入しない．

4. 有効性が確認できたらレイヤメッシュを挿入する．
5. メッシュを再度チェックし，不良箇所が見られる場合はレイヤを除去し 2 に戻る．

レイヤの追加の手順は *snappyHexMeshDict* の *addLayersControls* サブディクショナリの設定によって行われます．入力されるものは表 5.10 に示すとおりです．

キーワード	意味	例
layers	レイヤのディクショナリ	
relativeSizes	レイヤ厚さを，レイヤ外部の歪んでいないセルの大きさに対する相対値とするか，または絶対値とするか	true/false
expansionRatio	レイヤメッシュの拡大比率	1.0
finalLayerRatio	壁から最も遠い層の厚さ．relativeSizes エントリにより相対値か絶対値かが決まる	0.3
minThickness	セルのレイヤの最小の厚さ．相対値または絶対値（同上）	0.25
nGrow	点を押し出さないと生成されない面に結合されたレイヤの数．輪郭に近いレイヤ追加の収束に役立つ．	1
featureAngle	この角度以上では表面は押し出されない	60
nRelaxIter	スナップの緩和反復の最大回数	5
nSmoothSurfaceNormals	表面法線のスムージング反復数	1
nSmoothNormals	内部メッシュの運動方向のスムージング反復数	3
nSmoothThickness	表面パッチ上の滑らかなレイヤの厚さ	10
maxFaceThicknessRatio	極端にゆがんでいるセルでレイヤの生成を止める	0.5
maxThicknessToMedialRatio	中間の距離と厚さの比が大きくなるとレイヤの生成を抑制する	0.3
minMedianAxisAngle	中間の軸点選択に使う角度	130
nBufferCellsNoExtrude	新しいレイヤの末端のためのバッファ領域を作成	0
nLayerIter	レイヤを追加する反復計算全体の最大反復数	50
nRelaxedIter	この反復回数を超えた後は，meshQuality の *relaxed* サブディクショナリにおける制御値が使われる	20

表 5.10　*snappyHexMeshDict* の *addLayersControls* サブディクショナリのキーワード

レイヤのサブディクショナリはレイヤが適用される各パッチと必要な表面レイヤの数の入力を含んでいます．レイヤ追加は表面形状ではなく，できあがったメッシュに関連しているのでパッチ名が使われ，したがって表面領域に対してではなくパッチに対して適用されます．レイヤの入力例は以下のとおりです．

```
layers
 {
     sphere.stl_firstSolid
     {
         nSurfaceLayers 1;
     }
     maxY
     {
         nSurfaceLayers 1;
     }
 }
```

キーワード	意味	例
maxNonOrtho	非直交性上限角．180 にすると無効	65
maxBoundarySkewness	境界面ひずみ上限値． < 0 で無効	20
maxInternalSkewness	内部面ひずみ上限値． < 0 で無効	4
maxConcave	凹み上限角．180 で無効	80
minFlatness	実際の領域に対する最小の投影面積比率．-1 で無効	0.5
minVol	最小のピラミッドボリューム．大きな絶対値の負の数（例えば -1e30）で無効	1e-13
minArea	最小の界面の面積． < 0 で無効	
minTwist	最小の界面のねじれ． < -1 で無効	0.05
minDeterminant	最小正規化セル行列式．1 で六面体． ≤ 0 は不法なセル	0.001
minFaceWeight	$0 \rightarrow 0.5$	0.05
minVolRatio	$0 \rightarrow 1.0$	0.01
minTriangleTwist	Fluent 互換では > 0	
nSmoothScale	誤差分配反復数	4
errorReduction	エラー点の後方移動量のスケーリング係数	
relaxed	上述の各キーワードエントリに対して，レイヤ追加プロセス中に反復回数が nRelaxedIter を超えたときに使われる修正値を含んだサブディクショナリ	relaxed { ... }

表 5.11 *snappyHexMeshDict* の *meshQualityControls* サブディクショナリのキーワード

5.4.8 メッシュの品質制御

メッシュの品質は *snappyHexMeshDict* の *meshQualityControls* サブディクショナリへ入力することで制御できます．入力は表 5.11 に示します．

5.5 メッシュの変換

ユーザは，他のパッケージを使用してメッシュを生成し，OpenFOAM が用いる形式にそれらを変換できます．表 3.6 に示したような様々なメッシュ変換ユーティリティが用意されています．

よく使われるメッシュ変換ユーティリティのいくつかを以下に挙げ，使い方を紹介します．

fluentMeshToFoam Fluent の .msh メッシュファイルを読み込みます．2 次元，3 次元両方に使えます．

starToFoam STAR-CD/PROSTAR のメッシュファイルを読み込みます．

gambitToFoam GAMBIT の .neu ニュートラルファイルを読み込みます．

ideasToFoam ANSYS の .ans 形式で書かれた I-DEAS メッシュを読み込みます．

cfx4ToFoam .geo 形式で書かれた CFX メッシュを読み込みます．

5.5.1 fluentMeshToFoam

Fluent は，.msh 拡張子をもつ単一のファイルに，メッシュ・データを書き出します．ASCII 書式でファイルを書かなければなりませんが，それは，Fluent のデフォルト設定ではありません．2 次元の幾何形状を含んでいる単一の流れの Fluent メッシュを変換することは可能です．OpenFOAM では，2 次元幾何形状は，現在のところ，3 次元でメッシュを定義することで扱われます．そこでは，前面と背面は empty 境界パッチタイプと定義されます．2 次元の

Fluent メッシュを読みこむときに，コンバータは，自動的に 3 次元目の方向にメッシュを拡張し，`frontAndBackPlanes` と名づけ，空のパッチを加えます．

また，以下の特徴が見られます．

- OpenFOAM コンバータは，Fluent の境界条件の定義をできるだけ把握しようと試みるでしょう．しかしながら，OpenFOAM と Fluent の境界条件の間に明確で，直接的な対応は全くないので，ユーザはケースを実行する前に境界条件をチェックするべきです．
- 2 次元メッシュから軸対称なメッシュを生成することは現在サポートされていませんが，需要があれば実装されるでしょう．
- 複数の媒質からなるメッシュは受入れられません．もし複数の流体媒質が存在していると，それらは単一の OpenFOAM メッシュに変換されます．もし固体領域が検出されると，コンバータは，それを排除しようと試みます．
- Fluent ではメッシュの内部にパッチを定義することができます．つまり，面の両側にセルが存在する場合です．そのようなパッチは OpenFOAM では許容されていないので，コンバータはそれらを排除しようと試みます．
- 現在，埋め込まれたインタフェースと細分化ツリーに関してはサポートされていません．

Fluent `.msh` ファイルの変換手順は，まず必要なディレクトリとファイルを作成して新しい OpenFOAM ケースを作ることから始めます．ケースディレクトリには，*system* サブディレクトリの中に *controlDict* ファイルをおきます．そして以下のコマンドを実行します．

```
fluentMeshToFoam <meshFile>
```

ここで `<meshFile>` は絶対パスか相対パスによる `.msh` ファイルの名前です．

5.5.2 starToFoam

本節では STAR-CD コードで生成されたメッシュを，OpenFOAM のメッシュのクラスが読むことができる形式に変換する方法を説明します．メッシュは STAR-CD とともに供給されるどのパッケージでも生成できます．例えば PROSTAR，SAMM，ProAM およびそれらの派生物です．コンバータは，統合された任意のカップルマッチングを含むどんな単一流れのメッシュも受け入れ，すべてのセルタイプがサポートされます．コンバータがサポートしない特徴は以下のとおりです．

- 複数の流れのメッシュの仕様
- バッフル，すなわち，領域内に挿入された厚さなしの壁
- 部分境界，カップルマッチのうちの覆われていない部分は境界面であると考えられます．
- スライディング・インタフェース

複数の流れを含むメッシュに関しては，それぞれの独立した流れを別々のメッシュとして書き出し，それらを OpenFOAM で組み立て直すことで，メッシュ変換が実現できます．

OpenFOAM は，5.1 節で指定されたかなり厳しい妥当性評価基準に整合しているメッシュの入力だけを受け入れるという方針を採ります．無効なメッシュを用いて実行されることはなく，

それ自体が無効なメッシュは変換できません．以下の節では，STAR-CD とともに供給されたメッシュ生成パッケージを用いてメッシュを生成する際に，OpenFOAM 形式に変換できることを保証するために取らなければならない方法を説明します．これからの節において重複を避けるために，STAR-CD とともに供給されるメッシュ生成ツールは，STAR-CD という総称によって参照されることにします．

5.5.2.1 変換における一般的なアドバイス

starToFoam の変換を試みる前に，STAR-CD のメッシュチェックツールを実行することをお勧めします．そして変換の後に，新たに変換されたメッシュで checkMesh ユーティリティを実行するべきです．あるいは，starToFoam は，ユーザが問題のあるセルに注目することができるように，PROSTAR コマンドを含む警告を出すことがあります．問題の多いセルとマッチは，OpenFOAM でメッシュを使おうとする前にチェックして修正するべきです．OpenFOAM で動作しない無効なメッシュでも，課される妥当性評価基準が異なる別の環境では動くかもしれないことを覚えておいてください．

許容誤差のマッチングに関するいくつかの問題は，コンバータのマッチング許容誤差を使って解決することができます．しかしながら，その有効性には限界があり，マッチング許容誤差をデフォルトレベルから増加させることが明らかに必要であるということは，オリジナルのメッシュが正確でないことを示します．

5.5.2.2 不要なデータの消去

メッシュ生成が終了したら，流体セルが作成されて，他のすべてのセルが取り除かれると仮定して，あらゆる不要な頂点を取り除き，セル境界と頂点番号を圧縮してください．これは以下の PROSTAR コマンドで実行されます．

```
CSET NEWS FLUID
CSET INVE
```

CSET は空であるべきです．そうでないなら，CSET でセルを調べて，モデルを調整してください．もしセルを本当に必要としていないなら，PROSTAR コマンドを使用することでそれらを取り除くことができます．

```
CDEL CSET
```

同様に，頂点も取り除く必要があるでしょう．

```
CSET NEWS FLUID
VSET NEWS CSET
VSET INVE
```

これらの必要でない頂点を取り除く前に，不要な境界面を集めておかなければなりません．

```
CSET NEWS FLUID
VSET NEWS CSET
BSET NEWS VSET ALL
```

```
BSET INVE
```

BSET が空でないなら，不要な境界面は以下のコマンドを使用して削除することができます．

```
BDEL BSET
```

このとき，モデルは定義された境界面と同様に，流体セルとそれを支持する頂点だけを含むべきです．すべての境界面はセルの頂点によって完全に支えられるべきです．もしそうでないなら，すべてが正常になるまで幾何学形状を正常化し続けます．

5.5.2.3　デフォルトの境界条件の削除

デフォルトで STAR-CD は，明示的に境界領域と結びつけられていない全ての境界面に対して壁境界を適用します．そのほかの境界面は，default 境界領域に集められ，境界タイプ 0 として割り当てられます．これは人為ミスを誘発するので，OpenFOAM は，未定義の界面のための default 境界条件の概念を意図的に用意していません．例えば，すべての未定義の境界面にデフォルト条件を意図して与えたかどうかをチェックする手段は全くありません．

したがって，メッシュが首尾よく変換されるために，各 OpenFOAM メッシュに対するすべての境界を指定しなければなりません．以下で説明された手順を用いることで default 境界を実際の境界条件に変更する必要があります．

1. Wire Surface オプションで幾何形状をプロットしてください．
2. default 領域 0 と同じパラメータで追加の境界領域を定義してください．そして，境界ツールでゾーン指定を選択し，スクリーンに描かれたモデル全体のまわりに多角形を描くことで，全ての見えている面を 10 などの新しい領域に加えてください．PROSTAR で以下のコマンドを実行することによって，これができます．

   ```
   RDEF 10 WALL
   BZON 10 ALL
   ```

3. そのセットから，すでに定義されている境界タイプを全て外してください．境界領域から実行していきます．

   ```
   BSET NEWS REGI 1
   BSET NEWS REGI 2
   ... 3, 4, ...
   ```

   境界セットに関連している頂点を集め，次に頂点に関連している境界面を集めてください．それらは元のセットのように 2 倍あるでしょう．

   ```
   BSET NEWS REGI 1
   VSET NEWS BSET
   BSET NEWS VSET ALL
   BSET DELE REGI 1
   REPL
   ```

これは境界領域 1 の上で定義された境界領域 10 の面を与えるはずです．BDEL BSET と入力して，それらを削除してください．すべての領域にこれらを繰り返してください．

5.5.2.4 モデルの再番号付け

コマンドを使用することでモデルの番号を付け替えて，チェックしてください．

```
CSET NEW FLUID
CCOM CSET
VSET NEWS CSET
VSET INVE (Should be empty!)
VSET INVE
VCOM VSET
BSET NEWS VSET ALL
BSET INVE (Should be empty also!)
BSET INVE
BCOM BSET
CHECK ALL
GEOM
```

内部の PROSTAR の照合は，最後の二つのコマンドで実行されます．コマンドはいくつかの予見できない誤りを明らかにするかもしれません．また，PROSTAR は幾何学形状にではなく，STAR-CD のために因子を適用するだけであるので，スケール因子に注意してください．因子が 1 でないなら，OpenFOAM の scalePoints ユーティリティを使用してください．

5.5.2.5 メッシュデータの出力

メッシュがいったん完成されたら，モデルのすべての統合されたマッチをカップルタイプ 1 に置いてください．他のすべてのタイプが，任意のマッチを示すのに使用されるでしょう．

```
CPSET NEWS TYPE INTEGRAL
CPMOD CPSET 1
```

そして，計算格子の構成要素をそれら自身のファイルに書かなければなりません．これは以下のコマンドを実行し，境界に対して PROSTAR を用いることで行います．

```
BWRITE
```

デフォルトでは，これは.23 ファイル（3.0 の前のバージョン）か.bnd ファイル（バージョン 3.0 以降）に書きます．セルに対しては，以下のコマンド，

```
CWRITE
```

がセルを.14 か.cel ファイルに出力します．頂点に対しては，以下のコマンド，

```
VWRITE
```

が.15 か.vrt ファイルに出力します．現在の既定の設定では，ASCII 書式でファイルを書き出します．カップルが存在しているなら，拡張子.cpl をもつ追加カップルファイルが以下のコマンドをタイプすることによって書きだす必要があります．

```
CPWRITE
```

三つのファイルに出力した後に，PROSTAR を終了するか，ファイルを閉じてください．パネルに目を通して，すべての STAR-CD のサブモデル，材料，および流体の特性に注目してください．材料の特性と数学的モデルは，OpenFOAM ディクショナリファイルを作成し，編集することで設定する必要があります．

PROSTAR ファイルを変換する手順は最初に，必要なディレクトリを作成することで新しい OpenFOAM のケースを作ることです．同じディレクトリの中に PROSTAR ファイルを格納しなければなりません．そして，ユーザはファイル拡張子を変えなければなりません．.23 と.14 と.15 (STAR-CD バージョン 3.0 以前) か，.pcs と.cls と.vtx (STAR-CD バージョン 3.0 以降) から，それぞれ.bnd，.cel，および.vrt に変えます．

5.5.2.6　.vrt ファイルの問題

.vrt ファイルは，フリー・フォーマットというよりむしろ指定された幅に関するデータ列で書かれています．座標値が続く頂点番号を与えるデータの典型的な行は，以下のとおりです．

```
19422 -0.105988957 -0.413711881E-02 0.000000000E+00
```

縦座標が科学表記法で書かれていて，負であるなら，値の間には，スペースが全くないこともあります．例えば以下のような状況です．

```
19423 -0.953953117E-01-0.338810333E-02 0.000000000E+00
```

starToFoam コンバータは，縦座標の値を区切るためにスペースを区切り文字としてデータを読むので，前の例を読むとき，問題になります．したがって，OpenFOAM は必要なところで値の間にスペースを挿入するための簡単なスクリプト，foamCorrectVrt を含んでいます．すると，それが前の例を以下のように変換するでしょう．

```
19423 -0.953953117E-01 -0.338810333E-02 0.000000000E+00
```

したがって，必要ならば starToFoam コンバータを動かす前に，以下のようにタイプすることで foamCorrectVrt スクリプトを実行するべきです．

```
foamCorrectVrt <file>.vrt
```

5.5.2.7　OpenFOAM のフォーマットへのメッシュの変換

ここで，OpenFOAM の実行に必要な境界，セル，および頂点ファイルを作成するために，変換ユーティリティ starToFoam を実行できます．

```
starToFoam <meshFilePrefix>
```

<meshFilePrefix> は，メッシュファイルの絶対か相対パスを含んでいる接頭語の名前です．ユーティリティの実行後に，OpenFOAM 境界タイプは boundary ファイルを手で編集することによって指定します．

5.5.3 gambitToFoam

GAMBIT は.neu 拡張子をもつ単一のファイルにメッシュ・データを書き出します．GAMBIT の.neu ファイルを変換する手順は，最初に新しい OpenFOAM ケースを作成し，そしてユーザがコマンド・プロンプトで以下のコマンドを実行します．

```
gambitToFoam <meshFile>
```

ここで <meshFile> は絶対か相対パスによる.neu ファイルの名前です．

GAMBIT ファイル形式は例えば，壁，対称面，周期境界といったような境界パッチの種類に関する情報を提供しません．したがって，すべてのパッチがタイプパッチとして作成されます．メッシュ変換の後に必要に応じてリセットしてください．

5.5.4 ideasToFoam

OpenFOAM は I-DEAS によって生成されたメッシュを変換できますが，.ans ファイルとして ANSYS 形式で書きだされます．.ans ファイルを変換する手順は最初に新しい OpenFOAM ケースを作成し，そしてユーザがコマンド・プロンプトから以下のように実行します．

```
ideasToFoam <meshFile>
```

ここで <meshFile> は絶対か相対パスによる.ans ファイルの名前です．

5.5.5 cfx4ToFoam

CFX は.geo 拡張子をもつ単一のファイルにメッシュ・データを書き出します．CFX のメッシュ形式は，ブロック構造です．すなわち，メッシュは相互の関係と頂点の位置の情報をもつブロックの組として指定されます．OpenFOAM はメッシュを変換して，できるだけよく CFX 境界条件を得ようと試みるでしょう．単一の OpenFOAM メッシュに変換される全ての領域とともに，多孔質や固体領域などに関する情報を含む CFX の 3 次元の「パッチ」定義は無視されます．CFX は「デフォルト」パッチの概念をサポートし，そこでは，境界条件が定義されていない外部の面のそれぞれが壁として扱われます．これらの面はコンバータで集められ，OpenFOAM メッシュの defaultFaces パッチに入れられ，タイプ wall が与えられます．もちろん，それに続けてパッチタイプを変えることができます．

CFX での OpenFOAM の 2 次元形状は，一つのセルの厚さの 3 次元メッシュとして作成されます．もしユーザが CFX によって作成されたメッシュで 2 次元のケースを動かしたいなら，前後の面に関する境界条件は empty として設定します．ユーザは，計算面の他のすべての面に関する境界条件が正しく設定されていることを確かめるべきです．現在，2 次元の CFX メッシュから軸対称の幾何形状を作成するための機能はありません．

CFX の .geo ファイルを変換する手順は最初に新しい OpenFOAM ケースを作成し，そしてユーザがコマンド・プロンプトから以下のように実行します．

```
cfx4ToFoam <meshFile>
```

ここで <meshFile> は絶対か相対パスによる .geo ファイルの名前です．

5.6　異なるジオメトリ間のフィールドマッピング

mapFields ユーティリティは，別のジオメトリに対応するフィールドに与えられたジオメトリに関連する一つ以上のフィールドをマップします．フィールドが関連するジオメトリの間のどんな類似性も必要とされないほど完全に一般化されています．しかしながら，ジオメトリが一貫している場合は，マッピングの過程を簡素化する特別なオプションを用いて mapFields を実行できます．

mapFields について述べるために，いくつかの用語を定義する必要があります．まず，データがソースからターゲットまでマップされるといいます．ソースとターゲットフィールドの両方の幾何形状と境界タイプ，あるいは条件がまったく同じであるなら，フィールドは一貫していると考えられます．mapFields がマップするフィールド・データは，ターゲットとなるケースの *controlDict* の *startFrom*/*startTime* によって指定された時間ディレクトリの中のフィールドです．データは，ソースとなるケースの同等な時間ディレクトリから読み込まれて，ターゲットとなるケースの同等な時間ディレクトリに写像されます．

5.6.1　一貫したフィールドのマッピング

一貫したフィールドのマッピングは，以下の -consistent コマンドラインオプションを使用しながら，mapFields を（ターゲット）ケース上で実行することによって，簡単に実行されます．

```
mapFields <source dir> -consistent
```

5.6.2　一貫しないフィールドのマッピング

フィールドが図 5.16 のように一貫していないとき，mapFields はターゲットとなるケースの *system* ディレクトリに *mapFieldsDict* ディクショナリを必要とします．以下の規則がマッピングに適用されます．

- フィールド・データはどこでも，可能である限り，ソースからターゲットにマップされます．すなわち，私たちの例では，代替されないままで残っている網掛け領域を除いて，ターゲットとなる幾何形状に含まれるすべてのフィールド・データがソースからマップされます．
- 別の方法で *mapFieldsDict* ディクショナリで指定されない限り，パッチフィールド・データは代替されないままです．

mapFieldsDict ディクショナリは，パッチデータに関するマッピングを指定する二つのリストを含んでいます．最初のリストは，図 5.16 のように幾何形状が一致するソースとターゲットとなるパッチの組の間のデータのマッピングを指定する patchMap です．リストは，ソースとターゲットとなるパッチの名前のそれぞれの組を含んでいます．2 番目のリストは，ターゲットとなるパッチの名前を含む cuttingPatches です．そのターゲットのパッチの値は，ターゲットとなるパッチが切断するソースの内部のフィールドからマップされます．私たちの例における左下のターゲットとなるパッチにように，ターゲットとなるパッチがソースの内部のフィールドの一部を切断するだけの状況では，内部のフィールドに含まれるそれらの値はマップされ，外部にある値は変わりません．*mapFieldsDict* ディクショナリの例は以下に示します．

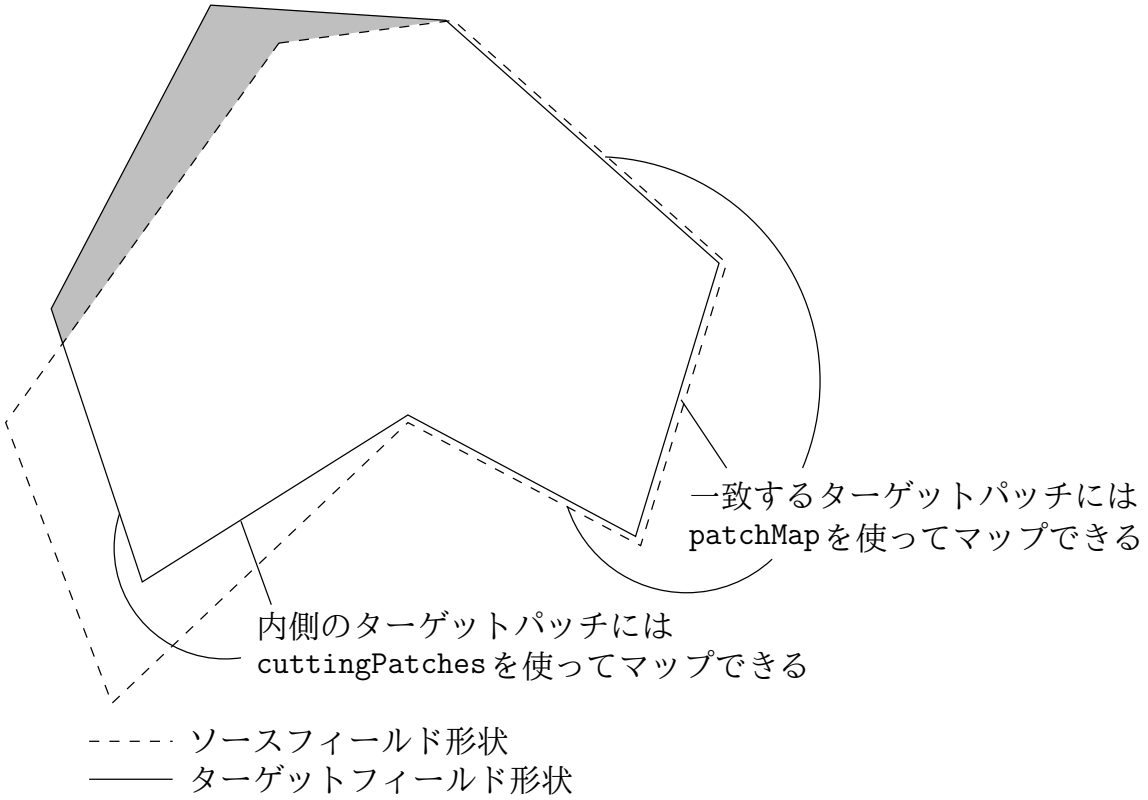


図 5.16 一貫しないフィールドをマップする

```
17
18 patchMap        ( lid movingWall );
19
20 cuttingPatches  ( fixedWalls );
21
22
23 // ************************************************************************* //
```

```
mapFields <source dir>
```

5.6.3　並列なケースのマッピング

mapFields を実行するとき，並列計算のためにソースとターゲットとなるケースのどちらかもしくは両方を分解するなら，追加オプションが必要になります．

`-parallelSource`　ソースケースが並列計算のために分解される場合
`-parallelTarget`　ターゲットケースが並列計算のために分解される場合

第6章
後処理

本章では，OpenFOAM での後処理のオプションについて述べます．OpenFOAM にはオープンソースの可視化アプリケーションである ParaView を用いた後処理のユーティリティである paraFoam が提供されており，これについては 6.1 節で述べています．後処理の別の方法としては，EnSight や Fieldview 等のサードパーティから供給されている製品を使う方法や Fluent の後処理を使う方法があります．

6.1 paraFoam

OpenFOAM で提供されているメインの後処理用のツールは，オープンソースの可視化アプリケーション ParaView から OpenFOAM のデータを読み込めるようにするモジュールです．このモジュールは，OpenFOAM により提供されている ParaView のバージョン 4.1.0 を用いている二つのライブラリである PV3FoamReader と vtkPV3Foam にコンパイルされています (バージョン 2.x の ParaView に対しては PVFoamReader および vtkFoam です)．最新のバイナリでリリースされているソフトウエアについても適切に動作するはずですが，このバージョンの ParaView をお使いになることを推奨します．ParaView に関する詳細な内容およびドキュメントについては http://www.paraview.org や http://www.kitware.com/products/paraviewguide.html のサイトから入手することができます．

ParaView はそのデータ処理とレンダリングのエンジンとして Visualization Toolkit (VTK) を使っているため，VTK フォーマットであれば，どのようなデータでも読み込むことができます．OpenFOAM には foamToVTK ユーティリティがあり，ネイティブな書式のデータを VTK の書式に変換することができ，つまり VTK ベースの画像ツールであれば，OpenFOAM のケースの後処理として使うことができます．これは，OpenFOAM で ParaView を使用する代替の手段となります．ユーザには高度な使い方，並列処理における可視化を経験してほしいことから，フリーの VisIt を推奨します．これは，http://llnl.gov/visit/ から入手できます．

要約すると，OpenFoam の後処理用のツールとしては，ParaView の読み込みモジュールを推奨します．代わりの方法としては，OpenFOAM のデータを ParaView に読み込ませるために VTK フォーマットに変換する方法と，VTK ベースのグラフィックツールを用いる方法があります．

6.1.1 paraFoam の概要

paraFoam は厳密には，OpenFOAM で提供されている読み込みモジュールを用いて ParaView

を立ち上げるスクリプトです．他の OpenFOAM のユーティリティ同様に，ケースディレクトリ内からコマンド一つだけで，あるいは引数として-case オプションでケース・パスを付けて，実行します．

```
paraFoam -case <caseDir>
```

ParaView が立ち上がり，オープンすると図 6.1 のようになります．ケースは左側のパネルでコントロールされますが，それには次のような項目があります．

Pipeline Browser は，ParaView の中でオープンしているモジュールをリストアップしており，選択されたモジュールは青くハイライトされ，このモジュールに関するグラフィックスは，脇の目のボタンをクリックすることにより，有効・無効の切り替えができます．

Properties パネル には，時間や領域，およびフィールドなどのケースに関する入力条件の選択項目があります．

Display パネル は，色など，選択されたモジュールの可視化の表現をコントロールします．

Information パネル はメッシュのジオメトリとサイズのようなケースの統計値を表示します．

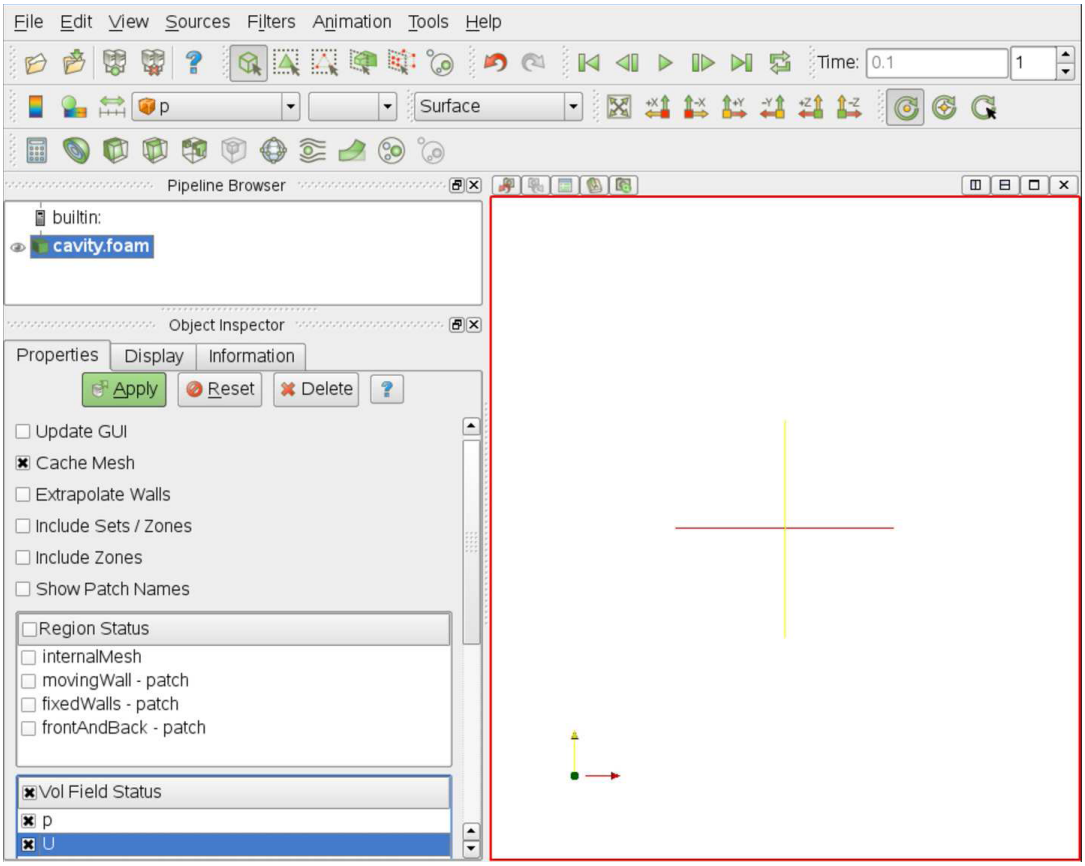


図 6.1 paraFoam の画面

ParaView はツリー構成に基づいた構造で操作するようになっており，その中で，トップレベルのケースのモジュールからサブモジュールのケースを作成するフィルタをかけることができます．例えば，圧力のコンタのプロットは，すべての圧力データをもつケースモジュールのサブモジュールとすることができます．ParaView の長所は，ユーザが数多くのサブモジュールを

作ることができることと，画像やアニメーションのどちらでも作ることができるという点にあります．例えば，ソリッドのジオメトリ，メッシュおよび速度ベクトル，圧力のコンタのプロットなどが追加できますし，これらアイテムについては必要に応じてオン・オフすることができます．

基本的な操作方法は，まず必要な項目の選択を行い，それから Properties パネルで緑色のをクリックします．そのほかのボタンとしては，必要に応じて GUI をリセットできる Reset ボタン，アクティブになっているモジュールを削除する Delete ボタンがあります．

6.1.2 Properties パネル

ケースモジュールの Properties パネルにはタイムステップや領域，およびフィールドの設定の機能があります．コントロール方法については，図 6.2 に説明を記載しています．現在の読み込みモジュールにおいて，ディレクトリ内のデータを ParaView に書き込むことは，特に価値はありません．6.1.4 項節に書いてあるように，現在の読み込みモジュールにおいて，Current Time Controls あるいは VCR Controls ツールバー内のボタンで，表示のための時間データを選択することができます．paraFoam の操作においては，何らかの変更を行ったときには Apply をクリックする必要があります．この Apply ボタンは，ユーザが変更するつもりがなかった場合を考慮して，警告を与えるために緑色にハイライトされます．この操作方法は，承諾する前に多くの選択ができるという長所をもっており，特に，大きなケースでは，データ処理が最小限で行えるという便利さがあります．しばしばファイルのケースデータが変更され，（たとえばフィールドデータが新しい時間ディレクトリに書き込まれたりしたために）ParaView を書き換える必要がある場合があります．変更を書き込む際には，Properties パネル一番上の Update GUI ボタンをチェックすることによって変更します．

6.1.3 Display パネル

Display パネルには，与えられたケースモジュールのデータの可視化に関する機能があります．以下が特に重要な点です．

- データのレンジは，フィールドの最大値・最小値に対して自動的に更新はされませんので，特に，初期のケースモジュールをロードしたときには，適切なインターバルを Rescale to Data Range で選択するように注意する必要があります．
- Edit Color Map ボタンでは，二つのパネルによるウィンドウが開きます．
  1. Color Scale パネルではスケールの色を選択することができます．標準の青〜赤の CFD スケールを選択するには，Choose Preset をクリックし，Blue to Red Rainbox HSV を選択します．
  2. Color Legend パネルではカラーバーの凡例の色を切り替えたり，フォントのような凡例のレイアウトを決定します．
- 基本となるメッシュは Style パネルにある Representation メニューの Wrireframe を選択することにより表示されます．
- Wrireframe が選択されている場合のメッシュのようなジオメトリは Color By メニューから Solid Color を選択し，Set Ambient Color ウィンドウで色を指定することにより可視化

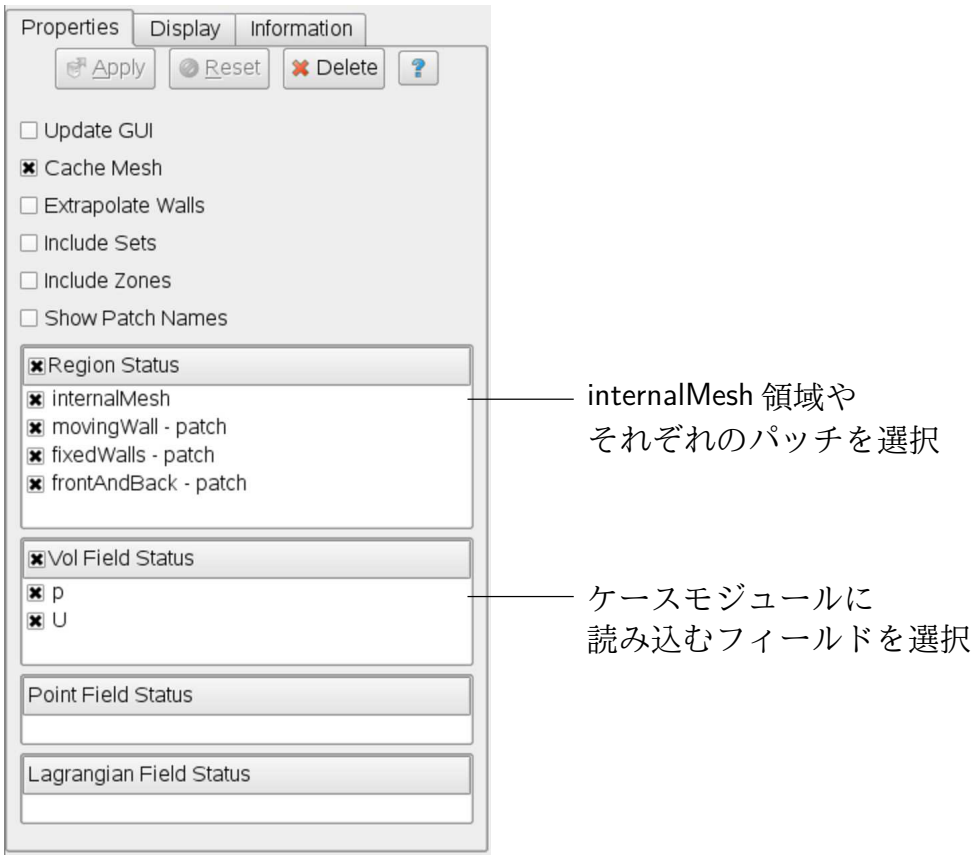


図 6.2　ケースモジュールのプロパティパネル

することができます．

- イメージは Opacity の値 $(1 = \text{solid}, 0 = \text{invisible})$ を修正することにより半透明にすることができます．

6.1.4　ボタンツールバー

ParaView の各機能はメインウインドウ上部のメニューバーのプルダウンメニューだけでなく，その下にあるボタンツールバーから選択することもできます．表示するツールバーは View メニューの Toolbars から選択することができます．各ツールバーの初期設定の配置は図 6.4 のようになっており，それぞれどのプルダウンメニューの項目に対応するかを示しています．多くのボタンの機能はアイコンから明快ですし，Help メニューの tooltips にチェックがされていればポインタを上に置いたときに簡潔な注を表示させることができます．

6.1.5　表示の操作

本節では，paraFoam におけるオブジェクトの表示の設定と取り扱いに関する操作について説明します．

6.1.5.1　表示の設定

View Settings を Edit メニューから選択すると，General，Lights，Annotation の 3 項目からなる View Settings (Render View) ウインドウが表示されます．General には以下のような項目が

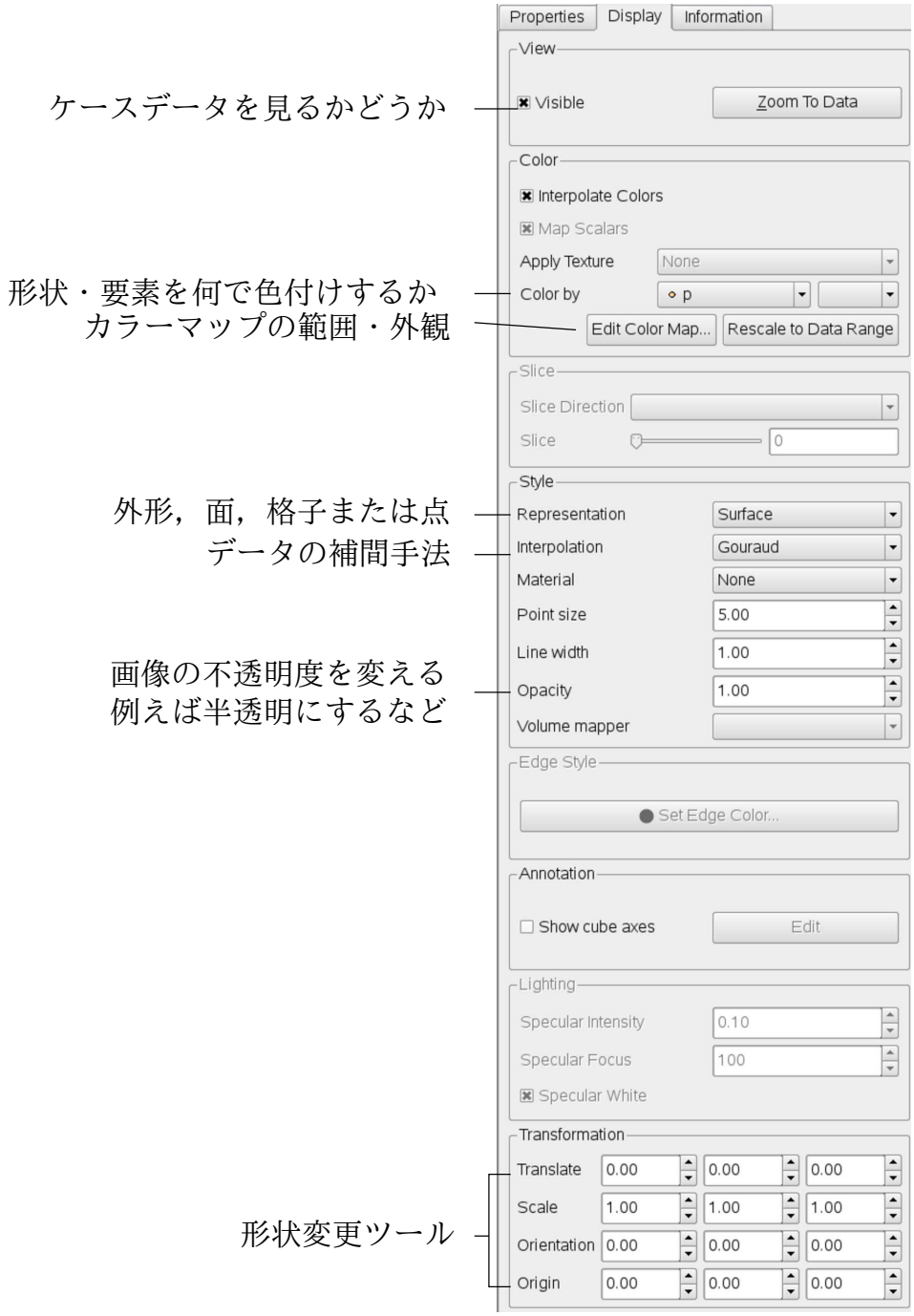


図 6.3 Display パネル

ありますので，開始時に設定するとよいでしょう．

- 背景色としては，印刷物には白が望ましいでしょう．そのためには Choose Color ボタンの横の下向き矢印から background を選択した後で，Choose Color ボタンをクリックして色を選択します．
- CFD，特に 2 次元のケースでは Use Parallel Projection（平行投影）がよく用いられます．

Lights パネルに含まれる Light Kit パネルでは光源の詳細設定ができます．Headlight パネルでは直接光をコントロールします．Headlight ボタンを白色光の強度 1 にすれば，等値面などの

図 6.4　ParaView のツールバー

鮮明な色の画像を得られるでしょう．

Annotation では，表示ウィンドウにおける軸や原点などの注釈の表示の有無を設定します．Orientation Axes で x，y，z 軸の色など，軸の表示設定をします．

6.1.5.2　一般的な設定

Settings を Edit メニューから選択すると General，Colors，Animations，Charts，および Render View の項目からなる Options ウインドウが表示されます．

General パネルでは ParaView の挙動の初期値を設定します．特に，Auto Accept をチェックすると Properties ウインドウで行った変更が Apply ボタンをクリックすることなく自動で表示に反映されるようになります．大きな解析ケースではこのオプションは使わない方がよいでしょう．というのもいくつもの変更を行う際にそのつど再描画されるのは煩わしく，一度で反映させた方がよいと思われるからです．

Render View パネルには General，Camera，Server の三つの項目があります．General パネルでは level of detail (LOD) で回転や平行移動，サイズ変更といった操作時のレンダリングの精度を設定できます．レベルを下げることで多数のセルからなるケースにおいても視点操作時の再描画速度を早くすることができます．

Camera パネルでは 3D または 2D における視点の移動を設定します．回転，平行移動，ズームといった操作は，マウスと shift キー，control キーを組み合わせて行うことができますが，割当ては任意に設定することができます．

6.1.6　コンタのプロット

コンタのプロットは，上部のメニューバーの Filter メニューから `Contour` を選択することにより作成することができます．フィルタはあたえられたモジュール上で役割を果たすことから，モジュール自体が 3D のケースの場合には，コンタは一定の値を表す 2D 表示（同値面：アイソサーフェス）に設定されます．コンタに関する Properties にはユーザが編集できる Isosurfaces のリストがあり，New Range ウィンドウにより使いやすくなっています．スカラフィールドはプルダウンメニューにより選択することができます．

6.1.6.1　断面の使い方

同一面でのコンタの作成でなく，断面のコンタを作成したい場合がよくあります．そうするには，最初に `Slice` フィルタを用いて，コンタをプロットしたい切断面を作成する必要があります．この `Slice` フィルタでは，ユーザは `Slice Type` メニューで `Plane`，`Box` または `Sphere` と

いう切断方法を指定することができます．それぞれ center および normal または radius の指定が必要です．マウスを使っても同じように切断面の操作を行うことができます．そして Contour フィルタを実行すれば，指定した切断面でコンタラインが作成されます．

6.1.7 ベクトルのプロット

ベクトルのプロットは Glyph フィルタを用いて作成します．このフィルタは Vectors で選択されたフィールドを読み込み，いくつかの Glyph Types が用意されていますが，Arrow によりクリアなベクトル画像が得られます．それぞれのグリフには，パネル上でユーザが操作して最も良い視覚効果を得るための，描画をコントロールする選択肢が備わっています．

Properties パネルの残りの部分にある主要なものはグリフの Scale Mode メニューです．その中でも最もよく使うオプションは，グリフの長さがベクトルの大きさに比例する Vector，全てのグリフを同じ長さにする Off です．また，Set Scale Factor はグリフの基本的な長さをコントロールします．

6.1.7.1 セルの中心でのプロット

ベクトルは，デフォルトではセルの頂点上に作成されますが，セルの中心にプロットデータを作成したい場合もあります．このためには，まずそのケースモジュールに対して Cell Centers フィルタを適用し，それから得られたセル中心データに対して Glyph フィルタを適用します．

6.1.8 流線

流線は，Stream Tracer フィルタを用いて作成されたトレーサラインを用いて作成されます．トレーサの Seed パネルで，Line Source あるいは Point Cloud 全般のトレーサポイントの配分を指定します．ユーザは線のようなトレーサソースを見ることができますが，白で表示させている場合は背景を変更しなければなりません．トレーサの軌跡の間隔とトレーサのステップの長さは Stream Tracer パネルの下にあるテキストボックスで指定します．望み通りのトレーサのラインを作成するプロセスは大部分が試行錯誤であり，ステップの長さを減少させることによりと同じように円滑にはっきりと表示することができますが，反面計算時間が長くなります．トレーサのラインが作成できた後は，より高品質な画像を作り出すために Tubes フィルタを *Tracer* モジュールに適用することができます．この Tubes は各々のトレーサのラインをたどっており，厳密な円筒型にはなっていませんが，固定された側面と半径の数値を持っています．上述のように側面の数値が 10 に設定されたとき，Tubes は円筒型として表示されますが，やはり，これにも計算コストがかかります．

6.1.9 画像の出力

ParaView から画像を出力する最も簡単な方法は File メニューから Save Screeshot を選択することです．選択すると，保存する画像の解像度を指定するウインドウが現れます．自動的に解像度が設定されるよう，アスペクト比を固定するボタンがあります．ピクセル解像度を設定すると画像が保存されます．より高画質で保存するには，x 方向の解像度を 1000 以上にするとよいでしょう．PDF 文書などに挿入する際に，A4 あるいは US レターサイズなどの書面に合うようにスケーリングすれば，シャープな仕上がりになります．

6.1.10 アニメーション出力

アニメーションを作成するには，まず File メニューから Save Animation を選択します．解像度などいくつかの項目を設定するダイアログウインドウが表示されるので，必要な解像度を指定します．それ以外では，タイムステップごとのフレーム数が重要です．これは直感的には 1 と設定するでしょうが，アニメーションのフレーム数を多くするためにより大きな値にしてもかまいません．この方法は，特に mpeg など，動画プレイヤの再生速度に制限がある場合に，アニメーションの速度を遅くしたいときに有効です．

Save Animation ボタンを押すと，ファイル名やファイル形式を設定する別のウインドウが現れます．OK を押すと，“<ファイル名>_<画像番号>.<拡張子>” という名前で一群の画像ファイルが保存されます．例えば，ファイル名を “animation” として jpg 形式で保存した場合の 3 番目の画像は，“animation_0002.jpg” となります．（<画像番号> は 0000 から始まります）

一連の画像が保存されると，適当なソフトを使って動画に変換することができます．ImageMagick パッケージに含まれる変換ユーティリティは，以下のようにコマンドラインから実行できます．

```
convert animation*jpg movie.mpg
```

mpg 動画を作成する際に初期設定の -quality 90% から動画のクオリティを上げるといいでしょう．これによって粒状ノイズを削減することができます．

6.2 Fluent による後処理

Fluent を，OpenFOAM で実行したケースに，ポストプロセッサとして適用することも可能です．その目的のために，二つの変換器が提供されています．foamMeshToFluent は，OpenFOAM のメッシュを Fluent フォーマットに変換し，それを .msh ファイルとして書き出します．そして，foamDataToFluent は，OpenFOAM の結果のデータを，Fluent が読むことができる .dat ファイルに変換します．foamMeshToFluent は，普通の方法で実行することができます．その結果のメッシュは，そのケースディレクトリの *fluentInterface* サブディレクトリに書き出されます．すなわち，*<caseName>/fluentInterface/<caseName>.msh* です．

foamDataToFluent は，OpenFOAM のデータの結果を，Fluent フォーマットに変換します．変換は，二つのファイルで制御します．まず *controlDict* ディクショナリに指定した startTime が，変換される計算結果の時刻となります．最新の結果を変換したいならば startFrom を latestTime と設定すればよいでしょう．二つめは翻訳方法を指定する *foamDataToFluentDict* ディクショナリです．これは *constant* ディレクトリ内に配置します．この *foamDataToFluentDict* ディクショナリの例を以下に示します．

```
1  /*--------------------------------*- C++ -*----------------------------------*\
2  | =========                 |                                                 |
3  | \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
4  |  \\    /   O peration     | Version:  2.3.0                                 |
5  |   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
6  |    \\/     M anipulation  |                                                 |
7  \*---------------------------------------------------------------------------*/
8  FoamFile
```

```
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      foamDataToFluentDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

p               1;

U               2;

T               3;

h               4;

k               5;

epsilon         6;

alpha1        150;


// ************************************************************************* //
```

ディクショナリは，次の形式のエントリを含んでいます．

```
<fieldName> <fluentUnitNumber>
```

<fluentUnitNumber>は，Fluent ポストプロセッサが使うラベルです．Fluent は，ある決まったセットのフィールドしか認識しません．<fluentUnitNumber>の数の基本的なセットは，表 6.1 に引用されています．

Fluent 名	ユニット番号	OpenFOAM での一般名
PRESSURE	1	p
MOMENTUM	2	U
TEMPERATURE	3	T
ENTHALPY	4	h
TKE	5	k
TED	6	epsilon
SPECIES	7	—
G	8	—
XF RF DATA VOF	150	gamma
TOTAL PRESSURE	192	—
TOTAL TEMPERATURE	193	—

表 6.1 ポストプロセッサのための Fluent のユニット番号

ディクショナリは，ユーザがポストプロセスに必要とする，全てのエントリを含まなければなりません．たとえば，我々の例では，圧力 p と速度 U のためのエントリをいれています．デフォルトエントリのリストは，表 6.1 に記述されています．ユーザは，他のユーティリティのように，foamDataToFluent を実行することができます．

Fluent でその結果を見るためには，ケースのディレクトリの *fluentInterface* サブディレクトリに移動して，3 次元のバージョンの Fluent を次のようにして開始します．

```
fluent 3d
```

メッシュとデータファイルは，ロードされ，その結果が可視化されます．メッシュは，File メニューの Read Case を選択することで読むことができます．あるデータタイプを読むためには，サポートアイテムを選択するべきです．例えば，k と epsilon の乱流データを読むには，ユーザは，Define -> Models -> Viscous メニューから k-epsilon を選択することになります．次に，データファイルは，File メニューの Read Data を選択することで，読むことができます．

注意すべき点 : ユーザは，OpenFOAM 形式への変換に使われたオリジナルの Fluent メッシュファイルを，OpenFOAM の解を Fluent フォーマットに変換したものと合わせて使ってはなりません．なぜなら，ゾーンの番号付けの順序が保証できないからです．

6.3 Fieldview による後処理

OpenFOAM は，OpenFOAM のケースを Fieldview でポストプロセスするための機能を提供しています．後処理ユーティリティの foamToFieldview を使って，OpenFOAM のケースデータを Fieldview .uns ファイルの形式に変換することができます．

foamToFieldview は，他の OpenFOAM のユーティリティと同じように実行することができます．foamToFieldview は *Fieldview* というディレクトリをケースディレクトリの中に作成します．すでに *Fieldview* ディレクトリが存在していた場合は削除されます．

デフォルトでは，foamToFieldview は全ての *time* ディレクトリのデータを読み込んで，*<case>_nn.uns* のようなファイルのセットに出力します．*nn* は連番で，最初の *time* ディレクトリの時刻歴データでは 1 から始まり，その後 2，3，4 と続きます．

ユーザーは，オプション -time <time> を使用して，特定の *time* ディレクトリのデータだけを変換することもできます．<time> は，一般的，科学的，または固定の形式です．

Fieldview の一部の機能は，境界条件に関する情報を必要とします．たとえば流線を計算するとき，境界面についての情報を使用します．foamToFieldview はデフォルトで境界面の情報を含むように試みます．ユーザは，コマンドオプション -noWall を使って，境界面情報を含まないように変換することもできます．

Fieldview の uns ファイルの拡張子は.uns です．変換元となった OpenFOAM のケース名にドット . を含んでいる場合，Fieldview は一連のデータを時系列データと解釈することができず，単一のデータ（定常データ）とみなすかもしれません．

6.4 EnSight による後処理

foamToEnsight を使って OpenFOAM のデータを EnSight の形式に変換するか，ensight74FoamExec モジュールを使って直接 EnSight から読み込むことによって EnSight で後処理を行うことができます．

6.4.1 EnSight の形式への変換

foamToEnSight は OpenFOAM のデータを EnSight の形式に変換します．foamToEnSight は普通のアプリケーションと同様に実行できます．foamToEnsight はケースディレクトリ内に *Ensight* というディレクトリを作成します．この際，既存の *Ensight* ディレクトリは削除されます．各時刻のディレクトリを読み込み，ケースファイルとデータファイルのまとまりとして書き込み

ます．ケースファイルは *EnSight_Case* という名前でデータファイルの詳細が含まれます．各データファイルは *EnSight_nn.ext* という名前で，*nn* は最初の時間ディレクトリの時刻を *1* として通し番号が入ります．*ext* は物理量に応じた拡張子です．たとえば，*T* は温度で，*mesh* はメッシュです．変換が完了すると EnSight で通常の方法で読み込むことができます．

- EnSight の GUI において，`File -> Data (Reader)` を選択する．
- ファイルボックス内で適切な *EnSight_Case* ファイルを強調表示させる．
- Format の選択肢は，EnSight のデフォルトの Case です．
- Case をクリックし，Okay を選択する．

6.4.2 ensight74FoamExec reader モジュール

EnSight ではユーザ定義のモジュールを用いて他の形式のファイルを EnSight に変換することが可能です．OpneFOAM には ensight74FoamExec というモジュールが libuserd-foam ライブラリにコンパイルされています．EnSight に必要なのはこのライブラリで，次節で述べるファイルシステムに置かれる必要があります．

6.4.2.1 EnSight の読込モジュールの設定

EnSight リーダの実行には，環境変数が適切である必要があります．*$WM_PROJECT_DIR/etc/apps/ensightFoam* 内の *bashrc* または *cshrc* ファイルで設定を行います．EnSight に関する環境変数は表 6.2 の`$CEI_`や`$ENSIGHT7_`です．EnSight の通常インストール時のパス設定では，`$CEI_HOME` のみ手動で設定すればよいはずです．

環境変数	説明とオプション
$CEI_HOME	EnSight がインストールされるパス (例：*/usr/local/ensight*) はデフォルトでシステムパスに加わる
$CEI_ARCH	*$CEI_HOME/ensight74/machines* のマシンディレクトリ名に対応する名前から選択したマシン構造．デフォルト設定では`linux_2.4` と`sgi_6.5_n32` を含む
$ENSIGHT7_READER	EnSight がユーザの定義した libuserd-foam 読込みライブラリを探すパス，デフォルトでは$FOAM_LIBBIN に設定
$ENSIGHT7_INPUT	デフォルトでは `dummy` に設定

表 6.2 EnSight で用いる環境変数の設定

6.4.2.2 読込モジュールの利用

EnSight reader を使う際の主要な問題は，解析ケースを OpenFOAM ではディレクトリで定義するのに対し，EnSight では特定のファイルによって定義されている必要があるということです．EnSight はディレクトリ名の選択ができないので，以下の手順で，特にケースの詳細を選択する際に注意しながら読込モジュールを使います．

1. EnSight の GUI において，`File -> Data (Reader)` を選択します．
2. Format メニューから OpenFOAM の選択ができるはずです．できない場合は，環境変数の設定に問題があります．
3. File Selection ウインドウからケースディレクトリを探し，Directories 欄の `/.` または `/..` で終わる，上二つのうち一つを強調表示させ，(Set) Geometry を選択します．

4. パスフィールドには解析ケースが入っています．(Set) Geometry の欄には/が含まれるはずです．
5. Okay をクリックすると EnSight がデータを読み込み始めます．
6. データが読込まれると，新しく Data Part Loader ウインドウが現れ，どの部分を読込むか尋ねられるので，Load all を選択します．
7. Data Part Loader のウインドウが表示されているといくつかの機能が動かないので，メッシュが EnSight のウインドウに表示されたら閉じます．

6.5　データのサンプリング

OpenFOAM はフィールドデータの任意の位置における値を取得するユーティリティ，sample を提供しています．グラフ上にプロットするために 1 次元の線上，または等値面画像を表示するために 2 次元平面上のデータが取得されます．データ取得位置は，ケースの *system* ディレクトリにある，*sampleDict* で指定します．データは良く知られているグラフパッケージ，Grace/xmgr，gnuplot，jPlot のような様々な形式で書き出すことができます．

sampleDict ディクショナリは，*$FOAM_UTILITIES/postProcessing/sampling/sample* の *sample* ソースコードディレクトリにある *sampleDict* の例をコピーすることで作成できます．*$FOAM_TUTORIALS/solidDisplacementFoam* の *plateHole* チュートリアルのケースにも 1D 線型データ取得のための記述例があります．

```
17
18  interpolationScheme cellPoint;
19
20  setFormat       raw;
21
22  sets
23  (
24      leftPatch
25      {
26          type    uniform;
27          axis    y;
28          start   ( 0 0.5 0.25 );
29          end     ( 0 2 0.25 );
30          nPoints 100;
31      }
32  );
33
34  fields          ( sigmaxx );
35
36
37  // ************************************************************************* //
```

このファイルには，次の入力項目があります．

interpolationScheme　データ内挿のスキーム
sets　フィールドが線型サンプルされる (1D) 解析領域の中の位置
surfaces　フィールドが面型サンプルされる (2D) 解析領域の中の位置
setFormat　線データ出力のフォーマット
surfaceFormat　面データ出力のフォーマット
fields　サンプルされるフィールド

キーワード	オプション	説明
interpolationScheme	cell	セル中心の値でセル全体が一定とみなす
	cellPoint	セルの値から線型重み付け補間
	cellPointFace	線型重み付けまたはセル界面から補間
setFormat	raw	ASCII 生データ
	gnuplot	gnuplot 形式データ
	xmgr	Grace/xmgr 形式データ
	jplot	jPlot 形式データ
surfaceFormat	null	出力しない
	foamFile	点，面，値のファイル
	dx	DX スカラまたはベクトル形式
	vtk	VTK ASCII 形式
	raw	xyz 座標と値．gnuplot の splot などで使われる
	stl	ASCII STL．表面のみ，値なし
fields	サンプルするフィールドのリスト，たとえば速度 U の場合，	
	U	$\boldsymbol{U}$ の全成分を出力
sets	1 次元の sets サブディクショナリのリスト．表 6.4 を参照	
surfaces	2 次元の surfaces サブディクショナリリスト．表 6.5 および表 6.6 を参照	

表 6.3 *sampleDict* におけるキーワード指定

interpolationScheme には，多面体の各セルを四面体に分割し，サンプルされる値が四面体頂点の値から補間される cellPoint と cellPointFace オプションがあります．cellPoint では，四面体の頂点は，多面体のセルの中心と，面の頂点三つからなります．セルの中心と一致する頂点は，セル中心のフィールド値を継承し，他の頂点はセルの中心の値から内挿した値をとります．cellPointFace では，四面体頂点の内の一つが，面の中心とも一致します．面が交差するセルの中心での値による内挿スキームによって，フィールド値を継承します．

ラインサンプリングのための setFormat エントリは，生データフォーマットと，グラフ描画パッケージのための gnuplot，Grace/xmgr，jPlot フォーマットがあります．データは，ケースディレクトリの *sets* ディレクトリに書き出されます．そのディレクトリは，一連の時刻ディレクトリに分割され，データファイルは，その中に格納されます．各々のデータファイルは，フィールド名，サンプルセット名，そして出力フォーマットに関係した拡張子をもつ名前が付けられます．拡張子は，生データでは.xy，Grace/xmgr 用には.agr，jPlot には.dat となります．gnuplot のフォーマットは，生の形式のデータと，それに加えてコマンドファイルを含んでいます．このコマンドファイルは，.gplt という拡張子をもち，グラフを生成するためのものです．sample が実行されるときは，既存の *sets* ディレクトリが削除されるので注意してください．

サーフェスサンプリングのための surfaceFormat エントリは，生データフォーマットとグラフ描画パッケージのための gnuplot，Grace/xmgr，jPlot フォーマットがあります．データは，ケースディレクトリの，*surfaces* ディレクトリに書き出されます．そのディレクトリは時間ディレクトリに分割され，データファイルは *sets* と同様にその中に格納されます．

fields リストは，データを取得したいフィールドが記述されます．sample ユーティリティは，次の限定された関数を，ベクトルやテンソルフィールドを修正することができるように，解析することができます．例えば，U のためには，

U.component(n) これは，ベクトル（テンソル）の n 番目の要素を書く．$n = 0, 1, \ldots$

mag(U) これは，ベクトル（テンソル）の大きさを書く

sets リストは，サンプルされるべきデータの位置の，サブディクショナリで構成されます．各サブディクショナリは，そのセットの名前に従って名前が付けられ，表 6.4 にも示すようにデータ取得位置に関する記述がなされます．

例えば，uniform サンプリングは，start と end ポイントで指定した直線上に，均等に分割した n 点でデータを取得します．全ての sets には，type とグラフ用の縦軸の長さを指定する axis キーワードを与えます．

		必要項目					
サンプル型	データ取得位置	name	axis	start	end	nPoints	points
uniform	線上に一様配分	•	•	•	•	•	
face	指定された線とセル界面の交点	•	•	•	•		
midPoint	線と界面の交点と交点の中点	•	•	•	•		
midPointAndFace	中点および界面	•	•	•	•		
curve	曲線に沿った指定された点	•	•				•
cloud	指定された点	•	•				•

入力項目	説明	オプション	
type	データ取得の型	上の一覧参照	
axis	Output of sample location	x	x 軸
		y	y 軸
		z	z 軸
		xyz	xyz 軸
		distance	点 0 からの距離
start	データ取得線の始点	e.g. (0.0 0.0 0.0)	
end	データ取得線の終点	e.g. (0.0 2.0 0.0)	
nPoints	データ取得点の数	e.g. 200	
points	データ取得点一覧		

表 6.4 *sets* サブディクショナリにおけるエントリ

キーワード	説明	オプション
basePoint	平面上の点	e.g. (0 0 0)
normalVector	平面の法線ベクトル	e.g. (1 0 0)
interpolate	補間の有無	true/false
triangulate	三角形で分割するか（オプション）	true/false

表 6.5 *surfaces* サブディクショナリにおける surfaces 用のエントリ

キーワード	説明	オプション
patchName	パッチ名	e.g. movingWall
interpolate	データ補間の有無	true/false
triangulate	三角形で分割するか（オプション）	true/false

表 6.6 *surfaces* サブディクショナリにおける patch 用のエントリ

surfaces リストは，データ取得位置のサブディクショナリのリストによって構成されます．各サブディクショナリは，表面の名前に従った名前が付けられ，type で始まる一連の記述で構成されます．平面上の点と法線ベクトルで定義され，表 6.5 に示される項目の記述がなされる plane か，または，既存の境界パッチと一致し，表 6.6 に示される項目の記述がなされる patch

のいずれかです．

6.6 ジョブのモニタと管理

本節では，まず正しく実行された OpenFOAM のジョブについて言及し，3.3 節節で説明したソルバの基本的な実行についてまで述べます．*$WM_PROJECT_DIR/etc/controlDict* ファイルの *DebugSwitches* の，`level` デバッグスイッチが 1 または 2（デフォルト）であったなら，ソルバの実行時に方程式の解の状態を標準出力，例えばスクリーンに出力します．以下では *cavity* チュートリアルを解く際の出力の冒頭部分を例として挙げています．ここから解かれる各々の方程式について，レポート行に，ソルバ名，解かれる変数，その初期と最終の残差，そして反復回数が書かれていることが読み取れます．

```
    Starting time loop

Time = 0.005

Max Courant Number = 0
BICCG: Solving for Ux, Initial residual = 1, Final residual = 2.96338e-06, No Iterations 8
ICCG: Solving for p, Initial residual = 1, Final residual = 4.9336e-07, No Iterations 35
time step continuity errors : sum local = 3.29376e-09, global = -6.41065e-20, cumulative = -6.41065e-20
ICCG: Solving for p, Initial residual = 0.47484, Final residual = 5.41068e-07, No Iterations 34
time step continuity errors : sum local = 6.60947e-09, global = -6.22619e-19, cumulative = -6.86725e-19
ExecutionTime = 0.14 s

Time = 0.01

Max Courant Number = 0.585722
BICCG: Solving for Ux, Initial residual = 0.148584, Final residual = 7.15711e-06, No Iterations 6
BICCG: Solving for Uy, Initial residual = 0.256618, Final residual = 8.94127e-06, No Iterations 6
ICCG: Solving for p, Initial residual = 0.37146, Final residual = 6.67464e-07, No Iterations 33
time step continuity errors : sum local = 6.34431e-09, global = 1.20603e-19, cumulative = -5.66122e-19
ICCG: Solving for p, Initial residual = 0.271556, Final residual = 3.69316e-07, No Iterations 33
time step continuity errors : sum local = 3.96176e-09, global = 6.9814e-20, cumulative = -4.96308e-19
ExecutionTime = 0.16 s

Time = 0.015

Max Courant Number = 0.758267
BICCG: Solving for Ux, Initial residual = 0.0448679, Final residual = 2.42301e-06, No Iterations 6
BICCG: Solving for Uy, Initial residual = 0.0782042, Final residual = 1.47009e-06, No Iterations 7
ICCG: Solving for p, Initial residual = 0.107474, Final residual = 4.8362e-07, No Iterations 32
time step continuity errors : sum local = 3.99028e-09, global = -5.69762e-19, cumulative = -1.06607e-18
ICCG: Solving for p, Initial residual = 0.0806771, Final residual = 9.47171e-07, No Iterations 31
time step continuity errors : sum local = 7.92176e-09, global = 1.07533e-19, cumulative = -9.58537e-19
ExecutionTime = 0.19 s
```

6.6.1 計算実行用の foamJob スクリプト

ユーザは，残差や反復回数，Courant 数などが，レポートデータとしてスクリーンを流れるのを確認すれば十分かもしれません．その代わりに，レポートをログファイルにリダイレクトすることで計算速度を向上させることもできます．このために foamJob スクリプトは，便利なオプションを提供しています．`<solver>`を指定して実行することで，計算がバックグラウンドで実行され，出力を *log* という名前のファイルに記録します．

```
foamJob <solver>
```

その他のオプションは，`foamJob -help` を実行することで見ることができます．*log* ファイルは，UNIX tail コマンドを用いることで見たいときに見ることができます．一般的には，follow を意味する`-f` オプションを一緒に用いることで *log* ファイルに新しいデータが記録されるのを捉えることができます．

```
tail -f log
```

6.6.2 計算モニタ用の foamLog スクリプト

log ファイルを読むことで，ジョブをモニタするには，限界があります．特に，長い期間にわたって，傾向を抽出するのは困難です．したがって，foamLog スクリプトによって残差や反復回数，Courant 数のデータを抽出し，グラフにプロットできるように一連のファイルとして出力することができます．スクリプトは次のように実行します．

```
foamLog <logFile>
```

ファイルは，ケースディレクトリの *logs* という名前のサブディレクトリの中に保存されます．各々のファイルは，*<var>_<subIter>*という名前が付けられます．ここで，`<var>` は，ログファイルの中で指定される変数の名前で，`<subIter>` は，そのタイムステップにおける繰り返し回数です．解かれた変数に対して，初期残差はその変数名`<var>`をとり，最終残差は*<var>FinalRes* という名前をとります．デフォルトでは，ファイルは，時刻と抽出された値という 2 列のフォーマットで表されます．

例として，*cavity* チュートリアルでは，解が定常状態に収束するのかを見るために，観察したいのは U_x 方程式の初期残差です．この場合，*logs/Ux_0* ファイルからデータを取り出し，図 6.5 のようにプロットします．ここでは，残差は単調に収束許容値まで減少しているのが読み取れます．

foamLog は，*log* ファイルから，うまくそれができるようにファイルを作成します．*cavity* チュートリアルの例では次のファイルがあります．

- Courant 数，*Courant_0*
- U_x 方程式の初期と最終の残差である，*Ux_0* と *UxFinalRes_0*, そして反復回数の *UxIters_0* (そしてこれと同等の U_y データ)
- 累積，全体そしてローカルの連続誤差．これは，p 方程式ごとに出す．*contCumulative_0*, *contGlobal_0*，*contLocal_0*，*contCumulative_1*，*contGlobal_1*，*contLocal_1*
- p 方程式から，残差と反復回数 *p_0*, *pFinalRes_0*，*pIters_0*，*p_1*，*pFinalRes_1*，*pIters_1*
- 実行時間，*executionTime*

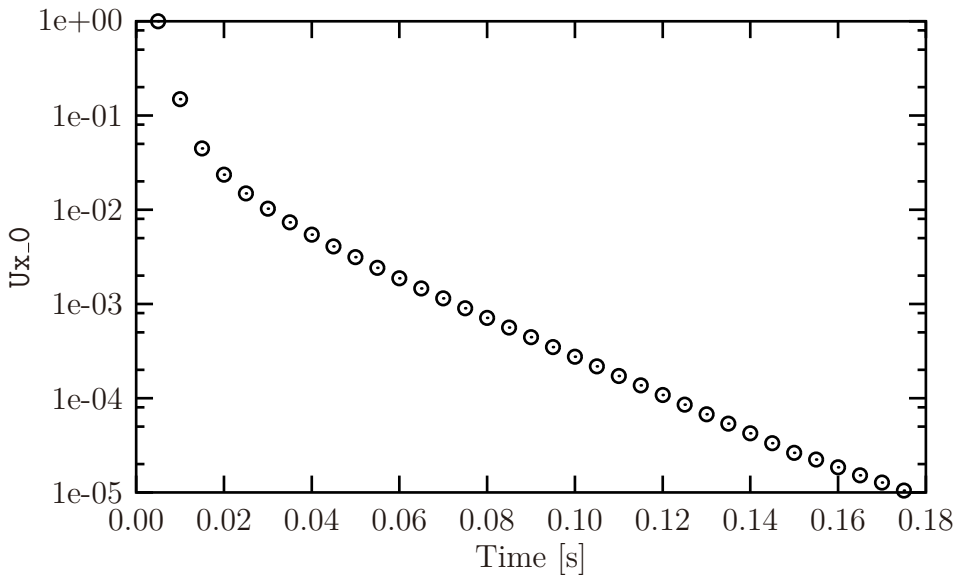


図 6.5 *cavity* チュートリアルにおける U_x の初期残差

第7章
モデルと物性値

OpenFOAM には，各々が特定の問題に特化して設計されたソルバが，幅広い範囲にわたって用意されています．ユーザは，特定のケースに対してモデリングを行う際に最初にソルバの選択ができるように，その方程式とアルゴリズムは一つ一つが異なったものとなっています．ソルバの選択には，通常，3.5 節にある各ソルバの説明に目を通して，そのケースに対して適切なソルバを見つけてください．各々のケースを定義するためには，最終的にはパラメータと物理的特性が必要となりますが，いくつかのモデリングのオプションはケースの *constant* ディレクトリの中のディクショナリに登録されている中から実行時に指定することができます．本章では，一般的なモデルと，実行時に指定される関連したプロパティについて詳しく説明します．

7.1 熱物理モデル

熱物性モデルは，エネルギ，熱および物理的な特性が関与しています．

thermophysicalProperties ディレクトリは，thermophysical モデルのライブラリを使用するすべてのソルバにより読み込まれます．熱物性モデルは，OpenFOAM の中では，その他のプロパティについても計算される圧力温度 $(p$–$T)$ システムとして構築されます．これは，シミュレーションの中で使用される完全な熱物性モデルを指定する `thermoType` と呼ばれる必須のディクショナリ登録です．熱物性のモデリングは，状態の基礎方程式を定義しているレイヤからスタートし，前のレイヤからプロパティを読込んだモデリングのレイヤを追加します．`thermoType` の名称は，表 7.1 にリストアップしているモデリングのマルチレイヤを意味しています．

状態方程式 — equationOfState	
adiabaticPerfectFluid	断熱完全気体の状態方程式
icoPolynomial	液体などの非圧縮性流体に対する多項式の状態方程式
perfectFluid	完全気体の状態方程式
incompressiblePerfectGas	一定の参照圧力を用いた非圧縮性気体の状態方程式．密度は温度と組成によってのみ変化する．
rhoConst	密度を一定とした状態方程式

標準熱特性 — thermo	
eConstThermo	内部エネルギ e とエントロピ s の評価を備えた，比熱 c_p 一定のモデル
hConstThermo	エンタルピ h とエントロピ s の評価を備えた，比熱 c_p 一定のモデル
hPolynomialThermo	h と s を評価する多項式の係数の関数により c_p が評価される
janafThermo	JANAF の熱物性表の係数から c_p が評価され，それにより h，s が評価される．

派生熱特性 — specieThermo	
specieThermo	c_p，h，そして/または，s から得られた特殊な熱特性

輸送特性 — transport	
constTransport	一定の輸送特性
polynomialTransport	多項式に基づく温度依存輸送特性
sutherlandTransport	温度依存する輸送輸送のための Sutherland の公式

混合特性 — mixture	
pureMixture	不活性混合気体の一般熱物理モデル計算
homogeneousMixture	正規化燃料質量分率 b に基づく混合気燃焼
inhomogeneousMixture	b と総燃料質量分率 f_{t} に基づく混合気燃焼
veryInhomogeneousMixture	b と f_{t} と未燃燃料質量分率 f_{u} に基づく混合気燃焼
basicMultiComponentMixture	複数の成分に基づく基本的な混合気
multiComponentMixture	複数の成分に基づく派生的な混合気
reactingMixture	熱力学と反応スキームを用いた混合気燃焼
egrMixture	排気ガス再循環の混合気
singleStepReactingMixture	素反応を伴う混合気

熱モデル — thermoModel	
hePsiThermo	圧縮率 ψ に基づく一般熱物理モデル計算
heRhoThermo	密度 ρ に基づく一般熱物理モデル計算
psiReactionThermo	ψ に基づいて燃焼混合気のエンタルピを計算する
psiuReactionThermo	ψ_{u} に基づいて燃焼混合気のエンタルピを計算する
rhoReactionThermo	ρ に基づいて燃焼混合気のエンタルピを計算する
heheupsiReactionThermo	未燃ガスおよび燃焼混合気のエンタルピを計算する

表 7.1　熱物性モデリングの階層

以下は `thermoType` のエントリの一例です．

```
thermoType
{
    type            hePsiThermo;
    mixture         pureMixture;
    transport       const;
    thermo          hConst;
    equationOfState perfectGas;
    specie          specie;
    energy          sensibleEnthalpy;
}
```

このキーワード・エントリは，例えば輸送 (`transport`) を定数（粘性と熱拡散を定数）として，理想気体の状態方程式 (`equationOfState`) を使うといった，熱物理モデルの選択を示しています．それに加えて `energy` というキーワード・エントリがあり，解析におけるエネルギの扱い方を指定することができます．エネルギとしては，内部エネルギかエンタルピ，そして生成熱 Δh_{f} を含めるかどうかを選択できます．生成熱を含める場合は**絶対**エネルギを使いますが，そうでない場合は**顕在**エネルギを使います．例えば，絶対エネルギ h と顕在エネルギ h_{s} の関係は以下のようになります．

$$h = h_{\mathrm{s}} + \sum_i c_i \Delta h_{\mathrm{f}}^i \tag{7.1}$$

ここで c_i と h_{f}^i は，それぞれ化学種 i のモル比と生成熱です．ほとんどのケースにおいて，反応によるエネルギ変化を扱いやすいように，顕在エネルギを使います．`energy` に対するキーワード・エントリは，例えば `sensibleEnthalpy`，`sensibleInternalEnergy`，そして `absoluteEnthalpy` などがあります．

7.1.1 熱物性データ

基本的な熱物性値は，各種類ごとに入力データに指定します．データエントリには，O2，H2O，mixture といった物質名を示すキーワードに続けて，以下のような係数のサブディクショナリを入力する必要があります．

specie　その物質のモル数 nMoles，およびモル質量 *molWeight を g/mol の単位で入力します．
thermodynamics　選択した熱物理モデル（後述）に対する係数を入力します．
transport　選択した輸送モデル（後述）に対する係数を入力します．

熱力学係数は見かけ上，比熱 c_p の評価に関与し，そこから他の物性値が導出されます．現在の thermo モデルは，以下に示すとおりです．

hConstThermo　c_p と融解熱 H_f を定数と仮定します．単純に Cp および Hf というキーワードで二つの値 c_p と H_f を指定します．
eConstThermo　c_v と融解熱 H_f を定数と仮定します．単純に Cv および Hf というキーワードで二つの値 c_v と H_f を指定します．
janafThermo　熱力学の JANAF 表から得られた一連の係数により，c_p を温度の関数として計算します．順に並べた係数のリストを表 7.2 に示します．関数は，温度の下限 T_l と上限 T_h の間で妥当性が確認されています．係数は二組示されています．最初の組は常温 T_c 以上 (そして T_h 以下) の温度についてのものであり，二組目は T_c より以下 (そして T_l 以上) の範囲のものです．c_p を温度の関数として表すと，

$$c_p = R((((a_4T + a_3)T + a_2)T + a_1)T + a_0) \tag{7.2}$$

これに加えて，高温と低温の両方に a_5，a_6 という積分定数があります．これらは，それぞれ h と s を評価するために使われます．
hPolynomialThermo　c_p を温度の関数として，任意次数の多項式によって計算します．次のケースにその使用例があります：

$FOAM_TUTORIALS/lagrangian/porousExplicitSourceReactingParcelFoam/filter

説明	入力	キーワード
下限温度	T_l (K)	Tlow
上限温度	T_h (K)	Thigh
常温	T_c (K)	Tcommon
高温度係数	$a_0 \ldots a_4$	highCpCoeffs (a0 a1 a2 a3 a4...
高温度エンタルピ補正	a_5	a5...
高温度エントロピ補正	a_6	a6)
低温度係数	$a_0 \ldots a_4$	lowCpCoeffs (a0 a1 a2 a3 a4...
低温度エンタルピ補正	a_5	a5...
低温度エントロピ補正	a_6	a6)

表 7.2　JANAF の熱力学係数

輸送係数は，粘性係数 †μ，熱伝導率 κ，層流熱伝導率（エンタルピ方程式のため）α を評価するために使われます．現在の transport モデルは，以下に説明するとおりです．

*訳注：原文では molecular weight となっているので直訳すれば「モル重量」だが，単位が g/mol とされているので「モル質量」とした．
†訳注：原文には dynamic viscosity とあるが，文脈からして動粘性係数ではない．

constTransport　μ と Prandtl 数 $Pr = c_p\mu/\kappa$ が一定であると仮定します．それぞれキーワード `mu` および `Pr` によって指定します．

sutherlandTransport　μ を温度 T の関数として計算します．これには，Sutherland 係数 A_S と Sutherland 温度 T_S が用いられ，キーワード `As` および `Ts` によって指定します．μ は，次のように計算されます．

$$\mu = \frac{A_S\sqrt{T}}{1 + T_S/T} \tag{7.3}$$

polynomialTransport　μ と κ を温度 T の関数として，任意次数の多項式から計算します．

次は，`fuel` という名前の種についてのエントリの例です．これは，sutherlandTransport と janafThermo を使ってモデル化されています．

```
fuel
{
    specie
    {
        nMoles       1;
        molWeight    16.0428;
    }
    thermodynamics
    {
        Tlow         200;
        Thigh        6000;
        Tcommon      1000;
        highCpCoeffs (1.63543 0.0100844 -3.36924e-06 5.34973e-10
                      -3.15528e-14 -10005.6 9.9937);
        lowCpCoeffs  (5.14988 -0.013671 4.91801e-05 -4.84744e-08
                      1.66694e-11 -10246.6 -4.64132);
    }
    transport
    {
        As           1.67212e-06;
        Ts           170.672;
    }
}
```

次に示すのは，`air` という名前の物質についての，エントリの例です．これは，constTransport と hConstThermo でモデル化されています．

```
air
{
    specie
    {
        nMoles          1;
        molWeight       28.96;
    }
    thermodynamics
    {
        Cp              1004.5;
        Hf              2.544e+06;
    }
    transport
    {
        mu              1.8e-05;
        Pr              0.7;
    }
}
```

7.2 乱流モデル

乱流のモデリングを含むあらゆるソルバは *turbulenceProperties* ディクショナリを読み込みます．このファイルの中では，`simulationType` キーワードで使用する乱流モデルとして次のいずれかを選択します．

laminar　乱流モデルを使用しない
RASModel　Reynolds 平均応力 (RAS) モデル
LESModel　ラージ・エディ・シミュレーション (LES) モデル

`RASModel` が選択されているとき，RAS モデリングの選択は，同じく *constant* ディレクトリにある *RASProperties* ファイルで設定します．RAS 乱流モデルは，表 3.9 に示した利用可能なモデルの長いリストから，`RASModel` エントリで選択します．同様に，`LESModel` が選択された場合，LES モデリングの詳細は *LESProperties* ディクショナリで記述し，LES 乱流モデルは `LESModel` エントリで選択します．*RASProperties* に必要なエントリは，表 7.3 に，また *LESProperties* ディクショナリについては表 7.4 に示します．

`RASModel`	RAS モデルの名前
`turbulence`	乱流モデルの on/off スイッチ
`printCoeffs`	シミュレーション開始時にモデル係数をターミナルに出力するスイッチ
`<RASModel>Coeffs`	各 RASModel における係数のディクショナリ（省略可能）

表 7.3　*RASProperties* ディクショナリにおけるキーワードエントリ

`LESModel`	LES モデルの名前
`delta`	デルタモデルの名前
`<LESModel>Coeffs`	対応する LES モデルにおける係数のディクショナリ
`<delta>Coeffs`	各デルタモデルにおける係数ディクショナリ

表 7.4　*LESProperties* ディクショナリにおけるキーワードエントリ

非圧縮性および圧縮性の RAS 乱流モデル，等容変化および非等容変化 LES モデル，そしてデルタモデルは，表 3.9 に示しています．これらの使用例は *$FOAM_TUTORIALS* 以下に見つかります．

7.2.1 モデル係数

RAS モデルの係数には，それぞれのソースコードの中でデフォルト値が与えられています．もしこのデフォルト値を上書きしたければ，モデル名に `Coeffs` を加えたキーワード名（たとえば `kEpsilon` モデルなら `kEpsilonCoeffs`）のサブディクショナリを，*RASProperties* ファイルに追加することで実現できます．もし *RASProperties* ファイルで `printCoeffs` スイッチが `on` になっていれば，計算開始時にモデルが作成されたときに，該当する `...Coeffs` ディクショナリの例が標準出力に表示されます．ユーザは，これを *RASProperties* にコピーして，必要に応じて数値を変更すればよいでしょう．

7.2.2　壁関数

OpenFOAM では，個別のパッチの境界条件として適用する，様々な壁関数が利用可能になっています．これにより，異なる壁領域に異なる壁関数モデルを適用することが可能になります．壁関数モデルの選択は，非圧縮性 RAS においては *0/nut* ファイルの ν_t，圧縮性 RAS においては *0/mut* ファイルの μ_t，非圧縮性 LES においては *0/nuSgs* ファイルの ν_{sgs}，圧縮性 LES においては *0/muSgs* ファイルの μ_{sgs} によって指定します．たとえば，ある *0/nut* ファイルは，

```
17
18 dimensions      [0 2 -1 0 0 0 0];
19
20 internalField   uniform 0;
21
22 boundaryField
23 {
24     movingWall
25     {
26         type            nutkWallFunction;
27         value           uniform 0;
28     }
29     fixedWalls
30     {
31         type            nutkWallFunction;
32         value           uniform 0;
33     }
34     frontAndBack
35     {
36         type            empty;
37     }
38 }
39
40
41 // ************************************************************************* //
```

本リリースでは様々な壁関数モデルが利用できます．たとえば，`nutWallFunction`，`nutRoughWallFunction`，`nutSpalartAllmarasStandardRoughWallFunction`，`nutSpalartAllmarasStandardWallFunction`，そして `nutSpalartAllmarasWallFunction`．ユーザは，当該ディレクトリから，すべての壁関数モデルのリストを参照できます．

```
find $FOAM_SRC/turbulenceModels -name wallFunctions
```

それぞれの壁関数境界条件では，`E`，`kappa`，そして `Cmu` というオプションのキーワードエントリで E，κ，そして C_μ のデフォルト値を上書きできます．*nut* や *mut* ファイルのいずれかのパッチで壁関数を選択したならば，*epsilon* フィールドの対応するパッチでは `epsilonWallFunction` を，乱流場 k，q，R の対応するパッチには `kqRwallFunction` を選択する必要があります．

索引

記号・数字

A

B

D

E

F

G

H

I

J

K

L

M

N

Q

R

S

T

U

V

W

さ

た

ま

や