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 1.5

2010 年 3 月 1 日

OpenFOAM ユーザー会

一般社団法人 オープン CAE 学会

OpenFOAM ユーザー会
一般社団法人 オープン CAE 学会

Typeset in pLaTeX.

GNU Free Documentation License

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The **"Document"**, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as **"you"**. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A **"Modified Version"** of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A **"Secondary Section"** is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document' s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The **"Invariant Sections"** are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The **"Cover Texts"** are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A **"Transparent"** copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called **"Opaque"**.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machinegenerated HTML, PostScript or PDF produced by some word processors for output purposes only.

The **"Title Page"** means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work' s title, preceding the beginning of the body of the text.

A section **"Entitled XYZ"** means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as **"Acknowledgements"**, **"Dedications"**, **"Endorsements"**, or **"History"**.) To **"Preserve the Title"** of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document' s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification

of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document' s license notice.
H. Include an unaltered copy of this License.
I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version' s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under

this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but

may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

Trademarks

ANSYS is a registered trademark of ANSYS Inc.
CFX is a registered trademark of AEA Technology Engineering Software Ltd.
CHEMKIN is a registered trademark of Sandia National Laboratories
CORBA is a registered trademark of Object Management Group Inc.
openDX is a registered trademark of International Business Machines Corporation
EnSight is a registered trademark of Computational Engineering International Ltd.
AVS/Express is a registered trademark of Advanced Visual Systems Inc.
Fluent is a registered trademark of Fluent Inc.
GAMBIT is a registered trademark of Fluent Inc.
Fieldview is a registered trademark of Intelligent Light
Icem-CFD is a registered trademark of ICEM Technologies GmbH
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
MICO is a registered trademark of MICO Inc.
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

OpenFOAM® is a registered trademark of OpenCFD Ltd.

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 1.5 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 2 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．

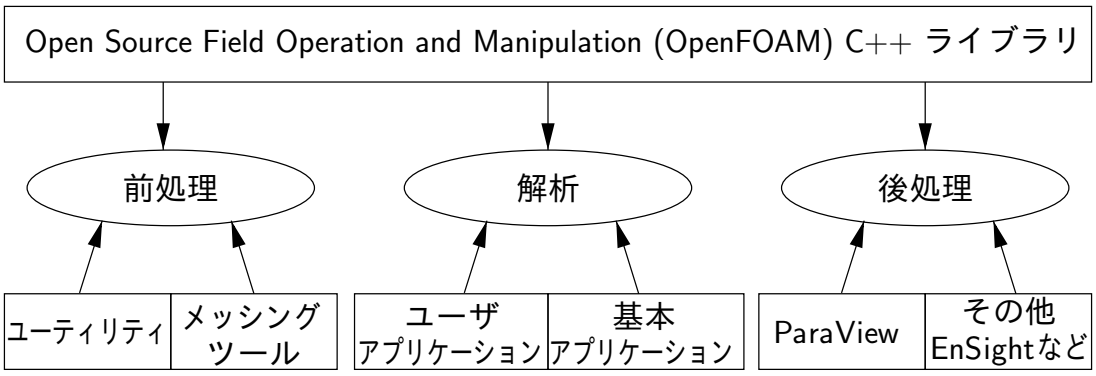


図 1.1　OpenFOAM の全体的な構造

前処理や OpenFOAM のケースの実行方法については，第 4 章と第 5 章で説明しますが，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．

OpenFOAM の開発にあたっては，いくつかの機能の開発を中断せざるをえない場合があるのですが，それは，ユーザが個人的に修正したバージョンの OpenFOAM で，それらの機能のメンテナンスをしていたことが原因かもしれません．そのような機能に関するドキュメントはメンテナンスされま

せんが，このガイドの付録でみられます．特に，バージョン 1.5 で開発が中断された FoamX に関するドキュメントは付録 A にあります．

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で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．チュートリアルはソルバのサブディレクトリに組み込まれています．例えば icoFoam のすべてのケースはサブディレクトリの *icoFoam* 内にあります．ユーザには *tutorials* の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強く勧めます．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，プリプロセス，計算，ポストプロセス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であるジオメトリを示します．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．これにより流れのレイノルズ数を増加させ，turbFoam ソルバを乱流，等温，非圧縮性流れに使用します．

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．それは，入出力が初心者でもわかりやすいキーワードをもつディクショナリ形式が使われているからです．以前のバージョンでは FoamX という GUI ケースエディタがありましたが，利用者がエディタによって編集できるファイル

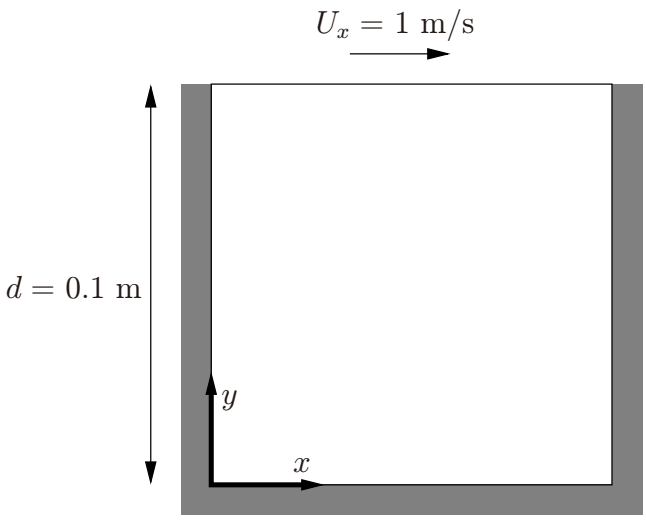


図 2.1　天井駆動キャビティのジオメトリ

を好んでおり，メンテナンスをあまりしないためバージョン 1.5 ではなくなりました．

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/icoFoam/cavity
```

2.1.1.1　メッシュ生成

OpenFOAM は常に 3 次元デカルト座標系で動くため，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．

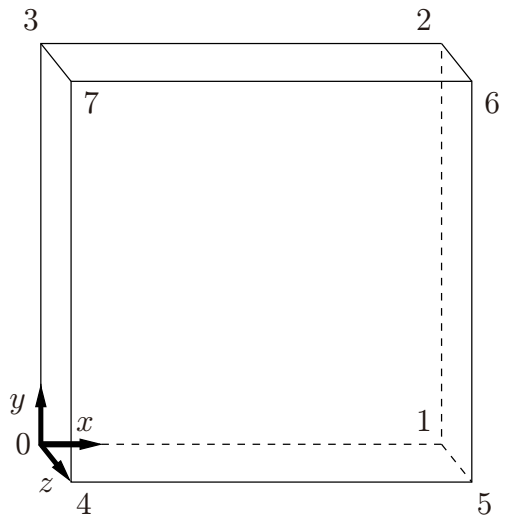


図 2.2　キャビティのメッシュのブロック構造

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．

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.5                                   |
|   \\  /    A nd           | Web:      http://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
(
);

patches
(
    wall movingWall
    (
        (3 7 6 2)
    )
    wall fixedWalls
    (
        (0 4 7 3)
        (2 6 5 1)
        (1 5 4 0)
    )
    empty frontAndBack
    (
        (0 3 2 1)
        (4 5 6 7)
    )
);

mergePatchPairs
(
);

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

ファイルの最初はバナー（1–7 行）形式のヘッダ情報で，ファイル情報は，波括弧 ({...}) で囲ま

れる *FoamFile* サブディクショナリの中で記述されます．

今後は，簡便化とスペースの都合上，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ケースファイルの引用の際に省きます．

まずファイルは初めにブロックの頂点の座標 `vertices` を指定します．それに続き，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す．

```
blockMesh
```

blockMesh の実行状況はターミナルウイ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．

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

internalField   uniform 0;

boundaryField
{
    movingWall
    {
        type            zeroGradient;
    }

    fixedWalls
    {
        type            zeroGradient;
    }

    frontAndBack
    {
        type            empty;
    }
}

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

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　物理量の次元を定義．ここでは動圧，つまり $\mathrm{m^2 s^{-2}}$（4.2.6 項に詳述）．

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，二つのパッチが使用されています．(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．`fixedWall` に関してはすべりなしのため `value` は `uniform (0 0 0)` となります．上面は 1 m/s で移動するので `uniform (1 0 0)` で固定値を設定します．

2.1.1.3 物理量

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エントリを見たり，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` (方程式で見られるギリシア語シンボル ν の音声ラベル) というキーワードになります．まず最初に，このケースはレイノルズ数を 10 で計算します．レイノルズ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．

```
nu              nu [0 2 -1 0 0 0 0] 0.01;

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

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 未満のクーラン数が必要です．クーラン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 以下のクーラン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 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．

```
application icoFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         0.5;

deltaT          0.005;

writeControl    timeStep;

writeInterval   20;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;
```

```
writeCompression uncompressed;

timeFormat      general;

timePrecision   6;

runTimeModifiable yes;

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

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/icoFoam/cavity
```

図 6.1 に示すように ParaView のウインドウが開きます．Pipeline Browser を見ると，ParaView が `cavity.foam`，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Region Status と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 Region Status 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 タブ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．

6.1.5 項で述べるように視点操作を試してみましょう．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．詳しくは 6.1.5.1 で述べます．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．

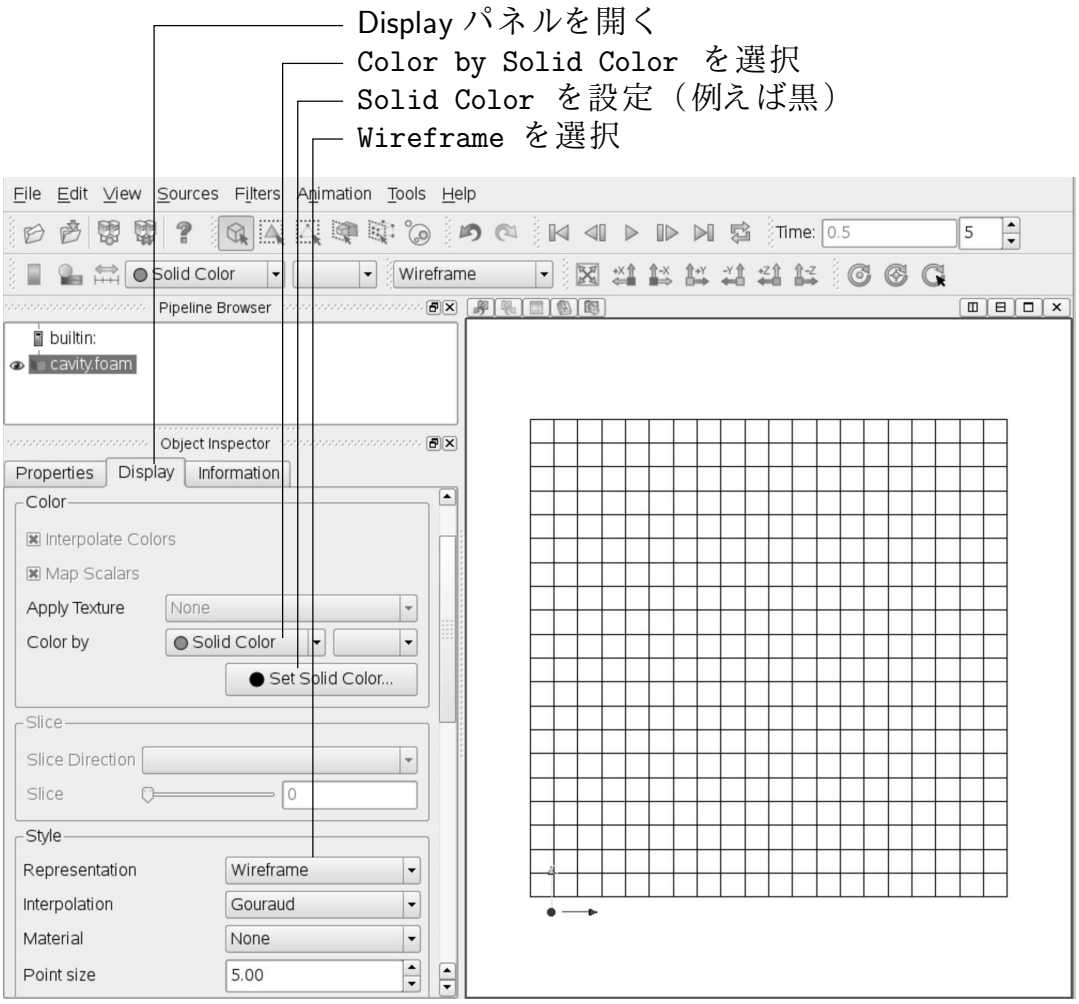


図 2.3　paraFoam でのメッシュの表示

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/icoFoam/cavity
```

ジョブの進捗は，ターミナルウインドウに表示されます．現在の時刻，最大クーラン数，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 ウインドウに戻って，`cavity.foam` ケースモジュールのために Properties パネルを選んでください．ケースモジュールのためのパネルが存在していないようならば，`cavity.foam` が黄色でハイライトされているか，それと並んだ目のボタンに表示が有効であることを示すスイッチは入っているか，フォームの View メニュー中で Source が選ばれているか，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 画面で Update GUI を選択し，緑の Apply ボタンをクリックします． そうすれば時間データはロードされます．

2.1.4.1 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 StylePanel 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 window 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 ウインドウに置くことも可能です．

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．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．

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

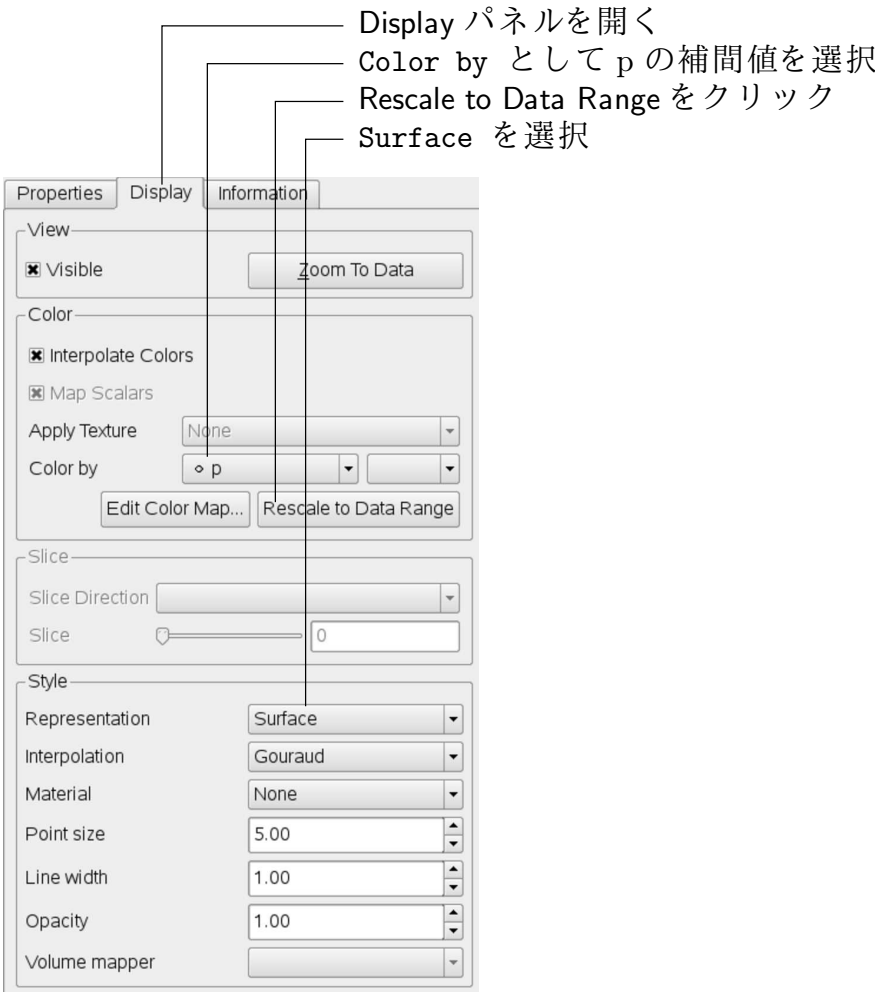


図 2.4　キャビティケースでの圧力等圧線の描画

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 メニューから Cell Centers を選択して Apply をクリックします．

Centers が強調表示され，Filter メニューから Glyph を選択します．図 2.6 のような Properties ウインドウが表示されます．Properties パネルの結果では vectors メニューはベクトル場は速度のみなので速度場 U が選択されています．Scale Mode は速度の Vector Magnitude が初期値として選択されていますが，off を選択し，Set Scale Factor に 0.005 をにして全体のの速度を見ます．Apply をクリックすると単色，例えば白，のベクトルが表示されます．通常は Display パネルで `Color by U` を選択して大きさに応じた色付けをします．`Edit Color Map` を Show Color Legend に設定し，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 View Settings の General Panel で白に設定されています．

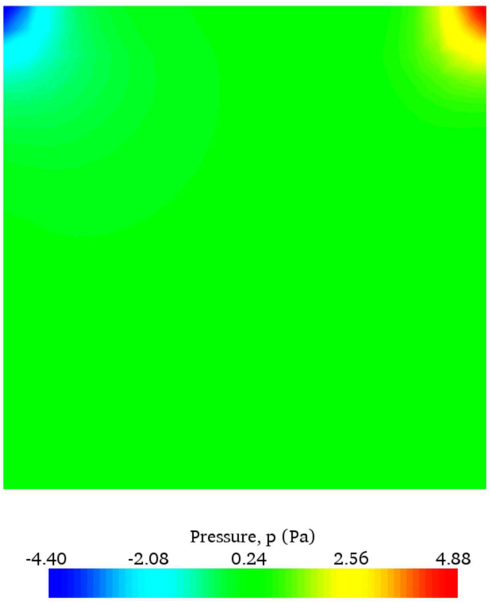


図 2.5　キャビティケースでの圧力

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.foam` 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 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，Init.Step Len 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を使いました．

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．

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/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．

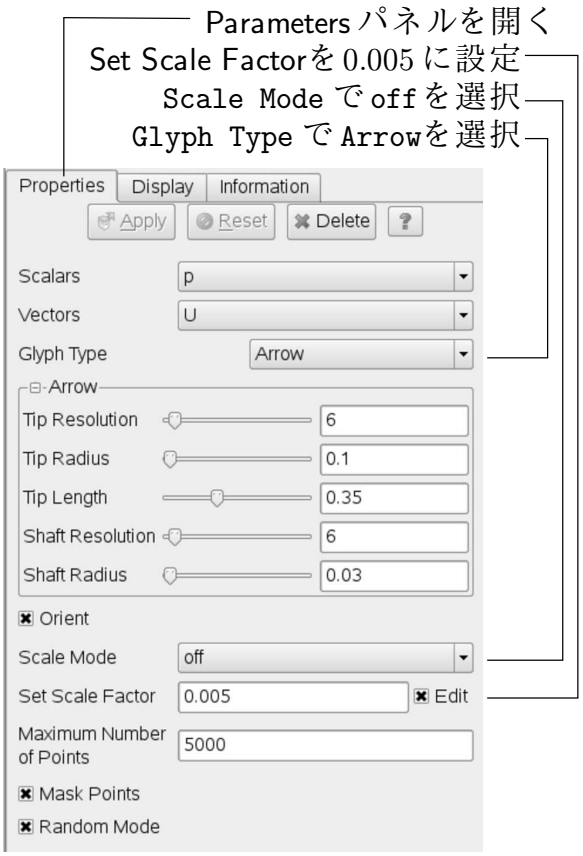


図 2.6　Glyph フィルタのパラメータパネル

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

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

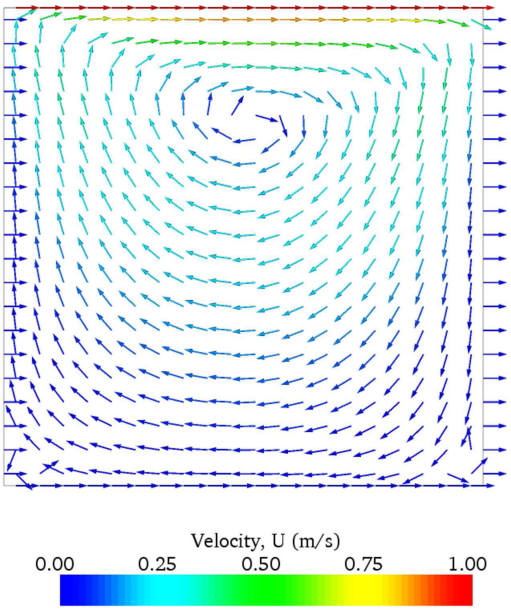


図 2.7　キャビティケースの速度

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．

n

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

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: 1681

Consistently creating and mapping fields for time 0.5

    interpolating p
    interpolating U

End
```

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 より小さいクーラン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

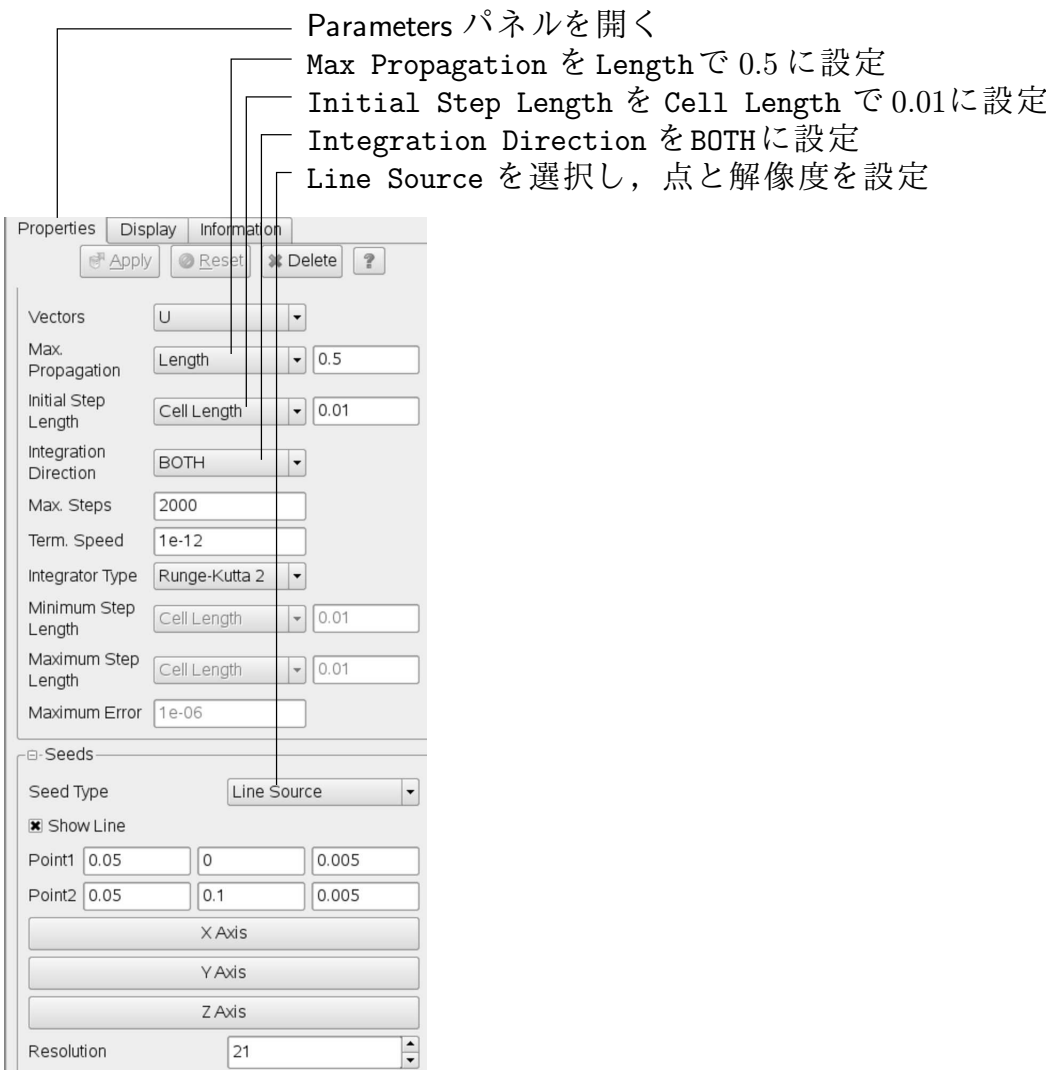


図 2.8　Stream Tracer フィルタのパラメータパネル

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 s 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
```

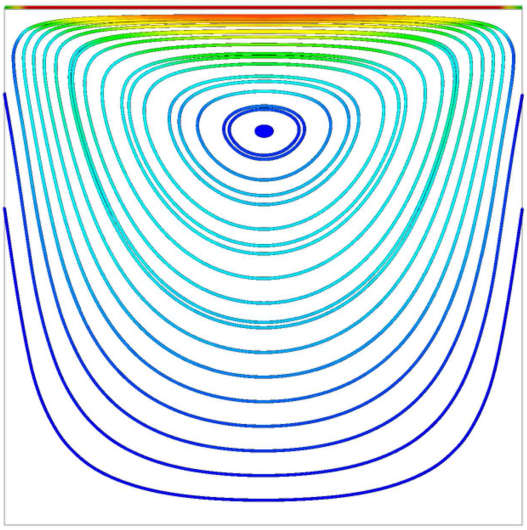


図 2.9 キャビティケースの流線

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 と呼ばれています．

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/icoFoam
touch cavityFine/cavityFine.foam
```

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.foam` 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．

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 `div`，`components`，`mag`，`magGrad`，`magSqr` 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

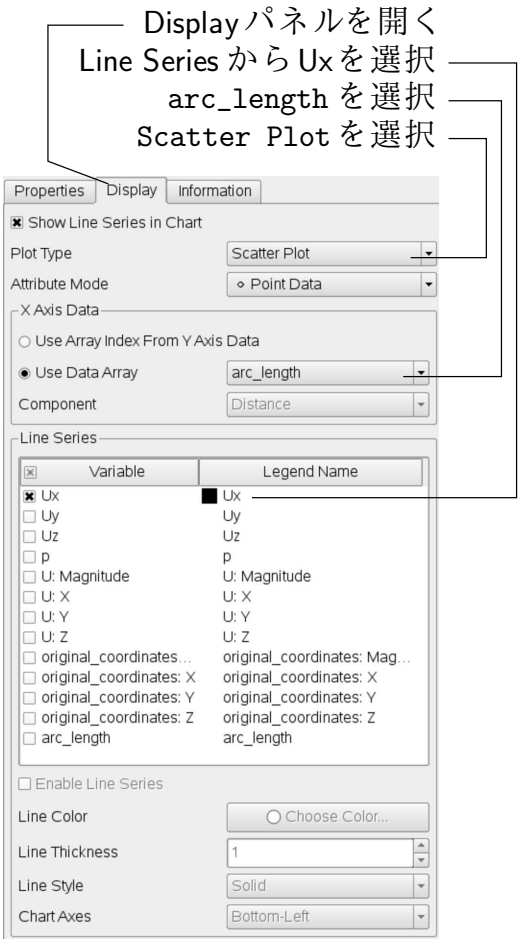


図 2.10　グラフ作図のためのフィールド選択

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:


(
div
components
mag
magGrad
magSqr
)
```

comonents 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．例として次のように入力し cavity に対して実行します．

```
foamCalc components U -case $FOAM_RUN/tutorials1.5/icoFoam/cavity
```

それぞれのコンポーネントが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，2.2.3項または6.5節で述べる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．作業をしている基本のモジュール，この場合`cavity.foam`のPropertiesパネルの上部にあるUpdate GUIボタンをチェックしします．Applyをクリックすることでで Vol Field Statusウインドウに新しいデータが読み込まれます．新しいデータの選択と変更の適用を確認し，必要ならApplyをもう一度クリックします．境界領域がRegion Statusパネルで選択されていると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`movingwall`や`fixedwall`，`frontAndBack`といったRegion Statusパネルのパッチの選択を解除して変更を適用します．

さて，ParaViewの図を描画しましょう．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ウインドウの傍に新しいXY Plotウインドウが開きます．Propertiesウイ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ウインドウに図が描画されます．Displayパネルの`Plot Type`をScatter Plotに設定し，Attribute Mode Point Dataにします．Use Data Arrayオプションの`arc_length`でX軸データがキャビティの底からの距離になります．

Displaysウインドウの 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．

グラフを初期化するにはXY Plot自体を動かします．カーソルがグラフ上にある状態で右クリックしてメニューからPropertiesを選択します．各軸のタイトルや形式の設定をするChart Optionsウインドウが表示されます．各軸のメニューはダブルクリックをしてLayout and Tytleにすることで拡大することができます．フォントや色，軸名の位置，軸の値の範囲や線形か対数かといった設定が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．

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

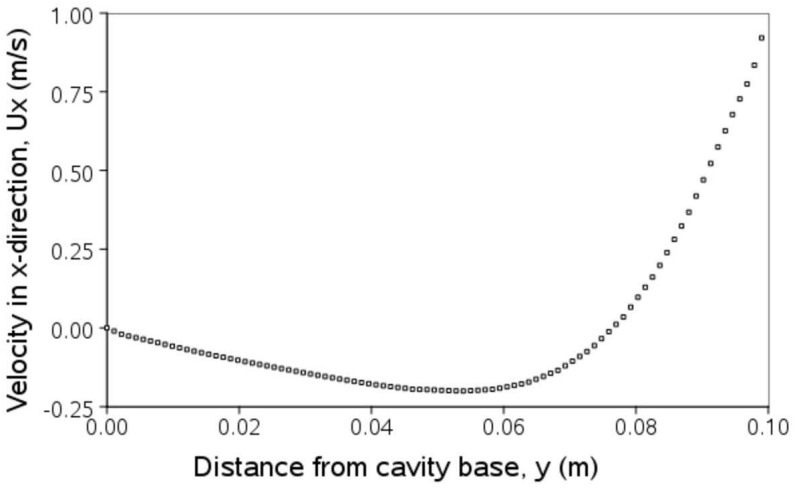


図 2.11　paraFoam でのグラフ作図

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/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．

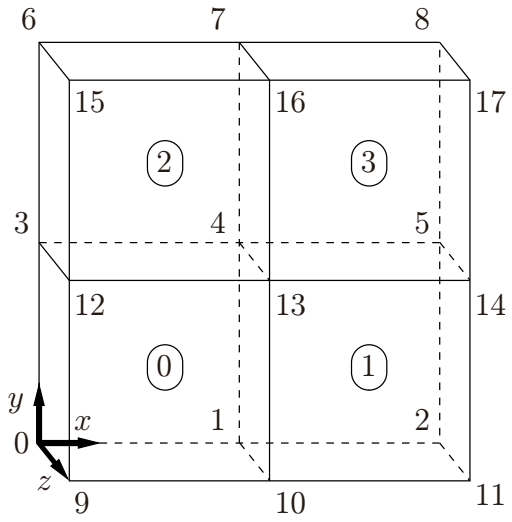


図 2.12　キャビティケースの勾配メッシュのブロック構造（ブロック番号）

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．

```
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
(
);

patches
(
    wall movingWall
    (
        (6 15 16 7)
        (7 16 17 8)
    )
    wall fixedWalls
    (
        (3 12 15 6)
        (0 9 12 3)
        (0 1 10 9)
        (1 2 11 10)
        (2 5 14 11)
        (5 8 17 14)
    )
    empty frontAndBack
    (
        (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

もっとも速い速度と小さいセルが上蓋に面することになり，したがって，2.1.1.4 で示したように，もっとも高いクーラン数が上蓋に面するセルに生じます．このようなことから上蓋に面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，

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

cavityGrade ケースにおいては，各方向のセルの数は 10 であり，もっとも大きなセルと小さなセルとの比は 2，ブロックの縦横は 0.05 m です．したがって，もっとも小さなセルサイズは 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 まで短くし，`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/icoFoam/cavityGrade
mapFields ../cavityFine -consistent
```

今度は，ケースディレクトリから icoFoam を実行します．そして，ランタイム情報をモニタリングします．そして，このケースの完全に収束した結果を見て，以前に 2.1.5.6 と 2.1.5.7 で説明した後処理ツールを使って他の結果と比較します．

2.1.7 レイノルズ数の増大

これまで解いたケースはレイノルズ数が 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，ここでレイノルズ数を 50 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/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．レイノルズ数を 10 倍に増加させるためには，動粘性係数を 10 分の 1 まで減らす必要があります（例えば 10 から）．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/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 = 1.63

Courant Number mean: 0.108642 max: 0.818175
DILUPBiCG:  Solving for Ux, Initial residual = 7.86044e-06, Final residual = 7.86044e-06,
No Iterations 0
DILUPBiCG:  Solving for Uy, Initial residual = 9.4171e-06, Final residual = 9.4171e-06,
No Iterations 0
DICPCG:  Solving for p, Initial residual = 3.54721e-06, Final residual = 7.13506e-07,
No Iterations 4
time step continuity errors : sum local = 6.46788e-09, global = -9.44516e-19,
cumulative = 1.04595e-17
DICPCG:  Solving for p, Initial residual = 2.15824e-06, Final residual = 9.95068e-07,
No Iterations 3
time step continuity errors : sum local = 8.67501e-09, global = 7.54182e-19,
cumulative = 1.12136e-17
ExecutionTime = 1.02 s  ClockTime = 1 s

Time = 1.635

Courant Number mean: 0.108643 max: 0.818176
DILUPBiCG:  Solving for Ux, Initial residual = 7.6728e-06, Final residual = 7.6728e-06,
No Iterations 0
DILUPBiCG:  Solving for Uy, Initial residual = 9.19442e-06, Final residual = 9.19442e-06,
No Iterations 0
DICPCG:  Solving for p, Initial residual = 3.13107e-06, Final residual = 8.60504e-07,
No Iterations 4
time step continuity errors : sum local = 8.15435e-09, global = -5.84817e-20,
cumulative = 1.11552e-17
DICPCG:  Solving for p, Initial residual = 2.16689e-06, Final residual = 5.27197e-07,
No Iterations 14
time step continuity errors : sum local = 3.45666e-09, global = -5.62297e-19,
cumulative = 1.05929e-17
ExecutionTime = 1.02 s  ClockTime = 1 s
```

2.1.8　高レイノルズ数流れ

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レイノルズ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，レイノルズ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レイノルズ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，レイノルズ平均応力 (RAS) 乱流モデルが平均流れの挙動を解くのに用いられ，ゆらぎ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（レイノルズ数 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 turbFoam 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/turbFoam 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．

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 `zeroGradient` 境界条件を，k については `fixedValue 0` を指定します．いかにして k と ε の初期値を決めるでしょうか．しかし，k，ε については，解法アルゴリズムにてゼロ除算を避けるために，非ゼロの値を与えます．

k，ε の適当な初期条件は，速度変動 $\boldsymbol{U}'$ と乱流長さスケール l を用いて設定することができ，次式に示すように表されます．

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

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

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．デカルト座標系では 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 方向速度ゆらぎ成分です．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．

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．レイノルズ数 10^4 を実現するために，式 (2.1) のレイノルズ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．

RASProperties ディクショナリを開き，RAS 乱流モデルを選択します．乱流モデルは `RASModel` エントリで選択されます．表 3.9 に示すように多くの使用可能なモデルが与えられています．ユーザは標準 k–ε モデルを表す `kEpsilon` をここでは選択します．そして，`turbulence` のスイッチを `on` にします．乱流モデルに関する係数は標準ディクショナリの `kEpsilonCoeffs` 以下に，また，同ディクショナリに `wallFunctionCoeffs` の設定もあります．

次いで，*controlDict* の `startTime`，`stopTime`，`deltaT`，そして `writeInterval` を設定します．クーラン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，`turbFoam` とタイプすることで turbFoam 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．したがって最大クーラン数は 0.2 以上にはなりません．クーラン数がより 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，turbFoam 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．

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
convertToMeters 0.1;

vertices
(
    (0 0 0)
    (0.6 0 0)
    (0 0.4 0)
    (0.6 0.4 0)
    (1 0.4 0)
    (0 1 0)
    (0.6 1 0)
    (1 1 0)
    (0 0 0.1)
    (0.6 0 0.1)
    (0 0.4 0.1)
    (0.6 0.4 0.1)
    (1 0.4 0.1)
    (0 1 0.1)
    (0.6 1 0.1)
    (1 1 0.1)
);

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

edges
(
);

patches
```

```
(
    wall lid
    (
        (5 13 14 6)
        (6 14 15 7)
    )
    wall fixedWalls
    (
        (0 8 10 2)
        (2 10 13 5)
        (7 15 12 4)
        (4 12 11 3)
        (3 11 9 1)
        (1 9 8 0)
    )
    empty frontAndBack
    (
        (0 2 3 1)
        (2 5 6 3)
        (3 6 7 4)
        (8 9 11 10)
        (10 11 14 13)
        (11 12 15 14)
    )
);

mergePatchPairs
(
);
// ************************************************************************* //
```

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 upper wall 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/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
);
```

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．

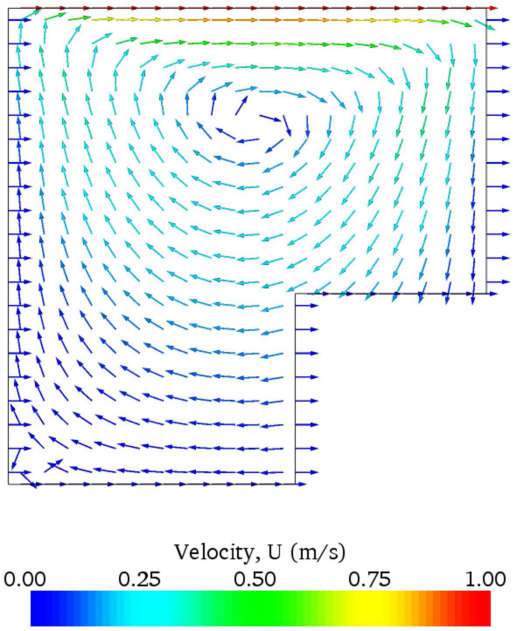


図 2.13　cavity ケースで解いた速度場を cavityClipped 上にマッピングした図

```
cuttingPatches
(
  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 を実行すればよいです．

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，これはは 2 次元のケースでは少し注意が必要です．ユーザは Filter menu から Extract Parts を選択することができ，Parameter panel にて，興味のあるパッチ，すなわち，`lid` と `fixedWalls` を，ハイライトすることができます．Accept ボタンをクリックすると，形状のこれらのアイテムは，ディスプレイ・パネルに表面ワイヤーフレームの選択により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

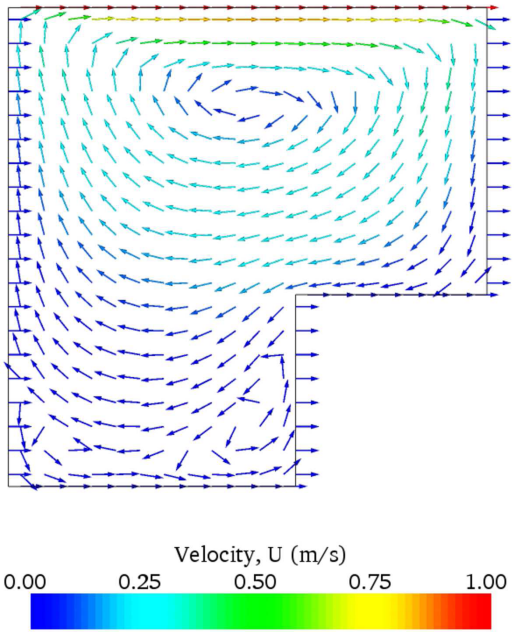


図 2.14 速度場の cavityClipped の解法

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．

本問題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化されます．デカルト座標系においては，3 次元構造の振舞いについて (1) 平面応力条件：本条件においては 2 次元の面外方向にはたらく応力成分は無視できるものと仮定される，(2) 平面ひずみ条件：本条件においては 2 次元の面外方向のひずみ成分は無視できるものと仮定される，のふたつの仮定が考えられます．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．垂直の対称面[1]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

[1] 訳注：の面

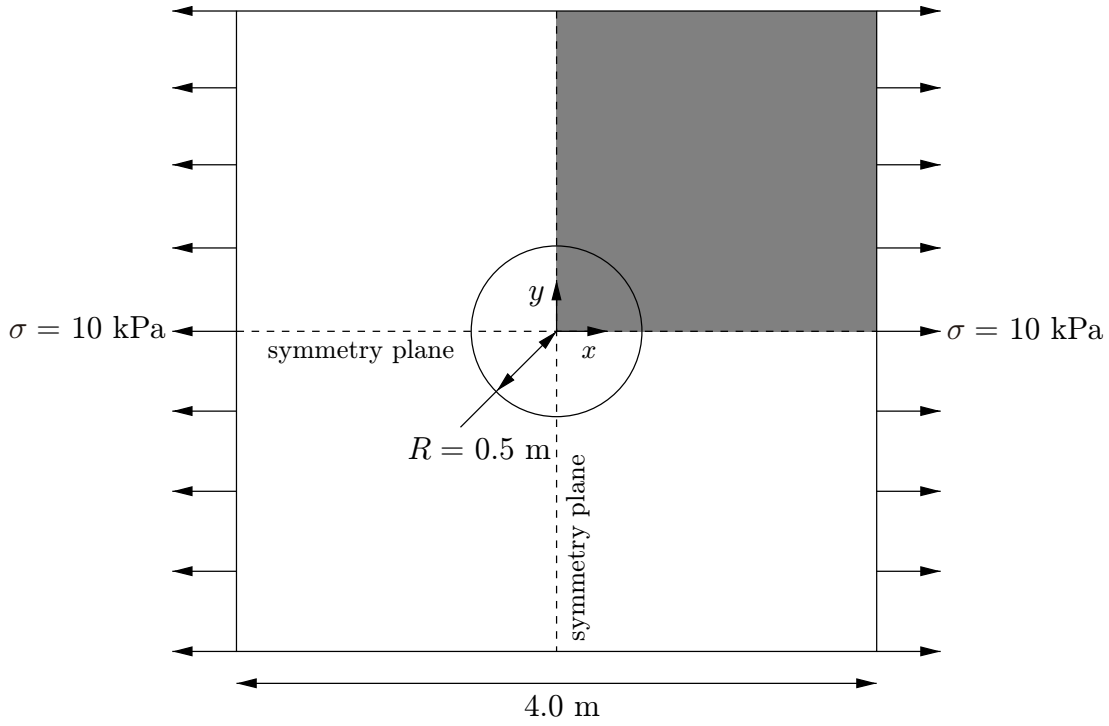


図 2.15　穴あき板の形状

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．

$FOAM_RUN/tutorials/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．

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

convertToMeters 1;

vertices
(
    (0.5 0 0)
    (1 0 0)
    (2 0 0)
    (2 0.707107 0)
    (0.707107 0.707107 0)
    (0.353553 0.353553 0)
    (2 2 0)
    (0.707107 2 0)
    (0 2 0)
    (0 1 0)
    (0 0.5 0)
    (0.5 0 0.5)
    (1 0 0.5)
    (2 0 0.5)
    (2 0.707107 0.5)
    (0.707107 0.707107 0.5)
    (0.353553 0.353553 0.5)
    (2 2 0.5)
    (0.707107 2 0.5)
    (0 2 0.5)
    (0 1 0.5)
    (0 0.5 0.5)
);

blocks
(
    hex (5 4 9 10 16 15 20 21) (10 10 1) simpleGrading (1 1 1)
```

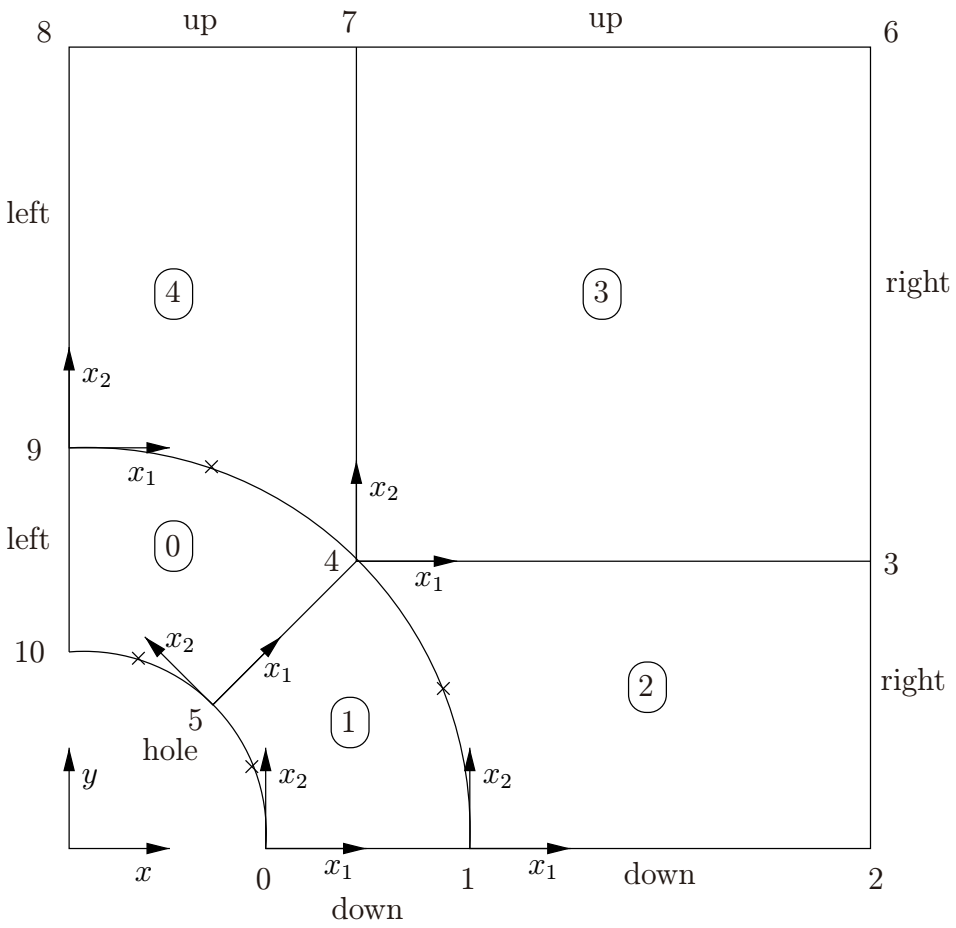


図 2.16 穴あき板解析のためのメッシュのブロック構造

```
    hex (0 1 4 5 11 12 15 16) (10 10 1) simpleGrading (1 1 1)
    hex (1 2 3 4 12 13 14 15) (20 10 1) simpleGrading (1 1 1)
    hex (4 3 6 7 15 14 17 18) (20 20 1) simpleGrading (1 1 1)
    hex (9 4 7 8 20 15 18 19) (10 20 1) simpleGrading (1 1 1)
);

edges
(
    arc 0 5 (0.469846 0.17101 0)
    arc 5 10 (0.17101 0.469846 0)
    arc 1 4 (0.939693 0.34202 0)
    arc 4 9 (0.34202 0.939693 0)
    arc 11 16 (0.469846 0.17101 0.5)
    arc 16 21 (0.17101 0.469846 0.5)
    arc 12 15 (0.939693 0.34202 0.5)
    arc 15 20 (0.34202 0.939693 0.5)
);

patches
(
    symmetryPlane left
    (
        (8 9 20 19)
        (9 10 21 20)
    )
    patch right
    (
        (2 3 14 13)
        (3 6 17 14)
    )
    symmetryPlane down
    (
```

```
        (0 1 12 11)
        (1 2 13 12)
    )
    patch up
    (
        (7 8 19 18)
        (6 7 18 17)
    )
    patch hole
    (
        (10 5 16 21)
        (5 0 11 16)
    )
    empty frontAndBack
    (
        (10 9 4 5)
        (5 4 1 0)
        (1 4 3 2)
        (4 7 6 3)
        (4 9 8 7)
        (21 16 15 20)
        (16 11 12 15)
        (12 13 14 15)
        (15 14 17 18)
        (15 18 19 20)
    )
);

mergePatchPairs
(
);

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

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一の配向性を有する訳ではありません．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．

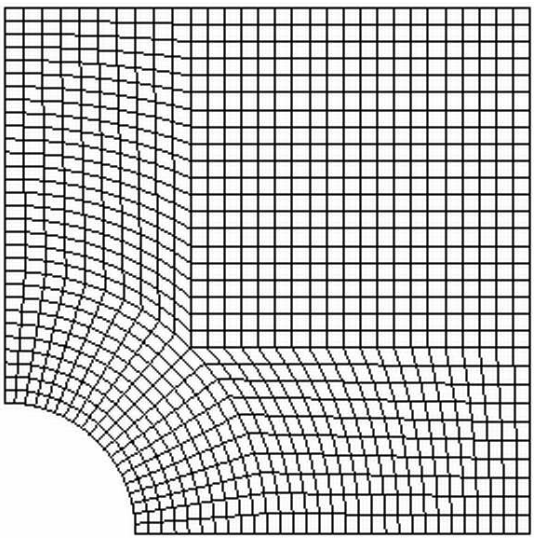


図 2.17 穴あき板問題のための解析メッシュ

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．

```
dimensions      [0 1 0 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    left
    {
        type            symmetryPlane;
    }
    right
    {
        type            tractionDisplacement;
        traction        uniform ( 10000 0 0 );
        pressure        uniform 0;
        value           uniform (0 0 0);
    }
    down
    {
        type            symmetryPlane;
    }
    up
    {
        type            tractionDisplacement;
        traction        uniform ( 0 0 0 );
        pressure        uniform 0;
        value           uniform (0 0 0);
    }
    hole
    {
        type            tractionDisplacement;
        traction        uniform ( 0 0 0 );
        pressure        uniform 0;
        value           uniform (0 0 0);
    }
    frontAndBack
    {
        type            empty;
    }
}
```

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

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
ヤング率	Pa	`E`	2×10^{11}
ポアソン比	—	`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方程式を解きません．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．

```
application     solidDisplacementFoam;
```

```
startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         100;

deltaT          1;

writeControl    timeStep;

writeInterval   20;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression uncompressed;

timeFormat      general;

timePrecision   6;

graphFormat     raw;

runTimeModifiable yes;

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

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．この勾配の正確かつ滑らかな評価によって計算は恩恵を受けます．通常，有限体積法における離散化は，ガウスの定理に基づいています．ガウス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．

```
d2dt2Schemes
{
    default     steadyState;
}

gradSchemes
{
    default         leastSquares;
    grad(D)         leastSquares;
    grad(T)         leastSquares;
}

divSchemes
{
    default         none;
    div(sigmaD)     Gauss linear;
}
```

```
laplacianSchemes
{
    default         none;
    laplacian(DD,D) Gauss linear corrected;
    laplacian(DT,T) Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         none;
}

fluxRequired
{
    default         no;
    D               yes;
    T               no;
}

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

fvSolution ディクショナリは，線形方程式ソルバおよび求解に使用されるアルゴリズムを制御します．まず *solvers* サブディクショナリを見て，U のソルバとして ICCG を選択してください．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，もしくはさらに高い値の 0.1 等となります．

```
solvers
{
    D GAMG
    {
        tolerance        1e-06;
        relTol           0.9;

        smoother         GaussSeidel;

        cacheAgglomeration true;

        nCellsInCoarsestLevel 20;

        agglomerator     faceAreaPair;
        mergeLevels      1;
    };

    T GAMG
    {
        tolerance        1e-06;
        relTol           0.9;

        smoother         GaussSeidel;

        cacheAgglomeration true;

        nCellsInCoarsestLevel 20;

        agglomerator     faceAreaPair;
        mergeLevels      1;
    };
}
```

```
stressAnalysis
{
    compactNormalStress yes;
    nCorrectors     1;
    D               1e-06;
}

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

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/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します．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

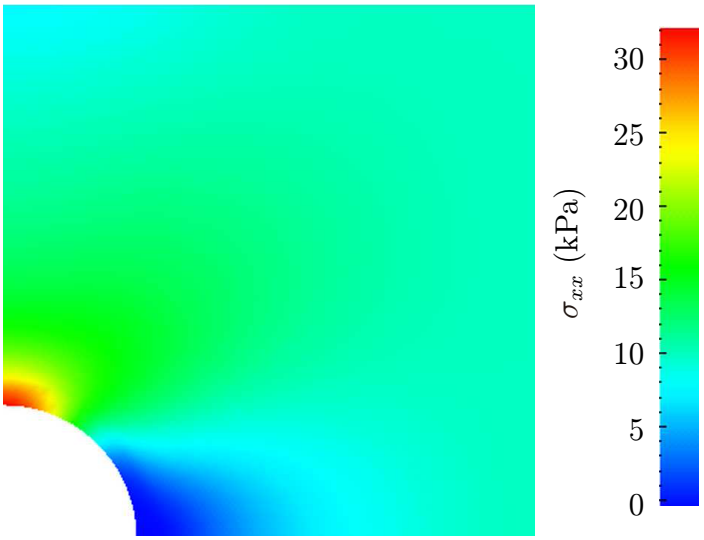


図 2.18　穴あき板における応力場

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．

```
interpolationScheme cellPoint;
setFormat raw;

sets
(
   leftPatch
   {
       type uniform;
       axis y;
       start (0 0.5 0.25);
       end (0 2 0.25);
       nPoints 100;
   }
);

surfaces
();

fields
(
   sigmaxx
);

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

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．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] '<datafile>', 1e4*(1+(0.125/(x**2))+(0.09375/(x**4)))
```

プロット例を図 2.19 に示します．

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．

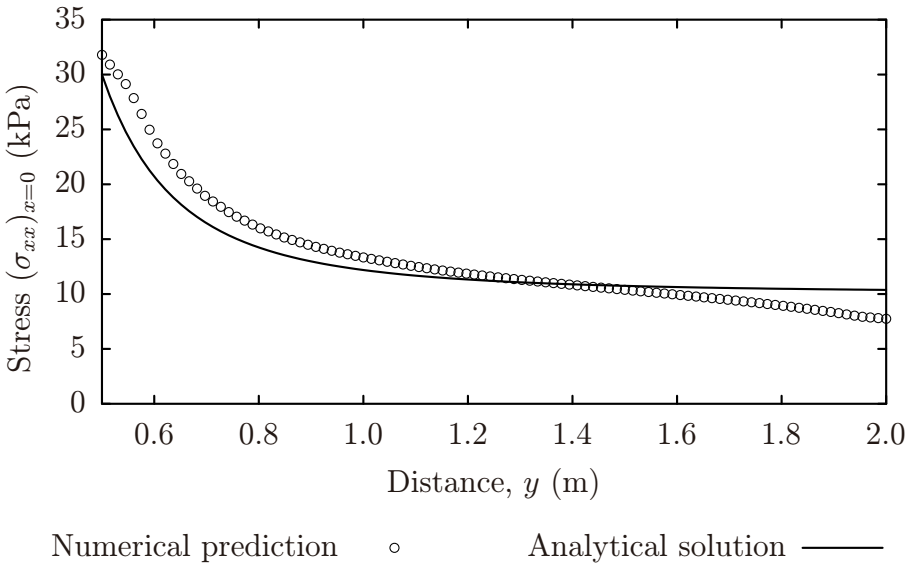


図 2.19　垂直方向対称面における法線方向応力

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．崩壊している間，水流は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．

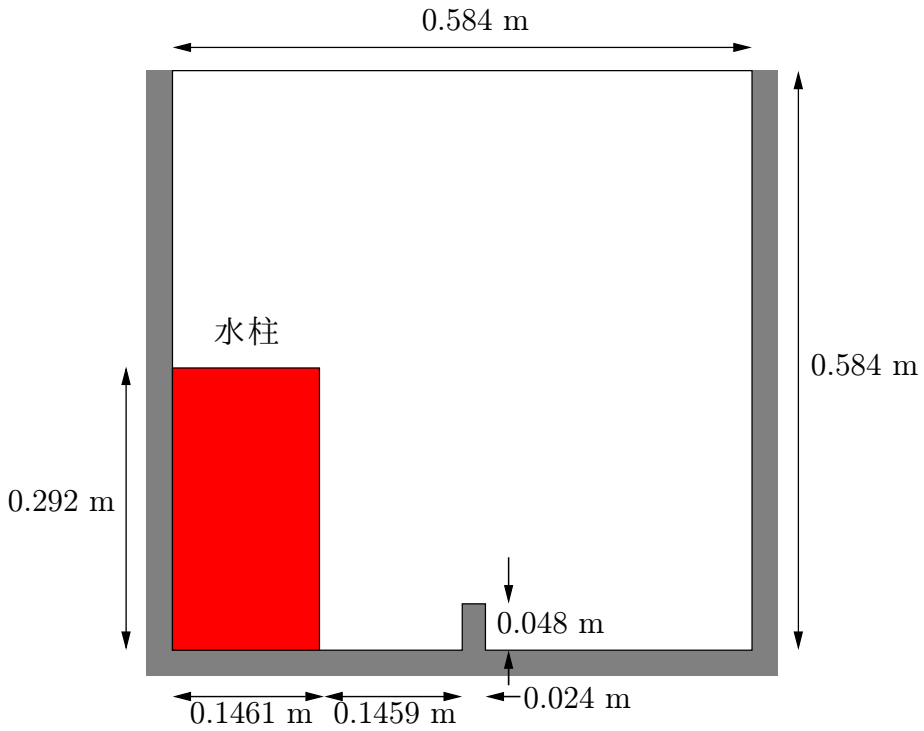


図 2.20　ダム決壊の計算形状

2.3.1　格子の生成

$FOAM_RUN/tutorials/interFoam にある *damBreak* のケースディレクトリに移動しましょう．前に述べた方法で blockMesh を走らせて格子を生成してください．この damBreak の格子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．

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

convertToMeters 0.146;

vertices
(
    (0 0 0)
    (2 0 0)
    (2.16438 0 0)
    (4 0 0)
    (0 0.32876 0)
    (2 0.32876 0)
    (2.16438 0.32876 0)
    (4 0.32876 0)
    (0 4 0)
    (2 4 0)
    (2.16438 4 0)
    (4 4 0)
    (0 0 0.1)
    (2 0 0.1)
    (2.16438 0 0.1)
    (4 0 0.1)
```

```
    (0 0.32876 0.1)
    (2 0.32876 0.1)
    (2.16438 0.32876 0.1)
    (4 0.32876 0.1)
    (0 4 0.1)
    (2 4 0.1)
    (2.16438 4 0.1)
    (4 4 0.1)
);

blocks
(
    hex (0 1 5 4 12 13 17 16) (23 8 1) simpleGrading (1 1 1)
    hex (2 3 7 6 14 15 19 18) (19 8 1) simpleGrading (1 1 1)
    hex (4 5 9 8 16 17 21 20) (23 42 1) simpleGrading (1 1 1)
    hex (5 6 10 9 17 18 22 21) (4 42 1) simpleGrading (1 1 1)
    hex (6 7 11 10 18 19 23 22) (19 42 1) simpleGrading (1 1 1)
);

edges
(
);

patches
(
    wall leftWall
    (
        (0 12 16 4)
        (4 16 20 8)
    )
    wall rightWall
    (
        (7 19 15 3)
        (11 23 19 7)
    )
    wall lowerWall
    (
        (0 1 13 12)
        (1 5 17 13)
        (5 6 18 17)
        (2 14 18 6)
        (2 3 15 14)
    )
    patch atmosphere
    (
        (8 20 21 9)
        (9 21 22 10)
        (10 22 23 11)
    )
);

mergePatchPairs
(
);

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

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 gamma (γ) 場の gammaContactAngle 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_B，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，gamma に対して gammaContact の境界条件を設定する代わり，zeroGradient の境界条件を課すことになります．

top の境界は大気に対して開放されていることから，atmosphere タイプの境界条件を与えます．また，この 2 次元問題において，前後の面のような defaultFaces の境界条件は通常通り empty タイプです．

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 γ に対して，以下のような非一様な初期条件を与えます．

$$\gamma = \begin{cases} 1 & \text{液相} \\ 0 & \text{気相} \end{cases} \tag{2.15}$$

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．

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

defaultFieldValues
(
    volScalarFieldValue gamma 0
    volVectorFieldValue U (0 0 0)
);

regions
(
    boxToCell
    {
        box (0 0 -1) (0.1461 0.292 1);

        fieldValues
        (
            volScalarFieldValue gamma 1
        );
    }
);

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

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，液体の領域における格子の集合を定義しています．また，この領域における相比率 γ を 1 と指定しています．

さて，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 gamma 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．

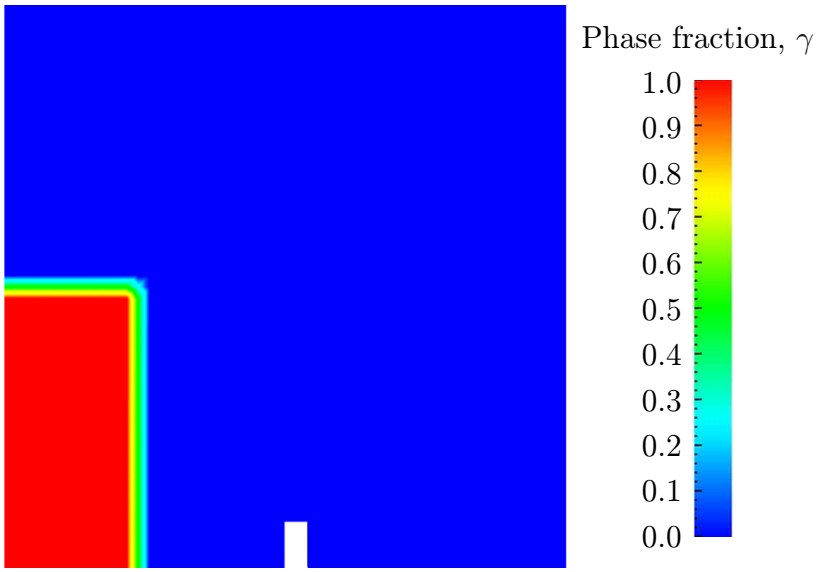


図 2.21 相比率 gamma の初期条件

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．このファイルは，phase1 と phase2 に分かれて，各流体の物性値を保持しています．

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．

environmentalProperties のディクショナリは，重力加速度ベクトルを指定していますが，このチュートリアルでは，この値は $(0, 9.81, 0)\,\mathrm{m\,s^{-2}}$ としてください．

2.3.5 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，クーラン数 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

phase1 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

phase2 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

領域において，およそ 0.2 である Co の上限値を超えてはいけません．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 maxCo を 0.2 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，ステップ自体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．

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

application interFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         1;

deltaT          0.001;

writeControl    adjustableRunTime;

writeInterval   0.05;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression uncompressed;

timeFormat      general;

timePrecision   6;

runTimeModifiable yes;
```

```
adjustTimeStep  yes;

maxCo           0.5;

maxDeltaT       1;

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

2.3.6 離散化スキーム

OpenFOAM における自由表面の扱いは，乱流の影響を考慮しません．これは乱流モデルに対するレイノルズ平均モデルの手法が空気と水の間のごく薄い表面の概念と適合しない事実に基づいています．結果として，全ての自由表面シミュレーションは流体の直接数値シミュレーション (DNS) としてみることができます．DNS はメッシュ数に対して一定の要求があり，それは我々のテストケースにおけるメッシュの解像度を遥かに超えています．

このソルバは，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，対流スキームの選択はそれが静的であるか，また閉鎖系であるかによって制限されません．

したがって，運動量の式における対流項では風上差分を使用して計算を安定させることとします．風上差分によって導入された数値拡散は計算を安定させることでしょう．

風上の設定は，*fvSchemes* ディクショナリの *divSchemes* サブディクショナリ内に作成されており，運動量方程式における $\nabla \cdot (\rho\phi\boldsymbol{U})$ 項に対応する `div(rho*phi,U)` キーワードは，`Gauss upwind` を読みこみます．相 γ に対する二つの方程式における対流項にはいくつかの注意を必要とします．適切な精度を維持しつつ有界性を保証しなくてはならないため，対流項内での補完間のためには，有界なスカラに対するリミッタ付きの線形なスキームである `limitedLinear01` を用います．

リミッタ付きの線形なスキームは 4.4.1 項に記述される係数を必要とします．ここで安定性を最重視して，キーワード `div(phi,gamma)` に対応する $\nabla \cdot (\phi\gamma)$ 項，キーワード `div(phirb,gamma)` に対応する $\nabla \cdot (\phi_{\mathrm{rb}}\gamma)$ 項で $\phi = 1.0$ の状態を選びます．

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．それゆえ *fvSchemes* ディクショナリのエントリは以下のようになるべきです．

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

ddtSchemes
{
    default Euler;
}

gradSchemes
{
    default         Gauss linear;
    grad(U)         Gauss linear;
    grad(gamma)     Gauss linear;
}

divSchemes
{
    div(rho*phi,U)  Gauss upwind;
    div(phi,gamma)  Gauss limitedLinear01 1;
    div(phirb,gamma) Gauss limitedLinear01 1;
}
```

```
laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

fluxRequired
{
    default         no;
    pd;
    pcorr;
    gamma;
}

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

2.3.7　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，相方程式の PISO ループに対する反復数も指定します．特に興味深いものは nGammaSubCycles と cGamma キーワードです．nGammaSubCycles は γ 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，四つの sub-cycle を指定しており，γ 方程式は実際の各時間ステップ内で 4 分の 1 の幅の時間スッテプで 4 回解かれていることを意味します．

cGamma 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．

2.3.8　コードの実行

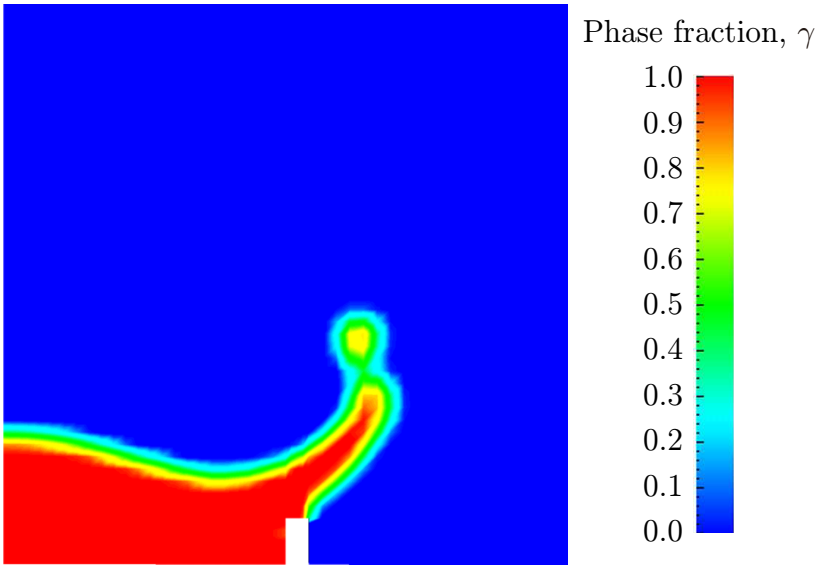
コードの実行方法については，前述のチュートリアルに詳細に記述しています．以下を試してください．tee によって計算内容をターミナルウインドウに表示しつつ，ログファイルに記録します．

```
cd $FOAM_RUN/tutorials/interFoam
interFoam | tee log
```

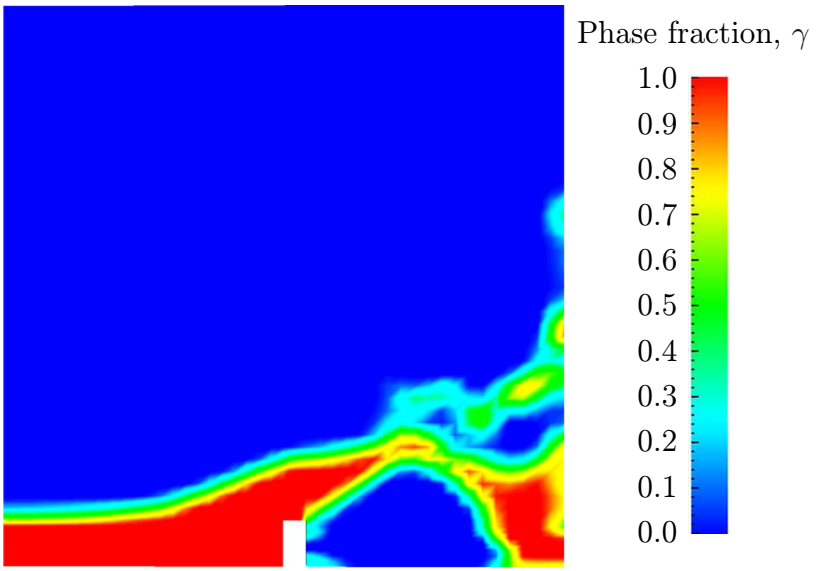
コードは，出力のコピーを *log* ファイル内に記録しつつ，対話形式で実行されます．

2.3.9　後処理

結果の後処理は，通常の方法で行えます．図 2.22 のように参照時間の経過とともに相比率 gamma の発達を見ることができます．



(a) $t = 0.25\,\mathrm{s}$ のとき



(b) $t = 0.50\,\mathrm{s}$ のとき

図 2.22 γ 相のスナップショット

2.3.10 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 の並列計算という能力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 *blockMesh-*

Dict 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．入力内容が正しければ，格子を生成します．

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 *gamma* 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 `gamma` 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` という場は変更する必要がな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持つように行いたいものです．つまり，その要素が $\gamma = 1$ か $\gamma = 0$ をもつようにです．mapFields でのフィールドの更新は界面に補間された値 $0 < \gamma < 1$ が生成されるかもしれないので，setFields ユーティリティを以下のように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 γ のバックアップファイル *0/gamma.org* を *0/gamma* にコピーします．

```
cd $FOAM_RUN/tutorials/interFoam/damBreakFine
cp -r 0/gamma.org 0/gamma
setFields $FOAM_RUN/tutorials/interFoam damBreakFine
```

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 次元方向のには分割され得ず，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．

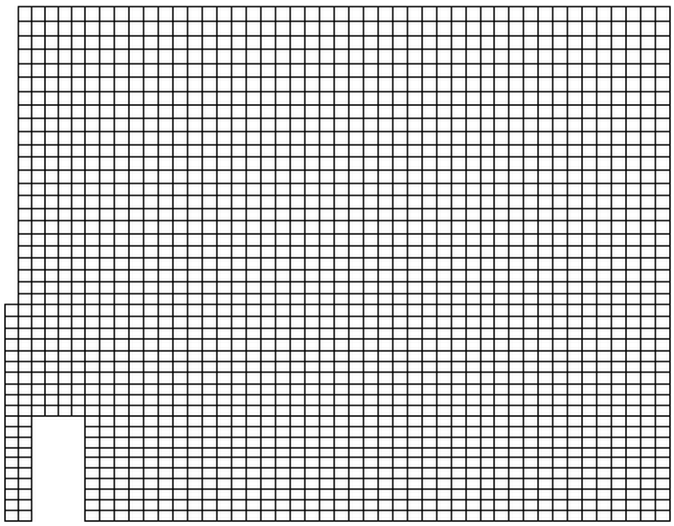


図 2.23　並列プロセスケースでのプロセッサ 2 のメッシュ

2.3.11　並列計算ケースの後処理

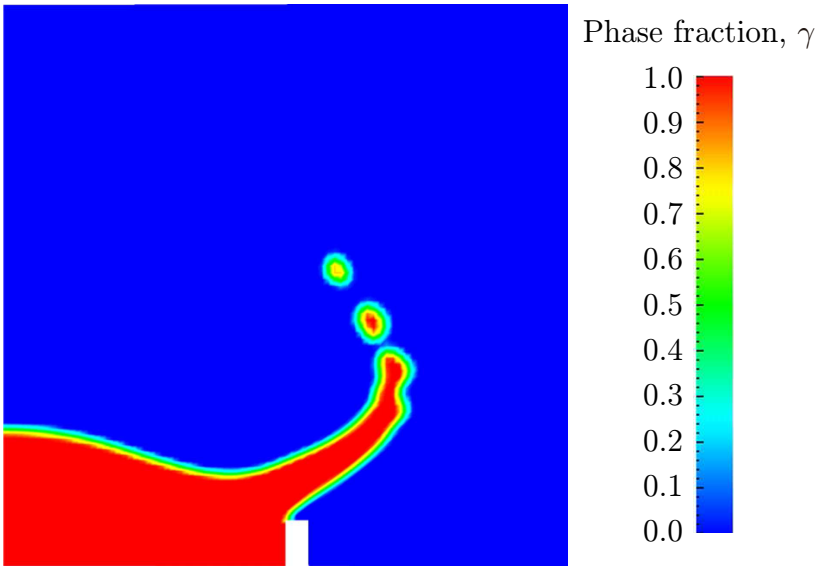
一度ケースの実行が完了したら，分解されたフィールドとメッシュは reconstructPar を実行して後処理のために再統合します．コマンドラインから容易に実行できます．細かい格子による結果は図 2.24 に表されます．インタフェースでの結果は粗い格子のものと比較して著しく改良されたことがみてとれます．

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．

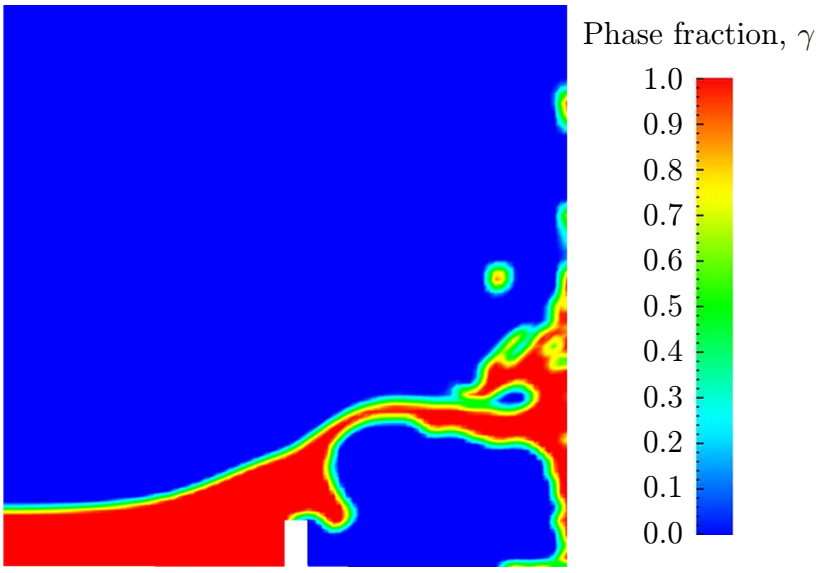
例えば，paraFoam を以下のように起動します．

```
paraFoam -case processor1
```

すると `processor1` は ParaView のケースモジュールとして表れます．`simple` 方式を使った領域分割を行った `processor1` から図 2.23 のような格子が見えます．



(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 γ 相のスナップショット

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．ソルバとユーティリティのコードの実際の書き方についてはここでは述べませんがプログラマ・ガイドに記載してあります．なお，プログラマ・ガイドは現在改訂中ですので，もし何か不明な点がありましたら，OpenFOAM の討議グループや OpenFOAM のウェブサイトで新たな情報が得られるかもしれません．

3.1　OpenFOAM のプログラミング言語

OpenFOAM ライブラリィのはたらき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．そのために不可欠な知識が本章にあります．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や OpenFOAM のメインプログラミング言語としての C++ の選択の背景として一般論で様々な考えを説明するための言語の概念に着目することです．

3.1.1　言語とは

口話と数学が普及には，特に抽象的な概念を表現する際の効率性が重要でした．

（概念を表現する際の効率性の）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．テンソル数学と値計算法についてはプログラマ・ガイドの第 1 章と第 2 章に記載しています．

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，各々のコードのピースがそれ自身，OpenFOAM のライブラリに依存しているコンポーネントにアクセスすることから，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

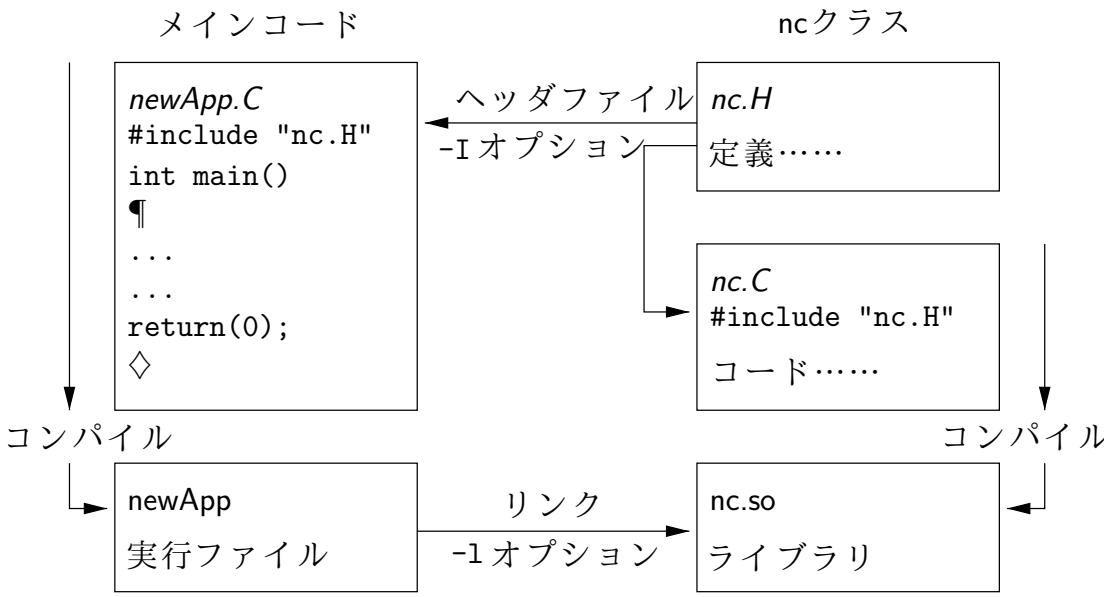


図 3.1　ヘッダファイル，ソースファイル，コンパイル，リンク

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．すべての内蔵コードはヘッダファイルに入れること，コード可読性を高めるためにメインコードに relevant location に含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．

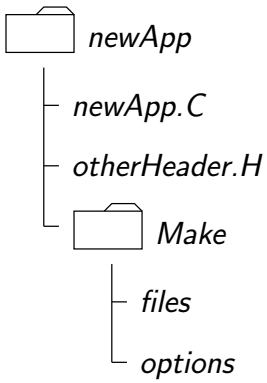


図 3.2 アプリケーションのディレクトリ構成

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/$WM_ARCH/*デイレクトリの中に設定された機種に依存するパス，例えば，*/usr/X11/*や*$(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ライブラリ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．

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．もし*Make*ディレクトリから依存ファイルと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*

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：`1.5`
$WM_PROJECT_DIR	OpenFOAM のバイナリ実行ファイル置き場へのフルパス， 例：*$HOME/OpenFOAM/OpenFOAM-1.5*
$WM_PROJECT_USER_DIR	ユーザのバイナリ実行ファイル置き場へのフルパス， 例：*$HOME/OpenFOAM/$USER-1.5*

その他のパスと設定	
$WM_ARCH	マシン構造：`cray decAlpha dec ibm linux linuxPPC sgi3 sgi32 sgi64 sgiN32 solaris sx4 t3d`
$WM_COMPILER	使用するコンパイラ：`Gcc3` - gcc 4.3.1, `KAI` - KAI
$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_JAVAC_OPTION	JAVA のためのコンパイルオプション：`Debug` - debugging, `Opt` optimisation.
$WM_LINK_LANGUAGE	ライブラリや実行ファイルのリンクに使うコンパイラ．多言語プロジェクトにおいて$WM_LINK_LANGUAGE は主要言語を決める．
$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_PROJECT_LANGUAGE	プロジェクトのプログラミング言語，例：`c++`
$WM_SHELL	wmake スクリプトに使うシェル：`bash`，`csh`，`ksh`，`tcsh`

表 3.2　wmake の環境変数の設定

ディレクトリも削除される必要があります．

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：turbFoam アプリケーション

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

```
/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 1991-2007 OpenCFD Ltd.
     \\/     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 2 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, write to the Free Software Foundation,
    Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Application
    turbFoam

Description
    Transient solver for incompressible, turbulent flow.

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

#include "fvCFD.H"
#include "incompressible/singlePhaseTransportModel/singlePhaseTransportModel.H"
#include "incompressible/turbulenceModel/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;

    for (runTime++; !runTime.end(); runTime++)
    {
        Info<< "Time = " << runTime.timeName() << nl << endl;

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

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

            fvVectorMatrix UEqn
            (
                fvm::ddt(U)
              + fvm::div(phi, U)
              + turbulence->divR(U)
            );

            if (momentumPredictor)
            {
                solve(UEqn == -fvc::grad(p));
            }

            // --- PISO loop

            for (int corr=0; corr<nCorr; corr++)
            {
                volScalarField rUA = 1.0/UEqn.A();

                U = rUA*UEqn.H();
                phi = (fvc::interpolate(U) & mesh.Sf())
                    + fvc::ddtPhiCorr(rUA, U, phi);

                adjustPhi(phi, U, p);

                // Non-orthogonal pressure corrector loop
                for (int nonOrth=0; nonOrth<=nNonOrthCorr; nonOrth++)
                {
```

```
                    // Pressure corrector

                    fvScalarMatrix pEqn
                    (
                        fvm::laplacian(rUA, p) == fvc::div(phi)
                    );

                    pEqn.setReference(pRefCell, pRefValue);
                    pEqn.solve();

                    if (nonOrth == nNonOrthCorr)
                    {
                        phi -= pEqn.flux();
                    }
                }

#               include "continuityErrs.H"

                U -= rUA*fvc::grad(p);
                U.correctBoundaryConditions();
            }
        }

        turbulence->correct();

        runTime.write();

        Info<< "ExecutionTime = " << runTime.elapsedCpuTime() << " s"
            << "  ClockTime = " << runTime.elapsedClockTime() << " s"
            << nl << endl;
    }

    Info<< "End\n" << endl;

    return(0);
}


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

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，*turbFoam.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．

turbFoam は cfdTools や incompressibleRASModels 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．

```
EXE_INC = \
    -I$(LIB_SRC)/finiteVolume/lnInclude \
    -I$(LIB_SRC)/turbulenceModels \
    -I$(LIB_SRC)/transportModels

EXE_LIBS = \
    -lincompressibleTurbulenceModels \
    -lincompressibleTransportModels \
    -lfiniteVolume \
    -lmeshTools
```

turbFoam は *turbFoam.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．

```
turbFoam.C

EXE = $(FOAM_APPBIN)/turbFoam
```

$FOAM_CFD/turbFoam ディレクトリで wmake とタイプすれば turbFoam 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 turbFoam.C

SOURCE_DIR=.
SOURCE=turbFoam.C ;
g++ -DFOAM_EXCEPTION -Dlinux -DlinuxOptMPICH
-DscalarMachine -DoptSolvers -DPARALLEL -DUSEMPI -Wall -O2 -DNoRepository
-ftemplate-depth-17 -I/export/warhol/chris/OpenFOAM/OpenFOAM-1.4/src/OpenFOAM/lnInclude
-IlnInclude
-I.
......
-lmpich -L/usr/X11/lib -lm
-o /export/warhol/chris/OpenFOAM/OpenFOAM-1.4/applications/bin/linuxOptMPICH/turbFoam
```

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: ‘/export/warhol/chris/OpenFOAM/OpenFOAM-1.4/applications/bin/linuxOptMPICH/turbFoam’
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 *$WM_PROJECT_DIR/.OpenFOAM-1.3/controlDict* ファイルの中にあり，設定を変更したい場合には，*$HOME* ディレクトリに (例えば*$HOME/.OpenFOAM-1.3/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．

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）.
`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

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．代わりに，OpenFOAM は再コンパイルの必要性を評価するために foamUser とよばれる特別なライブラリを用いています．これはデフォルトにより，最初に各アプリケーションの中にコンパイルされた foamUser ライブラリをもたせることにより有効になります．foamUser ライブラリは，*$FOAM_SRC/foamUser* のディレクトリの中にあるコードからコンパイルされます．ユーザが行うことは，foamUser と再コンパイルされた *foamUser* の *Make/options* ファイルの中にリンクされたライブラリに new ライブラリを追加するだけです．

前述の例の場合であれば，ユーザーは *foamUser* ディレクトリのローカルコピーを作り，例えば以下のようなディレクトリに移してください．

```
cp -r $WM_PROJECT_DIR/src/foamUser $WM_PROJECT_USER_DIR/applications
cd $WM_PROJECT_USER_DIR/applications/foamUser
```

この foamUser ライブラリを，*$FOAM_USER_LIBBIN* の中にローカルにコンパイルするように，以下のように *Make/files* ファイルを編集しましょう．

```
libfoamUser.C
LIB = $(FOAM_USER_LIBBIN)/libfoamUser
```

新しいライブラリを *Make/options* 内の `LIB_LIBS` に追加しましょう．

```
LIB_LIBS = \
    -l...  \
    -lnew
```

最後に，ライブラリを以下のコマンドで再コンパイルしましょう．

```
wmake libso
```

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．OpenFOAM は B.1 節で述べているように，MPI の実装である MPICH も使用することもできます．

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．

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

numberOfSubdomains 4;

method          simple;

simpleCoeffs
{
    n               (2 2 1);
    delta           0.001;
}

hierarchicalCoeffs
{
    n               (1 1 1);
    delta           0.001;
    order           xyz;
}

metisCoeffs
{
    processorWeights
    (
        1
        1
        1
        1
    );
}

manualCoeffs
{
    dataFile        "";
}

distributed     no;

roots
(
);

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

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．

`metis`　METIS 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，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．

`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．

必須入力		
numberOfSubdomains	サブドメインの総数	N
method	分割方法	simple/ hierarchical/ 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...
metisCoeffs エントリ		
processorWeights	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>)
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（オプション）—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

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
```

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 (openMPI) 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/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
    $FOAM_RUN/tutorials/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_APP/solvers* ディレクトリの中にあり，コマンドラインから app 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，非圧縮性，乱流の turbFoam ソルバ，といったように分かり易い名前が付けられています．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ラプラス方程式を解く
potentialFoam	シンプルなポテンシャル流のコード．完全ナビエ・ストークスコードを解く際の保存された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	
boundaryFoam	1 次元の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．
channelOodles	チャネル内流れ用の非圧縮 LES
icoDyMFoam	ダイナミック・メッシュをもつニュートン流体の非圧縮性，層流の非定常ソルバ
icoFoam	非圧縮性，層流の速度-圧力ソルバ．非ニュートン流体も可
nonNewtonianIcoFoam	非ニュートン流体の非圧縮性，層流の非定常ソルバ
oodles	非圧縮性の LES ソルバ
simpleFoam	非ニュートン流体の非圧縮性，乱流の定常状態ソルバ
turbDyMFoam	ダイナミックメッシュをもつニュートン流体の非圧縮性，乱流の非定常ソルバ
turbFoam	非圧縮性，乱流の非定常ソルバ
圧縮性流れ	

coodles	圧縮性の LES ソルバ
rhoCentralFoam	中央風上スキームに基づいた密度ベース圧縮性流れのソルバ
rhoPimpleFoam	換気と熱輸送のための圧縮性乱流用の非定常ソルバ
rhoPorousSimpleFoam	多孔性を陰的または陽的に扱う圧縮性流体のための非定常乱流ソルバ
rhopSonicFoam	圧力・密度に基づいた圧縮性流れのソルバ
rhoSimpleFoam	換気と熱輸送のある圧縮性流体の乱流の定常状態ソルバ
rhoSonicFoam	密度に基づいた圧縮性流れのソルバ
rhoTurbFoam	圧縮性，乱流の非定常ソルバ
sonicFoam	圧縮性，遷音速／超音速層流気体ソルバ
sonicFoamAutoMotion	移動メッシュをもつ圧縮性，遷音速／超音速層流気体ソルバ
sonicLiquidFoam	圧縮性，遷音速／超音速層流液体ソルバ
sonicTurbFoam	圧縮性，遷音速／超音速乱流ソルバ

多層流

bubbleFoam	液体の中の気泡のように非圧縮分散性 2 相 2 流体ソルバ
compressibleLesInterFoam	界面を捕獲するやり方で流体占有率を求め (VOF 法)，不混和性の圧縮性等温 2 相流を LES で解くソルバ
interDyMFoam	VOF 法と補助的な格子移動を用いて，界面を捕獲する非圧縮性の 2 相流のソルバ
interFoam	VOF 法を用いて界面を捕獲する非圧縮性の 2 相流のソルバ
interPhaseChangeFoam	VOF 法を用いて，キャビテーション等の相変化を伴なう不混和性の非圧縮性等温 2 相流を解くソルバ
lesCavitatingFoam	LES 乱流モデルによる非定常のキャビテーション用コード
lesInterFoam	インタフェースをもつ非圧縮 2 相流のソルバ．乱流のモデル化は各種の非圧縮性 LES モデルにより，これらは実行時に変更できる．
multiphaseInterFoam	VOF 法を使った多数のインタフェースをもつ非圧縮非混合性流れの任意数のソルバ
rasCavitatingFoam	RAS 乱流モデルによる非定常のキャビテーション用コード
rasInterFoam	インタフェースをもった 2 相の非圧縮性流れのソルバ．乱流は各種の非圧縮性 RAS モデルのランタイムを使ってモデル化される
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

coldEngineFoam	内燃機関のコールドフローのソルバ
dieselEngineFoam	ディーゼルエンジン用噴射・燃焼用コード
dieselFoam	ディーゼル噴射・燃焼用コード
engineFoam	エンジン内部の燃焼用コード
PDRFoam	格子では解像できないくらい小さな固体による抵抗を扱うために，多孔質の抵抗を分布させる PDR モデルを内蔵した圧縮性予混合／部分的予混合乱流燃焼コード
reactingFoam	化学反応コード
XiFoam	圧縮性予混合／部分的予混合乱流燃焼コード
Xoodles	圧縮性予混合／部分的予混合乱流燃焼ラージ・エディ・シミュレーション (LES) コード

熱輸送

buoyantFoam	換気と熱輸送がある圧縮性乱流・浮力流用の非定常ソルバ
buoyantSimpleFoam	
buoyantSimpleRadiationFoam	放射と換気，熱輸送がある圧縮性乱流・浮力流用の定常ソルバ
chtMultiRegionFoam	浮力駆動の流れと固体との熱輸送を連成するソルバ

lesBuoyantFoam	換気と熱輸送がある圧縮性乱流・浮力流用の LES 乱流モデルによる非定常ソルバ
電磁流体	
electrostaticFoam	静電方程式コード
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	選択が自由な熱拡散と熱応力をもった線形弾性や固体の微小ひずみの非定常分離有限体積ソルバ
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
分子力学	
gnemdFoam	任意形状の解析領域における原子を解析する汎用の分子力学用ソルバであり，場のデータを作成するために原子や分子の量を格子内で平均する
mdEquilibrationFoam	分子力学系の平衡やその前提条件を解く
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 標準のアプリケーション

OpenFOAM で提供されているユーティティは*$FOAM_APP/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，magU は速度場のデータから速度を計算し，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	
boxTurb	与えられたエネルギスペクトルに適合し，自由に発散する乱流の box を生成する
engineSwirl	エンジン計算のために旋回流を発生させる
FoamX	（不明）
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
setFields	ディクショナリでセルのセットを選択する
メッシュの生成 — 5.3 節参照	
blockMesh	メッシュを生成する
extrudeMesh	既存のパッチやファイルから読み込んだパッチを押し出す
メッシュの変換 — 5.5 節参照	
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
ccm26ToFoam	CCM バージョン 2.6 のライブラリを利用して CCM ををを変換する
cfxToFoam	CFX メッシュを OpenFOAM 形式へ変換する
fluentMeshToFoam	Fluent のメッシュを複数の領域と領域の境界の処理を含む OpenFOAM 形式に変換する
foamMeshToFluent	OpenFOAM メッシュを Fluent メッシュ形式で出力する
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* 形式メッシュを OpenFOAM 形式へ変換する
kivaToFoam	KIVA3vf グリッドを OpenFOAM 形式へ変換する
mshToFoam	アドベンチャーシステムによって作られた*.msh* 形式を読み込む
netgenNeutralToFoam	Netgen4.4 によって書かれた Neutral ファイル形式を読み込む
plot3dToFoam	Plot3d メッシュ（アスキー形式）を OpenFOAM 形式に変換
polyDualMesh	（不明）
sammToFoam	STAR-CDSAMM メッシュを OpenFOAM 形式へ変換する
starToFoam	STAR-CDPROSTAR 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* ファイルを読み込む
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位相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
cellSet	ディクショナリでセルのセットを選択する
checkMesh	メッシュの妥当性をチェックする
couplePatches	周期的なプロセッサのパッチを再編成する
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
faceSet	ディクショナリで面のセットを選択する
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，個々に接続している必要がある
mergeMeshes	二つのメッシュを合体させる
mirrorMesh	（不明）
moveDynamicMesh	メッシュの動作と位相変化のユーティリティ
moveEngineMesh	エンジンシミュレーション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
patchTool	（不明）
pointSet	ディクショナリで点のセットを選択する
refineMesh	複数の方向にあるセルを細分化する．`-all` オプションを適用してすべてのセル（3 次元には 3 次元細分化を，2 次元には 2 次元細分化を）を細分化するか，refineMeshDict の cellSet to refine を読み込んでいくつかの方向を細分化する．
renumberMesh	行列の帯幅を狭くするためにセルリストに順番を付け直す．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，順番を付け直すことで行う
rotateMesh	メッシュおよび場を方向 n1 から方向 n2 へと回転させる
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し，それらを連続した時刻ディレクトリに書く．各領域は，セル - 面 - セルと辿ることによって届くことができる領域として指定される．meshWave を使用する．平行して動くことはできるが，テストはされていない
stitchMesh	メッシュを縫う
subsetMesh	cellSet に基づいたメッシュの区分を選択する
tetDecomposition	面とセルの中心の分解を利用してメッシュを 4 面体に分解する
transformPoints	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，セルを締め上げる

画像の後処理 — 第 6 章参照

ensight76FoamExec	変換せずに OpenFOAM のデータを直接読むための EnSight 7.6 のモジュール
paraFoam	（不明）

データ変換の後処理 — 第 6 章参照

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
foamToFieldview9	OpenFOAM のメッシュをバージョン 3.0Fieldview-UNS 形式（バイナリ）へ変換する．Fieldview リリース 9 のレファレンスマニュアルで付録 D（体系化されていないデータの形式）を参照してください．Fieldview リストの *uns/write_binary_uns.c* から各種借用する
foamToGMV	形式の出力を GMV で読めるファイルに変換する．http://www-xdiv.lanl.gov/XCM/gmv/ から入手できるバイナリを用いて後処理を行う
foamToVTK	レガシーの VTK ファイル形式のライター．volScalar，volVector，pointScalar，pointVector，surfaceScalar 場を操作する．メッシュの接続形態が変化する．アスキーとバイナリの両方が用いられる．一度の操作で書き出す．部分集合だけを書き出す．セルが自動的に分解する．vtk によって操作されたため分解された境界上の多角形である
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	プログラムを書く上で設定可能なグラフ
divU	各時間の速度場 U の発散を計算し，書き出す
enstrophy	各時間の速度場 U のエンストロフィを計算し，書き出す
flowType	各時間の速度場 U の flowType を計算し，書き出す
Lambda2	各時間の，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	各時間の速度場 U のローカルマッチ番号を計算し，書き出す
magGradU	各時間の速度場 U の計数規模を計算し，書き出す
magU	各時間の速度場 U の勾配の計数規模を計算し，書き出す
Pe	各時間のファイの場から得られる surfaceScalarField として Pe 番号を計算し，書き出す
Q	各時間の速度勾配テンソルの 2 番目の不変条件を計算し，書き出す
streamFunction	各時間の速度場 U の流れ機能を計算し，書き出す
Ucomponents	各時間の速度場 U における各要素について，Ux，Uy，Uz の三つのスカラ場を書き出す
uprime	各時間の uprime $(\sqrt{\frac{2}{3}k})$ のスカラ場を計算し，書き出す
vorticity	各時間の速度場 U の渦巻き運動を計算し，書き出す

圧力場の後処理

R	現在のステップについてのレイノルズ圧力 R を計算し，書き出す
Rcomponents	各時間のレイノルズ圧力 R の六つの要素のスカラ場を計算し，書き出す
stressComponents	各時間の圧力テンソル sigma の六つの要素のスカラ場を計算し，書き出す

壁の後処理

checkYPlus	データベースの各時間について，全ての壁のパッチに対する yPlus を計算し，リポートする
wallGradU	壁における U の勾配を計算し，書き出す
wallHeatFlux	volScalar 場の境界面として全てのパッチに対する熱フラックスを計算し，書き出す．そして全ての壁に対する不可欠なフラックスも書き出す
wallShearStress	現在のタイムステップで壁の受ける応力を計算して書き出す
yPlusLES	LES のために壁近傍のセルの yplus を計算する

パッチの後処理

patchAverage	すべてのパッチにわたって領域の平均を計算する
patchIntegrate	すべてのパッチにわたって領域を融合する

様々な後処理

engineCompRatio	幾何的圧縮の係数を計算する．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

postChannel	チャンネル流計算のポストプロセスデータ
ptot	毎回，全圧を計算する
sample	展開スキームを選択し，計算領域をサンプリングする．その際，オプションをサンプリングしてフォーマットをかき出す
sampleSurface	並行処理の際に表面をサンプリングする．（ただし，点は結合しない）
wdot	wdot を毎回計算し，書き出す
writeCellCentres	三つのコンポーネントを，閾値化してポストプロセスで使えるように volScalarFields として書き出す

並行処理 — 3.4 節参照

decomposePar	OpenFOAM の平衡計算のためにケースのメッシュと計算領域を自動的に分割する
reconstructPar	OpenFOAM の平衡計算のために分割したメッシュと計算領域を再構成する
reconstructParMesh	幾何情報のみを使ってメッシュを再結合し，あとで reconstructPar が計算領域を再構成できるように点/面/セル procAddressing に書き込む

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

エラーの推量

estimateScalarError	標準フォームによるスカラ輸送方程式の解の誤差を予想する
icoErrorEstimate	非圧縮性層流 CFD アプリケーション icoFoam の解の誤差を予想する
icoMomentError	非圧縮性層流 CFD アプリケーション icoFoam の解の誤差を予想する
momentScalarError	標準フォームによるスカラ輸送方程式の解の誤差を予想する

様々なユーティリティ

foamDebugSwitches	すべてのライブラリのデバッグスイッチを書き出す
foamInfoExec	ケースを調べ，スクリーンに情報を表示する

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．

Library of basic OpenFOAM tools — 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
dimensionSet	次元設定クラス
dimensionedTypes	次元<Type>クラスと導関数
fields	領域クラス
finiteVolume	有限体積離散化クラス
global	広域設定
interpolations	補間スキーム
matrices	行列クラス
meshes	メッシュクラス
primitives	初期クラス

CFD ツールライブラリ — cfdTools	
adjustPhi	境界フラックスの調整
bound	スカラ領域境界
compressible	圧縮性流れ用 CFD ツール
incompressible	非圧縮性流れ用 CFD ツール
wallDist	壁面境界に関する計算

後処理ライブラリ	
incompressiblePostProcessing	比圧縮性流れのデータの後処理用ツール
sampling	領域における特定の場所での領域データの抽出用ツール

解法とメッシュ操作のライブラリ	
cellDecompFiniteElement	有限要素スキームのセル分割
dynamicMesh	移動メッシュをもつシステムの解法
edgeMesh	Foredge-based メッシュ記述の操作のため
errorEstimation	誤差推定ツール
faceDecompFiniteElement	Face decomposed 有限要素スキームの面分割
ODE	常微分方程式のソルバ
shapeMeshTools	標準形で定義されたセルをもつメッシュの操作のためのツール
meshTools	OpenFOAM メッシュ操作のためのツール
triSurface	標準三角 surface-based メッシュ記述の操作のため

ラグランジュ粒子追跡ライブラリ	
dieselSpray	ディーゼル噴霧追跡解スキーム
lagrangian	基本ラグランジュもしくは粒子追跡解スキーム

共有ドメインライブラリ	
mico-2.3.13	共通オブジェクト・リクエスト・ブローカー・アーキテクチャ (CORBA) の実装
mpich-1.2.4	並列処理のための移植可能なメッセージパッシングインターフェイス
openmpi-1.2.6	並列処理のための移植可能なメッセージパッシングインターフェイス
zlib-1.2.1	汎用データの圧縮

さまざまなライブラリ	
engine	計算エンジンのツール
Gstream	2 次元グラフィックス流れ
randomProcesses	分析と生成のランダムプロセスのツール

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	
hThermo	エンタルピに基づく一般熱物理モデル計算
pureMixture	パッシブガス混合物の一般熱物理モデル計算

燃焼モデル — combustionThermophysicalModels	
hMixtureThermo	混合気燃焼のエンタルピ計算
hhuMixtureThermo	不燃気体と混合気のエンタルピ計算
homogeneousMixture	標準燃料質量分率 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
dieselMixture	f_{t} と f_{u} に基づく混合気燃焼
multiComponentMixture	複数の要素に基づく混合気燃焼 [**]
chemkinMixture	CHEMKIN 熱力学と反応スキームデータベースファイルを使った混合気燃焼

層流火炎速度モデル — laminarFlameSpeedModels	
constLaminarFlameSpeed	一定層流火炎速度
guldersLaminarFlameSpeed	Gulder の層流火炎速度モデル

液体の熱物理特性 — liquids	
nHeptane	ノルマルヘプタンの熱物理特性
nOctane	ノルマルオクタンの熱物理特性
nDecane	ノルマルデカンの熱物理特性
nDodecane	ノルマルドデカンの熱物理特性
isoOctane	イソオクタンの熱物理特性
diMethylEther	ジメチルエーテルの熱物理特性
diEthylEther	ジエチルエーテルの熱物理特性
water	水の熱物理特性

ガス種の熱物理特性 — specie	
perfectGas	理想気体に対する状態方程式
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
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
sutherlandTransport	温度依存輸送特性のためのサザーランドの公式

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

確率密度関数 — pdf	
RosinRammler	ロジン・ラムラー分布
normal	正規分布
uniform	一様分布
exponential	指数分布
general	一般化分布
化学モデル — chemistryModel	
chemistryModel	化学反応モデル
chemistrySolver	化学反応ソルバ

表 3.8　熱物理モデルのライブラリ

非圧縮性流れ用乱流モデル — incompressibleRASModels

laminar	層流用ダミー乱流モデル
kEpsilon	壁関数付き標準 k–ε モデル
RNGkEpsilon	壁関数付き RNG k–ε モデル
NonlinearKEShih	壁関数付き非線形 Shih k–ε モデル
LienCubicKE	壁関数付き Lien cubic k–ε モデル
QZeta	q–ζ 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	壁関数付き Launder-Reece-Rodi RSTM
LaunderGibsonRSTM	wall-reflection 条件と壁関数付き Launder-Gibson RSTM
SpalartAllmaras	外部流のための Spalart-Allmaras 1-eqn mixing-length 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	壁関数付き標準 k–ε モデル
RNGkEpsilon	壁関数付き RNG k–ε モデル
LaunderSharmaKE	Launder-Sharma 低 Re k–ε モデル
LRR	壁関数付き Launder-Reece-Rodi RSTM
LaunderGibsonRSTM	wall-reflection 条件と壁関数付き Launder-Gibson RSTM

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

laplaceFilter	ラプラスフィルタ
simpleFilter	単一フィルタ
anisotropicFilter	異方性フィルタ

Large-eddy シミュレーション差分 — LESdeltas

PrandtlDelta	プラントルのデルタ
cubeRootVolDelta	セル体積の立方根差分
smoothDelta	差分のスムージング

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
dynSmagorinsky	同時 Smagorinsky
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
dynMixedSmagorinsky	同時 Smagorinsky とスケール相似の混合モデル
oneEqEddy	k 方程式渦粘性モデル
dynOneEqEddy	同時 k 方程式渦粘性モデル
locDynOneEqEddy	局部同時 k 方程式渦粘性モデル
spectEddyVisc	スペクトル渦粘性モデル
LRDDiffStress	LRR 差応力モデル
DeardorffDiffStress	Deardorff 差応力モデル
SpalartAllmaras	Spalart-Allmaras モデル

圧縮性 LES モデル — compressibleLESmodels

Smagorinsky	Smagorinsky モデル
oneEqEddy	k 方程式渦粘性モデル
dynOneEqEddy	同時 k 方程式渦粘性モデル
lowReOneEqEddy	低 Re k 方程式渦粘性モデル
DeardorffDiffStress	Deardorff 差応力モデル

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 低非線形粘性モデル
BirdCarreau	Bird-Carreau 非線形粘性モデル

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と構成を取り扱います．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}-1.5* 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/${USER}-1.5* に設定されることです．コマンドラインでプリセットエイリアス，*run* を実行することにより，素早くこのディレクトリに移動することができます．OpenFOAM の配布の際に添付されているチュートリアルのケースは，ケースのディレクトリ構造の有用な例を提供しています．チュートリアルは*$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下さい．

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．

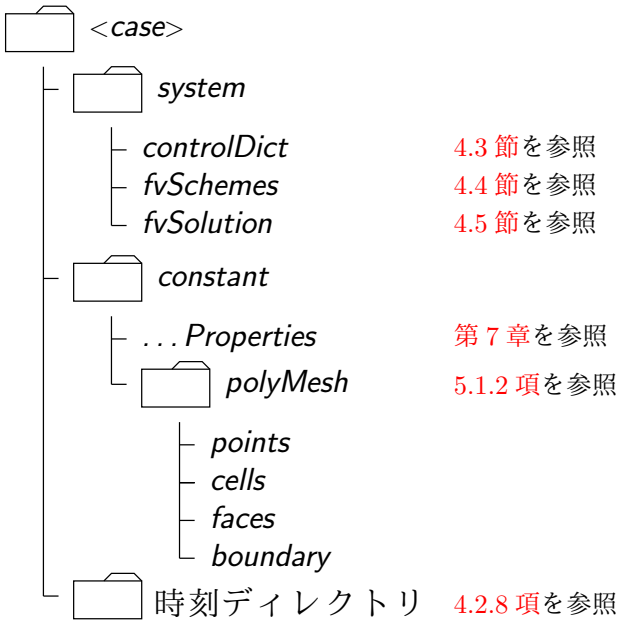


図 4.1 ケースディレクトリの構造

constant ディレクトリ サブディレクトリの *polyMesh* 内のケースメッシュと物理的性質を定めるファイルの完全な説明と関連するアプリケーション，例えば *transportProperties* を含みます．

system ディレクトリ 解析の手順そのもののためのパラメータの設定に関するディレクトリです．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を含んでいるようにコントロールを実行する *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実行のために設定する *fvSolution* です．

時刻ディレクトリ 特定領域のためにデータの個別のファイルをもっています．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 のファイルの結果として存在し得ます．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．私達は通常時間通常 $t = 0$ で私達のシミュレーションを始めて，初期の条件は指定された名前のフォーマットに依存して *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，非常にフレキシブル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に理解しやすい単純なルール類に従っているものであり，ファイルの書式は特に難しいものではなく直感的に理解できる多くのソフトウエアパッケージをもっていますが，これらについては特に記載はしておりません．OpenFOAM のファイルフォーマットの書式についての説明は次のセクションで行います．

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．

- ファイルは自由な形式をもち，不特定のどんなカラムにも割り当てられ，複数行にわたる場合の指示を指定する必要はありません．
- //のコメントデリミタで OpenFOAM は最後の行までテキストを無視しますが，その他については，行は特に意味を持っていません．
- 複数行にわたるコメントは，/*と*/で囲んで終了させます．

4.2.2 ディクショナリ

OpenFOAM は，データを特定する最も一般的な手段としてディクショナリを用いています．ディクショナリは，キーワードを用いて，I/O により読み出すことができるセットとしてデータエントリ

を含んでいます．キーワードの見出し語は一般的な形式に従います．

```
<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．ディクショナリの方式は以下のように波括弧 { } で囲まれた入力に従い，ディクショナリ名を指定します．

```
<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 もしくは領域，例：volVectorField
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

この表は，おそらくクラスの注目すべき例外はあるものの，ほとんどの入力において十分な各入力の短い説明を提供します．クラスの入力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．読み込まれるファイル呼び出す基礎的なコードの知識や OpenFOAM クラスの知識がなくては，ユーザはおそらくクラスの入力を正確に推測することはできません．しかし，単純なキーワード入力をもつほとんどのデータファイルは内部のディクショナリクラスの中に読に読み込まれ，それゆえそれらの場合，クラスの入力はディクショナリとなります．

以下の例は今のところ記載されている入力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からの解凍には二つのディクショナリ，ソルバ，PISO を含みます．ソルバディクショナリはソルバのための複数のデータ入力と p と U それぞれのキーワードによって表現される圧力方程式と速度方程式それぞれのための許容誤差を含み，PISO ディクショナリは制御アルゴリズムを含みます．

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

solvers
{
    p PCG
    {
        preconditioner   DIC;
        tolerance        1e-06;
```

```
        relTol           0;
    };

    U PBiCG
    {
        preconditioner   DILU;
        tolerance        1e-05;
        relTol           0;
    };
}

PISO
{
    nCorrectors     2;
    nNonOrthogonalCorrectors 0;
    pRefCell        0;
    pRefValue       0;
}

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

4.2.4　リスト

OpenFOAM アプリケーションは，例えば頂点のリストはメッシュ記述のために調節するリストを含んでいます．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表示された一つの数字である．vector はランク 1 の VectorSpace で 3 次元であり，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る．それゆえ，ベクトル $(1.0, 1.1, 1.2)$ は次のように書かれる．

```
(1.0 1.1 1.2)
```

OpenFOAM では，テンソルはランク 2 の VectorSpace で 3 次元であり，それゆえデータ入力はいつも九つの実数と決まっている．それゆえプログラマズガイドの 1.3.7 節で説明されている同一のテンソルは以下のように書かれる．

```
(
    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 ($kg{\cdot}m{\cdot}s^{-2}$) 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]
```

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)[1]	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

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

[1] 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．

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

そのデータはそれ自体の `dimentions` の入力から始まり，次に `referenceLevel` 値が続きます．フィールド変数は基準レベルの入力と関連した値として保存されます．基準レベルは通常 0 に設定されるが解法の正確さを改善させるために他の値に設定することもできます．これに続いて，ひとつの例として以下のような方法で書かれる `internalField` があります．

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，以下のようなフォームをとります．

```
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フォームを取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．強制的な入力，`type` はパッチのフィールドを分類するためのフィールド条件を書きます．残りの入力は選択されたパッチの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 で使えるパッチのフィールド条件は説明とそれを分類するデータともに表 5.2 と表 5.3 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．

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

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    movingWall
    {
        type            fixedValue;
        value           uniform (1 0 0);
    }

    fixedWalls
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }

    frontAndBack
    {
        type            empty;
    }
}

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

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 2 種類のディレクティブが利用可能できます．

`#include "<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．

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 $pressure
boundaryField
{
    patch1
    {
        type fixedValue;
```

```
        value $pressure;
    }
}
```

あくまでもこれはこの機能のはたらきを提示するだけの，単純でつまらない例です．この機能はケースデータを要求を満たすよう一般化する手段としてなど，多くのより便利な使い方で用いることができます．例えば同一の RSA 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，それを呼び出すことができます．適用範囲はほぼ限りないといえるでしょう．

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`†	タイムステップの `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`	周期的ベースで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
データの読み込み	
runTimeModifiable	controlDict のようないずれかのディクショナリの yes†/no スイッチは各タイムステップの始めに OpenFOAM によって再度読み込まれる．
Run-time loadable functionality	
libs	実行時にロードする (*$LD_LIBRARY_PATH* 上の) 追加ライブラリのリスト．例えば ("libUser1.so" "libUser2.so")
functions	機能のリスト．例えばランタイムにロードする probes は*$FOAM_TUTORIALS* の例を見る．

† 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．

```
application icoFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         0.5;

deltaT          0.005;

writeControl    timeStep;

writeInterval   20;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression uncompressed;

timeFormat      general;

timePrecision   6;

runTimeModifiable yes;

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

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．スキームの数学的な解説は，プログラマ・ガイドの 2.4 節を見てください．

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．

導関数項では，より一層この選択の自由が顕著です．ユーザは，まず離散化手法を選択することができますが，ここではガウスによる有限体積積分を用いるのが一般的です．ガウス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．

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`	ラプラシアン ∇^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

```
ddtSchemes
{
    default         Euler;
}

gradSchemes
{
    default         Gauss linear;
    grad(p)         Gauss linear;
}

divSchemes
{
    default         none;
    div(phi,U)      Gauss linear;
}

laplacianSchemes
{
    default         none;
    laplacian(nu,U) Gauss linear corrected;
```

```
    laplacian((1|A(U)),p) Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
    interpolate(HbyA) linear;
}

snGradSchemes
{
    default         corrected;
}

fluxRequired
{
    default         no;
    p;
}

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

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，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，そのスキームが適用される訳です．`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，ユーザに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明白に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）項のガウス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制限を行う改良版があります．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．

中心スキーム	
`linear`	線形補間（中心差分）
`cubicCorrection`	体積スキーム
`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
`QUICK`	2 次風上差分
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`	体積リミッタ
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

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，格子の界面で評価されますが，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，ガウス積分を使ってラプラシアン項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．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 次元

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．

離散化スキーム	説明
`Gauss`	1 次のガウス積分
`leastSquares`	2 次の最小二乗法
`fourth`	4 次の最小二乗法
`limited`	上記のスキームの制限バージョン

表 4.8 *gradSchemes* において使用できる離散化スキーム

`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;
```

`Gauss` キーワードは，ガウス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，次の `limited Gauss` スキームの例のように，離散化スキームの前に `limited` を付けることで選択できます．

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

4.4.4 ラプラシアンスキーム

laplacianSchemes サブディクショナリにはラプラシアン項を記述します．流体力学の中で見られる $\nabla\cdot(\rho\nabla\boldsymbol{U})$ といった典型的なラプラシアン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．先の例でのラプラス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,`

スキーム	数値的性質
corrected	無制限，2 次，保守的
uncorrected	制限，1 次，非保守的
limited ψ	補正と非補正の混合
bounded	制限スカラの一次
fourth	無制限，四次，保守的

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

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．

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，一般的なもの以外にも対流特有のものも含め，表 4.6 の中から選択します．表 4.10 に示すように，これらスキームにより数値的性質が大きく変わってきます．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	4 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 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．

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*

スキーム	説明
Euler	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

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．

```
solvers
{
    p PCG
    {
        preconditioner   DIC;
        tolerance        1e-06;
        relTol           0;
    };

    U PBiCG
    {
        preconditioner   DILU;
        tolerance        1e-05;
        relTol           0;
    };
}

PISO
{
    nCorrectors     2;
    nNonOrthogonalCorrectors 0;
    pRefCell        0;
    pRefValue       0;
}

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

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，標準のソルバに使われる大部分については標準のサブディクショナリの小さなセットがあります．これらのサブディクショナリはこのセクションの後半で説明する *solvers*，*relaxationFactors*，*PISO*，および *SIMPLE* を含んでいます．

4.5.1 線形ソルバ制御

例に挙げた最初のサブディクショナリやすべてのソルバアプリケーションのサブディクショナリは `solvers` です．それ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．強調していっておきますと特定の問題を解くために方程式とアルゴリズムを書いたアプリケーションソルバとは対照的にこのタームの線形ソルバは線形方程式の解の演算方法になります．「線形ソルバ」という言葉は以下意味が明確な場合には「ソルバ」と省略して使うこともあります．用語の文脈においていかなる曖昧さも避けたいと思います．ソルバの中の各項目の構文は特定の方程式で解かれる変数に関連するキーワードで始まります．例えば，icoFoam は速度 U と圧力 p の方程式，つまり U と p の項を解きます．変数名の後にソルバが使用するパラメータを含むソルバ名とディクショナリが続きます．OpenFOAM で使えるソルバを表 4.12 で記載します．`tolerance` を含む `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
初期条件付き共役勾配	PCG/PBiCG[2]
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

表 4.12 線形ソルバ

ソルバは左右対称のマトリクスと非対称のマトリクスを区別します．行列の対称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．より正確には，残差は，現在の解を方程式に代入して，左右両側の差の大きさを取ることによって評価されます．これはまた解析する問題のスケールにかかわらず正規化されます．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`;

ソルバの公差は解が十分正確であると考えることができるくらい小さい残差レベルにまでするべきです．ソルバの相対的な公差が初期値から最終的な解への相対的な再計算を制限します．解をソルバの

[2] PCG は対称用，PBiCG は非対称用

公差に収束させるためにはソルバの相対的公差を 0 に設定するのが一般的です．公差，tolerance，および relTol は全てのソルバによってディクショナリに定められる．

4.5.1.2 共役勾配ソルバの前提条件

ソルバディクショナリの preconditioner のキーワードにあるようなマトリクスの条件決めのためのさまざまなオプションの範囲が共役勾配ソルバにあります．preconditioners を表 4.13 に記載します．

前提条件	キーワード
対角不完全コレスキー分解 (対称)	DIC
高速対角不完全コレスキー分解 (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
前提条件なし	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は，特定の緩和法が必要です．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，マトリックス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
ガウス・ザイデル	GaussSeidel
対角不完全コレスキー分解（対称）	DIC
対角不完全コレスキー分解（対称）とガウス・ザイデル	DICGaussSeidel

表 4.14 緩和法オプション

また，公差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．緩和法によって異なったレベルのメッシュ密度で使われるスイープの数は 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 キーワードは粗さもしくは細かさのレベルによって実行速度を制御します．例えば mergeLevels 1 のように，単一のレベルで行うことは最適です．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．0.9 程度の α の値であれば多くの場合安定性が確保されます．ただし著しく低い値，例えば 0.2 は反復計算を遅くする場合等の非常に限られた値です．ユーザは最初に，ある領域に関連している word を要素として最初に指定することによって，特定の領域の緩和係数を指定できます．以下で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る緩和係数を参照できます．

```
solvers
{
    p PCG
    {
        preconditioner   DIC;
        tolerance        1e-06;
        relTol           0.01;
    };
    U PBiCG
    {
        preconditioner   DILU;
        tolerance        1e-05;
        relTol           0.1;
    };
    k PBiCG
    {
        preconditioner   DILU;
        tolerance        1e-05;
        relTol           0.1;
    };
    epsilon PBiCG
    {
        preconditioner   DILU;
        tolerance        1e-05;
        relTol           0.1;
    };
    R PBiCG
    {
```

```
        preconditioner   DILU;
        tolerance        1e-05;
        relTol           0.1;
    };
    nuTilda PBiCG
    {
        preconditioner   DILU;
        tolerance        1e-05;
        relTol           0.1;
    };
}

SIMPLE
{
    nNonOrthogonalCorrectors 0;
}

relaxationFactors
{
    p               0.3;
    U               0.7;
    k               0.7;
    epsilon         0.7;
    R               0.7;
    nuTilda         0.7;
}

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

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 は過渡状態の問題に，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，本節冒頭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成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．非直交の補正数は本節中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 `pd` であるところでは，名前はそれぞれ `pdRefValue` と `pdRefCell` 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，これまでこのセクション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タを加えて下さい．

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．プログラマ・ガイドの 2.3 節においてより詳細に述べますが，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ることだけを，ここでは述べておくことにします．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．

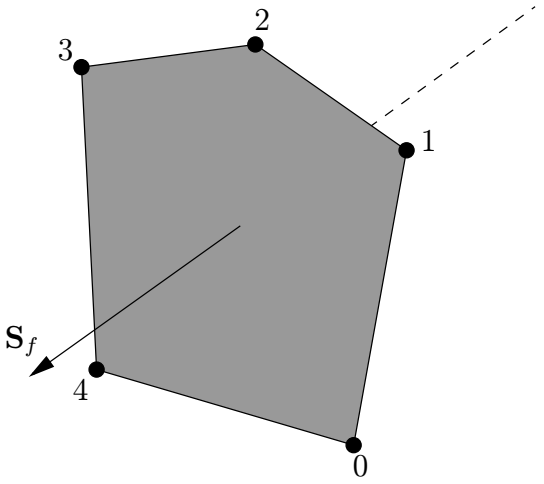


図 5.1　面における点の順序から決まる面領域ベクトル

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，既に述べましたように，内部のセル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

別の標準的（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．

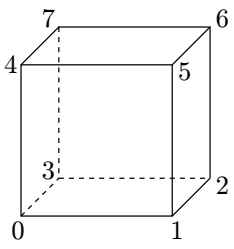
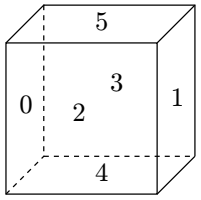
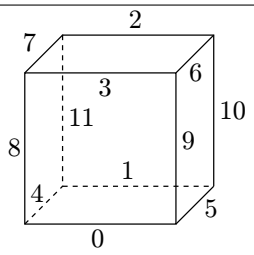
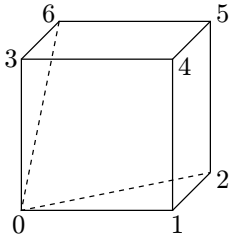
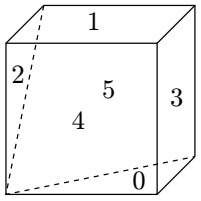
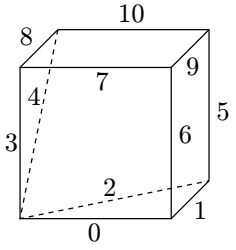
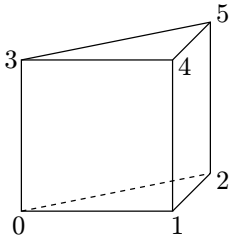
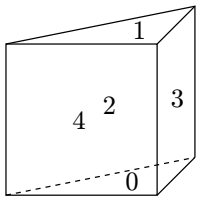
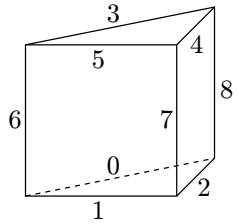
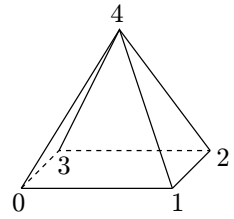
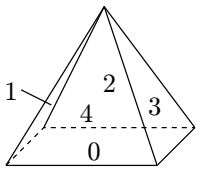
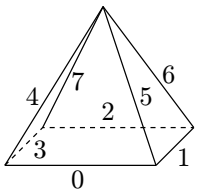
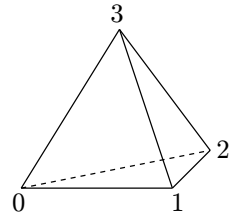
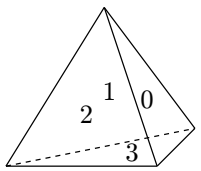
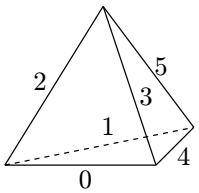
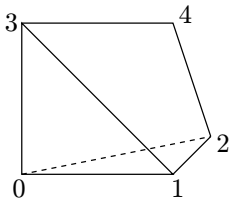
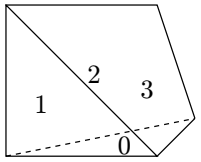
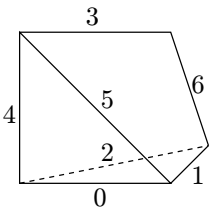
セルタイプ	キーワード	点の番号付け	面の番号付け	辺の番号付け
六面体	hex			
くさび形	wedge			
三角柱	prism			
四角錐	pyr			
四面体	tet			
くさび状四面体	tetWedge			

表 5.1 cellShapes における頂点，面，辺の番号付け

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，境界はバラバラに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 4 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

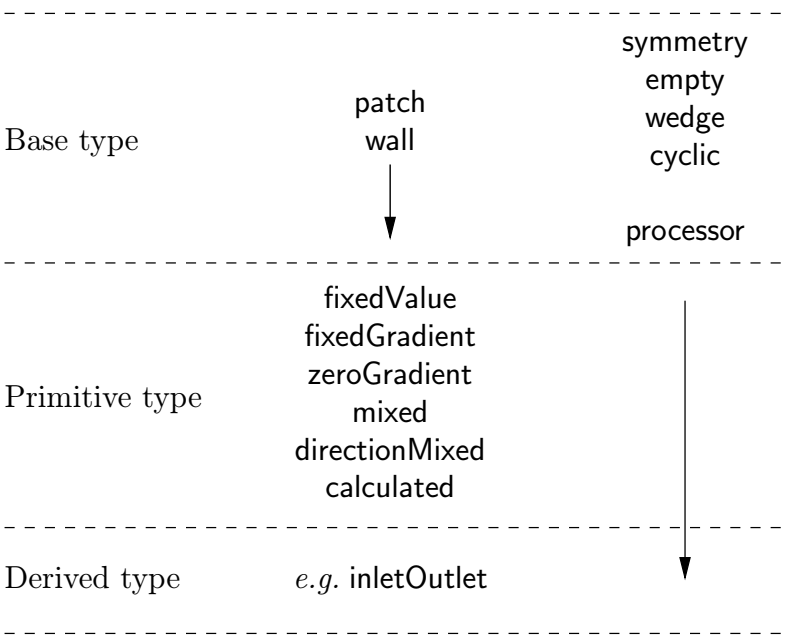


図 5.2　境界タイプの階層

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．

```
6
(
inlet
{
    type patch;
    nFaces 50;
    startFace 10325;
}

outlet
{
    type patch;
    nFaces 40;
    startFace 10375;
}

bottom
{
    type symmetryPlane;
    nFaces 25;
    startFace 10415;
}

top
{
    type symmetryPlane;
    nFaces 125;
    startFace 10440;
}

obstacle
{
    type patch;
    nFaces 110;
    startFace 10565;
}

defaultFaces
{
    type empty;
    nFaces 10500;
    startFace 10675;
}
)

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

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

internalField   uniform 1;

boundaryField
{
    inlet
    {
```

```
        type            fixedValue;
        value           uniform 1;
    }

    outlet
    {
        type            waveTransmissive;
        field           p;
        phi             phi;
        rho             rho;
        psi             psi;
        gamma           1.4;
        fieldInf        1;
        lInf            3;
        value           uniform 1;
    }

    bottom
    {
        type            symmetryPlane;
    }

    top
    {
        type            symmetryPlane;
    }

    obstacle
    {
        type            zeroGradient;
    }

    defaultFaces
    {
        type            empty;
    }
}

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

boundary ファイルにおける type には，symmetryPlane や empty といった形態的制約を受けるパッチを除くすべてのパッチに対し patch と記述されています．*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．

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	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

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

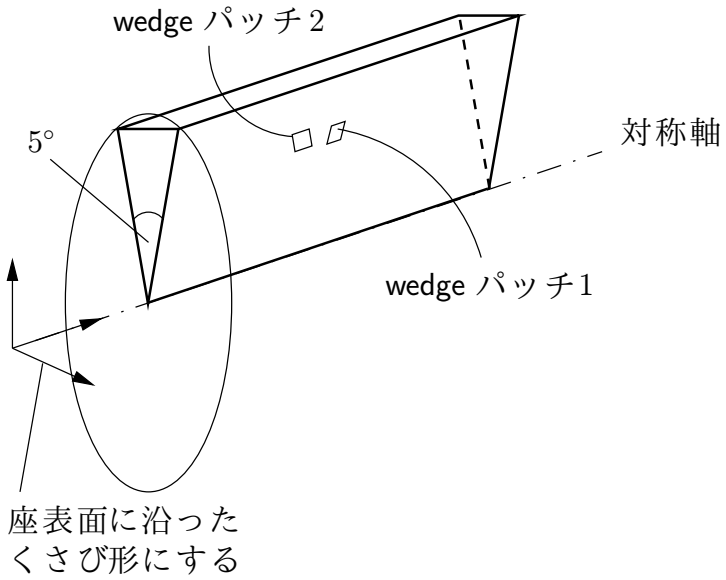


図 5.3 wedge パッチを利用した軸対象形状

(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，角度 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．単一の cyclic パッチは，faceList において面を二つに分割します．そして図 5.4 に示すように，二つの面のセットを結び付けます．面の各組は同じ領域のものである必要がありますが，同じ方向のものである必要はありません.

processor 数多くの処理の中で，計算が平行して行われている場合は，だいたい同じ数の格子を各処理が計算するために，メッシュは分けられる必要があります．メッシュの中の異なる部分間の境界は 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.

5.2.4 派生型

表 5.4 に派生型の種類を挙げます.

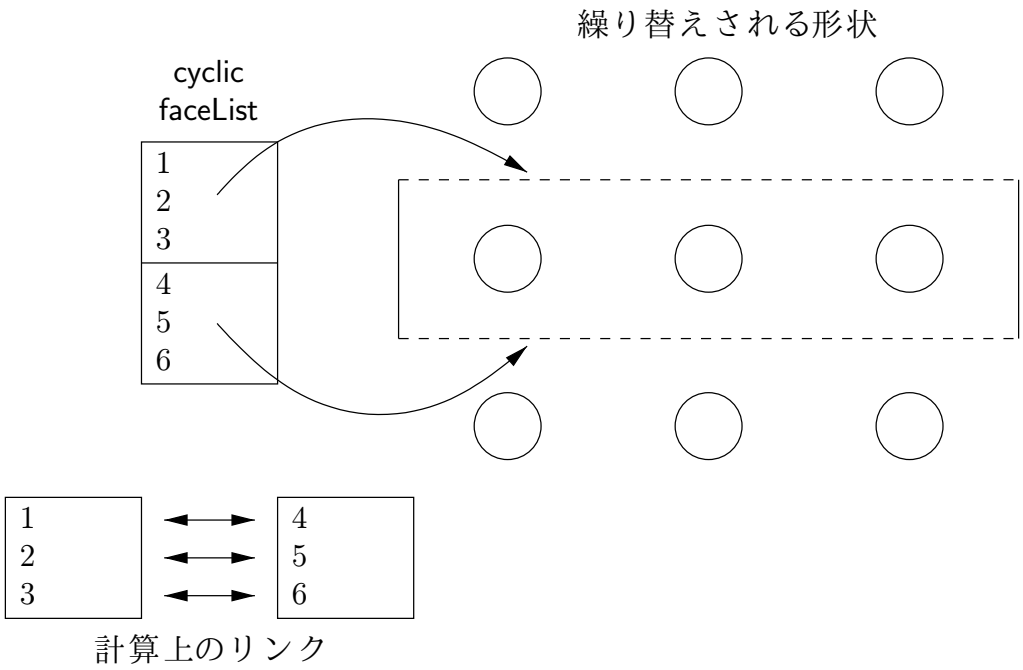


図 5.4　cyclic パッチを利用した周期境界の連続形状

種類	物理量 ϕ に対して与える条件	Data to specify
fixedValue	ϕ の値が一定	value
fixedGradient	ϕ の勾配が一定	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	パッチの法線方向に対して mixed，接線方向に対して fixedGradient	refValue, refGradient, valueFraction, value

表 5.3　基本型のパッチの種類

5.3　blockMesh ユーティリティを使ったメッシュ生成

このセクションでは，OpenFOAM とともに供給される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十分な情報です．

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．頂点はラベルを

fixedValue から派生	意味	指定するデータ
movingWallVelocity	ノーマルパッチの値を置き換えるのでパッチのフラックスは 0	`value`
pressureInletVelocity	流入口の p が分かっているとき，$\boldsymbol{U}$ は，フラックスから評価され，パッチはノーマル．	`value`
pressureDirectedInletVelocity	流入口の p が分かっているとき，$\boldsymbol{U}$ は，`inletDirection` のフラックスから計算される．	`value`, `inletDirection`
surfaceNormalFixedValue	大きさによって，ベクトル境界条件をノーマルパッチに指定します．ベクトルの +ve はドメインを指す．	`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`
wallBuoyantPressure	気圧勾配に基づく fixedGradient 圧を設定する	—
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．slip ならば 1.	`valueFraction`

Note: p は圧力, $\boldsymbol{U}$ は速度

表 5.4　派生型の種類

使用することで各頂点にアクセスできるようにリストに書かれています． 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 5.5 に示します．頂点 1 と 5 を接続する辺は，blockMesh で曲がった辺を指定できるのを読者に思いおこさせるために曲がっています．

5.3.3 項で説明されるように，1 組以上の頂点をお互いの上で潰すことによって八つ未満の頂点をもつブロックを生成することが可能です．

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．私たちの例では頂点 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 方向に動くことによって，見つけられます．
- 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 の方向に動くことで，同様に見つけられます．

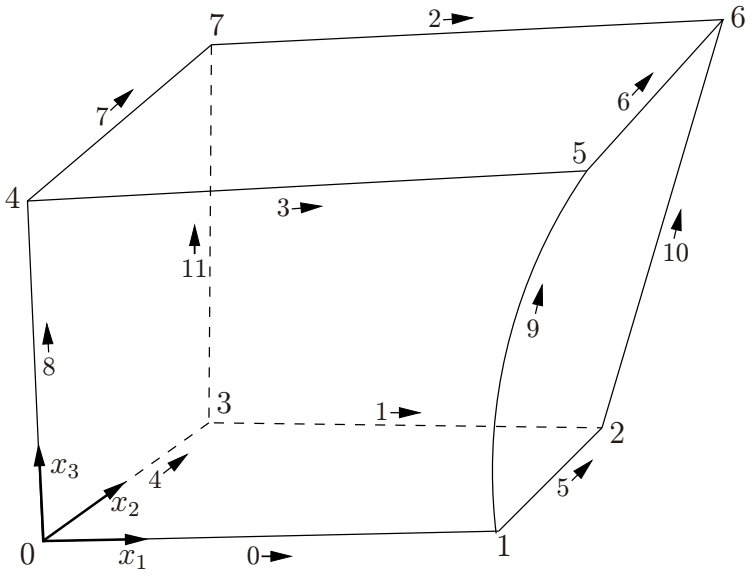


図 5.5　ひとつのブロック

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．

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))

表 5.5 *blockMeshDict* に使用するキーワード

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 5.5 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 頂点をつなぐ辺のそれぞれは，デフォルトでまっすぐであると仮定されています．しかしながら，どんな辺も，edges と名づけられるリストにおける入力で曲がるように指定されるかもしれません．そのリストはオプションです．幾何形状がどんな曲がった辺も含んでいないなら，それは省略されるかもしれません．

曲がった辺のための各入力は，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．

キーワード選択	説明	追加するエントリ
arc	円弧	円弧上の 1 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

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 5.6 での私たちの例のブロックでは，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，セクション 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-1.5/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．126 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 5.6 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 5.5 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．

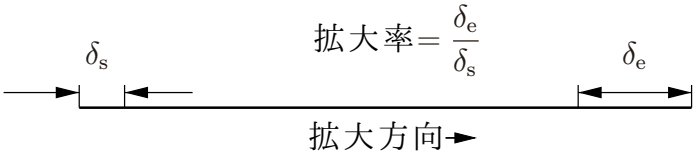


図 5.6　ブロックの辺に沿って段階わけされたメッシュ

5.3.1.4 パッチ

patches というリストでメッシュのパッチを与えます．リストにおける各パッチは以下を含む複合記入です．

- パッチタイプ，いくつかの境界条件が適用されている一般的なパッチか，表 5.1 にリストアップされていてセクション 5.2.2 項で説明される特定の幾何的条件のどちらか．
- パッチを作るブロックの面のリスト，便利にパッチを特定できる名前が推奨されるものの，名前はユーザの選択に任されます．例えば quoteTextinlet，この名前は，境界条件をフィールドデータファイルに設定するための識別子として使用されます．blockMesh は patches リストから省かれるどんな境界パッチからも面を集め，それらに defaultFaces とよばれる empty タイプからなる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 5.5 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．

```
patches // keyword
(
patch // patch type for patch 0
inlet // patch name
(
(0 4 7 3) // block face in this patch
) // end of 0th patch definition
patch // patch type for patch 1
outlet // arbitrary patch name
(
(1 2 6 5)
)
wall
walls
(
(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．

5.3.2 複数のブロック

1 ブロック以上を使用することでメッシュを作成できます．そのような事情では，メッシュは前述のテキストで説明されるように作成されます．唯一の追加設定がブロック間の接続です．そこに，二つの異なる可能性があります．

face matching あるブロックのパッチを包括する面の組が，別のブロックのパッチを包括する面の組と全く同じ位置にあるものです．

face merging あるブロックのパッチからの面のグループは，二つのブロックをつなげながら新しい内部の面の組を作成するために，別のブロックのパッチからの面の別のグループに関連づけられ

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が patches リストから単に無視されるべきです．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 5.7 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．

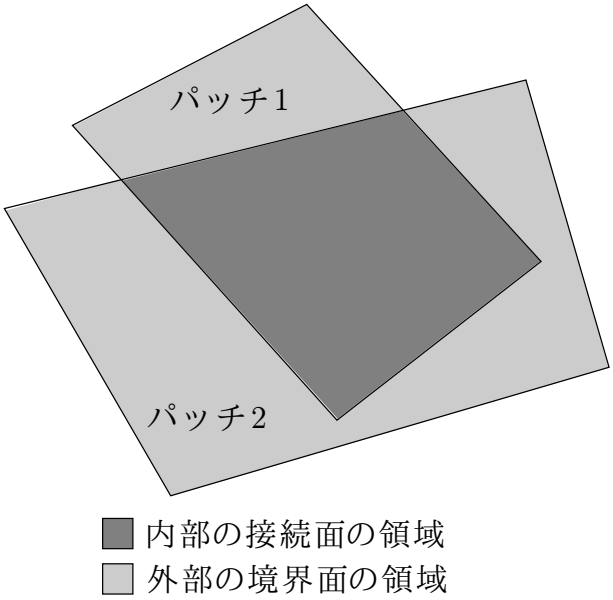


図 5.7　重なったパッチのマージ

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，セクション 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 5.8 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)
```

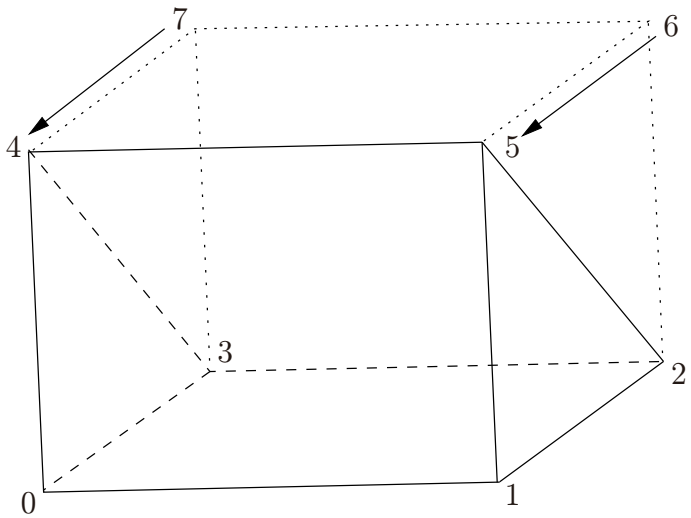


図 5.8 くさび形をしたブロックを六つの接点で作る

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，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
```

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．はじめのメッシュの細分化と，後に現れる分割の六進法のメッシュの表面形状への変形を繰り返して表面形状に近づいていきます．オプションとして，現れたメッシュを縮小させ，レイヤセル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．

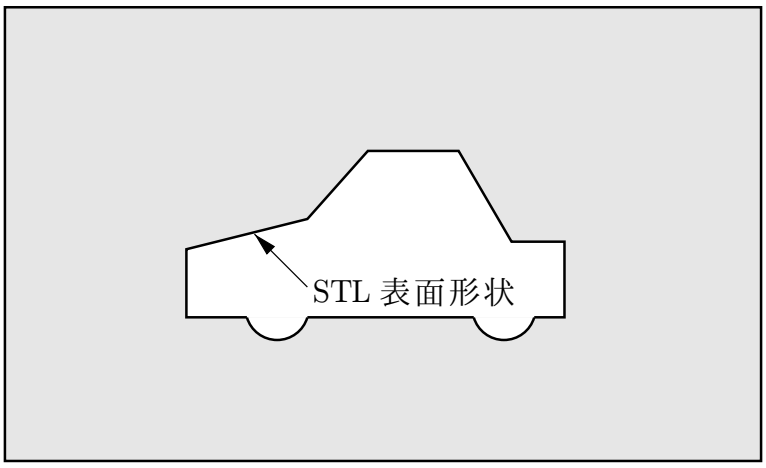


図 5.9　snappyHexMesh における 2 次元メッシュ問題の概略図

5.4.1　snappyHexMesh によるメッシュ生成の過程

図 5.9 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 *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．

キーワード	意味	例
castellatedMesh	ギザギザのメッシュを作成するかどうか	true
snap	表面のスナップの有無	true
doLayers	レイヤの追加の有無	true
mergeTolerance	初期メッシュの有界ボックスの比として許容値をまとめる	1e-06
debug	中間メッシュと画面プリントの記述の制御	
	最終メッシュのみ記述	0
	中間メッシュの記述	1
	後処理のため cellLevel を付けた volScalarField を記述	2
	.obj ファイルとして現在の交点を記述	4
geometry	表面に使用した全ての幾何学のサブディクショナリ	
castellatedMeshControls	城壁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	メッシュ特性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

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 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 5.10 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．

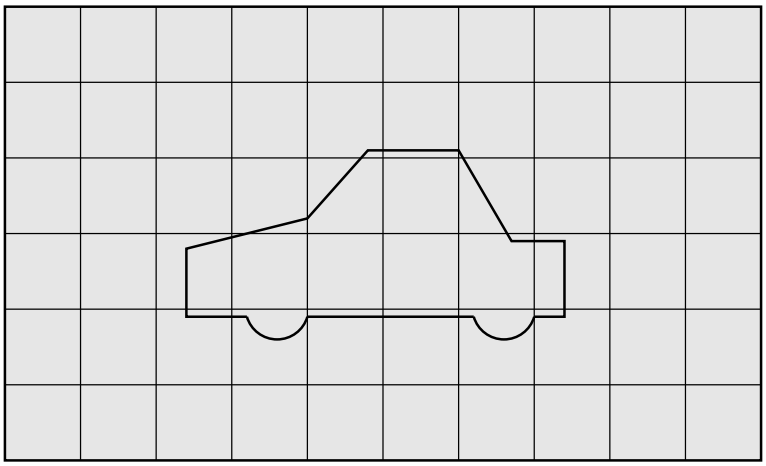


図 5.10　snappyHexMesh 実行前の基礎メッシュの生成

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 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`	細分化に対する機能リスト	
`refinementSurfaces`	細分化に対する表面ディクショナリ	
`refinementRegions`	細分化に対する領域ディクショナリ	

表 5.8　*snappyHexMeshDict* の *castellatedMeshControls* サブディクショナリのキーワード

図 5.11 で示されたように，最初に領域内で指定された輪郭に従って選択されたセルで分割が開始します．

`castellatedMeshControls` サブディクショナリの輪郭のリストにおいて *edgeMesh* ファイルの名前と細分化のレベルを記述します．

```
features
  (
      {
          file "someLine.eMesh"; // file containing edge mesh
          level 2;               // level of refinement
      }
  );
```

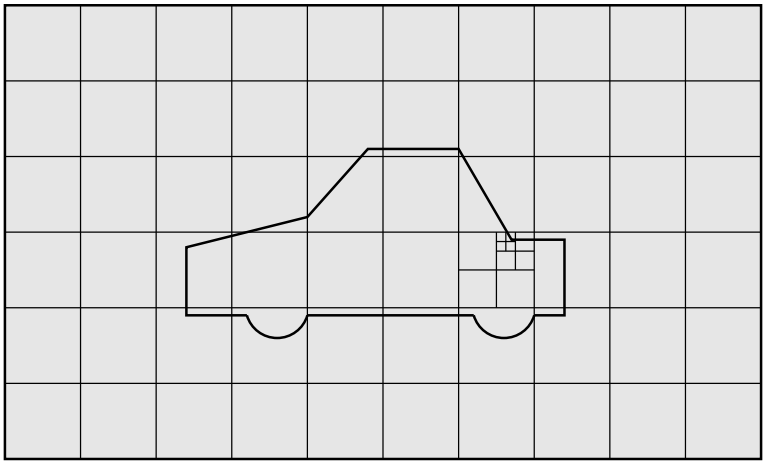


図 5.11 snappyHexMesh の輪郭によるセルの分割

輪郭の細分化に続き，図 5.12 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．

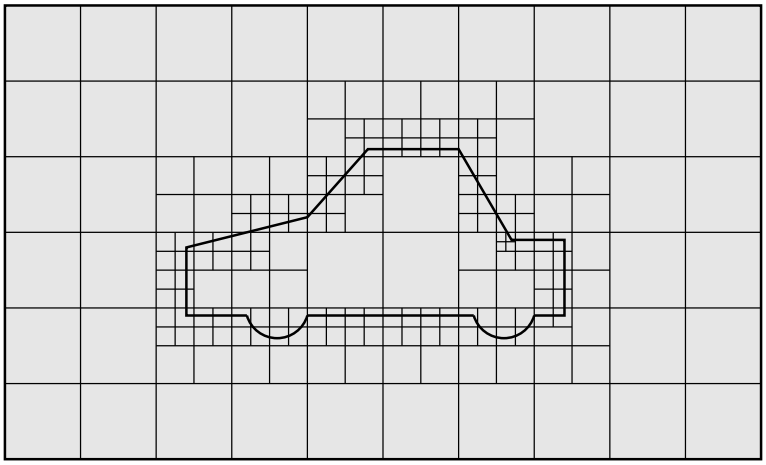


図 5.12 snappyHexMesh の表面によるセルの分割

`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 5.13 に示すように除去されます．

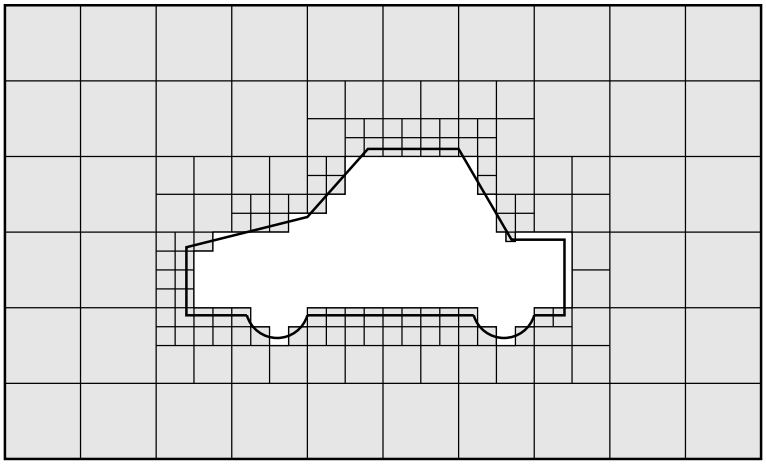


図 5.13　snappyHexMesh におけるメッシュの除去

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 5.14 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
    }
}
```

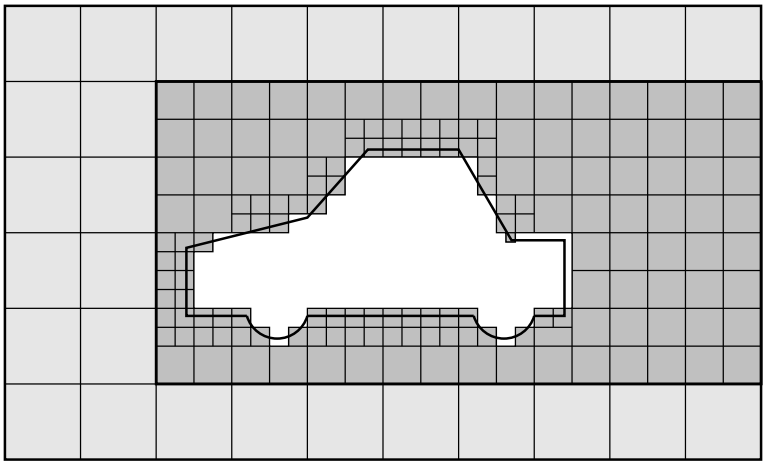


図 5.14 snappyHexMesh の領域によるセルの分割

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．

キーワード	意味	例
nSmoothPatch	表面との一致に至る前に行うパッチの平滑化の回数	3
tolerance	局所的な輪郭の最大長さに対する点と表面の距離の比の許容範囲	4.0
nSolveIter	メッシュの置き換え時の緩和計算の回数	30
nRelaxIter	メッシュのスナップ時の緩和計算の最大回数	5

表 5.9 snapControls のキーワード

図 5.15 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．）

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 5.16 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．

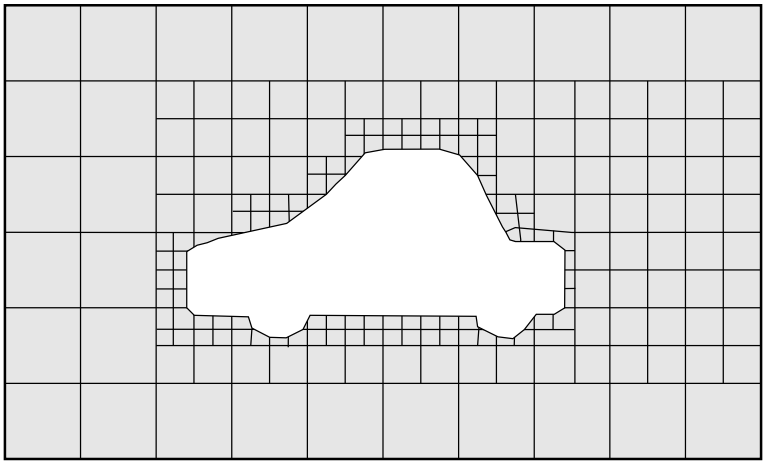


図 5.15　snappyHexMesh における表面のスナップ

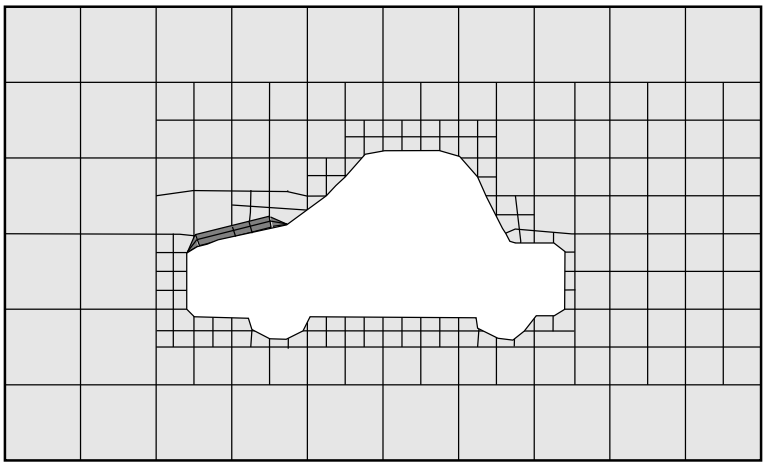


図 5.16　レイヤの挿入

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．

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．パッチ名は，レイヤ追加が表面幾何形状ではなく既存メッシュに関連付けられるので使われ，したがって，表面領域ではなく，パッチに適用されます．レイヤの入力例は以下のとおりです．

```
layers
 {
     sphere.stl_firstSolid
     {
```

キーワード	意味	例
layers	レイヤのディクショナリ	
expansionRatio	レイヤメッシュの拡大比率	1.0
finalLayerRatio	レイヤの外側の歪められていないセルの大きさに比例した壁から最も遠い層の厚さ	0.3
minThickness	レイヤの外側の歪められていないセルの大きさに比例したセルのレイヤの最小の厚さ	0.25
nGrow	点がなければ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	緩和反復のスナップ最大数	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減少する	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

表 5.10 *snappyHexMeshDict* の *addLayersControls* サブディクショナリのキーワード

```
        nSurfaceLayers 1;
    }
    maxY
    {
        nSurfaceLayers 1;
    }
}
```

キーワード	意味	例
maxNonOrtho	非直交性上限角．180 は不可	65
maxBoundarySkewness	境界面ひずみ上限値． < 0 は不可	20
maxInternalSkewness	内部面ひずみ上限値． < 0 は不可	4
maxConcave	凹み上限角．180 は不可	80
minFlatness	実際の領域に対する最小の投影面積比率．-1 は不可	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）は不可	1e-13
minArea	最小面領域． < 0 は不可	
minTwist	最小面ねじれ． < -1 は不可	0.05
minDeterminant	最小正常セルの行列式．1 = hex， ≤ 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 計算可能性では > 0	
nSmoothScale	エラー分布反復数	4
errorReduction	エラー点の置換のための減少量	

表 5.11 *snappyHexMeshDict* の *meshQualityControls* サブディクショナリのキーワード

5.4.8 メッシュの水準

メッシュの水準は *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．メッシュ変換コードは，命名規則に従います．利用可能なメッシュコンバータは以下の通りです．

fluentMeshToFoam　は，2 次元，3 次元両方の場合に動く Fluent `.msh` メッシュファイルを読みます．
starToFoam　は，STAR-CD PROSTAR のメッシュファイルを読みます．
gambitToFoam　は，GAMBIT `.neu` のニュートラルファイルを読みます．
ideasToFoam　は，ANSYS `.ans` 形式で書かれた I-DEAS メッシュを読みます．
cfxToFoam　は，`.geo` 形式で書かれた CFX メッシュを読みます．

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 のデフォルトの選択ではありません．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，ご要望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るでしょう．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るでしょう．
- Fluent はメッシュの内部にパッチを定義することをユーザに許しています．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るでしょう．
- 現在，埋め込まれたインタフェースと細分化のツリーに関するサポートは全くありません．

Fluent `.msh` ファイルの変換は，必要なディレクトリとファイルを作成することによってまず新しい OpenFOAM ケースを作ることで始まります．ケースディレクトリは *system* のサブディレクトリに *controlDict* ファイルを含みます．そしてコマンド・プロンプトにおいて，ユーザは以下を実行することになります．

```
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

このセクションは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書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ただ一つの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．
- スライドするインターフェース

複数の流れのメッシュに関しては，メッシュ変換は，別々のメッシュとしてそれぞれの個々の流れを書くことによって実現され，OpenFOAMでそれらを組み立て直すことができます．

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．以下のセクションは，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．これからのセクション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

ユーザはstarToFoamの変換を試みる前に，STAR-CDのメッシュをチェックするツールを動かすべきです．そして，変換の後に，checkMeshユーティリティは新たに変換されたメッシュで実行されるべきです．あるいはまた，starToFoamはユーザが問題のあるセルをより近くで見ることができるようにするためのPROSTARコマンドを含む警告を発行するかもしれません．問題の多いセルとマッチは，OpenFOAMを用いてメッシュを使おうとする前に，チェックされ修正されるべきです．無効なメッシュはOpenFOAMで動きませんが，それが正当性評価基準を課さない別の環境では動くかもしれないということを覚えていてください．

コンバータにおいて許容度を合わせることで，許容度のマッチングに関するいくつかの問題を克服できます．しかしながら，有効性への限界があり，デフォルトレベルからマッチング許容度を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．これがそうでないなら，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，頂点も取り除かれる必要があるでしょう．

```
CSET NEWS FLUID
VSET NEWS CSET
VSET INVE
```

これらの必要とされていない頂点を取り除く前に，必要とされていない境界面は，除かれる前に，集められ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，必要とされていない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に関連づけられていないどんな境界面に対して壁境界を適用します．残っている境界面は，割り当てられた境界タイプ 0 として `default` 境界領域に集められます．OpenFOAM は，人為ミスを誘発するので，意図的に未定義の界面のための `default` 境界条件の概念をもっていません．例えば，すべての関連付けられていない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．default 境界は，以下で説明された手順を用いることで実体をもつものに変えられる必要があります．

1. `Wire Surface` オプションで幾何学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余分な境界領域を定義してください．そして，すべての見えている面を，境界ツールでゾーンオプションを選択して，モデルのスクリーンに描かれている全体の周りに多角形を描くことによって，10 といった新しい領域に加えてください．PROSTAR の以下のコマンドを発行することによって，これができます．

   ```
   RDEF 10 WALL
   BZON 10 ALL
   ```

3. 私たちはセットからすべての以前に定義された境界タイプを外すことになるでしょう．境界領域に行ってください．

   ```
   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 と共に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．これはコマンドを発行し，境界に対して PROSTAR を用いることで行われます．

```
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される必要があります．

```
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される必要があるでしょう．

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，以下のとおりであるかもしれません．

```
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かもしれません．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，およびポイント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

指定されるべきです．

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 cfxToFoam

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 次元幾何形状は，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 として設定されるべきです．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．

```
cfxToFoam <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 5.17 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 5.17 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．

```
patchMap
(
```

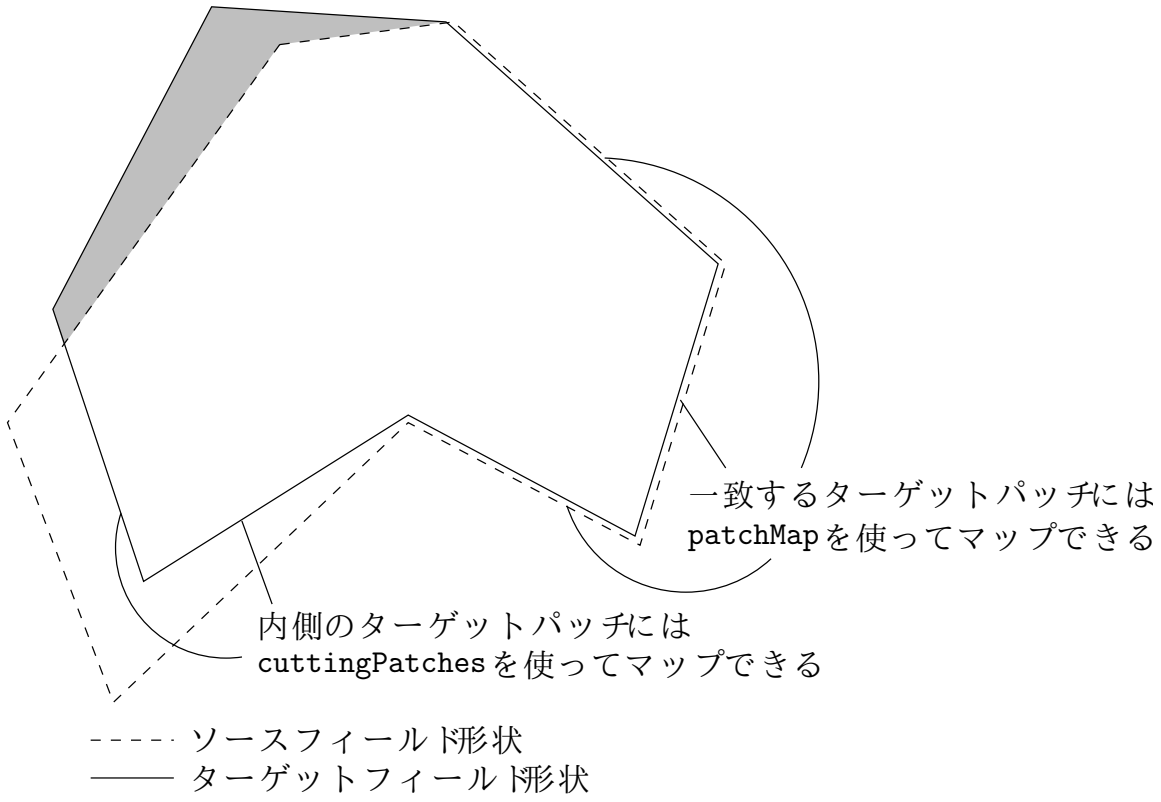


図 5.17 一貫しないフィールドをマップする

```
lid movingWall
);

cuttingPatches
(
fixedWalls
);

// ************************************************************************* //
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 AVS/Express 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である ParaView で走る読み込みのモジュールです．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 3.3 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．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走るはずですが，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に 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，このことは，VTK ベースの画像ツールであれば，OpenFOAM の case の後処理として使えることを意味しています．このことは，OpenFOAM で ParaView を使うことの代替法を提供しています．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，OpenFOAM で提供されている読み込みモジュールを用いて，ParaView を立ち上げる厳密なスクリプトです．他の OpenFoam のユーティリティ同様に，ルートディレクトリのパスまたは -case オプションと，引数としてのケース名を入力して実行されます

```
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黄色にハイライトされ，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描画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．

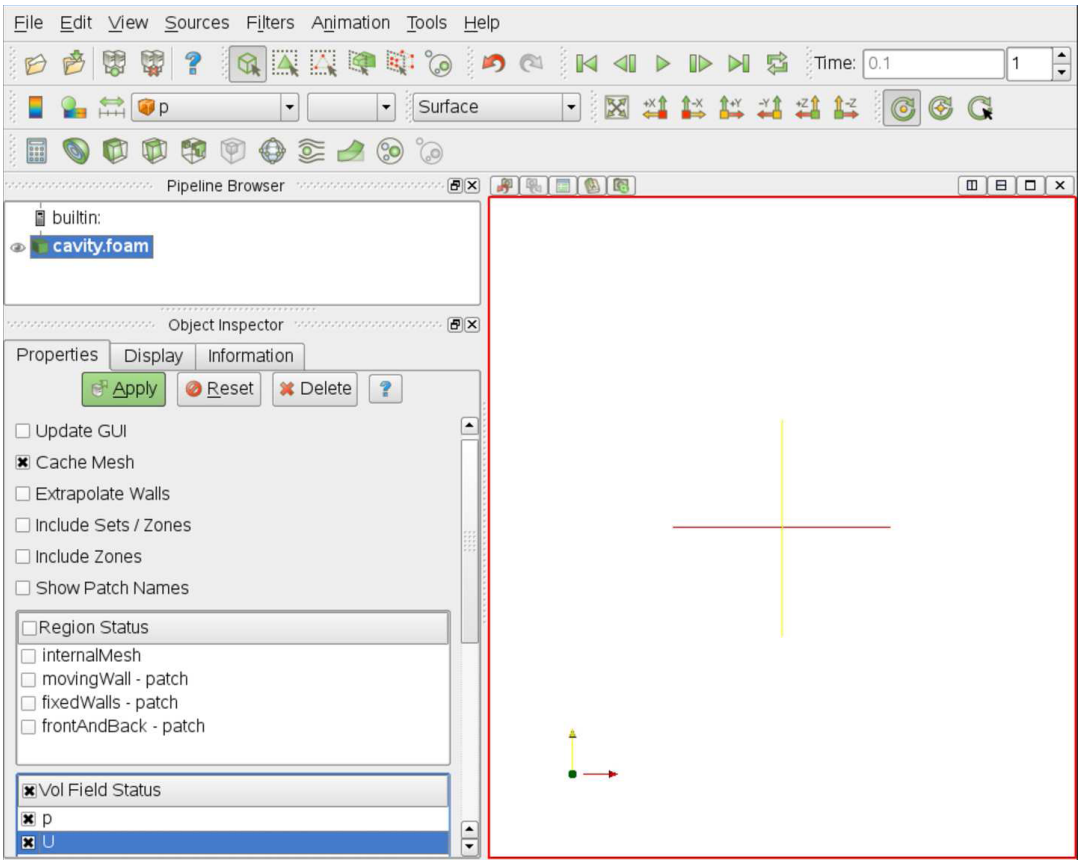


図 6.1 paraFoam の画面

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．

システムの一般的な操作は選択をし，Properties パネルの緑の Apply ボタンをクリックすることを基本としています．追加項目ボタンとしては，必要に応じて GUI のリセットを行う 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 Accept をクリックする必要があります．この Accept 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．

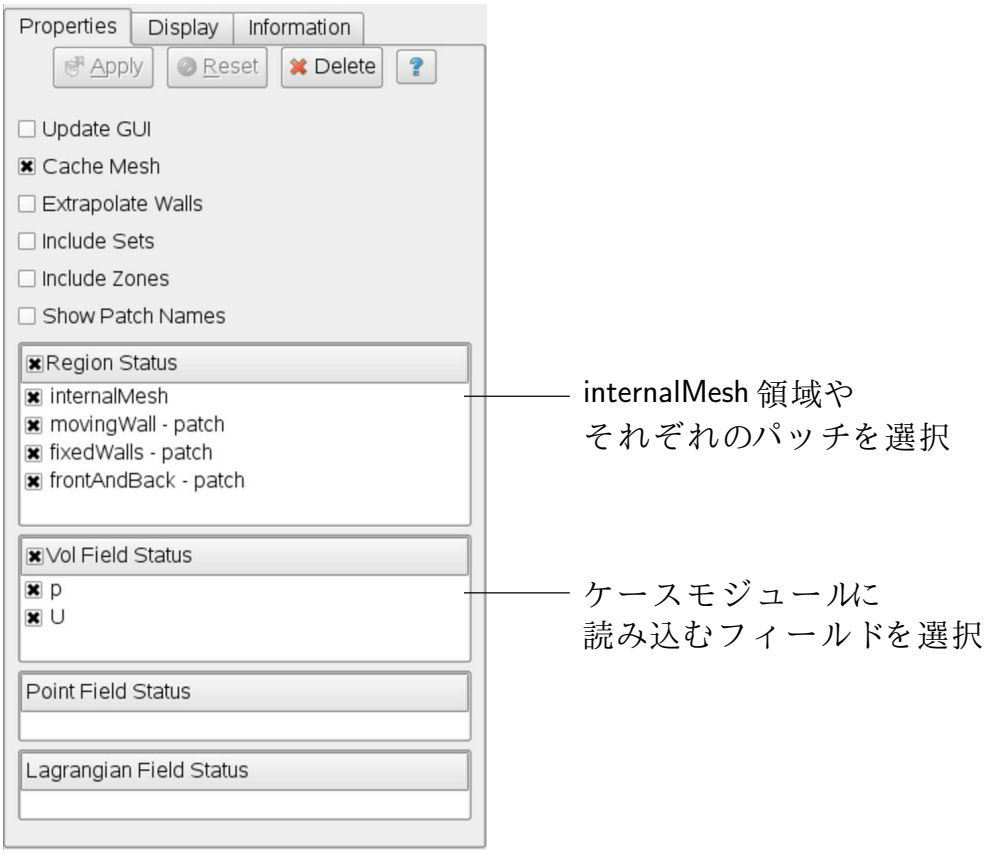


図 6.2 ケースモジュールのプロパティパネル

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 Solid Color ウィンドウで指定することにより可視化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 ビューの操作

本セクションでは，paraFoam におけるオブジェクトのビュー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 View settings

`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 Render View Options ウインドウが表示されます．General には開始時に設定すべき以下の項目があります．

- 背景色，印刷物には白が望ましい
- CFD，特に 2 次元のケースでは Use Parallel Projection（平行投影）が通常用いられる

Lights には Light Kit パネルに光源の詳細設定があります．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 にすれば鮮明な色の画像を得られるでしょう．

Annotation では，ビュー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 General settings

Settings を Edit メニューから選択すると General と 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をマウスと 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 constant value を表す 2D 表示（同一面：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 cutting plane の使い方

まれに，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あります．このためには，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 フィルタにより，ユーザはそれぞれ center と normal/radius を使って，Slice Type メニューの中に Plane，Box または Sphere のカッティングを指定することができます．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．

その後，コンタのラインを作成するために，切断された面で Contour フィルタを実行することができます．

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．フィルタは Vectors で選択されたフィールドを読み込み，Arrow によりクリアなベクトル画像を提供するための Glyph Types のレンジを用意します．それぞれのグリフは，ユーザが最も効果的にパネルをコントロールするために選択されています． Promerties パネルのリマインダには，グリフのための主要な Scale Mode メニューがあります．その中でも最もよく使うオプションは，ベクトルの大きさに比例したグリフの長さの Vector，各々のグリフの長さが同じにする Off，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，デフォルトにより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．この場合には，最初に case モジュールに対して Cell Centers を適用し，その後セルの中心の計算結果のために 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，くどいようですが，これには計算コストがかかります．

6.1.9 画像の出力

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，解像度を幅 1000 ピクセル以上にするとよいでしょう．A4 サイズの書面や PDF の図として，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，動画プレイヤー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 animation というケースの 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，二つのファイルに制御されます．まず，*controlDict* ディクショナリは，`startTime` を設定し，変換される結果のセットを与えます．これは，変換される結果のセットを与えます．もしあなたが，最新の結果を変換したければ，`startFrom` を `latestTime` と設定することができます．`translation` を指定する 2 番目のファイルは，*foamDataToFluentDict* ディクショナリです．このファイルは，*constant* ディレクトリに置かれています．*foamDataToFluentDict* ディクショナリの例を以下に示します．

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      / F ield          | OpenFOAM: The Open Source CFD Toolbox           |
| \\    /  O peration       | Version: 1.5                                    |
|  \\ /    A nd             | Web:      http://www.OpenFOAM.org               |
|   \\/    M anipulation    |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      foamDataToFluentDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

p               1;

U               2;

T               3;

h               4;

k               5;

epsilon         6;

gamma           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	共通 OpenFOAM 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．

```
interpolationScheme cellPoint;

setFormat      raw;

sets
(
   leftPatch
   {
```

```
        type            uniform;
        axis            y;
        start           (0 0.5 0.25);
        end             (0 2 0.25);
        nPoints         100;
    }
);

surfaces
();

fields
(
    sigmaxx
);

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

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表面から補間
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
	null	出力しない
surfaceFormat	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
	U.component(0)	成分 0 を出力．つまり U_x
	U.component(1)	成分 1 を出力．つまり U_y
	mag(U)	ベクトル，テンソルの大きさを出力．つまり $\|\boldsymbol{U}\|$
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

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

`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．

`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

		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表面の交点	•	•	•	•		
midPoint	線・面の交点と交点の中点	•	•	•	•		
midPointAndFace	中点および面	•	•	•	•		
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

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
```

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

```
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

残差や反復回数，クーラン数などがレポートデータとしてスクリーンを横切るのをモニタすれば十分でしょう．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 -h 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，クーラン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．

- クーラン数，*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*

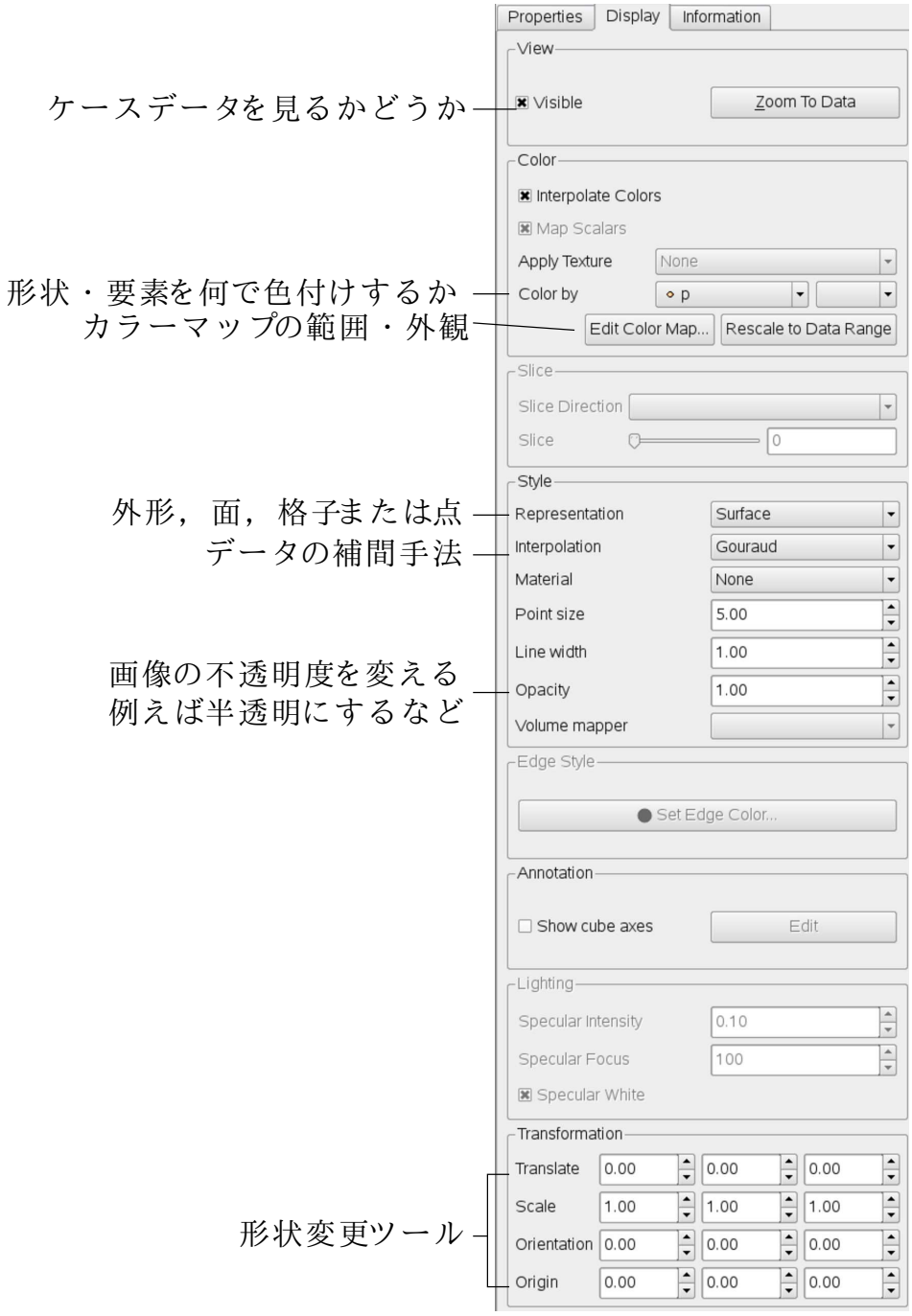


図 6.3　Display パネル

図 6.4　ParaView のツールバー

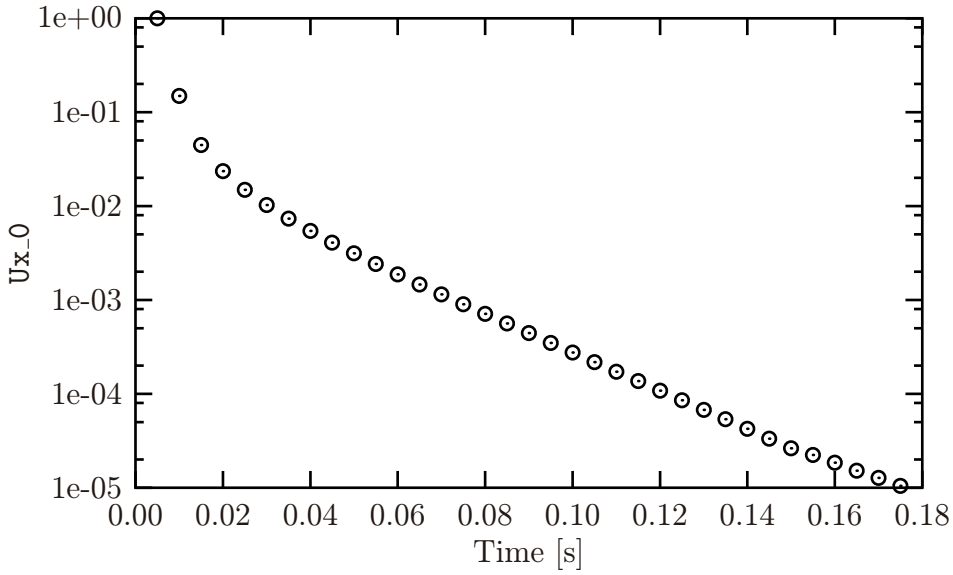


図 6.5 *cavity* チュートリアルにおける U_x の初期残差

第7章
モデルと物性値

OpenFOAM は特定のクラスの問題に対し，各々に対応した幅広い範囲のソルバを内蔵しています．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．ソルバーは，一般的には 3.5 節に，ケースに対して適切なソルバが見つけやすいように記述してありますので，この中から選択して下さい．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	
perfectGas	理想気体状態方程式
標準熱特性 — thermo	
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
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
sutherlandTransport	温度依存する輸送輸送のためのサザーランドの公式
混合特性 — mixture	
pureMixture	不活性ガス混合の一般熱モデル計算
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
dieselMixture	f_{t} と f_{u} に基づく混合気燃焼
multiComponentMixture	複数の成分に基づく混合気燃焼 [**]
chemkinMixture	CHEMKIN 熱力学と反応スキームデータベースファイルを用いた混合気燃焼

熱モデル — thermoModel	
hThermo	エンタルピ h に基づく一般熱モデル計算
hMixtureThermo	混合気燃焼のエンタルピ計算
hhuMixtureThermo	未燃ガスと混合気燃焼のエンタルピ計算

表 7.1　熱物性モデリングの階層

`thermoType` のエントリは，次の形式をとる．

```
thermoModel<mixture<transport<specieThermo<thermo<equationOfState>>>>>
```

それで，次に示すのは，`thermoType` のエントリの例である．

```
hThermo<pureMixture<constTransport<specieThermo<hConstThermo<perfectGas>>>>>
```

7.1.1　熱物性データ

基本的な熱物理の性質は，入力データから，各々の種 (species) のために指定される．そのデータは，種のために，次に示す形式の複合的なエントリを使って指定される．この複合的なエントリは，キーワード `mixture` が使われる．

```
mixture <specieCoeffs> <thermoCoeffs> <transportCoeffs>
```

物性係数 `<specieCoeffs>` は，表 7.2 にリストされるエントリを含み，その順番は入力で指定される順番となる．

説明	入力
文字列名	e.g. `mixture`
この種のモル数	n_{moles}
分子量	W (kg/kmol)

表 7.2　物性係数

熱物理の係数 `<thermoCoeffs>` は，見かけ上，他の性質がそれから導出される，比熱容量を評価することに関連している．現在の `thermo` モデルは，以下に示す通りである．

hConstThermo　一定の c_p と融解熱 H_{f} を仮定する．これらは，単純に `<specieCoeffs>` の後に二つの値 c_p と H_{f} を続けて指定する．

janafThermo　c_p を温度の関数として計算する．このとき，一連の係数は，熱力学の JANA 表のものを用いる．順序づけられた係数のリストを，表 7.3 に示した．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る．係数のセットが二つ指定される．最初のセットは常温 T_{c} 以上の温度についてのものである．(そして，T_{h} 以下である．二つめのセットは T_{c} より低く T_{l} より高い範囲についてのもの)．c_p を温度の関数として表すと，

$$c_p = R((((a_4T + a_3)T + a_2)T + a_1)T + a_0) \tag{7.1}$$

加えて，a_5，a_6 という積分定数がある．これらは，それぞれ，h と s を評価するために使われ，高温と低温の両方で同じ定数が用いられる．

説明	入力
下限温度	T_l (K)
上限温度	T_h (K)
常温	T_c (K)
高温度係数	$a_0 \ldots a_4$
高温度エンタルピ補正	a_5
高温度エントロピ補正	a_6
低温度係数	$a_0 \ldots a_4$
低温度エンタルピ補正	a_5
低温度エントロピ補正	a_6

表 7.3 JANAF 熱力学的係数

移動係数 `<transportCoeffs>` は，動的粘性率，熱伝導度，層流熱伝導度（エンタルピ方程式のため）を評価するために使われる．現在の transport モデルは，以下に説明する通りである．

constTransport μ とプラントル数 $Pr = c_p\mu/\kappa$ が一定であると仮定する．ここでは，`<thermoCoeffs>` の後に μ と Pr の二つの値を続けて指定する．

sutherlandTransport μ を温度 T の関数として計算する．これには，サザーランド係数 A_s とサザーランド温度 T_s を用いる．この二つの数は，`<thermoCoeffs>` の後に続けて指定する．μ は，次のように計算される．

$$\mu = \frac{A_\mathrm{s}\sqrt{T}}{1 + T_\mathrm{s}/T} \tag{7.2}$$

次は，`fuel` という名前の種についてのエントリの例である．これは，sutherlandTransport と janafThermo を使ってモデルされている．また，エントリの説明のためにコメントが入っている．

```
fuel                                            // keyword
fuel 1 44.0962                                  // specie
200 5000 1000                                   // -- janafThermo --
7.53414 0.0188722 -6.27185e-06 9.14756e-10 -4.78381e-14
-16467.5 -17.8923
0.933554 0.0264246 6.10597e-06 -2.19775e-08 9.51493e-12
-13958.5 19.2017                                // -----------------
1.67212e-06 170.672;                            // sutherlandTransport
```

次に示すのは，`air` という名前の種についての，エントリの例である．これは，constTransport と hConstThermo でモデルされている．エントリの説明のためにコメントが入っている．

```
mixture         // keyword
air 1 28.9      // specie
1000 2.544e+06 // hConstThermo
1.8e-05 0.7;    // constTransport
```

7.2 乱流モデル

RASProperties ディクショナリは，乱流モデルとして Reynolds-averaged stress (RAS) モデルを使っているソルバが読み込みます．*LESProperties* ディクショナリは，乱流モデルとして large-eddy simulation (LES) モデルを使っているソルバが読み込みます．`RASProperties` に必要なエントリは，表 7.4 に示します．また，*LESProperties* ディクショナリのエントリは，表 7.5 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
`<RASModel>Coeffs`	各 RASModel における係数
`wallFunctionCoeffs`	壁関数の係数

表 7.4　*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`	各 LES モデルにおける係数
`<delta>Coeffs`	各デルタモデルにおける係数
`kappa`	フォン・カルマン定数
`wallFunctionCoeffs`	壁関数の係数

表 7.5　*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．関連するケースの *turbulenceProperties* ディクショナリを見て，各々のモデルのために必要とされる係数のリストとデフォルト値を参考にしてください．必要とされる係数は，RAS モデルが非圧縮性か圧縮性かにより，あるいは LES モデルが等容変化か非等容変化かによって異なります．参考として，それらのモデルはそれぞれ，*$FOAM_TUTORIALS* の中にある以下の例題ケースの `RASProperties` か *LESProperties* ディクショナリに示しています．

turbFoam/cavity　非圧縮性 RAS モデル
sonicTurbFoam/prism　圧縮性 RAS モデル
oodles/pitzDaily　等容 LES モデル
Xoodles/pitzDaily　非等容 LES モデル

付録A

FoamXケースマネージャ（v1.5では廃止）

OpenFOAMは，ケースの実行を管理するためのFoamXユーティリティと一緒に提供されています．FoamXはGUIであり，ほとんどの場合はローカルマシンのケースの管理に使われていますが，インターネットのようなネットワーク上に割り当てられたケースを管理することもできます．

本章は主に，FoamXに関する参考資料となっており，A.3節とA.4節でFoamXの一般的な使い方に関する有効なアドバイスを提示していますが，新規のユーザはFoamXの使い方を習得するために，最初にチュートリアル（第2章）を参照して下さい．

ネットワーク上でケースを実行するメカニズムは，他のマシンのJAVA GUIから呼び出すことができるサービスを提供するホストマシンをもつことによって成立しています．JAVA GUIとこれらサービス (ホストブラウザ，ケースブラウザ，およびケースサーバ．C++で書かれています．) の間のインタフェースは，Common ObjectRequest Broker Architecture (CORBA) を実装しているMICOです．ローカルのマシンでケースを管理するだけの場合には，そのマシンからホストブラウザとJAVA GUIの両方を立ち上げることができます．

以下のセクションにおいては，これをノーマルモードとすることにします．オプション類を以下に要約します．

ローカルでホストブラウザが実行（ノーマルモード）　この場合には，ユーザはrunFoamXを実行させることにより，ホストブラウザとGUIの両方を立ち上げることができます．

```
runFoamX
```

リモートでホストブラウザが実行（リモートモード）　この場合には，最初にrunFoamXHBによって，ホストブラウザがホストマシン上で立ち上げられます．

```
runFoamXHB
```

そして実行中のホストブラウザに接続するrunFoamXを実行することにより，GUIがローカルで立ち上げられます．

```
runFoamX
```

これらのオプションの中に含まれているプロセスを図A.1に示します．runFoamXが実行されるとき，runFoamは実行中のホストブラウザを探します．どれか一つが実行中であれば，すなわち先にrunFoamXHBが立ち上がっていれば，runFoamXはこれに接続します．そうでなければ，自分自身がホストブラウザを起動します．A.1節，A.2節，A.3節では，FoamXの一般的な操作方法が，ネット

ワーク上でどのように操作されるかを特に強調しながら述べられています．続いて A.4 節では，ケースサーバを通じた OpenFOAM のケースの実行について述べられ，A.5 節では FoamX に関係した設定について述べられています．

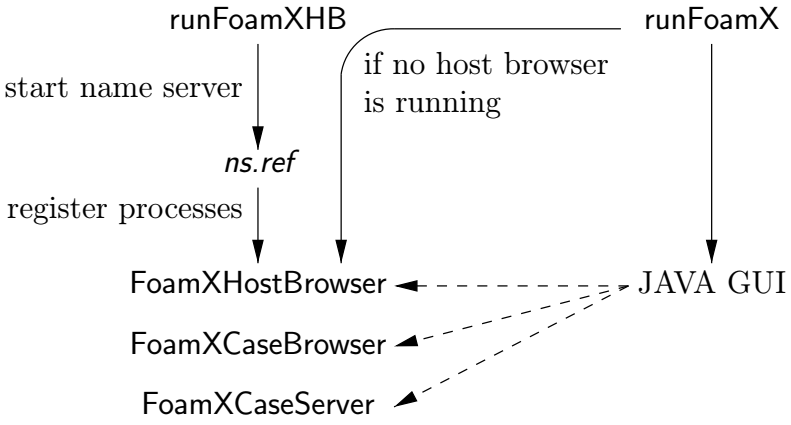
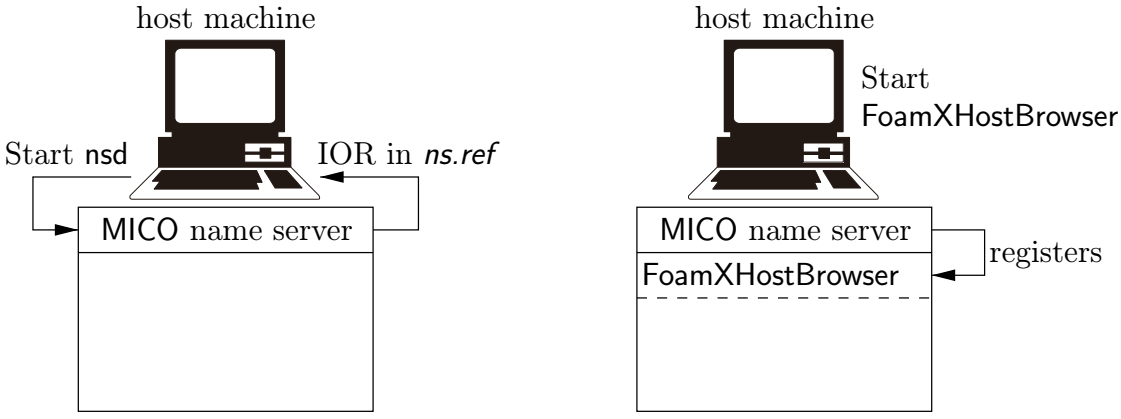


図 A.1 FoamX 実行のオプション

A.1 ネームサーバとホストブラウザ

ホストマシン上で FoamX のホストブラウザを起動させるためには，runFoamXHB スクリプトを走らせるか，あるいはホストブラウザがローカルで動いているケース（ノーマルモード）では，runFoamX を走らせることで runFoamX 自身が runFoamXHB を立ち上げます．runFoamX の二つの機能を図 A.2 に示します．



(a) ネームサーバ nsd の起動 (b) FoamXHostBrowser の起動

図 A.2 runFoamXHB の実行

- nsd と呼ばれるプロセスである MICO ネームサーバは，ホストマシンからスタートされます．MICO はホストネームとデフォルトのポートアドレスを使用します．このポートアドレスは *.OpenFOAM-1.5/apps/FoamX* ディレクトリの中にある *FoamXClient.cfg* ファイルへの `org.omg.CORBA.ORBInitialHost` と `org.omg.CORBA.ORBInitialPort=` のエントリによりマニュアルで設定できます．ネームサーバは同じディレクトリの中にある *ns.ref* に IOR の host/port を書き込みます．

- FoamXHostBrowser プロセスは，nsd がスタートされ，FoamXHostBrowser という名で登録されている host/port 上でスタートされます．

したがって，コマンドプロンプトで次のようにタイピングして runFoamXHB を実行することにより，

```
runFoamXHB
```

画面に以下のように出力させるネームサーバとホストブラウザが立ち上げられます．

```
Starting NameServer with inet:<host>:<port>...
Starting FoamX Host Browser with inet:<host>:<port>...
```

ここで，<host>:<port> は，デフォルトで設定されているか，あるいは *FoamXClient.cfg* ファイルの中で指定されています．FoamXHostBrowser は OpenFOAM のロゴの商号と，正しく実行しているかどうかのステータスの詳細を画面上に表示します．

A.1.1 ネームサーバの実行に関する注記

ns.ref ファイルは，以下のようにタイピングすることにより変換され，閲覧できます．

```
iordump < $FOAMX_USER CONFIG/ns.ref
```

MICO に関する管理者用ツールは，以下のようにタイピングすることによりスタートします．

```
nsadmin -ORBNamingAddr inet:<host>:<port>
```

ここで，inet:<host>:<port> エントリは，*ns.ref* ファイルを閲覧することにより見つけられます．登録されたサービスをリストするために ls を含んでいるツールの中のオプションを閲覧するためには，help をタイプしてください．

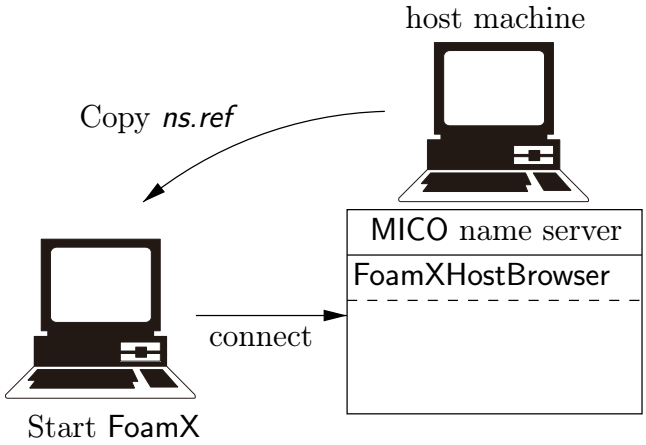
A.2 JAVA GUI

すべてのリモートマシン，あるいはホストマシン自体は，IOR を提供するために先に作成され *ns.ref* ファイルのコピーを使って，ネームサーバに接続することができます．リモートマシンでもまた，*FoamXClient.cfg* ファイルのホストマシンの名前に設定するために，A.1.1 項で述べた */etc/hosts* ファイルへの対応する入力とともに，org.omg.CORBA.ORBInitialHost= を入力する必要があります．

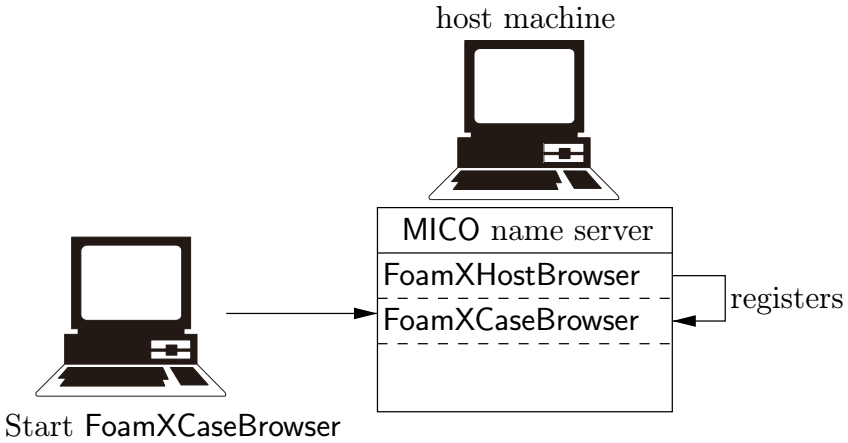
図 A.3 (a) に示しているように，リモートマシンで FoamXJAVAGUI をスタートするためには，runFoamX スクリプトを走らせる必要があります．このスクリプトは runFoamXHB により既に立ち上げられているネームサーバを探します．ユーザは，このサーバに接続しようとすることを認めるようコマンドラインで促されます．

```
Found server reference \$FOAMX\_USER\_CONFIG/ns.ref
Do you want to connect to this server ? (n)
```

ユーザが既存のネームサーバに接続しない場合や，runFoamXHB が実行されておらず，ネームサーバがない場合には，新規のネームサーバがローカルに作成されます．これにより，ホストブラウザと GUI の両方がローカルで走っているときは，runFoamXHB を走らせることなく，runFoamX を実行させれば十分です．コマンドプロンプトで，次のようにタイピングすることにより，図 A.4 に示すように，JAVA のブラウザ画面が開きます．



(a) ネームサーバへの接続



(b) FoamXCaseBrowser の実行

図 A.3 runFoamX の実行

`runFoamX`

ブラウザは以下の領域に分割されています．

Menu bar and buttons（上部） ケースの作成と構築，および実行に用いる操作があります．
case panel（左側） ケースブラウザの中のケースディレクトリと，ケースサーバの中の OpenFOAM のケースのコンテンツがあります．
Editing panel（右側ブルーの部分） ケースの入力が修正されます．
Progress history panel（上部） 実行された動作に関する情報を提示するダイアログボックスです．

デフォルトにより，ケースパネルはネームサーバが走っているホストマシンを表示します．他のリモートマシンのケースにアクセスしたい場合には*.OpenFOAM-1.5/controlDict* ファイルのホストにマシンをリストする必要があります．FoamX の画面は，通常の方法で大きさの変更ができます．FoamX の中の個々の画面は，画面を分割している斑点のバーをクリックし，カーソルをドラッグすることにより大きさを変更できます．

ブラウザへのパスコマンドには 3 通りの方法があります．

- アイテムを選択し，ダブルクリックします．一般的なコンテンツの開き方です．
- アイテムを選択し，マウスの右ボタンをクリックして，このアイテムで実行可能なメニューを

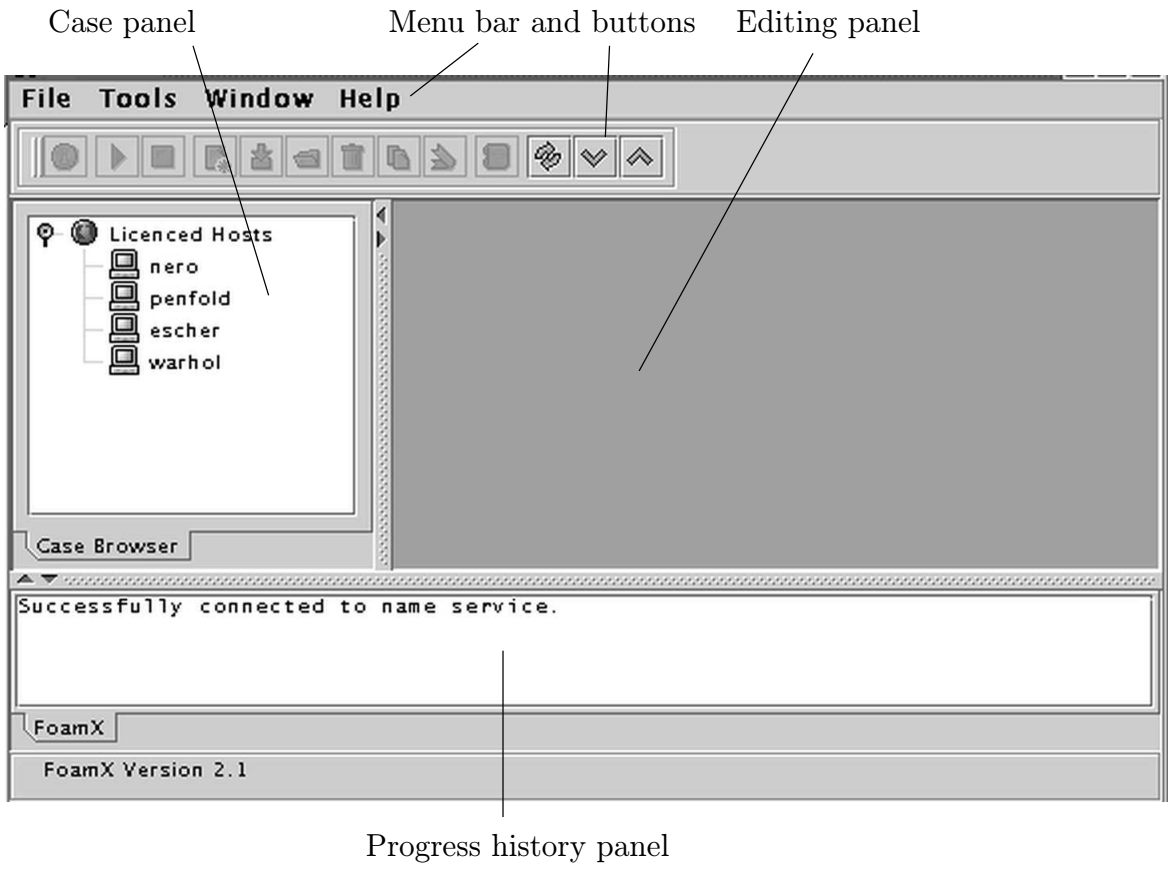


図 A.4 FoamX のメイン・ブラウザ画面

表示させます．

- メニューバーとボタンからアイテムを選択することにより，他の操作を行うことができます．

注記：メニューボタンの上にカーソルを重ねると，カーソルの下に小さいダイアログボックスが表れ，簡単な用途の記述が表示されます．

A.3 ケースブラウザ

JAVA GUI から，ケースパネルの中にリストアップされているマシンのケースブラウザを開くためには，ホストアイコンをダブルクリックするか，あるいはシングルクリックでホストをハイライトさせてメニューボタンかマウスの右のボタンから Open Case Browser で選択します．この操作は図 A.3 (b) に示しているように，FoamXCaseBrowser を開くために FoamXHostBrowser を呼び出しています．この FoamXCaseBrowser はネームサーバを参照して自らを登録するために *ns.ref* ファイルを読み込みます．その後，JAVA GUI は FoamXCaseBrowser を調べることができるようになり，たとえばケース上で作業を開始するために FoamXCaseServer を起動するときに FoamXCaseBrowser を呼び出します．FoamXCaseServer は自らをネームサーバ上に登録し，サービスを登録したり呼び出したりしてプロセスは続いていきます．

ケースブラウザは，引数としてホスト名をもたせて runFoamX を実行することにより，JAVA GUI が立ち上がるときに自動的に開きます．

```
runFoamX [host]
```

ホストマシンでケースブラウザをスタートすると，図 A.5 に示すように，OpenFOAM が格納されているルートパスディレクトリのツリーのリストが作られます．ユーザの *.OpenFOAM-1.5/controlDict* ファイルの中で指定されたケースのルートについて，その追加や削除に関しては A.5.2 項を参照してください．

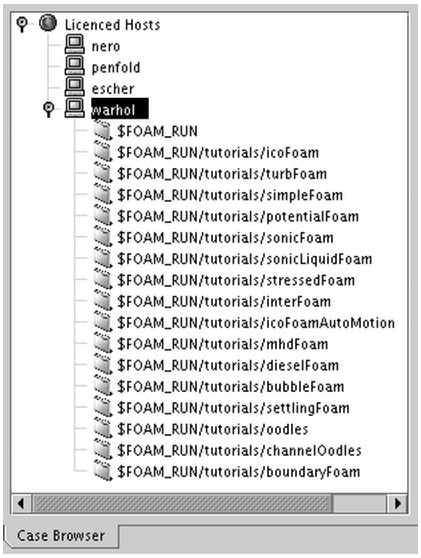
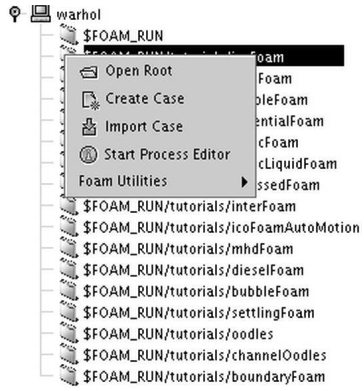


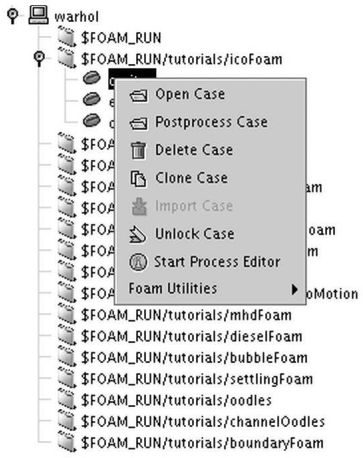
図 A.5 ケースのルートディレクトリのツリー

以下のマニュアルでの注意．本文中に書かれた FoamX に関するあらゆる操作は，特に注記がない限り，メニューバーまたはボタンから，あるいはマウスで右クリックすることにより選択されたものです．

図 A.6 に示すように，ケースブラウザは一連の機能を提供しています．ルートディレクトリのアイコンを選択することにより，ユーザはディレクトリを開いたり，新しいケースを作成したり，ケースをインポートしたり，ユーティリティを走らせることができます．ケースの名前のアイコンをハイライトさせることにより，ケースのオープン，削除，複製あるいは開放，ケース上での OpemFOAM のユーティリティの実行を行うことができます．



(a) ケース・ディレクトリの選択



(b) ケース・ネームの選択

図 A.6 ケース・ブラウザの機能

A.3.1　ルートディレクトリのオープン

図 A.6 (a) に示すように，カーソルをルートディレクトリの上に置いて右クリックしてメニューを表示するか，あるいはルートディレクトリのアイコンをダブルクリックすることで Open Root の機能を選択することにより，ケースのルートディレクトリの中にある現在のケースのセットを閲覧することができます．図 A.7 に示すように，ルートディレクトリのケースのツリーを表示するためにディレクトリを開きます．

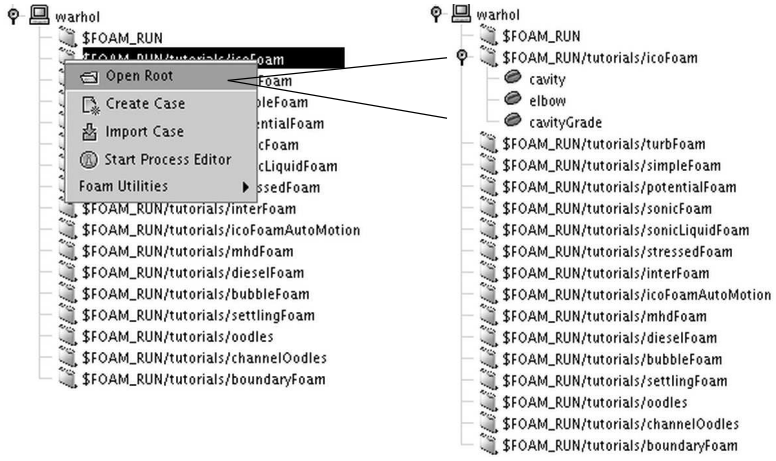


図 A.7　ケース・ルートを開く

A.3.2　新規のケースの作成

図 A.8 に示すように，メニューボタンから，もしくはカーソルをホストのアイコンかケースのディレクトリの上に置いて右クリックして Create Case 機能を選択することにより，新規のケースを作成することができます．図 A.8 に示すように，Class，Case Name および Case Root へのデータの登録ボックスを伴った小さなウィンドウが現れます．

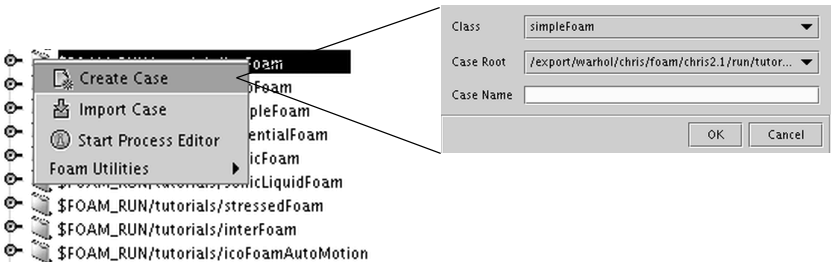


図 A.8　新規ケースの作成

Class は，icoFoam や turbFoam のような OpenFOAM のソルバの名前を含んだスクロールメニューを提供しています．FoamX は，選択されたソルバにより要求されるケースファイルの中に必要なデータを入力します．したがって，正しいソルバを選ぶことは非常に重要です．Case Name と Case Root はそれぞれ，ディレクトリパスとディレクトリ名となっており，その中でケースのデータは，4.1 節で述べてられているファイル構造にしたがって格納されています．正しく入力できたら OK をクリックします．新規のケースのケースサーバが開かれ，A.4 節で述べるように，ケースファイルの修正，ソルバやユーティリティの実行などができるようになります．

A.3.3 既存のケースを開く

図 A.9 に示すように，Open Case の機能は，ケースサーバの中にある既存のケースを開きます．A.4 節で述べるように，ケースサーバではケースファイルの修正，ソルバやユーティリティの実行ができるようになっています．

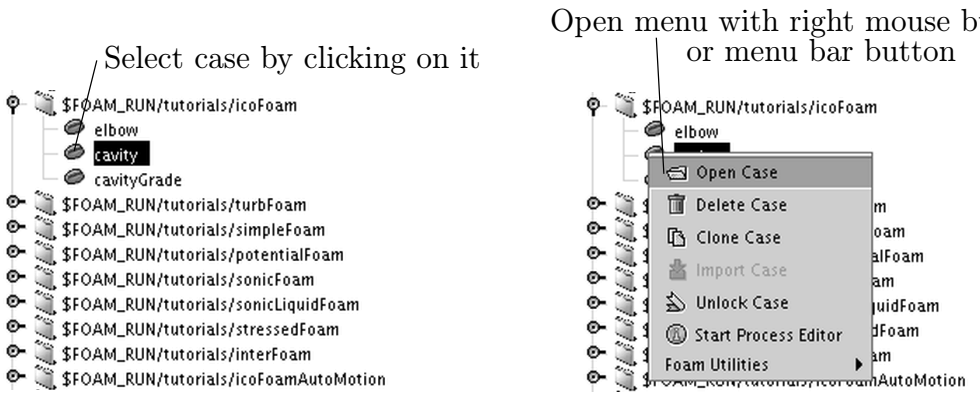


図 A.9 既存ケースを開く

A.3.4 既存のケースの削除

ハードディスクからケースディレクトリを削除するためには，ケースをハイライトさせ，Delete Case 機能を選択します．図 A.10 に示すように，機能に関する画面が表示され，削除するかどうかを聞いてきますので，同意する場合には Yes ボタンを，そうでない場合には No ボタンをクリックします．

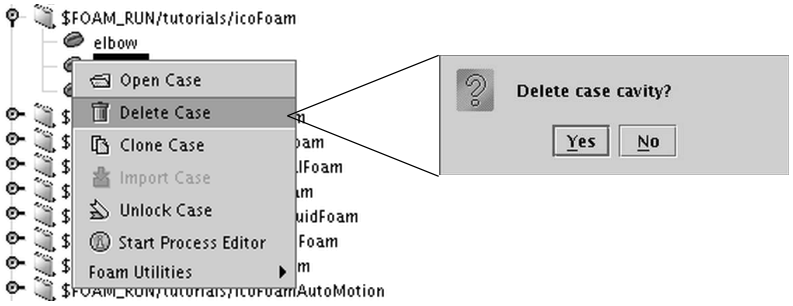


図 A.10 既存ケースの削除

A.3.5 既存のケースの複製

Clone Case 機能は選択されたケースから既存のファイルをコピーする新しいケースを作成します．図 A.11 に示すように，最初に複製をしたいケースをハイライトさせ，Clone Case 機能を選択する必要があります．これによりテーブルを開きますが，このテーブルの中で新しいケース名が指定されていなければならず，またルートパスと applicationClass は複製されるケースのものに変更されるかもしれません．最後に，`times` の入力を行うことで複製の作業中にコピーされる時刻ディレクトリを選ぶことができます．このオプションを表 A.1 に示します．

的確な情報を入力し，Close ボタンをクリックすると，複製操作が完了したことが知らされます．A.3.3 項で説明しているように，新しいケースを開くことができるようになります．

オプション	内容
`firstTime`	最も古い時刻ディレクトリをコピーします
`latestTime`	一番新しい時刻ディレクトリをコピーします
`allTime`	すべての時刻ディレクトリをコピーします
`noTime`	時刻ディレクトリをコピーしません

表 A.1　Clone Case 操作における時刻ディレクトリのコピーのオプション

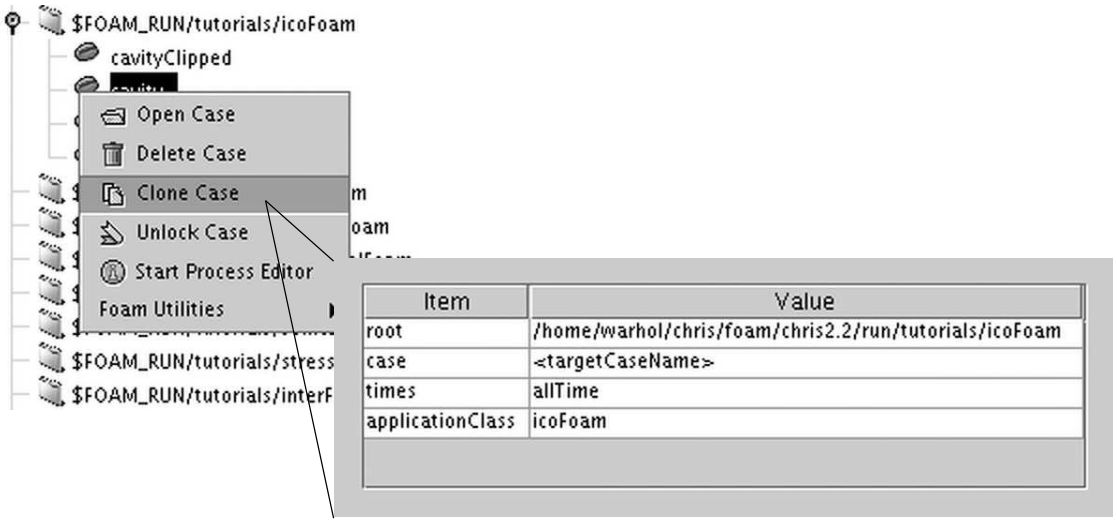


図 A.11　既存のケースの複製

A.3.6　既存のケースを解放

ケースが作成されているとき，あるいは開かれているときには，このケースが異なるサーバでオープンされることを防ぐためにロックファイルが作成されます．ケースが閉じられるときには，再度開くことができるように，このロックファイルは取り除かれます．ある種の環境では，たとえケースサーバでケースがそれ以上進行できなくなるとしても，ロックファイルは削除されないかもしれません．例えば，ケースサーバでケースが開かれているのにホストブラウザが閉じられてしまうような場合がこれに該当します．このため，Unlock Case 機能はロックファイルを削除するオプションを備えています．図 A.12 に示すように，ケースが他のユーザによって進められているかもしれないことを警告する画面が表示されます．そのときには，ロックファイルの削除を受け入れる前に他の場所でケースが進められていないことを確認することはユーザの責任となります．

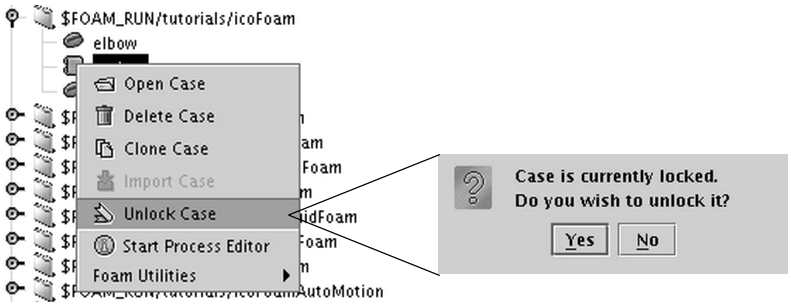
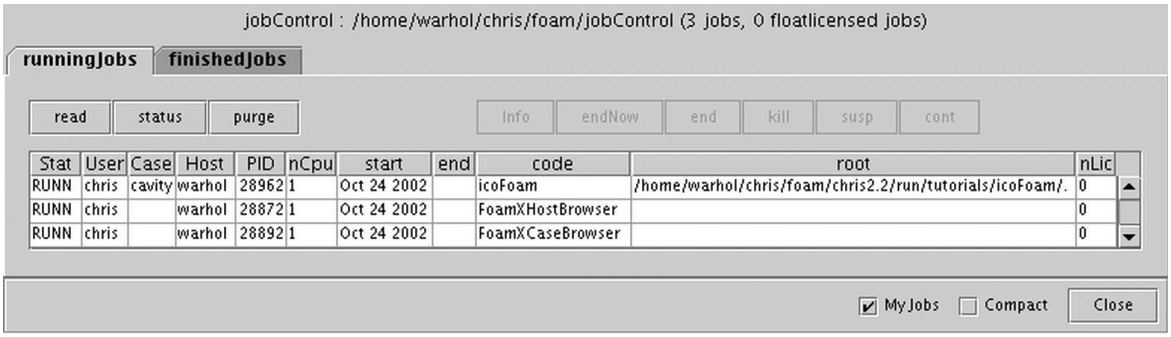


図 A.12　既存のケースのロックを解除する

A.3.7 プロセスエディタ

Start Process Editor 機能は，終了した，あるいは現在実行中のすべての OpenFOAM のジョブをモニタできるエディタを開きます．エディタは単純な GUI で，インストール時に *$FOAM_LIC_DIR* ディレクトリの中に置かれており，*runningJobs* と *finishedJobs* のディレクトリにあるファイルを読み込みます．これは図 A.13 に示すような構成になっています．



(a) 実行中のジョブ一覧

(b) 終了したジョブ一覧

図 A.13 プロセスの管理

それは図 A.13 に示すような画面からなります．タグで，`runningJobs` テーブルと `finishedJobs` テーブルを切り替えられます．テーブルは非常にわかりやすいジョブの詳細が入っています．runningJobs テーブルの左上に表 A.2 にリストアップされたタスクを実行するボタンがあります．runningJobs テーブル上でクリックすることによって，ジョブを選択することもできます．するとテーブルの右上のボタンが有効になります．これらのボタンで，表 A.2 に記載されているジョブを制御することができます．

finishedJobs テーブルは OpenFOAM で実行されたが何らかの理由で終了されたジョブの情報のアーカイブです．役立つ入力は保存して，そうでないものは削除することが自由にできます．テーブルには入力を削除するための 2 個のボタンがあります．パージボタンは 7 日以上前に終了しているジョブを削除します．リムーブボタンはテーブルから選択された入力を取り除きます．

プロセスエディタ画面の下部に，表 A.2 に記載してあるように，どのジョブが runningJobs テーブルに入り，どのジョブが finishedJobs テーブルに入るかを管理するチェックボックスが二つあります．

A.3.8 OpenFOAM ユーティリティの実行

Foam Utilities 機能では，OpenFOAM ユーティリティを実行することができます．この機能はケースサーバでも提供されており，そこで使う方がより一般的です．それについては A.4 節で説明します．

メインボタン	
read	runningJobs と finishedJobs ディレクトリのジョブを再度読込む
status	ジョブのステータスを更新するためにホストマシンに接続する
purge	これ以上実行しないジョブを取り除く

ジョブ実行ボタン	
Info	ジョブの情報画面を表示する
endNow	次のタイムステップでジョブを終わらせる
end	ジョブがフィールドデータをファイルに出力する次のタイムステップでジョブを終わらせる
kill	即座にジョブを終了
suspend	即座にジョブを一時停止
cont	一時停止したジョブを再実行

チェックボックス	
My Jobs	現在のユーザのジョブのみ見る
Compact	リストから FoamX 関連のジョブを取り除く

表 A.2　プロセスエディタボタン

A.4　ケースサーバ

ケースブラウザからケースを開くと，ケースサーバが起動します．図 5.14 に示すように，ケース画面にディレクトリのツリーが表示されます．ユーザはケース画面の底部にあるタグを用いて新規のケースとケースブラウザのやりとりを行うことができます．ディレクトリのツリーには，トップレベルの 3 種類の入力項目があります．

Dictionaries　ケースのコントロールと物理特性を設定するディクショナリをもっています．
Fields　フィールドの初期条件と境界条件を設定します．
Mesh　メッシュの読み込みとインポート，メッシュのパッチに関する境界条件を設定します．

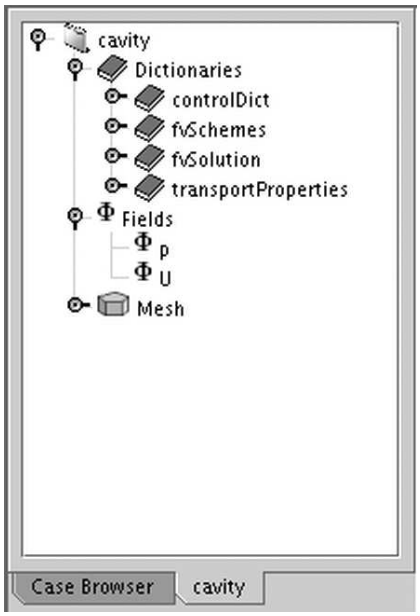


図 A.14　ケースサーバのウィンドウ

A.4.1 既存のメッシュのインポート

ケースにはメッシュが必要であり，そのメッシュは 5.3 節で述べている blockMesh ユーティリティを使うか，あるいは OpenFOAM のメッシュ変換機と結合したサードパーティのソフトウエアを用いて作成されます．OpenFOAM のメッシュはケースの *constant/polyMesh* ディレクトリに以下のものとして保存されます．boundary，cells 等の OpenFOAM のメッシュを構成するファイル．OpenFOAM のメッシュを作成するために blockMesh が使う *blockMeshDict* ファイル．あるいはその両方があります．図 A.15 に示すように，ユーザはこれらすべてのファイルを Import Mesh ファンクションを使って既存の *constant/polyMesh* ディレクトリからそれぞれの case にインポートします．

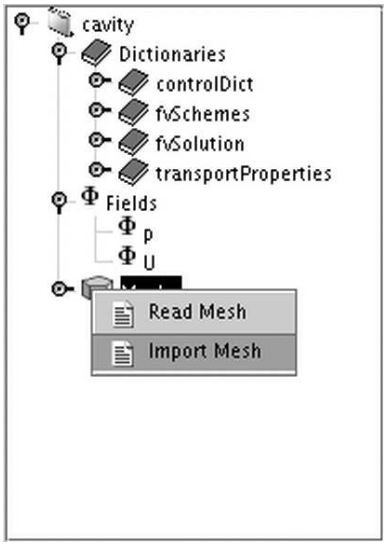


図 A.15 OpenFOAM のメッシュのインポート

A.4.2 メッシュの読み込み

constant/polyMesh ディレクトリの中に，直接インポートされた，または blockMesh かメッシュの変換ユーティリティにより生成されたメッシュファイルが存在すると，Read Mesh&Fields 機能を用いてケースサーバにこれらを読み込むことができるようになります．

この機能を試すには，チュートリアルの例のいずれかのファイルを開き，A.4.8 項で述べている blockMesh ユーティティを使ってメッシュを生成する必要があります．

A.4.3 境界のパッチの設定

図 A.16 に示すように，Read Mesh&Fields 機能を実行した後は，ディレクトリのツリーはメッシュに関する境界のパッチのリストを表示するようになります．その後，パッチをハイライトさせ，Define Boundary Type 機能を選択することにより，物理的な境界条件をパッチに合わせることができます．

これにより，編集パネルの中に patch description（パッチの記述）画面が表示されます．図 A.17 のイラストに示すように，Boundary Type の表示の右側の ... ボタンをクリックすることにより，物理的な境界条件のタイプを選択することができます．特定のソルバに利用可能な物理的な境界条件のタイプのリストが画面に現れます．リストから選択して，OK をクリックすると，ウィンドウは閉じ，patch description 画面に戻ります．物理的な Boundary Type の表示の下には，ソルバの初期変数や

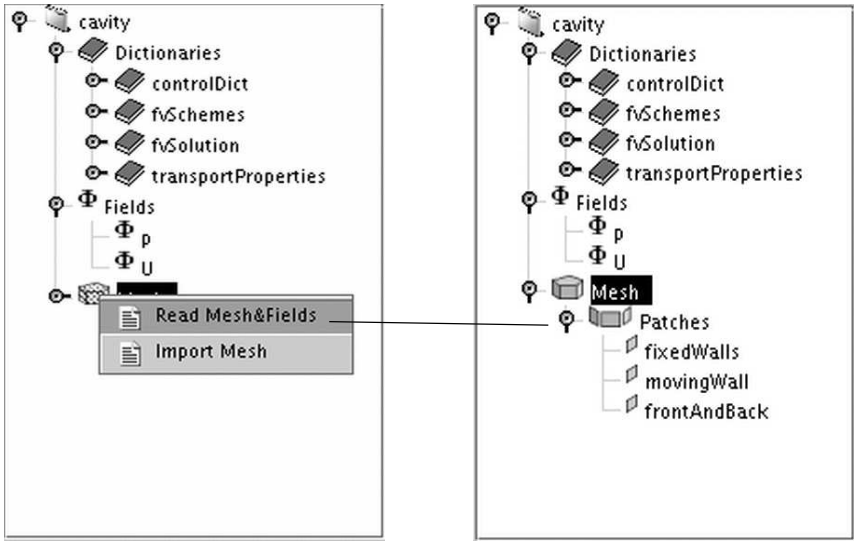


図 A.16　OpenFOAM メッシュの読み込み

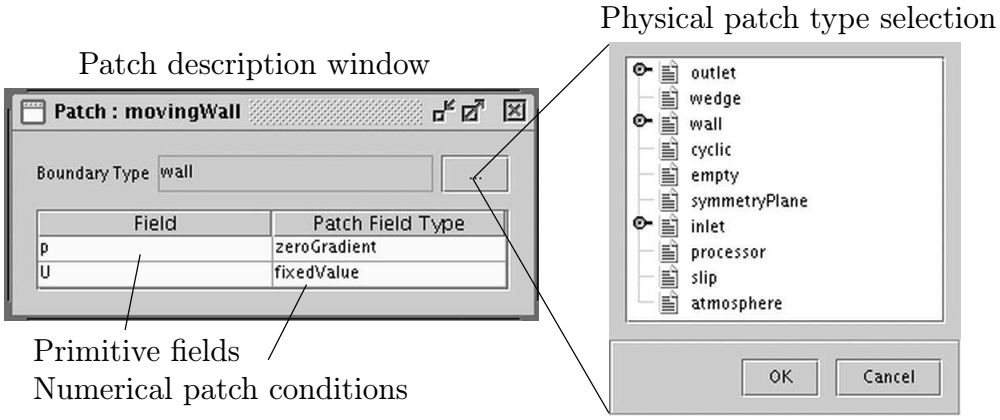


図 A.17　物理的境界条件の選択

解析時のそれらのパッチタイプ，または境界条件をリストした表があります．このとき，2 次元のケースにおいて一列に並んだ前後のパッチが empty タイプになるように注意してすべてのパッチに対して物理的な境界条件のタイプを選択します．

A.4.4　フィールドの設定

いったん全ての物理的なパッチタイプを決定すると，いつものようにハイライトさせて右クリックするかアイコンをダブルクリックすることによって選択される Fields は，`Edit Field` 機能を使って編集できます．

Edit Field 機能は図 A.18 に示すように編集パネルにフィールド画面を表示します．表は 4.2.8 項で概説した各フィールドで必要な一連のデータ値を記載しており，`internalField` や `referenceLevel`，そして物理的タイプの決定に必要な 1 個以上のパッチに対応する全ての値が記されています．物理的なパッチタイプの仕様の変化に対応するためにパッチリストはアップデートされることに注意してください．値を変えるときには Value 欄の入力箇所をクリックします．図 A.18 では例として `movingWall` という名のパッチで $(1, 0, 0)$ m/s の等速度の設定を示しています．

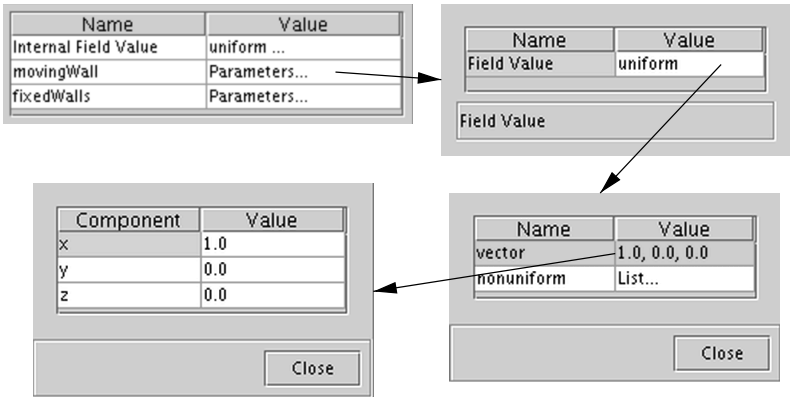


図 A.18　フィールドの編集とパッチ条件の設定

A.4.5　ディクショナリの編集

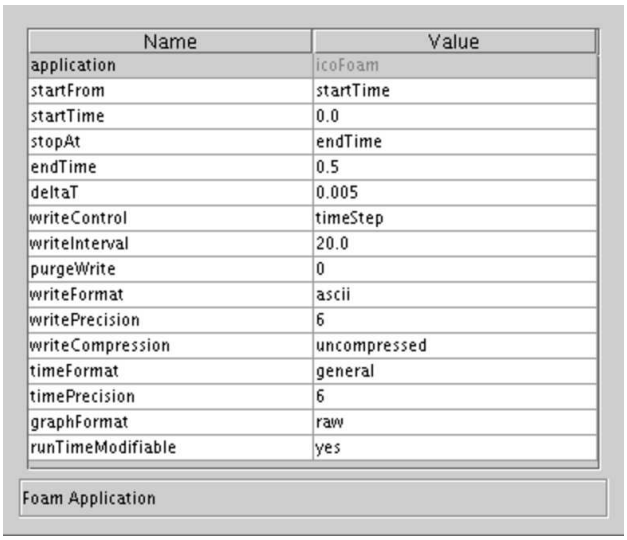


図 A.19　ディクショナリ画面の例：*controlDict*

Dictionaries のデータは編集可能です．ディクショナリは図 A.19 に示す *controlDict* や 4.3 節，4.4 節，4.5 節でそれぞれ示した fvSchemes や fvSolution，そして材料プロパティの元となるものを含んでいます．ディクショナリは右欄にデータ入力箇所をもつ表形式の入力となっています．入力箇所をクリックすることで値を直接編集でき，また同じ方法により値が編集可能なサブディクショナリを開くこともできます．例えば図 A.19 の applicationClass のように灰色で表示された入力箇所は編集できないので注意してください．また入力箇所によっては Selection Editor から選択することもあり，この場合には選択された入力欄は緑色にハイライトされます．

A.4.6　データの保存

ボタンバーから Save Case 機能を選択すれば，ケースへの変更を保存できます．ディクショナリやフィールド，メッシュのデータが保存されます．

A.4.7　ソルバの実行

ソルバは二つの方法から選択して実行できます．フォアグラウンドですぐに実行したい場合は，ボタンバーから Start Calculation Now 機能を選択します．詳しい情報の表示なく OpenFOAM ソルバはすぐに実行されます．

一方，ボタンバーから start Calculation 機能を選択することもできます．これは図 A.20 に示すように Run Application の画面を表示します．

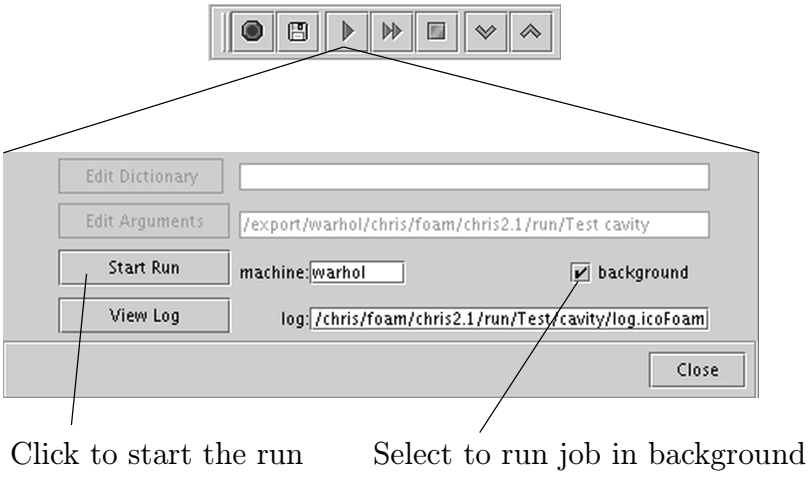


図 A.20　Start Calculation 機能を使ったソルバの実行

Start Run ボタンを押す前にバックグラウンドボタンをクリックすると，バックグラウンドでケースは実行されます．バックグラウンドで実行したケースの場合，途中経過はログテキストボックスで指定されたログファイルに書かれ，View Log ボタンを押せば見ることができます．

A.4.8　実行ユーティリティ

OpenFOAM には多くのユーティリティが供給されており，図 A.21 に示すようにケースサーバ画面でケース名アイコンをハイライトさせ，右クリックでユーティリティを含むメニューの階層を開くことで実行できます．

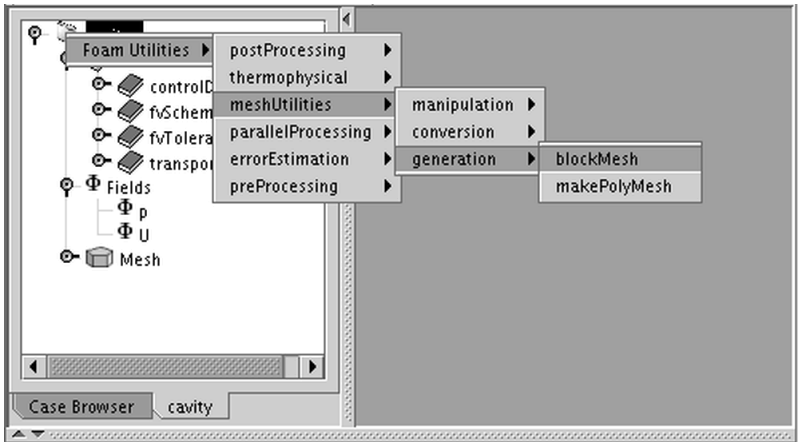


図 A.21　ユーティリティの実行

図 A.22 の例では blockMesh のユーティリティを選択すると，もしそのユーティリティに関連のあるディクショナリがあれば編集画面が開きます．必須のコマンドラインの変数は編集されているケースのデフォルト値に設定されています．ユーザは表から任意の変数を選択できます．

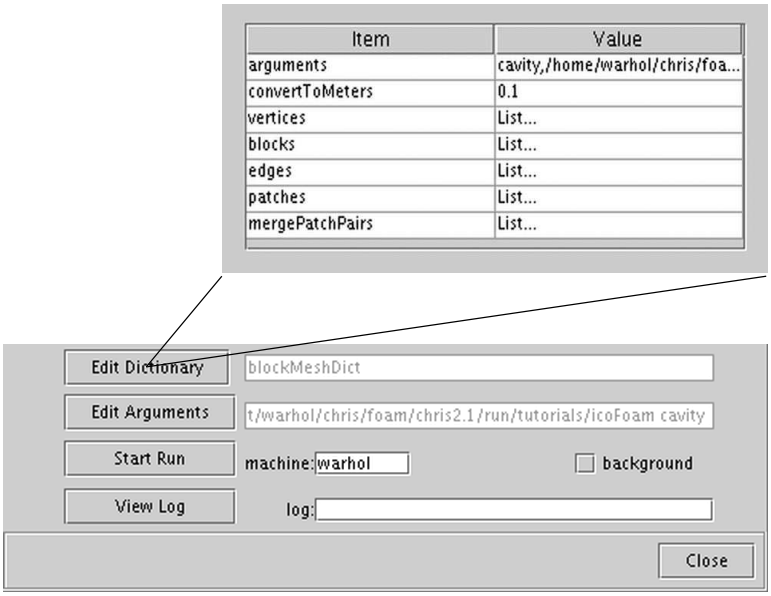


図 A.22　ユーティリティディクショナリの表示

A.4.9　ケースサーバーの終了

ケースサーバ画面を閉じて，ケースブラウザに戻るためには Close Case ボタンをクリックしてください．

A.5　FoamX の設定

FoamX ユーザ設定ファイルはユーザの *OpenFOAM-1.5/apps/FoamX* ディレクトリにあり，ディレクトリの構造を保持しています．そしてそのファイルはユーザの *$HOME* にコピーされているかもしれません．ユーザが設定可能なファイルは以下の通りです．

FoamXClient.cfg　では FoamX のネットワークや外観を設定します．特に以下の設定が可能です．

- `org.omg.CORBA.ORBInitialHost=` や `org.omg.CORBA.ORBInitialPort=` へ入力することにより与えられる host/port アドレスの設定．
- `FoamX.Browser=` を編集して netscape や mozilla，konqueror 等のブラウザへの入力を行ったり，URL が与えられている実行ファイルを編集することによるデフォルトブラウザの設定．
- `FoamX.Editor=` に関連のある入力をコメントアウト (`#`) し，internal や nedit，xemacs 等からその編集を削除することによるデフォルトエディタの設定．

FoamX.cfg　では編集可能な processControl を設定します．特に，リモート・シェルかセキュア・シェルのどちらを実行しているかによって rsh または ssh に remoteShell を設定します．このファイルでは接続の時間切れやコマンドの再試行に関する時間の設定もでき，問題が発生したときの

コマンドを増やすこともできます．

FoamX 編集に関する環境変数は*$FOAMX_*で始まり，表 A.3 に記載してあります．

環境変数	説明とオプション
$FOAMX_PATH	FoamX インストールへのパス，*$FOAM_UTIL/FoamX*
$FOAMX_SYSTEM_CONFIG	FoamX システム構成ファイルへのパス，*$FOAMX_PATH/config*
$FOAMX_USER_CONFIG	FoamX ユーザ構成ファイルへのパス，*$HOME/$FOAM_DOT_DIR/apps/FoamX*

表 A.3 FoamX の環境変数

A.5.1 JAVA

FoamX ケースブラウザは必ず必要なバージョンというわけではないかもしれませんが標準でインストールされている JAVA 1.5 を使います．そのため OpenFOAM に付属しており，$JAVA_HOME の環境変数は JAVA の最上位ディレクトリの*$WM_PROJECT_DIR/.bashrc* (もしくは*.cshrc*) におけるデフォルトで決められています．システム管理者は適宜*$JAVA_HOME* の設定を行うことで，代わりの場所に JAVA 1.5 をインストールしても構いません．

A.5.2 ケースファイルへのパス

FoamX は*.OpenFOAM-1.5/controlDict* ファイルへの `caseRoots` の入力からユーザのケース・ファイルへのパスを見つけます．デフォルトでは以下のように設定されています．

```
caseRoots
(
    "."
    "$FOAM_RUN/tutorials/icoFoam"
    "$FOAM_RUN/tutorials/turbFoam"
    ...
);
```

デフォルトでは$FOAM_RUN は*$HOME/OpenFOAM/${USER}-1.5/run* のディレクトリを示します．これは，デフォルトでは，*run* ディレクトリにコピーされたチュートリアルのディレクトリの中にあるケースや，FoamX が起動されるディレクトリの中のケースを開くことができることを意味します．もしそれらパスを設定したい場合は，*$HOME/.OpenFOAM-1.5* ディレクトリの *controlDict* ファイルのローカルコピーの中で設定を行ってください．

付録B
その他の参考情報

現在本章は，ユーザガイドの主要部分に入れるのに相応わしくないと思われる情報の置き場になっており，それは例えば，情報が細かすぎるとか，古いとかの理由なのですが，特定の状況ではユーザの役に立つかもしれません．

B.1　MPICH を用いた領域分解ケースの並列実行

この節では，3.4.2 項で説明した openMPI ではなく，MPI/MPICH を使って OpenFOAM のケースを並列計算する方法を説明します．

実行するアプリケーションがすべてのプロセッサノードに同じパス名をもっているかどうかで，MPI/MPICH の起動は異なります．以下の二つの場合では，実行ファイルへのパス名は異なります．

- プロセッサが全て同じ UNIX/Linux 構造に属さない場合
- すべてのノードから実行ファイルへのアクセスができるネットワークファイルシステム (NFS) がなく，したがって，それは異なったノードの異なった場所にインストールされる場合

B.1.1　全てのノードで同じ実行パス名の場合

プロセッサノードがすべてローカルである単一マシンにおいて，以下のコマンドを実行します．その際は ‘ は逆向きの引用文字であり一般的にはキーボードの最上部右側にある（’ ではない）ことに気をつけてください．

```
mpirun -np <nProcs> ‘which <foamExec>‘
  <otherArgs> -parallel < /dev/null >& log &
```

<nProcs> はプロセッサの数です．<foamExec> は実行ファイル，例えば，icoFoam です．そして，出力は *log* というファイルに転記されます．例えば，icoFoam が，*$FOAM_RUN/tutorials/icoFoam* ディレクトリのキャビティチュートリアルを三つのノードで実行する場合は，以下のコマンドを実行します．

```
mpirun -np 3 ‘which icoFoam‘ $FOAM_RUN/tutorials/icoFoam cavity
  -parallel < /dev/null >& log &
```

アクセスしたいプロセッサがマシンのクラスタの向こう側にあるとき，以下のコマンドを実行します．

```
mpirun  -machinefile <machinesFile> -np <nProcs> ‘which <foamExec>‘
<otherArgs> -parallel < /dev/null >& log &
```

ノード名を含む <machinesFile> ファイルが各行にひとつあり，現在ユーザがログオンしているマシンが最初のものであることを除いて，従来と同じです．<machinesFile> は，MPICH で読まれる

ファイルであり，したがって，マシン名と各マシンで使用するプロセッサ数が必要なだけで，ヘッダは必要ありません．例えば，マシン arp で 1 プロセッサ，マシン noddy で 2 プロセッサ実行するには，ファイルは以下のようにします．

```
arp:1
noddy:2
```

注意：共有メモリをもつマシンのクラスタにおける性能の最適化は，MPICH ライブラリを再コンパイルする必要があるかもしれません．やり方は MPICH 資料を見てください．

B.1.2 ノード間で実行パス名が異なる場合

異なるパス名をもつ実行ファイルを異なるノードで実行するには，全てのノードに同じバージョンの OpenFOAM をインストールし rsh を使用して実行する能力が必要です．後者は全てのノードで，例えば icoFoam のようなアプリケーションを実行することで確認できます．

```
rsh <machineName> icoFoam <root> <case>
```

実行ファイルの異なったパス名はノード名とそれぞれの実行ファイルへのパス名を含む <p4pgFile> ファイルを通してを指定できます．例えば Linux マシン arp と Solaris マシン noddy 上で icoFoam を実行するためには，<p4pgFile> は以下のように入力します．

```
arp   0 /usr/local/OpenFOAM/OpenFOAM-1.5/applications/bin/linuxOptMPICH/icoFoam
noddy 1 /usr/local/OpenFOAM/OpenFOAM-1.5/applications/bin/solarisOptMPICH/icoFoam
```

各行の 2 番目の入力，ここでは 0 と 1，は各マシンあたりの追加プロセスの数です．MPI の実行が arp から始められるので，マスタプロセスはそこで実行され，他の追加プロセスはそこで始めてはいけません．ジョブは実行によって始まります．

```
mpirun  -p4pg <p4pgFile> ‘which <foamExec>‘
<otherArgs> -parallel < /dev/null >& log &
```

索引

記号・数字

A

B

C

D

E

F

M

N

O

P

Q

R

T

X

Y

Z

あ

キーワードエントリ

ま

や

ユーティリティ

ら