



开源 CFD 工具箱

用户指南

2.3.1

2014 年 3 月 9 日

Copyright©2011-2014 OpenFOAM Foundation.

This work is licensed under a **Creative Commons Attribution - NonCommercial NoDerivs 3.0 Unported License**.

版权声明©2011-2014 OpenFOAM Foundation

本指南遵守： **Creative Commons Attribution - NonCommercial NoDerivs 3.0 Unported License**.

License

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. TO THE EXTENT THIS LICENSE MAY BE CONSIDERED TO BE A CONTRACT, THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. Definitions

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

in timed-relation with a moving image ("synching") will be considered an Adaptation for the purpose of this License.

- b. "**Collection**" means a collection of literary or artistic works, such as encyclopedias and anthologies, or performances, phonograms or broadcasts, or other works or subject matter other than works listed in Section 1(f) below, which, by reason of the selection and arrangement of their contents, constitute intellectual creations, in which the Work is included in its entirety in unmodified form along with one or more other contributions, each constituting separate and independent works in themselves, which together are assembled into a collective whole. A work that constitutes a Collection will not be considered an Adaptation (as defined above) for the purposes of this License.
- c. "**Distribute**" means to make available to the public the original and copies of the Work through sale or other transfer of ownership.
- d. "**Licensor**" means the individual, individuals, entity or entities that offer(s) the Work under the terms of this License.
- e. "**Original Author**" means, in the case of a literary or artistic work, the individual, individuals, entity or entities who created the Work or if no individual or entity can be identified, the publisher; and in addition (i) in the case of a performance the actors, singers, musicians, dancers, and other persons who act, sing, deliver, declaim, play in, interpret or otherwise perform literary or artistic works or expressions of folklore; (ii) in the case of a phonogram the producer being the person or legal entity who first fixes the sounds of a performance or other sounds; and, (iii) in the case of broadcasts, the organization that transmits the broadcast.
- f. "**Work**" means the literary and/or artistic work offered under the terms of this License including without limitation any production in the literary, scientific and artistic domain, whatever may be the mode or form of its expression including digital form, such as a book, pamphlet and other writing; a lecture, address, sermon or other work of the same nature; a dramatic or dramatico-musical work; a choreographic work or entertainment in dumb show; a musical composition with or without words; a cinematographic work to which are assimilated works expressed by a process analogous to cinematography; a work of drawing, painting, architecture, sculpture, engraving or lithography; a photographic work to which are assimilated works expressed by a process analogous to photography; a work of applied art; an illustration, map, plan, sketch or three-dimensional work relative to geography, topography, architecture or science; a performance; a broadcast; a phonogram; a compilation of data to the extent it is protected as a copyrightable work; or a work performed by a variety or circus performer to the extent it is not otherwise considered a literary or artistic work.
- g. "**You**" means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.

- h. "**Publicly Perform**" means to perform public recitations of the Work and to communicate to the public those public recitations, by any means or process, including by wire or wireless means or public digital performances; to make available to the public Works in such a way that members of the public may access these Works from a place and at a place individually chosen by them; to perform the Work to the public by any means or process and the communication to the public of the performances of the Work, including by public digital performance; to broadcast and rebroadcast the Work by any means including signs, sounds or images.
- i. "**Reproduce**" means to make copies of the Work by any means including without limitation by sound or visual recordings and the right of fixation and reproducing fixations of the Work, including storage of a protected performance or phonogram in digital form or other electronic medium.

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

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

- a. to Reproduce the Work, to incorporate the Work into one or more Collections, and to Reproduce the Work as incorporated in the Collections; and,
- b. to Distribute and Publicly Perform the Work including as incorporated in Collections.

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

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

- a. You may Distribute or Publicly Perform the Work only under the terms of this License. You must include a copy of, or the Uniform Resource Identifier (URI) for, this License with every copy of the Work You Distribute or Publicly Perform. You may not offer or impose any terms on the Work that restrict the terms of this License or the ability of the recipient of the Work to exercise the rights granted to that recipient under the

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

- b. You may not exercise any of the rights granted to You in Section 3 above in any manner that is primarily intended for or directed toward commercial advantage or private monetary compensation. The exchange of the Work for other copyrighted works by means of digital file-sharing or otherwise shall not be considered to be intended for or directed toward commercial advantage or private monetary compensation, provided there is no payment of any monetary compensation in connection with the exchange of copyrighted works.
- c. If You Distribute, or Publicly Perform the Work or Collections, You must, unless a request has been made pursuant to Section 4(a), keep intact all copyright notices for the Work and provide, reasonable to the medium or means You are utilizing: (i) the name of the Original Author (or pseudonym, if applicable) if supplied, and/or if the Original Author and/or Licensor designate another party or parties (e.g., a sponsor institute, publishing entity, journal) for attribution ("Attribution Parties") in Licensor's copyright notice, terms of service or by other reasonable means, the name of such party or parties; (ii) the title of the Work if supplied; (iii) to the extent reasonably practicable, the URI, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work. The credit required by this Section 4(c) may be implemented in any reasonable manner; provided, however, that in the case of a Collection, at a minimum such credit will appear, if a credit for all contributing authors of Collection appears, then as part of these credits and in a manner at least as prominent as the credits for the other contributing authors. For the avoidance of doubt, You may only use the credit required by this Section for the purpose of attribution in the manner set out above and, by exercising Your rights under this License, You may not implicitly or explicitly assert or imply any connection with, sponsorship or endorsement by the Original Author, Licensor and/or Attribution Parties, as appropriate, of You or Your use of the Work, without the separate, express prior written permission of the Original Author, Licensor and/or Attribution Parties.
- d. For the avoidance of doubt:
 - i. **Non-waivable Compulsory License Schemes.** In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme cannot be waived, the Licensor reserves the exclusive right

- to collect such royalties for any exercise by You of the rights granted under this License;
- ii. **Waivable Compulsory License Schemes.** In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme can be waived, the Licensor reserves the exclusive right to collect such royalties for any exercise by You of the rights granted under this License if Your exercise of such rights is for a purpose or use which is otherwise than noncommercial as permitted under Section 4(b) and otherwise waives the right to collect royalties through any statutory or compulsory licensing scheme; and,
 - iii. **Voluntary License Schemes.** The Licensor reserves the right to collect royalties, whether individually or, in the event that the Licensor is a member of a collecting society that administers voluntary licensing schemes, via that society, from any exercise by You of the rights granted under this License that is for a purpose or use which is otherwise than noncommercial as permitted under Section 4(b).
- e. Except as otherwise agreed in writing by the Licensor or as may be otherwise permitted by applicable law, if You Reproduce, Distribute or Publicly Perform the Work either by itself or as part of any Collections, You must not distort, mutilate, modify or take other derogatory action in relation to the Work which would be prejudicial to the Original Author's honor or reputation.

5. Representations, Warranties and Disclaimer

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

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

7. Termination

- a. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Collections from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.
- b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

- a. Each time You Distribute or Publicly Perform the Work or a Collection, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.
- b. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.
- c. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.
- d. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.
- e. The rights granted under, and the subject matter referenced, in this License were drafted utilizing the terminology of the Berne Convention for the Protection of Literary and Artistic Works (as amended on September 28, 1979), the Rome Convention of 1961, the WIPO Copyright Treaty of 1996, the WIPO Performances and Phonograms Treaty of 1996 and the Universal Copyright Convention (as revised on July 24, 1971). These rights and subject matter take effect in the relevant jurisdiction in which the License terms are sought to be enforced according to the corresponding provisions of the implementation of those treaty provisions in the applicable national law. If the

standard suite of rights granted under applicable copyright law includes additional rights not granted under this License, such additional rights are deemed to be included in the License; this License is not intended to restrict the license of any rights under applicable law.

Trademarks

ANSYS is a registered trademark of ANSYS Inc.

CFX is a registered trademark of Ansys Inc.

CHEMKIN is a registered trademark of Reaction Design Corporation

EnSight is a registered trademark of Computational Engineering International Ltd.

Fieldview is a registered trademark of Intelligent Light

Fluent is a registered trademark of Ansys Inc.

GAMBIT is a registered trademark of Ansys Inc.

Icem-CFD is a registered trademark of Ansys Inc.

I-DEAS is a registered trademark of Structural Dynamics Research Corporation

JAVA is a registered trademark of Sun Microsystems Inc.

Linux is a registered trademark of Linus Torvalds

OpenFOAM is a registered trademark of OpenCFD Ltd

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

This *OpenFOAM User Guide, Chinese Edition* is translated into Chinese by Dongyue Li¹, except following sections: Houcun Zhou²(Chaper 1), Jianzhi Li³(Section 2.3), Chunlai Tian⁴(Section 5.5.2)

¹Beijing University of Chemical Engineering, Beijing, China, 100029. ²National University of Defense Technology, HuNan, China, 410073 ³JiangSu University, JiangSu, China, 212013 ⁴Beijing Institute of Technology, Beijing, China, 100081

《*OpenFOAM 用户指南中文版*》由李东岳¹翻译，其他参加翻译的人员有周后村²(第一章)、李建治³(第 2.3 节)，田春来⁴(第 5.5.2 节)

¹ 北京化工大学，北京，100029. ² 国防科技大学，湖南，410073. ³ 江苏大学，江苏，212013. ⁴ 北京理工大学，北京，100081.

目录

版权声明.....	1
第一章	15
第二章	17
2.1 顶盖驱动流.....	17
2.1.1 前处理.....	18
2.1.1.1 网格生成.....	18
2.1.1.2 边界和初始条件.....	20
2.1.1.3 物理特性.....	21
2.1.1.4 控制.....	21
2.1.1.5 离散方法和矩阵求解器设置.....	23
2.1.2 查看网格.....	23
2.1.3 运行算例.....	24
2.1.4 后处理.....	25
2.1.4.1 等值面和云图.....	25
2.1.4.2 矢量图.....	27
2.1.4.3 绘制流线图.....	27
2.1.5 网格细化.....	30
2.1.5.1 使用存在的算例创造新算例.....	30
2.1.5.2 生成细网格.....	30
2.1.5.3 投影算例结果.....	30
2.1.5.4 控制参数.....	31
2.1.5.5 后台运行.....	31
2.1.5.6 在细网格上绘制矢量.....	32
2.1.5.7 绘图.....	33
2.1.6 网格非均匀分布.....	34
2.1.6.1 创建非均匀化网格.....	35
2.1.6.2 调整时间步.....	37
2.1.6.3 投影场.....	37
2.1.7 增加雷诺数.....	37
2.1.7.1 前处理.....	38
2.1.7.2 运行算例.....	38
2.1.8 高雷诺数流动.....	39
2.1.8.1 前处理.....	39
2.1.8.2 运行.....	41
2.1.9 改变算例几何.....	41
2.1.10 后处理.....	43
2.2 带孔盘体应力分析.....	45
2.2.1 网格生成.....	46
2.2.1.1 边界和初始条件.....	49
2.2.1.2 物理特性.....	49
2.2.1.3 热物理特性.....	50
2.2.1.4 控制.....	50

2.2.1.5 离散格式和求解器控制	51
2.2.2 运行	52
2.2.3 后处理	53
2.2.4 练习	54
2.2.4.1 增加网格数量	54
2.2.4.2 引入网格非均匀化	54
2.2.4.3 改变平板尺寸	55
2.3 漩涡.....	55
2.3.1 生成网格	55
2.3.2 边界条件	57
2.3.3 设置初始场	58
2.3.4 流体特性	58
2.3.5 湍流模型	59
2.3.6 时间步长控制	60
2.3.7 离散格式	61
2.3.8 矩阵求解器控制	61
2.3.9 运行程序	62
2.3.10 后处理	62
2.3.11 并行运行	62
2.3.12 算例的并行后处理	65
第三章	67
3.1 OpenFOAM 编程语言	67
3.1.1 普适性编程语言	67
3.1.2 面向对象和 C++	68
3.1.3 方程呈现	68
3.1.4 求解器代码	69
3.2 编译程序和库	69
3.2.1 头文件	70
3.2.2 使用 wmake 进行编译	71
3.2.2.1 包含文件头	71
3.2.2.2 链接库	72
3.2.2.3 编译源文件	72
3.2.2.4 运行 wmake	73
3.2.2.5 wmake 环境变量设置	73
3.2.3 移除依赖包文件: wclean 和 rmdepall	73
3.2.4 编译实例: pisoFoam 求解器	74
3.2.5 调试与优化开关	77
3.2.6 使用自定义库	78
3.3 运行程序	79
3.4 并行计算	79
3.4.1 网格分解与初始场数据	79
3.4.2 运行分解算例	81
3.4.3 多硬盘数据分布	82
3.4.4 并行后处理	83

3.4.4.1 重组网格和数据.....	83
3.4.4.2 分解场后处理.....	83
3.5 标准求解器.....	83
3.6 标准工具.....	88
3.7 标准库.....	95
第四章	103
4.1 OpenFOAM 文件结构	103
4.2 基本输入输出格式.....	104
4.2.1 通用语法规则.....	104
4.2.2 字典.....	104
4.2.3 头文件.....	105
4.2.4 链表.....	106
4.2.5 Scalar 标量、Vector 矢量、Tensor 张量.....	107
4.2.6 量纲.....	107
4.2.7 单位类型.....	108
4.2.8 场.....	108
4.2.9 指令和宏替换.....	109
4.2.10 #include 和#inputMode 指令	109
4.2.11 #codeStream 指令	110
4.3 时间和输入输出控制.....	111
4.4 离散格式.....	113
4.4.1 插值格式(interpolationSchemes).....	115
4.4.1.1 严格有界标量格式.....	115
4.4.1.2 针对矢量场的格式.....	116
4.4.2 面法向梯度格式(snGradSchemes)	117
4.4.3 梯度格式(gradSchemes).....	117
4.4.4 拉普拉斯格式(laplacianSchemes).....	118
4.4.5 散度格式.....	118
4.4.6 时间格式.....	119
4.4.7 通量计算.....	120
4.5 求解和算法控制.....	120
4.5.1 矩阵求解器.....	121
4.5.1.1 求解误差.....	121
4.5.1.2 预处理双共轭梯度求解器.....	122
4.5.1.3 光顺矩阵求解器.....	122
4.5.1.4 GAMG	123
4.5.2 低松弛.....	123
4.5.3 PISO 和 SIMPLE 算法.....	124
4.5.3.1 参考压力.....	124
4.5.4 其它参数.....	125
第五章	127
5.1 网格.....	127
5.1.1 网格规范以及限制.....	127
5.1.1.1 点.....	128

5.1.1.2 面.....	128
5.1.1.3 网格单元.....	128
5.1.1.5 边界.....	129
5.1.2 polyMesh.....	129
5.1.3 cellShape	130
5.1.4 一维、二维以及轴对称问题.....	131
5.2 边界.....	132
5.2.1 在 OpenFOAM 中指定边界条件.....	132
5.2.2 基本类型.....	134
5.2.3 主要类型.....	135
5.2.4 衍生类型.....	135
5.3 blockMesh 网格生成程序	136
5.3.1 编写 blockMeshDict 文件	138
5.2.1.1 顶点.....	139
5.3.1.2 边.....	139
5.3.1.3 块.....	140
5.3.1.4 边界.....	141
5.3.2 多块网格.....	142
5.3.3 少于 8 个顶点的 block.....	144
5.3.4 运行 blockMesh.....	144
5.4 snappyHexMesh 网格生成工具	145
5.4.1 snappyHexMesh 网格生成	145
5.4.2 创建六面体背景网格	146
5.4.3 特征边和特征面网格切分	147
5.4.4 网格移除	149
5.4.5 指定区域网格细化	149
5.4.6 表面对齐	150
5.4.7 网格边界层	150
5.4.8 网格质量控制	153
5.5 网格转换	154
5.5.1 fluentMeshToFoam	154
5.5.2 starToFoam	155
5.5.2.1 转换 STAR-CD 网格的一般建议	155
5.5.2.2 消除多余数据	155
5.5.2.3 去掉默认边界条件	156
5.5.2.4 模型重新编号	157
5.5.2.5 写入数据	158
5.5.2.6 .vrt 文件可能的问题	158
5.5.2.7 转换为 OpenFOAM 可用格式	159
5.5.3 gambitToFoam	159
5.5.4 ideasToFoam	159
5.5.5 cfx4ToFoam	159
5.6 不同几何上的投影场	160
5.6.1 连续场投影	160

5.6.2 非连续场投影.....	160
5.6.3 并行投影.....	161
第六章	163
6.1 paraFoam.....	163
6.1.1 paraFoam 概述.....	163
6.1.2 Properties 面板.....	164
6.1.3 display 面板	165
6.1.4 工具栏.....	167
6.1.5 效果展示.....	167
6.1.5.1 View 设置	167
6.1.5.2 常规设定.....	167
6.1.6 云图绘制.....	168
6.1.6.1 剖面.....	168
6.1.7 矢量图绘制.....	168
6.1.7.1 在网格中心绘制.....	168
6.1.8 流线图.....	169
6.1.9 图形输出.....	169
6.1.10 动画输出.....	169
6.2 使用 Fluent 来后处理	170
6.3 使用 Fieldview 后处理.....	171
6.4 使用 EnSight 进行后处理.....	171
6.4.1 转换数据为 EnSight 格式.....	172
6.4.2 ensight74FoamExec 插件	172
6.4.2.1 EnSight 用户插件	172
6.4.2.2 使用用户自定义插件.....	172
6.5 提取数据.....	173
6.6 监控和管理进程.....	176
6.6.1 运行进程的 foamJob 脚本	177
6.6.2 监控进程的 foamLog 脚本	177
第七章	179
7.1 热物理模型.....	179
7.1.1 热物理数据.....	181
7.2 湍流模型.....	183
7.2.1 模型系数.....	184
7.2.2 壁面函数.....	184

第一章

导论

本手册是 OpenFOAM（开源场运算与操作 C++类库）使用指南。本手册详细描述了 OpenFOAM 的基本操作，首先通过第二章一系列教程练习，再在后续的章节对 OpenFOAM 中包含的各类工具组件进行了更详细地描述。亲自学习

首先，OpenFOAM 就是一个 C++类库，用于创建可执行文件，比如应用程序(application)。应用程序分成两类：求解器 (solver) 与工具 (utilities)。其中求解器是为了解决特定的连续介质力学问题而设计的；工具则是为了执行包括数据操作等任务而设计的。OpenFOAM 包括了大量的求解器和工具，牵涉的问题也比较广泛，其将在第三章进行详尽的描述。

OpenFOAM 的一个特点是用户可以通过必要的一些知识（包括数学，物理和编程技术等）创建新的求解器和工具。

OpenFOAM 需要前处理和后处理环境。OpenFOAM 本身的工具包含前处理、后处理接口，以确保不同环境之间数据传输的协调性。图 1.1 是 OpenFOAM 的结构示意图。在第四章中，将对前处理和算例运行进行阐述，第五章将阐述如何利用 OpenFOAM 自带的网格划分工具生成网格，以及将第三方软件生成的网格进行格式转换。第六章介绍后处理。

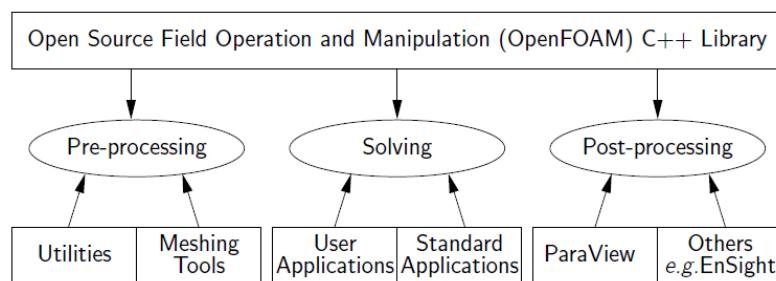


图 1.1 OpenFOAM 总体结构示意图

第二章

教程

这一章我们描述如何设置算例、如何进行模拟以及如何对 OpenFOAM 算例进行后处理，主要目的是给用户提供一个如何运行 OpenFOAM 的基本概念。\$FOAM_TUTORIALS 目录里面有许多算例来展示如何使用各种求解器，以及如何使用 OpenFOAM 附带的各种工具。在运行算例之前，用户必须首先确定它们正确的安装了 OpenFOAM。 
 教程里的例子描述了如何使用前处理器 blockMesh¹，如何进行算例设置，如何使求解器运行起来， 以及如何使用 paraFoam 进行后处理。对于使用第三方后处理工具的用户有两种选择：要么直接跟进 paraFoam 教程，要么跳至第 6 章查看相关描述。

所有的算例教程，都位于 OpenFOAM 的安装目录中。教程中的例子依据流体类型有序的整理为一系列的文件夹，并通过求解器的名字再次分类。例如，所有 icoFoam 求解器的算例 都存储在 incompressible/icoFoam 文件夹下，其中 incompressible 代表流体类型。如果用户想要运行一些例子，我们建议用户把教程下的例子拷贝到本地的 run 文件夹下，可以在终端这样执行²：

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

2.1 顶盖驱动流

在这个教程中，我们讲述如何对一个绝热二维方腔的不可压缩流算例进行前处理、运行、以及后处理。几何如图 2.1 所示，方腔所有的边界都是壁面，顶部壁面以 1m/s 的速度在 x 方向移动，其它壁面均为固定壁面。开始，我们假定流体为层流，并使用 icoFoam 在一个均匀网格上求解绝热不可压层流。接下来，我们将研究网格非均匀化以及壁面网格非均匀化对计

icofoam

¹ 网格生成工具

² 对于初次使用 OpenFOAM 的用户，由于是在 linux 下运行，请注意命令的大小写以及不要遗漏空格。全书类似的命令全部在终端执行（通过 Ctrl+T 打开）。

算结果的影响。最后，增加雷诺数，使用 pisoFoam³来求解绝热不可压湍流。

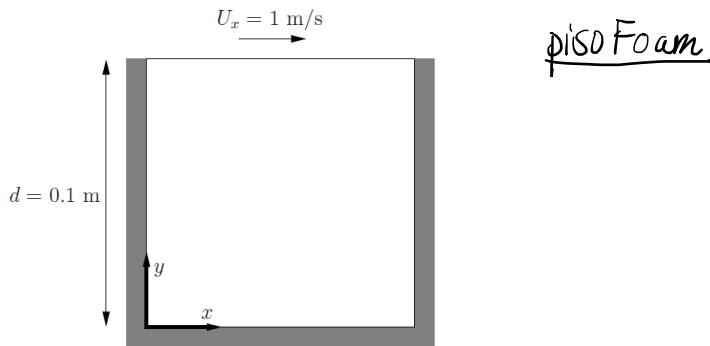


图 2.1：顶盖驱动流的几何

2.1.1 前处理

我们需要编辑算例文件来对 OpenFOAM 的算例文件进行设置。用户需要选择一个文本编辑器来进行编辑，例如 emacs, vi, gedit, kate, nedit 等。OpenFOAM 允许对算例文件进行编辑，这样对一个初学者来说，使用文本文档里面的关键词来输入输出是简单易懂的。

运行算例需要一系列的数据：网格、场、物性、控制参数等。正如 4.1 节所说，OpenFOAM 中这些数据存储在算例目录下的一系列文件中，而不是像其它 CFD 软件那样把它们存储在一个文件中。每个算例目录都有一个合适的名字。例如，第一个教程我们简单的称其为 cavity。在准备编辑算例文件以及运行第一个 cavity 算例之前，用户应该在终端运行一下命令来切换到算例目录下⁴：

`cd $FOAM_RUN/tutorials/incompressible/icoFoam/cavity /cavity (5 版本)`

2.1.1.1 网格生成

OpenFOAM 采用三维笛卡尔坐标体系，并且所有的算例网格都是三维的⁵。默认情况下，OpenFOAM 在三个维度上求解算例，但是如果指定某些面的边界（例如垂直于第三个方向的平面）条件为“empty”，那么它就可以用来求解二维算例。

cavity 几何由一个 xy 平面上边长为 0.1 米的正方形组成。最开始我们使用均一的 20*20 网格。block 结构参见图 2.2. 我们采用 OpenFOAM 提供的 blockMesh 来生成网格，它通过读取指定的字典文件来生成网格，这个字典文件位于算例文件夹下的 constant/polyMesh 文件夹下。blockMeshDict 文件信息如下所示：

```
1 /*-----* C++ -*------*
2 | ======
3 | \   / Field      | OpenFOAM:      The Open Source CFD Toolbox
4 | \| / Operation   | Version:      2.3.0
5 | \| / And          | Web:          www.OpenFOAM.org
6 | \|/ Manipulation |               |
7 */
8 FoamFile
```

在 system 文件夹下

实际用到版本为 5

³ 本手册中所有 xxxFoam 均为某个求解器的名字

⁴ 在终端切换

⁵ 这并不是说明都是三维算例，在 OpenFOAM 中，2D 算例的网格在第三个方向有一定厚度

```

9 {
10    version 2.0;

```

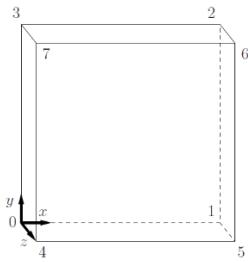


图 2.2 cavity 算例的 block 结构

```

11    format ascii;
12    class dictionary;
13    object blockMeshDict;
14 }
15 //*****//*****//*****//*****//*****//*****//*****//*****//*****//*****//*****//*****//*****//
16
17    convertToMeters 0.1;
18
19    vertices   vertex 顶点
20    (
21      0 (0 0 0)
22      1 (1 0 0)
23      2 (1 1 0)
24      3 (0 1 0)
25      4 (0 0 0.1)
26      5 (1 0 0.1)
27      6 (1 1 0.1)
28      7 (0 1 0.1)
29    );
30
31    blocks
32    (
33      hex (0 1 2 3 4 5 6 7) (20 20 1) simpleGrading (1 1 1)
34    );
35
36    edges
37    (
38    );
39
40    boundary
41    (
42      movingWall
43      {
44        type wall;
45        faces
46        (
47          (3 7 6 2)
48        );
49      }
50      fixedWalls
51      {
52        type wall;
53        faces
54        (
55          (0 4 7 3)
56          (2 6 5 1)
57          (1 5 4 0)
58        );
59      }
60      frontAndBack
61      {
62        type empty;
63        faces
64        (
65          (0 3 2 1)
66          (4 5 6 7)
67        );
68      }
69    );
70
71    mergePatchPairs
72    (
73    );
74
75 //*****//*****//*****//*****//*****//*****//*****//*****//*****//*****//*****//*****//*****//

```

参考 2.2 结构

分块信息

顺时针

文件的第一行到第七行是文件头信息，然后具体的字典信息通过的 `FoamFile` 后的`{...}`来指定。

在手册的剩余部分中：

为了避免赘述和节省空间，在引用具体的算例字典文件时，文件头，包括 `FoamFile` 子字典，都将省略

这个文件首先指定各个 `block` 顶点的坐标；然后通过顶点编号来定义 `block`，最后，定义边界。用户可以查阅 5.3 节来详细了解 `blockMeshDict` 的具体意义。

网格通过在这个算例目录下运行 `blockMesh` 命令来生成。在算例目录下，通过在终端简单地键入：

`blockMesh`

来完成，`blockMesh` 命令会把运行的情况输出到终端。`blockMeshDict` 中的任何错误 `blockMesh` 都会检测到并在终端输出错误，并告诉用户第几行错了。目前我们运行这个算例的时候，不会有错误。

2.1.1.2 边界和初始条件

一旦网格生成完毕，用户可以查看这个算例的初始场信息。在这个算例中，我们从 0 秒开始计算，因此初始的场信息存储在 `cavity` 目录下的 `0` 文件夹下。`0` 文件夹下又包含两个文件，`p` 和 `U`，它们是压力场和速度场，它们的初值和边界条件必须指定。下面我们来看一下 `p` 文件：

```

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

```

场文件中有 3 个主要信息：

- dimensions 指定场的单位，此处为运动压力⁶，单位为 $\text{m}^2 \text{s}^{-2}$. 用户可查阅 4.2.6 以获取更多信息；

⁶ 即为压力除以密度之后的压力

- internalField 内部场可以采用用一个值就可以描述的均匀场；或者每个网格的数据都需要指定的非均匀场（请查阅 4.2.8 节）；
- boundaryField 指定边界场信息，包括边界条件以及所有边界面所需的信息（请查阅 4.2.8 节）；

边界场 { 对于这个 cavity 算例，边界场由 walls 组成，并分为两个 patches，(1) fixedWalls，其用来指定固定边界，即 cavity 几何的底部和侧面。(2) movingWall，cavity 几何的上部。作为壁面，压力边界条件均为 zeroGradient，意味着压力的法相梯度为 0。frontAndBack 面即为 2D 算例的前后面，因此必须指定为 empty。 **法向速度为 0**

在这个算例中（以后我们也会经常遇到），初始场我们都设置为 uniform。这里的压力为运动压力，作为不可压缩流，它的绝对值是没有意义的。因此我们设置其为 0。这样比较简单⁷。

用户可以采用同样的方法来查看 O/U 文件。dimensions 是速度的单位。内部场初始化为 0，对于速度来说，这个值必须是一个矢量，即：uniform (0 0 0)。请查阅 4.2.5 节以获取更多信息。

速度场和压力场的 frontAndBack 都为 empty，其它的 patches 均为壁面：fixedWalls 我们指定无滑移边界条件，因此为 fixedValue，以及 value uniform (0 0 0)。我们假设顶部的壁面以每秒 1 米的速度在 x 方向移动，这是一个固定速度即 fixedValue，其具体的值为 uniform(1 0 0)。
或用 type noSlip

2.1.1.3 物理特性

物理特性 → transportProperties:

算例的物理特性存储在以 Properties 为后缀的文件中，对于 icoFoam 算例，唯一需要给定的参数即为运动粘度，它存储在 **transportProperties** 字典文件中。用户可以打开 transportProperties 文件来查看运动粘度是否设置正确。运动粘度的关键词为 nu，类似于希腊字母 ν 的发音。开始，我们的算例雷诺数为 10，雷诺数如下定义：

$$Re = \frac{d|U|}{\nu} \quad (2.1)$$

d 和 |U| 分别是特征长度和速度，ν 为运动粘度。此处 d 为 0.1m，|U| 为 1m/s，运动粘度为 0.01 m²/s。所以雷诺数为 10。正确地指定运动粘度场应该如下所示：

```

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

```

2.1.1.4 控制

在 controlDict 字典内，我们可以对时间步、输入输出时间、场数据读取和写入等进行控制。用户应该查阅这个文件，作为一个算例控制文件，它位于 system 文件夹下。

起始时间、终止时间以及求解时间步必须要进行设定。OpenFOAM 提供了灵活的时间

⁷ 本算例压力边界条件全部为 zeroGradient，因此需要设置压力参考点和参考值。此处压力设置 0,10,100 均可。不可压缩流中关心的是相对压力而不是绝对压力。

步控制, 请查阅 4.3 节。在这个教程里我们打算在 $t = 0$ 的时候开始进行计算, 这意味着 OpenFOAM 需要读取 0 文件夹下的场数据, 请查阅 4.1 节来获取更多有关算例文件结构的相关信息。因此我们设定 startFrom 关键字信息为 startTime, 然后指定 startTime 为 0。

运算结束的时候, 我们希望能达到稳定的状态, 即流体在空腔内循环。一个通用准则是, 层流下流体应该穿过计算域 10 次。在这个算例中, 我们没有进口和出口, 因此流体没有穿过计算域。因此结束时间应该设置为在空腔内循环 10 次的时间。例如 1s。实际上, 我们发现 0.5s 就可以了, 故我们设置结束时间为 0.5s。我们通过把 stopAt 关键字指定为 endTime 然后把 endTime 指定为 0.5 来完成。

现在我们需要设定时间步, 即 deltaT。当运行 icoFoam 的时候, 为了达到数值稳定以及时间计算精度, 库郎数⁸应该小于 1。每个网格的库郎数这样定义:

$$Co = \frac{\delta t |U|}{\delta x} \quad (2.2)$$

δt 为时间步, $|U|$ 为网格内的速度模量, δx 为速度方向的网格长度。流体域内速度并不均匀, 为了保证库郎数在各处都小于 1, 我们依据最大速度和最小网格尺寸来计算库郎数, 在保证其小于 1 的前提下指定时间步。在这个算例中网格大小是固定的, 因此在顶盖附近, 速度接近 1m/s 的地方库郎数最大, 网格大小为:

$$\delta x = \frac{d}{n} = \frac{0.1}{20} = 0.005m \quad (2.3)$$

为了使库郎数在整个流体域都小于或等于 1. 时间步必须小于或等于:

$$\delta t = \frac{Co \delta x}{|U|} = \frac{1 * 0.005}{1} = 0.005s \quad (2.4)$$

随着计算的进行, 我们希望在某个固定的时间间隔下写入数据。以便我们可以在后处理中进行查看。对于如何设定写入时间步有很多选择, 它可以通过 writeControl 关键字来控制。我们决定在 0.1s, 0.2s, ..., 0.5s 写入我们的结果。由于计算时间步为 0.005s, 因此我们需要每 20 个时间步输出一次结果, 这样我们设定 writeInterval 为 20。 $Ts = 0.005$

$$\text{Interval} = \frac{0.1}{0.005} = 20$$

OpenFOAM 在当前的时间步后创建一个新的文件夹, 例如 0.1。正如 4.1 节所说, 每个时间步下有一系列数据。对于 icoFoam 求解器, 它把速度和压力场写入在每个时间步文件夹中。对于这个算例, controlDict 信息如下:

```

17
18 application           icoFoam;
19 startFrom             startTime;
20 startTime              0;
21
22 startTime             endTime;
23
24 stopAt                0.5;
25
26 endTime               0.5;
27
28 deltaT                0.005;
29
30 writeControl          timeStep;
31
32 writeInterval          20;

```

⁸ Courant number

```

33
34 purgeWrite      0;
35
36 writeFormat     ascii;
37
38 writePrecision   6;
39
40 writeCompression off;
41
42 timeFormat      general;
43
44 timePrecision    6;
45
46 runTimeModifiable true;
47
48
49 // ****

```

2.1.1.5 离散方法和矩阵求解器设置

用户在 system 下的 fvScheme 文件中指定有限体积法的离散格式。在 system 文件夹下的 fvSolution 下指定方程组矩阵求解器、残差以及其它算法控制。用户可以查看这些字典文件，但是目前除了 fvSolution 里面的 PISO 子字典中的 pRefCell 以及 pRefValue，我们不讨论这些关键词信息。在一个封闭的不可压体系中，例如这个空腔，压力是相对的。跟绝对压力值来说，压力范围比较重要。在这种例子中，求解器通过 pRefCell 以及 pRefValue 来设置参考值。在这个例子中，我们都设置为 0。改变其中任何一个值都会改变绝对压力场。但是不会改变相对压力场和速度场。

2.1.2 查看网格

在这个例子运行之前，我们最好检查一下网格以查看是否有错误。网格在 OpenFOAM 自带的后处理程序 paraFoam 中查看，它通过在已经切换到算例目录的终端下键入

paraFoam

来启动。其实它也可以在非算例目录下的终端运行，但应该使用-case 命令参数，例如：

paraFoam –case \$FOAM_RUN/tutorials/incompressible/icoFoam/cavity

如图 6.1，这会启动 paraview 窗口，在 Pipeline Browser 下，用户可以看出 paraview 已经打开了这个算例模块并称之为 cavity_OpenFOAM，在应用 apply 按钮之前，用户需要在 Mesh Parts 里面选择要显示的几何。因为这个算例文件很小，最简单的是在 Mesh Parts 旁的窗口里选择全部几何数据，这会自动的检查所有几何单元。然后用户点击 Apply，网格文件就会被 paraview 加载。

然后用户应该打开 Display 面板⁹，它控制所选模块的视觉效果。在 Display 面板下用户应该参照图 2.3 所示来进行操作：(1)设置 Color By Solid Color; (2)点击 Set Ambient Color 然后选择合适的颜色，例如黑色（如果是白色的背景色）；(3)在 Style 面板中¹⁰。

⁹ 不同的 paraview 版本，此处有区别，但是大体相同。最新版的 paraview 没有 display 单独面板

¹⁰ 本手册为了和英文版的页码保持一致，会对排版进行适当调整

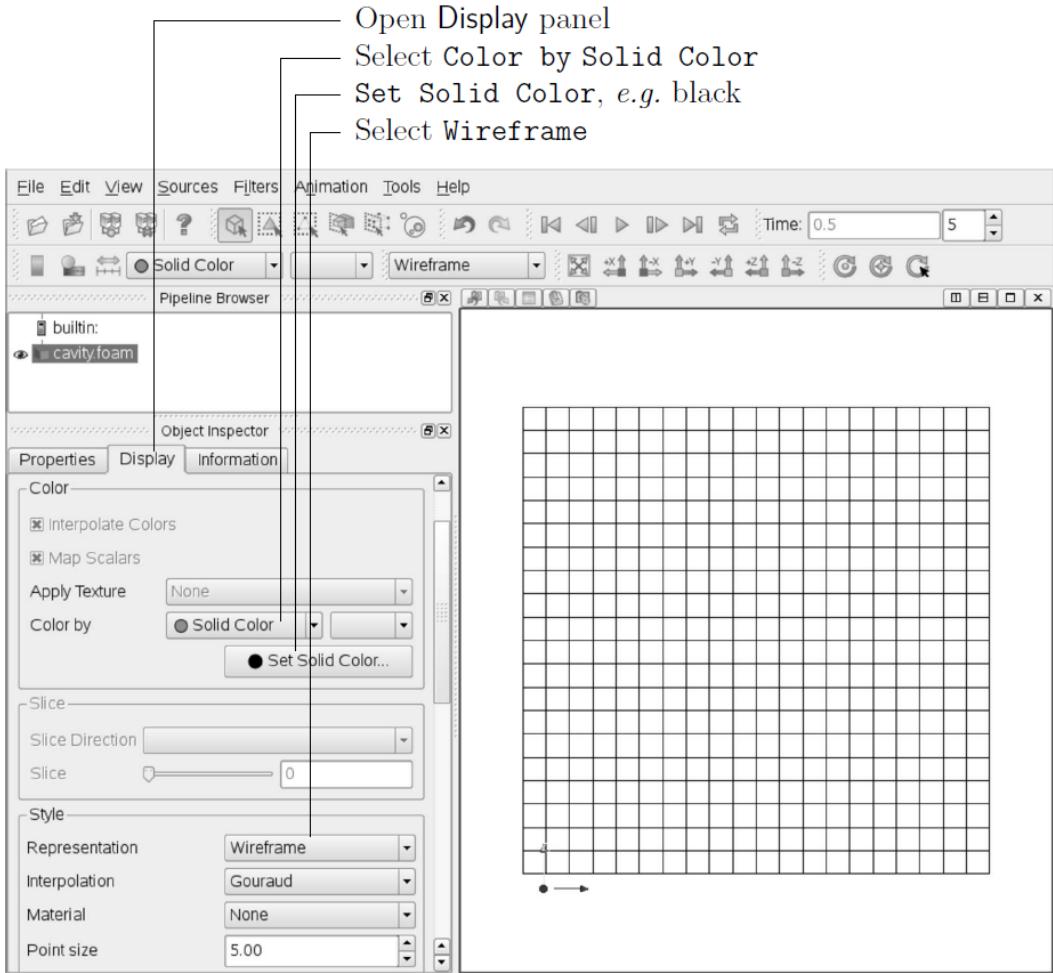


图 2.3: 通过 paraFoam 查看网格

的 Representation 菜单下，选择 Wireframe。背景色可以通过顶部菜单栏中的 Edit 下的 View Setting 中来设置。

用户第一次运行 paraview 的时候，我们建议按照 6.1.5 节来进行操作。由于这个算例是一个二维算例，我们建议在 Edit 下的 View Setting 中的 General 面板中勾选 Use Parallel Projection。在 Annotation 面板下我们也可以选择打开或者关闭 Orientation Axes。

2.1.3 运行算例

正如其它任何 UNIX/Linux 可执行程序一样，OpenFOAM 程序可以通过下面两种方式来运行，一个是前置进程，就是在运行下一个命令之前，得在这个程序运行完毕之后才可以。另一种是后台进程，这样的话程序自动进入后台，终端可以继续运行其它命令。

我们以前置进程的方法运行 icoFoam，一种方法是，可以在算例目录下的终端键入：

```
icoFoam
```

来运行，另一种方法是可以通过-case 命令参数来指定算例目录：

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

在终端的窗口里，我们会看到命令的输出信息，它告诉用户当前的时间步，最大库郎数，所有场的初始残差和最终残差。

2.1.4 后处理

在把结果写入时间步文件之后，我们可以用 paraFoam 来进行后处理，回到 paraFoam 窗口，选择 cavity.OpenFOAM 模块。如果当前没有显示算例的网格，请确保 cavity.OpenFOAM 处于高亮绿色。并且在左边高亮显示一个眼睛的图标，这样就会显示当前的算例；

为了让 paraFoam 展示所需要的数据信息，我们首先要确保加载 0.5s 时间步的数据。如果运行算例的时候 paraview 已经打开，新的时间步数据不会自动加载。为了加载新的时间步数据，用户应该在 Properties 中点击 Refresh Times。这会自动加载新的时间步数据文件。

2.1.4.1 等值面和云图

如果用户打算查看压力场，则应该切换到 Display 面板，这里控制所选模块的视觉效果。为了简单地做一个压力云图，用户应该按照图 2.4 来设置。在 Style 面板中的 Representation 菜单下选择 Surface；在 Color 面板中，选择 Color by $\circ p$ ，和 Rescale to Data Range。现在我们来查看 0.5s 的结果，参见图 6.4，在顶部的 paraview 窗口中，菜单栏的下面有一系列工具栏，其中有 VCR Controls 或者 Current Time Controls，用户可以在里面把时间步改为 0.5s。这样，压力场就会呈现，正如期望的，空腔的左部顶端为低压区，右部顶端为高压区，详细请查看图 2.5。

由于我们选择的是 $\circ p$ ，在这种情况下，每个网格压力场都进行了插值，这样压力场看起来是连续的。如果用户在 Color by 菜单下选择： $\circ \bar{p}$ ，这样会显示每个网格的压力值，并且整个场的每个网格值并不是连续的。

颜色条可以通过点击 Active Variable Controls 工具栏下的 Toggle Color Legend Visibility 按钮来显示，也可以通过选择 View 菜单下的 Show Color Legend 来实现。点击 Active Variable Controls 工具栏下或者 Display 窗口下的 Color 面板中的 Edit Color Map，用户可以对颜色条进行设置，例如字体大小、字体、数据格式等。颜色条一般位于窗口的右边，它可以用鼠标来拖拽。

Paraview 默认使用蓝-白-红颜色条，而不是常见的蓝-绿-红（彩虹）颜色条。因此当用户第一次打开 paraview 的时候，它们可能打算改变颜色条的样式。这可以通过在 Color Scale Editor 中选择 Choose Preset，然后选择 Blue to Red Rainbow。在点击 OK 确认之后，用户可以点击 Make Default，这样 paraview 就会默认采用这种颜色样式。

如果用户旋转视角，可以发现整个几何表面都为压力场的颜色。如果想创建一个云图，用户应该首先创建一个切面¹¹，或者通过 slice 滤镜来创建一个 slice。

¹¹ Paraview 可以在 source 菜单中选择 plane 来创建切面

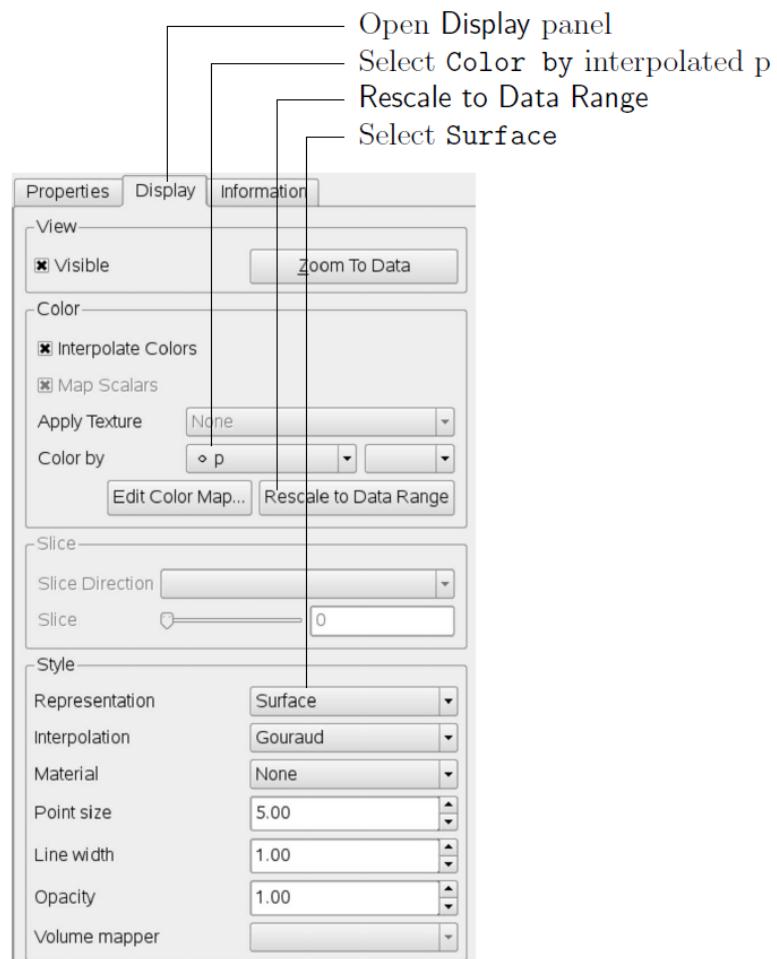


图 2.4 cavity 算例的压力云图设置

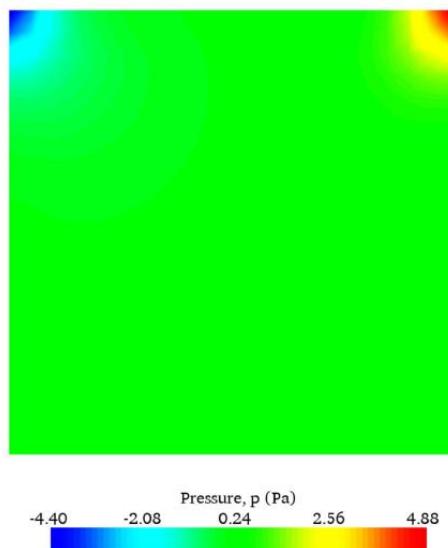


图 2.5 cavity 算例的压力云图

详细内容请参考 6.1.6.1 节。切割平面中心在(0.05,0.05,0.05), 法向为(0,0,1) (或者点击 Z Normal 来自动设定)。生成切割平面之后, 云图可以通过应用 Contour 滤镜来生成, 更多内容请参阅 6.1.6。

2.1.4.2 矢量图

在我们绘制矢量图之前, 我们有必要屏蔽其它模块, 例如上文生成的 slice 和 Contour 模块。这有两种选择: 要么全部删除, 即首先在 Pipeline Browser 中高亮相关模块, 然后在相应的 Properties 中点击 Delete¹²; 或者关闭模块左边的眼睛标示以隐藏它。

现在我们打算在每个网格中心来生成矢量图。正如 6.1.7.1 所说, 我们首先需要把数据处理到网格中心。首先在 Pipeline Browser 中高亮 cavity.OpenFOAM, 然后用户在 Filter -> Alphabetical 菜单中选择 Cell Centers 滤镜并应用。

高亮 Pipeline Browser 中的 Centers, 然后在 Filter->Alphabetical 中选择 Glyph 滤镜。会出现图 2.6 所示的面板。在这个面板的 vectors 菜单中, 速度场 U 被自动选取, 因为它是目前唯一的矢量场。现在我们想查看整个流体域的矢量场, 用户应该选择 off, 并设置 Set Scale Factor 为 0.005, 点击 Apply, 矢量场会展现, 但可能为单一颜色, 例如白色。用户可以通过在 Display 面板中选择 Color by U 来在箭头上显示速度的大小。除此之外, 用户也需要在 Edit Color Map 中选择 Show Color Legend。请见图 2.7, 我们设定大写的 Times Roman 字体作为 Color Legend, 在 Label Format 中键入%#6.2f, 以及取消选择 Automati Label Format 并为其指定为上下 2 个固定值。背景颜色在 View Settings 的 General 面板中设定为白色, 详见 6.1.5.1。

我们注意到在左右的壁面上, 矢量指示其穿透墙壁。然而, 在仔细观察之后, 我们发现垂直于壁面的矢量大小为 0。这种令人困惑的结果是 Paraview 导致的而不是计算结果的问题, 这是因为我们选择了 glyph scaling 为 off, 因此 paraview 会显示所有网格的矢量, 即使其大小为 0。

2.1.4.3 绘制流线图

在用户在 paraview 中进行后处理之前, 用户应该把之前绘制的矢量图取消显示, 现在我们打算创建流线图, 更详细的内容请见 6.1.8 节。

在 pipeline browser 中把 cavity.OpenFOAM 模块高亮, 然后在 Filter 中选择 Stream tracer 并 Apply。如图 2.4, 会出现一个 Properties 窗口, 其中的 Seed 应该选择 Line Source, 它垂直于几何中心, 例如从(0.05,0,0.005)到(0.05,0.1,0.005), 针对我们显示的这个图, 我们指定: 点的 Resolution 为 21, Max Propagation 长度为 0.5, Initial Step Length 为 Cell Length 0.01, 以及 Integration Direction BOTH, 和默认的 Runge-Kutta 2 Intergrator Type。

再点击 Apply 之后, 流线图就会生成了。然后用户应该选择 Filter 中的 Tube 滤镜来创造高清晰度的流线, 图中我们就采用了此方法。在这个算例中, 我们设置

¹² 直接在模块上右击并 delete 更为方便

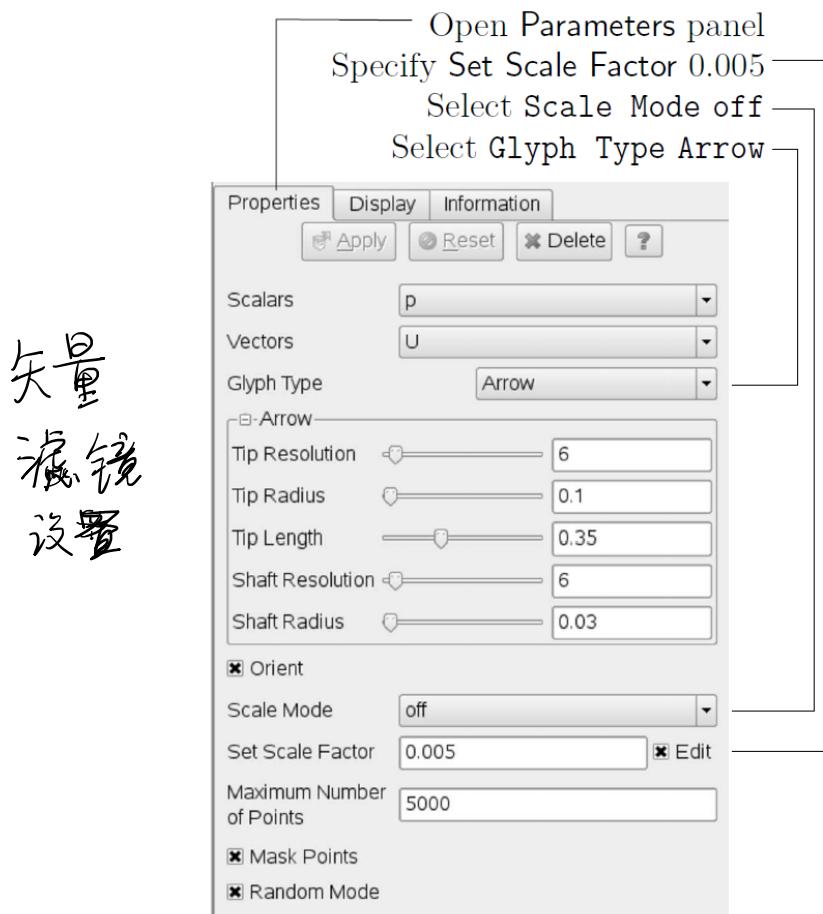


图 2.6 矢量滤镜设置

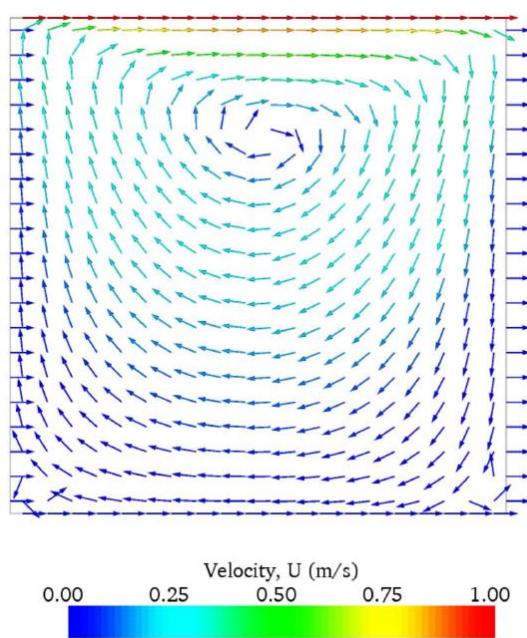


图 2.7 cavity 算例的速度矢量图

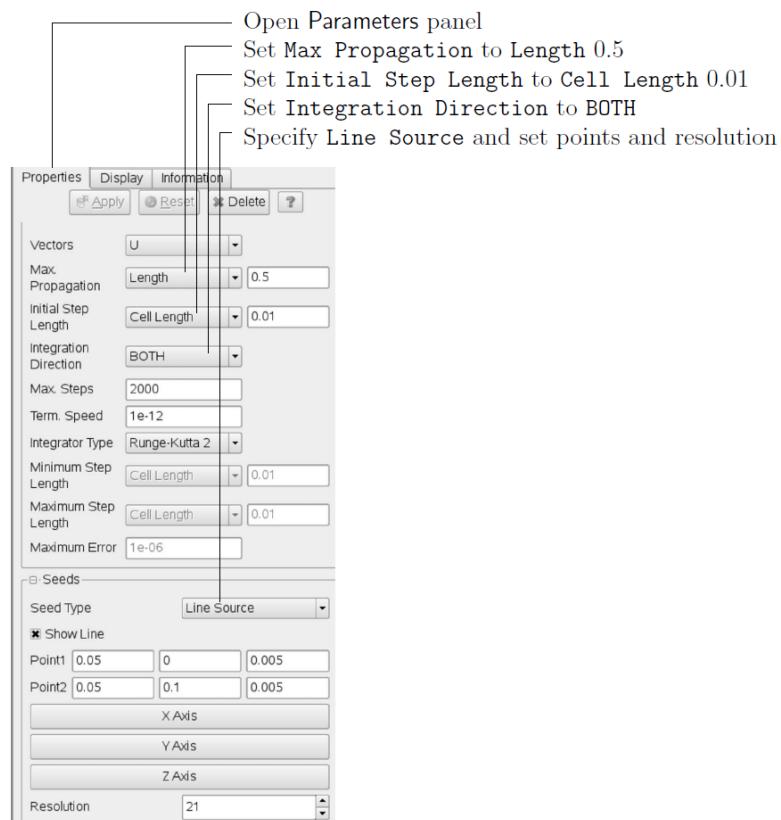


图 2.8 流线图设置

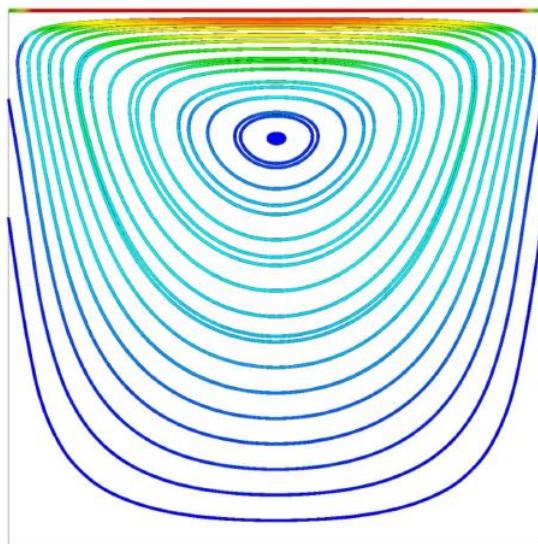


图 2.9 cavity 算例的流线图

Num.sides 为 6; Radius 为 0.0003; Radius factor 为 10。管线用速度大小来显示，然后点击 Apply，图 2.9 会生成。

2.1.5 网格细化

我们把网格数在每个方向上增加 2 倍，并把粗粗网格的结果投影到细网格并作为初始条件。然后对比细网格和粗网格的结果。

2.1.5.1 使用存在的算例创造新算例

现在我们打算在 cavity 算例的基础上创造一个新的算例并称为 cavityFine，用户应该复制 cavity 算例并编辑必要的文件。首先，用户需要在 cavity 同级的目录下创建一个新的算例目录：

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

然后用户需要把 cavity 算例下的文件拷贝到 cavityFine，然后进入 cavityFine 算例目录

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

2.1.5.2 生成细网格

我们打算用 blockMesh 来增加网格数量。用户应该用一个文本编辑器打开 blockMeshDict 并对其进行编辑。在 blocks 关键词下来指定 blocks。详细的句法请参见 5.3.1.3。在目前的阶段，我们只需知道 hex 后面是 block 的各个顶点，然后是每个方向的网格数量。在 Cavity 算例中，我们设置为(20 20 1)，在这个算例中我们设置为(40 40 1)并保存文件。新的细化网格应该通过运行 blockMesh 来生成。

2.1.5.3 投影算例结果

mapFields 程序把一个给定几何的场信息，投影到另一个几何的对应场。在我们的例子中，场被认为是一致的，因为被投影场和原始场的几何，边界类型和条件都是一样的。因此在我们执行 mapFields 程序时，使用-consistent 附加命令选项。

mapFields 首先读取某个时间步的场信息，这通过被投影场算例的 controlDict 中的 startFrom/startTime（例如需要被投影的时间步）来指定，在这个算例中，我们打算把粗网格的最终结果投影到 cavityFine 的细网格中，由于 cavity 算例的最终结果存储在 0.5 文件夹中，因此 controlDict 文件中的 startTime 应该设定为 0.5，startFrom 设定为 startTime。

现在这个算例就可以运行 mapFields 了，键入 mapFields -help 可以显示 mapFields 可以附加的命令参数。我们看出，提供源文件地址是必须的，另外我们还需要使用-consistent 选

项，因此在 cavityFine 终端里输入为：

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

程序应该输出以下信息：

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

Create databases as time
Case   : ../cavity
nProcs : 1

Source time: 0.5
Target time: 0.5

Create meshes

Source mesh size: 400  Target mesh size: 1600
Consistently creating and mapping fields for time 0.5
Creating mesh-to-mesh addressing ...
Overlap volume: 0.0001
Creating AMI between source patch movingWall and target patch movingWall ...

    interpolating p
    interpolating U

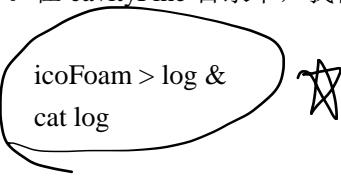
End
```

2.1.5.4 控制参数

正如 2.1.1.4 所说，在算例求解的时候，最好要保持 Co 小于 1¹³。网格大小减半了，因为时间步长必须减半。因此，controlDict 字典中的 deltaT 需要设定为 0.0025。目前场信息设置为每多少个固定时间步写一次。现在我们学习如何在固定时间间隔来写文件¹⁴。在 controlDict 中的 writeControl 中，之前设置为每个固定 timeStep 输出，我们也可以指定为 runTime 来指定每多少个时间间隔来写文件。在这个算例中我们打算每 0.1 秒写结果。因此我们设定 writeControl 为 runTime，writeInterval 为 0.1。最后，由于算例从一个粗网格的结果来进行计算，我们只需要运行短暂的时间就可以达到稳态¹⁵。因此我们将 endTime 设置为 0.75，确认所有的设置正确后我们保存文件。

2.1.5.5 后台运行

用户应该知道如何以后台程序的方式来运行 icoFoam，并把输出信息输入到 log 文件以便查看。在 cavityFine 目录下，我们输入：



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

¹³ 库郎数的问题是 CFD 的基本问题，建议参阅网上相关资料以加深理解

¹⁴ 固定时间步间隔为每多少个 deltaT 一写，固定时间间隔为每 0.5 秒一写之类

¹⁵ 原文为 steady-state，由于 icoFoam 为瞬态求解器，原文的意思即为稳定的状态

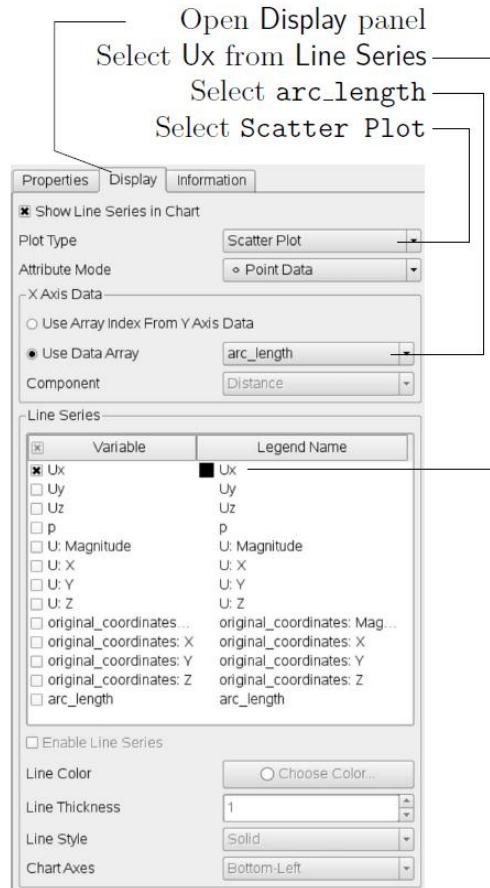


图 2.10: 选择 plot 场

2.1.5.6 在细网格上绘制矢量

用户可以在 paraview 同时中打开多个算例，每个算例只是 Pipeline Browser 下的一个模块。但是在打开一个新的 paraView 算例的时候会有一个小小的不便之处。即所打开的文件必须具有相应的扩展名。然而，OpenFOAM 算例的特点是每个算例具有多个文件（夹），它们以一定的文件结构存储在一个文件夹下。因此，paraFoam 脚本自动创建一个后缀为.OpenFOAM 的文件，这样 cavity 算例就被称为 cavity.OpenFOAM。

然而，如果用户想在 paraview 中打开新的算例，它们需要创建这样一个空文件。例如，如果想加载 cavityFine 算例，应该执行以下命令¹⁶:

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

现在，点击 File 菜单中的 Open，并在导航窗口中选择 cavityFine.OpeFOAM 模块。paraview 就可以加载 cavityFine 算例了。用户可以在这个细化的网格上用 paraview 来绘制矢量图。如果想对比数据，可以把两个算例的矢量图同时打开进行。

¹⁶ 对于熟悉 windows 操作系统的用户，也可以在窗口中右键创建空白文档后改名为 cavitiFine.OpenFOAM

2.1.5.7 绘图

用户可能打算对速度的某个量来进行可视化处理并沿着计算域内的一条线来绘制一个 2 维图。OpenFOAM 处理这种数据得心应手。有大量的程序可以用来进行数据处理操作。一些简单的后处理运算被整合到一个单一的程序：foamCalc。它这样来被执行：

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

计算操作类型由 calcType 来指定。在需要写的时间步下，我们植入了下面的数学运算：addSubtract; randomise; div; components; mag; magGrad; magSqr; interpolate。用户可以通过调用一个不存在的 calcType 来获取所有的 calcType 类型，这时 foamCalc 会抛出错误信息并列出所有可选类型¹⁷。例如：

```
>> foamCalc xxxx
Selecting calcType xxxx
unknown calcType type xxxx, constructor not in hash table
Valid calcType selections are:
8
(
    randomise ✓
    magSqr ✓
    magGrad ✓
    addSubtract ✓
    div ✓
    mag ✓
    interpolate ✓
    components ✓
)
```

其中 Components 和 mag 可以对速度进行操作。当在某个算例下，例如 cavity 下，运行 foamCalc components U 的时候，它在每个时间步读取速度场文件，并在相应的文件夹下写入 Ux, Uy, Uz 的场信息。类似，foamCalc mag U 在每个对应的时间步写入 magU 这个标量场，即为速度的模量。

用户可以在 cavity 和 cavityFine 算例中以 components 作为 calcType (运算类型) 来运行 foamCalc。例如，在 cavity 算例中，用户应该切换到 cavity 目录下并这样执行命令：



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

foamCalc components U

这样，在 paraview 中就可以绘制速度的分量图了。Paraview 可以快速，方便的对图表进行控制，因此输出的图表是非常标准化的。但是如果为了发表文章，用户可能喜欢用专业的画图软件，例如 gnuplot 和 Grace/xmgr 等来对原始数据进行操作来绘制。如果想进行这样的操作，我们建议使用 sample 程序，详见 6.5 和 2.2.3 节。

 在开始绘图之前，用户需要让 paraview 加载新生成的 Ux, Uy, Uz 场。用户可以单击 Properties 面板下的 Refresh Times 按钮，这样 paraview 就会加载新的场并出现在 volume field 列表中。选中需要加载的新场，单击 Apply 按钮。

¹⁷ 普适性很强的关键技术，在某些字典文件中，如果用户想知道存在某些植入的关键词，可以故意写错备选关键词以查看 OpenFOAM 输出的备选关键词。例如，在速度场中，用户可以定义 type fixedV；在运行的时候，终端会报错输出大量的已经植入的速度边界类型。

另外我们会发现，如果在 Mesh Parts 中我们选择了边界区域，那么在边界处数据的插值有错。因此用户需要在 Mesh Parts 面板中，取消勾选界面，例如 movingWall, fixedWalls 以及 frontAndBack 并 Apply。

现在，为了在 paraview 中绘图，用户应该选择当前模块例如 cavity.OpenFoam，然后在 Filter 选项中选择 Data Analysis 下的 Plot Over Line。这会在现存的 3D 视图旁边打开一个新的 XYPlot 窗口，同时在 Properties 面板下创建一个 PlotOverLine 模块，用户可以设置线的起始端和结束端。在这个例子中，用户应该创建一个穿过计算域中心的垂直线。例如，在 Point1 中设置起始点为(0.05,0,0.005)，在 Point2 中设置终点为(0.05,0.1,0.005)。Resolution 可以设置为 100。

单击 Apply，在 XYPlot 视图中会生成一个图表。在 Display 面板中，用户应该把 Attribute Mode 设置为 Point Data。X Axis Data 我们选择 Use Data Array 以及 arc_length，这样 x 轴代表了离底距离。

接下来，用户可以在 Display 窗口中的 Line Series 面板中选择显示的场，在现有的标量场中，我们可以看见默认设置就有速度的模以及 3 个分量。例如 U:X¹⁸，这意味着我们不需要利用 foamCalc 来生成 Ux 数据。用户应该仅仅选择 Ux 或者 U:X。在旁边的彩色方框内代表的是线条的颜色。用户可以对此进行随意的设置。

如果想对图标进行润饰，用户可以在 Line Series 面板下修改设置，例如 Line color, Line Thickness, Line Style, Marker Style 以及 Chart Axes。

同时，用户注意到曲线图的左上方有一些小工具可供使用。第三个按钮可以用来对视觉选项进行设置，例如设置图标的名称，每个轴的单位，并且可以设置字体，颜色以及坐标轴名称，坐标轴刻度也可以分为线性和 log 坐标轴。

图 2.11 是使用 paraview 绘制的图表。用户可以任意的来绘图。图 2.11 我们使用了下述选项：采用标准格式，指定了坐标轴范围，标题为 Sans Serif 12 号字。这个图我们没有激活线选项（在 Display 面板中选中 Enable Line Series），相反的我们用一系列的点来绘制。需要注意的是，如果这个按钮是灰色的，我们可以通过在 Line Series 面板中选择或反选需要绘制的场来激活它。一旦激活 Enable Line Series，用户就可以依据它们的喜好随意的选择 Line Style 和 Marker Style。

2.1.6 网格非均匀分布

在某些区域，真实解的分布型线会和选定的数值格式的形式有很大的不同。如果这种区别很大的话，在这些区域求解的误差会比较大。例如，如果真实解本身为线性形式的话，基于线性的数值格式才会预测准确的解。如果某些区域的真实解本身并不是线性形式，那么在这些区域（比如梯度较大的区域）的求解结果会有较大误差。细化网格会使误差减小。

在对任何算例进行求解之前，我们都应该对解的形式有一个直觉的概念。这样我们就可以预测哪里误差比较大然后在这一部分细化网格，以使得这些区域的网格较小。在 cavity 算例中，速度最大的变动发生在壁面处，因此我们的教程中，这一部分的网格将会被非均匀化。

¹⁸ 不同版本的 paraview 可能有不同的名称，例如 U(1), U(x)

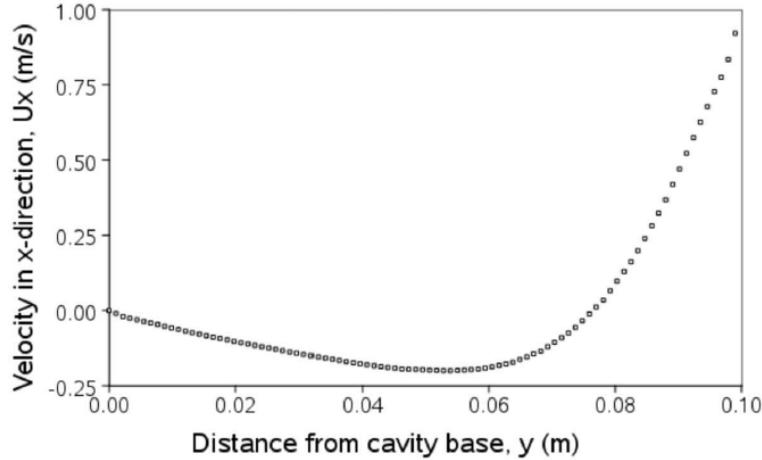


图 2.11：在 paraFoam 中绘图

通过这种方法，我们就可以利用同样数量的网格，在不增加计算机资源的情况下获得更准确的解。

我们接下来针对顶盖驱动流算例创建一个 20*20 的网格，壁面处引入网格非均匀特性，然后把 2.1.5.2 节中的细网格算例结果被投影到非均匀网格中作为初始条件。并把非均匀网格的计算结果和之前的算例结果进行对比。在非均匀的网的算例中，blockMeshDict 字典文件的改动非常大，因此我们重新创建一个算例，这个算例可以在\$FOAM_RUN/tutorials/incompressible/icoFoam 文件目录的 icoFoam 目录下找到。

2.1.6.1 创建非均匀化网格

在这个算例中，这个网格需要 4 个 blocks，上下面和左右面都需要网格非均匀化。图 2.12 中即为 block 结构。用户可以在 cavityGrade 的 constant/polyMesh 下查阅 blockMeshDict 文件，我们把这个字典文件展示如下，每个 block 在 x,y 方向都有 10 个网格，最大网格和最小网格的大小比为 2：

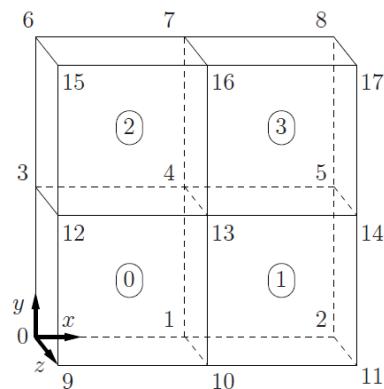


图 2.12：顶盖驱动流的非均匀 block 结构

```

19 vertices
20 (
21     (0 0 0)
22     (0.5 0 0)
23     (1 0 0)
24     (0 0.5 0)
25     (0.5 0.5 0)
26     (1 0.5 0)
27     (0 1 0)
28     (0.5 1 0)
29     (1 1 0)
30     (0 0 0.1)
31     (0.5 0 0.1)
32     (1 0 0.1)
33     (0 0.5 0.1)
34     (0.5 0.5 0.1)
35     (1 0.5 0.1)
36     (0 1 0.1)
37     (0.5 1 0.1)
38     (1 1 0.1)
39 );
40
41 blocks
42 (
43     hex (0 1 4 3 9 10 13 12) (10 10 1) simpleGrading (2 2 1)
44     hex (1 2 5 4 10 11 14 13) (10 10 1) simpleGrading (0.5 2 1)
45     hex (3 4 7 6 12 13 16 15) (10 10 1) simpleGrading (2 0.5 1)
46     hex (4 5 8 7 13 14 17 16) (10 10 1) simpleGrading (0.5 0.5 1)
47 );
48
49 edges
50 (
51 );
52
53 boundary
54 (
55     movingWall
56     {
57         type wall;
58         faces
59         (
60             (6 15 16 7)
61             (7 16 17 8)
62         );
63     }
64     fixedWalls
65     {
66         type wall;
67         faces
68         (
69             (3 12 15 6)
70             (0 9 12 3)
71             (0 1 10 9)
72             (1 2 11 10)
73             (2 5 14 11)
74             (5 8 17 14)
75         );
76     }
77     frontAndBack
78     {
79         type empty;
80         faces
81         (
82             (0 3 4 1)
83             (1 4 5 2)
84             (3 6 7 4)
85             (4 7 8 5)
86             (9 10 13 12)
87             (10 11 14 13)
88             (12 13 16 15)
89             (13 14 17 16)
90         );
91     }
92 );
93
94 mergePatchPairs
95 (
96 ;
97
98 // ****

```

目前我们已经对 blockMeshDict 字典文件很熟悉，用户可以直接执行 blockmesh 命令，非均匀化之后的网格可以像之前一样执行 paraFoam 来查看，详见 2.1.2 节。

2.1.6.2 调整时间步

因为顶盖附近的速度最高，网格最小，因此顶盖附近的库郎数也会变大，原因请参考 2.1.1.4 节。因此我们首先算一下顶盖附近网格的大小，以此来计算本算例合适的时间步。

当使用一个非均匀网格的时候，`blockMesh` 会计算网格大小。如果长度为 L ，内置 n 个网格单元，最末端和起始端网格比为 R ，那么最小的网格大小就为 δx ，其大小为：

$$\delta x = l \frac{r-1}{\alpha r - 1} \quad (2.5)$$

r 是相邻网格单元的大小比：

$$r = R^{\frac{1}{n-1}} \quad (2.6)$$

且有：

$$\alpha = \begin{cases} R & \text{for } R > 1 \\ 1 - r^{-n} + r^{-1} & \text{for } R < 1 \end{cases} \quad (2.7)$$

对于 `cavityGrade` 算例，每个方向的网格数为 10，最大网格单元和最小网格单元的大小比为 2，`block` 的高宽为 0.05m。因此，最小的网格单元为 3.45mm。从方程 2.2 可以看出，时间步应该小于 3.45 毫秒，这样才能保证库郎数小于 1。为了方便的写入每个时间步的结果¹⁹，`deltaT` 设置为 2.5 毫秒。`writeInterval` 设置为 40，因此每隔 0.1 秒写一个结果文件。所有这些设置可以在 `cavityGrade/system/controlDict` 里面查看。

本算例的 `startTime` 需要设置为 `cavityFine` 的结束时间，即 0.7 秒。由于 `cavity` 和 `cavityFine` 算例都已经收敛的很好了，因此这个算例我们只需要计算 0.1s，结束时间 `endTime` 设为 0.8。

2.1.6.3 投影场

在 2.1.5.3 中，我们使用 `mapFields` 把 `cavity` 的网格结果投影到 `cavityFine` 中²⁰。现在我们把 `cavityFine` 的结果投影到 `cavityGrade` 中：

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

接下来我们在算例目录下运行 `icoFoam` 并监控运行的输出信息。对结果进行后处理并和之前的算例结果相对比。如何进行后处理请参阅 2.1.5.6 和 2.1.5.7 节。

2.1.7 增加雷诺数

目前的算例雷诺数非常低，仅为 10，其可以快速的达到稳定状态。并且只有一个非常小的涡旋存在于空腔的底角。我们现在把雷诺数增加到 100。这会明显的增加收敛时间。首先，我们使用最粗糙的 `cavity` 网格。用户应该拷贝 `cavity` 算例，并把它命名为 `cavityHighRe`。这个可以这样来实现：

¹⁹ 这样下面就可以每 40 步一写来写入一个整数的时间步

²⁰ 原文有误

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

2.1.7.1 前处理

进入 cavityHighRe 算例，编辑 transportProperties 字典文件，由于雷诺数需要扩大 10 倍。我们可以通过把粘度降低 10 倍来实现。变为 $1e-3 \text{ m}^2/\text{s}$ 。现在我们可以以 cavity 算例的最终结果为初始条件来进行算例。这通过把 startTime 改为 latestTime 来实现，这意味着 icoFoam 会从最后一个时间步来进行计算，对于当前这个算例是 0.5s。结束时间 endTime 应该是 2s。

2.1.7.2 运行算例

接下来我们应该在算例目录下运行算例并查看输出。如果以后台进程的方式来运行算例的话，会使用到下面的 UNIX 命令：

- **nohup:** 当用户退出登陆的时候，这个程序依然会继续运行
- **nice:** 调整进程优先级，-20 对应最高优先进程，19 对应最低优先进程

有时用户会希望在远程运行算例，并且不打算进行监控，且想把进程调为低优先级。这个时候这两个命令会非常有用。当用户退出远程系统的时候，nohup 会让进程继续运行，nice 可以设置为 19 来降低进程优先程度。对于我们的这个算例，我们可以这样来执行命令：

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

在之前的算例中，我们会发现 icoFoam 在非常短的时间内就停止了对速度进行迭代²¹，但它一直对压力进行求解直到求解终止。实际上，在 icoFoam 停止求解速度场，并且在压力的初始残差小于 fvSolution 设定的残差值（一般为 $10e-6$ ）之后，求解就已经收敛，在求解器把结果场写出来的时候就可以停止了²²。例如（参见下面的 log 文件，这个是 cavityHighRe 算例的计算输出）：速度在 1.395 秒以后已经收敛。并且压力的初始残差非常小；No Iterations 0 表示速度求解已经停止：

```
1 Time = 1.43
2
3 Courant Number mean: 0.221921 max: 0.839902
4 smoothSolver: Solving for Ux, Initial residual = 8.73381e-06, Final residual = 8.73381e-06, No Iterations 0
5 smoothSolver: Solving for Uy, Initial residual = 9.89679e-06, Final residual = 9.89679e-06, No Iterations 0
6 DICPCG: Solving for p, Initial residual = 3.67506e-06, Final residual = 8.62986e-07, No Iterations 4
7 time step continuity errors : sum local = 6.57947e-09, global = -6.6679e-19, cumulative = -6.2539e-18
8 DICPCG: Solving for p, Initial residual = 2.60898e-06, Final residual = 7.92532e-07, No Iterations 3
9 time step continuity errors : sum local = 6.26199e-09, global = -1.02984e-18, cumulative = -7.28374e-18
10 ExecutionTime = 0.37 s ClockTime = 0 s
11
12 Time = 1.435
13
```

²¹ 通过 No Iterations 来查看

²² 在停止求解器计算之前，务必查看是否已经写入结果。因为当求解器没有写入结果的时候，它也可以被终止并不写入最后的计算结果

```

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

```

2.1.8 高雷诺数流动

我们通过 paraFoam 来查看结果并查看速度矢量。可以发现角落里的第二个窝略微变大。用户可以继续减少粘度来增加雷诺数并重新运行算例。涡的数量会增加，因此就需要在涡的附近增加网格来求解更复杂的流型。另外，高雷诺数也增加了收敛的时间。用户应该监控残差，并延长求解时间以确保收敛。

实际上，在求解湍流域的时候，这种提高网格数量和增加求解时间是不切实际的，求解稳定性也较差。当然许多工程问题的雷诺数都很高，直接对湍流域进行求解带来的计算资源消耗很不经济²³。另一种方法是，我们可以使用 Reynolds-Average Simulation (RAS) 湍流模型来求解流体的平均行为以及统计波动。在这个算例演示中(雷诺数为 10e4 的顶盖驱动流)，我们使用附带壁面函数的标准 $k - \epsilon$ 模型来进行求解。这会出现两个新变量：k，湍流动能场； ϵ ，湍流耗散率场。求解这个模型的求解器为 pisoFoam。

2.1.8.1 前处理

现在我们切换到 \$FOAM_RUN/tutorials/incompressible/pisoFoam/ras 下的 cavity 算例下，使用 blockMesh 生成网格。当使用附带壁面函数的标准 $k - \epsilon$ 模型的时，没有必要引入非均匀网格。原因为壁面附近的流型已经被模化而不是直接求解。

OpenFOAM 自带的壁面函数边界条件可以分别用于不同边界面。不同的壁面，可以使用不同的壁面边界条件。我们可以在湍流粘度场 ϑ_t 内进行修改以使用不同壁面函数，其位于 0/nut 文件夹内：

```

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

```

²³ 请参考 DNS 模拟相关内容

在这个算例中，通过把 movingWall 和 fixedWall 指定为 nutWallFunction 类型来使用标准壁面函数模型。其它的壁面边界模型，例如粗糙壁面边界类型，可以通过 nutRoughWallFunction 关键词来指定。

用户现在可以打开 k 和 ε 文件来查看它们的边界类型。对于其中的壁面条件， ε 为 epsilonWallFunction 边界条件， k 为 kqRwallFunction 边界条件。后者是一个普适边界条件，它可以用于任何的湍动能类场，例如 k , q , 雷诺应力 R 场。最初的 k 和 ε 值通过对速度波动量和湍流尺度量进行预估来计算， k 和 ε 的计算公式如下：

$$k = \frac{1}{2} \overline{U' \cdot U'} \quad (2.8)$$

$$\varepsilon = \frac{C_\mu^{0.75} k^{1.5}}{l} \quad (2.9)$$

其中， C_μ 是 $k - \varepsilon$ 模型的标准参数，其值为 0.09。对于笛卡尔坐标体系， k 的计算公式为：

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

其中， U'_x^2 , U'_y^2 , U'_z^2 是速度在 x , y , z 方向的波动速度。我们假定其为各项同性湍流，例如 $U'_x^2 = U'_y^2 = U'_z^2$ ，其值为顶盖速度的 5%。湍流尺度为正方体宽度的 20%，0.1m。这样 k 和 ε 可以这样计算：

$$U'_x = U'_y = U'_z = \frac{5}{100} \text{ m s}^{-1} \quad (2.11)$$

$$\rightarrow k = \frac{3}{2} \left(\frac{5}{100} \right)^2 \text{ m}^2 \text{s}^{-2} = 3.75 \times 10^{-3} \text{ m}^2 \text{s}^{-2} \quad (2.12)$$

$$\varepsilon = \frac{C_\mu^{0.75} k^{1.5}}{l} \approx 7.65 \times 10^{-4} \text{ m}^2 \text{s}^{-2} \quad (2.13)$$

这样我们就可以计算出 k 和 ε 的初始条件。速度和压力的边界条件跟之前的相同，为 $(0, 0, 0)$ 以及 0。

目前有很多的湍流模型，例如 RAS 湍流模型以及 LES 大涡模拟法。它们都植入进了 OpenFOAM。在大多数瞬态求解器中，湍流模型是在运行时进行选择的²⁴。在求解器最开始运行的时候，它会读取 turbulenceProperties 字典文件里面的 simulationType 关键词来获取相关信息。用户可以在 constant 文件夹下面打开这个 turbulenceProperties 字典文件会发现以下信息：

```

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

```

可选的 simulationType 有 laminar, RASModel 以及 LESModel。这个算例中我们选择了 RASModel，更进一步的 RAS 湍流模型在 RASProperties 字典文件中指定，其位于 constant 文件夹中。表 3.9 列举了所有可选的 RAS 湍流模型。kEpsilon 对应的是标准 $k - \varepsilon$ 模型；同时，用户应该确保 turbulence 设置为 on 的状态。

²⁴ 此处涉及到 OpenFOAM 求解器源代码中极为常用的 runTimeMechanism（运行时选择机制）

每个湍流模型的参数都存储在每个湍流模型的代码中并且具有一个默认值。printCoeffs 开关如果设置为 on 的话，这些参数的值会被输出到终端。它们以子字典的形式被输出，我们会看到每个模型的名字后面都加上了 Coeffs，例如 kEpsilon 模型会输出 kEpsilonCoeffs。这些模型的参数也可以通过在 RASProperties 字典文件中自定义来修改。

接下来用户应该设置层流粘度。通过设置动力粘度为 $10e-5$ ，雷诺数就可以达到 $10e4$ ，计算方法参考方程 2.1。

最终，用户应该设置 controlDict 里面的 startTime, stopTime, deltaT 以及 writeInterval。为了保证库郎数小于 1，我们把 deltaT 设置为 0.005 秒。结束时间设置为 10 秒。

2.1.8.2 运行

我们在终端键入 pisoFoam 命令来运行 pisoFoam 程序。在这个算例中，粘度非常低，顶盖附近的边界层会非常薄。又因为我们顶盖附近的网格比较糙。因此在这些网格中的速度要比顶盖的速度小得多。实际上，大约 100 个时间步之后，我们可以看出顶盖附近的网格单元速度的上限大约为 $0.2m/s$ 。因此最大库郎数不会超过 0.2。因此我们可以通过调节时间步来增大库郎数，并保持小于 1 即可。因此我们设置 deltaT 为 0.02。在这种情况下，设置 startFrom 为 latestTime。这表示 pisoFoam 会从最新的时间步开始读取并计算，例如在第 10 秒的时候。endTime 应该被设置为 20s，因为湍流算例比层流算例收敛要慢的多。在设置好之后，重新开始运行，监控残差，并查看结果检测算例是否已经达到稳态，或者算例的某个结果场达到了一种循环的状态。在后一种情况下，可能永远不会收敛，但这并不意味着计算的不精准。

2.1.9 改变算例几何

用户可能打算改变一下几何模型然后重新计算。在这种情况下，最好保留所有的计算过的文件，并在新算例中使用它们。复杂的是，在新的几何模型中，场数据可能不一样了。这样我们可以使用 mapFields 程序来投影场数据，它可以处理完全一样的几何或者不一样的几何，重合边界或者非重合边界。

举例：我们在 icoFoam 目录中切换到 cavityClipped 算例，在这个算例中，几何模型和 cavity 算例是一样的，但是在底部的右边，有个边长为 0.04 米的正方形被移除，下面是 blockMeshDict 字典文件：

```

17 convertToMeters 0.1;
18
19 vertices
20 (
21     (0 0 0)
22     (0.6 0 0)
23     (0 0.4 0)
24     (0.6 0.4 0)
25     (1 0.4 0)
26     (0 1 0)
27     (0.6 1 0)
28     (1 1 0)
29
30     (0 0 0.1)
31     (0.6 0 0.1)
32     (0 0.4 0.1)
33     (0.6 0.4 0.1)
34     (1 0.4 0.1)
35     (0 1 0.1)
36     (0.6 1 0.1)
37     (1 1 0.1)

```

```

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

```

我们用 `blockMesh` 来生成网格，`patches` 参考之前算例的设定。为了在 `mapFields` 的过程中便于区分。我们在这个算例中，把原始 cavity 算例中的 `movingWall` 命名为 `lid`。

在非连续几何的投影中，不能保证所有的场数据都能投影过来。在某些区域，原始场并不包含相关数据，这些区域的数据还是被投影场的原始数据。因此在投影之前，在相应的时间步内就应该存在相应的场数据。在这个算例中，我们从 0.5 秒开始进行投影，因此我们在 `controlDict` 中设定 `startTime` 为 0.5s。因此，用户需要把 0 秒的场数据拷贝到 0.5s 中。我们使用以下命令：

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

```
cp -r 0 0.5
```

在进行投影之前，用户可以查看 0.5s 的场数据。

现在我们打算把 cavity 的速度和压力场数据投影到 cavityClipped 算例中。因为投影是不连续的，因此我们需要编辑 system 文件夹下的 mapFieldsDict 文件。这个文件包含两个关键词，一个是 patchMap，另一个是 cuttingPatches。当想把原始场的 patches 条件投影到被投影场的 patches 的时候，我们使用 patchMap。在 cavityClipped 算例中，我们的 lib 边界条件跟 cavity 算例的 movingWall 边界条件是一致的，这样我们这样来设置 patchMap：

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

当用户打算把原始场的内场数据投影到被投影场的边界的时候，我们使用 cuttingPatches 列表。在这个算例中，我们想通过算例来演示这个过程，因此我们在 cuttingPatches 里面指定 fixedWalls：

```
cuttingPatches
(
    fixedWalls
);
```

现在用户可以在 cavityClipped 目录下执行 mapFields 命令：

```
mapFields ..//cavity
```

参见图 2.13，这即为投影后的场数据。边界面的场信息确实由原始场的内场数据获得。然而，我们打算把 fixedWalls 的边界条件重置为(0 0 0)，因此我们需要编辑 U 文件，把 fixed -Walls 的边界条件从 nonuniform 修改为 uniform (0 0 0)。在 nonuniform 之后括号内的相关内容也应该删去。现在使用 icoFoam 重新运行这个算例。

2.1.10 后处理

速度场可以按照前文所述的方法来进行后处理，我们可以对比 0.5s 的初始场和 0.6s 的最终场。另外，我们简单说一下对于 2D 算例的特殊处理以使其更加漂亮。用户可以在 Filter 菜单中选择 Extract Block 滤镜，然后在 Parameter 面板中，高亮感兴趣的边界面例如 lid 和 fixedWalls。点击 Apply，然后在 Display 面板中选择 Wireframe，这些几何就会显示出来。参见图 2.14。其中的各个面显示为黑色，我们也可以清楚地看出在这个几何模型下底部存在漩涡。

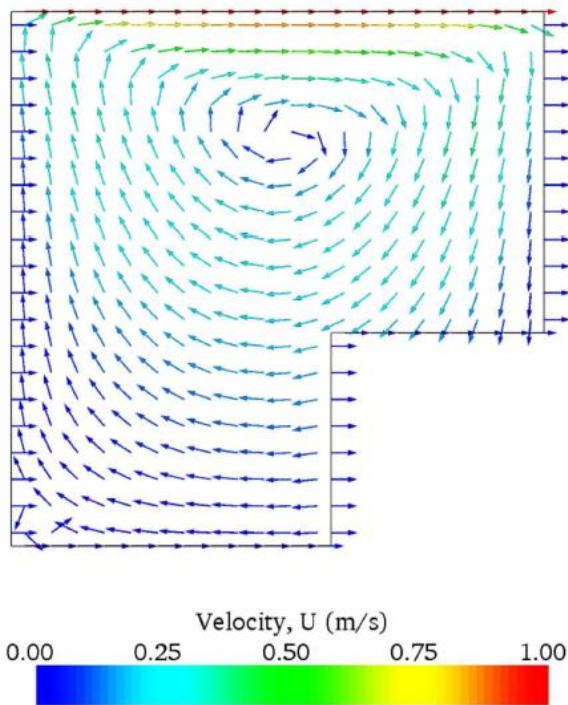


图 2.13 将 cavity 算例的结果投影到 cavityClipped

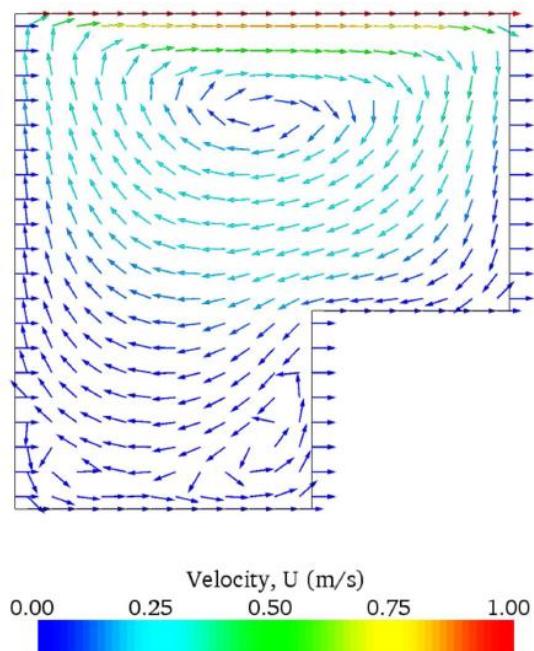


图 2.14 cavityClipped 的速度矢量图

2.2 带孔盘体应力分析

这个算例我们讲解在一个中心带有圆形空洞的方盘上，如何进行前处理、线弹性稳态应力分析以及如何进行后处理。盘子的几何尺寸为：边长 4 米，中心的空洞半径 R 为 0.5 米。这个方盘左右面分别承受了一个均匀的 10kPa 的载荷，如图 2.15 所示。这个几何体上可以看出有两个对称平面。因此求解区域只需要覆盖整个几何体的四分之一，即 2.15 图中的阴影部分面积。

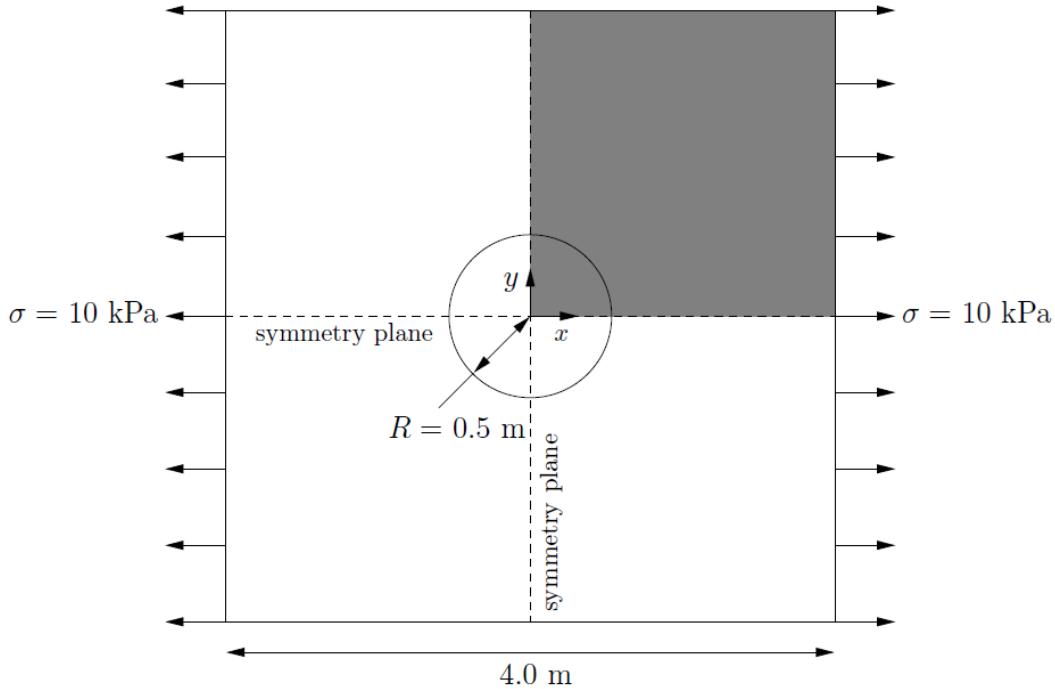


图 2.15 二维空心盘体几何

由于负载被附加于盘体的一个平面上，因此这个问题可以看做是 2 维的。在笛卡尔坐标系下面，关于第三个维度的行为存在两种假定：(1) 应力平面条件：适用于在这个 2D 平面外的第三维度压力是可忽略的情况；(2) 应变平面条件：适用于在这个 2D 平面外的第三维度的应变是可忽略的情况。应力平面条件对于那些第三维度方向非常薄的算例是合适的。应力平面条件对于第三维方向比较厚的情况下适用。

对于一个无限大的中心带有圆形空洞的薄盘，它存在解析解。垂直于应力对称平面的解析解为：

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

在这个算例中，模拟的结果将和解析解相对比。在这个教程结束后，读者可以对网格相关性进行分析，或者增加盘子大小获得数值解来和无限平板的解析解相对比并比较误差。

2.2.1 网格生成

整个区域分为 4 个 block，其中一些具有曲边。xy 平面的 Block 结构如图 2.16 所示。正如 2.1.1.1 节所说，即使是一个 2D 算例，所有的几何在 OpenFOAM 里面都是 3 维的形式。因此 z 方向的厚度应该指定，例如 0.5 米。这个数值的大小并不影响求解。因为牵引力边界条件以单位面积应力来指定，因此求解结果横截面的大小无关。

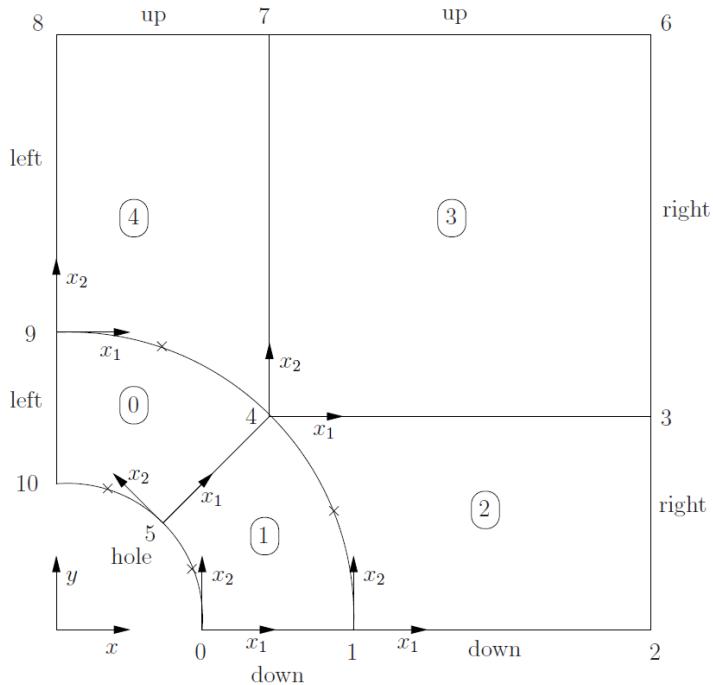


图 2.16 中心空洞平板的 block 结构

用户从 tutorials/stressAnalysis/solidDisplacementFoam 目录中进入 plateHole 算例，并打开 constant/polyMesh/blockMeshDict 文件，如下所示：

```

17 convertToMeters 1;
18
19 vertices
20 (
21   (0.5 0 0)
22   (1 0 0)
23   (2 0 0)
24   (2 0.707107 0)
25   (0.707107 0.707107 0)
26   (0.353553 0.353553 0)
27   (2 2 0)
28   (0.707107 2 0)
29   (0 2 0)
30   (0 1 0)
31   (0 0.5 0)
32   (0.5 0 0.5)
33   (1 0 0.5)
34   (2 0 0.5)
35   (2 0.707107 0.5)
36   (0.707107 0.707107 0.5)
37   (0.353553 0.353553 0.5)
38   (2 2 0.5)
39   (0.707107 2 0.5)
40   (0 2 0.5)

```

```

41   (0 1 0.5)
42   (0 0.5 0.5)
43 );
44
45 blocks
46 (
47   hex (5 4 9 10 16 15 20 21) (10 10 1) simpleGrading (1 1 1)
48   hex (0 1 4 5 11 12 15 16) (10 10 1) simpleGrading (1 1 1)
49   hex (1 2 3 4 12 13 14 15) (20 10 1) simpleGrading (1 1 1)
50   hex (4 3 6 7 15 14 17 18) (20 20 1) simpleGrading (1 1 1)
51   hex (9 4 7 8 20 15 18 19) (10 20 1) simpleGrading (1 1 1)
52 );
53
54 edges
55 (
56   arc 0 5 (0.469846 0.17101 0)
57   arc 5 10 (0.17101 0.469846 0)
58   arc 1 4 (0.939693 0.34202 0)
59   arc 4 9 (0.34202 0.939693 0)
60   arc 11 16 (0.469846 0.17101 0.5)
61   arc 16 21 (0.17101 0.469846 0.5)
62   arc 12 15 (0.939693 0.34202 0.5)
63   arc 15 20 (0.34202 0.939693 0.5)
64 );
65
66 boundary
67 (
68   left
69   {
70     type symmetryPlane;
71     faces
72     (
73       (8 9 20 19)
74       (9 10 21 20)
75     );
76   }
77   right
78   {
79     type patch;
80     faces
81     (
82       (2 3 14 13)
83       (3 6 17 14)
84     );
85   }
86   down
87   {
88     type symmetryPlane;
89     faces
90     (
91       (0 1 12 11)
92       (1 2 13 12)
93     );
94   }
95   up
96   {
97     type patch;
98     faces
99     (
100      (7 8 19 18)
101      (6 7 18 17)
102    );
103  }
104  hole
105  {
106    type patch;
107    faces
108    (
109      (10 5 16 21)
110      (5 0 11 16)
111    );
112  }
113  frontAndBack
114  {
115    type empty;
116    faces
117    (

```

```

118      (10 9 4 5)
119      (5 4 1 0)
120      (1 4 3 2)
121      (4 7 6 3)
122      (4 9 8 7)
123      (21 16 15 20)
124      (16 11 12 15)
125      (12 13 14 15)
126      (15 14 17 18)
127      (15 18 19 20)
128  );
129 }
130 );
131
132 mergePatchPairs
133 (
134 );
135
136 // ****

```

迄今为止我们在之前的教程中，只制作了直边 block，现在我们需要制定曲边 block。这些通过修改关键词 edges 下面的相关信息来完成，关键词 edges 列举了非直线的 block 边。Edge 下面的语句要确保每行信息以 arc, simpleSpline, polyLine 等开头，详述请参阅 5.3.1 节。在这个例子中，所有的曲边都是圆形的²⁵，因此可以用 arc 关键词来指定。后面的信息是弧边的起始点和终止点的标示，以及这个弧边通过的点的坐标。

blockmeshDict 里面的 block 具有方向且各不相同。正如 2.16 所示，0block 的 x2 方向就是 4block 的-x1 方向。这意味着当定义每个 block 内的网格数和节点分布的时候需要多加小心，以使它们互相匹配。

6 个 patch 如下定义：平板的每个边一个 patch，空洞一个 patch，前后面一个 patch。左边和下面的 patch 是对称面。这个对称面是一个几何方面的对称面。因此它需要在网格里面进行定义而不是简单的定义一个初始边界条件。因此它们在 blockmeshDict 里面使用 symmetryPlane 关键词来定义。

frontAndBack，这个 patch 代表了这是一个 2D 的算例，在计算中这个 patch 将被忽略。再次提醒一下，这是一个几何的限制，因此需要在网格内来进行定义。它使用 empty 在 blockMeshDict 内进行定义。关于其它更多的边界条件以及几何限定信息，请查阅 5.2.1 节。

剩余的就是常规 patch 设定，网格应该用 blockMesh 程序来生成，可以用 paraFoam 来查看，详述请查阅 2.1.2 节。显示如 2.17 所示：

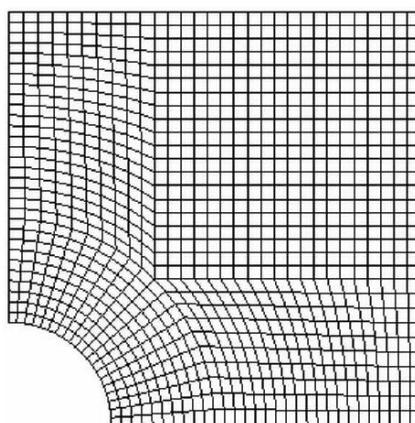


图 2.17 中心空洞算例的网格

²⁵ 即半圆，或者圆的一部分。而非椭圆之类

2.2.1.1 边界和初始条件

一旦网格生成完全，边界条件的初始场必须进行指定。对于无热应力的单纯应力分析，只有位移量 \mathbf{D} 需要设定。在 O/D 文件下面我们有：

```

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

```

首先，可以看出位移量初始条件设定为(0 0 0)米。left 以及 down 这两个 patch 必须设定为 symmetryPlane，因为它们在网格中就被设定了，这可以打开 constant/polyMesh/boundary 文件加以确认。frontAndBack 设定为 empty。

其它 patches 是牵引力边界条件，牵引力边界条件通过 traction 字符来指定。它由以下两个的关键词组合来构成：(1) 在关键词 traction 下面指定牵引力边界矢量和大小。(2) 在 pressure 关键词下，如果这个边界面法向牵引力的压力指向表面之外，就被定义为负值，因为它的 up 和 hole 这两个面牵引力为 0，因此边界牵引力和压力设定为 0。对于 right 边界牵引力应该为(1e4 0 0)Pa，pressure 应该为 0 Pa。

2.2.1.2 物理特性

这个算例的物理特性放置在 constant 文件夹下的 mechanicalProperties 文件夹下面。我们需要制定钢材料的物理特性，见表 2.1。在物理特性字典中，用户需要指定 planeStress 为 yes。

物性	单位	关键词	值
密度	kg/m ³	rho	7854
杨氏模量	Pa	E	2×10^{11}
泊松比	—	nu	0.3

表 2.1 钢的机械特性

2.2.1.3 热物理特性

在 solidDisplacementFoam 求解器中我们可以计算温度场，因此用户可以选择求解这个和动量方程耦合（其通过生成的热应力来耦合）在一起的热物理方程。在运行之前，用户可以通过在 thermalProperties 字典下的 thermalStress 开关来指定是否求解热物理方程。这个字典同样设置了这个算例的热物理特性，例如钢的热物理特性，见表 2.2。

物性	单位	关键词	值
热容	/JkgK	C	434
热导	/WmK	k	60.5
热膨胀系数	/K	alpha	1.1×10^{-5}

表 2.2 钢的热物理特性

在这个例子中我们不想求解热物理方程，因此我们必须在 thermalProperties 字典中设定 thermalStress 关键词为 no。

2.2.1.4 控制

正如之前所说，求解的控制信息在 controlDict 字典中。对于这个算例，开始时间是 0 秒。由于这是一个稳态问题因此时间步无关紧要²⁶；在这个情况下，最好设定时间步长 deltaT 为 1，在稳态问题中，它表示迭代步。终止时间 endTime 设定为 100，即最高迭代数。写入控制 writeInterval 设定为 20。controlDict 的信息如下：

```

17
18 application      solidDisplacementFoam;
19
20 startFrom       startTime;
21
22 startTime        0;
23
24 stopAt          endTime;
25
26 endTime          100;
27
28 deltaT          1;
29

```

²⁶ 在 OpenFOAM 中，如果求解器采用的是 SIMPLE 算法，那么 deltaT 设定为多少是无关紧要的。如果其设定为 1，那么运行 100 秒的时候即为迭代 100 次，如果设定为 0.1，运行 100s 的时候即为迭代 1000 次。为了简化，我们在使用 SIMPLE 求解器的 control 字典中，通常设为 1

```

30 writeControl      timeStep;
31
32 writeInterval    20;
33
34 purgeWrite       0;
35
36 writeFormat      ascii;
37
38 writePrecision   6;
39
40 writeCompression off;
41
42 timeFormat       general;

```

2.2.1.5 离散格式和求解器控制

现在，我们来分析一下 fvSchemes 字典文件。首先，我们分析的是一个稳态问题，因此用户应该在 timeScheme 的时间离散里面选择 SteadyState。这就屏蔽掉了时间离散项。在流体力学中，并不是所有的求解器，可以即用稳态又用瞬态来进行计算。但是 solidDisplacementFoam 可以即算稳态又计算瞬态，因为对于这个求解器，两种类型的基本算法大体相同。

线弹性应力分析的动量方程包含几个含有位移梯度量的显性项。梯度项的精准计算和光滑性对结果有很大影响。一般情况下，有限体积离散建立于高斯定律之上，高斯定律对于大部分的模拟目的来说是足够精准的。但在这个算例中，我们使用 least squares 方法²⁷。因此，用户应该打开 system 文件夹下的 fvSchemes 字典，确保 gradSchemes 字典下 grad (U)，grad (T) 采用 leastSquares 方法：

```

17
18 d2dt2Schemes
19 {
20     default      steadyState;
21 }
22
23 ddtSchemes
24 {
25     default Euler;
26 }
27
28 gradSchemes
29 {
30     default      leastSquares;
31     grad(D)     leastSquares;
32     grad(T)     leastSquares;
33 }
34
35 divSchemes
36 {
37     default none;
38     div(sigmaD) Gauss linear;
39 }
40
41 laplacianSchemes
42 {
43     default none;
44     laplacian(DD,D) Gauss linear corrected;
45     laplacian(DT,T) Gauss linear corrected;
46 }
47
48 interpolationSchemes
49 {
50     default linear;
51 }
52
53 snGradSchemes
54 {

```

²⁷ cfدونline 有人测试 least squares 对网格较差的算例比较好，译者并未考证

```

55     default      none;
56 }
57
58 fluxRequired
59 {
60     default      no;
61     D            yes;
62     T            no;
63 }
64
65
66 // ****

```

system 文件下面的 fvSolution 字典文件控制求解线性方程组使用的矩阵求解器。用户应该首先查看 solvers 子字典，会留意到 D 的矩阵求解器为 GAMG。求解器的 tolerance 是 10^{-6} 。矩阵求解器的相对误差由 relTol 控制，控制每次迭代的残差减小量。本例中设置一个很小的残差是不经济的，因为方程组中很多项是显性的，并且采用分离迭代求解技术。因此，合理的迭代残差为 0.01，更高也是可以的，比如 0.1，在某些例子中，0.9 也是可以的（比如这个例子）：

```

17
18 solvers
19 {
20     "(D|T)"
21     {
22         solver          GAMG;
23         tolerance       1e-06;
24         relTol          0.9;
25         smoother        GaussSeidel;
26         cacheAgglomeration true;
27         nCellsInCoarsestLevel 20;
28         agglomerator    faceAreaPair;
29         mergeLevels     1;
30     }
31 }
32
33 stressAnalysis
34 {
35     compactNormalStress yes;
36     nCorrectors        1;
37     D                  1e-06;
38 }
39
40
41 // ****

```

fvSolution 字典下面包含一个子字典：stressAnalysis。它包含了这个求解器所需的控制参数。首先，nCorrectors 表示整个方程组求解的外循环数，包括每个时间步的拉伸边界条件。由于这是一个稳态问题，我们用时间步代表迭代数以直到收敛。因此我们可以设置 nCorrectors 为 1。

关键词 D 指定外循环的收敛残差，例如：设置一个残差，当达到这个值的时候迭代停止。这个残差，本例为 10^{-6} ，应该在运行之前来设定。

2.2.2 运行

用户应该在后台运行程序，这样就可以在 log 文件中查看收敛信息。运行方式如下：

```

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

```

用户应该在 log 文件中查看收敛信息，它包括了迭代数、每步迭代的初始残差和最终残差。依靠这个残差设定，最终残差应该总是小于最初残差的 0.9 倍。一旦初始残差小于收敛残差 10^{-6} 的时候，程序收敛，用户可以终结程序。

2.2.3 后处理

后处理可参考 2.1.4 节的方法来进行，OpenFOAM 经常采用数学符号的名字来表示变量，solidDisplacementFoam 求解器以对称张量场 σ 的形式输出应力场 σ ，这和 OpenFOAM 的通常做法是一致的。对于希腊字母，通常按照发音来命名。

如果处理张量的某个分量（张量的分量为一个标量场）。例如： σ_{xx} 、 σ_{xy} 等。它可以由 foamCalc 程序来计算，如 2.1.5.7 节所说，在这个算例中我们采用这个命令：

```
foamCalc components sigma
```

张量分量被命名为 sigmaxx 、 sigmaxy ，并且写在了每个时间步内。下图为 σ_{xx} 在 paraFoam 中的显示，见图 2.18：

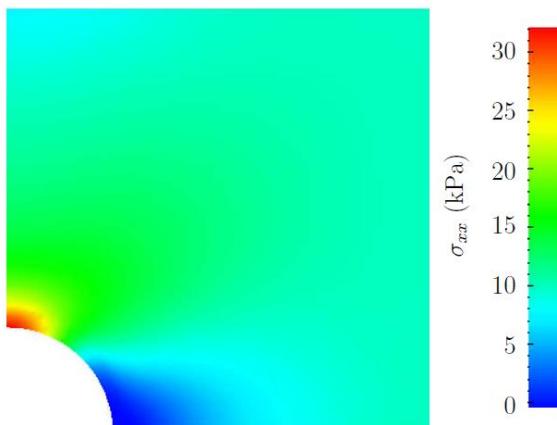


图 2.18 中心空洞圆盘的 σ_{xx} 场

我们想把方程 2.14 的解析解和我们的数值解相对比。因此我们必须把计算域中对称面的左边的 σ_{xx} 数据输出出来。用户可以用 sample 程序来生成需要的数据。这个程序需要调用 system 文件夹下面的 sampleDict 字典，如表 6.3 所示。其中的 sets 关键词指定的 sample line 为 (0.0, 0.5, 0.25) 到 (0.0, 2.0, 0.15) 之间的线段，提取的场在 fields 关键词内指定：

```

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

```

```

34 fields      ( sigmaxx );
35
36
37 // ****

```

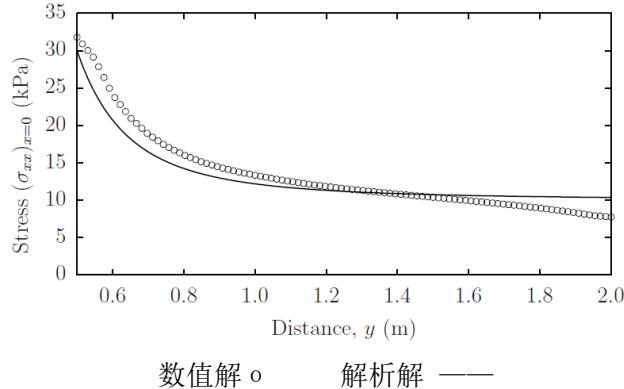


图 2.19 $x=0$ 处的垂直平面法向应力分布

用户应该执行 `sample` 命令。写入格式为 `raw`。其为 2 列的原始数据。数据随后被写入 `postProcessing/sets` 下的每个时间步文件中。例如：100s 的场数据存储在 `sets/100/leftPatch_sigmaxx.xy` 中。依靠 GnuPlot²⁸，可以使用如下命令来把精确解和数值解画在同一个图中：

```

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

```

图表请参阅 2.19。

2.2.4 练习

用户可以通过下面的练习来熟悉 `solidDisplacementFoam`：

2.2.4.1 增加网格数量

在 x 、 y 方向增加网格数量。参考 2.2.3 节，使用 `mapFields` 来把粗网格结果投影到细网格上并作为初始条件。

2.2.4.2 引入网格非均匀化

如果我们使用非均匀网格，会使靠近空洞的网格比远处网格更加细密。我们来重新设计网格，使得相邻网格的大小比小于 1.1。这使得相邻块网格之间的网格比例很相近。网格非均匀化在 2.1.6 节有提及。我们还可以使用 `mapFields` 把 2.2.3 节使用的粗网格的结果投影到非均匀化网格的初始场上。对比这些结果，或者和分析解相对比。用非均匀化网格方法计算的结果会比之前的结果好么？

²⁸ 一个软件，需要下载安装后，键入 `gnuplot` 来进入界面

2.2.4.3 改变平板尺寸

解析解是针对一个无限大的带有空洞的二维平板，因此这个对于有限大小的平板来讲不是特别精准。为了估算误差，可以在保持空洞大小不变的情况下增加原盘大小来试试。

2.3 溃坝

在本教程中我们将用 `interFoam` 求解一个 2 维简化的溃坝问题。该问题的特征，两种液体被一种尖锐的界面（或自由表面）分隔的瞬态流动。`interFoam` 中的两相算法基于流体体积分数（VOF）法，在该方法中，每个网格中的相体积分数（或相分数 alpha）通过求解一个组分输运方程确定。物理属性则基于这个相分数通过加权平均计算。VOF 方法的本质意味着组分间的界面不是计算出来的，而是作为相分数场的一个属性表现出来²⁹。由于相分数可以为 0 和 1 之间的任何值，所以相界面并没有被严格定义，因此其为尖锐界面本应存在的那些网格单元。

算例的物理过程如下：设置为一个静止的水柱位于水箱左侧，在水箱的底部有一个小的障碍，在 $t=0$ s 时刻，让水柱自由流动，产生水柱崩塌。在崩塌的过程中，水撞击水箱底部的一个障碍形成复杂的流场结构，其中包括若干被水包裹的空气微团。几何和初始设置如图 2.20 所示：

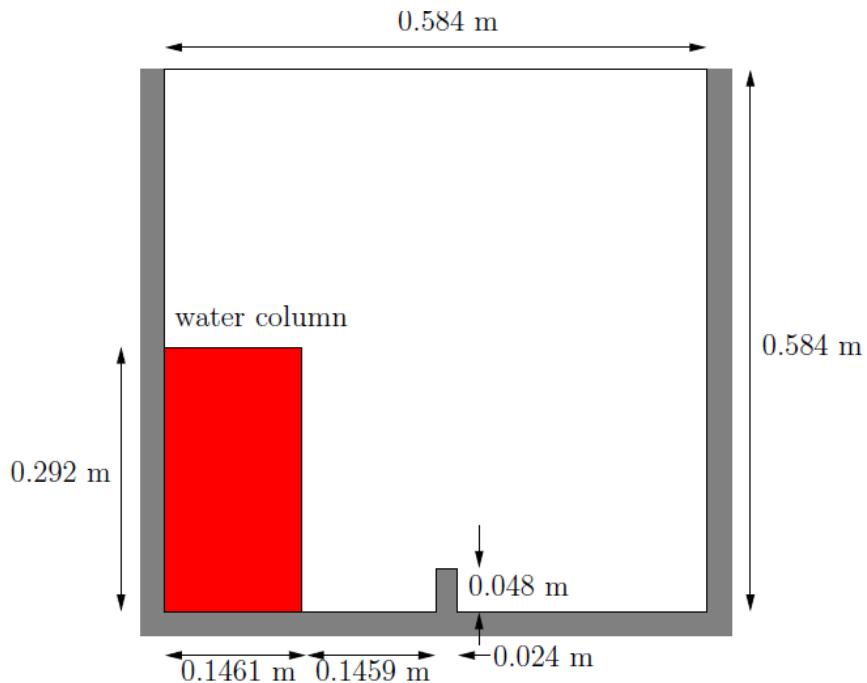


图 2.20 溃坝的几何场

2.3.1 生成网格

用户应从 \$FOAM_RUN/tutorials/multiphase/interFoam/laminar 文件夹中进入 `damBreak` 算例。

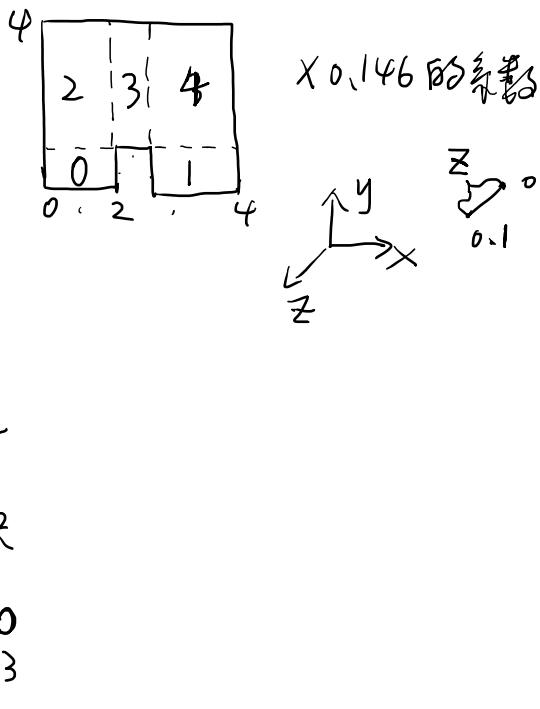
²⁹ VOF 的界面是由后处理得出来的结果，详细的 VOF 模型请参考其它文档

然后根据前面所描述的运行 blockMesh 命令生成网格。damBreak 的网格包含 5 个 block，其 blockMeshDict 的相关信息如下：

```

17 convertToMeters 0.146;
18
19 vertices
20 (
21   (0 0 0)
22   (2 0 0)
23   (2.16438 0 0)
24   (4 0 0)
25   (0 0.32876 0)
26   (2 0.32876 0)
27   (2.16438 0.32876 0)
28   (4 0.32876 0)
29   (0 4 0)
30   (2 4 0)
31   (2.16438 4 0)
32   (4 4 0)
33   (0 0 0.1)
34   (2 0 0.1)
35   (2.16438 0 0.1)
36   (4 0 0.1)
37   (0 0.32876 0.1)
38   (2 0.32876 0.1)
39   (2.16438 0.32876 0.1)
40   (4 0.32876 0.1)
41   (0 4 0.1)
42   (2 4 0.1)
43   (2.16438 4 0.1)
44   (4 4 0.1)
45 );
46
47 blocks
48 (
49   hex (0 1 5 4 12 13 17 16) (23 8 1) simpleGrading (1 1 1) 0
50   hex (2 3 7 6 14 15 19 18) (19 8 1) simpleGrading (1 1 1) 1
51   hex (4 5 9 8 16 17 21 20) (23 42 1) simpleGrading (1 1 1) 2
52   hex (5 6 10 9 17 18 22 21) (4 42 1) simpleGrading (1 1 1) 3
53   hex (6 7 11 10 18 19 23 22) (19 42 1) simpleGrading (1 1 1) 4
54 );
55
56 edges
57 (
58 );
59
60 boundary
61 (
62   leftWall
63   {
64     type wall;
65     faces
66     (
67       (0 12 16 4) 0
68       (4 16 20 8) 2
69     );
70   }
71   rightWall
72   {
73     type wall;
74     faces
75     (
76       (7 19 15 3) 1
77       (11 23 19 7) 4
78     );
79   }
80   lowerWall
81   {
82     type wall;
83     faces
84     (
85       (0 1 13 12) 0
86       (1 5 17 13) 3
87       (5 6 18 17) 1
88       (2 14 18 6) 1

```



```

89           (2 3 15 14)  ]
90   );
91 }
92 atmosphere
93 {
94   type patch;
95   faces
96   (
97     (8 20 21 9)
98     (9 21 22 10)
99     (10 22 23 11)
100  );
101 }
102 );
103
104 mergePatchPairs
105 (
106 );
107
108 // ****

```

2.3.2 边界条件

用户可以通过查看 constant/polyMesh 文件夹下的 boundary 文件来检查 blockMesh 命令生成的几何边界。该文件包含一个由 5 个 patch 构成的列表： leftWall, rightWall, lowerWall, atmosphere 和 defaultFaces。用户应该注意各个 patch 的 type 关键字。其中 atmosphere 是一个标准的 patch，也就是没有特别的属性，只不过是一个可以在其上面定义边界条件的实体。由于 defaultFaces 的法相垂直于我们的求解平面，我们对其并不求解，因此 defaultFaces 的类型是 empty。而 leftWall, rightWall 和 lowerWall 各自的类型都是 wall，像简单的 patch 一样，wall 类型不包含关于网格的任何几何或拓扑信息，它和简单 patch 唯一的不同之处在于它是一个壁面。这意味着可以在其上应用壁面函数模型。

庆幸的是，interFoam 求解器植入了对位于界面和壁面接触点处表面张力的模拟。该模型通过设置 alpha(α)场为 alphaContactAngle 边界条件来实现。如此，用户必须指定如下信息：静态接触角 θ_0 ，前缘和后缘动态接触角 θ_A 和 θ_R 和一个用于动态接触角的速度尺度函数 $u\theta$ 。

alpha
文件设置 在本算例中我们将忽略界面和壁面间的表面张力效应。这可以通过设定静态接触角 $\theta = 90$ 度，以及速度尺度函数为 0 来实现。不过在这里我们选择更简单的方法，即设置 alpha 的边界条件为 zeroGradient，而不是用 alphaContactAngle 边界条件。

本算例的上边界与大气环境自由相通，因此应根据内部流动允许出流和入流。于是我们用速度和压力边界条件的组合来实现这个目的，并同时保证稳定性。它们是：

U
文件设置

- totalPressure: 一种 fixedValue 条件，利用指定的总压 p_0 和当地速度 U 计算获得；
- pressureInletOutletVelocity: 对所有分量应用 zeroGradient 条件，当流动为入流时，对边界切向的分量应用 fixedValue 条件；
- inletOutlet: 出流时为 zeroGradient 条件，入流时则为 fixedValue 条件。

在所有壁面边界处，压力场采用 fixedFluxPressure 边界条件，它自动调整压力梯度使边界通量符合速度边界条件。

P
文件

defaultFaces patch 代表此 2 维问题的前后面，像往常一样，它们是 empty 类型。

2.3.3 设置初始场

与以往的例子不同，我们现在应为相分数 α_{water} 指定一个非均匀的初始条件，其中

$$\alpha_{water} = \begin{cases} 1 & \text{水相} \\ 0 & \text{空气相} \end{cases} \quad (2.15)$$

这可以通过运行 setFields 工具实现。它需要一个 setFieldsDict 字典，其位于 system 文件夹，本算例的相关信息如下：

```

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

```

其中 defaultFieldValues 关键字用来设置场的默认值，也就是说这个值将在 regions 子字典中指定的区域以外采用。这个 regions 子字典包含一系列子字典，它包括要在指定区域设定某个场的关键词 fieldValues。我们用一些点、单元或面的集合，通过拓扑约束来建立一个 topoSetSource 区域。在这里，boxToCell 通过定义一个最小和最大的向量来创建一个盒子区域，在此区域内为水相。该区域内相分数 α_{water} 被设置为 1。

setFields 工具从文件读取场并重新定义这些场，然后把它们重新写入文件。原始的文件将会被覆盖，因此我们建议在执行 setFields 前先进行备份。在 damBreak 例子中，场 alpha.water 最初通过存为一个名为 alpha.water.org 的文件来备份。在执行 setFields 前，用户首先应复制 alpha.water.org 为 alpha.water，比如通过执行以下命令：

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

用户接下来应在执行其它任何工具之前执行 setFields。利用 paraFoam，检查 alpha.water 的初始场是否符合图 2.21 所示的分布。

2.3.4 流体特性

让我们看一下 constant 文件夹下的 transportProperties 文件。该字典包含流体的物质属性，分别列在 water 和 air 两个子字典里。各相的传递模型通过关键字 transportModel 进行选择。用户应选择 Newtonian，此时运动粘度为常数且通过关键字 nu 指定。其它模型如 CrossPowerLaw 的粘性参数在通用名为<model>Coeffs 的子字典内指定，如果采用 CrossPowerLaw 模型那么这个粘性参数关键词即为 CrossPowerLawCoeffs。密度在关键字 rho 下指定。

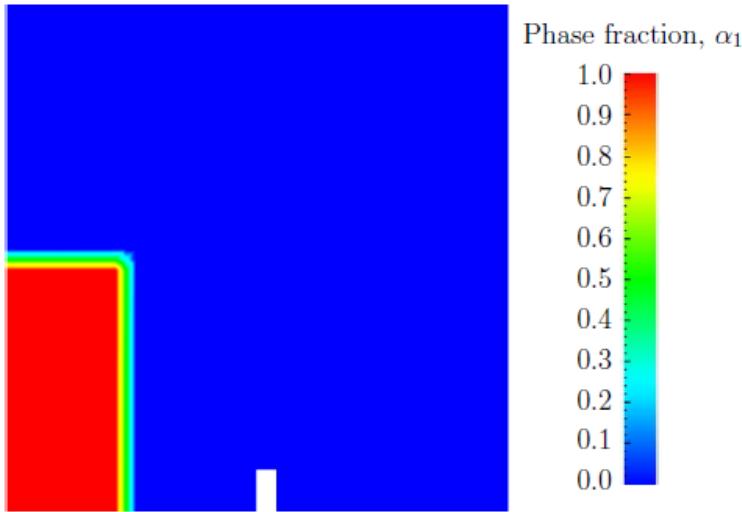


图 2.21 相分数 α_{water} 的初始场

两相间的表面张力通过关键字 `sigma` 指定。本例中使用的参数列于表 2.3 中：

水的物性			
运动粘度	m^2/s	<code>nu</code>	1.0×10^{-6}
密度	kg/m^3	<code>rho</code>	1.0×10^3
空气的物性			
运动粘度	m^2/s	<code>nu</code>	1.48×10^{-5}
密度	kg/m^3	<code>rho</code>	1.0
两相特性			
表面张力	N/m	<code>sigma</code>	0.07

表 2.3 漏坝的流体特性

重力加速度在整个求解域内统一分布，通过 `constant` 文件夹内名为 `g` 的文件来指定。与普通的场文件(如 `U` 和 `p`)不一样，`g` 是 `uniformDimensionedVectorField` (均一的带有量纲的矢量场) 类型，因此只是简单的包含一个 `dimensions` 关键字和一个 `value` 关键字，如下所示本例中为 $(0, 9.81, 0)\text{m/s}^2$ ：

```

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

```

2.3.5 湍流模型

如 `cavity` 算例一样，湍流模型的选择是在运行时读取字典 `turbulenceProperties` 里的 `simulationType` 关键字来进行。在本例中，我们希望运行时不带求解湍流模型，于是我们设置为 `laminar`：

```

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

```

2.3.6 时间步长控制

时间步长控制在自由表面追踪问题中是一个重要的问题，因为表面追踪算法比标准流体算法对库朗数更为敏感。理想状态下，在界面区域库朗数不应超过上限如 0.5。在某些速度很容易预测的算例中，用户应指定一个满足 Co 条件的固定时间步长。对于更为复杂的算例，这却相当困难。因此 interFoam 求解器在 controlDict 中提供了自动调整时间步长。用户需指定 adjustTimeStep 为 on，并设置最大库朗数 maxCo 为和最大相场的库朗数 maxAlphaCo 为 1.0。时间步长的上限 maxDeltaT 可以设置为一个本次模拟中不会超过的值，比如 1.0。

通过使用时间步长自动控制，时间步本身就是一个不确定的时间步，其具有随机性。因此，如果我们要求 OpenFOAM 按照一个固定的时间步间隔保存结果，会发现保存结果的时间步会比较乱。值得一提的是，即使采用自动调整时间步，OpenFOAM 也允许用户在指定的时间步来保存数据，在这种情况下，OpenFOAM 在那一时间点强制调整时间步长，以使运算的时间正好“碰上”所指定的那些输出结果的时间。用户可以通过指定 controlDict 字典里 writeControl 关键字为 adjustableRunTime 来实现。controlDict 字典的相关信息应为：

```

17
18 application      interFoam;
19
20 startFrom       startTime;
21
22 startTime        0;
23
24 stopAt          endTime;
25
26 endTime         1;
27
28 deltaT          0.001;
29
30 writeControl    adjustableRunTime;
31
32 writeInterval   0.05;
33
34 purgeWrite      0;
35
36 writeFormat     ascii;
37
38 writePrecision  6;
39
40 writeCompression uncompressed;
41
42 timeFormat      general;
43
44 timePrecision   6;
45
46 runTimeModifiable yes;
47
48 adjustTimeStep   yes;
49
50 maxCo           1;
51 maxAlphaCo      1;
52
53 maxDeltaT       1;
54
55 // ****

```

2.3.7 离散格式

interFoam 求解器在求解相场的时候采用了 OpenCFD 公司开发的 MULES 方法，这个方法可以保证在任何数值格式，网格类型的情况下相场有界。因此在使用 interFoam 求解器的算例中，对流项格式的选择不再局限于那些稳定且有界的格式，如迎风格式。

对流项格式设置在 fvSchemes 字典文件的子字典 divSchemes 中实现。本例中，动量方程的对流项为 $\nabla \cdot (\rho U U)$ ，在字典中体现为关键字 div(rho*phi,U)，我们采用 Gauss linearUpwind grad(U) 以实现良好的精度。正如 4.4.1 节中描述的一样，有界线性格式需要一个系数 ϕ 。这里，我们选择 $\phi = 1.0$ 以实现最好的稳定性。关键字 div(phi,alpha) 代表的 $\nabla \cdot (U \alpha_1)$ 项采用 vanLeer 格式。因为有界性已由 MULES 算法保证，因此关键字 div(phirb,alpha) 代表的 $\nabla \cdot (U_{rb} \alpha_1)$ 项可采用二阶精度的 linear (中心) 格式。

其它离散项采用常用的格式，因此 fvSchemes 字典的相关信息应为：

```

17
18 ddtSchemes
19 {
20     default      Euler;
21 }
22
23 gradSchemes
24 {
25     default      Gauss linear;
26 }
27
28 divSchemes
29 {
30     div(rhoPhi,U)           Gauss linearUpwind grad(U);
31     div(phi,alpha)          Gauss vanLeer;
32     div(phirb,alpha)        Gauss linear;
33     div((muEff*dev(T(grad(U))))) Gauss linear;
34 }
35
36 laplacianSchemes
37 {
38     default      Gauss linear corrected;
39 }
40
41 interpolationSchemes
42 {
43     default      linear;
44 }
45
46 snGradSchemes
47 {
48     default      corrected;
49 }
50
51 fluxRequired
52 {
53     default      no;
54     p_rgh;
55     pcorr;
56     alpha.water;
57 }
58
59
60 // ****

```

2.3.8 矩阵求解器控制

在 fvSolution 中，PISO 子字典包含 interFoam 所特有的一些参数。其中包括常规的动量方程

的修正步数，也有 PISO 循环关于 α 相方程的修正步数。其中最让人感兴趣的是关键字 nAlphaSubCycles 和 cAlpha。nAlphaSubCycles 代表 α 方程中子循环的数目；子循环是在一个给定时间步内对一个方程的附加求解³⁰。它可以在不降低时间步长的情况下，保证解的稳定，并使得可求解的时间变的更长。在这里我们指定为 2，这意味着 α 方程在各个实际的时间步内以半个实际时间步长的步长求解了两次。

关键字 cAlpha 是一个控制界面压缩的因子，其中 0 相当于无压缩，1 相当于守恒压缩；并且，任何大于 1 的数表示强化界面压缩。我们通常建议取 1，就如本例一样。

2.3.9 运行程序

运行程序已经在前面的教程里详细介绍过。现在尝试下面的命令，采用 tee 命令将程序输出写到标准输出和文件：

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

现在程序会交互式地运行³¹，并且备份输出到文件 log。

2.3.10 后处理

现在可按照之前的方法对结果进行后处理。用户可以监控相分数 alpha.water 随时间的发展变化，如图 2.22。

2.3.11 并行运行

前面的溃坝算例是使用一个相当粗糙的网格来生成的。我们现在希望增加网格数量并重新运行算例。新的算例如果采用单核计算会运行好几个小时，因此，如果用户能够使用多核运算，这样会大大减少运行时间。我们将介绍 OpenFOAM 的并行处理能力。用户首先应复制一份 damBreak 算例，执行以下命令：

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

然后，进入新算例的文件夹并将 blockMeshDict 字典中的 blocks 修改为：

```
blocks
(
```

³⁰ 调整 nAlphaSubCycles 的值会在求解压力之前多次求解相方程，调节 nAlphaCorr 会在当前时间步内对相场和压力场的循环做多次求解，更多信息请查阅 OpenFOAM 有关 interFoam 求解器的介绍

³¹ 终端也会显示

```
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)
);
```

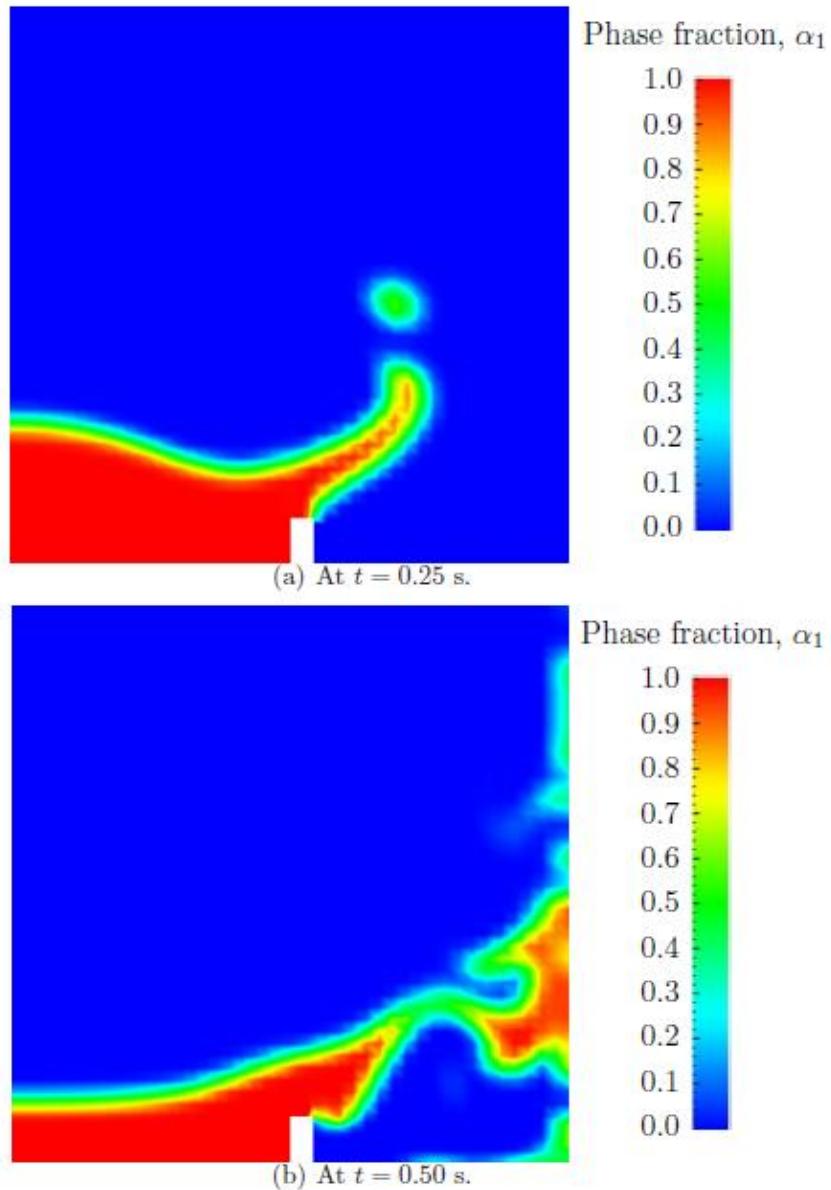


图 2.22 相分数场 α

在这里，用户必须改变网格密度，比如把其中 3 个方向的的网格数目改为 (46 10 1)，网格非均匀关键词也应该适当修改例如改为 (1 2 1)。一旦这些信息修改正确，运行命令即可生成网格。

由于现在的网格与之前的 damBreak 算例网格相比有变化，因此用户必须重新初始化 0 文件夹里的相场 alpha.water，因为在新的网格下，alpha.water 的非均匀 (nonuniform) 内部场有很大的变动。我们没有必要改变 U 和 p_rgh 因为它们被指定为 uniform，独立于场中的元素个数。我们希望初始化相分数场使之含有尖锐界面，也就是它的元素只含有 $\alpha = 1$ 或 $\alpha = 0$ 。通过 mapFields 更新场可能会在界面处产生插值的 $0 < \alpha < 1$ ，所以最好是重新运行 setFields 工具。用户应在执行 setFields 前将一个名为 0/alpha.water.org 的初始均匀 α 场的备份复制到 0/alpha.water：

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

OpenFOAM 所采用的并行计算方法称为区域分解法，该方法中，几何和相关的场被分割为若干部分并被分配到不同的处理器进行求解。因此并行运行一个算例的第一步是利用 decomposePar 工具分解求解域。算例的 system 目录下有一个与 decomposePar 程序相关的字典文件，其名为 decomposeParDict。同其它许多工具一样，它可以在特定工具的源码目录里找到一个默认的字典文件，本例中的字典文件就位于 \$FOAM_UTILITIES/parallelProcessing/decomposePar 目录下。

第一项是 numberOfSubdomains，它指定了要将算例分割称为的子区域数量，通常与可用于运算此算例的处理器数一样。

本例中，区域分割的方法我们选择为 simple，其对应的 simpleCoeffs 应按如下条件进行编辑。求解域在 x, y 和 z 方向被分割为子块（子区域），各个方向子区域的数量由向量 n 给定。由于本算例的几何是 2 维的，第 3 个方向，z 方向，不能被分割，所以必须等于 1。 n 向量的 n_x 和 n_y 分量在 x 和 y 方向分割求解域，必须指定来使由 n_x 和 n_y 决定的子区域的数量与由 numberOfSubdomains 指定的相同，也就是 $n_x \times n_y = \text{numberOfSubdomains}$ 。我们最好使每个子区域的网格连接面数量为最少。因此，对于这个正方形几何，最好保持 x 和 y 方向的分割数量相等且为偶数。关键字 delta 应设置为 0.001。

举例来说，假设我们希望用 4 个处理器运行程序。我们应设置 numberOfSubdomains 为 4 并且 $n = (2,2,1)$ 。运行 decomposePar 的时候，我们可以从屏幕消息中看到求解域被平均分配在各个处理器之中。

关于怎么并行运行一个算例的详细介绍读者应参看 3.4 节，本例中我们仅仅是举一个并行运算的例子。我们使用 openMPI 作为 MPI 的实现。如果用户想测试这个例子，可以在单节点即本地主机上并行运行，通过以下命令实现：

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

用户也可以在网络上的多个节点间运行，如 3.4.2 节中介绍的那样，通过创建一个文件，列出算例将要在其上运行的主机的主机名等来进行设置，算例应在后台来运行，用户可以像往常一样通过监测 log 文件跟踪其进度。

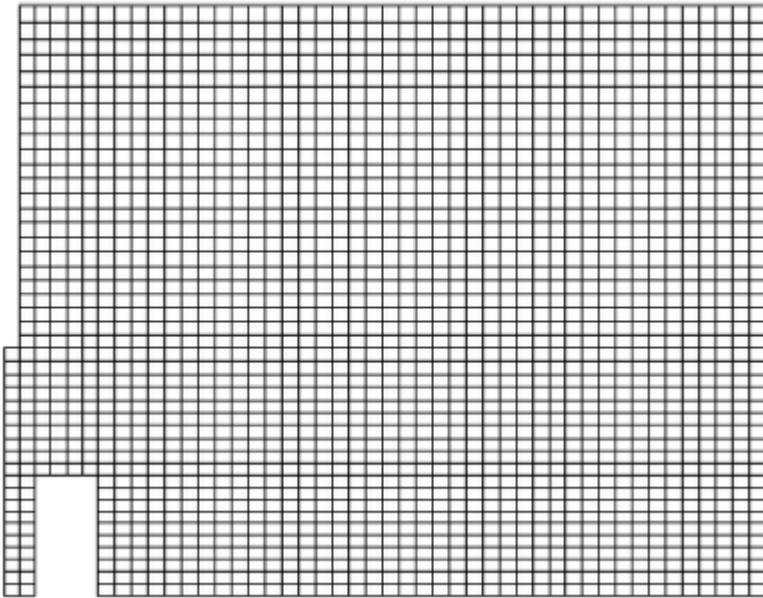


图 2.23 并行处理算例中处理器 2 的网格

2.3.12 算例的并行后处理

一旦算例运行结束，为了使用其它软件来进行后处理必须使用 `reconstructPar` 工具将分割后的场和网格重新合并。简单地以命令行的方式来运行它就可以。细网格的结果如图 2.24 所示，用户可以看到界面的分辨率较粗网格明显提高了很多。

用户也可以单独地对分割后的求解域的一部分进行后处理，思想即为：简单地将各处理器文件夹看做一个独立的算例。用户可以通过以下的命令来运行 `paraFoam`:

```
paraFoam -case processor1
```

于是 `processor1` 将会以算例模块出现在 `ParaView` 中。图 2.23 显示了采用 `simple` 方法分割后的处理器 1 的网格。

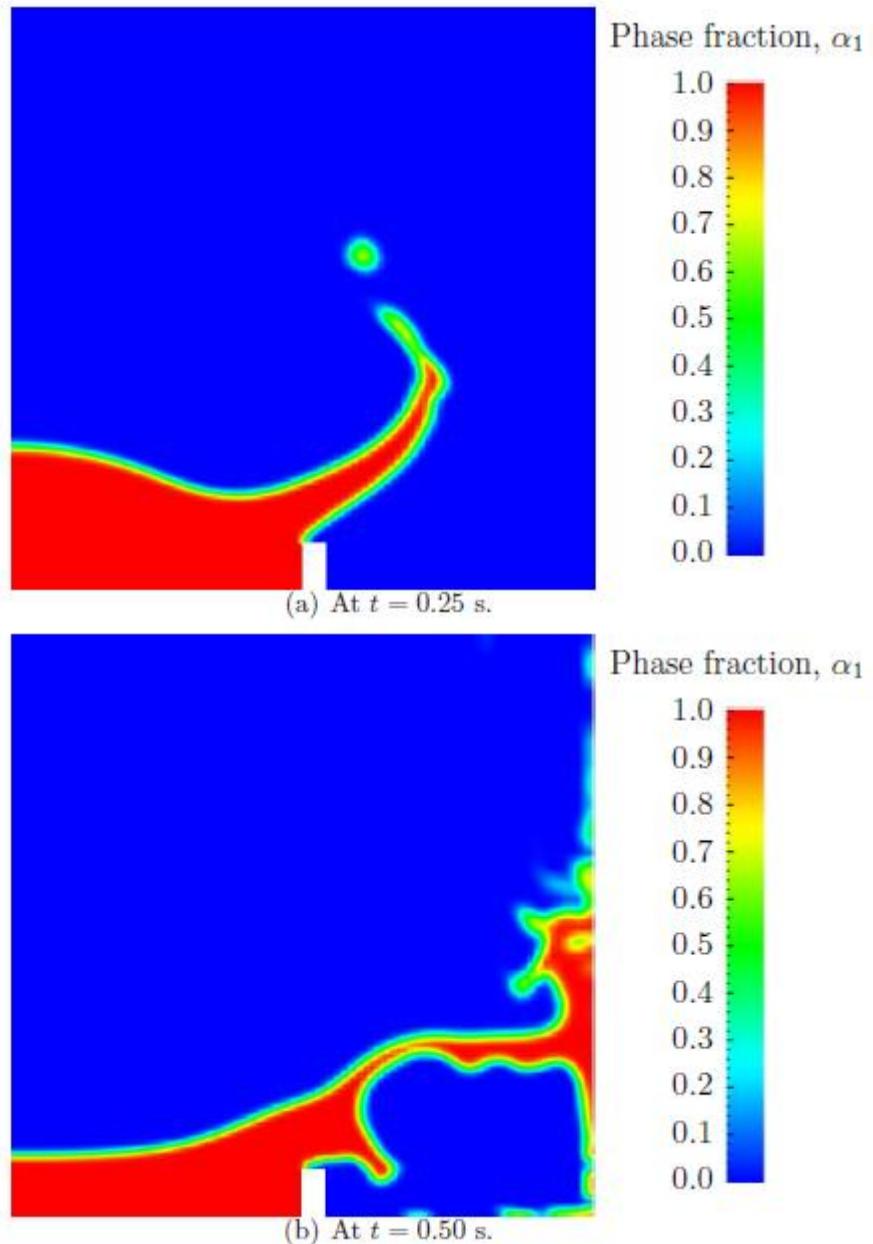


图 2.24 改善网格后的 α 相分数场

第三章

应用和库

我们反复重申: OpenFOAM 只是一个 C++ 库, 用于创造可执行程序 (applications)。OpenFOAM 在发布的时候附带了大量可编译的 (源代码) 或已经编译的 (deb 包) 程序。但用户也可以随心所欲的创建它们自己的程序。程序在 OpenFOAM 中分为两个类型:

- solvers (程序): 依据连续介质机制模型创造的求解器, 每个都用来求解一个特定的问题。
- utilities (工具): 执行简单的前后处理任务, 主要负责数据操作和代数几何运算。

OpenFOAM 内部包含了许多编译好了的库, 在编译程序和工具时候它们动态地被连接在一起。其中的物理模型库以源代码的方式提供, 所以用户可以添加自己的模型在各种库中。这一章我们概览一下这些程序, 工具和库。以及如何创建、修改、编译、执行。

3.1 OpenFOAM 编程语言

为了理解 OpenFOAM 库是如何工作的, 了解一些 C++ (OpenFOAM 使用的语言) 的背景知识是非常必要的。这些必要的信息将在这一个章节呈现。在这之前, 我们针对计算机语言这个概念来谈一下对对象编程, 以及为什么选择 C++ 作为 OpenFOAM 语言的问题。

3.1.1 普适性编程语言

语言以及数学的成功建立在有效性基础之上, 特别是在表达抽象概念的时候。例如, 在流体中, 我们使用“速度场”来作为文字表达, 它没有包含任何有关流体的信息和数据。这个名词将运动的方向、大小以及其它物理量封装起来。在数学中, 我们用一个符号来代表速度场: \mathbf{U} , 然后用这个符号进一步表示其它信息。例如“速度场的大小”我们用 $|\mathbf{U}|$ 表示。数学符号优越于语言符号的根本原因在于它的有效性, 数学符号可以简洁的表示复杂的概念。

我们想解决的连续介质问题无法使用计算机可识别的概念 (例如 bits, bytes, integers) 来表示。因此, 连续介质问题首先使用文字语言来呈现, 然后以时间和空间三个维度的偏微

分方程来表示。偏微分方程会包含如下信息：标量、矢量、张量、张量运算，量纲分析等。这些方程的求解涉及到离散、矩阵运算、求解器以及求解算法。

3.1.2 面向对象和 C++

面向对象编程语言，比如 C++，提供了一个有用的工具：类 (class)，来声明自己的类型和操作，这和工程以及科学中的数学语言是一样的。我们之前用 **U** 来表示的速度场，在编程中也可以用字符 **U** 来表示，速度场的大小 $|U|$ 我们在编程中用 **mag(U)** 来表示。在面向对象编程中，速度场以 **vectorField** 这个类来表示。速度场 **U** 可以说是 **vectorField** 类的一个对象，这就是面向对象编程。

在编程中使用对象来表示物理对象和抽象实体带来的便捷性不容低估。创建这种可以使
用代码中所有结构的类分层使得代码管理更容易。新的类可以从其它的类上继承。例如
vectorField 可以从 **vector** 类以及 **Field** 类上继承过来。C++也提供了模板类机制 例如
Field<Type> 可以表示任何的 **<Type>** 场，例如标量、矢量、张量。模板类的特性保留在任何从
模板类创造的类型上。模板和继承大大简化了重复代码并且使得整体的代码结构更加清晰。

继承
模板类

3.1.3 方程呈现

OpenFOAM 代码设计的中心主题就是使用 OpenFOAM 的类来编写求解器，这些类应该能够很好的表示某个特定的待求解的偏微分方程。例如这个方程：

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

它可以用如下代码来呈现：

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

如果想让 OpenFOAM 使用自己的类表示这个偏微分方程，并且满足一些其它的一些某些要求，这就需要 OpenFOAM 语言具有面向对象的特性，例如继承、模板类、虚函数以及操作符重载。某些计算机语言声称可以实现这些特性（例如 FORTRAN-90），但实际上这些特性在这些语言中是非常有限的。但是 C++ 完全拥有这些特性，同时 C++ 还具有由于其规范标准进而被广泛使用的优势，从而可以使用可靠的编译器来编写高效的可行程序。这就是 OpenFOAM 为什么采用 C++ 的原因。

3.1.4 求解器代码

求解器代码的编写是有步骤可循的。因为本身求解器代码就是数学算法和方程组的体现，算法和方程组求解本身，也是有步骤化的。编写一个求解器，用户并不需要纯熟的面向对象 C++ 编程技能，但是，应该知道面向对象的类型主要规则，以及了解基本的 C++ 句法。相对于编程语言，理解这个方程组、模型和求解步骤更加重要。

用户并不需要每时每刻都研究 OpenFOAM 的代码。面向对象编程的精髓就在于用户不需要这么做。仅仅知道这个类的存在以及它的功能已经足够。每个类及其功能的描述在 doxygen 中都有描述。请查阅：\$WM_PROJECT_DIR/doc/Doxygen/html/index.html。



3.2 编译程序和库

编译是程序开发必不可少的过程，由于每一段代码都需要跟某些独立的 OpenFOAM 库来进行对接，因此需要小心对待。在 Unix/Linux 系统中，这些通常需要使用 UNIXmake 程序来传递给编译器进行编译。然而 OpenFOAM 提供了 wmake 程序脚本，它建立在 make 之上，但是 wmake 更加多样性并且简单好用；实际上，不仅仅在 OpenFOAM 的代码上，wmake 也可以用在任何代码上使用。为了理解编译过程，我们首先需要讲解一下 C++ 及其文件结构，参见图 3.1。类是通过类构造函数和类成员函数等一系列代码来构成，这些代码存在某些文件中。类的文件后缀名为.C. 例如 nc 类的文件名为 nc.C。这个文件可以进行单独编译，编译后为一个共享的库文件可供其它程序调用，其后缀名为.so，比如 nc.so。当编译一个使用这个 nc 类的新程序的时候，比如 newApp.C 的时候，nc.C 不需要重新编译。这即为动态编译。

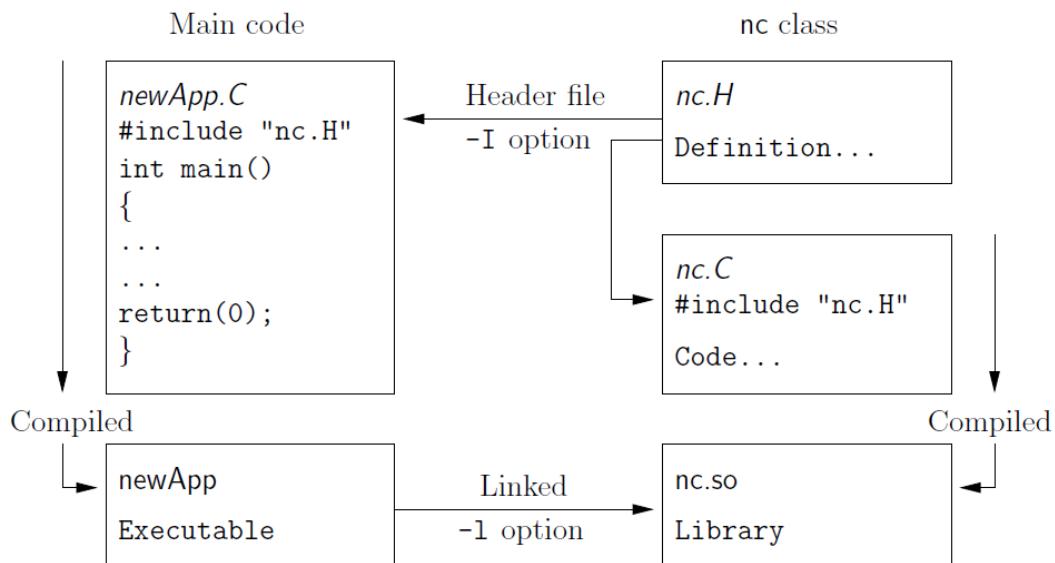


图 3.1 头文件、源代码、编译和连接

3.2.1 头文件

作为预防错误的发生，编译器必须知道它编译的类以及执行的类函数真实的存在。因此每个类都有一个类声明，它们被放置在后缀为.H 头文件中。例如 nc.H. 这个文件里面包含了类的名称以及类函数，在任何使用该类的代码中都需要包含这个文件。以.C 文件为后缀的程序如果需要这些类必须在文件开始提前声明这些以.H 为后缀的类声明。这样，通过递归下降的方式，搜索类层次结构，可以产生一个完整的、以顶层.C 文件为最终依赖的头文件列表，这些.h 文件称为依赖项。对于一个依赖项列表，编译器可以检查自上次编译后源文件是否有更新，以选择性地只编译那些需要重新编译的文件。

代码中，我们通过# include 语句来包含头文件，例如：

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

会让编译器从当前的文件中暂停读取，去读取指定的文件。任何独立代码块都可以放进头文件中，并在主函数中的任意位置被包含，以提高代码的可读性。例如，在大多数的 OpenFOAM 程序中，创建场和读取场的代码经常被放在 createFields.H 中，这个头文件在程序开始就被调用。在这种方式下，头文件就并不是仅仅是类声明了。

在 OpenFOAM 中，wmake 可以用于执行以下所列功能中的维护依赖文件列表的任务，并可用于编译源程序：

- 依赖文件列表的自动生成和变更。依赖文件列表就是那些包含在程序中的文件名的列表，程序依靠这些文件的组合来被编译；
- 通过合适的目录结构，多平台编译和链接；
- 多语言编辑和链接，例如 C, C++, JAVA;
- 多种编译和链接选项，例如调试，最优化，平行，性能分析；
- 支持源代码分析程序，例如 lex, yacc, IDL, MOC;
- 简单的源文件列表语法规则；
- 源文件列表自动生成；
- 静态库或共享库的简单处理；
- 新平台的可扩展性和兼容性；
- 便携性，可以在有 make, sh, ksh 或者 csh, lex, cc 的系统上应用；
- 已经在下面的电脑上通过测试：Apollo, SUN, SGI, HP (HPUX), Compaq (DEC), IBM (AIX), Cray, Ardent, Stardent, PC Linux, PPC Linux, NEC, SX4, Fujitsu VP1000；

3.2.2 使用 wmake 进行编译

OpenFOAM 程序以一种规范的格式来架构，每个程序的源代码放置在以这个程序命名的文件夹中。最顶层的源文件以.C 命名。例如，一个叫做 newApp 的源代码放置在 newApp 的文件夹，顶层文件即为 newApp.C，参见图 3.2.:

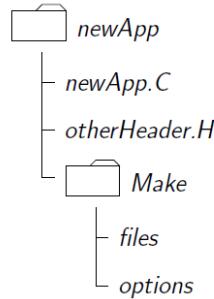


图 3.2 程序的文件架构

这个文件夹必须包含一个叫做 Make 的文件夹，其中有 options 以及 files 文件，我们在后面来讨论它们的作用。

3.2.2.1 包含文件头

编译器按照以下顺序在某个文件夹下寻找被包含的头文件，在 wmake 中，它们以-I 标识符来指定：

1. \$WM_PROJECT_DIR/src/OpenFOAM/InInclude 文件夹；
2. 本地 InInclude 文件夹，例如 newApp/InInclude；
3. 本地文件夹，例如 newApp；
4. \$WM_PROJECT_DIR/wmake/rules/-\$WM_ARCH 文件中设定的依赖文件路径，例如，/usr/X11/include 目录或者\$(MPICH_ARCH_PATH)/include 目录；
5. 其它在 Make/options 文件中通过-I 指定的其它文件夹；

Make/options 使用按照下面的语句规则来包含文件路径：

```

EXE INC = \
-I<directoryPath1> \
-I<directoryPath2> \
... \
-I<directoryPathN>
  
```

首先需要注意的是，每个文件夹路径的前面都有一个 **I** 的标签，在 EXE_INC 之后以及每个文件路径后需要使用 \，最后的文件路径后没有 \

3.2.2.2 链接库

编译器在以下路径链接库文件，在 wmake 中，它们通过-L 标识符来指定：

1. \$FOAM_LIBBIN 目录；
2. \$WM_DIR/rules/\$WM_ARCH 文件中设定的依赖文件路径，例如./usr/X11/lib 以及 \$(MPICH_ARCH_PATH)/lib；
3. Make/options 中指定的其它目录；

链接的库文件必须通过-l 标示符来指定，并且去掉 lib 前缀以及.so 后缀。例如：libnew.so，应该使用-lnew。默认情况下，wmake 调用下面的库文件：

1. \$FOAM_LIBBIN 文件夹下面的 libOpenFOAM.so 库；
2. \$WM_DIR/rules/\$WM_ARCH 文件中设定的依赖库路径，例如./usr/X11/lib 以及 \$(MPICH_ARCH_PATH)/lib；
3. Make/options 中指定的其它目录；

Make/options 文件使用下述语法指定库文件路径以及名称：

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

值得重申的是，库文件路径使用-L 标识符指定，库名称使用-l 标识符指定。

3.2.2.3 编译源文件

编译器编译的时候需要一个文件列表，除了主程序.C 文件外，需要包含其它没有作为库动态链接的文件并且打算被编译的文件，这个文件列表是必须存在。例如，用户想创建一个新的类或者在一个存在的类上添加一些新的函数。在 Make/files 中必须指定完整的以.C 为后缀的文件列表。许多程序的文件列表只有主程序的文件名。例如，我们之前的例子的文件列表

中只有 newApp.C。

Make/files 文件包含了需要进行编译的程序全路径以及名字（用 EXE = 来指定）。标准做法是使用求解器的名字，例如在我们的例子中我们使用 newApp。OpenFOAM 提供了两种可选的路径，一个是标准的路径，编译好的求解器存储在\$FOAM_APPBIN 中；另一个是用户自己的路径存储在\$FOAM_USER_APPBIN 中。

如果用户打算开发它们自己的新程序，我们建议它们在\$WM_PROJECT_USER_DIR 中创建一个子目录，并把它们自己的代码放置于其中。标准程序中，每一个 OpenFOAM 求解器代码都存储在相应的目录。用户自定义的求解器和标准求解器的唯一区别在于 Make/files 文件，它指定用户自定义程序写入\$FOAM_USER_APPBIN 中。Make/files 文件应该是这样的：

源代码位置
编译好的文件位置

newApp.C

EXE = \$(FOAM_USER_APPBIN)/newAPP

3.2.2.4 运行 wmake

wmake 可以这样来运行：

wmake <optionalArguments> <optionalDirectory>

<optionalDirectory> 是被编译程序的文件路径。一般来说，程序在自己的路径下进行编译，这样<optionalDirectory>就可以省略。

如果用户想创建一个可执行程序，那么就不需要<optionalArguments>。但是，如果在编译库的时候，<optionalArguments>是需要制定的。参见表 3.1：

命令参数	编译类型
lib	静态链接库
libso	动态链接库
libo	静态链接目标文件
jar	JAVA 存档文件
exe	可执行文件

表 3.1 wmake 的可选附加命令

3.2.2.5 wmake 环境变量设置

表 3.2 明确的列举了 wmake 所需设置的环境变量。

3.2.3 移除依赖包文件：wclean 和 rmdepall

在编译的过程中，wmake 会创建一个依赖包文件，扩展名为.deb（在我们的例子中为 newApp.dep）。并在 Make/\$WM_OPTIONS 中产生一系列文件。如果用户想要删除这些文件，

比如在一些代码被改动的情况下，用户可以运行 wclean 来删除：

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

跟 wmake 相同的是，<optionalDirectory>是被编译程序的文件路径。一般来说，程序在自己的路径下来运行 wclean，这样，<optionalDirectory>就可以省略。

路径	
\$WM_PROJECT_INST_DIR	安装目录路径，例如\$HOME/OpenFOAM
\$WM_PROJECT	编译工程名
\$WM_PROJECT_VERSION	工程版本号，例如 2.3.0
\$WM_PROJECT_DIR	可执行文件目录的全路径，例如 \$HOME/OpenFOAM/OpenFOAM-2.3.0
\$WM_PROJECT_USER_DIR	用户可执行文件目录的全路径，例如 \$HOME/OpenFOAM/\${USER}-2.3.0
其他设置	
\$WM_ARCH	主机架构：Linux, SunOS
\$WM_ARCH_OPTION	机器位数，32 位或者 64 位
\$WM_COMPILER	所使用的编译器：Gcc43 – gcc 4.5+, ICC - Intel
\$WM_COMPILER_DIR	编译器安装目录
\$WM_COMPILER_BIN	编辑器二进制文件\$WM_COMPILER_BIN/bin
\$WM_COMPILER_LIB	编译器库\$WM_COMPILER_BIN/lib
\$WM_DIR	wmake 全路径
\$WM_MPLIB	并行库：LAM, MPI, MPICH, PVM
\$WM_OPTIONS	= \$WM_ARCH\$WM_COMPILER...\$WM_COMPILE OPTION\$WM_MPLIB，例如 linuxGcc3OptMPICH
\$WM_PRECISION_OPTION	编译二进制文件的精度，SP 为单精度，DP 为双精度

表 3.2 wmake 环境变量设置³²

如果用户想要删除依赖文件和 Make 目录中的其它文件，就不需要<optionalArguments>。然而，如果<optionalArguments>中指定了 lib，那么本地的 inInclude 文件也会被删除。另一个程序是 rmdepall，它从执行点开始自上而下移除所有的.dep 文件，这在升级 OpenFOAM 库的时候是非常有用的。

3.2.4 编译实例： pisoFoam 求解器

pisoFoam 的源代码在\$FOAM_APP/solvers/incompressible/pisoFoam 的文件目录下，最顶层的源代码文件为 pisoFoam.C。如下所示：

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

³² 默认不需更改

```
=====
 \ / Field      | OpenFOAM: The Open Source CFD Toolbox
  \ / Operation  | Copyright (C) 2011-2012 OpenFOAM Foundation
   \ / And       |
    \ Manipulation |
```

License

This file is part of OpenFOAM.

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

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

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

Application

pisoFoam

Description

Transient solver for incompressible flow.

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

```
/*
 *-----*/
#include "fvCFD.H"
#include "singlePhaseTransportModel.H"
#include "turbulenceModel.H"

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

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

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

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

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

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

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

            UEqn.relax();

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

            // --- PISO loop

            for (int corr=0; corr<nCorr; corr++)
            {
                volScalarField rAU(1.0/UEqn.A());
                volVectorField HbyA("HbyA", U);
                HbyA = rAU*UEqn.H();
                surfaceScalarField phiHbyA
                (
                    "phiHbyA",
```

```

        (fvc::interpolate(HbyA) & mesh.Sf())
        + fvc::ddtPhiCorr(rAU, U, phi)
    );
adjustPhi(phiHbyA, U, p);

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

    fvScalarMatrix pEqn
    (
        fvm::laplacian(rAU, p) == fvc::div(phiHbyA)
    );
    pEqn.setReference(pRefCell, pRefValue);

    if
    (
        corr == nCorr-1
        && nonOrth == nNonOrthCorr
    )
    {
        pEqn.solve(mesh.solver("pFinal"));
    }
    else
    {
        pEqn.solve();
    }

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

#include "continuityErrs.H"

U = HbyA - rAU*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;
}

// ****

```

代码最开始是对求解器的一段简单描述。//用于单行评论，多行的评论位于/*...*/之间³³。在这之后，代码是# include 语句，例如# include “fvCFD.H”，这时，编译器会暂停读取 pisoFoam.C 文件，转而去读取 fvCFD.H 文件。

pisoFoam 使用 incompressibleRASModels, incompressibleLESModels 和 incompressibleTransportModels 库。因此它们需要必要的头文件，它们通过 EXE_INC=-I ... 来指定，链接的库文件通过 EXE_LIBS = -l ... 来指定。Make/options 里面信息如下：

```

1 EXE_INC = \
2   -I$(LIB_SRC)/turbulenceModels/incompressible/turbulenceModel \
3   -I$(LIB_SRC)/transportModels \
4   -I$(LIB_SRC)/transportModels/incompressible/singlePhaseTransportModel \
5   -I$(LIB_SRC)/finiteVolume/lnInclude
6
7 EXE_LIBS = \
8   -lincompressibleTurbulenceModel \

```

³³ 评论相当于注释。起解释说明的作用，编译的时候跳过

```

9   -lincompressibleRASModels \
10  -lincompressibleLESModels \
11  -lincompressibleTransportModels \
12  -lfiniteVolume \
13  -lmeshTools

```

pisoFoam 仅仅包含 pisoFoam.C 源文件, 可执行程序和其它程序一样写入到\$FOAM_APPBIN 文件目录中。Make/Files 包含以下代码:

```

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

```

用户可以切入到\$FOAM_SOLVERS/incompressible/pisoFoam 文件目录下, 键入:

wmake

来进行编译。编译过程会产生下列信息:

```

Making dependency list for source file pisoFoam.C
SOURCE DIR=.
SOURCE=pisoFoam.C ;
g++ -DFOAM_EXCEPTION -Dlinux -DlinuxOptMPICH
-DscalarMachine -DoptSolvers -DPARALLEL -DUSEMPI -Wall -O2 -DNoRepository
-ftemplate-depth-17 -I../OpenFOAM/OpenFOAM-2.3.0/src/OpenFOAM/lnInclude
-IlnInclude
-I.
.....
-lmpich -L/usr/X11/lib -lm
-o .../OpenFOAM/OpenFOAM-2.3.0/applications/bin/linuxOptMPICH/pisoFoam

```

用户可以尝试再次编译这个程序, 它会显示下面的信息, 表明可执行程序已经是最新版本且没有必要重新编译了:

```

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

```

用户可以通过:

wclean

移除依赖文件, 从新开始使用 wmake 来进行编译。

3.2.5 调试与优化开关

OpenFOAM 中存在一个“运行时消息输出机制”, 它们有助于对编译或者运行时出现的问题进行调试。调试开关在\$WM_PROJECT_DIR/etc/controlDict 文件中设置; 如果用户想改变默认设置, 它们最好把它备份到\$HOME 文件下, 例如备份到这个位置: \$HOME/.OpenFOAM/2.3.0/controlDict。默认的调试开关非常多, 我们可以通过 foam 调试 Switches 程序来查看。大多数开关和类以及可执行程序相对应。它们可以通过在 controlDict 中设置为 1 来打开; 例如, OpenFOAM 可以通过把 controlDict 中的 dimensionSet 设置为 1 来检查量纲的一致性。

表 3.3 我们列举了一下最高等级的调试开关选项。

另外，有些调试开关和某个操作和最优化问题相对应。它们在表 3.3 中列出。其中比较重要的是 `FileModificationSkew` 选项。`OpenFOAM` 通过对数据的写入时间确定数据是否被更改。当在计算网络上运行算例的时候，不同计算机的时钟可能不同，某些数据可能会被认为是修改的，这可能会引起混乱，例如下面这种情况：当 `OpenFOAM` 把所有这些数据当做刚刚修改的新数据并进行读取。`fileModificationSkew` 关键词是文件更新系统的最大等待时间，其以秒为单位，它用来判断文件是否是刚刚修改的。

调试开关—— DebugSwitches 子字典	
<code>level</code>	<code>OpenFOAM</code> 总体调试等级：共 3 个，即为 0, 1, 2
<code>lduMatrix</code>	运行时求解器收敛调试等级：共 3 个，即为 0, 1, 2
优化开关—— OptimisationSwitches 子字典	
<code>fileModificationSkew</code>	<code>OpenFOAM</code> 运行的计算网络不同计算机的最大延迟，以秒为单位，其值应该高于这个最大延迟
<code>fileModificationChecking</code>	通过读取 <code>timeStamp</code> 或者使用 <code>inotify</code> 来判断是否在运行时有文件被修改
<code>commsType</code>	并行通信类型： <code>nonBlocking</code> , <code>scheduled</code> , <code>blocking</code>
<code>floatTransfer</code>	如果为 1，则在通信之前转换为单精度，默认为 0 ³⁴
<code>nProcsSimpleSum</code>	<code>simple</code> 并行分解法的最高值，超于此数值使用 <code>hierarchical</code> 进行分解，默认为 16

表 3.3 运行时输出消息开关

3.2.6 使用自定义库

当用户打算创建一个新的库的时候，比如名称为 `new` 的库，它希望这个库可以被一系列求解器所调用。例如，用户可能打算创建一个新的边界条件，并编译进入 `new` 中，并且被求解器、前处理后处理程序以及网格工具等识别。在正常的情况下，用户需要把 `new` 连接到所有的程序并重新编译。

实际上 `OpenFOAM` 提供了一种动态链接机制以实现连接一个或多个附加库。我们只需要简单的在 `controlDict` 中添加一些关键词即可，并在其中键入新的库名称。例如，如果用户在运行的时候打算使用 `new1` 和 `new2` 库，我们只需要把下列信息添加到 `controlDict` 中：

```
libs
(

```

³⁴ 在通信带宽称为计算速度的瓶颈的时候，可以尝试

```

“libnew1.so”
“libnew2.so”
);

```

3.3 运行程序

对某个算例文件进行读取和写入的程序都可以在终端通过命令行来运行。每个算例的数据文件都存储在这个算例之下，参考 4.1 节。在这里我们用`<caseDir>`来表示算例的全路径。对于任何可执行程序，都可以在可执行程序后加上`-help` 来查找命令行的附加命令。例如键入：

```
blockMesh -help
```

会输出以下信息：

```

Usage: blockMesh [-region region name] [-case dir] [-blockTopology]
                  [-help] [-doc] [-srcDoc]

```

方括号中的参数为可选参数。在某个算例目录下执行某个可执行程序，那么这个程序便对这个算例目录下的文件进行操作。相反的，如果我们键入`-case<caseDir>`附加命令，这个程序就会对系统内任意目录下的数据文件进行操作。

正如任何 Unix/Linux 可执行程序一样，程序可以以后台的方式来运行。比如，在用户执行其它命令的时候，这个程序可以一直执行下去。如果用户想以后台的方式来执行程序，并把输出流打印到 log 中，它们可以这样键入：

```
blockMesh > log &
```

3.4 并行计算

这一章我们讨论如何使用多个节点来并行计算。`OpenFOAM` 使用的并行计算方法为拆分计算域法。在这个方法中几何和附属场被拆分为单独的块，每个块用单独的 cpu 来进行计算。并行计算主要涉及到网格和场的分解、并行运行程序、分解场的后处理。我们会在下面的章节详细讨论。在并行计算中我们需要借助第三方 MPI 工具，例如 `openMPI`。

3.4.1 网格分解与初始场数据

`decomposePar` 程序用来分解网格和场。我们的目的是以最小的资源分解计算域，通过 `decomposeParDict`（位于算例的 `system` 文件夹下）字典文件的参数，几何和场文件就可以被分解。在 `interFoam/damBreak` 算例中我们已经有了一个 `decomposeParDict` 文件，我们可以拷贝它来使用。字典信息如下：

```

17
18 numberOfSubdomains 4;
19
20 method simple;

```

```

21
22 simpleCoeffs
23 {
24     n          ( 2 2 1 );
25     delta      0.001;
26 }
27
28 hierarchicalCoeffs
29 {
30     n          ( 1 1 1 );
31     delta      0.001;
32     order      xyz;
33 }
34
35 manualCoeffs
36 {
37     dataFile    "";
38 }
39
40 distributed    no;
41
42 roots         ();
43
44
45 // ****

```

用户有 4 个分解方法，它们通过 `method` 关键词来指定：

- `simple`: 简单的几何分解。计算域依据方向被切分，例如 x 方向两块，y 方向一块；
- `hierarchical`: 顺序几何分解法和简单分解法差不多，只不过用户指定首先切分哪个方向，例如首先切分 x 方向，然后切分 y 方向；
- `scotch`: `scotch` 分解方法不需要用户输入指定参数，并且力求使核的边界面最小化³⁵。针对不同的计算机处理器，用户可以针对计算机资源而为每个计算机分配不同的权重，这可以通过 `processorweights` 关键词来指定。另外一个可选关键词是 `strategy`，它可以对 `scotch` 分解法做出一些限定。更多的信息请查阅相关源文件：
`$FOAM_SRC/decompositionMethods/decompositionMethods/scotchDecomp/scotchDecomp.C`;
- `manual`: 手动分解法，用户可以直接指定给某个求解器某一片网格区域；

对于每个 `method`³⁶，我们都需要指定一些参数，我们可以在 `decomposeParDict` 字典中的`<method>-Coeffs` 来指定。表 3.4 指明了 `decomposeParDict` 所需要的全部关键信息。`decomposePar` 程序可以这样来执行：

decomposePar

然后它会在算例文件的目录下为每个核产生一个子文件夹。它们的名字为 `processorN`，此处 N 为 0, 1... 其代表着求解器的编号，内部包含了时间步文件，分解场信息，以及 `constant/-ployMesh` 文件夹（分解后的网格）。

³⁵ 请参考：<http://www.cfd-online.com/Forums/openfoam-solving/81051-large-case-parallel-efficiency.html>，此方法分解后的网格块是不规则形状。

³⁶ 另外一个主要的分解法是 `metis`，分解后的网格和 `scotch` 一样不具有规则性，此方法需要下载 `metis` 库来执行。其他方法略去。

关键信息		
numberOfSubdomains	分解域的总数	N
method	分解方法	simple hierarchical scotch metis manual
simpleCoeffs 字典		
n	x, y, z 方向上的分解数量	(n_x, n_y, n_z)
delta	网格偏斜因子	0.001
hierarchicalCoeffs 字典		
n	x, y, z 方向上的分解数量	(n_x, n_y, n_z)
delta	网格偏斜因子	0.001
order	分解顺序	xyz/yzx/xzy
scotchCoeffs 字典		
processorWeights (可选)	每个处理器分配的权重, <wt1>就是第一个处理器分配的权重, 值可以随意选取	(<wt1>...<wt2>)
strategy (可选)	分解策略, 默认为 b	
manualCoeffs 字典		
dataFile	给各个处理器分配任务的字典文件名称	“<filename>”
数据分布字典 (可选)	查阅 3.4.3 节	
distributed	数据写入不同的硬盘?	yes/no
roots	算例目录路径, 例如<rt1>就是节点 1 的路径	(<rt1>...<rtN>)

表 3.4 decompositionDict 字典文件

3.4.2 运行分解算例

这样, 一个分解过的 OpenFOAM 算例就可以并行运行了。这需要借助 MPI 工具, 例如

openMPI。

在多核计算机上，使用 openMPI 并行计算非常简单。但是如果通过计算机网络来并行计算，我们必须创建一个包括宿主机器名字的文件。它可以放在任意位置，文件名随意。在下面的讨论中我们用<machines>代表这个包含全路径的文件。

在<machines>文件中，需要把每个使用 openMPI 并行求解的计算机名单行列出。这个名字应该和机器本身的/etc/hosts 文件中的名字相对应。这个文件必须包含这些名字。当某个节点内有多个核心可供计算的时候，在这个节点的名字后面还必须包含 cpu = n，其中 n 即为 openMPI 可以使用的核数。

例如，如果我们打算在 aaa 的机器上使用 openMPI 在 aaa, bbb（具有两个核心）, ccc 上并行计算。<machines>文件应该是这样的：

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

且使用下面的语句来运行：

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

其中：<nProcs>是核心数量；<foamExec>是可执行程序的名称，例如 icoFoam；log 为输出的日志文件。举例，如果使用 icoFoam 在 4 个节点（其由 machines 指定）上计算 cavity 算例（其位于\$FOAM_RUN/tutorials/incompressible/icoFoam 文件夹中），我们应该执行下面的命令：

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

3.4.3 多硬盘数据分布

用户有的时候可能打算在不同的硬盘里写入数据，在这种情况下，我们说写入的数据是分布式的。在进行分布式写入数据的时候，用户会发现在不同的节点中，写入数据的根目录是不同的。因此，在 decomposePar 字典文件中我们需要指定 distributed 和 roots 关键词，distributed 关键词这样指定：

```
distributed yes;
```

roots 下要包含每个节点的数据路径，例如<root0>, <root1>

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

其中`<nRoots>`是路径数。

依据 `decomposeParDict` 字典文件的设置，并行算例每个处理器的计算路径下都应该包含相应的 `processorN` 文件夹。除此之外，`system` 文件夹以及 `constant` 文件夹也应该存在。值得注意的是：`polyMesh` 文件夹可以省略，但除了 `polyMesh` 文件夹其他 `constant` 下的文件都应该包含。

3.4.4 并行后处理

当一个算例以并行的方式来计算的时候，用户有下面两种方式来进行后处理：

- 重组网格场和数据以呈现完整的网格和场，这样就可以正常的进行后处理；
- 单独对每个域进行后处理；

3.4.4.1 重组网格和数据

当算例并行计算完毕，就可以重组来进行后处理了。算例每个核下面的时间步文件，通过重组为当前算例目录下每个时间步下的文件并被整合为单一时间步的文件。重组程序 `reconstructPar` 可以这样来执行：

```
reconstructPar
```

当时据分布在几个硬盘中的时候，需要把他们都拷贝在宿主机再进行重组。

3.4.4.2 分解场后处理

正如 6.1 节所说，用户可以使用 `paraFoam` 来对每个核下的算例来进行分别后处理。并行计算的算例可以通过重组场的方式来后处理，以及对每个分解场进行单独后处理，这样的话，每个核下面的结果被当做一个单独的算例来看待。

3.5 标准求解器

OpenFOAM 的求解器源代码位于`$FOAM_SOLVERS` 文件夹下面，这可以通过 `sol` 命令快速的切换到此处。这个文件夹还依据模型类别的不同区分为不同的子文件夹，例如不可压缩流体求解器类，燃烧求解器类，应力分析求解器类等。每个求解器的名字非常具有描述性，例如 `icoFoam` 就是求解 `incompressible flow`(不可压缩流体) 的求解器。表 3.5 列举了 OpenFOAM 提供的求解器类型。

基本求解器	
<code>laplacianFoam</code>	拉普拉斯方程求解器，例如固体中的导热问题

potentialFoam	势流求解器, 求解后, 可做为 NS 方程的初始场
scalarTransportFoam	标量传输方程求解器
不可压缩求解器	
adjointShapeOptimizationFoam	对区域应用 blockage 的槽道稳态不可压缩非牛顿流体求解器(参见: Implementation of a continuous adjoint for topology optimization of ducted flows)
boundaryFoam	稳态不可压缩一维湍流求解器, 主要用于为进口生成边界层条件
icoFoam	牛顿流体瞬态不可压缩求解器
nonNewtonianIcoFoam	非牛顿流体瞬态不可压缩求解器
pimpleDyMFoam	采用 PIMPLE 算法的瞬态不可压缩牛顿湍流求解器, 可使用动网格
pimpleFoam	采用 PIMPLE 算法的大时间步瞬态不可压缩流求解器
pisoFoam	采用 PISO 算法的瞬态不可压缩流求解器
porousSimpleFoam	附带多孔介质模型的稳态不可压缩湍流求解器
shallowWaterFoam	瞬态无粘有旋潜水方程求解器
simpleFoam	稳态不可压缩湍流求解器
SRFSimpleFoam	稳态单一旋转参考系下的不可压缩湍流求解器
SRFPimpleFoam	采用 PIMPLE 算法和单一旋转参考系下的大时间步瞬态不可压缩流求解器
可压缩求解器	
rhoCentralDyMFoam	基于 Kurganov&Tadmor 中心迎风格式的密度基可压缩湍流求解器, 可使用动网格
rhoCentralFoam	基于 Kurganov&Tadmor 中心迎风格式的密度基可压缩湍流求解器
rhoLTSPimpleFoam	可压缩湍流瞬态求解器。支持多重参考系、多孔介质模型以及局部时间步下的稳态算法
rhoPimpleFoam	基于 PIMPLEC 算法的可压缩流求解器 (暖通类应用)
rhoPimpleFoam	基于 PIMPLE 算法的可压缩流求解器 (暖通类应用)
rhoPorousSimpleFoam	稳态附带 RANS 湍流模型的可压缩求解器, 附带多孔介质模型
rhoSimpleFoam	基于 SIMPLEC 的附带 RANS 湍流模型的稳态可压缩求解器
rhoSimpleFoam	基于 SIMPLE 的附带 RANS 湍流模型的稳态可压缩求解器

	解器
sonicDyMFoam	瞬态超声速可压气体求解器，可使用动网格
sonicFoam	瞬态超声速可压气体求解器
sonicLiquidFoam	瞬态超声速可压液体求解器
多相流求解器	
cavitatingDyMFoam	基于均相平衡模型（用于计算液体/蒸汽的混合可压性）的气蚀求解器，可使用动网格以及自适应网格
cavitatingFoam	基于均相平衡模型（用于计算液体/蒸汽的混合可压性）的气蚀求解器
compressibleDyMIn terFoam	基于 VOF 模型的可压、绝热、不可溶两相界面捕获求解器，可使用动网格
compressibleInterFo am	基于 VOF 模型的可压、绝热、不可溶两相界面捕获求解器
compressibleMultip haseInterFoam	基于 VOF 模型的可压、绝热、不可溶多相界面捕获求解器
interFoam	基于 VOF 模型的不可压、绝热、不可溶两相界面捕获求解器
interDyMFoam	基于 VOF 模型的不可压、绝热、不可溶两相界面捕获求解器，可使用动网格
interMixingFoam	三相不可压缩流体，其中两相互溶，基于 VOF 模型的界面捕获求解器
interPhaseChangeFo am	基于 VOF 模型的不可压、绝热、不可溶、存在相变的两相界面捕获求解器
interPhaseChangeDy MFoam	基于 VOF 模型的不可压、绝热、不可溶、存在相变的两相界面捕获求解器，可使用动网格
LTSInterFoam	局部时间步下使用 VOF 模型的不可压、绝热、不可溶两相界面捕获求解器
MRFInterFoam	多重参考系下使用 VOF 模型的不可压、绝热、不可溶两相界面捕获求解器
MRFMultiphaseInte rFoam	多重参考系下使用 VOF 模型的不可压、绝热、不可溶多相界面捕获求解器
multiphaseEulerFoa m	基于双流体模型的多相流体混合可压求解器，附带传热模型
multiphaseInterFoa m	基于 VOF 模型的不可压、绝热、不可溶多相界面捕获求解器
porousInterFoam	基于 VOF 模型的不可压、绝热、不可溶两相界面捕获

	求解器，附带多孔介质模型
potentialFreeSurfaceFoam	包含波高（zeta）的不可压缩 NS 方程求解器，可用于在单相下模拟自由表面的波高
settingFoam	模拟沉降的不可压缩两相求解器
twoLiquidMixingFoam	不可压缩两相混合求解器
twoPhaseEulerFoam	高相分数不可压缩两相求解器，典型应用为鼓泡床
直接数值模拟	
dnsFoam	计算域为立方体的各向同性湍流直接模拟求解器 ³⁷
燃烧求解器	
chemFoam	单网格化学反应问题求解器，主要用于和其他软件 ³⁸ 的求解结果相对比。网格和场从初始条件即时生成
coldEngineFoam	冷态内燃机求解器
engineFoam	内燃机求解器
fireFoam	瞬态火灾或湍流扩散火焰求解器
LTSReactingFoam	局部时间步长下的稳态可压层流或湍流求解器，附加化学反应模型
PDRFoam	附带湍流的可压预混燃烧求解器
reactingFoam	附带化学反应的燃烧求解器
rhoReactingBuoyantFoam	密度基于热力学模型、浮力强化模型的燃烧求解器，附带化学反应模型
rhoReactingFoam	密度基于热力学模型附带化学反应的燃烧求解器
xiFoam	附带湍流模型的可压预混/部分预混燃烧求解器
传热以及浮力驱动求解器	
buoyantBoussinesqPimpleFoam	附加湍流的瞬态不可压浮力驱动求解器
buoyantBoussinesqSimpleFoam	附加湍流的稳态不可压浮力驱动求解器
buoyantPimpleFoam	附加湍流的瞬态可压浮力驱动求解器（暖通和传热）
buoyantSimpleFoam	附加湍流的稳态可压浮力驱动求解器

³⁷ 计算域的每个方向网格需要为 2 的 n 次幂

³⁸ 例如 chemkin

chtMultiRegionFoam	heatConductionFoam 和 buoyantFoam 的结合，固体导热和液体传热的耦合求解
chtMultiRegionSimpleFoam	chtMultiRegionFoam 的稳态版本
粒子跟踪求解器	
coalChemistryFoam	包含以下性质的瞬态求解器：可压缩性、湍流、煤粉和石灰岩粒子注入、能量源项模型、燃烧模型
DPMFoam	瞬态流体-固体粒子耦合求解器，粒子可打包为粒子云，附带粒子相分数对连续相的影响
icoUncoupledKinematicParcelDyMFoam	瞬态流体-固体粒子耦合求解器，粒子可打包为粒子云，可使用动网格
icoUncoupledKinematicParcelFoam	瞬态流体-固体粒子耦合求解器，粒子可打包为粒子云
LTSReacttingParcelFoam	局部时间步长下的稳态、可压、化学反应（或无）、多相拉格朗日粒子、多孔介质、附带质量能量源项的求解器
reactingParcelFilmFoam	基于 PISO 的瞬态可压拉格朗日化学反应求解器，附带表面膜模型
reactingParcelFoam	基于 PIMPLE 的稳态可压、附带多相拉格朗日粒子的化学反应求解器，附带源项
simpleReactingParcelFoam	基于 SIMPLE 的稳态可压、附带多相拉格朗日粒子的化学反应求解器，附带源项
sprayEngineFoam	基于 PIMPLE 的瞬态可压发动机喷雾求解器
sprayFoam	基于 PIMPLE 的瞬态可压喷雾求解器
uncoupledKinematicParcelFoam	在流场已知的算例上将粒子打包为粒子云瞬态求解器
分子动力学模拟	
mdEquilibrationFoam	分子动力系统平衡或预处理求解器
mdFoam	用于流体动力学的分子动力学求解器
DSMC 求解器	
dsmcFoam	DSMC 求解器
电和磁	
electrostaticFoam	静电场求解器

magneticFoam	永磁场的磁场求解器
mhdFoam	不可压缩, 层流, 磁力驱动 MHD 求解器
应力分析	
solidDisplacementFoam	刚体微小线弹性应变的有限体积瞬态分离式求解器, 可选热扩散和热应力
solidEquilibriumDisplacementFoam	刚体微小线弹性应变的有限体积稳态分离式求解器, 可选热扩散和热应力
金融类	
financialFoam	期权定价 Black-Scholes 方程求解器

表 3.5 标准求解器

3.6 标准工具

OpenFOAM 附带的程序位于\$FOAM_UTILITIES 目录中, 我们可以在终端键入 util 快速的切换到文件目录。程序的名字带有描述性, 例如 ideasToFoam 就是把 IDEAS 网格转换为 OpenFOAM 可读取的网格形式。表 3.6 列举了当前的 OpenFOAM 附带的工具:

前处理	
applyBoundaryLayer	依据当前的速度场通过简单的七分之一定律计算边界层
applyWallFunctionBoundaryConditions	对 OpenFOAM 中的 RAS 算例使用 OpenFOAM-1.6 版本以后的壁面函数进行场更新
boxTurb	创建符合连续性方程的各项同性湍流场 ³⁹
changeDictionary	用来批处理改变字典信息, 例如可以用来修改 polyMesh/boundary 文件内的场的类型
createExternalCoupledPatchGeometry	和 externalCoupled 边界条件一起使用, 用来为某个场生成边界几何
dsmcInitialise	通过读取 system/dsmcInitialise 字典文件来为 dsmcFoam 进行初始化
engineSwirl	为发动机计算生成涡流场
faceAgglomerate	使用角系数辐射模型合并边界面。将原始网格的场映射到

³⁹ 各个方向的网格数需要为 2 的幂

	合并后的网格上
foamUpgradeCyclics	对 cyclics 边界面的网格和场进行更新
foamUpgradeFvSolution	对 system/fvSolution 中的 solvers 语法进行更新
mapFields	对某个网格上的场进行读取并插值最后投影到其他算例，其中时间步一一对应。可采用自己特有的命令参数来并行计算，也可单核计算。
mdInitialise	分子模拟初始场
setFields	通过本身的字典文件对网格和边界的场值进行设置
viewFactorsGen	基于融合 (agglomerated) 面计算角系数辐射模型的辐射角
wallFunctionTable	读取一个字典文件来计算合适的壁面函数值列表
网格生成	
blockMesh	多块网格生成器
extrudeMesh	从现有的 patch 或者几何文件上的 patch 上抽取网格，默认为对法向延伸
extrude2DMesh	在一个 2D 网格 (只需要 faces, points, cells, boundary) 上延伸成为一个 3D 网格，厚度可以指定
extrudeToRegionMesh	把网格的 faceZones 抽取为一个单独的网格，可以用来生成液膜域网格
foamyHexMesh	Conformal Voronoi 网格自动生成器
foamyHexMeshBackgroundMesh	写入由 foamyHexMesh 组建的背景网格并组建 distance - Surface
foamyHexMeshSurfaceSimplify	重提取并简化表面
foamyQuadmesh	Conformal Voronoi 二维抽取网格生成器
snappyHexMesh	自动进行表面贴合细化的六面体网格切分器
网格转换	
ansysToFoam	把 I-DEAS 制作的 ANSYS 的网格转换为 OpenFOAM 可用格式
ccm26ToFoam	把 CCM 网格转换为 OpenFOAM 可用格式
cfx4ToFoam	把 CFX 网格转换为 OpenFOAM 可用格式
datToFoam	读取 datToFoam(.dat) 网格文件并输出点文件，一般和 blockMesh 结合使用
fluent3DMeshToFoam	把 fluent 网格转换为 OpenFOAM 可用格式
fluentMeshToFoam	把 fluent2D 网格转换为 OpenFOAM 可用格式
foamMeshToFluent	把 OpenFOAM 网格转化为 fluent 网格格式

foamToStarMesh	把 OpenFOAM 网格转化为 PROSTAR 网格格式
foamToSurface	读取 OpenFOAM 网格并把边界转化为几何
gambitToFoam	把 gambit 网格转换为 OpenFOAM 可用格式
gmshToFoam	把.msh 网格转换为 OpenFOAM 可用格式
ideasUnvToFoam	把.I-Deas 网格转换为 OpenFOAM 可用格式
kivaToFoam	把 KIVA 网格转换为 OpenFOAM 可用格式
mshToFoam	把 Adventurede 生成的.msh 网格转换为 OpenFOAM 可用格式
netgenNeutralToFoam	把 Netgen 4.4 生成的.msh 网格转换为 OpenFOAM 可用格式
plot3DToFoam	把 Plot3d 网格转换为 OpenFOAM 可用格式
sammToFoam	把 STAR-CD(v3)的 SAMM 网格转换为 Open -FOAM 可用格式
star3ToFoam	把 STAR-CD(v3)的 PROSTAR 网格转换为 Open -FOAM 可用格式
star4ToFoam	把 STAR-CD(v4)的 PROSTAR 网格转换为 Open -FOAM 可用格式
tetgenToFoam	把 tetgen 生成的.ele, .node, .face 文件转换为 OpenFOAM 可用格式
vtkUnstructuredToFoam	把 paraview 生成的.vtk 文件转换为 Open -FOAM 可用格式
writeMeshObj	网格 debug 程序, 把网格写为三个独立的 OBJ 格式文件并可用 paraview 打开
网格处理	
attachMesh	将拓扑分离的网格连接
autoPatch	依据特征角把外部网格面切分为 patch
checkMesh	检查网格
createBaffles	把内部面转化为边界面。和 mergeOrSplitBaffles 不同, 其不复制点
createPatch	在现有的网格面上 (从存在的 patch 上或者 faceSet 面上) 创建 patch
deformedGeom	依据速度场通过一个放大因子对网格进行变形
flattenMesh	把一个 2D 网格的前后面平面化
insideCells	选择网格中心处于一个面内的网格, 这个面必须是封闭的
mergeMeshes	将两套网格合并
mergeOrSplitBaffles	检测共享点的面 (例如挡板), 合并他们并复制
mirrorMesh	针对一个平面做网格镜像
moveDynamicmesh	网格拓扑运动运行工具
moveEngineMesh	用于发动机计算的动网格运行工具
moveMesh	动网格运行工具
objToVtk	读取 obj 的线并转化为 vtk 格式

orientFaceZone	修正面域方向
polyDualMesh	创建多面体网格，保留特征面和边界面
refineMesh	在多个方向细化网格
rotateMesh	把网格从 n1 方向旋转到 n2 方向
setSet	对 cell/face/point/set 进行操作
setsToZones	把 pointSets/faceSets/cellSets 转化为 pointZones/faceZones -/cellZones
singleCellMesh	读取场数据并把他们投影到一个移除内部场的网格上 (singleCellFvMesh)，这个网格被写入为 singleMesh。主要用来为边界层生成网格以及场数据。本工具很容易产生坏网格，因此一般只用于在 paraview 中可视化
splitMesh	使用 attachDetach 把内部面外部化
splitMeshRegions	切分多域网格
stitchMesh	网格缝合
subsetMesh	基于 cellSet 选择网格
topoSet	通过字典文件对 cellSets/faceSets/pointSets 进行划分
transformPoints	对网格点的位置坐标进行操作，例如缩放、旋转等
zipUpMesh	读取网格并缝合网格单元，确保有效的多面体网格单元封闭
其他网格工具	
autoRefineMesh	细化表面附近网格
collapseEdges	短边坍塌，把一条线上的边结合
combinePatchFaces	检查网格上的多个面并把他们结合。多余的面可能产生于移除相邻网格单元的过程中（留下宿主网格单元的 4 个面）
modifyMesh	操作网格单元
PDRMesh	用于 PDR 模拟的网格及场操作
refineHexMesh	对六面体进行 2*2*2 细化
refinementLevel	在进行表面贴合之前，对细化等级等指标进行输出
refineWallLayer	对 patch 附近的网格进行细化
removeFaces	移除网格单元的面
selectCells	选择和面相关的网格单元
splitCells	使用平面切割网格单元
图形后处理	
ensightFoamReader	直接读取 OpenFOAM 数据的 EnSight 库
数据转换后处理	
foamDataToFluent	转换 OpenFOAM 数据为 Fluent 可用格式

foamToEnsight	转换 OpenFOAM 数据为 EnSight 可用格式
foamToEnsightParts	转换 OpenFOAM 数据为 EnSight 可用格式并为某个网格区域和 patch 创建 Ensight 部件
foamToGMV	转换 OpenFOAM 数据为 GMV 可用格式
foamToTecplot360	Tecplot 二进制文件生成器
foamToVTK	VTK 文件生成器
smapToFoam	将 STAR-CD SMAP 程序转换为 OpenFOAM 可用格式
速度场后处理	
Co	通过 phi 计算库郎数 (volScalarField)
enstrophy	计算速度涡能
flowType	计算速度流型
lambda2	λ_2 判据: 速度梯度张量的对称部分和反对称部分平方和的第二大特征值
Mach	计算每个时间步的局部马赫数
Pe	通过 phi 计算 Pe 数 (surfaceScalarField)
Q	Q 判据: 速度梯度张量的第二不变量
streamFunction	计算速度的流函数
uprime	计算 $\sqrt{2k/3}$ 标量场
vorticity	计算速度涡量
应力场后处理	
stressComponents	计算 sigma 应力张量的六个分量
标量场后处理	
pPrime2	计算每个时间步的 pPrime2: $(p - \bar{p})^2$
壁面后处理	
wallGradU	计算壁面的速度梯度
wallHeatFlux	计算所有 patch 的热通量 (volScalarField) 并输出所有壁面的通量积分
wallShearStress	在使用 RAS 湍流模型的时候, 计算剪切应力
yPlusLES	在使用 LES 湍流模型的时候, 计算并输出 yPlus
yPlusRAS	在使用 RAS 湍流模型的时候, 计算并输出 yPlus
湍流后处理	
createTurbulenceFields	创建湍流场

R	计算雷诺应力 R
patch 数据后处理	
patchAverage	对指定 patch 计算特定场的平均
patchIntegrate	对指定 patch 计算特定场的积分
拉格朗日后处理	
particleTracks	对于使用拉格朗日粒子云的算例生成 VTK 文件
steadyParticleTracks	对于使用稳态粒子云的算例（重组后）的粒子生成 VTK 文件
提取数据后处理	
probeLocations	探针位置
sample	使用特定的插值格式、提取选项、数据格式来提取数据
其它后处理工具	
dsmcFieldsCalc	从 DSMC 的广延场(extensive fields)计算强度场(intensive fields: 速度、温度)
engineCompRatio	计算几何压缩比。如果存在阀体等附加体积本工具有时候不会成功运行
execFlowFunctionObjects	在写入的时间步内执行 controlDict (在 system 文件夹下)文件中的 function 函数
foamCalc	指定时间步的简单数学操作
foamListTimes	通过时间选择器列出时间步
pdfPlot	读取指定字典的概率分布函数图片生成器
postChannel	对于槽道流的后处理工具
ptot	计算每个时间步总压
temporalInterpolate	在不同的时间步之间进行场插值，可用于动画计算
wdot	计算 wdot (用于 PDRFoam)
writeCellCentres	以 volScalarFields 的形式写入网格单元中心的三个分量，可用于阈值后处理
几何处理工具	
surfaceAdd	通过点融合操作来添加两个面，不检查三角形的重叠与否
surfaceAutoPatch	通过特征角对表面进行分片，类似 autoPatch
surfaceBooleanFeatures	通过对两个交界面进行 boolean 操作生成 extendedFeature-EdgeMesh
surfaceCheck	检查面几何和拓扑

surfaceClean	移除挡板，小边坍塌，移除三角形，通过切分边、通过投影点把微小的三角形转化，几何清理
surfaceCoarsen	使用 bunnylod 方法进行表面糙化
surfaceConvert	转换面网格格式
surfaceFeatureExtract	面特征抽取并写入为 edgeMesh 格式
surfaceFeatureConvert	转换 edgeMesh 格式
surfaceFind	寻找最近的面和顶点
surfaceHookUp	把最近的开放边缝合到附近的面
surfaceInertia	计算指定几何（刚体或者薄层）的惯性张量、主轴和矩。
surfaceLambdaMuSmooth	使用 lambda/mu 方法来平滑表面，需要设置 lambda 的松弛因子以及将 mu 的松弛因子设为 0
surfaceMeshConvert	转换几何表面格式，可以进行放大或者旋转等操作
surfaceMeshConvertTesting	转换几何表面格式，主要用于测试函数
surfaceMeshExport	将 surfMesh 输出为第三方格式，可以进行放大或者旋转等操作
surfaceMeshImport	将第三方格式输出为 surfMesh，可以进行放大或者旋转等操作
surfaceMeshTriangulate	从 polyMesh 生成由三角形构成的几何面。依据输出格式进行三角形切分。三角形的数量和 polyMesh 中的 patch 数量一致，也可以选择性的生成几何
surfaceOrient	面法向调整（面法向和用户提供的外部点一致，也可以为内部点，需要使用 -inside 命令参数）
surfacePointMerge	把面中某距离内的点进行融合，注意其为绝对距离
surfaceRedistributePatch	几何重新分布（对已经分解的表面或者没有分解的表面进行操作），以使得三角面和网格重叠
surfaceRefineRedGreen	将三角形的三个边进行切分（red 细化），详见：(A review of a posteriori error estimation and adaptive mesh refinement techniques”， Wiley-Teubner, 1996)
surfaceSplitByPatch	依据 patch 名称将几何文件分割
surfaceSplitByTopology	切分挡板几何为三角面
surfaceSplitNonManifolds	通过复制点来切分边，写入 borderEdge（一个边连接 4 个面）、borderPoint（一个点和两个 borderEdge 相连）、borderLine（borderEdges 的列表）
surfaceSubset	面几何分析工具，依据 subsetMesh 建立感兴趣的一部分子几何
surfaceToPatch	读取面几何并将面域应用到网格，困难的操作使用 boundaryMesh 处理
surfaceTransformPoints	和 transformPoints 一样，用来旋转面几何

并行后处理	
decomposePar	为 OpenFOAM 并行运行自动分解网格以及场
redistributePar	对当前分解的网格和场通过 decomposeParDict 进行重新分配
reconstructParMesh	对网格进行重组
热物理相关工具	
adiabaticFlameT	对给定燃料在一些列的未燃温度和等值比下计算绝热火焰温度
chemkinToFoam	将 CHEMKIN3 热力反应数据转换为 OpenFOAM 格式
equilibriumCO	计算一氧化碳的平衡等级
equilibriumFlameT	对给定燃料在一些列的未燃温度和等值比下计算平衡火焰温度，需要考虑氧气、水、二氧化碳的分离性
mixtureAdiabaticFlameT	指定温度指定混合率下计算绝热火焰温度
其它程序	
expandDictionary	读取某个字典文件并输出，附带输出宏替换信息
foamDebugSwitches	列出调试开关列表
foamFormatConvert	读取 controlDict 文件中的格式设置，将所有输出场的类型进行转换
foamHelp	输出某个程序或工具的帮助信息
foamInfoExec	查询和打印信息输出到标准输出
patchSummary	输出算例时间步下计算之后的场信息、边界信息

表 3.5 标准工具

3.7 标准库

OpenFOAM 附带的标准库文件位于\$FOAM_LIB/\$WM_OPTIONS 文件夹中，可以使用 lib 来快速切换到相应的目录。再次，所有的库文件都以前缀 lib 来命名，并且具有描述性，例如 incompressibleTransportModels 就是不可压缩传递模型库，为了简约性，库文件分为两类：

- 均一模型库：均一的类库以及子程序⁴⁰，参见表 3.7；
- 模型库：连续介质中的模型库文件，参见表 3.8，3.9，3.10；

⁴⁰ 作者的意思是针对所有求解器而写的库

OpenFOAM 的基本库——OpenFOAM	
algorithms	算法
containers	容器类
db	数据类
dimensionedTypes	dimensioned<Type>以及衍生类
dimensionSet	dimensionSet 类
fields	场类
global	全局设置
graph	图类
interpolations	插值格式
matrices	矩阵类
memory	内存管理工具
meshes	网格类
primitives	原始类

有限体积库——finite Volume	
cfdTools	CFD 工具
fields	体积、表面、patch、边界条件类
finiteVolume	有限体积离散
fvMatrices	矩阵（有限体积离散）
fvMesh	网格（有限体积离散）
interpolation	场插值类以及投影类
surfaceMesh	网格面数据（有限体积离散）
volMesh	网格体数据（有限体积离散）

后处理库	
cloudFunctionObjects	输出拉格朗日粒子云信息并写入文件的库
fieldFunctionObjects	场函数库，包括场的平均、最大、最小等操作
foamCalcFunctions	foamCalc 程序库
forces	对力（升力、曳力等）进行后处理的函数库
FVFuntionObjects	fvcDiv, fvcGrad 函数库
jobControl	控制函数对象的任务工具
postCalc	后处理操作函数库
sampling	在计算域进行提取场的工具
systemCall	运行算例时候的系统调用接口
utilityFunctionObjects	程序函数对象

求解以及网格操作库	
autoMesh	snappyHexMesh 工具库
blockMesh	blockMesh 工具库
dynamicMesh	求解动网格库
dynamicFvMesh	有限体积法下包含拓扑改变的动网格库
edgeMesh	基于边的网格描述库
fvMotionSolvers	有限体积网格运动求解器
ODE	偏微分方程求解器
meshTools	OpenFOAM 网格处理工具
surfMesh	不同格式的面网格处理库
triSurface	标准三角面几何处理库
topoChangerFvMes h	拓扑变形程序库
拉格朗日粒子跟踪库	
coalCombustion	煤粉燃烧模型
distributionModels	粒子分布函数模型
dsmc	蒙特卡洛直接模拟
lagrangian	基本拉格朗日（粒子跟踪）求解格式
lagrangianIntermediate	粒子跟踪动力学、热动力、多组分反应、粒子力模型
potential	用于分子模拟的分子势能
molecule	分子动力学类
molecularMeasurements	分子动力学测量类
solidParticle	固体粒子植入
spray	喷雾注入模型
turbulence	基于湍流的粒子分散以及布朗运动模型
其他库	
conversion	网格和数据转换工具
decompositionMethods	计算域分解工具
engine	发动机计算工具
fileFormats	第三方包写入输出规则
genericFvPatchField	均一的 patch 场
MGridGenGAMGAgglomeration	使用 MGridGen 进行网格融合（agglomeration）库
pairPatchAgglomera	基本的 patch 对融合（agglomeration）方法

tion	
OSspecific	操作系统特定函数
randomProcesses	生成和分析随机进程的工具
并行库	
decompose	网格以及场分解库
distributed	分布面的寻找以及输入输出工具
metisDecomp	Metis 分解库
reconstruct	网格以及场重组库
scotchDecomp	Scotch 分解库
ptscotchDecomp	PTScotch 分解库

表 3.7 一般用途的共享库

基本热物理模型——basicThermophysicalModels	
hePsiThermo	基于可压缩性 φ 的均一热物理模型计算
heRhoThermo	基于密度 ρ 的均一热物理模型计算
pureMixture	用于计算气体混合的均一热物理模型
反应模型——reactionThermophysicalModels	
psiReactionThermo	基于 φ 的混合燃烧焓值计算
psiuReactionThermo	基于 φ_u 的混合燃烧焓值计算
rhoReactionThermo	基于 ρ 的混合燃烧焓值计算
heheupsiReactionThermo	计算未燃气体和混燃的焓值
homogeneousMixture	基于燃料标准化质量分数 b 的混合燃烧
inhomogeneousMixture	基于 b 和燃料总质量分数 f_t 的混合燃烧
veryInhomogeneousMixture	基于 b , f_t 和未燃烧燃料质量分数 f_u 的混合燃烧
basicMultiComponentMixture	多组分基本混合
multiComponentMixture	多组分衍生混合
reactingMixture	基于化学反应动力学模型的混合燃烧
egrMixture	废气再循环混合
singleStepReactingMixture	单步反应混合

辐射模型——radiationModels	
P1	P1 模型
fvDOM	有限体积离散坐标法
opaqueSolid	黑体辐射：能量源项为 0 同时创建 adsorptionEmissionModel 和 scatterModel
viewFactor	角系数辐射模型
层流火焰速度模型——laminarFlameSpeedModels	
constant	层流火焰常速度模型
GuldersLaminarFlameSpeed	层流火焰 Gulder 速度模型
GuldersEGRLaminarFlameSpeed	附加废气循环混合的层流火焰 Gulder 速度模型
RaviPetersen	附带 Ravi 和 Peterson 修正的层流火焰速度模型
正压模型——barotropicCompressibilityModels	
linear	线性可压缩模型
Chung	Chung 可压缩模型
Wallis	Wallis 可压缩模型
类气态组分热物理特性——specie	
adiabaticPerfectFluid	绝热理想气体状态方程
icoPolynomial	不可压缩多项式状态方程（例如对于液体）
perfectFluid	理想气体状态方程
incomprehensiblePerfectFluid	使用参考压力的不可压缩气体状态方程。密度仅仅和温度、组分有关
rhoConst	常量密度状态方程
eConstThermo	常热导率模型，内能 e、熵 s 由此计算
hConstThermo	常热导率模型，内能 e、焓 h 由此计算
hPolynomialThermo	从附带一系列参数的多项式来计算 C_p ，进而用于计算 h 和 s
janafThermo	从使用 JANAF 热力表的函数计算 C_p ，进而用于计算 h 和 s
specieThermo	从 c_p, h, s 得到的组分热力特性
constTransport	常传递特性
polynomialTransport	基于温度传递的多项式方程
sutherlandTransport	基于温度传递的 sutherland 方程

热物理特性表	
NSRDSfunctions	美国标准参考数据系统（NSRDS）美国化学工程协会数据系统
APIfunctions	美国石油协会制定的关于蒸汽质量扩散表
化学模型	
chemistryModel	化学反应模型
chemistriSolver	化学反应求解器
其他库	
liquidProperties	液体的热物理特性
liquidMixtureProperties	液体混合的热物理特性
basicSolidThermo	固体的热物理模型
hExponentialThermo	状态方程模化的指数热力模型包
SLGThermo	固体、液体、气体热力模型
solidChemistryModel	固体化学（分解）热力模型
solidProperties	固体热力模型
solidMixtureProperties	固体混合热力模型
solidSpecie	固体反应速率和传递模型
solidThermo	固体能量模型

表 3.8 热物理模型库

不可压缩 RAS 湍流模型——incomprehensibleRASModels	
laminar	层流模型（设置后湍流不参加计算）
kEpsilon	标准高雷诺数 $k - \varepsilon$ 模型
kOmega	标准高雷诺数 $k - \omega$ 模型
kOmegaSST	$k - \omega$ SST 模型
RNGkEpsilon	重整化 $k - \varepsilon$ 模型
NonlinearKEShih	非线性 Shih $k - \varepsilon$ 模型
LienCubicKE	立方 Lien $k - \varepsilon$ 模型
qZeta	$q - \zeta$ 模型

kkOmega	不可压缩低雷诺数 $k - kl - \omega$ 模型
LaunderGibsonRS TM	附带壁面反射项的 Launder&GibsonRST 模型
realizableKE	可实现 $k - \varepsilon$ 模型
SpalartAllmaras	Spalart&Allmaras 一方程混合长模型
v2f	不可压缩 Lien&Kalitzin v2f 湍流模型
可压缩 RAS 湍流模型——comprehensibleRASModels	
laminar	层流模型（设置后湍流不参加计算）
kEpsilon	标准高雷诺数 $k - \varepsilon$ 模型
kOmegaSST	$k - \omega SST$ 模型
RNGkEpsilon	重整化 $k - \varepsilon$ 模型
launderSharmaKE	地雷诺数 Launder&Sharmak $- \varepsilon$ 模型
LRR	Launder&Reece&Rodi RST 模型
LaunderGibsonRS TM	Launder&Gibson RST 模型
realizableKE	可实现 $k - \varepsilon$ 模型
SpalartAllmaras	Spalart&Allmaras 一方程混合长模型
v2f	可压缩 Lien&Kalitzin v2f 湍流模型
LES 过滤器——LESfilters	
laplaceFilter	laplace 过滤
simpleFilter	Simple 过滤
anisotropicFilter	Anisotropic 过滤
LES deltas ——LESdeltas	
PrandtlDelta	普朗特 delta
cubeRootVolDelta	网格单元立方根 delta
maxDeltaxyz	(仅适用于结构六面体网格) x、y、z 最大
smoothDelta	delta 光顺
不可压缩 LES 湍流模型——incomprehensibleLESModels	
Smagorinsky	Smagorinsky 模型
Smagorinsky2	3D 滤波的 Smagorinsky 模型
homogenousDynSmagorinsky	均一动态 Smagorinsky 模型
dynLagrangian	拉格朗日两方程涡粘模型
spectEddyVisc	谱空间涡粘模型
LRDDiffStress	LRR 微分应力模型

DeardorffDiffStress	Deardorff 微分应力模型
SpalartAllmaras	Spalart&Allmaras 模型
SpalartAllmarasDES	Spalart&Allmaras 分离涡模型
SpalartAllmarasIDDES	Spalart&Allmaras 分离涡模型加强型
vanDriestDelta	不可压缩 LES 中的网格单元立方根 delta
可压缩 LES 湍流模型——compressibleLESModels	
Smagorinsky	Smagorinsky 模型
oneEqEddy	k 方程涡粘模型
lowReOneEqEddy	低雷诺数 k 方程涡粘模型
homogenousDynOneEqEddy	不可压缩一方程涡粘模型
DeardorffDiffStress	Deardorff 微分应力模型
SpalartAllmaras	Spalart&Allmaras 一方程混合长模型
vanDriestDelta	不可压缩 LES 中的网格单元立方根 delta

表 3.9 RAS、LES 湍流模型库

不可压缩传递模型——incompressibleTransportModels	
Newtonian	牛顿粘性模型
crossPowerLaw	非牛顿 Cross 粘性模型
BirdCarreau	非牛顿 Bird-Carreau 模型
HerschelBulkley	非牛顿 Herschel-Bulkley 模型
powerLaw	非牛顿幂律粘性模型
interfaceProperties	界面模型，例如多相流中的接触角模型
其他传递模型库	
interfaceProperties	界面特性计算
twoPhaseProperties	两相混合模型（包括边界条件）
surfaceFilmModels	表面膜模型

表 3.10 传递模型

第四章

OpenFOAM 算例

这一章我们分析 OpenFOAM 算例的文件结构。一般来讲，用户需要对每个算例命名，例如顶盖驱动流我们可以称之为 cavity。Cavity 目录就是这个算例所有文件存储的地方。算例文件放在什么地方都可以，但是我们建议把它们放在 run 文件夹内，例如：第二章提到的 \$HOME/OpenFOAM/\${USER}-2.3.0 文件夹内。这样放置算例的一个优点是，\$FOAM_RUN 环境变量默认设置为 \$HOME/OpenFOAM/\${USER}-2.3.0/run；因此用户可以快速的使用 run 命令切换到这个文件夹。

OpenFOAM 的典型文件结构，可参考其附带的各种算例，他们位于 \$FOAM_TUTORIAL-S 文件夹下，可以执行 tut 命令快速的切换到此文件夹。当读者在阅读这个章节的时候你可以顺便看看这些算例的文件夹结构。

4.1 OpenFOAM 文件结构

OpenFOAM 算例的基本文件结构包含了所必须的文件集。如图 4.1 所示：

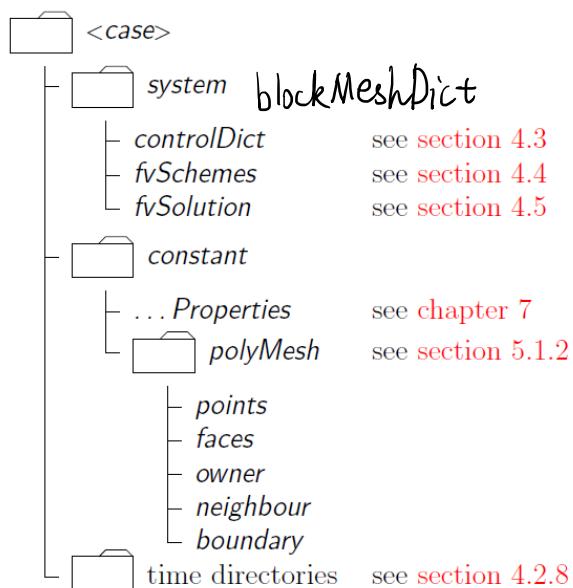


图 4.1 算例文件结构

- constant 文件夹下包含了程序所需要的物理特性文件例如 transportProperties，constant 下的 polyMesh 文件夹包含了网格数据文件。
- system 文件夹主要用于设置求解算法的参数。它至少包含三个文件：controlDict：用于控制求解开始/终止时间，时间步，以及输出数据参数；fvSchemes 包含各种离散格式；fvSolution 包含矩阵求解器设置、残差、以及其他算法控制。
- Time 一系列文件夹包含某些场的计算结果。其中包括必须指定的初始数据以及边界条件，以及 OpenFOAM 求解器的计算结果。值得一提的是即使在稳态问题中（稳态问题只需要边界条件），OpenFOAM 的初始场也必须被初始化。每个时间步的名称由模拟结果决定，这在 4.3 节有详细说明。由于我们经常在 0 时刻进行我们的模拟，因此初始条件通常存储在 0 文件夹或者 0.00000e+00 中（取决于指定的文件名格式）。例如，cavity 算例中，速度场和压力场存储在 0/U 和 0/p 文件夹中。

4.2 基本输入输出格式

OpenFOAM 需要读取一系列数据，例如字符串、标量、矢量、张量、列表和场。OpenFOAM 输入输出机制是很灵活的，用户可以对其进行修改。不同于其它软件，它们的输入输出系统很难理解，并且还不对外公布。OpenFOAM 的输入输出机制遵守一些简单的规则，它们是非常容易理解的。下面我们来阐述 OpenFOAM 的文件格式。

4.2.1 通用语法规则

遵循 C++的一些基本原则：

- 文件形式任意，书写自由，不需要对列整齐，其会忽略分行信息⁴¹
- // 这个符号意味着 OpenFOAM 将会忽略这一行的内容直到行尾。
- 多行的注释由/*.....*/来完成。

4.2.2 字典

OpenFOAM 使用字典来输入数据。字典文件可以依靠关键词 keywords 来进行输入和输出(I/O)，关键词遵循如下形式：

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

大多数关键词只有一个数据：

⁴¹ 单行写入代码和多行写入代码是一样的，参见 107 页 4.2.5 节一个例子

<keyword> <dataEntry>

某些 OpenFOAM 的字典文件本身又包含了一系列带有关键词的子字典。字典的构建相当具有逻辑性和继承性，即字典本身也可以包含一个或者更多的子字典。字典的形式是在字典名称之后用 { } 把字典内的关键词信息包含进来，例如：



```

<dictionaryName>
{
    ... keyword entries
}

```

4.2.3 头文件

所有 OpenFOAM 读写的文件都以 FoamFile 开头，它包含一系列关键词信息，见表 4.1。此表对其进行了简单的描述，表中我们提供了简要的可供关键词使用的信息 (entry)，他们适

关键词	描述	信息
version	输入输出格式版本号	2.0
format	数据格式	ascii/binary
location	文件路径，“...”	可选
class	OpenFOAM 文件构建的类	一般为 dictionary 或者场文件例如 volVectorField
object	文件名	例如 controlDict

表 4.1 文件头信息

用于大部分情况。class 关键词中的信息就是表明这个文件是构建一个类。如果用户不知道那代码中的那部分来调取这个文件，也不熟悉 OpenFOAM 的类，那它可能不知道这个类名。然而大多数需要指定关键词的数据文件都通过 dictionary 类来表示，因此大部分情况下 class 关键词的信息为 dictionary⁴²。

下面这个例子我们来介绍关键词用法，它告诉用户如何用关键词为算例提供数据。fvSolution 字典文件包含两个子字典，solvers 和 PISO。solvers 字典包含了求解器需要的大量数据，以及压力和速度方程需要的残差（他们在 p 和 U 关键词里面指定）。PISO 字典包含了算法控制信息。

```

17
18 solvers
19 {
20     p
21     {
22         solver          PCG;
23         preconditioner DIC;
24         tolerance      1e-06;
25         relTol          0;
26     }
27
28     U

```

⁴² 对于数据场通常为 vol<type>Field

```

29  {
30      solver          smoothSolver;
31      smoother        symGaussSeidel;
32      tolerance       1e-05;
33      relTol          0;
34  }
35 }
36
37 PISO
38 {
39     nCorrectors      2;
40     nNonOrthogonalCorrectors 0;
41     pRefCell         0;
42     pRefValue        0;
43 }
44
45
46 // ****

```

4.2.4 链表

OpenFOAM 的程序包含很多链表。例如描述网格坐标的链表。链表通常在输入输出的文件中以特有的形式存在，通常由（）包含。在链表的（）之前也需要遵守一定的格式：

- 单一链表：关键词下面尾随着圆括号

```

<listName>
(
    ...
);

```

- 复合链表：关键词后面由<n>指定链表数

```

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

```

- 占位标识：

```

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

```

值得注意的是 list<scalar>里面的<scalar>不是一个大的类名（例如 scalar, double 等），而是一个特定的数。

单一链表用起来非常方便。其它形式运行起来更快，因为链表的大小在读取数据之前就分配了内存。因此，小链表通常使用单一链表的形式，这样读取时间最小。长链表建议使用

其它形式。

4.2.5 Scalar 标量、Vector 矢量、Tensor 张量

scalar 标量就是一个数。vector 矢量是一个 3 维 1 阶的矢量空间 vectorSpace。由于维数固定为 3。通常我们使用最简单的链表形式。因此，一个矢量(1.0, 1.1, 1.2)写成如下形式：

$$(1.0 \ 1.1 \ 1.2)$$

在 OpenFOAM 中， tensor 张量是一个 3 维 2 阶量，因此一个张量有 9 个实数。它通常写成如下形式：

$$\begin{pmatrix} & \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

下面这个例子表明 OpenFOAM 会自动忽略行信息，所以上面这个张量写成单行形式也是一样的：

$$(1 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 1)$$

4.2.6 量纲

在连续介质里，物理量都具有量纲。例如质量：kg，体积：m³，压力：Pa。这些物理量的代数几何操作要求量纲保持一致。对于同样量纲的物理量，只有加、减、等于是具有物理意义的。为了防止对物理量进行无意义或者错误的运算，OpenFOAM 将量纲依附于场数据以及某个物理量，在对其进行张量操作的时候，执行量纲检查。dimensionSet 的输入输出格式设置为方括号内有限的 7 个标量。例如：

$$[0 \ 2 \ -1 \ 0 \ 0 \ 0 \ 0]$$

编号	名称	SI 单位	USCS 单位
1	质量	kg	lbm
2	米	m	ft
3	时间	s	s
4	开尔文	K	°R
5	摩尔质量	kgmol	lbmol
6	电流	A	A
7	光强	cd	cd

表 4.2 SI 和 USCS 标准量纲

量纲括号[]内的每个值代表每个物理单位的幂, 见表 4.2. 这个表是采取 SI 单位制以及 USCS 单位制, 但 OpenFOAM 可以使用任何量纲单位制。你所需要做的就是确保输入数据准确。另外, OpenFOAM 有时需要一些带有量纲的常数。例如热物理模型中的热力学常数 R 。它的单位已经在 OpenFOAM 安装目录 (\$WM_PROJECT_DIR/etc/controlDict) 下 controlDict 的 DimensionedConstant 子字典下指定。它默认使用 SI 单位制。想使用 USCS 单位或者其它单位制的应该修改这个文件进行单位匹配。

4.2.7 单位类型

物理量要依附于量纲。接口以如下形式表示, 以 dimensionedScalar 为例:

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

第一个 nu 是关键字, 第二个 nu 是存储在 word 类中的 word 名, 通常和第一个关键字一样, 下一个带中括号的是 dimensionSet, 最后一个是 nu 的值。

4.2.8 场

大多数 OpenFOAM 输入输出的数据是张量场, 例如速度、压力, 它们会写入时间步文件。OpenFOAM 采用下列关键词来写入数据, 见表 4.3:

关键词	描述	举例
量纲	场的量纲	[1 1 -2 0 0 0]
内部场	内部场的值	uniform (1 0 0)
边界场	边界场	参见 4.2.8 节的链表

表 4.3 场文件内的主要关键词

数据首先以 dimensions 为接口, 内部场尾随其后。它有两种类型:

- **均一场:** 场内所有网格内被指定了一个值, 采用如下形式表示:

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

- **非均一场:** 每个网格单元有不同的值, 采取如下形式表示:

```
internalField nonuniform <Lists>;
```

边界场 boundaryField 包含了一系列 patch (面), 它们和 polyMesh 文件夹下 boundary 文件里的 patch 相对应。每个 patch 是一个字典文件, 其中包含了一系列关键词。type 后面必须要指定这个边界的边界条件类型。

在确定边界类型之后，可以指定具体的边界信息作为初始条件。表 5.3 列举了 OpenFOAM 可用的边界条件，表 5.4 描述了这些边界条件必须指定的内容。下面这个速度场字典可以作为一个例子来说明：

```

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

```

4.2.9 指令和宏替换

OpenFOAM 在设置算例的过程中，可以使用指令和宏替换，他们具有很大的灵活性。指令就是在文件内以 # 开头的命令。宏替换是以 \$ 开头的命令。目前 OpenFOAM 有四种指令：

- `#include "<fileName>"` (或`#includeIfPresent "<fileName>"`) 它读取`<fileName>`文件
- `#inputMode` 有两个选择：`merge`，它会使调用这个关键词的字典的关键词处使用这个值，因此如果在一个地方把关键词进行指定，如果对这个关键词进行修改，其他所有调用该关键词的信息都被修改；`overwrite`，对整个字典文件进行重写；一般情况下我们使用 `merge`。
- `#remove <keywordEntry>` 将关键词信息去掉；可以用一个单词或者一个表达式来表述；
- `#codeStream` 后面既是 C++ 语言，可以进行实时编译、调用、执行。

4.2.10 `#include` 和 `#inputMode` 指令

举例，用户想在边界处为内部压力场初始化。我们可以创立一个文件叫做 `initialConditions`，包含以下信息：

```

pressure 1e+5;
#inputMode merge

```

为了让内部场和初始场都使用这个压力值，用户可以简单的把上面的压力场替换到下面的地方：

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

这个非常小的案例表明了如何使用这个工具。然而，这个程序的功能很强大。例如，如果用户有一系列算例，每个算例文件都要设置同样的湍流条件，创建一个这样的单一文件并把它们包含在每个算例 RASProperties 文件里，将会非常方便。这个功能的作用不仅仅是设置同样的值，边界条件也可以采用类似的方法来设定。可以说这个小代码的功能很强大。

4.2.11 #codeStream 指令

#codeStream 指令包含的相关字典信息可以用来当做标准的 C++ 代码来编译和执行。它需要指定以下关键词：

- **code:** 指定代码部分。在代码中可以使用 OStream& os 进行输出或者通过 const dictionary& dict 调用字典。例如：这可用于调用当前算例的信息；
- **codeInclude**（可选）：用#include 包含指定的 C++ 代码
- **codeOptions**（可选）：指定附加头文件路径，参考 Make/options 的 EXE_LNC 中的格式
- **codeLibs**（可选）：指定附加库。参考 Make/options 的 LIB_LIBS 中的格式

代码可以用#{...#}来包含起来，并可以多行书写。

下面是一个#codeStream 的例子，在 controlDict 字典文件里面的这个代码表示它将寻找起始时间和终止时间然后做一个简单的运算并输出。

```
startTime 0;
endTime 100;
...
writeInterval #codeStream
{
    code
```

```

#{

    scalar start = readScalar(dict.lookup("startTime"));
    scalar end = readScalar(dict.lookup("endTime"));
    label nDumps = 5;
    os << ((end - start)/nDumps);

};

}

```

4.3 时间和输入输出控制

在使用 OpenFOAM 进行运行之前，应该首先对数据的输入输出形式进行设定。OpenFOAM 依据这个数据中心的设定来输入输出数据。由于一般在写数据的时候都是在运行的时候，时间设定是在所难免的。controlDict 字典文件内包含了数据输入输出的关键信息。表 4.4 列举了 controlDict 常见的关键词信息。只有时间控制和 writeInterval 部分是必须存在的，表 4.4 中的默认值后面附带了+号。其它可以省略的关键词没有标注默认值。

时间控制	
startFrom	控制模拟的起始时间
-firstTime ⁴³	时间步文件夹中的最开始的时间步文件夹
-startTime	依据 startTime 关键词信息开始计算
-latestTime	从最新的时间步开始计算
startTime	如果把 startFrom 设置为 startTime，此处指明计算的开始时间步
stopAt	控制模拟的结束时间
-endTime	依据 endTime 关键词信息结束运算
-writeNow	在当前的时间步进行完整的一次运算然后写入数据
-noWriteNow	在当前的时间步进行完整的一次运算但是不写入数据
-nextWrite	依据 writeControl 设置的时间周期，在下一个时间步写入后停止运算
endTime	如果把 stopAt 设置为 endTime， endTime 需要指定为结束的时间步
deltaT	计算的时间步长
数据写入	
writeControl	控制输出
-timeStep	每 writeInterval 时间步写一次数据
-runTime	每 writeInterval 时间间隔（秒）写一次数据
-adjustableRunTime	每 writeInterval 时间间隔（秒）写一次数据，必要的

⁴³ 在设置中请去除“-”号

	时候调整时间步以使得在 writeInterval 的物理时间点写入数据，通常在自动调整时间步的算例中使用
-cpuTime	每 writeIntervalCPU 时间（秒）写一次数据
-clockTime	每 writeInterval 运算时间（秒）写一次数据
writeInterval	和 writeControl 结合使用的一个标量
purgeWrite	一种循环写入数据的方式，其值代表循环时间步数。例如如果我们的起始时间为 5s，每个时间步设定为 1s 一写，purgeWrite 设置为 2，那么在计算完 6s，7s 的数据的时候写入，在计算完 8s，9s 的数据后覆盖写入之前的 6s，7s 文件夹。 如果不使用此功能，可以设置为 purgeWrite 0；对于稳态计算，我们可以设置 purgeWrite 为 1；
writeFormat	指定数据格式
-ascii	ASCII 格式，由 writePrecision 控制有效数字位数
-binary	Binary 格式
writePrecision	和前文的 writeFormat 联合使用，默认为 6
-uncompressed	不压缩
-compressed	gzip 压缩格式
timeFormat	时间步文件夹的名称指定
-fixed	$\pm m. dddddd$, 其中 d 由 timePrecision 来控制
-scientific	$\pm m. dddddd \pm xx$, d 由 timePrecision 来控制
-general	使用 scientific 格式，默认小数点后有 4 位有效位数，也可通过 timePrecision 来调节
timePrecision	和前文的 timeFormat 联合使用，默认为 6
graphFormat	后处理程序写入的数据格式
-raw	ASCII 原始数据
-gnuplot	gnuplot 专用格式
-xmgr	Grace/xmgr 专用格式
-jplot	jPlot 专用格式
可调节时间步	
adjustTimeStep	设定 OpenFOAM 是否开启可调节时间步功能，可设置为 yes 或者 no
maxCo	最大库郎数，例如 0.5
数据读取	

runTimeModifiable	通过设置为 yes 或者 no, 让 OpenFOAM 在每个时间步读取或不读取所修改的字典文件, 例如 controlDict
运行时加载库	
libs	运行时需要加载的一系列的附加库 (位于\$LD_LIBRARY_PATH 路径下), 例如 ("libUser1.so" "libUser2.so")
functions	需要加载的函数, 例如 probes 函数, 参见具体算例其如何设置\$FOAM_TUTORIALS

表 4.4 controlDict 的关键词信息

controlDict 的一个范例参考如下:

```

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

```

4.4 离散格式

system 文件里面的 fvSchemes 用来设置离散格式, 凡是求解器里面出现的方程, 都需要在这里面设置离散格式。这一节主要阐述如何在 fvSchemes 里设置离散格式。

要在 fvScheme 里面指定的项有很多, 从梯度项到点插值都需要指定。OpenFOAM 开源代码的目的是给用户充分自由的选择权。例如, 一般情况下线性插值是非常有效的, 但 OpenFOAM 也提供了一系列的非线性插值格式供用户选择。

OpenFOAM 用户具有充分的自由选择各种格式的权利体现在导数项的离散上。首先, 用户有很多离散方法可以选择, 通常使用高斯积分定律进行体积分。高斯积分定律建立在把所有的面值加和的基础之上, 这就需要从节点值获得面值, 这个过程就是插值。用户有充分自由的选择怎样来进行插值。值得注意的是, 某些插值格式是专门为了某些导数而设计, 比如是对流项 $\nabla \cdot$ 。

`fvSchemes` 中需要指定的信息分成了以下小类，见表 4.5。进一步的，这个表中的关键词也是一个子字典，例如 `gradSchemes` 包含所有梯度项的格式，例如 `grad(p)`，代表 ∇p 。更多的例子请参考下面这个从 `fvSchemes` 字典文件中抽取的片段：

关键词	数学形式
<code>interpolationSchemes</code>	点对点插值格式
<code>snGradSchemes</code>	面法相梯度格式
<code>gradSchemes</code>	梯度项 ∇ 格式
<code>divSchemes</code>	对流项 $\nabla \cdot$ 格式
<code>laplacianSchemes</code>	拉普拉斯项 ∇^2 格式
<code>timeScheme</code>	时间项的一阶 $\frac{\partial}{\partial t}$ 、二阶 $\frac{\partial^2}{\partial^2 t}$ 导数格式
<code>fluxRequired</code>	需要用来计算通量的场

表 4.5 `fvScheme` 的主要关键词

```

17
18 ddtSchemes
19 {
20     default      Euler;
21 }
22
23 gradSchemes
24 {
25     default      Gauss linear;
26     grad(p)      Gauss linear;
27 }
28
29 divSchemes
30 {
31     default      none;
32     div(phi,U)   Gauss linear;
33 }
34
35 laplacianSchemes
36 {
37     default      Gauss linear orthogonal;
38 }
39
40 interpolationSchemes
41 {
42     default      linear;
43 }
44
45 snGradSchemes
46 {
47     default      orthogonal;
48 }
49
50 fluxRequired
51 {
52     default      no;
53     p ;
54 }
55
56
57 // ****

```

`fvSchemes` 字典文件包含了以下信息：

- ...Schemes 子字典包含了不同项需要指定的关键字，在每个子字典下面，包括 `default` 关键字；其它对应于这个求解器方程中出现的项，例如 `grad(p)` 代表 ∇p

- `fluxRequired` 子字典下的场意味着需要用这些场来组建通量，例如例子中的压力 p

如果...`Schemes` 子字典中的 `default` 关键词已经被指定，那么子字典中所有的项都会应用这个格式。例如，在 `gradScheme` 中为 `default` 指定某个格式，那么所有的 ∇p , ∇U 都会使用这个格式。

当 `default` 被指定的时候，就没有必要重复指定子字典的其它信息了。例如 ∇p , ∇U 等。然而，如果你继续指定它们的格式，那么它会覆盖默认格式并使用自定义格式。

用户也可以指定 `default` 为 `none`，这种情况下，用户必须强制为每个项指定离散格式。把 `default` 设置为 `none` 看起来没有必要并且很繁琐，因为 `default` 可以被覆盖。但是，指定 `none` 的话强制用户对方程的每个需要离散的项进行设定，这会提醒用户哪些项在求解器中有所呈现。下面的章节介绍表 4.5 提到可用的离散格式。

4.4.1 插值格式(interpolationSchemes)

`interpolationSchemes` 子字典包含了各种插值格式，主要是从网格中心对网格面的插值格式。OpenFOAM 可用的点到面插值格式在表 4.6 中列出，它分为四类：第一类为普通格式，其它三类主要和高斯法结合使用并主要用于对流项⁴⁴，详见 4.4.5 节。对于需要做普通插值的场，用户不太可能使用针对对流项的插值格式，但是所有格式我们在这里都有所提及，针对对流项的特定插值格式参阅 4.4.5 节。值得一提的是，像 UMIST 这种格式 OpenFOAM 也有，但我们建议使用表 4.6 列举的插值格式。

普通的插值格式可以这样指定，例如线性（`linear`）插值可以这样指定：

```
default linear
```

针对对流项的插值格式需要在通量的基础上计算插值。这些格式的指定需要知道这些通量的名字。在大多数求解器中，通量的名字是 `phi`，一般情况下它是一个 `surfaceScalarField`（面标量场），也即速度通量 ϕ 。本节推荐的三个针对对流项的插值格式为：普通对流格式、NVD 格式、TVD 格式。除了混合(blended)格式之外，普通对流格式和 TVD 格式由通量和插值格式来指定。例如建立在 `phi` 通量基础上的迎风 `upwind` 格式这样定义：

```
default upwind phi
```

一些 TVD 和 NVD 格式需要一个系数 φ ，其大于 0 小于 1， $\varphi=1$ 对应的是 TVD 标准格式，它能达到最好的收敛结果。 $\varphi=0$ 对应的是最好的精准性。通常我们建议 $\varphi=1$ 。例如一个建立在通量上 $\varphi=1$ 的 `limitedLinear` 格式这样指定：

```
default limitedLinear 1.0 phi
```

4.4.1.1 严格有界标量格式

有些标量场是严格有界的，一些具有限制器的增强版格式由此而生。用户如果想指定有界场

⁴⁴ 高斯积分定律需要把网格中心的值对面进行插值

的上下限，应该标注 limited，并且分别指定上下限的值。例如：为了将 vanLeer 格式在 -2, 3 之间有界。用户需要如下指定：

```
default limitedVanLeer -2.0 3.0
```

很多标量场都是在 0, 1 之间有界，因此有一些插值格式衍生了一些特定形式，例如可以通过在离散格式上添加 01 来实现。例如：将 vanLeer 格式限制在 0, 1 之间，我们可以指定：

```
default vanLeer01
```

下面几个格式可以设定为有界：limitedLinear, vanLeer, Gamma, limitedCubic, MUSCL 和 SuperBee。

4.4.1.2 针对矢量场的格式

一些格式专门为矢量场而设定，这些格式考虑了场方向。它可以通过在格式上添加 V 来实现，例如 limitedLinearV 就是 limitedLinear 的变种，V 版本在以下格式中有：limitedLinearV, vanLeerV, GammaV, limitedCubicV 和 SFCDV：

中心格式	
linear	中心插值
cubicCorrection	立方正交格式
midpoint	对称权重的中心插值
迎风对流格式	
upwind	迎风格式
linearUpwind	线性迎风格式
skewLinear	附带畸变修正的线性迎风格式
filteredLinear2	高频波过滤线性迎风格式
TVD 格式	
limitedLinear	有界线性格式
vanLeer	vanLeer 有界格式
MUSCL	MUSCL 有界格式
limitedCubic	立方正交有界格式
NVD 格式	
SFCD	自过滤中心格式
Gamma	Gamma 格式

表 4.6 插值格式

4.4.2 面法向梯度格式(snGradSchemes)

snGradSchemes 子字典包含了面法向梯度项。面法向梯度项在网格单元表面上计算，其为某个量在两个相连的网格中心单元处的值的面梯度分量⁴⁵。面法向梯度在进行指定的时候需要使用高斯积分来处理拉普拉斯项。

表 4.7 列举了可用的格式，用它们进行指定即可。其中有界的格式要求指定系数 φ ，对于大于 0 小于 1 的 φ 意味着：

$$\varphi = \begin{cases} 0 & \text{无修正} \\ 0.333 & \text{非矩形修正} \leq 0.5 \times \text{矩形修正}^{46} \\ 0.5 & \text{非矩形修正} \leq \text{矩形修正} \\ 1 & \text{完全修正} \end{cases} \quad (4.1)$$

如果我们打算把 default 指定为 limited 格式，并把 φ 设定为 0.5，即为：

```
default limited 0.5
```

格式	描述
corrected	显性矩形修正
uncorrected	无矩形修正
limited φ	有限的矩形修正
bounded	对正值标量的有界修正
fourth	四阶

表 4.7 面法相梯度格式

4.4.3 梯度格式(gradSchemes)

gradSchemes 子字典包含梯度项，表 4.8 列举了可以选择的离散格式。

格式	描述
Gauss <interpolationScheme>	二阶，高斯积分
leastSquares	二阶，最小二乘法
fourth	四阶，最小二乘法
cellLimited <gradScheme>	上述格式的网格单元有界版本
faceLimited <gradScheme>	上述格式的面有界版本

表 4.8 gradSchemes 可用的离散格式

⁴⁵ 面梯度和面法向的内积

⁴⁶ 意义不详，有好的意见请联系译者

leastSquares 和 fourth 格式简单的这样指定就可以：

```
grad(p) leastSquares
```

Gauss 关键词意味着采用高斯积分定律来进行标准有限体积离散，这需要把节点的值插值到面上。因此，指定 Gauss 的时候还需要用表 4.6 的格式进行从点到面的格式指定。大部分情况我们使用线性插值格式。使用其它格式的时候并不是很常见。例如：

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

表 4.8 的前三种 Gauss, leastSquares, fourth 可以通过 cellLimited 或者 faceLimited 限定，例如网格有界高斯格式为：

```
grad(p) cellLimited Gauss Linear 1
```

4.4.4 拉普拉斯格式(laplacianSchemes)

laplacianSchemes 子字典包含了拉普拉斯项，我们来讨论一下流体力学中的这个拉普拉斯项： $\nabla \cdot (\varrho \nabla U)$ ，它由文件头 `laplacian(nu,U)` 来指定。在我们的例子中，只有 Gauss 格式可选，并且需要选择为粘度选择的插值格式，以及面法向梯度格式 ∇U 的格式，如下：

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

interpolationScheme 在表 4.6 中选择，一般我们选择线性插值。snGradScheme 在表 4.7 中选择，他们对拉普拉斯项产生的数学影响见表 4.9。典型的拉普拉斯项可以如下指定：

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

格式	数学性质
corrected	无界、二阶、守恒
uncorrected	有界、一阶、非守恒
limited φ	corrected 和 uncorrected 的混合形式
bounded	有界、一阶
fourth	无界、四阶、守恒

表 4.9 拉普拉斯项中面法相梯度的数学行为

4.4.5 散度格式

divSchemes 子字典包含散度项。我们现在来讨论流体力学中对流项 $\nabla \cdot (\rho U U)$ 的句法规则，在 OpenFOAM 中它经常由 `div(phi,U)` 来表示，phi 通常表示 ρU 。

Gauss 格式是唯一可选的格式，并且需要为通量的依赖场（例如 U ）来选择点到面的插

值格式。参见下面的具体形式：

Gauss <interpolationScheme>

表 4.6 列举了普通的点到面插值格式以及针对对流项的点到面插值格式的名称。表 4.10 描述了它们相对的数学行为。需要注意的是，在这个格式中，并没有对已知的通量指定格式，例如 $\text{div}(\phi, U)$ 中的 ϕ ，因为在求解中这个 ϕ 是一个已知量⁴⁷，在这里继续指定离散格式是没有意义的。例如，我们可以这样为对流项指定迎风格式：

$\text{div}(\phi, U)$ Gauss upwind

格式	数学描述
linear	无界, 二阶
skewLinear	无界, 歪斜修正, 二阶
cubicCorrected	四阶, 无界
upwind	一阶, 有界
linearUpwind	一/二阶, 有界
QUICK	一/二阶, 有界
TVD 格式	一/二阶, 有界
SFCD	二阶, 有界
NVD 格式	一/二阶, 有界

表 4.10 divScheme 所用格式的数学描述

4.4.6 时间格式

时间项($\partial / \partial t$)在子字典 ddtSchemes 下制定。可选的离散格式可以在表 4.11 中指定。Crank-Nicholson 格式中有一个偏移系数 φ ，通过这个系数可以设定格式是否和 Euler 格式混合。 $\varphi=1$ 意味着纯的 Crank-Nicholson 格式， $\varphi=0$ 意味着纯欧拉格式。大于 0 小于 1 的系数可以提高稳定性，因为在有些情况下纯 CrankNicholson 格式并不稳定。

格式	描述
Euler	一阶, 有界, 隐性
localEuler	局部时间步, 一阶, 有界, 隐形
CrankNicholson	二阶, 有界, 隐性
backward	二阶, 隐性
steadyState	忽略时间导数项 (稳态)

表 4.11 ddtSchemes 可用的离散关键词

⁴⁷ 大多数求解器代码中都包含了 createPhi.H 文件用于计算 ϕ

在指定时间项格式的时候，我们必须注意到对于某一个求解器，如果这个求解器本身是瞬态求解器，就不能用于稳态计算。反之亦然。例如，如果为瞬态的不可压缩求解器 `icoFoam` 的算例指定为 `steadyState`，那么他将不会收敛。相反，`simpleFoam` 求解器却适用于不可压缩稳态的情况。二阶时间偏导项($\frac{\partial^2}{\partial t^2}$)在 `d2dt2Schemes` 子字典中指定，只有 `Euler` 可以选择。

4.4.7 通量计算

`fluxRequired` 子字典列举了求解器中需要生成的通量场。例如，许多求解器中，在求解压力方程之后我们需要生成通量。这可以通过在 `fluxRequired` 子字典中这样简单地指定（`p` 代表压力）

```
fluxRequired
{
    p;
}
```

4.5 求解和算法控制

方程求解器、残差、算法在 `system` 文件夹下的 `fvSolution` 文件中控制。下面是 `icoFoam` 求解器所需的一个典型的 `fvSolution` 设置：

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

`fvSolution` 包含一系列子字典来为求解器的顺利运行提供必要信息。大部分求解器都是参照这个图中来进行设置。这些子字典中包括 `solvers`, `relaxationFactors`, `PISO` 以及 `SIMPLE` 等，在后面的章节中我们都有所描述。

4.5.1 矩阵求解器

这个例子中第一个子字典（所有求解器的第一项均为）是 solvers。它指定求解离散项需要的矩阵求解器。这里矩阵求解器和我们说的流体力学方程求解器是不同的。矩阵求解器是指用来求解大型矩阵的程序。相反的是，流体力学方程求解器是求解一系列偏微分方程组的，包含一系列算法的程序。在之后的讨论中，我们用求解器来表示矩阵求解器，我们不希望读者产生混乱。

求解器中的出现的 word 关键词代表着方程组中的每一个变量。例如，icoFoam 求解速度和压力场。因此存在 U 和 p。关键词之后跟随的是字典文件，里面有求解器类型，以及求解器使用的参数。在 solvers 关键词下我们有不同的场，在每个场内的 solver 关键词下指定求解器。表 4.12 列举了可使用的求解器。一些参数比如 tolerance, relTol, preconditioner, 将在下面进行描述。

矩阵求解器	关键词
预处理共轭梯度求解器	PCG/PBiCG
光顺求解器	smoothSolver
多重网格求解器	GAMG
显性离散对角矩阵求解器	diagonal
PCG 用于对称矩阵， PBiCG 用于非对称矩阵	

表 4.12 矩阵求解器

矩阵求解器分为对称矩阵求解器和非对称矩阵求解器。矩阵的对称性和求解的方程有关，用户应该会知道矩阵类型，如果用户使用不合适的矩阵求解器 OpenFOAM 会报错。

```
--> FOAM FATAL IO ERROR : Unknown asymmetric matrix solver PCG
Valid asymmetric matrix solvers are :
3
(
PBiCG
smoothSolver
GAMG
)
```

4.5.1.1 求解误差

稀疏矩阵求解采用迭代的方式来处理。它们建立在把残差降为最小。残差是在求解矩阵的时候产生的，这个残差越小，结果越精准。准确来讲，残差就是把当前的解输入到方程中，然后取方程左右两边差的绝对值。同时需要对残差进行统一处理以保证残差规模和待解决问题的规模无关。

在求解某个场方程之前，初始残差凭当前时间步的值来评估。在每次迭代之后，残差再次评估，在达到下面的条件之后，求解器停止计算。

- 残差降到求解器设定的误差 tolerance 之后;
- 当前残差和初始残差的比降到求解器设定的相对残差比之后, relTol;
- 迭代数超过最大值, maxIter;

求解器误差表示当残差小于这个值的时候,我们认为计算结果已经足够精准的时候的误差。求解器的相对残差比限制了从初始场到最终结果的步进。在瞬态计算中,一般我们设定求解器相对残差比为 0, 强制求解器在每个时间步内收敛。tolerance 以及 relTol 在所有的求解器中必须要指定; maxIter 可以不进行设定。

4.5.1.2 预处理双共轭梯度求解器

在共轭梯度求解器中,可以对矩阵进行预处理,它在求解器字典中通过 preconditioner 关键词指定。表 4.13 列出了可选的预处理:

预处理	关键词
不完全的对角 Cholesky 预处理 (对称矩阵)	DIC
不完全的快速对角 Cholesky 预处理(带缓存的 DIC)	FDIC
对角不完全 LU 预处理 (非对称)	DILU
对角预处理	diagonal
多重网格预处理	GAMG
不适用预处理	none

表 4.13 预处理选项

4.5.1.3 光顺矩阵求解器

使用光顺器的求解器需要指定关键词: smoother。表 4.14 列举了可选的光顺器,一般情况下 GaussSeidel 是最可靠的选择,但对于有些不好的矩阵, DIC 的收敛性可能更好。另外,在某些情况下,使用 GaussSeidel 进行后光顺强化是非常有效的。例如 DICGaussSeidel:

光顺器	关键词
高斯赛德尔光顺	GaussSeidel
不完全的对角 Cholesky 光顺 (对称矩阵)	DIC
不完全的对角 Cholesky-高斯赛德尔光顺 (对称矩阵)	DICGaussSeidel

表 4.14 光顺求解器选项

其中,在重新计算残差之前需要通过为 nSweeps 关键词来指定 sweep 数。

4.5.1.4 GAMG

GAMG⁴⁸的概念是：在粗网格上快速生成一个解，然后使用这个解做为初始值并把这个解投影到细网格上，然后在细网格上获得准确解。当先在粗糙网格计算消耗的时间比网格精细化和映射场数据少时，当在粗网格计算消耗的时间比网格精细化和投影场数据消耗时间少时，GAMG 比标准方法要快。实际上，GAMG 求解首先需要读取用户指定的网格然后进行糙网格或细网格的过渡进程。用户只需要定义 nCoarsestCells 关键字即可通过糙网格级别来指定最糙的网格大小。

不同网格层的融合（agglomeration）方法通过融合（agglomeration）关键词指定。我们建议采用 faceArea Pair 方法。但是用户也可以选择 MGridGen 方法，这个方法需要加载一个 MGridGen 库：

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

从 OpenCFD 的经验来看，相比于 faceAreaPair 方法，MGridGen 并没有非常明显的优势。对于所有的 agglomeration 方法，糙网格的缓存解通过 cacheAgglomeration 关键词来控制。

正如 4.5.1.3 节所说，光顺可以通过 smoother 关键词来指定。光顺器在不同级的网格上使用的 sweep 数可以通过 nPreSweeps, nPostSweeps, nFinestSweeps 来指定。nPreSweeps 信息用于在多重网格求解器向更糙网格步进时候的迭代数，nPostSweeps 信息用于向更细的网格步进时候的迭代数，nFinestSweeps 指定最终网格的迭代数⁴⁹。

mergeLevels 关键词控制向糙网格或者细网格步进时候的速度。我们最好把细化(糙化)级别设置为 1，例如把 mergeLevels 设置为 1。然而在某些情况下，特别是那些很简单的网格算例，如果一次进行两次细化(糙化)，有可能会加速求解。

4.5.2 低松弛

fvSolution 中的另一个子字典是 relaxationFactors，这个关键字主要用来控制低松弛（提高求解稳定性，特别是稳态问题）。低松弛技术就是通过限制每次变量迭代后改变的大小来实现。其要么通过在迭代之前修改矩阵系数和源项来实现，要么通过直接修改求解后的场来实现。松弛因子 α 小于 1 大于 0，它用来调节松弛量。 $\alpha=1$ 代表没有松弛， $\alpha=0$ 表示完全的松弛。在 $\alpha=0$ 的算例，即使成功迭代每次的解也不会改变。最好的 α 是：它足够小以确保稳定计算，并且足够大确保迭代迅速有效。例如，如果 $\alpha=0.9$ ，在某些情况下就已经可以保证稳定性了。如果再选取比较低的数值，比如 0.2，这可能会导致降低迭代效率。

用户可以对一个特定的场指定松弛因子：先指定场，再指定松弛因子。在算例 simpleFoam 中，我们可以看一下不可压层流稳态算例的松弛因子：

```
17
18 solvers
19 {
20   p
21   {
22     solver      GAMG;
23     tolerance  1e-06;
24     relTol    0.1;
```

⁴⁸ 更多信息请参考 <http://aerojet.engr.ucdavis.edu/fluenthelp/html/ug/node1380.htm>,

⁴⁹ 更多信息请参考：<http://www.cfd-online.com/Forums/openfoam-solving/96572-parameters-multigrid-solver.html>

```

25     smoother      GaussSeidel;
26     nPreSweeps   0;
27     nPostSweeps  2;
28     cacheAgglomeration on;
29     agglomerator faceAreaPair;
30     nCellsInCoarsestLevel 10;
31     mergeLevels   1;
32 }
33
34 "(U|k|epsilon|R|nuTilda)"
35 {
36     solver        smoothSolver;
37     smoother      symGaussSeidel;
38     tolerance     1e-05;
39     relTol       0.1;
40 }
41 }
42
43 SIMPLE
44 {
45     nNonOrthogonalCorrectors 0;
46
47     residualControl
48     {
49         p           1e-2;
50         U          1e-3;
51         "(k|epsilon|omega)" 1e-3;
52     }
53 }
54
55 relaxationFactors50
56 {
57     fields
58     {
59         p           0.3;
60     }
61     equations
62     {
63         U           0.7;
64         k           0.7;
65         epsilon     0.7;
66         R           0.7;
67         nuTilda    0.7;
68     }
69 }
70
71
72 // ****

```

4.5.3 PISO 和 SIMPLE 算法

大多数 OpenFOAM 的求解器使用 PISO 或者 SIMPLE 算法。这些算法采用迭代来求解速度和压力场。PISO 算法用来处理非稳态问题，SIMPLE 算法用来处理稳态问题。

这两个算法的基本特点都是先计算某些场的初始值然后对其进行修正。SIMPLE 算法在一个迭代步仅仅做一次修正但是 PISO 算法在一个时间步内进行多次修正，但一般不超过 4 次。用户必须通过 `nCorrectors` 关键词来指定 PISO 的修正数，参见 120 页的范例。

OpenFOAM 采用 PISO 算法和 SIMPLE 算法的求解器中，都可以选择附加的非正交修正。如果网格是正交的，这意味着面法向和网格单元中心的矢量是平行的。例如矩形网格，他们的网格单元面是和笛卡尔坐标系对齐的。非正交修正数通过 `nNonOrthogonalCorrectors` 关键词来指定。请参见 120 页的范例。非正交修正数应该和网格形状匹配，例如正交网格使用 0 次修正。网格非正交性越强，非正交修正数越多。比如 20 次。

4.5.3.1 参考压力

在一个封闭的不可压系统内部，压力是一个相对值。压差（而非绝对压力）才是主要的。在

⁵⁰ 此处采用了两种松弛技术

这些例子中，求解器给 `pRefCell` 设定一个参考值 `pRefValue`。`p` 即为压力变量的名字。有些算例中压力是 `p_rgh`，那么参考压力和参考网格就是 `p_rghRefValue` 以及 `p_rghRefCell`。这些信息存储在 PISO/SIMPLE 的子字典中，有些特定的算例需要指定这些信息并供给求解器使用。如果省略了这些信息，求解器将不会运行，但是会提醒用户问题所在。

4.5.4 其它参数

OpenFOAM 标准求解器中 `fvSolution` 文件包含的信息，本章均已罗列。然而，`fvSolution` 文件还可以包含任何其它用于控制求解器、算法或其它东西的相关参数。对于一个求解器，用户应该去查看它的源代码看看哪些参数是必须存在的。任何参数或者子字典的未定义，求解过程都将会终止并输出错误警告。用户应该把没有包含的信息添加进去。

第五章

网格生成和转换

这一章我们讨论所有和网格生成有关的话题。5.1 节总览一下 OpenFOAM 所用的网格方式；5.2 节介绍网格边界类型；5.3 节介绍自动生成简单六面体网格的程序 `blockMesh`；5.4 节介绍 `snappyHexMesh` 程序，它用来依据几何文件全自动切分六面体网格并生成复杂的多面体网格；5.5 节介绍如何用程序把第三方网格转换成 OpenFOAM 可用的格式。

5.1 网格

这一节我们介绍 OpenFOAM 中使用的 C++类如何处理网格。网格是数学求解的主要部分，并且必须满足某些标准以确保求解有效和准确。在运行过程中，OpenFOAM 有一套非常严格的检查网格的标准，如果不被满足的话运行将会停止。这样的不利后果是，在修正第三方软件生成的大型网格以确保 OpenFOAM 可以使用的时候，用户可能会感到很困难并非常沮丧。这非常的不幸。但是确保网格的有效性是非常重要的。否则，在求解开始之前，结果就存在缺陷。

默认情况下，OpenFOAM 可以识别下面这种网格单元形式：即一个使用任意数量的面来定义的一个有界的 3D 多面体。例如，单一的网格单元可以有无数个面，对于每个面，边的数量也没有限制，边的结合方式也没有限制。具有这样结构的网格在 OpenFOAM 中被熟知为 `polyMesh`。在几何非常复杂的时候，或者网格随时间改变的时候，这对网格生成以及操作提供了最大的自由。然而，这种灵活的网格结构带来的代价是，对于某些传统的网格生成工具生成的网格转换比较困难。因此 OpenFOAM 库提供了 `cellShape` 来预定义传统形状网格格式。

5.1.1 网格规范以及限制

在对 OpenFOAM 网格格式 `polyMesh` 文件夹，以及 `cellShape` 工具进行描述之前，我们首先制定了一些网格规则：

5.1.1.1 点

点文件由一个三维空间中的一个位置矢量来定义，单位为米。点文件被编写成了一个列表（list），每个点文件由 label 标示，从 0 开始，代表在 list 中的位置。不能在一个相同的物理位置定义两个不同的点，也不允许有不在任何面内的点存在。

5.1.1.2 面

面是排列有序的点集，点由其序号来表示。每个点通过边来相连，并以此来指定点的顺序。例如：如果你跟随着这个点，你将围着这个面环绕一周。面被汇集在一个列表中，每个面由它的 label 标示，代表它在列表中的位置。面法向矢量由右手规则定义。例如，当你看一个面得时候，如果一个面上的点成逆时钟方向排序，那么这个面矢量就是面对你的。如图 5.1 所示：

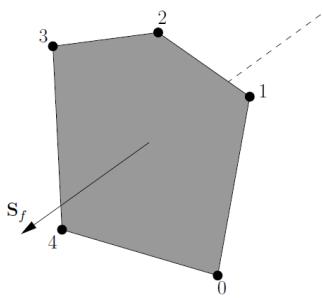


图 5.1 依据点序来查看面矢量

这里存在两种面：

- **内部面：**这些面将两个网格单元连接起来（仅仅是两个网格单元）。每个内部面的面法向向量指向标识数比较大的网格单元。点由面法向的方向进行排序。例如对于连接网格单元 2 和网格单元 5 的内部面，面法向向量指向网格单元 5；
- **边界面：**它们仅属于一个网格单元，由于它们和边界重合。因此边界面由一个网格单元和一个 patch 来指定。面法向向量指向计算域之外，点由面法向向量的方向来排序；

网格的面应该是凸起的，至少面心应该在面内。面也可以歪斜，并不是所有的面都是平面。

5.1.1.3 网格单元

网格单元是面的汇集，单个网格单元必须具有以下特点：

- **连续性：**所有的网格单元必须完整的覆盖计算域，并且彼此不能重合；
- **凸起的：**每个网格单元必须是凸起的，并且网格单元中心在网格内；

- **封闭的:** 不管从几何来讲, 还是拓扑结构来说, 每个网格单元必须是封闭的;
 1. 几何封闭要求当把网格单元的面矢量加和的时候, 总和应该等于 0 (机械精度);
 2. 拓扑封闭要求一个网格单元内所有的边都由两个面共有;
- **正交性:** 对于网格单元内的所有内部面, 我们定义连接两个网格单元中心的矢量为中心对中心矢量(*center-to-center vector*)。它从序号低的网格单元指向序号高的网格单元。正交性规则要求对于所有的内部面, 面矢量和中心对中心矢量的夹角要小于 90° ;

5.1.1.5 边界

边界是一系列 patch 的集合, 每个边界和一个边界条件相对应。每个 patch 是一系列面得集合, 这些面只包含边界面并且不能是内部面。边界必须是封闭的, 例如, 所有的边界面矢量之和为 0 (机器精度)。

5.1.2 polyMesh

constant 文件夹下的 polyMesh 文件夹包含了对 polyMesh 的全部描述。polyMesh 建立在我们之前讨论过点、内部面(见上述讨论)以及边界面(通过一系列网格单元的面以及边界 patch 来指定)基础之上。每个面被指派了一个 owner 和一个 neighbor, 因此每个连接两个网格的面都具有一个 owner 和 neighbor。对于边界, 和它连接的网格是 owner, neighbor 被指定为 -1。记住这点之后, 网格的输入输出包含以下文件:

points: 一系列描述网格顶点的位置矢量, 第一个矢量代表点 0, 第二个矢量代表定点 1, 以此类推;

faces: 包含了一系列的面, 每个面用点序号的组成并标志, 第一行信息代表面 0。以此类推;

owner: 包含一系列的 owner 标识, 这里面的信息和面相关。第一行的信息表示面 0 的 owner 序号, 第二行信息标示面 1 的 owner 序号, 以此类推;

neighbor: 一系列的 neighbor 标示, 参考上述 owner 的描述;

boundary: 包含一系列的 patch (不同的 patch 具有不同的名字), 每个面具有相应的字典信息, 例如:

```
movingWall
{
    type patch;
```

```

nFaces 20;
startFace 760;
}

```

`startFace` 是这个 patch 中面 1 的标识, `nFaces` 是这个 patch 中面得总数。如果用户想知道计算域内有多少网格, 在 `owner` 文件中的文件头 `FoamFile` 中, `note` 里面包含了 `nCells` 的相关信息。

5.1.3 cellShape

如果想把一些标准的简单网格形状输出为 OpenFOAM 可用的网格文件, 我们可以用 `cellShape`⁵¹来进行预定义。

由于许多网格软件以及后处理系统支持的多面体结构数量非常有限, 它们用一系列三维几何形状(即所谓的网格类型)来定义网格。OpenFOAM 自带的库包含了这些标准网格类型的预定义, 并可以将其从这种网格类型转换到之前我们讲述的 `polyMesh` 文件中。

表 5.1 中列举了 OpenFOAM 支持的网格形状, 每个形状的点、面、边数量都已经预定义。点序、面序、边序在预定义的数量上进行定义, 参见表 5.1。某个形状的点数量指定不能让网格形状发生扭曲、塌陷甚至变为其他类型。例如, 在某个形状中, 同样的点在排序中只能用一次。依靠当前的形状规则已经可以指定各种形状, 因此在 OpenFOAM 中没必要进行点重叠来重新定义。

网格单元的描述由两部分构成, 网格单元名称以及排列有序的标示链表。因此, 如果我们有以下的点集:

```

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)
)

```

一个六面体网格可以这样描述:

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

在这里六面体网格形状用 `hex` 来表述。其它网格形状可以参考表 5.1。

⁵¹ 原文为 `cellShape tool`, 但其并不是一个工具 (utility)

5.1.4 一维、二维以及轴对称问题

OpenFOAM 代码主要针对 3 维问题并且按照 3 维问题来定义网格。然而，OpenFOAM 同样也可以计算 1, 2 维以及轴对称问题，你首先需要生成一个三维网格，然后在任何的 patch（那个不需计算的 patch）上应用一个特定的边界条件即可。在 1, 2 维问题上的第三方向我们使用 empty 边界条件，轴对称问题使用 wedge 边界类型。请参阅 5.2.2 章节的详细描述，轴对称问题的 wedge 几何生成在 5.3.3 节有描述。

Cell type	Keyword	Vertex numbering	Face numbering	Edge numbering
Hexahedron	hex			
Wedge	wedge			
Prism	prism			
Pyramid	pyr			
Tetrahedron	tet			
Tet-wedge	tetWedge			

表 5.1 不同网格形状的点、面、边的数量

5.2 边界

这一节我们讨论 OpenFOAM 中的边界条件。在之前的描述中，边界问题我们很少提及，因为它们的角色不仅仅是一个几何实体，相反，它们是边界求解和数值运算的一部分，也可以说是把内部场和边界条件联系起来的纽带。之前我们在讨论网格、场、离散、计算过程的时候讨论边界有点不太合适。因此它被放在这一章是不错的选择。

我们首先需要了解到，边界通常被分割为一系列 patch 的集合，每个 patch 包含了一系列的边界面（封闭的面），边界面不需要是物理相连的。

我们按照继承的关系，列举了边界类型。patch 上的边界主要具有三种特性，参见图 5.2，他们依据继承关系被描述。图中表示了不同的 patch 名称以及某些可选的边界条件类型。图中这种继承关系和 OpenFOAM 库采用的 C++ 语言的继承有所类似但不相同。

- **基本类型**: 仅仅描述几何类型, 或者交互渠道;
 - **主要类型**: 边界上某个场的基本数值边界条件;
 - **衍生类型**: 一个复杂的边界条件, 从主要类型中衍生, 为边界上某个场指定;

基本类型	patch wall 	symmetry empty wedge cyclic processor
主要类型	fixedValue fixedZeroGradient zeroGradient mixed directionMixed calculated 	
衍生类型	inletOutlet	

表 5.2 patch 特性

5.2.1 在 OpenFOAM 中指定边界条件

OpenFOAM 算例中，边界类型需要在网格和场中来指定。详细来说：

- 在 constant/polyMesh 的 boundary 文件夹下， type 关键词后应该指定基本类型；
 - 数学边界类型，不管是主要边界类型还是衍生类型，都需要在场数据的 type 中指定；

下面是 sonicFoam 的 p 文件边界条件范例：

```

17
18 6
19 (
20   inlet
21   {
22     type      patch;
23     nFaces   50;
24     startFace 10325;
25   }
26   outlet
27   {
28     type      patch;
29     nFaces   40;
30     startFace 10375;
31   }
32   bottom
33   {
34     type      symmetryPlane;
35     inGroups 1(symmetryPlane);
36     nFaces   25;
37     startFace 10415;
38   }
39   top
40   {
41     type      symmetryPlane;
42     inGroups 1(symmetryPlane);
43     nFaces   125;
44     startFace 10440;
45   }
46   obstacle
47   {
48     type      patch;
49     nFaces   110;
50     startFace 10565;
51   }
52   defaultFaces
53   {
54     type      empty;
55     inGroups 1(empty);
56     nFaces   10500;
57     startFace 10675;
58   }
59 )
60
61 // ****

```

```

17 dimensions [1 -1 -2 0 0 0];
18
19 internalField uniform 1;
20
21 boundaryField
22 {
23   inlet
24   {
25     type      fixedValue;
26     value    uniform 1;
27   }
28
29   outlet
30   {
31     type      waveTransmissive;
32     field    p;
33     phi      phi;
34     rho      rho;
35     psi      thermo psi;
36     gamma   1.4;
37     fieldInf 1;
38     IInf    3;
39     value    uniform 1;
40   }
41
42   bottom
43   {
44     type      symmetryPlane;
45   }
46
47   top
48   {
49     type      symmetryPlane;
50   }
51
52   obstacle

```

```

53  {
54      type      zeroGradient;
55  }
56
57 defaultFaces
58 {
59     type      empty;
60 }
61 }
62
63 // ****

```

本例中除了那些具有特定几何限制的 patch, 例如 symmetryPlane 和 empty, 边界中的 type 就指定为 patch。在 p 文件中, 我们可以看出在 inlet 和 bottom 面使用了主要类型, outlet 上使用了复杂的衍生类型。对比这两个文件我们可以看出, 在基本类型不是 patch 的时候, 例如对于 empty 和 symmetryPlane 类型, 边界类型和数值类型是一致的。

5.2.2 基本类型

下面我们来描述基本几何类型, OpenFOAM 用来指定它们的关键词请参考表 5.2:

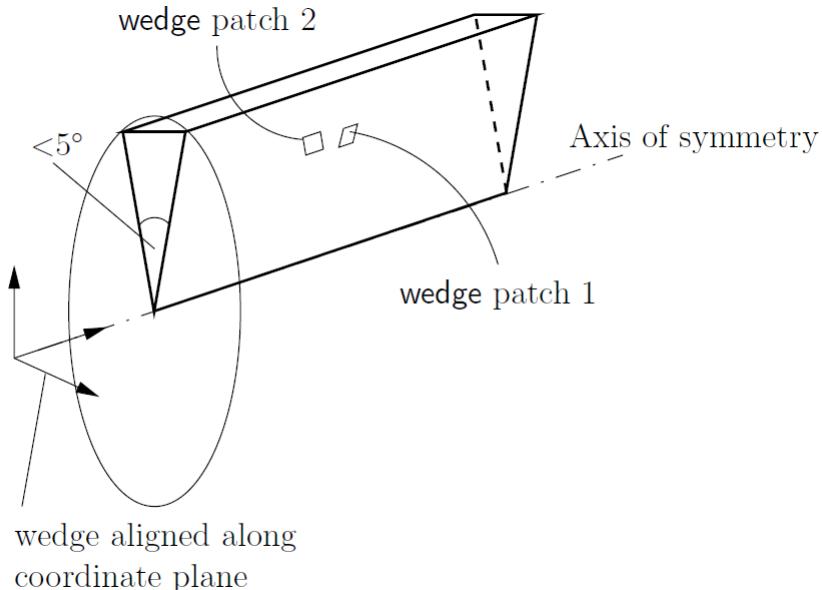


图 5.3 wedge 轴对称几何

- patch: 若指定 patch 边界条件, 说明网格上没有几何拓扑信息 (wall 除外), 例如 inlet 和 outlet;
- wall: 主要用于可识别的壁面, 在这种情况下, 需要指定为 wall, 这样才能使某些特定的模型可选。一个很好的例子就是在湍流壁面函数中, 壁面必须指定为 wall, 这样壁面和网格中心的距离才会被存储;
- symmetryPlane: 对称平面;

关键词	描述
patch	普通 patch
symmetryPlane	对称平面
empty	2D 几何的前后面
wedge	轴对称问题的前后面
cyclic	周期平面
wall	壁面, 用来指定湍流的壁面函数
processor	内部处理器边界 ⁵²

表 5.2 基本类型

- empty: OpenFOAM 生成的是三维网格, 它可以通过把第三个方向(不在此方向进行求解)上的 patch 指定为 empty 边界类型来实现;
- wedge: 对于 2 维轴对称算例, 例如圆柱。几何可以被定义一个楔形, 它的夹角很小, 例如要小于 5 度, 它的轴要和圆柱的轴平行, 如图 5.3 所示。轴对称的 wedge 平面必须分别通过不同的 patch 来指定。5.3.3 节我们将讨论如何用 blockMesh 来生成 wedge 几何;
- cyclic: 这种边界条件可以使两个不相连的 patch“物理上连接起来”。主要用于周期几何, 例如散热片。想把一个 cyclic 边界和另一个 cyclic 边界连接起来, 需要在 boundary 文件中指定另一个为 neighbourPatch。互相匹配面的面积需要相同, 至少在设置的误差内, 它可以在 boundary 文件中的 matchTolerance 关键词后设置。这两个面不需要方向相同;
- processor: 如果需要并行计算, 那么就需要分割网格, 只有这样每个处理器才能大体计算相同数量的网格。不同核之间处理的网格的边界成为 processor 边界;

5.2.3 主要类型

表 5.3 列举了主要类型

5.2.4 衍生类型

OpenFOAM 中有大量衍生边界条件, 不能一一列举。其中一小部分在表 5.4 中列出。如果用户希望获得可使用模型的全部列表。它应该查阅 OpenFOAM 源代码。衍生边界条件存放在此处:

- \$FOAM_SRC/finiteVolume/fields/fvPatchFields/derived

⁵² 并行计算有采用, 自动生成

- 在某个模型库中，通过在终端中键入以下命令行，终端会显示目录信息：

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

类型	描述	需要指定的数据
fixedValue	指定 φ 的值	value
fixedGradient	指定 φ 的法相梯度	gradient
zeroGradient	φ 的法相梯度为 0	——
calculated	φ 的值从其他场的值计算而来	——
mixed	依据 valueFraction 的值而变的 fixedValue 和 fixedGradient 混合形式	refValue refGradient valueFraction value
directionMixed	valueFraction 具有方向的 mixed 形式，例如，对于法向和切向的混合级数不同的情况	refValue refGradient valueFraction value

表 5.3 主要类型

- 在某个求解器中，通过在终端键入以下命令：

```
find $FOAM SOLVERS -name "*fvPatch"
```

5.3 blockMesh 网格生成程序

这一章节我们讨论 OpenFOAM 提供的 blockMesh 程序，blockMesh 程序用来创建均匀或者非均匀分布以及曲边的网格。

网格通过位于 constant/polyMesh 字典下的 blockMeshDict 来生成，blockMesh 读取这个字典，在同样的文件位置下生成网格，输出点、面、网格和边界信息。

blockMesh 的原则是把计算域分解成为 1 个或者多个三维的六面体块。这些块的边可以为直线也可以为曲线。在每个六面体块上，指定每个方向的网格数量。使用这些信息就足够用来生成网格数据了。

每个 block 由 8 个顶点来定义，它们位于六面体的每个角上。顶点被写入了链表因此它们可以通过标志号来被计算机识别。OpenFOAM 就是 C++ 程序，第一个标志永远是“0”。表 5.4 就是一个 block 的典型例子。每个顶点都进行了排号。连接顶点 1 和顶点 5 的边是曲边，这表明 blockMesh 也可以指定曲边。

blockMesh 也可以生成少于 8 个顶点的 block，这需要将一个或者一对的顶点去除，5.3.3 节会有详细的描述。

每个 block 的局部坐标系统(x1,x2,x3)都要遵循右手规则。右手规则的定义是：当用户顺着 oz 方向看的时候 (o 点为离用户近的点)，ox 上的点到 oy 上的点的弧的方向为顺时针。

局部坐标系统的定义是有顺序的，block 的顶点这样来定义：

fixedValue 的衍生类型		需要指定的关键词
movingWallVelocity	改变边界面的法向值以使得通过边界的通量为 0 ⁵³	value
pressureInletVelocity	进口压力已知,速度通过边界的法向通量计算	value
pressureDirectedInletVelocity	进口压力已知,速度通过边界的 inlet - Direction 方向的通量计算	value inletDirection
surfaceNormalFixedValue	通过固定大小为边界面矢量指定法向边界条件	value
totalPressure	$p_0 = p + 0.5\rho u^2$ 固定,速度改变的时候,压力随之改变	p_0
turbulentInlet	通过平均波动尺度计算波动量	referenceField fluctuationScale
fixedGradient/zeroGradient 的衍生类型		
fluxCorrectedVelocity	通过进口通量计算速度分量	value
buoyantPressure	依据气压设定 fixedGradient 压力条件	——
mixed 的衍生类型		
inletOutlet	依据速度的方向把速度和压力的边界条件在 fixedValue 和 zeroGradient 之间进行切换	inletValue value
outletInlet	依据速度的方向把速度和压力的边界条件在 fixedValue 和 zeroGradient 之间进行切换	outletValue value
pressureInletOutletVelocity	inletOutlet 和 pressureInletVelocity 的混合	value
pressureDirectedInletVelocity	inletOutlet 和 pressureDirectedInletVelocity 的混合	value inletValue
pressureTransmissive	压力传递边界条件	pInf
supersonicFreeStream	非直角波震压力 p_∞ , 温度 T_∞ , 速度 U_∞ 传递边界条件	pInf, TInf, UInf
其他类型		
slip	对于标量场使用 zeroGradient 条件,对于矢量场的切向分量为 zeroGradient, 法相分量为 fixedValue	——
partialSlip	依据 valueFraction 的 zeroGradient 和 slip 的混合形式, =0 意味着 slip	valueFraction
p 是压力, U 是速度		

表 5.4 衍生边界条件

⁵³ 主要用于动网格

- 坐标轴顶点是 block 定义的第一条信息，我们的例子中是顶点 0；
- 从顶点 0 到顶点 1 是 x_1 轴；
- 从顶点 1 到顶点 2 是 x_2 轴；
- 顶点 0, 1, 2, 3 定义 $x_3=0$ 的平面；
- 顶点 4 位于将顶点 0 顺着 x_3 方向移动的轴上；
- 类似的，5, 6, 7 顶点就是位于 1, 2, 3 顶点分别沿着 x_3 移动的轴上；

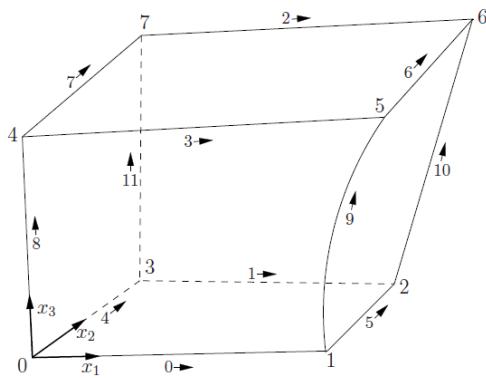


图 5.4 单一 block 块

关键词	描述	范例
convertToMeters	点位置矢量缩放因子	0.001, 缩放为 mm
vertices	点位置列表	(0 0 0)
edges	用于指定 arc 以及 spline 边	arc 1 4 (0.939 0.342 -0.5)
block	通过有序的点定义，以及每个方向的网格数量	hex (0 1 2 3 4 5 6 7) (10 10 1)
patches	patches 列表	symmetryPlane base ((0 1 2 3))
mergePatchPairs	需要合并的 patches 列表	详见 5.3.2 节

表 5.5 blockMeshDict 关键词

5.3.1 编写 blockMeshDict 文件

blockMeshDict 使用表 5.5 所列的关键字来定义，convertToMeters 指定点矢量缩放因子，所有的顶点坐标矢量都要乘以这个因子。例如：

```
convertToMeters    0.001
```

意味着所有坐标值都乘以 0.001。这样，blockMeshDict 中的值就是以 mm 为单位。

5.2.1.1 顶点

block 顶点通过一个标准链表 vertices 来指定，例如，表 5.4 中的 block 顶点这样定义：

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

5.3.1.2 边

连接两个顶点的边默认认为是直线，然而很多边都可以通过 edges 来指定为曲线。这个 list 是可以选择的，如果 block 没有曲边，它可以省略。每个曲边通过表 5.6 列举的关键词来指定：

关键词	描述	附加信息
arc	圆弧	需要一个插值点
simpleSpline	样条曲线	一系列插值点
polyline	线集	一系列插值点
polySpline	样条曲线集	一系列插值点
line	直线	——

表 5.6 blockMeshDict 可用的边类型

在表 5.6 列举的关键词之后需要定义连接这个边两个点。如果这个边通过某些插值点，他们也需要被指定。例如，对于一个 arc，只需要定义一个 arc 经过的插值点信息即可。对于 simpleSpline，polyLine 以及 polySpline，需要定义一系列插值点。line 就是缺省的边，不需要插值点信息。实际上我们并不需要给定 line 的信息，但是我们包含了它以确保完整性。图 5.4 中的 block 我们指定一个 arc 边来连接顶点 1 和顶点 5，并且它们通过了插值点(1.1,0.0,0.5)：

```
edges
```

```
(
```

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

5.3.1.3 块

block 块通过 blocks 链表来定义。每个 block 需要一系列的信息组合来定义。包括 block 的顶点（它们的顺序在 5.3 节中有描述）、每个方向上的网格数量、每个方向上的网格单元膨胀率。

其 block 这样定义：

```
blocks
(
    hex (0 1 2 3 4 5 6 7)      // 顶点数
    (10 10 10)                // 每个方向的网格数
    simpleGrading (1 2 3)      // 网格单元膨胀率
);
```

每个 block 的信息如下：

- **顶点数：**在顶点数信息之前定义的是 block 的形状信息，它们在 OpenFOAM-2.3.0/cellModels 文件中声明。由于 blocks 经常是六边形，因此 hex 居多。然后它列举了顶点信息，以 136 页所提的方式来排序；
- **网格数量：**第二行信息定义 x1, x2, x3 方向的网格数量；
- **网格单元膨胀率：**第三行信息定义每个网格方向上的网格单元膨胀率。膨胀率使网格在特定方向上能够非均匀化或细化。参阅图 5.5，膨胀率就是一条边上最后一个网格的宽度 δ_e 除以最起始网格的宽度 δ_s 。下面列举的两个关键词是 blockMesh 可用的非均匀化关键词；

simpleGrading: 非均匀化最简单的方式，对 x1, x2, x3 方向设置统一的膨胀率，它可以这样来定义：

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

edgeGrading: 在这种方式中，可对 block 的每个边定义网格单元膨胀率。他们通过图 5.4 的规则来排序，图中箭头的方向指的是边的起始方向，如果有这样的信息：

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

这意味着 0-3 边的膨胀率为 1, 4-7 边的膨胀率为 2, 8-11 边的膨胀率为 3，这和 simpleGrading 的描述类似。

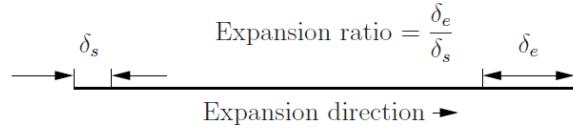


图 5.5 block 边的非均匀处理

5.3.1.4 边界

网格的边界通过 boundary 来定义。边界被拆分为 patch，每个 patch 都有它们的名字，用户可以进行选择，但是我们建议把 patch 命名为具有描述性的名字，例如 inlet。patch 下的信息以子字典的形式来包含：

- type: 表述 patch 类型，要么是一个普通类型，要么就是一个特殊的几何条件，详见表 5.2 以及 5.2.2 节；
- faces: 一系列的 block 面，它们可以组成一个 patch，每个 patch 都有它们的名字，用户可以进行选择，但是我们建议把 patch 命名为好识别的名字，例如 inlet；

如果用户没有在 boundary 中指定某些面信息，blockMesh 默认它们为 defaultFaces，类型为 empty。这意味着是一个 2D 的几何，如果用户知道 blockMesh 会收集省略的面信息并把它们指定为 empty，它就可以省略这部分面定义。

下面我们回到图 5.4 的例子，如果它在左边的面上有一个 inlet，右面的面上有一个 outlet，其它的面都是 walls，那么它可以这样定义：

```

boundary //关键词
(
    inlet //此 patch 名
    {
        type patch; //此 patch 的类型
        faces
        (
            (0 4 7 3); //此 patch 中的 block 面定义
        );
    } //此 patch 定义结束
    outlet //此 patch 名
    {
        type patch; //此 patch 的类型
        faces
        (
            (1 2 6 5)
        );
    }
    walls
)

```

```

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

```

每个 block 面通过 4 个顶点来定义，顶点的顺序必须这样定义，从 block 里面来看，它们起始于任何顶点，其它顶点以顺时针方向来排序。

如果要在 blockMesh 里面指定 cyclic 边界条件，用户需要 neighbourPatch 关键词来指定相关的 cyclic 边界。例如，一对 cyclic 边界需要这样指定：

```

left
{
    type          cyclic;
    neighbourPatch right;
    faces         ((0 4 7 3));
}

right
{
    type          cyclic;
    neighbourPatch left;
    faces         ((1 5 6 2));
}

```

5.3.2 多块网格

网格可以用多个 block 块来创建，在这种条件下，单个 block 块中生成的网格参见前文所述，唯一的问题是 blocks 之间如何相连，这存在两种情况：

- **面匹配：**在某一个 block 中，组成某个 patch 的面由某些顶点来构成，这些顶点又被另一个 block 的 patch 下的面所用；
- **面融合：**某个 block 下的 patch 下的面和另一个 block 下某个 patch 下的面相连，通过这种方式来创造连接两个 block 的内部面；

通过面匹配来连接两个 block 的时候，这个 patch 的信息可以不定义。blockMesh 会识别这些 patch，因为它们不是作为外部边界来存在的，因此 blockMesh 将它们来匹配成为内部面，这个 patch 连接两个 block 的网格。

相对应的，面融合首先要求融合的面在 patch 中必须被定义。然后在 mergePatchPairs 中把需要融合的 patch 信息包含进去，参考如下格式：

```
mergePatchPairs
(
    (<masterPatch><slavePatch>) //面融合的第一个 patch 对
    (<masterPatch><slavePatch>) //面融合的第二个 patch 对
    ...
)
```

成对的 patch 可以这样理解：第一个面是主面，第二个面是次面。融合规则如下：

- 主面按照原先的定义，所有的面顶点在它们最原始的坐标上；
- 次面被投影到主面上，即使次面和主面之间是分离的，这也完全可行；
- 在融合进行的过程中，如果有些面的边小于设定的最小值，blockMesh 会调整次面上的顶点来消除这些边；
- 参阅图 5.6，如果两个面有这样的重叠，重叠的部分成为内部面，没有融合的部分依然为外部面，需要指定边界条件；
- 如果 patch 上所有的面都融合了，那么这个 patch 将被完全的移除；

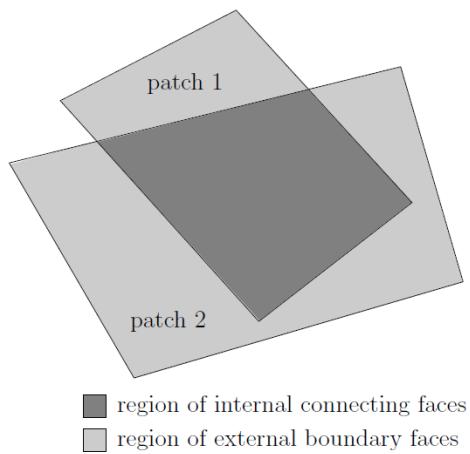


图 5.6 重叠 patch 的融合

我们进行面融合的一个后果是，次面的原始几何不会被完整的保存。因此，如果某个算例中一个圆柱的 block 要和一个更大的 block 相连，最好指定圆柱为主面，这样圆柱的形状会得以保留。为了融合成功，我们提供了以下几个意见：

- 在 2D 几何中，第三维的网格大小，例如 2D 平面外的那个方向的长度，应该和 2d

平面内网格的长度大体一致；

- 不建议融合一个 patch 两次，例如在 mergePatchPairs 把他们包含两次；
- 当两个 patch 面共享一个边的时候，这两个 patch 都需要被指定为主面；

5.3.3 少于 8 个顶点的 block

可以通过删掉一对或者几对顶点来创造少于 8 个顶点的 block。最常见的例子就是创建针对 2 维轴对称算例的楔形网格，它使用 5.2.2 章节提到的 wedge 类型。图 5.7 是一个非常简单的例子。如果用户想把顶点 7 坍塌到顶点 4，顶点 6 坍塌到顶点 5 来创造一个楔形 block，我们可以简单的把语法中的顶点 7 用顶点 4，顶点 6 用顶点 5 的替换来实现，这样就变成：

hex (0 1 2 3 4 5 5 4)

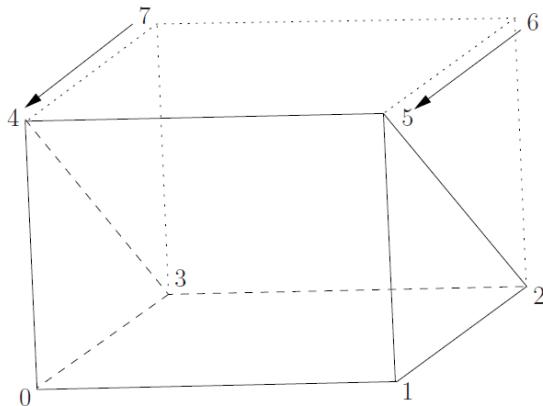


图 5.7 使用 6 个顶点来创造楔形网格

语句变更的要点即为修改需要坍塌点的顺序，例如原先六面体的 block 上部的面为(4 5 6 7)，在楔形 block 的上部面变成了(4 5 5 4)，实际上这是一个面积为 0 的面，它不含有任何面网格，我们可以通过查阅 ployMesh 字典文件的边界以确认。这个 patch 在 blockMeshDict 中应该被指定为 empty，并且边界条件亦应该为 empty。

5.3.4 运行 blockMesh

正如 3.3 节我们阐述的，下面的命令行用于执行 blockMesh，它表示读取<case>文件夹下的 blockMeshDict 并执行：

```
blockMesh -case <case>
```

在 constant/polyMesh 文件夹中必须有 blockMeshDict

5.4 snappyHexMesh 网格生成工具

这一章节我们描述 OpenFOAM 自带的 snappyHexMesh 这个网格生成工具，snappyHexMesh 可以自动地从 STL, OBJ 文件生成六面体及多面体网格。网格依靠迭代将一个初始网格细化，并将细化后的网格变形以依附于表面。在这个过程之后可以选择是否插入网格边界层。snappyHexMesh 通过一个预先定义的网格质量标准进行控制，这个标准非常灵活，表面处理贴合功能非常强健并且可以并行运算。

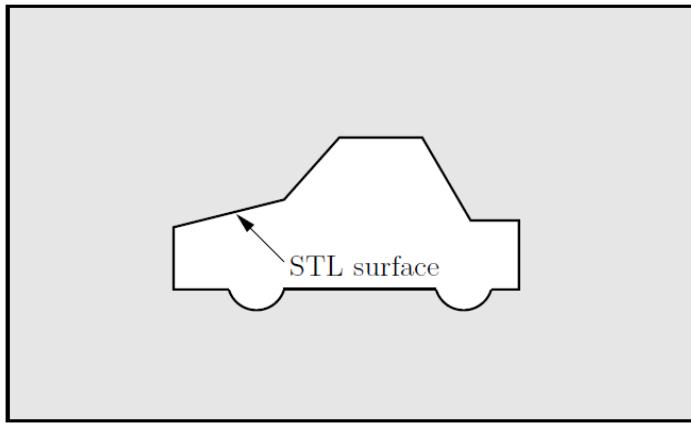


图 5.8 一个使用 snappyHexMesh 来生成 2D 网格的几何

5.4.1 snappyHexMesh 网格生成

snappyHexMesh 生成网格的步骤可以用图 5.8 这个简图来表示。在这里我们生成网格的目的是在这个矩形区域网格内（阴影处）生成一个小汽车（STL 文件几何）的网格。这种网格可以用来进行空气动力学模拟。值得注意的是 snappyHexMesh 是一个 3D 网格生成工具，但这是一个 2 维的简图，这样做的目的是为了让用户理解起来更容易。为了运行 snappyHexMesh，用户需要提供以下内容：

- 提供一个 STL 文件格式的几何文件（binary 或者 ascii 格式）存储在 constant/triSurface 子字典中；
- 一个背景六面体网格，定义计算域范围和背景网格，一般由 blockMesh 生成，详见 5.4.2 节；
- snappyHexMeshDict 字典，它提供了生成网格的必要信息，位于 system 子文件夹下；

snappyHexMeshDict 文件包含了网格质量控制标准，每个标准都是相应的子字典并具有开关来控制。详见表 5.7。

snappyHexMesh 所用的几何文件通过 snappyHexMeshDict 文件中的 geometry 子字典来指定，它可以是一个 STL 文件或者一个有限边界的几何体。参见以下：

关键词	描述	例子
castellatedMesh	切割网格？	true
snap	进行网格贴合（对齐）？	true
doLayers	添加网格边界层？	true
mergeTolerance	自定义区域融合绝对误差	1e-6
调试	过程控制以及屏幕输出 ——仅仅输出最终网格 ——输出过渡网格 ——输出 cellLevel 场数据 ——输出交互文件 (.obj)	0 1 2 4
geometry	所用几何文件子字典	
castellatedMeshControls	切割网格子字典	
snapControls	贴合网格子字典	
addLayersControls	边界层网格子字典	
meshQualityControls	网格质量控制子字典	

表 4.7 snappyHexMeshDict 字典文件的一级关键词

```

geometry
{
    sphere.stl                                // STL 文件名
    {
        type triSurfaceMesh;
        regions
        {
            secondSolid                      // STL 文件内域名
            {
                name mySecondPatch;          // 自定义 patch 名
                }                           // 未定义则指定为 sphere.stl_secondSolid
            }
        }
    }

    box1x1x1 // User defined region name
    {
        type searchableBox;               // 用户定义的方体
        min (1.5 1 -0.5);
        max (3.5 2 0.5);
    }

    sphere2 // User defined region name
    {
        type searchableSphere;           // 用户定义的球体
        centre (1.5 1.5 1.5);
        radius 1.03;
    }
};

```

5.4.2 创建六面体背景网格

在执行 snappyHexMesh 之前，用户需要创建一个充满全部区域的六面体背景网格，见图 5.9。这可以依靠 blockMesh 程序来创建。在创建背景网格的时候，它需要满足以下标准：

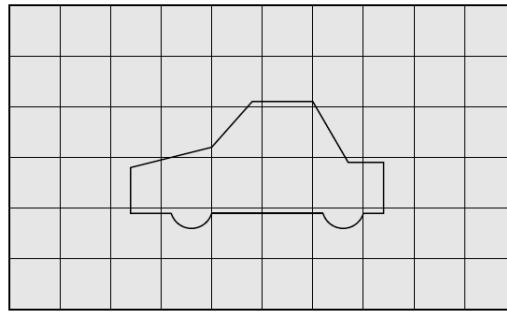


图 5.9 用于 snappyHexMesh 程序的背景网格

- 网格必须为纯六面体；
- 在随后要使用 snap 的表面附近，网格长宽高的比应该大体为 1。否则网格贴合的收敛过程会变慢，有时候还会导致失败；
- 网格单元必须至少有一个边和 STL 文件的面相交叉。例如，如果背景网格是单独的一个大网格单元是不行的；

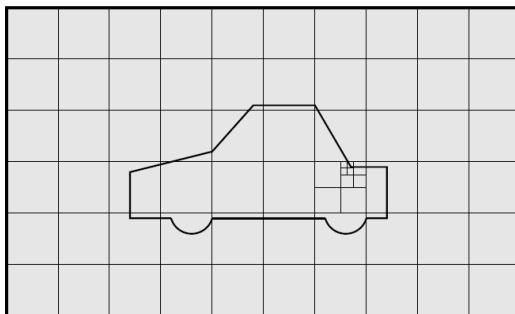


图 5.10 snappyHexMesh 程序的特征边切分过程

5.4.3 特征边和特征面网格切分

网格切分控制通过 snappyHexMesh 中的 castellatedMeshControls 子字典中的相关参数来控制。castellatedMeshControls 中相关信息参见表 5.8。

参考图 5.10，分割的过程从指定的特征边开始，在 castellatedMeshControls 中的 features 下可以指定 edgeMesh 文件以及细化的程度：level，例如：

```

features
(
{
    file "features.eMesh"; // 特征边文件
    level 2; // 细化等级
}
);

```

关键词	描述	范例
locationInMesh	网格保留域（位置矢量），这个点的位置不能和网格单元的面或边重合	(5 0 0)
maxLocalCells	细化网格过程中每个处理器处理的最大数量网格数	1e6
maxGlobalCells	移除网格之前进行网格细化的最大网格数量	2e6
minRefinementCells	如果需要细化的网格数量小于这个值，细化停止	0
nCellsBetweenLevels	不同级别细化过程中的缓冲层数量	1
resolveFeatureAngle	在面或边的弯曲角度超过这个角度的时候应用最大等级的细化	30
features	特征的细化参数	
refinementSurfaces	指定细化表面	
refinementRegions	指定细化区域	

表 5.8 snappyHexMeshDict 文件中 castellatedMeshControls 子字典的关键词

特征边可以依靠 surfaceFeatureExtract 从 STL 文件中提取。例如下面这条命令：

```
surfaceFeatureExtract -includeAngle 150 surface.stl features
```

参见图 5.11 中指定的面，在特征细化的过程中，选取某些网格单元以进行特征边细化并切分，castellatedMeshControls 中的 refinementSurfaces 字典指定需要细化的 STL 几何面以及它们的细化的最小和最大程度(<min> <max>)。几何体的表面应用最小的细化等级。当某些几何面或边的弯曲角度超过 resolveFeatureAngle 设定的角度的时候应用最大的细化等级。

细化的程度可以通过对 STL 文件某个区域进一步的指定而被覆盖。区域信息定义在 regions 子字典中。



表 5.11 snappyHexMesh 程序进行的网格切分

每个区域里的名字就是 STL 文件中定义的名字，细化程度在下一级的子字典中指定。可以参考下面的例子：

```

refinementSurfaces
{
    sphere.stl
    {
        level (2 2);           // 默认最小最大表面细化等级
        regions
        {
            secondSolid
            {
                level (3 3);   // 某域细化等级（可选）
            }
        }
    }
}

```

5.4.4 网格移除

一旦特征边和表面切分完毕，网格移除过程开始。网格移除过程要求指定一些区域，这些区域依靠完全有界的面来封闭。需要保留的区域通过一个位置矢量（点）来指定，关键词是 `castellatedMeshControls` 里面的 `locationInMesh`。如果网格单元的 50%（粗略估计）在这个区域内，这些网格单元就会被保留。其余的网格将会被移除，参见图 5.12

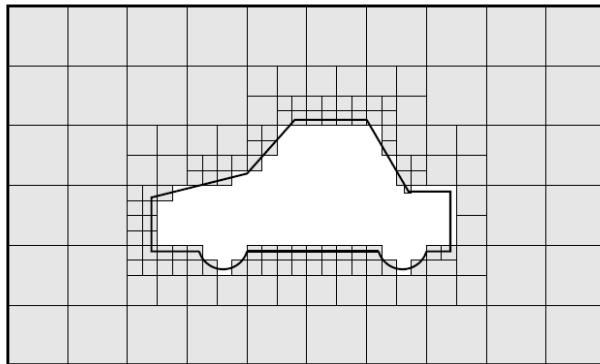


图 5.12 snappyHexMesh 的网格移除过程

5.4.5 指定区域网格细化

参见图 5.13，指定区域内的网格可以进一步进行细化。图 5.13 中的细化区域用黑色的阴影来表示。在 `castellatedMeshControls` 的 `refinementRegions` 子字典中指定需要细化区域的细化等级，这些区域的具体形状需要在之前的 `geometry` 子字典中指定。`Mode` 是细化模式，它应用于每个区域：

- `inside`: 细化区域内的网格；
- `outside`: 细化区域外的网格；
- `distance`: 通过和表面的距离来细化，通过 `levels` 关键词可以在不同的距离进行不

同的细化水平；

对于 refinementRegions，细化等级通过 levels 指定，格式为: (<distance> <level>)。在 inside 和 outside 细化模式下，并不需要<distance>，因此它被省略。请参考下面的例子：

```
refinementRegions
{
    box1x1x1
    {
        mode inside;
        levels ((1.0 4));           // 细化等级 4
    }

    sphere.stl
    {
        mode distance;
        levels ((1.0 5) (2.0 3)); // 距离 1 米内细化等级 5
                                    // 距离 2 米内细化等级 3
                                    // 从最近的距离开始指定
    }
}
```

5.4.6 表面对齐

下一个步骤就是把某些网格单元的顶点向几何对齐并贴合，以移除锯齿形网格。步骤分为：

1. 把锯齿网格的顶点移动到 STL 表面；
2. 依据新的网格点重新排布内网格点（采用松弛算法迭代处理）；
3. 寻找那些使网格质量降低的顶点；
4. 对于那些被移动的顶点，减少那些顶点的位移并重复进行第二个步骤，以确保网格质量被满足；

snappyHexMesh 里 snapControls 子字典所用的方法在表 5.9 中列出。图 5.14 有一个例子（这个例子的网格贴合虽然看起来有些不太实际）：

关键词	描述	范例
nSmoothPatch	面对应之前的面光顺迭代	3
tolerance	网格单元顶点和几何表面的点或特征面的距离/本地最大边长	4.0
nSolveIter	网格移动最大迭代数	30
nRelaxIter	网格贴合最大迭代数	5

图 5.9 snappyHexMeshDict 中的 snapControls 子字典关键词

5.4.7 网格边界层

在进行网格对齐之后，这种网格可能已经达到了计算的要求。但是它会在边界处产生一些不规则的网格单元。我们可以在边界面上增加一些依附六面体网格的附加边界层网格，参见图

5.15 中阴影部分的网格单元:

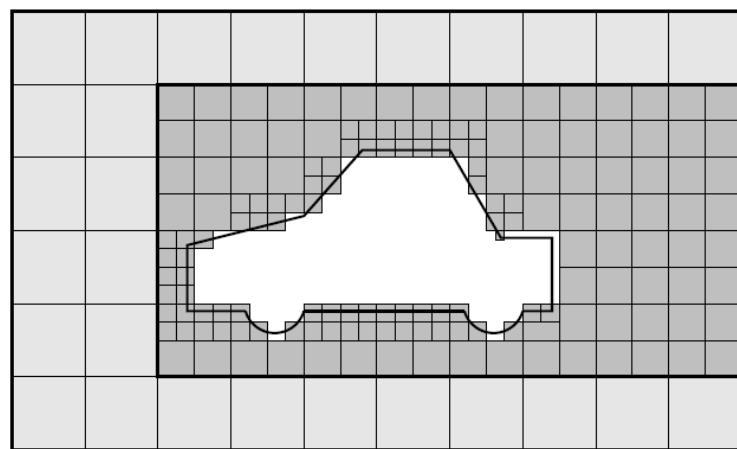


图 5.13 snappyHexMesh 的区域网格切分

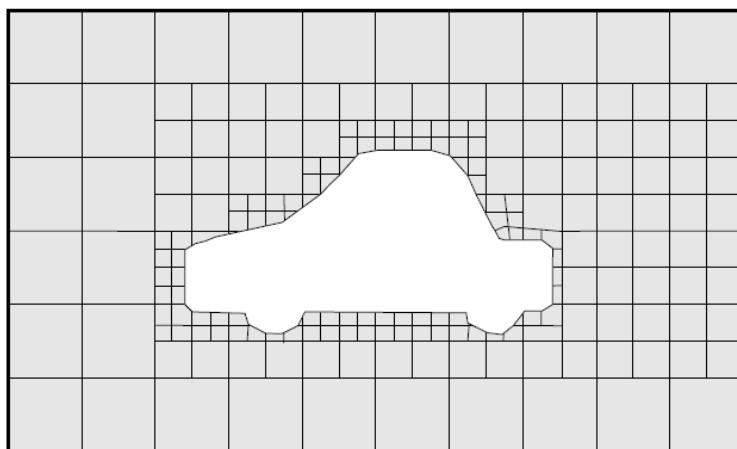


图 5.14 snappyHexMesh 的网格对齐

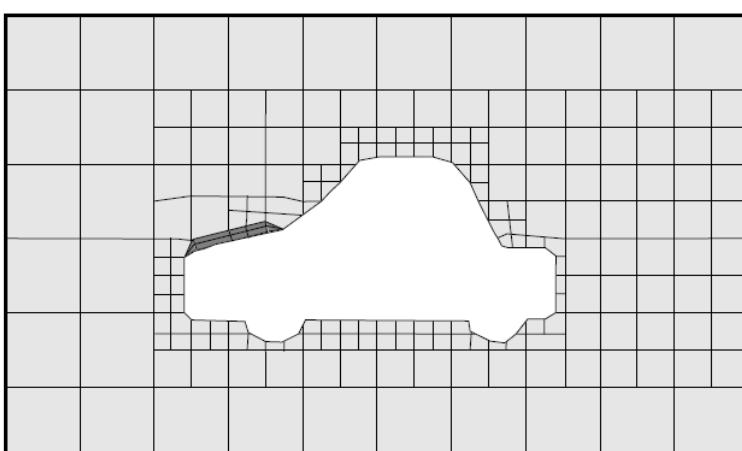


图 5.15 snappyHexMesh 的边界层添加

添加网格边界层的过程是将现有的网格单元在边界处向内收缩，然后插入边界层网格，按照如下步骤：

1. 依据一个指定的距离（边界层厚度），网格在几何面法向的方向向后投影；
2. 依据新的网格点重新排布内网格点（采用松弛算法迭代处理）；
3. 检查网格标准是否满足，没有的话减少投影量并返回到第二步。如果任何距离下都不能满足网格标准，则放弃插入边界层网格；
4. 如果网格标准满足，插入网格边界层；
5. 再次检查网格，如果不满足网格标准，去掉边界层并重新进行第二步；

边界层划分通过 snappyHexMesh 中的 addLayersControls 子字典控制，表 5.10 列举了所需要的信息，layer 子字典列举了需要添加边界层网格的几何表面，以及边界层数量：

关键词	描述	范例
layers	边界层字典	——
relativeSizes	绝对边界层厚度还是相对边界层厚度	true/false
expansionRatio	边界层网格膨胀因子	1.0
finalLayerThickness	壁面最近的边界层厚度，通过 relativeSizes 设定为相对值或者绝对值	0.3
minThickness	边界层网格的最小厚度(相对值或者绝对值,如上)	0.25
nGrow	如果没有抽取点则生成指定数量的边界以连接面，这使得特征边附近的边界层添加过程更容易收敛	1
featureAngle	边界层自动坍塌特征角	60
nRelaxIter	对齐过程的最大迭代数	5
nSmoothSurfaceNormal s	面法向光顺迭代数	1
nSmoothNormals	内部网格移动的光顺迭代数	3
nSmoothThickness	面上边界层厚度光顺数	10
maxFaceThicknessRati o	停止边界层增长的包裹网格率	0.5
maxThicknessToMedial Ratio	距离中轴比的最大距离	0.3
minMedianAxisAngle	中轴点角度	130
nBufferCellsNoExtrude	新的边界层外缓冲层数	0
nLayerIter	最大边界层添加迭代数	50
nRelaxedIter	最大网格回放迭代数，当此数越界，寻找 meshQuality 中的 relaxed 子字典的相关信息并迭代	20

表 5.10 snappyHexMesh 的 addLayersControls 中的关键词

在这里我们需要使用 patch 的名字而不是几何表面的信息，因为边界层附加与现有的网格相关。参考下面的例子：

```
layers
{
    sphere.stl_firstSolid
    {
        nSurfaceLayers 1;
    }
    maxY
    {
        nSurfaceLayers 1;
    }
}
```

关键词	描述	范例
maxNorOrtho	最大非正交角，设置为 180 关闭此标准	65
maxBoundarySkewness	最大边界网格偏斜度，0 关闭（同上）	20
maxInternalSkewness	最大内部网格偏斜度，0 关闭	4
maxConcave	最大凹度，180 关闭	80
minFlatness	最小投影面积和实际面积的比值，-1 关闭	0.5
minVol	最小棱锥体体积，-1e30 关闭	1e-13
minArea	最小网格面，<0 关闭	-1
minTwist	最小面扭曲，<1 关闭	0.05
minDeterminant	最小网格行列式值；1 为理想六面体；<=0 为负体积	0.001
minFaceWeight	0→0.5	0.05
minVolRatio	0→1.0	0.01
minTriangleTwist	>0 则和 Fluent 匹配	-1
nSmoothScale	每次回放的光顺数	4
errorReduction	网格内某些点的回放量	0.75
relaxed	当进行边界层附加的时候，如果 nRelaxedIter 越界，可以通过在本子字典指定其他的网格标准以覆盖	relaxed { }

表 5.11 snappyHexMesh 字典文件中的 meshQualityControls 关键词

5.4.8 网格质量控制

参见表 5.11，网格质量控制信息在 snappyHexMeshDict 中的 meshQualityControls 子字典中控制。

5.5 网格转换

用户可以使用其它软件包生成的网格并转换成为 OpenFOAM 可识别的格式。表 3.6 列举了可用的网格转换程序。这一章节我们讨论如何使用下面这些转换器：

- fluentMeshToFoam: 读取 fluent.msh 网格文件, 它对于 2D, 3D 算例都适用⁵⁴
- starToFoam: 读取 STAR-CD/PROSTAR 网格文件
- gambitToFoam: 读取 GAMBIT.neu 文件
- ideasToFoam: 读取 I-DEAS 网格写成 ANSYS.ans 格式
- cfx4ToFoam: 读取 CFX 网格并写成.geo 格式

5.5.1 fluentMeshToFoam

fluent 的网格采用以.msh 为后缀的单一文件。文件必须写成 ASCII 格式, 这并不是 Fluent 的默认设置。在 OpenFOAM 中也可以成功的转换 Fluent 二维的 singleStream 网格。但转换后的网格实际上是三维网格。如果我们把第三维度的前后面设置为 empty 就可以把这个三维网格当做拟 2D 算例实现。当读取一个二维的 Fluent 网格的时候, 转换器会自动在第三维度抽取网格并添加 empty 面, 并称为 frontAndBackPlanes。用户也需要了解下面一些特性:

- OpenFOAM 转换器会尽可能的捕获 Fluent 的边界条件; 然而, OpenFOAM 和 Fluent 的边界条件之间并没有一个清晰地对应, 因此用户在运行算例之前需要仔细检查边界条件;
- 建立一个轴对称的 2 维网格目前不可行, 但是这可以被植入;
- 不支持附加材料属性的网格。如果存在多个流体域, 它们会被转换成一个单一 OpenFOAM 网格文件; 如果存在固体域, 转换器会直接忽略⁵⁵;
- Fluent 允许用户定义一个网格内的内部面。比如这个实体面从属于两个网格。OpenFOAM 不允许这种操作, 如果这样的面存在, OpenFOAM 直接忽略;
- 目前不支持嵌入式界面和求精树;

要转换一个 Fluent.msh 网格文件, 首先需要创建一个文件系统; 在 system 文件夹下需要包含 controlDict 文件。然后用户需要在终端键入以下命令:

⁵⁴ 经多次试验, 译者发现 fluent3DMeshToFoam 对 3D 网格支持性更好。

⁵⁵ 在多流体域计算中, 比如 MRF 方法, 可以通过定义不同的 cellZone 来实现。

```
fluentMeshToFoam <meshFile>
```

meshFile 是网格文件名，包括其相对路径或者绝对路径⁵⁶。

5.5.2 starToFoam

这一节我们描述如何将 STAR-CD 的网格转换为 OpenFOAM 可读取的网格。网格可以使用任何 STAR-CD 的软件包来生成，例如 PROSTAR, SAMM, ProAM 以及它们的其它变种。转换器接受任何 singleStream 网格但是转换器不支持的特性有：

- multiStream 网格；
- 挡板，即流体域内零厚度的面；
- partial 边界；
- 滑移交界面；

对于 multiStream 网格，可以通过将每个单独的 stream 写为单独的网格文件，并将它们在 OpenFOAM 中进行装配。

OpenFOAM 仅仅转换那些严格符合 5.1 节描述的网格标准的网格。对于无效的网格转换器不会运行。下面的章节我们描述在使用 STAR-CD 软件包生成网格的时候，确保生成 OpenFOAM 可识别网格的具体步骤。为了防止赘述，STAR-CD 提供的网格生成器我们简称为 STAR-CD。

5.5.2.1 转换 STAR-CD 网格的一般建议

在运行 starToFoam 转换器之前，我们强烈建议用户先运行 STAR-CD 网格检查工具。starToFoam 本身也可以显示网格中的不正确信息和警告以方便用户进一步的了解网格存在的问题。在转换之后，我们建议运行 OpenFOAM 的 checkMesh 网格检查工具。对于一个 OpenFOAM 使用的网格，有问题的网格单元应该仔细检查并修复。值得一提的是对于无效的网格 OpenFOAM 不会运行，但是有些时候可以在某些其它的环境下运行，例如当网格检验标准不是很严格的时候⁵⁷。

对于那些由于匹配误差引起的问题，可以通过设定匹配误差来解决。然而，这种方法作用有限，并且，如果需要增加匹配误差这直接表明原始网格是不精准的。

5.5.2.2 消除多余数据

网格生成完毕之后，假定流体网格部分已经创建并且需要移除其他网格单元，我们接下来需要移除不需要的顶点，压缩网格单元边界，对顶点进行编号。这个通过以下 PROSTAR 命令

⁵⁶ 如果在算例文件下运行，直接运行 fluentMeshToFoam fluent.msh 即可

⁵⁷ 例如其它的 CFD 求解器

来执行：

```
CEST NEWS FLUID
CEST INVE
```

其中 CSET 应该是空的，如果这个网格不是算例的网格，检查 CSET 中的网格并调整模型。如果网格单元不符合需要的条件，它们可以用 PROSTAR 命令来移除：

```
CDEL CSET
```

同样，顶点需要去除，命令如下：

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

在去除不需要的顶点的时候，我们先需要把不需要的面收集起来然后再进行清理：

```
CSET NEWS FLUID
VSET NEWS CSET
BSET NEWS VSET ALL
BEST INVE
```

如果 BEST 不是空的，我们可以使用下面的命令移除不必要的边界面：

```
BDEL BSET
```

这样的话，模型应该仅仅包含流体网格单元以及所需的顶点和相关的边界面了。所有的边界面应该和网格的顶点相对应，如果不是这样的话，需要继续清理几何直到所有的几何清理完毕。

5.5.2.3 去掉默认边界条件

默认情况下，STAR-CD 为那些没有明确的和边界区域进行关联的边界施加壁面边界条件。余下的边界面都汇总在 default 边界区域里面，并设定为 0 边界类型。考虑到可能带来的使用错误，OpenFOAM 对这些未定义的边界面有意地不采用默认边界条件类型。这样做的原因是 OpenFOAM 不清楚那些没有关联的面是否应该被指定为默认边界条件。

因此，在网格成功转换完毕后，OpenFOAM 网格中所有的边界必须重新指定。另外，通过下面的步骤可以把 STAR 中的 default 边界转换为真实边界：

1. 使用 Wire Surface 命令参数来绘制几何；
2. 使用与 default 区域 0 相同的参数来定义一个新的边界域 10，并且将全部可见的面添加进来。这可以通过在界面操作中将区域选项开启，然后通过鼠标点选一个矩形

来包络屏幕中的全部模型来完成。也可以通过输入以下 PROSTAR 命令实现：

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

3. 接下来，我们需要从集合 (set) 中删除之前定义的全部边界类型：

```
BSET NEWS REGI 1
BSET NEWS REGI 2
...3,4,...
```

然后收集和相关边界相关的顶点，再收集和顶点相关的边界面（这里需要执行两次收集操作，因为有些对象在原始集合里面）：

```
BSET NEWS REGI 1
VSET NEWS BSET
BSET NEWS VSET ALL
BSET DELE REGI 1
REPL
```

这样，就给边界区域 10（位于边界区域 1 的顶部）指定了边界面。然后使用 BDEL BSET 删除边界区域。再重复以上操作。

5.5.2.4 模型重新编号

重新对模型编号并检查：

```
CSET NEW FLUID
CCOM CSET
VSET NEWS CSET
VSET INVE (空)
VSET INVE
VCOM VSET
BSET NEWS VSET ALL
BSET INVE (空)
BSET INVE
BCOM BSET
CHECK ALL
GEOM
```

PROSTAR 模型检查使用最后两个命令执行。这样可能会显示一些无法预料的错误。同时，还需要注意模型的缩放因子，因为 PROSTAR 只接受 STAR-CD 模型中的缩放因子，而不接受几何模型中的缩放因子。因此，如果缩放因子不是 1，那么需要在 OpenFOAM 中使用 scalePoints 工具调整缩放因子。

5.5.2.5 写入数据

一旦网格模型完成，替换模型中的全部积分匹配为耦合类型 1。其他全部类型将用于描述任意匹配：

```
CPSET NEWS TYPE INTEGRAL
CPMOD CPSET 1
```

计算域网格必须写入它们自己的文件中。可以通过 PROSTAR 命令实现：

BWRITE

执行后，在默认的情况下，会将边界写入一个后缀名为.23 的文件（3.0 版之前）或.bnd 文件（3.0 版以后）中。对于网格单元，使用如下命令：

CWRITE

网格单元信息会输出在后缀名为.14 的文件（3.0 版之前）或.cel 的文件（3.0 版以后）中。对于顶点，使用如下命令：

VWRITE

执行之后顶点信息会输出在后缀名为.15 的文件（3.0 版之前）或.vrt 的文件（3.0 版以后）中。默认的输出文件采用 ASCII 格式。如果网格中存在耦合，则需要输入一个后缀名为.cpl 的耦合文件，命令如下：

CPWRITE

以上三个文件输出完成后，退出 PROSTAR 或关闭文件。通过程序面板可以注意到，STAR-CD 中已经定义的全部子模型、材料、流体属性都需要在 OpenFOAM 中重新进行定义。

PROSTAR 文件转换步骤为：首先创建一个新的 OpenFOAM 算例文件夹，把 PROSTAR 文件存放其中。同时，我们需要修改上文输出的文件扩展名，分别是.23（或 pcs）修改为.bnd，.14（或 pcs）修改为.cel，.15（或 vtx）修改为.vrt。

5.5.2.6 .vrt 文件可能的问题

.vrt 文件的输出格式采用了指定宽度的列数据，而不是自由格式。一个典型的数据行如下，其中给出了一个顶点的坐标：

```
19422 -0.105988957 -0.413711881E-02 0.000000000E+00
```

如果坐标使用科学计数法表示并且是负数，数字之间可能没有空格，例如：

```
19423 -0.953953117E-01-0.338810333E-02 0.000000000E+00
```

starToFoam 转换程序需要使用空格来读取坐标值，因此，上面的例子无法读取。OpenFOAM 提供了一个简单的脚本程序 foamCorrectVrt 来为坐标值插入空格。通过 foamCorrectVrt 可以

把上面的例子转换为下述：

```
19423 -0.953953117E-01 -0.338810333E-02 0.000000000E+00
```

foamCorrectVrt 程序应该在 starToFoam 转换器之前运行，参见如下命令：

```
foamCorrectVrt <file>.vrt
```

5.5.2.7 转换为 OpenFOAM 可用格式

现在可以用 starToFoam 来创建 OpenFOAM 所需的边界、网格、以及点文件了：

```
starToFoam <meshFilePrefix>
```

其中 meshFilePrefix 为网格文件，需要包含全路径或者相对路径。在网格转换完成之后，需要手动更改 OpenFOAM 的 boundary 文件来设置自己的边界类型。

5.5.3 gambitToFoam

GAMBIT 网格文件的扩展名为.neu。转换 GAMBIT 网格的步骤为：首先创建一个 OpenFOAM 算例，然后在终端键入下列命令行：

```
gambitToFoam <meshFile>
```

其中 meshFile 为网格文件，需要包含全路径或者相对路径。

GAMBIT 网格文件没有包含相应的边界信息例如壁面、对称平面、周期面等。因此所有的边界都被指定为 patch，用户需要在转换后手动的进行修改。

5.5.4 ideasToFoam

OpenFOAM 可以转换 I-DEAS 生成的网格文件，以 ANSYS 格式输出后的网格后缀名为.ans。转换.ans 网格文件的首先需要创建一个 OpenFOAM 的算例，然后在终端键入以下命令：

```
ideasToFoam <meshFile>
```

其中 meshFile 为网格文件，需要包含全路径或者相对路径。

5.5.5 cfx4ToFoam

CFX 的网格文件为单一文件且后缀为.geo。CFX 的网格形式是结构块形式，即网格被指定为一系列的块，他们具备连接信息以及点位置信息。OpenFOAM 会尽可能的把他们转换并且捕获相应的边界信息。但是有些 3 维的边界定义，例如多孔介质、固体域等信息他们会在转换过程中被忽略。CFX 支持默认的 patch，在这种没有定义边界条件的 patch 中被处理为壁面。这些面在转换的过程中会被 OpenFOAM 转换器收集并处理为 defaultFaces，边界类型

为 wall；当然，用户需要在转换后手动的进行修改。

跟 CFX 的 2 维算例相同，OpenFOAM 在转换 2DCFX 网格的时候会在第三方向添加一层厚度。如果用户希望运行 CFX 可用的 2D 算例，它需要把转换后网格文件的前后面设置为 empty 类型。目前 OpenFOAM 不能转换 CFX 的 2 维轴对称几何文件。

要转换 CFX 支持的.geo 网格文件的步骤为：首先创建一个新的 OpenFOAM 算例，然后在终端键入以下命令：

```
cfx4ToFoam <meshFile>
```

<meshFile>是.geo 为后缀的文件，其中要包含文件的全路径或者相对路径。

5.6 不同几何上的投影场

mapFields 工具可以把现存几何上的一个或多个场投影到另外一个几何上的场。在这两个几何之前，完全不需要有任何的类似。然而，对于那些相同的一盒，mapFields 可以使用一个附加参数来简化投影过程。

在我们讨论 mapFields 的过程中，我们需要定义一些词语。首先，我们定义原始场为 source，被投影的场为 target。如果 source 和 target 的几何和边界类型都一样，那么我们认为这两个场是 consistent 的。mapFields 对 startFrom/startTime 中指定的时间步文件内的场来进行投影。它从原始场指定的时间步内来读取场文件，并把它投影到被投影场相应的时间步中。

5.6.1 连续场投影

连续场投影可以非常简单的执行，在被投影（target）场的目录下，使用-consistent 附加命令来运行：

```
mapFields <source dir> -consistent
```

5.6.2 非连续场投影

当几何不是连续的时候，例如图 5.16 所示。被投影场的 system 文件夹下需要一个 mapFieldsDict 文件。在投影的过程中，有以下规则：

- mapFields 会尽可能的把所有原始场数据投影到目标场中，在我们的例子中，所有的被投影场数据都是从原始场中投影过来的，除了那些阴影面积的区域⁵⁸；
- 边界的场数据不会改变，除非我们在 mapFields 中进行指定；

mapFieldsDict 文件包含两个列表，它们制定投影的 patch。第一个列表是 patchMap，它指定原始场和被投影场相吻合的 patch，如图 5.16 所示。在其中要指定原始场和被投影场 patch 对的名字。第二个列表为 cuttingPatches，它指定被投影场中的一些 patch，这些 patch 和原

⁵⁸ 此处不存在原始场数据

始场的 patch 不吻合，因此只会投影被投影场的 patch 和原始场相交的那一部分。例如在下面这个例子中，底部左边的被投影场的 patch 仅仅和原始场的内场相交。在没有相交的被投影场保持原始数据。mapFieldsDict 字典文件如下：

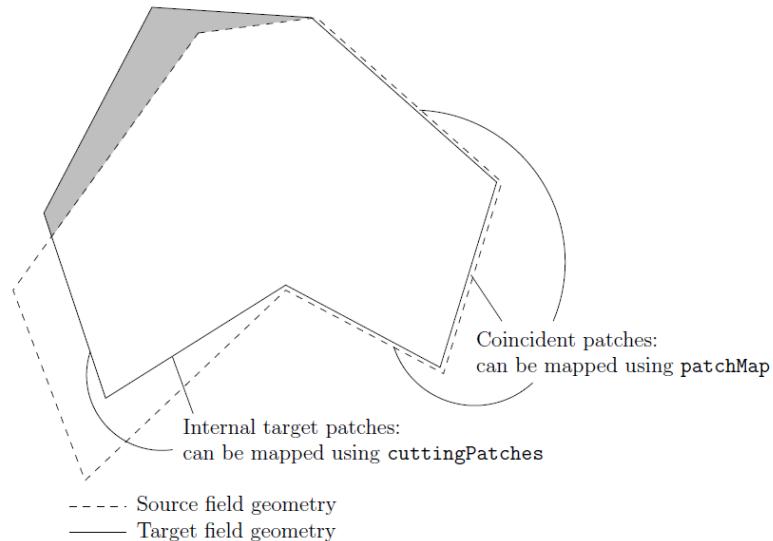


图 5.16 投影不连续场

```

17
18 patchMap      ( lid movingWall );
19
20 cuttingPatches ( fixedWalls );
21
22
23 // ****

```

mapFields <source dir>

5.6.3 并行投影

原始场和被投影场如果都需要或者某个场需要并行运算，我们需要在执行 mapFields 的时候需要添加附加的命令参数：

- **-parallelSource:** 如果原始场使用并行投影⁵⁹；
- **-parallelTarget:** 如果被投影场使用并行投影；

⁵⁹ 请使用 decomposePar 对把原始场进行分解，然后使用 mapFields <dir> -consistent -parallelSource 命令

第六章

后处理

这一章我们讨论 OpenFOAM 可用的后处理。OpenFOAM 提供了 paraFoam (依托 paraview) 来进行后处理。6.1 节我们来讨论这个开源可视化软件——paraview。其它后处理方法也有提及，例如 EnSight, Fieldview 以及 Fluent 附带的后处理软件

6.1 paraFoam

OpenFOAM 提供的后处理软件是建立在 paraview 之上的插件。插件编译成为两个库：PV3FoamReader 和 vtkPV3Foam，它们使用 OpenFOAM 提供的 paraview4.1.0。（PV3Foam - Reader 和 vtkFoam 使用的为 paraview2.x）。虽然最新的 paraview 包也可以使用，但我们建议使用 paraview4.1.0⁶⁰。关于 paraview 的更多信息请查阅 <http://www.paraview.org> 以及 <http://www.kitware.com/products/paraviewguide.html>

Paraview 使用 Visualisation Toolkit(VTK)来处理数据和渲染工具，它可以读取任何 VTK 数据。OpenFOAM 包含了 foamToVTK 程序，它可以把 OpenFOAM 的数据转换成 VTK 格式，这意味着任何能读取 VTK 的图形程序都可以对 OpenFOAM 算例来进行后处理。因此我们可以使用除了 paraview 之外的其它后处理程序。对于那些想使用更高级的并行可视化软件，可以选择 VisIt，参见：<http://www.llnl.gov/visit>

总的来说，我们建议使用 paraview 作为首选后处理软件。另外，OpenFOAM 数据也可以转换为 VTK 格式供 paraview 或者其它 VTK 图形软件使用。

6.1.1 paraFoam 概述

paraFoam 由 OpenFOAM 提供的用来运行 paraview 的插件。像任何其它的 OpenFOAM 程序一样，要么通过在算例文件夹的终端下执行单一命令，要么就使用-case 命令来对某个算例执行操作：

```
paraFoam –case <caseDir>
```

如图 6.1，paraview⁶¹会运行并打开，算例通过左面板来控制：

⁶⁰ 从译者经验来看，Paraview 最稳定的版本为 3.12。paraview4.1 在透明显示方面存在某些 bug。但是 paraview3.12 加载数量多的网格较慢。

⁶¹ OpenFOAM 的用户指南中，paraview 后处理部分并没有更新（其它章节的更新也很少），但本章节的截图和用户运行的 paraFoam 会有很大出入，但相关操作大体一致。

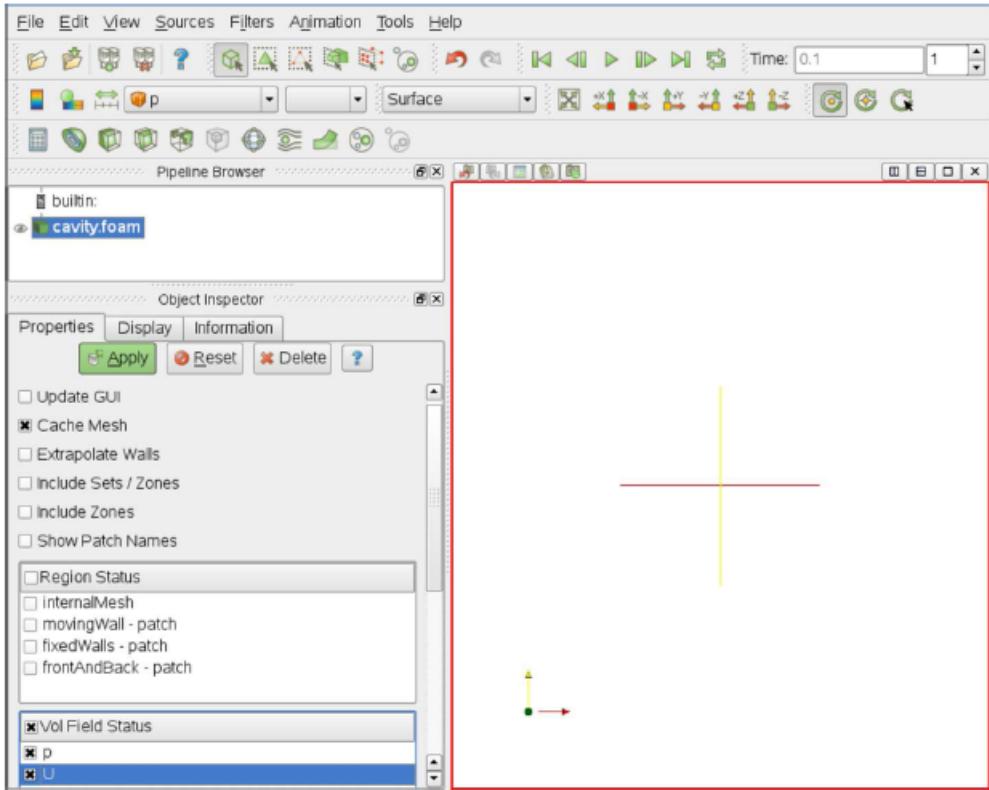


图 6.1 paraFoam 窗口

- **Pipeline browser** 面板：列举 paraview 打开的模块，当前选择的模块被蓝色高亮显示，激活前面的眼睛的标志，可以显示图形；
- **Properties** 面板：包含了算例所需要的一些必要文件和相关设置，例如时间、区域、场信息等；
- **Display** 面板：用来控制可选模块的可视化效果；
- **Information** 面板：可以显示网格几何以及大小等；

Paraview 以树状结构操作数据，用户可以对顶部的模块应用滤镜并创建子模块。例如：压力的云图可以是一个建立与包含压力场模块下的子模块。Paraview 的强大在于用户可以创建一系列的子模块并展示任何你想展示的图像或者动画。例如，用户可以在压力云图上随意添加一些单纯的几何文件，网格，或者速度矢量，并且在任何时候随开随关。

通常的操作为用户做一些操作然后点击 Properties 面板中的绿的 Apply 按钮。其它的按钮是：Reset 开关（在需要的时候可以重启 GUI）；Delete 按钮可以删除模块。

6.1.2 Properties 面板

Properties 面板包含时间步、区域、场设置等相关信息，参见图 6.2。值得注意的是

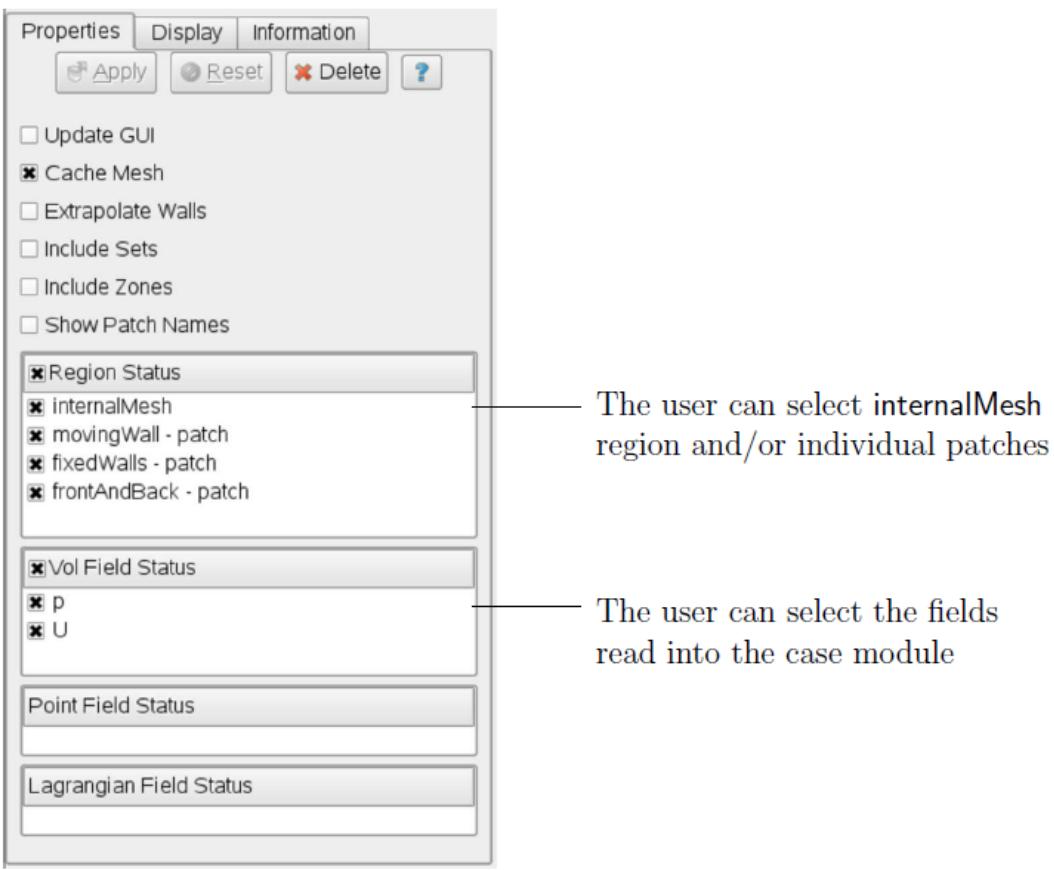


图 6.2 Properties 面板

在当前的模块中，所有时间步的数据都加载在 paraview 中（在 paraview2.x 的模块中，它用一系列的 check boxes 控制展示的时间）。默认情况下，current Time Controls 以及 VCR Controls 工具栏都已经显示，它们展示的时间步数据以及相关控制。详见 6.1.4 节。

任何 paraFoam 中的操作，做的任何选择或者改变，用户都需要点击 Apply。Apply 按键被高亮为绿色以提醒用户已发生改变但没有应用。这样，用户就可以做一系列操作但不应用它们，这对大型算例是非常好的，可以减少数据处理负担。

有时数据文件会被更改并且 paraview 需要读取它们，例如一个场数据被写入了新的时间步。为了读取这些更改，用户应该点击 Properties 面板顶部的 update GUI⁶²按钮并应用。

6.1.3 display 面板

Display 面板包含了对于一个给定的算例所需的可视化设置。下面几点是非常重要的：

- 对于数据场的最大值以及最小值，并不会被自动加载，用户应该在合适的情况下，特别是载入初始算例的时候选择 Rescale to Data Range；

⁶² 在 paraview 最新版本中，update GUI 按钮已经缺失，用户如果想让 paraview 读取新的时间步，可以点击 refresh 按键，或者删除目前模块，重新载入。

- 点击 Edit Color Map 按钮，会开启一个新窗口，包含两个面板：

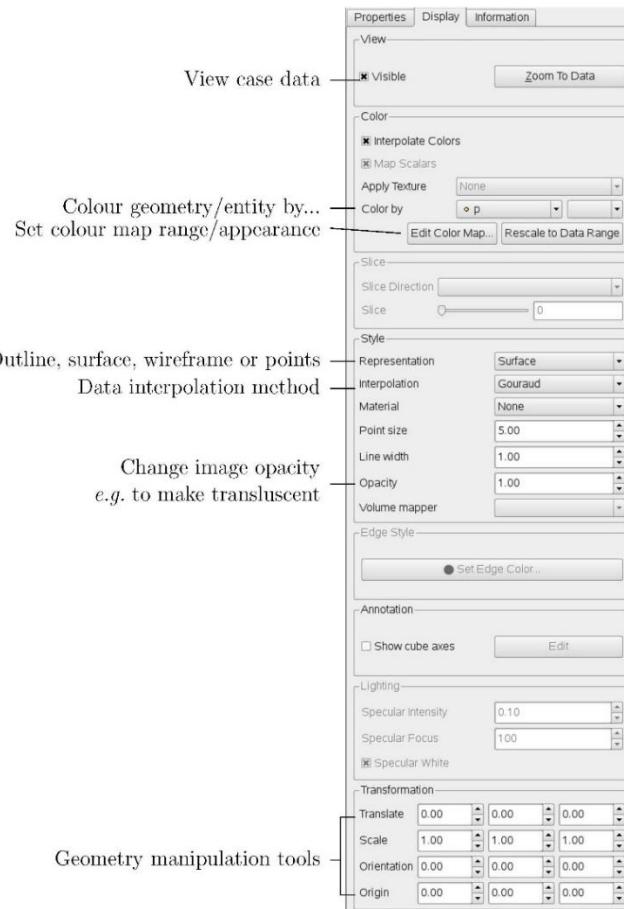


图 6.3 Display 面板

1. 色系的选择可以在 Color Scale 面板中选择。可以这样来选择标准的 CFD 红绿色系：点击 Choose Preset，然后选择 Blue to Red Rainbow HSV；
 2. Color Legend 面板有一个开关按钮，它用于设置色带以及色带布局控制，例如字体；
- 网格可以通过 WireFrame 来显示，它在 Style 面板的 Representation 菜单下进行选择；
 - 几何部分，例如网格（如果选取的是 wireframe），可以单色显示，通过在 Set Ambient Color 菜单中的 Color By 选项中的 Solid Color 来调节；
 - 图像可以通过 Style 面板中的 Opacity 工具来实现透明化处理，0 为全透明，1 为不透明；

6.1.4 工具栏

Paraview 将菜单栏以及面板中的几个功能整合在了一起并放置在工具栏中，位于菜单栏之下。用户可以在 view 菜单下的 Toolbars 中展示或隐藏这些工具栏。图 6.4 中是默认的布局设置，它们的每个名称已经标出。它们的功能可以通过图标的样子来分辨，在 Help 菜单中，也包含了它们的使用提示，用户可以查阅任何功能按钮的简要描述来获取相关信息。

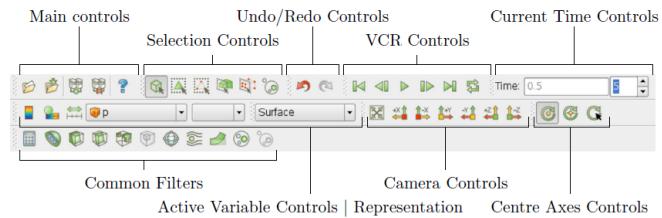


图 6.4 paraview 的工具栏

6.1.5 效果展示

这一章节描述如何在 paraFoam 中进行设置以及如何对算例进行可视化操作。

6.1.5.1 View 设置

在 Edit 菜单里面有 View Settings 设置⁶³，这会打开一个 View Settings(Render View)窗口，里面包含 3 个子项：通用 (General)，灯光 (lights)，注释 (Annotation)。通用面板里面包含了下述内容，它们最好在后处理之前就设置好：

- 背景色，其通常选为白色，这可以通过 choose color 旁的 down-arrow 来选择背景色 (background)；
- 对于 2D 的算例，通常选择 Use parallel projection；

Lights 面板中的 light kit 包含了详细的灯光颜色控制。Headlight 面板控制图像的亮度。通过调节 Headlight 工具条，可以看出设置强度为 1，会使图像更加明亮，这对等值面的展示很有帮助。

Annotation 面板包含图像设置接口。Orientation Axes 控制坐标轴的显示，例如设置坐标轴名称 x, y, z 的颜色。

6.1.5.2 常规设定

在 Edit 菜单栏中，有 Settings 均一设定选项，这包含一些常规选项例如 General, Colors,

⁶³ 最新版本的 paraview 中，软件编排在这里区别很大。

Animations, Charts 以及 Render View 菜单项。

General 面板控制 Paraview 的默认行为。特别地是，这里有一个 Auto Accept 按钮，能使 Paraview 自动的应用，而不需要每次做出操作之后再点击绿色的 Apply 按钮来应用。大部分情况下，我们不建议开启此选项。因为用户并不想在做一系列操作的时候中途每次都应用，用户只想在完成一系列操作之后应用一次。

Render View 面板包含 3 个子项：General, Camera, Server。General 面板包含了 level of detail (LOD) 控制，他们用来控制图线操作以及渲染，例如转换、重置大小、旋转、重置图片分辨率等。通过合理的进行设置，用户可以在数量巨大的网格算例中快速渲染。

Camera 面板包含了 2D 以及 3D 的鼠标移动选项。它定义了鼠标左右中键的操作来实现旋转，转换，放大控制。用户可以对其进行自定义操作。

6.1.6 云图绘制

云图可以通过顶部工具栏的 Filter 菜单下的 Contour 滤镜来实现。滤镜只能在一个指定的模块中应用，如果是一个 3D 的模块，云图就是一个 2 维的面，表示一个等值量，例如等值面。云图的 Properties 面板可以用来设定一系列等值面的值。在 New range 窗口内可以查看这些值。在下拉菜单中可以选择需要处理的场⁶⁴。

6.1.6.1 剖面

用户可能想在一个平面创建等值线而不是创建等值面。这样的话，用户应该首先用 Slice 创建一个需要创建云图的剖面。Slice 滤镜允许用户指定一个切割面、盒子、球（指定中心和半径）。用户可以通过鼠标来操作剖面。用户也可以在剖面上运行 Contour 滤镜来创建云图等值线。

6.1.7 矢量图绘制

矢量图可以通过 Glyph 滤镜来绘制。滤镜对选取的 Vectors 场进行读取，在绘制的过程中，有一系列 Glyph Types (矢量类型) 可以选择，Arrow 可以提供一个清晰地矢量图。通过这些矢量图形控制，用户可以实现最好的视觉效果。

Properties 面板中的其它内容主要包括 Scale Mode 菜单，通常我们选择为 Vectors，在这种情况下矢量长度和矢量大小成比例。Off 选项会使所有的矢量具有相同的长度。Set Scale Factor 可以控制矢量箭头的长度。

6.1.7.1 在网格中心绘制

默认情况下，矢量在网格顶点生成，但我们经常想在网格中心生成矢量。首先，我们在选定模块上应用 Cell centers 滤镜，然后应用 Glyph 矢量。

⁶⁴ 根据版本的不同，有的 paraFoam 为 Value range，下拉菜单在 contoured by 右侧

6.1.8 流线图

流线图首先应该通过 Stream Tracer 滤镜来创建示踪轨迹。示踪 Seed 面板指定示踪点，它们分布在一个 Line source 上或者 Point Cloud 上。用户可以查看示踪源，比如一条线。它们通常显示为白色，因此它们需要通过改变背景颜色来让它更清楚。

在 Stream Tracer 面板中，用户可以指定流线覆盖的范围、流线的长度。创建一个自己想要的美观的、数量合适的流线图需要凭借用户自己的经验。使用光顺器来进行光顺会引起误差但是会看起来更加平滑，这也会增加计算时间。

一旦流线创立，用户可以使用 tube 滤镜并把它应用在 tracer 模块上用来创建高质量的流线。这些流管跟流线重合，有的时候它们并不是非常的圆。但是有固定的边数以及一个固定半径。当边被设定之后，比如 10，流管看起来像是圆的，越圆的流管，越需要耗费资源计算。

6.1.9 图形输出

从 paraview 输出图形最简单的方式就是在 File 菜单下选择 Save Screenshot。选择后，一个窗口会显示出来，用户可以选择输出图形的分辨率。旁边还有一个按钮，点击之后会锁定图形的宽高比。因此如果用户在一个方向上改变了分辨率。那么另一个方向的像素会自动变化。在选择分辨率之后，图像可以保存。为了达到高质量的输出，用户可以把 x 方向的像素分辨率设为 1000 或者更高，这样当图像放大在 A4 或者 US letter 文件，或者 PDF 中的时候，图像会非常锐利。

6.1.10 动画输出

创建动画，用户可以首先在 File 菜单中选择 Save Animation。会出现一个对话框，用户可以指定一系列例如分辨率等的相关参数。用户需要按需指定分辨率。其它的设定是每个时间步的帧数。一般这个都是 1，但可以设置更大的数值以人工加帧。这个技术在创建慢动画的时候非常有用，因为有些动画播放器的速度控制功能有限，尤其对 mpeg 格式。

点击 Save Animation 之后，另一个窗口会出现，用户可以指定文件名，比如 root，以及图片格式，点击 OK 之后，会生成一系列文件，文件名标准为：“<路径>_<图片编号>.<后缀名>”，例如文件名为 animation 的第三个文件，如果以 jpg 格式保存的话，就是“animation_0002.jpg”。图片起始于编号 0000。

一旦保存这一系列图形之后，用户可以使用软件把它们转换成动画。ImageMagick 中的 convert 命令可以这样来执行⁶⁵：

```
convert animation*.jpg movie.mpg
```

在创建 mpg 文件的时候，我们可以调节相应的默认设置来增加质量，例如设置 -quality 90% 可以减少默认设置可能带来的噪点。

⁶⁵ 需要另行下载

6.2 使用 Fluent 来后处理

OpenFOAM 运行的算例可以使用 Fluent 来进行后处理。我们有两个程序可以用来实现这样的功能，foamMeshToFluent 可以把 OpenFOAM 的网格转换为 Fluent 可识别的格式并输出为后缀为.msh 的网格文件。foamDataToFluent 可以把 OpenFOAM 结果转换成 Fluent 可读的.dat 文件。foamMeshToFluent 可以在终端直接执行。网格文件在当前目录下的 fluentInterface 子文件夹中生成，例如<caseName>/fluentInterface/<caseName>.msh。

foamDataToFluent 可以把 OpenFOAM 数据转换为 Fluent 可读取的格式，转换需要两个文件。首先，在 controlDict 字典中，需要指定 startTime，它告诉程序需要转换的结果文件。如果你需要转换最新的计算结果，就把 startFrom 设置为 latestTime。转换所需的第二个文件就是 foamDataToFluentDict 字典文件，其位于 constant 文件夹下。如下：

```

1 / *----- C++ -----*/
2 | =====
3 | \  / Field      | OpenFOAM:   The Open Source CFD Toolbox
4 | \| / Operation | Version:    2.3.0
5 | \| / And        | Web:        www.OpenFOAM.org
6 | \| / Manipulation |
7 \*-----*/
8 FoamFile
9 {
10    version 2.0;
11    format ascii;
12    class dictionary;
13    object blockMeshDict;
14 }
15 // ****
16
17
18    p           1;
19
20    U           2;
21
22    T           3;
23
24    h           4;
25
26    k           5;
27
28    epsilon     6;
29
30    alpha1      150;
31 // ****

```

字典中的相关信息采取以下格式：

<fieldname> <fluentUnitNumber>

<fluentUnitNumber>是 fluent 后处理器可识别的标示符。基本的<fluentUnitNumber>详见表 6.1。字典必须包含用户想进行后处理的所有信息，例如在我们的例子中我们有压力 p 和速度 U 的信息。表 6.1 是默认的一些信息。用户可以像任何程序一样来执行 foamDataToFluent。

为了使用 Fluent 来查看结果，切入 fluentInterface 目录，通过下面这个命令行来运行一个三维的 Fluent 程序⁶⁶：

fluent 3d

这时网格和数据都被加载并可视化。网格可以通过 File 菜单下的 Read Case 来读取，其中相

⁶⁶ 本手册为 linux 下的 fluent

关内容应该被选择。

Fluent 名	序号	对应的 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 Fluent 中的后处理编号

例如为了读取 k 和 epsilon, 用户应该在 Define-Models-Viscous 菜单下选择 k-epsilon。这样, 在 File 目录下, 通过选择 Read Data 就可以读取数据。需要注意的是, 用户不能使用这两种组合: 即原始的 Fluent 网格, 和由这个原始网格转换为 OpenFOAM 网格后由 OpenFOAM 求解后的结果 (Fluent 格式)。因为两个数据的文件序号不一致。

6.3 使用 Fieldview 后处理

OpenFOAM 可以使用 Fieldview 软件来进行后处理。这需要运行一个后处理程序: foamToFieldview 来把 OpenFOAM 的结果转换为 Fieldview 的.uns 文件格式。对于一个算例, foamToFieldview 可以像任何一个可执行程序一样来执行。它创建一个文件夹叫 Fieldview, 其位于算例文件夹之下, 并删除所有已经存在的 Fieldview 文件。默认情况下, 转换器读取所有时间步的数据并写一系列文件, 文件名采用<case>_nn.uns 的格式。nn 是从 1 开始的计数指标, 第一个时间步为 1, 第二个为 2, 以此类推。如果用户打算转换某一个时间步, 可以采取以下附加命令参数: -time<time>。

Fieldview 提供了一些函数, 这些函数需要一些边界条件的信息。例如采用壁面边界信息来绘制流线图。默认条件下, 转换器尽可能的转换所有信息。用户可以在终端执行-noWall 附加命令来屏蔽包含的壁面信息。就像之前所说, Fieldview 的数据文件扩展名为.uns。如果原始的 OpenFOAM 算例包含一个点“.”, 在转换多个时间步文件为单一文件的时候 Fieldview 可能会出现问题。

6.4 使用 EnSight 进行后处理

OpenFOAM 算例可以使用 EnSight 来进行后处理, 它可以采用以下两种方法:

- 使用 foamToEnSight 程序把 OpenFOAM 的数据转换为 EnSight 格式;

- 使用 ensight74FoamExec 插件直接读取 OpenFOAM 数据;

6.4.1 转换数据为 EnSight 格式

foamToEnsight 把 OpenFOAM 数据转换为 Ensight 的文件格式。对于一个给定的算例，foamToEnsight 可以像其它程序一样来执行。foamToEnsight 在算例目录下创建一个 Ensight_Case 文件夹并删除之前存在的 Ensight 文件。然后，转换器读取所有时间步的数据并写数据⁶⁷。每个数据文件都采用 Ensight_nn.ext 的规则来命名。nn 是从 1 开始的计数指标，第一个时间步为 1，第二个为 2，以此类推。ext 是文件扩展名，依据场信息的不同而命名。例如 T 就是温度场，mesh 就是网格。一旦数据转换完毕，EnSight 可以通过以下步骤读取数据：

1. 从 EnSight 的图形界面读取数据：用户应该在 File 菜单下选择 Data(Reader);
2. 在 File 窗口读取数据，并高亮显示 EnSight_Case;
3. 依据 EnSight 的默认设置，Format 选择器应该设定为 Case;
4. 然后用户点击 Case 并 Okay;

6.4.2 ensight74FoamExec 插件

除了读取 EnSight 的标准格式数据，EnSight 还可以使用用户自定义插件来读取其它格式的数据。OpenFOAM 自带了供其使用的 ensight74FoamExec 插件，它被编译进入 OpenFOAM 库并成为 libuserd-foam 库。EnSight 可以使用这个库，这意味着它在现存的文件系统下必须能够调用这个库。详见下述：

6.4.2.1 EnSight 用户插件

为了运行 EnSight 插件，需要设置一些环境变量。我们可以这样进行设置：在 \$WM_PROJECT_DIR/etc/apps/ensightFoam 文件夹中的 bashrc 文件中进行设定。EnSight 有关的环境变量以 \$CEI 或者 \$ENSIGHT7_ 为前缀，其它的详见表 6.2。在标准的设置下，只有 \$CEI_HOME 需要手动设置 EnSight 的安装路径。

6.4.2.2 使用用户自定义插件

使用 EnSight 插件来处理数据的主要困难在于 EnSight 希望算例的数据是一个单独的文件，而不是 OpenFOAM 那种各自的目录。因此在下面的叙述中，用户应该特别注意算例的选择，因为 EnSight 不允许选择文件夹。

1. 从 EnSight 的图形界面开始，用户应该在 File 菜单下选择 Data(Reader);

⁶⁷ 原文此处有重复，故省略

2. 用户在 Format 菜单下选择 OpenFOAM; 如果没有的话, 就是设置有问题, 请重新设置环境变量;

环境变量	描述与选项
\$CEI_HOME	EnSight 的安装路径例如/usr/local/ensight, 默认添加
\$CEI_ARCH	平台架构, 和\$CEI_HOME/ensight74/mach-ines 中的默认设置 (linux_2.4, sgi_6.5_n32) 相对应
\$ENSIGHT7_READER	Ensight 寻找自定义 libuserd-foam 库的路径, 缺省设置为\$FOAM_LIBBBIN
\$ENSIGHT7_INPUT	默认设置为 dummy

表 6.2 EnSight 环境变量设置

3. 用户应该在 File Selection 窗口下找到它们的算例, 并把 Directories 的控件高亮, 点击 (Set) Geometry;
4. 现在路径场应该包含了算例的相关信息, 注意:(Set)Geometry 下应该包含一个/号;
5. 用户可以点击 Okay, EnSight 会开始读取数据;
6. 当数据读取的时候, 会出现一个 Data Part Loader, 问你读取哪些数据。用户应该选择 Load all;
7. 在 EnSight 视窗中, 当展示网格的时候, 用户应该关闭 Data Part Loader 窗口, 因为有些 EnSight 的特性, 在这个窗口打开的时候不会展示;

6.5 提取数据

OpenFOAM 提供了一个 sample 程序用来提取数据, 要么在一条线上、要么在一个 2D 平面上进行提取。具体的提取位置设定通过 system 文件夹下的 sampleDict 来进行设置。数据可以写成一系列的格式, 并可以使用其它著名的制图软件如 Grace/xmgr, gnuplot, 以及 jPlot 来进行读取。

用户可以在源代码\$FOAM_UTILITIES/postProcessing/sampling/sample 下拷贝 sampleDict 字典文件。\$FOAM_TUTORIALS/solidDisplacementFoam 中的 plateHole 算例包含了一个提取 1 维线场信息的例子:

```

17
18 interpolationScheme cellPoint;
19
20 setFormat      raw;
21
22 sets
23 (
24   leftPatch

```

```

25  {
26      type      uniform;
27      axis      y;
28      start    ( 0 0.5 0.25 );
29      end      ( 0 2 0.25 );
30      nPoints  100;
31  }
32 );
33
34 fields   ( sigmaxx );
35
36
37 // ****

```

关键词	选项	描述
interpolationScheme	cell	网格单元值，默认为整个网格的值都一样
	cellPoint	网格单元值和顶点值
	cellPointFace	网格单元值、顶点值、面值
setFormat	raw	按列写入的 ASCII 原始格式
	gnuplot	gnuplot 格式
	xmgr	Grace/xmgr 格式
	jplot	jPlot 格式
surfaceFormat	null	无输出
	foamFile	point、face、values 文件
	dx	DX 标量或矢量格式
	vtk	VTK ASCII 格式
	raw	xyz 的值，可调用 gnuplotsplot
	stl	ASCII STL 面文件
fields	需要提取的场，例如速度 U	
	U	写入 U 的分量
sets	查阅表 6.4 相关的子字典文件（1 维）	
surfaces	查阅表 6.5、6.6 相关的面子字典文件（2 维）	

表 6.3 sampleDict 的关键词信息

字典应该包含以下信息⁶⁸:

interpolationScheme	数据的差值格式
sets	提取场数据的 1D 线的具体位置
surfaces	提取场数据的 2D 面的具体位置
setFormat	线数据输出格式
surfaceFormat	面数据输出格式
fields	需要提取的场

插值格式包括 cellPoint 和 cellPointFace 等，进行插值的时候每个多面体网格被分解为四面体，提取的值从四面体顶点插值而来。在 cellPoint 下，四面体顶点包括多面体的网格单元中心和面的三个顶点。和网格单元中心重合的顶点的场值使用当前的值，其他的顶点值从网格

⁶⁸ 格式有变以使得文档和英文版相对应

单元中心值插值而来。对于 cellPointFace，其中和面心重合的四面体的顶点使用插值格式插值（面心所在的网格单元值）而来。

2D 线场提取中的 setFormat 信息包括了原始数据格式和可供 gnuplot、Grace/xmgr、jPlot 可用的数据格式。数据写入到算例文件的 sets 文件中。文件被切分为不同的时间文件夹来储存。每个文件夹中包含了相应的数据。每个数据文件的名字都以场信息的名字来指定，程序本身就设定好了它们的名字以及跟输出格式有关的后缀名。例如原始数据的后缀为.xy，Grace/xmgr 的后缀为.agr，jPlot 的后缀为.dat。gnuplot 格式跟原始数据相同，但是它还生成一个以.gplt 为后缀的文件用来生成图形。需要注意的是，在运行中进行 sample 的时候，sets 中之前存在的所有数据都会被删除。

面信息提取所需的 surfaceFormat 信息也包括原始数据格式以及供 gnuplot、Grace/xmgr、jPlot 可用的数据格式。数据被写入到算例的 surfaces 文件夹中。它们被分为不同的时间步。文件名信息跟线数据提取类似。

用户需要提取的场信息应该在 fields 中设定。sample 程序可以分析下面的函数并使用户可以操作向量场和张量场。例如对于 U：

- U.component(n): 写入矢量/张量的第 n 个分量， $n=0,1,\dots;$
- Mag(U): 写入矢量/张量的模；

sets 下也同样包含一系列子字典，指定从哪里提取数据。它们的名字依据集的特性来命名。详见表 6.4。它提供了可提取数据的位置信息，例如，uniform 提取方法中的点在起始点和终止点中间均匀分布。所有提取集都需要指定一个 type，和一个 axis，通过 axis 来指定线的方向。

		所需信息						
提取类型	提取位置	名字	轴	起始点	终止点	点数	点？	
uniform	线上均一分部	✓	✓	✓	✓	✓		
face	指定线和网格单元面的交叉	✓	✓	✓	✓			
midpoint	线—面交叉的中点	✓	✓	✓	✓			
minPointAndFace	minPoint 和 face 的结合	✓	✓	✓	✓			
curve	曲线上的指定点	✓	✓				✓	
cloud	指定点	✓	✓					✓

关键词	描述	选项范例	
type	提取类型	如上	
axis	输出提取位置	x	x 轴
		y	y 轴
		z	z 轴
		xyz	xyz 轴
		distance	从 0 点开始的距离
start	起始点	(0.0 0.0 0.0)	

end	终止点	(0.0 0.0 0.0)
nPoints	点数量	200
points	点列表	

表 6.4 sets 子字典的相关信息

surfaces 里面也包含一系列子字典，指定从哪里提取数据。每个数据文件的名字都以场信息的名字来指定并包含一系列类型，例如 type 或者是 plane：

关键词	描述	选项范例
basePoint	面上的点	(0 0 0)
normalVector	面法向矢量	(1 0 0)
interpolate	插值数据？	true/false
triangulate	三角形？	true/false

表 6.5 surfaces 子字典中的 plane 信息

关键词	描述	选项范例
patchName	patch 信息	movingWall
interpolate	插值数据？	true/false
triangulate	三角形？	true/false

表 6.6 surfaces 子字典中的 patch 相关信息

这些关键词依靠附加的一个子字典中的点和法向来定义，详见表 6.5。它也有可能是一个 patch，这个 patch 和现存的边界面对应，它依靠附加的一个子字典来定义，详见表 6.6。

6.6 监控和管理进程

这一节我们讨论 OpenFOAM 的求解器进程并拓展一下求解器的基本操作。求解器的描述请参阅 3.3 节。如果在\$WM_PROJECT_DIR/etc/controlDict 下的调试开关的调试等级被设置为 1 或者 2，那么当一个求解器被执行的时候，它会输出（例如输出在当前屏幕）方程的求解状况。教程中的 cavity 算例的求解状况可以作为一个例子并参见如下信息。对于每一个求解的方程，报告会输出所使用的矩阵求解器、求解变量、初始残差、最终残差以及迭代数。

```

Starting time loop

Time = 0.005

Max Courant Number = 0
BICCG: Solving for Ux, Initial residual = 1, Final residual = 2.96338e-06, No Iterations 8
ICCG: Solving for p, Initial residual = 1, Final residual = 4.9336e-07, No Iterations 35
time step continuity errors : sum local = 3.29376e-09, global = -6.41065e-20, cumulative = -6.41065e-20
ICCG: Solving for p, Initial residual = 0.47484, Final residual = 5.41068e-07, No Iterations 34
time step continuity errors : sum local = 6.60947e-09, global = -6.22619e-19, cumulative = -6.86725e-19

```

```

ExecutionTime = 0.14 s

Time = 0.01

Max Courant Number = 0.585722
BICCG: Solving for Ux, Initial residual = 0.148584, Final residual = 7.15711e-06, No Iterations 6
BICCG: Solving for Uy, Initial residual = 0.256618, Final residual = 8.94127e-06, No Iterations 6
ICCG: Solving for p, Initial residual = 0.37146, Final residual = 6.67464e-07, No Iterations 33
time step continuity errors : sum local = 6.34431e-09, global = 1.20603e-19, cumulative = -5.66122e-19
ICCG: Solving for p, Initial residual = 0.271556, Final residual = 3.69316e-07, No Iterations 33
time step continuity errors : sum local = 3.96176e-09, global = 6.9814e-20, cumulative = -4.96308e-19
ExecutionTime = 0.16 s

Time = 0.015

Max Courant Number = 0.758267
BICCG: Solving for Ux, Initial residual = 0.0448679, Final residual = 2.42301e-06, No Iterations 6
BICCG: Solving for Uy, Initial residual = 0.0782042, Final residual = 1.47009e-06, No Iterations 7
ICCG: Solving for p, Initial residual = 0.107474, Final residual = 4.8362e-07, No Iterations 32
time step continuity errors : sum local = 3.99028e-09, global = -5.69762e-19, cumulative = -1.06607e-18
ICCG: Solving for p, Initial residual = 0.0806771, Final residual = 9.47171e-07, No Iterations 31
time step continuity errors : sum local = 7.92176e-09, global = 1.07533e-19, cumulative = -9.58537e-19
ExecutionTime = 0.19 s

```

6.6.1 运行进程的 foamJob 脚本

用户可能喜欢在屏幕上监控残差、迭代数、库郎数的变更。另一种方法是，用户可以把输出的内容写入 log 文件来提高计算速度。foamJob 脚本可以完成这样的工作，它在执行的时候需要指定专门的求解器来进行计算，然后把输出流写入 log 文件：

```
foamJob <solver>
```

其它选项用户可以执行 `foamJob -help` 来查看。用户可以通过 `UNIXtail` 命令来随时监控 log 文件，一般我们指定-f，这样会显示最新的 log 文件信息：

```
tail -f log
```

6.6.2 监控进程的 foamLog 脚本

仅仅读取 log 文件来监控进程是具有局限性的，特别是它很难展现随时间变化的趋势。foamLog 脚本可以在 log 文件中抽取数据的相关信息例如残差、迭代数、库郎数，并把它们写入一系列文件以供绘制曲线。foamLog 脚本可以这样来执行：

```
foamLog <logFile>
```

这些文件存储在算例文件夹下的 logs 子文件中。每个文件以<var>_<subtler>来命名，其中<var>是 log 中指定的变量，<subtler>是每个时间步内的迭代数。求解变量的最初残差使用<var>来命名，最终残差使用<var>FinalRes 来命名。默认条件下，文件内是两列，一列是时间，一列是数据。

例如，在 cavity 教程中，我们想观察 Ux 方程的最初残差来观察是否求解达到稳态。在这个例子中，我们想从 logs/Ux_0 文件中来绘制类似图 6.5 这种的图形。可以看出残差一直

在下降，直到 10^{-5} 。

在 log 文件中的各项数据，foamLog 都会尽可能的对其进行处理。例如在 cavity 算例教程中，这包括：

- 库郎数，Courant_0；
- Ux 方程的初始和最终残差，Ux_0 和 UxFinalRes_0，迭代数 UxIters_0 (Uy 也是同样的内容)；
- 在 2 个 P 方程求解之后的累积、总体、局部连续性误差 contCumulative_0, contGlobal_0, contLocal_0 和 contCumulative_1, contGlobal_1, contLocal_1；
- 两个 p 方程中的残差和迭代数 p_0, pFinalRes_0, pIters_0 和 p_1, pFinalRes_1, pIters_1；
- 执行时间，executionTime；

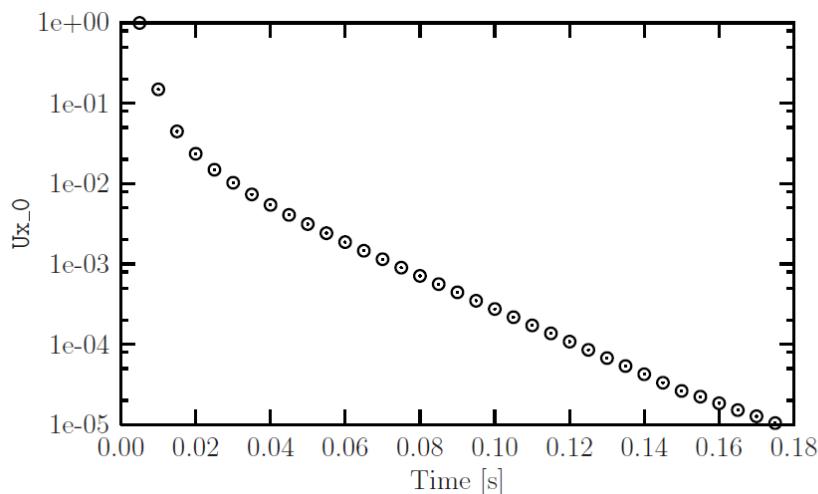


图 6.5 cavity 算例 Ux 的初始残差

第七章

模型和物理特性

OpenFOAM 包含了很多针对具体问题的求解器。每个求解器所用的方程和算法都不相同。因此用户需要在最初的时候针对不同的例子，对使用哪些模型进行决定。可选择的求解器在表 3.5 中列出。求解器决定了这个算例需要定义的物理参数，其中有一些模型用户可以在运行之后，通过 `constant` 字典中的字典文件进行更改。这一章节我们讨论常见的模型以及运行之前需要定义的参数。

7.1 热物理模型

热物理模型跟能量、热能等物理特性有关。使用 `thermoPhysical` 模型的求解器会读取 `thermophysicalProperties` 字典。OpenFOAM 中，热物理模型通过压力-温度体系而建立，通过这个体系来求解其它的物理参数值。一个必须设定的关键词是 `thermoType`，它指定运行中的热物理模型。热物理模型从第一层开始，这一层定义了基本的状态方程。然后在这一层的基础上添加附加模型。表 7.1 中 `thermoType` 列表反映了这种多层模型结构：

状态方程—— <code>equationOfState</code>	
<code>adiabaticPerfectFluid</code>	绝热气体状态方程
<code>icoPolynomial</code>	不可压缩状态多项式方程（液体）
<code>perfectFluid</code>	气体状态方程
<code>incompressiblePerfectGas</code>	参考压力固定的不可压缩气体状态方程，密度随温度和组分改变
<code>rhoConst</code>	常密度状态方程
基本热物理模型—— <code>thermo</code>	

eConstThermo	常热导率模型，内能 e、熵 s 由此计算
hConstThermo	常热导率模型，内能 e、焓 h 由此计算
hPolynomialThermo	从附带一系列参数的多项式来计算 C_p ，进而用于计算 h 和 s
janafThermo	从使用 JANAF 热力表的函数计算 C_p ，进而用于计算 h 和 s
衍生热物理模型——specieThermo	
specieThermo	从 c_p, h, s 得到的组分热力特性
传输特性——transport	
constTransport	常传输特性
polynomialTransport	基于温度的传输特性多项式
SutherlandTransport	基于温度的的传输 Sutherland 方程
混合特性——mixture	
pureMixture	用于计算气体混合的普通混合热物理模型
homogeneousMixture	基于燃料标准化质量分数 b 的混合燃烧模型
inhomogeneousMixture	基于 b 和燃料总质量分数 f_t 的混合燃烧模型
veryInhomogeneousMixture	基于 b, f_t 和未燃烧燃料质量分数 f_u 的混合燃烧模型
basicMultiComponentMixture	多组分基本混合模型
multiComponentMixture	多组分衍生混合模型
reactingMixture	基于化学反应动力学模型的混合燃烧模型
egrMixture	废气再循环混合
singleStepReactionMixture	单步反应混合模型
热物理模型——thermoModel	
hePsiThermo	基于可压缩性 φ 的普通热物理模型计算
heRhoThermo	基于密度 ρ 的普通热物理模型计算
psiReactionThermo	基于 φ 的混合燃烧焓值计算
psiuReactionThermo	基于 φ_u 的混合燃烧焓值计算
rhoReactionThermo	基于 ρ 的混合燃烧焓值计算
heheupsireactionThermo	计算未燃气体和混燃的焓值

表 7.1 热物理模型结构

下面是一个 thermoType 的范例：

```

thermo
{
    type          hePsiThermo;
    mixture       pureMixture;
    transport     const;
    thermo        hConst;
    equationOfState perfectGas;
    specie        specie;
    energy         sensibleEnthalpy;
}

```

关键词指定了热物理模型，例如 `transport` 指定常传递模型量（粘度不变，热导率不变），`equationOfState` 指定理想气体。另外，通过 `energy`，用户可以指定用于求解的能量形式以及热力特性。能量可以是内能，也可以为焓，可以包含热源 Δh_f 或者不包含。我们使用 A 能量（absolute energy）表示包含热源的概念，S 能量（sensible energy）表示没有包含热源的概念。例如对于和 S 焓有关的 A 焓，我们有公式：

$$h = h_s + \sum_i c_i \Delta h_f^i \quad (7.1)$$

其中， c_i 和 h_f^i 是组分 i 的分子质量和热源。大多数情况下，我们使用更容易解释化学反应带来的能量变化的 S 能。`energy` 关键词可以指定为： `sensibleEnthalpy`, `sensibleInternalEnergy` 和 `absoluteEnthalpy`。

7.1.1 热物理数据

每个组分都需要输入基本的热物理参数。他们必须以组分名作为关键词，例如 O2, H2O, mixture，其后是子字典，包含的参数主要有：

- `specie`: 包括组分的摩尔数，分子质量，摩尔分子质量（单位为 g/mol）；
- `thermoDynamics`: 所选热力模型的相关参数；
- `transport`: 所选传递模型的相关参数；

热力模型参数和热导率 C_p 有关，别的相关特性可以从热导率计算而来，当前的 `thermo` 模型如下：

- `hConstThermo`: 指定 C_p 以及 H_f 为常量。它们通过两个关键词 `Cp` 和 `Hf` 来指定；
- `eConstThermo`: 指定 C_v 以及 H_f 为常量。它们通过两个关键词 `Cv` 和 `Hf` 来指定；
- `janafThermo`: 从 JANAF 热力表选择参数，以温度的函数来计算 C_p 。参数顺序参见表 7.2。此函数在温度限 T_1 和 T_2 之间有效。他有两套参数需要指定，第一套的参数设定应该在温度 T_c 以上，在 T_h 以下。第二套的参数设定应该在温度 T_l 以上，在 T_e

以下：

$$c_p = R((a_4T + a_3)T + a_2)T + a_1)T + a_0 \quad (7.2)$$

另外，一些积分常数，例如 a_5, a_6 （对应最高温度和最低温度值）用来计算 h 和 s ；

- **hPloynomialThermo:** 从一个可以指定任意阶数的函数来计算 C_p 。参见下面的这个例子：`$FOAM_TUTORIALS/-lagrangian/porousExplicitSourceReactingParcelFoam/filter;`

描述	信息	关键词
温度下限	$T_l(K)$	Tlow
温度上限	$T_h(K)$	Thigh
温度	$T_c(K)$	Tcommon
高温系数	a_0, \dots, a_4	highCpCoeffs(a0 a1 a3 a4 a4 a5...)
高温焓补偿	a_5	a5...
高温熵补偿	a_6	a6...
低温系数	a_0, \dots, a_4	lowCpCoeffs(a0 a1 a3 a4 a4 a5...)
低温焓补偿	a_5	a5...
低温熵补偿	a_6	a6...

表 7.2 JANAF 热力模型系数

传输系数用来计算动力粘度 μ 、热导率 ϵ 、以及层流热导率 α （用于焓方程）。参见下面的描述：

- **constTransport:** 粘度 μ 为常数，并有普朗特数公式： $Pr = c_p\mu/\epsilon$ 来计算，其中需要指定 mu 和 Pr ；
- **sutherlandTransport:** 通过温度 T ，sutherland 系数 A_s ，sutherland 系数 T_s 的函数来计算粘度 μ 。需要指定 As 和 Ts ；计算公式为：

$$\mu = \frac{A_s \sqrt{T}}{1 + T_s/T} \quad (7.3)$$

- **polynomialTransport:** 从一个可以指定任意阶数的函数通过温度 T 来计算粘度 μ 和热导率 ϵ ；

下面是一个组分名称为 fuel，使用 `sutherlandTransport` 模型和 `janafThermo` 的范例：

```

fuel
{
    specie
    {
        nMoles      1;
        molWeight   16.0428;
    }
    thermodynamics
    {
        Tlow        200;
        Thigh       6000;
        Tcommon     1000;
        highCpCoeffs (1.63543 0.0100844 -3.36924e-06 5.34973e-10
                      -3.15528e-14 -10005.6 9.9937);
        lowCpCoeffs (5.14988 -0.013671 4.91801e-05 -4.84744e-08
                      1.66694e-11 -10246.6 -4.64132);
    }
    transport
    {
        As          1.67212e-06;
        Ts          170.672;
    }
}

```

下面是一个使用 `constTransport` 和 `hConstThermo` 模型指定 `air` 类的一个例子：

```

air
{
    specie
    {
        nMoles      1;
        molWeight   28.96;
    }
    thermodynamics
    {
        Cp          1004.5;
        Hf          2.544e+06;
    }
    transport
    {
        mu          1.8e-05;
        Pr          0.7;
    }
}

```

7.2 湍流模型

任何包含湍流的求解器都会读取 `TurbulenceProperties` 字典文件。在这个文件中包含一个关键词为 `simulationType`，它控制使用的湍流模型：

- `laminar`: 不使用湍流模型;
- `RASModel`: 使用雷诺平均模型 (RAS);
- `LESModel`: 使用大涡模拟 (LES);

如果选择了 `RASModel`，可以在 `constant` 目录下的 `RASProperties` 文件中指定 RAS 模型。具体的可选的 `RASModel` 参见表 3.9。类似的，如果选择了 `LESModel`，那么具体的 LES 模型应该在 `LESProperties` 中通过 `LESModel` 来指定。

表 7.3 中列举的是 `RASProperties` 所需的关键词信息，`LESProperties` 的信息见表 7.4。

RASModel	RAS 湍流模型的名称
turbulence	湍流开关控制
printCoeffs	在运行最初输出湍流模型系数
<RASModel>Coeffs	RASModel 模型系数

表 7.3 RASProperties 内的关键词

LESModel	LES 湍流模型的名称
delta	delta 模型名称
<LESModel>Coeffs	LES 模型系数
<delta>Coeffs	delta 模型系数

表 7.4 LESProperties 内的关键词

可压及不可压 RAS 湍流模型, LES 模型和 delta 模型都在表 3.9 中有阐述。相关例子在 \$FOAM_TUTORIALS 中。

7.2.1 模型系数

RAS 湍流模型的各个系数都存在一个默认值, 它们在源代码中被指定。如果用户想更改这些默认值, 它们可以在 RASProperties 文件中添加一个子字典, 关键词就是相关的模型附加上 Coeffs。例如 kEpsilon 模型通过 kEpsilonCoeffs 作为关键词。如果 RASProperties 中的 printCoeffs 开关设置为 on, 那么在求解器最开始运行的时候, 相关的...Coeffs 字典就会被输出。用户可以简单地把输出的信息拷贝到 RASProperties 中并加以编辑。

7.2.2 壁面函数

OpenFOAM 提供了可用于边界条件的一系列壁面函数模型可供选择。可以在不同的壁面上使用不同的壁面函数。壁面函数的指定如下: 不可压缩 RAS 的壁面函数通过 0/nut (ν_t) 文件指定, 可压缩 RAS 的壁面函数通过 0/mut (μ_t) 文件指定, 不可压缩 LES 的壁面函数通过 0/nuSgs (ν_{sgs}) 文件指定, 可压缩 LES 的壁面函数通过 0/muSgs (μ_{sgs}) 文件指定, 例如, 一个 0/nut 文件夹:

```

17
18 dimensions          [0 2 -1 0 0 0];
19
20 internalField      uniform 0;
21
22 boundaryField
23 {
24   movingWall
25   {
26     type      nutkWallFunction;
27     value    uniform 0;
28   }
29   fixedWalls
30   {
31     type      nutkWallFunction;
32     value    uniform 0;
33   }

```

```

34     frontAndBack
35     {
36         type      empty;
37     }
38 }
39
40
41 // ****

```

OpenFOAM 提供了一系列壁面函数，例如: nutWallFunction, nutRoughWallFunction, nutSpalart -AllmarasStandardRoughWallFunction, nutSpalartAllmarasStandardWallFunction 以及 nutSpalart -AllmarasWallFunction。用户可以通过下面的命令来查看壁面函数模型的完整列表:

```
find $FOAM_SRC/turbulenceModels -name wallFunctions
```

在每个壁面函数条件下，用户可以通过指定 E、kappa、以及 Cmu 关键词来重置默认的 E , κ , C_μ 这些系数。

在 nut 或者 mut 文件中，针对不同的 patch 选择了不同的壁面函数，用户应该在 epsilon 场中的对应的 patch 上选择 epsilonWallFunction，以及 k, q, R 场中对应的 patch 上选择 kqRwallFunction。

索引

操作符 数 A B C D E F G H I J K L M N O P Q R S T U V W X Z

操作符

*

tensor member function, [P-21](#)

+

tensor member function, [P-21](#)

-

tensor member function, [P-21](#)

/

tensor member function, [P-21](#)

/*...*/

C++ syntax, [U-76](#)

//

C++ syntax, [U-76](#)

OpenFOAM file syntax, [U-104](#)

include

C++ syntax, [U-70, U-76](#)

&

tensor member function, [P-21](#)

&&

tensor member function, [P-21](#)

^

tensor member function, [P-21](#)

<LESModel>Coeffs keyword, [U-184](#)

<RASModel>Coeffs keyword, [U-184](#)

<delta>Coeffs keyword, [U-184](#)

0.000000e+00 directory, [U-104](#)

1-dimensional mesh, [U-130](#)

1D mesh, [U-130](#)

2-dimensional mesh, [U-130](#)

2D mesh, [U-130](#)

数

0 directory, [U-104](#)

A

access functions, [P-19](#)

addLayersControls keyword, [U-146](#)

adiabaticFlameT utility, [U-95](#)

adiabaticPerfectFluid model, [U-99, U-179](#)

adjointShapeOptimizationFoam solver, [U-83](#)

adjustableRunTime

keyword entry, [U-60, U-111](#)

adjustTimeStep keyword, [U-60, U-112](#)

agglomerator keyword, [U-123](#)

algorithms tools, [U-96](#)

alphaContactAngle

boundary condition, [U-57](#)

analytical solution, [P-41](#)

Animations window panel, [U-168](#)

anisotropicFilter model, [U-100](#)

Annotation window panel, [U-24, U-167](#)

ansysToFoam utility, [U-89](#)

APIfunctions model, [U-99](#)

applications, [U-67](#)

Apply button, [U-164, U-168](#)

applyBoundaryLayer utility, [U-88](#)

applyWallFunctionBoundaryConditions utility, [U-88](#)

arbitrarily unstructured, [P-27](#)

arc

keyword entry, [U-139](#)

arc keyword, [U-138](#)

As keyword, [U-182](#)

ascii

keyword entry, [U-112](#)

attachMesh utility, [U-90](#)

Auto Accept button, [U-168](#)

autoMesh

library, [U-96](#)
autoPatch utility, [U-90](#)
autoRefineMesh utility, [U-91](#)
axes
 right-handed, [U-136](#)
 right-handed rectangular Cartesian, [P-11](#), [U-18](#)
axi-symmetric cases, [U-135](#), [U-144](#)
axi-symmetric mesh, [U-130](#)

B

background
 process, [U-24](#), [U-79](#)
backward
 keyword entry, [U-119](#)
Backward differencing, [P-35](#)
barotropicCompressibilityModels
 library, [U-98](#)
basicMultiComponentMixture model, [U-98](#), [U-180](#)
basicSolidThermo
 library, [U-99](#)
basicThermophysicalModels
 library, [U-98](#)
binary
 keyword entry, [U-112](#)
BirdCarreau model, [U-101](#)
blended differencing, [P-34](#)
block
 expansion ratio, [U-140](#)
block keyword, [U-138](#)
blocking
 keyword entry, [U-78](#)
blockMesh
 library, [U-96](#)
blockMesh solver, [P-43](#)
blockMesh utility, [U-37](#), [U-89](#), [U-136](#)
blockMesh executable
 vertex numbering, [U-140](#)
blockMeshDict
 dictionary, [U-18](#), [U-20](#), [U-35](#), [U-48](#), [U-136](#), [U-144](#)
blocks keyword, [U-20](#), [U-30](#), [U-140](#)
boundaries, [U-132](#)
boundary, [U-132](#)
boundary
 dictionary, [U-129](#), [U-136](#)
boundary keyword, [U-141](#)
boundary condition
 alphaContactAngle, [U-57](#)
 buoyantPressure, [U-137](#)
 calculated, [U-136](#)
 cyclic, [U-135](#), [U-142](#)
 directionMixed, [U-136](#)
 empty, [P-59](#), [P-65](#), [U-18](#), [U-130](#), [U-135](#)
 fixedGradient, [U-136](#)
 fixedValue, [U-136](#)
 fluxCorrectedVelocity, [U-137](#)
 inlet, [P-65](#)
 inletOutlet, [U-137](#)
 mixed, [U-136](#)
 movingWallVelocity, [U-137](#)
 outlet, [P-65](#)
 outletInlet, [U-137](#)
 partialSlip, [U-137](#)
 patch, [U-134](#)
 pressureDirectedInletVelocity, [U-137](#)
 pressureInletVelocity, [U-137](#)
 pressureOutlet, [P-59](#)
 pressureTransmissive, [U-137](#)
 processor, [U-135](#)
 setup, [U-20](#)
 slip, [U-137](#)
 supersonicFreeStream, [U-137](#)
 surfaceNormalFixedValue, [U-137](#)
 symmetryPlane, [P-59](#), [U-134](#)
 totalPressure, [U-137](#)
 turbulentInlet, [U-137](#)
 wall, [U-40](#)
 wall, [P-59](#), [P-65](#), [U-57](#), [U-134](#)
 wedge, [U-130](#), [U-135](#), [U-144](#)
 zeroGradient, [U-136](#)
boundary conditions, [P-39](#)
 Dirichlet, [P-39](#)
 inlet, [P-40](#)
 Neumann, [P-39](#)

no-slip impermeable wall, [P-40](#)
outlet, [P-40](#)
physical, [P-40](#)
symmetry plane, [P-40](#)

boundaryField keyword, [U-21](#), [U-108](#)
boundaryFoam solver, [U-84](#)
bounded
 keyword entry, [U-117](#), [U-118](#)

boxToCell keyword, [U-58](#)
boxTurb utility, [U-88](#)
breaking of a dam, [U-55](#)
buoyantBoussinesqPimpleFoam solver, [U-86](#)
buoyantBoussinesqSimpleFoam solver, [U-86](#)
buoyantPimpleFoam solver, [U-86](#)
buoyantPressure
 boundary condition, [U-137](#)
buoyantSimpleFoam solver, [U-86](#)
button
 Apply, [U-164](#), [U-168](#)
 Auto Accept, [U-168](#)
 Choose Preset, [U-166](#)
 Delete, [U-164](#)
 Edit Color Map, [U-165](#)
 Enable Line Series, [U-34](#)
 Orientation Axes, [U-24](#), [U-167](#)
 Refresh Times, [U-25](#)
 Rescale to Data Range, [U-25](#)
 Reset, [U-164](#)
 Set Ambient Color, [U-166](#)
 Update GUI, [U-165](#)
 Use Parallel Projection, [U-24](#)
 Use parallel projection, [U-167](#)

C

C++ syntax
 /*...*/, [U-76](#)
 //, [U-76](#)
 # include, [U-70](#), [U-76](#)

cacheAgglomeration keyword, [U-123](#)
calculated
 boundary condition, [U-136](#)

cAlpha keyword, [U-61](#)

cases, [U-103](#)
castellatedMesh keyword, [U-146](#)
castellatedMeshControls
 dictionary, [U-147](#)–[U-149](#)

castellatedMeshControls keyword, [U-146](#)
cavitatingDyMFoam solver, [U-85](#)
cavitatingFoam solver, [U-85](#)
cavity flow, [U-17](#)
ccm26ToFoam utility, [U-89](#)

CEI ARCH
 environment variable, [U-173](#)

CEI HOME
 environment variable, [U-173](#)

cell
 expansion ratio, [U-140](#)

cell class, [P-27](#)

cell
 keyword entry, [U-174](#)

cellLimited
 keyword entry, [U-117](#)

cellPoint
 keyword entry, [U-174](#)

cellPointFace
 keyword entry, [U-174](#)

cells
 dictionary, [U-136](#)

central differencing, [P-34](#)

cfdTools tools, [U-96](#)
cfx4ToFoam utility, [U-89](#), [U-154](#)
changeDictionary utility, [U-88](#)
Charts window panel, [U-168](#)
checkMesh utility, [U-90](#), [U-155](#)
chemFoam solver, [U-86](#)
chemistryModel
 library, [U-99](#)

chemistryModel model, [U-99](#)
chemistrySolver model, [U-99](#)
chemkinToFoam utility, [U-95](#)
Choose Preset button, [U-166](#)
chtMultiRegionSimpleFoam solver, [U-87](#)
chtMultiRegionFoam solver, [U-87](#)
Chung
 library, [U-99](#)

class

cell, [P-27](#)
dimensionSet, [P-21](#), [P-28](#), [P-29](#)
face, [P-27](#)
finiteVolumeCalculus, [P-29](#)
finiteVolumeMethod, [P-29](#)
fvMesh, [P-27](#)
fvSchemes, [P-32](#)
fvc, [P-32](#)
fvm, [P-32](#)
pointField, [P-27](#)
polyBoundaryMesh, [P-27](#)
polyMesh, [P-27](#), [U-127](#), [U-129](#)
polyPatchList, [P-27](#)
polyPatch, [P-27](#)
scalarField, [P-25](#)
scalar, [P-19](#)
slice, [P-27](#)
symmTensorField, [P-25](#)
symmTensorThirdField, [P-25](#)
tensorField, [P-25](#)
tensorThirdField, [P-25](#)
tensor, [P-19](#)
vectorField, [P-25](#)
vector, [P-19](#), [U-107](#)
word, [P-21](#), [P-27](#)
class keyword, [U-105](#)
clockTime
 keyword entry, [U-111](#)
cloud keyword, [U-175](#)
cloudFunctionObjects
 library, [U-96](#)
cmptAv
 tensor member function, [P-21](#)
Co utility, [U-92](#)
coalChemistryFoam solver, [U-87](#)
coalCombustion
 library, [U-97](#)
cofactors
 tensor member function, [P-21](#)
coldEngineFoam solver, [U-86](#)
collapseEdges utility, [U-91](#)
Color By menu, [U-166](#)
Color Legend window, [U-27](#)
Color Legend window panel, [U-166](#)
Color Scale window panel, [U-166](#)
Colors window panel, [U-168](#)
compressibleInterDyMFoam solver, [U-85](#)
compressibleInterFoam solver, [U-85](#)
compressibleMultiphaseInterFoam solver, [U-85](#)
combinePatchFaces utility, [U-91](#)
comments, [U-76](#)
commsType keyword, [U-78](#)
compressed
 keyword entry, [U-112](#)
compressibleLESModels
 library, [U-101](#)
compressibleRASModels
 library, [U-100](#)
constant directory, [U-104](#), [U-179](#)
constant model, [U-98](#)
constTransport model, [U-99](#), [U-180](#)
containers tools, [U-96](#)
continuum
 mechanics, [P-11](#)
control
 of time, [U-111](#)
controlDict
 dictionary, [P-61](#), [U-21](#), [U-30](#), [U-41](#), [U-50](#),
 [U-60](#), [U-104](#), [U-160](#)
controlDict file, [P-46](#)
convection, *see* divergence, [P-34](#)
convergence, [U-38](#)
conversion
 library, [U-97](#)
convertToMeters keyword, [U-138](#)
coordinate
 system, [P-11](#)
coordinate system, [U-18](#)
corrected
 keyword entry, [U-117](#), [U-118](#)
Courant number, [P-38](#), [U-22](#)
Cp keyword, [U-181](#)
cpuTime
 keyword entry, [U-111](#)
Crank Nicholson
 temporal discretisation, [P-38](#)

CrankNicholson
 keyword entry, [U-119](#)
createExternalCoupledPatchGeometry
utility,
 [U-88](#)
createBaffles utility, [U-90](#)
createPatch utility, [U-90](#)
createTurbulenceFields utility, [U-92](#)
cross product, **see** tensor, vector cross
product
CrossPowerLaw
 keyword entry, [U-58](#)
CrossPowerLaw model, [U-101](#)
cubeRootVolDelta model, [U-101](#)
cubicCorrected
 keyword entry, [U-119](#)
cubicCorrection
 keyword entry, [U-116](#)
curl, [P-33](#)
curl
 fvc member function, [P-33](#)
Current Time Controls menu, [U-25](#), [U-165](#)
curve keyword, [U-175](#)
Cv keyword, [U-181](#)
cyclic
 boundary condition, [U-135](#), [U-142](#)
cyclic
 keyword entry, [U-135](#)
cylinder
 flow around a, [P-41](#)

D

d2dt2
 fvc member function, [P-33](#)
 fvm member function, [P-33](#)
dam
 breaking of a, [U-55](#)
datToFoam utility, [U-89](#)
db tools, [U-96](#)
ddt
 fvc member function, [P-33](#)
 fvm member function, [P-33](#)
DeardorffDiffStress model, [U-101](#)
debug keyword, [U-146](#)

decompose model, [U-97](#)
decomposePar utility, [U-79](#), [U-80](#), [U-95](#)
decomposeParDict
 dictionary, [U-79](#)
decomposition
 of field, [U-79](#)
 of mesh, [U-79](#)
decompositionMethods
 library, [U-97](#)
decompression of a tank, [P-58](#)
defaultFieldValues keyword, [U-58](#)
deformedGeom utility, [U-90](#)
Delete button, [U-164](#)
delta keyword, [U-81](#), [U-184](#)
deltaT keyword, [U-111](#)
dependencies, [U-70](#)
dependency lists, [U-70](#)
det
 tensor member function, [P-21](#)
determinant, **see** tensor, determinant
dev
 tensor member function, [P-21](#)
diag
 tensor member function, [P-21](#)
diagonal
 keyword entry, [U-121](#), [U-122](#)
DIC
 keyword entry, [U-122](#)
DICGaussSeidel
 keyword entry, [U-122](#)
dictionary
 LESProperties, [U-184](#)
 PISO, [U-23](#)
 blockMeshDict, [U-18](#), [U-20](#), [U-35](#), [U-48](#),
 [U-136](#), [U-144](#)
 boundary, [U-129](#), [U-136](#)
 castellatedMeshControls, [U-147](#)–[U-149](#)
 cells, [U-136](#)
 controlDict, [P-61](#), [U-21](#), [U-30](#), [U-41](#),
 [U-50](#),
 [U-60](#), [U-104](#), [U-160](#)
 decomposeParDict, [U-79](#)
 faces, [U-129](#), [U-136](#)

fvSchemes, [U-61](#), [U-104](#), [U-113](#)
fvSolution, [U-104](#), [U-120](#)
mechanicalProperties, [U-49](#)
neighbour, [U-129](#)
owner, [U-129](#)
points, [U-129](#), [U-136](#)
thermalProperties, [U-50](#)
thermophysicalProperties, [U-179](#)
transportProperties, [U-21](#), [U-38](#), [U-41](#)
turbulenceProperties, [U-40](#), [U-59](#), [U-183](#)
differencing
 Backward, [P-35](#)
 blended, [P-34](#)
 central, [P-34](#)
 Euler implicit, [P-35](#)
 Gamma, [P-34](#)
 MINMOD, [P-34](#)
 SUPERBEE, [P-34](#)
 upwind, [P-34](#)
 van Leer, [P-34](#)
DILU
 keyword entry, [U-122](#)
dimension
 checking in OpenFOAM, [P-21](#), [U-107](#)
dimensional units, [U-107](#)
dimensioned<Type> template class, [P-21](#)
dimensionedTypes tools, [U-96](#)
dimensions keyword, [U-20](#), [U-108](#)
dimensionSet class, [P-21](#), [P-28](#), [P-29](#)
dimensionSet tools, [U-96](#)
directionMixed
 boundary condition, [U-136](#)
directory
 0.000000e+00, [U-104](#)
 0, [U-104](#)
 Make, [U-71](#)
 constant, [U-104](#), [U-179](#)
 fluentInterface, [U-170](#)
 polyMesh, [U-104](#), [U-129](#)
 processorN, [U-80](#)
 run, [U-103](#)
 system, [P-46](#), [U-104](#)
 tutorials, [P-41](#), [U-17](#)
discretisation
 equation, [P-29](#)
Display window panel, [U-23](#), [U-25](#), [U-164](#), [U-165](#)
distance
 keyword entry, [U-149](#), [U-175](#)
distributed model, [U-97](#)
distributed keyword, [U-81](#), [U-82](#)
distributionModels
 library, [U-97](#)
div
 fvc member function, [P-33](#)
 fvm member function, [P-33](#)
divergence, [P-33](#), [P-35](#)
divSchemes keyword, [U-114](#)
dnsFoam solver, [U-86](#)
doLayers keyword, [U-146](#)
double inner product, [see](#) tensor,double inner
 product
DPMFoam solver, [U-87](#)
dsmc
 library, [U-97](#)
dsmcFieldsCalc utility, [U-93](#)
dsmcFoam solver, [U-87](#)
dsmcInitialise utility, [U-88](#)
dx
 keyword entry, [U-174](#)
dynamicFvMesh
 library, [U-97](#)
dynamicMesh
 library, [U-96](#)
dynLagrangian model, [U-101](#)
dynOneEqEddy model, [U-101](#)

E

eConstThermo model, [U-99](#), [U-179](#)
edgeGrading keyword, [U-140](#)
edgeMesh
 library, [U-97](#)
edges keyword, [U-138](#)
Edit menu, [U-167](#), [U-168](#)
Edit Color Map button, [U-165](#)
egrMixture model, [U-98](#), [U-180](#)

electrostaticFoam solver, [U-88](#)
empty
 boundary condition, [P-59](#), [P-65](#), [U-18](#),
 [U-130](#), [U-135](#)
empty
 keyword entry, [U-135](#)
Enable Line Series button, [U-34](#)
endTime keyword, [U-22](#), [U-111](#)
energy keyword, [U-181](#)
engine
 library, [U-97](#)
engineCompRatio utility, [U-93](#)
engineFoam solver, [U-86](#)
engineSwirl utility, [U-88](#)
ensight74FoamExec utility, [U-172](#)
ENSIGHT7 INPUT
 environment variable, [U-173](#)
ENSIGHT7 READER
 environment variable, [U-173](#)
ensightFoamReader utility, [U-91](#)
enstrophy utility, [U-92](#)
environment variable
 CEI ARCH, [U-173](#)
 CEI HOME, [U-173](#)
 ENSIGHT7 INPUT, [U-173](#)
 ENSIGHT7 READER, [U-173](#)
 FOAM RUN, [U-103](#)
 WM ARCH OPTION, [U-74](#)
 WM ARCH, [U-74](#)
 WM COMPILER BIN, [U-74](#)
 WM COMPILER DIR, [U-74](#)
 WM COMPILER LIB, [U-74](#)
 WM COMPILER, [U-74](#)
 WM COMPILE OPTION, [U-74](#)
 WM DIR, [U-74](#)
 WM MPLIB, [U-74](#)
 WM OPTIONS, [U-74](#)
 WM PRECISION OPTION, [U-74](#)
 WM PROJECT DIR, [U-74](#)
 WM PROJECT INST DIR, [U-74](#)
 WM PROJECT USER DIR, [U-74](#)
 WM PROJECT VERSION, [U-74](#)
 WM PROJECT, [U-74](#)
wmake, [U-73](#)

equationOfState keyword, [U-181](#)
equilibriumCO utility, [U-95](#)
equilibriumFlameT utility, [U-95](#)
errorReduction keyword, [U-153](#)
Euler
 keyword entry, [U-119](#)
Euler implicit
 differencing, [P-35](#)
 temporal discretisation, [P-38](#)
examples
 decompression of a tank, [P-58](#)
 flow around a cylinder, [P-41](#)
 flow over backward step, [P-49](#)
 Hartmann problem, [P-63](#)
 supersonic flow over forward step, [P-54](#)
execFlowFunctionObjects utility, [U-93](#)
expandDictionary utility, [U-95](#)
expansionRatio keyword, [U-152](#)
explicit
 temporal discretisation, [P-38](#)
extrude2DMesh utility, [U-89](#)
extrudeMesh utility, [U-89](#)
extrudeToRegionMesh utility, [U-89](#)

F

face class, [P-27](#)
face keyword, [U-175](#)
faceAgglomerate utility, [U-88](#)
faceAreaPair
 keyword entry, [U-123](#)
faceLimited
 keyword entry, [U-117](#)
faces
 dictionary, [U-129](#), [U-136](#)

FDIC
 keyword entry, [U-122](#)
featureAngle keyword, [U-152](#)
features keyword, [U-147](#), [U-148](#)
field
 U, [U-22](#)
 p, [U-22](#)
 decomposition, [U-79](#)
FieldField<Type> template class, [P-28](#)

fieldFunctionObjects
 library, [U-96](#)
fields, [P-25](#)
 mapping, [U-160](#)
fields tools, [U-96](#)
fields keyword, [U-174](#)
Field<Type> template class, [P-25](#)
fieldValues keyword, [U-58](#)
file
 Make/files, [U-72](#)
 controlDict, [P-46](#)
 files, [U-71](#)
 g, [U-59](#)
 options, [U-71](#)
 snappyHexMeshDict, [U-145](#)
 transportProperties, [U-58](#)
file format, [U-104](#)
fileFormats
 library, [U-97](#)
fileModificationChecking keyword, [U-78](#)
fileModificationSkew keyword, [U-78](#)
files file, [U-71](#)
filteredLinear2
 keyword entry, [U-116](#)
finalLayerThickness keyword, [U-152](#)
financialFoam solver, [U-88](#)
finite volume
 discretisation, [P-23](#)
 mesh, [P-27](#)
finiteVolume
 library, [U-96](#)
finiteVolume tools, [U-96](#)
finiteVolumeCalculus class, [P-29](#)
finiteVolumeMethod class, [P-29](#)
fireFoam solver, [U-86](#)
firstTime keyword, [U-111](#)
fixed
 keyword entry, [U-112](#)
fixedGradient
 boundary condition, [U-136](#)
fixedValue
 boundary condition, [U-136](#)
flattenMesh utility, [U-90](#)
floatTransfer keyword, [U-78](#)
flow
 free surface, [U-55](#)
 laminar, [U-17](#)
 steady, turbulent, [P-49](#)
 supersonic, [P-55](#)
 turbulent, [U-17](#)
flow around a cylinder, [P-41](#)
flow over backward step, [P-49](#)
flowType utility, [U-92](#)
fluent3DMeshToFoam utility, [U-89](#)
fluentInterface directory, [U-170](#)
fluentMeshToFoam utility, [U-89, U-154](#)
fluxCorrectedVelocity
 boundary condition, [U-137](#)
fluxRequired keyword, [U-114](#)
OpenFOAM
 cases, [U-103](#)
FOAM RUN
 environment variable, [U-103](#)
foamCalc utility, [U-33, U-93](#)
foamCalcFunctions
 library, [U-96](#)
foamCorrectVrt script/alias, [U-158](#)
foamDataToFluent utility, [U-91, U-170](#)
foamDebugSwitches utility, [U-95](#)
FoamFile keyword, [U-105](#)
foamFile
 keyword entry, [U-174](#)
foamFormatConvert utility, [U-95](#)
foamHelp utility, [U-95](#)
foamInfoExec utility, [U-95](#)
foamJob script/alias, [U-177](#)
foamListTimes utility, [U-93](#)
foamLog script/alias, [U-177](#)
foamMeshToFluent utility, [U-89, U-170](#)
foamToEnsight utility, [U-91](#)
foamToEnsightParts utility, [U-91](#)
foamToGMV utility, [U-91](#)
foamToStarMesh utility, [U-89](#)
foamToSurface utility, [U-89](#)
foamToTecplot360 utility, [U-91](#)
foamToVTK utility, [U-91](#)
foamUpgradeCyclics utility, [U-88](#)
foamUpgradeFvSolution utility, [U-88](#)

foamyHexMeshBackgroundMesh utility, [U-89](#)
foamyHexMeshSurfaceSimplify utility, [U-89](#)
foamyHexMesh utility, [U-89](#)
foamyQuadMesh utility, [U-89](#)
forces
 library, [U-96](#)
foreground
 process, [U-24](#)
format keyword, [U-105](#)
fourth
 keyword entry, [U-117](#), [U-118](#)
functions keyword, [U-113](#)
fvc class, [P-32](#)
fvc member function
 curl, [P-33](#)
 d2dt2, [P-33](#)
 ddt, [P-33](#)
 div, [P-33](#)
 gGrad, [P-33](#)
 grad, [P-33](#)
 laplacian, [P-33](#)
 lsGrad, [P-33](#)
 snGrad, [P-33](#)
 snGradCorrection, [P-33](#)
 sqrGradGrad, [P-33](#)
fvDOM
 library, [U-98](#)
FVFunctionObjects
 library, [U-96](#)
fvm class, [P-32](#)
fvm member function
 d2dt2, [P-33](#)
 ddt, [P-33](#)
 div, [P-33](#)
 laplacian, [P-33](#)
 Su, [P-33](#)
 SuSp, [P-33](#)
fvMatrices tools, [U-96](#)
fvMatrix template class, [P-29](#)
fvMesh class, [P-27](#)
fvMesh tools, [U-96](#)
fvMotionSolvers
 library, [U-97](#)
fvSchemes
 dictionary, [U-61](#), [U-104](#), [U-113](#)
fvSchemes class, [P-32](#)
fvSchemes
 menu entry, [U-51](#)
fvSolution
 dictionary, [U-104](#), [U-120](#)

G

g file, [U-59](#)
gambitToFoam utility, [U-89](#), [U-154](#)
GAMG
 keyword entry, [U-52](#), [U-121](#), [U-122](#)
Gamma
 keyword entry, [U-116](#)
Gamma differencing, [P-34](#)
Gauss
 keyword entry, [U-117](#)
Gauss's theorem, [P-32](#)
GaussSeidel
 keyword entry, [U-122](#)
General window panel, [U-167](#), [U-168](#)
general
 keyword entry, [U-112](#)
genericFvPatchField
 library, [U-97](#)
geometric-algebraic multi-grid, [U-122](#)
GeometricBoundaryField template class, [P-28](#)
geometricField<Type> template class, [P-28](#)
geometry keyword, [U-146](#)
gGrad
 fvc member function, [P-33](#)
global tools, [U-96](#)
gmshToFoam utility, [U-89](#)
gnuplot
 keyword entry, [U-112](#), [U-174](#)
grad
 fvc member function, [P-33](#)
(Grad Grad) squared, [P-33](#)
gradient, [P-33](#), [P-36](#)
 Gauss scheme, [P-36](#)
 Gauss's theorem, [U-51](#)

least square fit, [U-51](#)
least squares method, [P-36](#), [U-51](#)
surface normal, [P-36](#)

gradSchemes keyword, [U-114](#)

graph tools, [U-96](#)

graphFormat keyword, [U-112](#)

GuldersEGRLaminarFlameSpeed model, [U-98](#)

GuldersLaminarFlameSpeed model, [U-98](#)

H

hConstThermo model, [U-99](#), [U-179](#)

heheupsiReactionThermo model, [U-98](#), [U-180](#)

Help menu, [U-167](#)

hePsiThermo model, [U-98](#), [U-180](#)

heRhoThermo model, [U-98](#), [U-180](#)

HerschelBulkley model, [U-101](#)

hExponentialThermo
 library, [U-99](#)

Hf keyword, [U-181](#)

hierarchical
 keyword entry, [U-80](#), [U-81](#)

highCpCoeffs keyword, [U-182](#)

homogenousDynOneEqEddy model, [U-101](#)

homogenousDynSmagorinsky model, [U-101](#)

homogeneousMixture model, [U-98](#), [U-180](#)

hPolynomialThermo model, [U-99](#), [U-180](#)

I

I
 tensor member function, [P-21](#)

icoFoam solver, [U-17](#), [U-21](#), [U-22](#), [U-24](#), [U-84](#)

icoPolynomial model, [U-99](#), [U-179](#)

icoUncoupledKinematicParcelDyMFoam solver,
 [U-87](#)

icoUncoupledKinematicParcelFoam solver, [U-87](#)

ideasToFoam utility, [U-154](#)

ideasUnvToFoam utility, [U-89](#)

identities, **see** tensor, identities

identity, **see** tensor, identity

incompressibleLESModels
 library, [U-101](#)

incompressiblePerfectGas model, [U-99](#), [U-179](#)

incompressibleRASModels
 library, [U-100](#)

incompressibleTransportModels
 library, [P-50](#), [U-101](#)

incompressibleTurbulenceModels
 library, [P-50](#)

index
 notation, [P-12](#), [P-13](#)

Information window panel, [U-164](#)

inhomogeneousMixture model, [U-98](#), [U-180](#)

inlet
 boundary condition, [P-65](#)

inletOutlet
 boundary condition, [U-137](#)

inner product, **see** tensor, inner product

inotify
 keyword entry, [U-78](#)

inotifyMaster

keyword entry, [U-78](#)

inside
 keyword entry, [U-149](#)

insideCells utility, [U-90](#)

interPhaseChangeDyMFoam solver, [U-85](#)

interPhaseChangeFoam solver, [U-85](#)

interDyMFoam solver, [U-85](#)

interfaceProperties
 library, [U-102](#)

interfaceProperties model, [U-102](#)

interFoam solver, [U-85](#)

interMixingFoam solver, [U-85](#)

internalField keyword, [U-21](#), [U-108](#)

interpolation tools, [U-96](#)

interpolationScheme keyword, [U-174](#)

interpolations tools, [U-96](#)

interpolationSchemes keyword, [U-114](#)

inv
 tensor member function, [P-21](#)

iterations

maximum, [U-121](#)

J

janafThermo model, [U-99](#), [U-180](#)

jobControl

 library, [U-96](#)

jplot

 keyword entry, [U-112](#), [U-174](#)

K

kEpsilon model, [U-100](#)

keyword

 As, [U-182](#)

Cp, [U-181](#)

Cv, [U-181](#)

FoamFile, [U-105](#)

Hf, [U-181](#)

LESModel, [U-184](#)

Pr, [U-182](#)

RASModel, [U-184](#)

Tcommon, [U-182](#)

Thigh, [U-182](#)

Tlow, [U-182](#)

Ts, [U-182](#)

addLayersControls, [U-146](#)

adjustTimeStep, [U-60](#), [U-112](#)

agglomerator, [U-123](#)

arc, [U-138](#)

blocks, [U-20](#), [U-30](#), [U-140](#)

block, [U-138](#)

boundaryField, [U-21](#), [U-108](#)

boundary, [U-141](#)

boxToCell, [U-58](#)

cAlpha, [U-61](#)

cacheAgglomeration, [U-123](#)

castellatedMeshControls, [U-146](#)

castellatedMesh, [U-146](#)

class, [U-105](#)

cloud, [U-175](#)

commsType, [U-78](#)

convertToMeters, [U-138](#)

curve, [U-175](#)

debug, [U-146](#)

defaultFieldValues, [U-58](#)

deltaT, [U-111](#)

delta, [U-81](#), [U-184](#)

dimensions, [U-20](#), [U-108](#)

distributed, [U-81](#), [U-82](#)

divSchemes, [U-114](#)

doLayers, [U-146](#)

edgeGrading, [U-140](#)

edges, [U-138](#)

endTime, [U-22](#), [U-111](#)

energy, [U-181](#)

equationOfState, [U-181](#)

errorReduction, [U-153](#)

expansionRatio, [U-152](#)

 face, [U-175](#)

 featureAngle, [U-152](#)

 features, [U-147](#), [U-148](#)

 fieldValues, [U-58](#)

 fields, [U-174](#)

 fileModificationChecking, [U-78](#)

 fileModificationSkew, [U-78](#)

 finalLayerThickness, [U-152](#)

 firstTime, [U-111](#)

 floatTransfer, [U-78](#)

 fluxRequired, [U-114](#)

 format, [U-105](#)

 functions, [U-113](#)

 geometry, [U-146](#)

 gradSchemes, [U-114](#)

 graphFormat, [U-112](#)

 highCpCoeffs, [U-182](#)

 internalField, [U-21](#), [U-108](#)

 interpolationSchemes, [U-114](#)

 interpolationScheme, [U-174](#)

 laplacianSchemes, [U-114](#)

 latestTime, [U-38](#)

 layers, [U-152](#)

 leastSquares, [U-51](#)

 levels, [U-150](#)

 libs, [U-78](#), [U-112](#)

 locationInMesh, [U-148](#), [U-149](#)

 location, [U-105](#)

 lowCpCoeffs, [U-182](#)

 manualCoeffs, [U-81](#)

 maxAlphaCo, [U-60](#)

maxBoundarySkewness, [U-153](#)
maxConcave, [U-153](#)
maxCo, [U-60](#), [U-112](#)
maxDeltaT, [U-60](#)
maxFaceThicknessRatio, [U-152](#)
maxGlobalCells, [U-148](#)
maxInternalSkewness, [U-153](#)
maxIter, [U-121](#)
maxLocalCells, [U-148](#)
maxNonOrtho, [U-153](#)
maxThicknessToMedialRatio, [U-152](#)
mergeLevels, [U-123](#)
mergePatchPairs, [U-138](#)
mergeTolerance, [U-146](#)
meshQualityControls, [U-146](#)
method, [U-81](#)
midPointAndFace, [U-175](#)
midPoint, [U-175](#)
minArea, [U-153](#)
minDeterminant, [U-153](#)
minFaceWeight, [U-153](#)
minFlatness, [U-153](#)
minMedianAxisAngle, [U-152](#)
minRefinementCells, [U-148](#)
minThickness, [U-152](#)
minTriangleTwist, [U-153](#)
minTwist, [U-153](#)
minVolRatio, [U-153](#)
minVol, [U-153](#)
mode, [U-149](#)
molWeight, [U-181](#)
mu, [U-182](#)
nAlphaSubCycles, [U-61](#)
nBufferCellsNoExtrude, [U-152](#)
nCellsBetweenLevels, [U-148](#)
nFaces, [U-130](#)
nFinestSweeps, [U-123](#)
nGrow, [U-152](#)
nLayerIter, [U-152](#)
nMoles, [U-181](#)
nPostSweeps, [U-123](#)
nPreSweeps, [U-123](#)
nRelaxIter, [U-150](#), [U-152](#)
nRelaxedIter, [U-152](#)
nSmoothNormals, [U-152](#)
nSmoothPatch, [U-150](#)
nSmoothScale, [U-153](#)
nSmoothSurfaceNormals, [U-152](#)
nSmoothThickness, [U-152](#)
nSolveIter, [U-150](#)
neighbourPatch, [U-142](#)
numberOfSubdomains, [U-81](#)
n, [U-81](#)
object, [U-105](#)
order, [U-81](#)
pRefCell, [U-23](#), [U-124](#)
pRefValue, [U-23](#), [U-124](#)
p rhgRefCell, [U-124](#)
p rhgRefValue, [U-124](#)
patchMap, [U-160](#)
patches, [U-138](#)
preconditioner, [U-121](#), [U-122](#)
pressure, [U-49](#)
printCoeffs, [U-41](#), [U-184](#)
processorWeights, [U-80](#)
processorWeights, [U-81](#)
purgeWrite, [U-112](#)
refGradient, [U-136](#)
refinementRegions, [U-148](#), [U-150](#)
refinementSurfaces, [U-148](#)
refinementRegions, [U-149](#)
regions, [U-58](#)
relTol, [U-52](#), [U-121](#)
relativeSizes, [U-152](#)
relaxed, [U-153](#)
resolveFeatureAngle, [U-148](#)
roots, [U-81](#), [U-82](#)
runTimeModifiable, [U-112](#)
scotchCoeffs, [U-81](#)
setFormat, [U-174](#)
sets, [U-174](#)
simpleGrading, [U-140](#)
simulationType, [U-40](#), [U-59](#), [U-183](#)
smoother, [U-123](#)
snGradSchemes, [U-114](#)
snapControls, [U-146](#)
snap, [U-146](#)
solvers, [U-120](#)

solver, [U-52](#), [U-121](#)
specie, [U-181](#)
spline, [U-138](#)
startFace, [U-130](#)
startFrom, [U-22](#), [U-111](#)
startTime, [U-22](#), [U-111](#)
stopAt, [U-111](#)
strategy, [U-80](#), [U-81](#)
surfaceFormat, [U-174](#)
surfaces, [U-174](#)
thermoType, [U-179](#)
thermodynamics, [U-181](#)
timeFormat, [U-112](#)
timePrecision, [U-112](#)
timeScheme, [U-114](#)
tolerance, [U-52](#), [U-121](#), [U-150](#)
topoSetSource, [U-58](#)
traction, [U-49](#)
transport, [U-181](#)
turbulence, [U-184](#)
type, [U-132](#), [U-133](#)
uniform, [U-175](#)
valueFraction, [U-136](#)
value, [U-21](#), [U-136](#)
version, [U-105](#)
vertices, [U-20](#), [U-138](#), [U-139](#)
writeCompression, [U-112](#)
writeControl, [U-22](#), [U-60](#), [U-111](#)
writeFormat, [U-54](#), [U-112](#)
writeInterval, [U-22](#), [U-31](#), [U-111](#)
writePrecision, [U-112](#)
<LESModel>Coeffs, [U-184](#)
<RASModel>Coeffs, [U-184](#)
<delta>Coeffs, [U-184](#)
keyword entry
 CrankNicholson, [U-119](#)
 CrossPowerLaw, [U-58](#)
 DICGaussSeidel, [U-122](#)
 DIC, [U-122](#)
 DILU, [U-122](#)
 Euler, [U-119](#)
 FDIC, [U-122](#)
 GAMG, [U-52](#), [U-121](#), [U-122](#)
 Gamma, [U-116](#)
 GaussSeidel, [U-122](#)
 Gauss, [U-117](#)
 LESModel, [U-40](#), [U-183](#)
 MGridGen, [U-123](#)
 MUSCL, [U-116](#)
 Newtonian, [U-58](#)
 PBiCG, [U-121](#)
 PCG, [U-121](#)
 QUICK, [U-119](#)
 RASModel, [U-40](#), [U-183](#)
 SFCD, [U-116](#), [U-119](#)
 UMIST, [U-115](#)
 adjustableRunTime, [U-60](#), [U-111](#)
 arc, [U-139](#)
 ascii, [U-112](#)
 backward, [U-119](#)
 binary, [U-112](#)
 blocking, [U-78](#)
 bounded, [U-117](#), [U-118](#)
 cellLimited, [U-117](#)
 cellPointFace, [U-174](#)
 cellPoint, [U-174](#)
 cell, [U-174](#)
 clockTime, [U-111](#)
 compressed, [U-112](#)
 corrected, [U-117](#), [U-118](#)
 cpuTime, [U-111](#)
 cubicCorrected, [U-119](#)
 cubicCorrection, [U-116](#)
 cyclic, [U-135](#)
 diagonal, [U-121](#), [U-122](#)
 distance, [U-149](#), [U-175](#)
 dx, [U-174](#)
 empty, [U-135](#)
 faceAreaPair, [U-123](#)
 faceLimited, [U-117](#)
 filteredLinear2, [U-116](#)
 fixed, [U-112](#)
 foamFile, [U-174](#)
 fourth, [U-117](#), [U-118](#)
 general, [U-112](#)
 gnuplot, [U-112](#), [U-174](#)
 hierarchical, [U-80](#), [U-81](#)
 inotifyMaster, [U-78](#)

inotify, [U-78](#)
inside, [U-149](#)
jplot, [U-112](#), [U-174](#)
laminar, [U-40](#), [U-183](#)
latestTime, [U-111](#)
leastSquares, [U-117](#)
limitedCubic, [U-116](#)
limitedLinear, [U-116](#)
limited, [U-117](#), [U-118](#)
linearUpwind, [U-116](#), [U-119](#)
linear, [U-116](#), [U-119](#)
line, [U-139](#)
localEuler, [U-119](#)
manual, [U-80](#), [U-81](#)
metis, [U-81](#)
midPoint, [U-116](#)
nextWrite, [U-111](#)
noWriteNow, [U-111](#)
nonBlocking, [U-78](#)
none, [U-115](#), [U-122](#)
null, [U-174](#)
outside, [U-149](#)
patch, [U-135](#), [U-176](#)
polyLine, [U-139](#)
polySpline, [U-139](#)
processor, [U-135](#)
raw, [U-112](#), [U-174](#)
runTime, [U-31](#), [U-111](#)
scheduled, [U-78](#)
scientific, [U-112](#)
scotch, [U-80](#), [U-81](#)
simpleSpline, [U-139](#)
simple, [U-80](#), [U-81](#)
skewLinear, [U-116](#), [U-119](#)
smoothSolver, [U-121](#)
startTime, [U-22](#), [U-111](#)
steadyState, [U-119](#)
stl, [U-174](#)
symmetryPlane, [U-135](#)
timeStampMaster, [U-78](#)
timeStamp, [U-78](#)
timeStep, [U-22](#), [U-31](#), [U-111](#)
uncompressed, [U-112](#)
uncorrected, [U-117](#), [U-118](#)
upwind, [U-116](#), [U-119](#)
vanLeer, [U-116](#)
vtk, [U-174](#)
wall, [U-135](#)
wedge, [U-135](#)
writeControl, [U-111](#)
writeNow, [U-111](#)
xmgr, [U-112](#), [U-174](#)
xyz, [U-175](#)
x, [U-175](#)
y, [U-175](#)
z, [U-175](#)
kivaToFoam utility, [U-89](#)
kkLOmega model, [U-100](#)
kOmega model, [U-100](#)
kOmegaSST model, [U-100](#)
kOmegaSSTSAS model, [U-101](#)
Kronecker delta, [P-16](#)

L

lagrangian
 library, [U-97](#)
lagrangianIntermediate
 library, [U-97](#)
Lambda2 utility, [U-92](#)
LamBremhorstKE model, [U-100](#)
laminar model, [U-100](#), [U-101](#)
laminar
 keyword entry, [U-40](#), [U-183](#)
laminarFlameSpeedModels
 library, [U-98](#)
laplaceFilter model, [U-100](#)
Laplacian, [P-34](#)
laplacian, [P-33](#)
laplacian
 fvc member function, [P-33](#)
 fvm member function, [P-33](#)
laplacianFoam solver, [U-83](#)
laplacianSchemes keyword, [U-114](#)
latestTime
 keyword entry, [U-111](#)
latestTime keyword, [U-38](#)
LaunderGibsonRSTM model, [U-100](#)
LaunderSharmaKE model, [U-100](#)

layers keyword, [U-152](#)
leastSquares
 keyword entry, [U-117](#)
leastSquares keyword, [U-51](#)
LESdeltas
 library, [U-100](#)
LESfilters
 library, [U-100](#)
LESModel
 keyword entry, [U-40](#), [U-183](#)
LESModel keyword, [U-184](#)
LESProperties
 dictionary, [U-184](#)
levels keyword, [U-150](#)
libraries, [U-67](#)
library
 Chung, [U-99](#)
 FVFunctionObjects, [U-96](#)
 LESdeltas, [U-100](#)
 LESfilters, [U-100](#)
 MGridGenGAMGAgglomeration, [U-97](#)
 ODE, [U-97](#)
 OSspecific, [U-97](#)
 OpenFOAM, [U-96](#)
 P1, [U-98](#)
 PV3FoamReader, [U-163](#)
 PVFoamReader, [U-163](#)
 SLGThermo, [U-99](#)
 Wallis, [U-99](#)
 autoMesh, [U-96](#)
 barotropicCompressibilityModels, [U-98](#)
 basicSolidThermo, [U-99](#)
 basicThermophysicalModels, [U-98](#)
 blockMesh, [U-96](#)
 chemistryModel, [U-99](#)
 cloudFunctionObjects, [U-96](#)
 coalCombustion, [U-97](#)
 compressibleLESModels, [U-101](#)
 compressibleRASModels, [U-100](#)
 conversion, [U-97](#)
 decompositionMethods, [U-97](#)
 distributionModels, [U-97](#)
 dsmc, [U-97](#)
 dynamicFvMesh, [U-97](#)
 dynamicMesh, [U-96](#)
 edgeMesh, [U-97](#)
 engine, [U-97](#)
 fieldFunctionObjects, [U-96](#)
 fileFormats, [U-97](#)
 finiteVolume, [U-96](#)
 foamCalcFunctions, [U-96](#)
 forces, [U-96](#)
 fvDOM, [U-98](#)
 fvMotionSolvers, [U-97](#)
 genericFvPatchField, [U-97](#)
 hExponentialThermo, [U-99](#)
 incompressibleLESModels, [U-101](#)
 incompressibleRASModels, [U-100](#)
 incompressibleTransportModels, [P-50](#), [U-101](#)
 incompressibleTurbulenceModels, [P-50](#)
 interfaceProperties, [U-102](#)
 jobControl, [U-96](#)
 lagrangianIntermediate, [U-97](#)
 lagrangian, [U-97](#)
 laminarFlameSpeedModels, [U-98](#)
 linear, [U-99](#)
 liquidMixtureProperties, [U-99](#)
 liquidProperties, [U-99](#)
 meshTools, [U-97](#)
 molecularMeasurements, [U-97](#)
 molecule, [U-97](#)
 opaqueSolid, [U-98](#)
 pairPatchAgglomeration, [U-97](#)
 postCalc, [U-96](#)
 potential, [U-97](#)
 primitive, [P-19](#)
 radiationModels, [U-98](#)
 randomProcesses, [U-97](#)
 reactionThermophysicalModels, [U-98](#)
 sampling, [U-96](#)
 solidChemistryModel, [U-99](#)
 solidMixtureProperties, [U-100](#)
 solidParticle, [U-97](#)
 solidProperties, [U-99](#)
 solidSpecie, [U-100](#)

solidThermo, [U-100](#)
specie, [U-99](#)
spray, [U-97](#)
surfMesh, [U-97](#)
surfaceFilmModels, [U-102](#)
systemCall, [U-96](#)
thermophysicalFunctions, [U-99](#)
thermophysical, [U-179](#)
topoChangerFvMesh, [U-97](#)
triSurface, [U-97](#)
turbulence, [U-97](#)
twoPhaseProperties, [U-102](#)
utilityFunctionObjects, [U-96](#)
viewFactor, [U-98](#)
vtkFoam, [U-163](#)
vtkPV3Foam, [U-163](#)
libs keyword, [U-78](#), [U-112](#)
lid-driven cavity flow, [U-17](#)
LienCubicKE model, [U-100](#)
LienCubicKELowRe model, [U-100](#)
LienLeschzinerLowRe model, [U-100](#)
Lights window panel, [U-167](#)
limited
 keyword entry, [U-117](#), [U-118](#)
limitedCubic
 keyword entry, [U-116](#)
limitedLinear
 keyword entry, [U-116](#)
line
 keyword entry, [U-139](#)
Line Style menu, [U-34](#)
linear
 library, [U-99](#)
linear
 keyword entry, [U-116](#), [U-119](#)
linearUpwind
 keyword entry, [U-116](#), [U-119](#)
liquid
 electrically-conducting, [P-63](#)
liquidMixtureProperties
 library, [U-99](#)
liquidProperties
 library, [U-99](#)
lists, [P-25](#)
List<Type> template class, [P-25](#)
localEuler
 keyword entry, [U-119](#)
location keyword, [U-105](#)
locationInMesh keyword, [U-148](#), [U-149](#)
lowCpCoeffs keyword, [U-182](#)
lowReOneEqEddy model, [U-101](#)
LRDDiffStress model, [U-101](#)
LRR model, [U-100](#)
lsGrad
 fvc member function, [P-33](#)
LTSLinterFoam solver, [U-85](#)
LTSLreactingFoam solver, [U-86](#)
LTSLreactingParcelFoam solver, [U-87](#)

M

Mach utility, [U-92](#)
mag
 tensor member function, [P-21](#)
magneticFoam solver, [U-88](#)
magnetohydrodynamics, [P-63](#)
magSqr
 tensor member function, [P-21](#)
Make directory, [U-71](#)
make script/alias, [U-69](#)
Make/files file, [U-72](#)
manual
 keyword entry, [U-80](#), [U-81](#)
manualCoeffs keyword, [U-81](#)
mapFields utility, [U-30](#), [U-37](#), [U-41](#), [U-54](#),
[U-89](#),
 [U-160](#)
mapping
 fields, [U-160](#)
Marker Style menu, [U-34](#)
matrices tools, [U-96](#)
max
 tensor member function, [P-21](#)
maxAlphaCo keyword, [U-60](#)
maxBoundarySkewness keyword, [U-153](#)
maxCo keyword, [U-60](#), [U-112](#)
maxConcave keyword, [U-153](#)
maxDeltaT keyword, [U-60](#)
maxDeltaxyz model, [U-101](#)

maxFaceThicknessRatio keyword, [U-152](#)
maxGlobalCells keyword, [U-148](#)
maximum iterations, [U-121](#)
maxInternalSkewness keyword, [U-153](#)
maxIter keyword, [U-121](#)
maxLocalCells keyword, [U-148](#)
maxNonOrtho keyword, [U-153](#)
maxThicknessToMedialRatio keyword, [U-152](#)
mdEquilibrationFoam solver, [U-87](#)
mdFoam solver, [U-87](#)
mdInitialise utility, [U-89](#)
mechanicalProperties
 dictionary, [U-49](#)
memory tools, [U-96](#)
menu
 Color By, [U-166](#)
 Current Time Controls, [U-25](#), [U-165](#)
 Edit, [U-167](#), [U-168](#)
 Help, [U-167](#)
 Line Style, [U-34](#)
 Marker Style, [U-34](#)
 VCR Controls, [U-25](#), [U-165](#)
 View, [U-167](#)
menu entry
 Plot Over Line, [U-34](#)
 Save Animation, [U-169](#)
 Save Screenshot, [U-169](#)
 Settings, [U-168](#)
 Show Color Legend, [U-25](#)
 Solid Color, [U-166](#)
 Toolbars, [U-167](#)
 View Settings..., [U-24](#)
 View Settings, [U-24](#), [U-167](#)
 Wireframe, [U-166](#)
 fvSchemes, [U-51](#)
mergeLevels keyword, [U-123](#)
mergeMeshes utility, [U-90](#)
mergeOrSplitBaffles utility, [U-90](#)
mergePatchPairs keyword, [U-138](#)
mergeTolerance keyword, [U-146](#)
mesh
 1-dimensional, [U-130](#)
 1D, [U-130](#)
 2-dimensional, [U-130](#)
 2D, [U-130](#)
 axi-symmetric, [U-130](#)
 basic, [P-27](#)
 block structured, [U-136](#)
 decomposition, [U-79](#)
 description, [U-127](#)
 finite volume, [P-27](#)
 generation, [U-136](#), [U-145](#)
 grading, [U-136](#), [U-140](#)
 grading, example of, [P-49](#)
 non-orthogonal, [P-41](#)
 refinement, [P-58](#)
 resolution, [U-30](#)
 specification, [U-127](#)
 split-hex, [U-145](#)
 Stereolithography (STL), [U-145](#)
 surface, [U-145](#)
 validity constraints, [U-127](#)
Mesh Parts window panel, [U-23](#)
meshes tools, [U-96](#)
meshQualityControls keyword, [U-146](#)
meshTools
 library, [U-97](#)
message passing interface
 openMPI, [U-81](#)
method keyword, [U-81](#)
metis
 keyword entry, [U-81](#)
metisDecomp model, [U-97](#)
MGridGenGAMGAgglomeration
 library, [U-97](#)
MGridGen
 keyword entry, [U-123](#)
mhdFoam solver, [P-65](#), [U-88](#)
midPoint
 keyword entry, [U-116](#)
midPoint keyword, [U-175](#)
midPointAndFace keyword, [U-175](#)
min
 tensor member function, [P-21](#)
minArea keyword, [U-153](#)
minDeterminant keyword, [U-153](#)
minFaceWeight keyword, [U-153](#)

minFlatness keyword, [U-153](#)
minMedianAxisAngle keyword, [U-152](#)
MINMOD differencing, [P-34](#)
minRefinementCells keyword, [U-148](#)
minThickness keyword, [U-152](#)
minTriangleTwist keyword, [U-153](#)
minTwist keyword, [U-153](#)
minVol keyword, [U-153](#)
minVolRatio keyword, [U-153](#)
mirrorMesh utility, [U-90](#)
mixed
 boundary condition, [U-136](#)
mixedSmagorinsky model, [U-101](#)
mixtureAdiabaticFlameT utility, [U-95](#)
mode keyword, [U-149](#)
model
 APIfunctions, [U-99](#)
 BirdCarreau, [U-101](#)
 CrossPowerLaw, [U-101](#)
 DeardorffDiffStress, [U-101](#)
 GuldersEGRLaminarFlameSpeed, [U-98](#)
 GuldersLaminarFlameSpeed, [U-98](#)
 HerschelBulkley, [U-101](#)
 LRDDiffStress, [U-101](#)
 LRR, [U-100](#)
 LamBremhorstKE, [U-100](#)
 LaunderGibsonRSTM, [U-100](#)
 LaunderSharmaKE, [U-100](#)
 LienCubicKELowRe, [U-100](#)
 LienCubicKE, [U-100](#)
 LienLeschzinerLowRe, [U-100](#)
 NSRDSfunctions, [U-99](#)
 Newtonian, [U-101](#)
 NonlinearKEShih, [U-100](#)
 PrandtlDelta, [U-101](#)
 RNGkEpsilon, [U-100](#)
 RaviPetersen, [U-98](#)
 Smagorinsky2, [U-101](#)
 Smagorinsky, [U-101](#)
 SpalartAllmarasDDES, [U-101](#)
 SpalartAllmarasIDDES, [U-101](#)
 SpalartAllmaras, [U-100](#), [U-101](#)
 adiabaticPerfectFluid, [U-99](#), [U-179](#)
anisotropicFilter, [U-100](#)
basicMultiComponentMixture, [U-98](#), [U-180](#)
chemistryModel, [U-99](#)
chemistrySolver, [U-99](#)
constTransport, [U-99](#), [U-180](#)
constant, [U-98](#)
cubeRootVolDelta, [U-101](#)
decompose, [U-97](#)
distributed, [U-97](#)
dynLagrangian, [U-101](#)
dynOneEqEddy, [U-101](#)
eConstThermo, [U-99](#), [U-179](#)
egrMixture, [U-98](#), [U-180](#)
hConstThermo, [U-99](#), [U-179](#)
hPolynomialThermo, [U-99](#), [U-180](#)
hePsiThermo, [U-98](#), [U-180](#)
heRhoThermo, [U-98](#), [U-180](#)
heheupsiReactionThermo, [U-98](#), [U-180](#)
homogenousDynOneEqEddy, [U-101](#)
homogenousDynSmagorinsky, [U-101](#)
homogeneousMixture, [U-98](#), [U-180](#)
icoPolynomial, [U-99](#), [U-179](#)
incompressiblePerfectGas, [U-99](#), [U-179](#)
inhomogeneousMixture, [U-98](#), [U-180](#)
interfaceProperties, [U-102](#)
janafThermo, [U-99](#), [U-180](#)
kEpsilon, [U-100](#)
kOmegaSSTSAS, [U-101](#)
kOmegaSST, [U-100](#)
kOmega, [U-100](#)
kkLOmega, [U-100](#)
laminar, [U-100](#), [U-101](#)
laplaceFilter, [U-100](#)
lowReOneEqEddy, [U-101](#)
maxDeltaxyz, [U-101](#)
metisDecomp, [U-97](#)
mixedSmagorinsky, [U-101](#)
multiComponentMixture, [U-98](#), [U-180](#)
oneEqEddy, [U-101](#)
perfectFluid, [U-99](#), [U-179](#)
polynomialTransport, [U-99](#), [U-180](#)
powerLaw, [U-101](#)

psiReactionThermo, [U-98](#), [U-180](#)
psiReactionThermo, [U-98](#), [U-180](#)
ptsotchDecomp, [U-97](#)
pureMixture, [U-98](#), [U-180](#)
qZeta, [U-100](#)
reactingMixture, [U-98](#), [U-180](#)
realizableKE, [U-100](#)
reconstruct, [U-97](#)
rhoConst, [U-99](#), [U-179](#)
rhoReactionThermo, [U-98](#), [U-180](#)
scaleSimilarity, [U-101](#)
scotchDecomp, [U-97](#)
simpleFilter, [U-100](#)
singleStepReactingMixture, [U-98](#), [U-180](#)
smoothDelta, [U-101](#)
specieThermo, [U-99](#), [U-180](#)
spectEddyVisc, [U-101](#)
sutherlandTransport, [U-99](#), [U-180](#)
v2f, [U-100](#)
vanDriestDelta, [U-101](#)
veryInhomogeneousMixture, [U-98](#), [U-180](#)
modifyMesh utility, [U-91](#)
molecularMeasurements
 library, [U-97](#)
molecule
 library, [U-97](#)
molWeight keyword, [U-181](#)
moveDynamicMesh utility, [U-90](#)
moveEngineMesh utility, [U-90](#)
moveMesh utility, [U-90](#)
movingWallVelocity
 boundary condition, [U-137](#)
MPI
 openMPI, [U-81](#)
MRFInterFoam solver, [U-85](#)
MRFMultiphaseInterFoam solver, [U-85](#)
mshToFoam utility, [U-89](#)
mu keyword, [U-182](#)
multiComponentMixture model, [U-98](#), [U-180](#)
multigrid
 geometric-algebraic, [U-122](#)
multiphaseEulerFoam solver, [U-85](#)
multiphaseInterFoam solver, [U-86](#)
MUSCL
 keyword entry, [U-116](#)

N

n keyword, [U-81](#)
nabla
 operator, [P-23](#)
nAlphaSubCycles keyword, [U-61](#)
nBufferCellsNoExtrude keyword, [U-152](#)
nCellsBetweenLevels keyword, [U-148](#)
neighbour
 dictionary, [U-129](#)
neighbourPatch keyword, [U-142](#)
netgenNeutralToFoam utility, [U-89](#)
Newtonian
 keyword entry, [U-58](#)
Newtonian model, [U-101](#)
nextWrite
 keyword entry, [U-111](#)
nFaces keyword, [U-130](#)
nFinestSweeps keyword, [U-123](#)
nGrow keyword, [U-152](#)
nLayerIter keyword, [U-152](#)
nMoles keyword, [U-181](#)
non-orthogonal mesh, [P-41](#)
nonBlocking
 keyword entry, [U-78](#)
none
 keyword entry, [U-115](#), [U-122](#)
NonlinearKEShih model, [U-100](#)
nonNewtonianIcoFoam solver, [U-84](#)
noWriteNow
 keyword entry, [U-111](#)
nPostSweeps keyword, [U-123](#)
nPreSweeps keyword, [U-123](#)
nRelaxedIter keyword, [U-152](#)
nRelaxIter keyword, [U-150](#), [U-152](#)
nSmoothNormals keyword, [U-152](#)
nSmoothPatch keyword, [U-150](#)
nSmoothScale keyword, [U-153](#)
nSmoothSurfaceNormals keyword, [U-152](#)
nSmoothThickness keyword, [U-152](#)

nSolveIter keyword, [U-150](#)

NSRDSfunctions model, [U-99](#)

null

 keyword entry, [U-174](#)

numberOfSubdomains keyword, [U-81](#)

O

object keyword, [U-105](#)

objToVTK utility, [U-90](#)

ODE

 library, [U-97](#)

oneEqEddy model, [U-101](#)

Opacity text box, [U-167](#)

opaqueSolid

 library, [U-98](#)

OpenFOAM

 applications, [U-67](#)

 file format, [U-104](#)

 libraries, [U-67](#)

OpenFOAM

 library, [U-96](#)

OpenFOAM file syntax

 //, [U-104](#)

openMPI

 message passing interface, [U-81](#)

 MPI, [U-81](#)

operator

 scalar, [P-24](#)

 vector, [P-23](#)

Options window, [U-168](#)

options file, [U-71](#)

order keyword, [U-81](#)

Orientation Axes button, [U-24](#), [U-167](#)

orientFaceZone utility, [U-90](#)

OSspecific

 library, [U-97](#)

outer product, **see** tensor, outer product

outlet

 boundary condition, [P-65](#)

outletInlet

 boundary condition, [U-137](#)

outside

 keyword entry, [U-149](#)

owner

dictionary, [U-129](#)

P

p field, [U-22](#)

P1

 library, [U-98](#)

p rhgRefCell keyword, [U-124](#)

p rhgRefValue keyword, [U-124](#)

pairPatchAgglomeration

 library, [U-97](#)

paraFoam, [U-23](#), [U-163](#)

parallel

 running, [U-79](#)

partialSlip

 boundary condition, [U-137](#)

particleTracks utility, [U-93](#)

patch

 boundary condition, [U-134](#)

patch

 keyword entry, [U-135](#), [U-176](#)

patchAverage utility, [U-92](#)

patches keyword, [U-138](#)

patchIntegrate utility, [U-92](#)

patchMap keyword, [U-160](#)

patchSummary utility, [U-95](#)

PBiCG

 keyword entry, [U-121](#)

PCG

 keyword entry, [U-121](#)

pdfPlot utility, [U-93](#)

PDRFoam solver, [U-86](#)

PDRMesh utility, [U-91](#)

Pe utility, [U-92](#)

perfectFluid model, [U-99](#), [U-179](#)

permutation symbol, [P-15](#)

pimpleDyMFoam solver, [U-84](#)

pimpleFoam solver, [U-84](#)

Pipeline Browser window, [U-23](#), [U-164](#)

PISO

 dictionary, [U-23](#)

pisoFoam solver, [U-17](#), [U-84](#)

Plot Over Line

 menu entry, [U-34](#)

plot3dToFoam utility, [U-90](#)

pointField class, [P-27](#)
pointField<Type> template class, [P-29](#)
points
 dictionary, [U-129, U-136](#)
polyBoundaryMesh class, [P-27](#)
polyDualMesh utility, [U-90](#)
polyLine
 keyword entry, [U-139](#)
polyMesh directory, [U-104, U-129](#)
polyMesh class, [P-27, U-127, U-129](#)
polynomialTransport model, [U-99, U-180](#)
polyPatch class, [P-27](#)
polyPatchList class, [P-27](#)
polySpline
 keyword entry, [U-139](#)
porousInterFoam solver, [U-86](#)
porousSimpleFoam solver, [U-84](#)
post-processing, [U-163](#)
 post-processing
 paraFoam, [U-163](#)
postCalc
 library, [U-96](#)
postChannel utility, [U-93](#)
potentialFreeSurfaceFoam solver, [U-86](#)
potential
 library, [U-97](#)
potentialFoam solver, [P-42, U-83](#)
pow
 tensor member function, [P-21](#)
powerLaw model, [U-101](#)
pPrime2 utility, [U-92](#)
Pr keyword, [U-182](#)
PrandtlDelta model, [U-101](#)
preconditioner keyword, [U-121, U-122](#)
pRefCell keyword, [U-23, U-124](#)
pRefValue keyword, [U-23, U-124](#)
pressure keyword, [U-49](#)
pressure waves
 in liquids, [P-58](#)
pressureDirectedInletVelocity
 boundary condition, [U-137](#)
pressureInletVelocity
 boundary condition, [U-137](#)
pressureOutlet

boundary condition, [P-59](#)
pressureTransmissive
 boundary condition, [U-137](#)
primitive
 library, [P-19](#)
primitives tools, [U-96](#)
printCoeffs keyword, [U-41, U-184](#)
processorWeights keyword, [U-80](#)
probeLocations utility, [U-93](#)
process
 background, [U-24, U-79](#)
 foreground, [U-24](#)
processor
 boundary condition, [U-135](#)
processor
 keyword entry, [U-135](#)
processorN directory, [U-80](#)
processorWeights keyword, [U-81](#)
Properties window panel, [U-25, U-164](#)
psiReactionThermo model, [U-98, U-180](#)
psiReactionThermo model, [U-98, U-180](#)
ptot utility, [U-93](#)
ptsotchDecomp model, [U-97](#)
pureMixture model, [U-98, U-180](#)
purgeWrite keyword, [U-112](#)
PV3FoamReader
 library, [U-163](#)
PVFoamReader
 library, [U-163](#)

Q

Q utility, [U-92](#)
QUICK
 keyword entry, [U-119](#)
qZeta model, [U-100](#)

R

R utility, [U-92](#)
radiationModels
 library, [U-98](#)
randomProcesses
 library, [U-97](#)
RASModel
 keyword entry, [U-40, U-183](#)

RASModel keyword, [U-184](#)
RaviPetersen model, [U-98](#)
raw
 keyword entry, [U-112, U-174](#)
reactingFoam solver, [U-86](#)
reactingMixture model, [U-98, U-180](#)
reactingParcelFilmFoam solver, [U-87](#)
reactingParcelFoam solver, [U-87](#)
reactionThermophysicalModels
 library, [U-98](#)
realizableKE model, [U-100](#)
reconstruct model, [U-97](#)
reconstructPar utility, [U-83](#)
reconstructParMesh utility, [U-95](#)
redistributePar utility, [U-95](#)
refGradient keyword, [U-136](#)
refineHexMesh utility, [U-91](#)
refinementRegions keyword, [U-149](#)
refinementLevel utility, [U-91](#)
refinementRegions keyword, [U-148, U-150](#)
refinementSurfaces keyword, [U-148](#)
refineMesh utility, [U-90](#)
refineWallLayer utility, [U-91](#)
Refresh Times button, [U-25](#)
regions keyword, [U-58](#)
relative tolerance, [U-121](#)
relativeSizes keyword, [U-152](#)
relaxed keyword, [U-153](#)
relTol keyword, [U-52, U-121](#)
removeFaces utility, [U-91](#)
Render View window, [U-168](#)
Render View window panel, [U-168](#)
renumberMesh utility, [U-90](#)
Rescale to Data Range button, [U-25](#)
Reset button, [U-164](#)
resolveFeatureAngle keyword, [U-148](#)
restart, [U-38](#)
Reynolds number, [U-17, U-21](#)
rhoPorousSimpleFoam solver, [U-84](#)
rhoReactingBuoyantFoam solver, [U-86](#)
rhoCentralDyMFoam solver, [U-84](#)
rhoCentralFoam solver, [U-84](#)
rhoConst model, [U-99, U-179](#)
rhoLTSPlmleFoam solver, [U-84](#)
rhoPimpleFoam solver, [U-84](#)
rhoPimplecFoam solver, [U-84](#)
rhoReactingFoam solver, [U-86](#)
rhoReactionThermo model, [U-98, U-180](#)
rhoSimpleFoam solver, [U-84](#)
rhoSimplecFoam solver, [U-84](#)
rmdepall script/alias, [U-74](#)
RNGkEpsilon model, [U-100](#)
roots keyword, [U-81, U-82](#)
rotateMesh utility, [U-90](#)
run
 parallel, [U-79](#)
run directory, [U-103](#)
runTime
 keyword entry, [U-31, U-111](#)
runTimeModifiable keyword, [U-112](#)

S

sammToFoam utility, [U-90](#)
sample utility, [U-93, U-173](#)
sampling
 library, [U-96](#)
Save Animation
 menu entry, [U-169](#)
Save Screenshot
 menu entry, [U-169](#)
scalar, [P-12](#)
 operator, [P-24](#)
scalar class, [P-19](#)
scalarField class, [P-25](#)
scalarTransportFoam solver, [U-83](#)
scale
 tensor member function, [P-21](#)
scalePoints utility, [U-157](#)
scaleSimilarity model, [U-101](#)
scheduled
 keyword entry, [U-78](#)
scientific
 keyword entry, [U-112](#)
scotch
 keyword entry, [U-80, U-81](#)
scotchCoeffs keyword, [U-81](#)
scotchDecomp model, [U-97](#)
script/alias

foamCorrectVrt, [U-158](#)
foamJob, [U-177](#)
foamLog, [U-177](#)
make, [U-69](#)
rmdepall, [U-74](#)
wclean, [U-73](#)
wmake, [U-69](#)
second time derivative, [P-33](#)
Seed window, [U-169](#)
selectCells utility, [U-91](#)
Set Ambient Color button, [U-166](#)
setFields utility, [U-58](#), [U-89](#)
setFormat keyword, [U-174](#)
sets keyword, [U-174](#)
setSet utility, [U-90](#)
setsToZones utility, [U-90](#)
Settings
 menu entry, [U-168](#)
settlingFoam solver, [U-86](#)
SFCD
 keyword entry, [U-116](#), [U-119](#)
shallowWaterFoam solver, [U-84](#)
shape, [U-140](#)
Show Color Legend
 menu entry, [U-25](#)
SI units, [U-107](#)
simpleReactingParcelFoam solver, [U-87](#)
simple
 keyword entry, [U-80](#), [U-81](#)
simpleFilter model, [U-100](#)
simpleFoam solver, [P-50](#), [U-84](#)
simpleGrading keyword, [U-140](#)
simpleSpline
 keyword entry, [U-139](#)
simulationType keyword, [U-40](#), [U-59](#), [U-183](#)
singleCellMesh utility, [U-91](#)
singleStepReactingMixture model, [U-98](#), [U-180](#)
skew
 tensor member function, [P-21](#)
skewLinear
 keyword entry, [U-116](#), [U-119](#)
SLGThermo
 library, [U-99](#)
slice class, [P-27](#)
slip
 boundary condition, [U-137](#)
Smagorinsky model, [U-101](#)
Smagorinsky2 model, [U-101](#)
smapToFoam utility, [U-91](#)
smoothDelta model, [U-101](#)
smoother keyword, [U-123](#)
smoothSolver
 keyword entry, [U-121](#)
snap keyword, [U-146](#)
snapControls keyword, [U-146](#)
snappyHexMesh utility
 background mesh, [U-146](#)
 cell removal, [U-149](#)
 cell splitting, [U-147](#)
 mesh layers, [U-150](#)
 meshing process, [U-145](#)
 snapping to surfaces, [U-150](#)
snappyHexMesh utility, [U-89](#), [U-145](#)
snappyHexMeshDict file, [U-145](#)
snGrad
 fvc member function, [P-33](#)
snGradCorrection
 fvc member function, [P-33](#)
snGradSchemes keyword, [U-114](#)
Solid Color
 menu entry, [U-166](#)
solidChemistryModel
 library, [U-99](#)
solidDisplacementFoam solver, [U-88](#)
solidDisplacementFoam solver, [U-50](#)
solidEquilibriumDisplacementFoam solver, [U-88](#)
solidMixtureProperties
 library, [U-100](#)
solidParticle
 library, [U-97](#)
solidProperties
 library, [U-99](#)
solidSpecie
 library, [U-100](#)
solidThermo

library, [U-100](#)
solver
 DPMFoam, [U-87](#)
 LTSInterFoam, [U-85](#)
 LTSReactingFoam, [U-86](#)
 LTSReactingParcelFoam, [U-87](#)
 MRFInterFoam, [U-85](#)
 MRFMultiphaseInterFoam, [U-85](#)
 PDRFoam, [U-86](#)
 SRFPimpleFoam, [U-84](#)
 SRFSimpleFoam, [U-84](#)
 XiFoam, [U-86](#)
 adjointShapeOptimizationFoam, [U-83](#)
 blockMesh, [P-43](#)
 boundaryFoam, [U-84](#)
 buoyantBoussinesqPimpleFoam, [U-86](#)
 buoyantBoussinesqSimpleFoam, [U-86](#)
 buoyantPimpleFoam, [U-86](#)
 buoyantSimpleFoam, [U-86](#)
 cavitatingDyMFoam, [U-85](#)
 cavitatingFoam, [U-85](#)
 chemFoam, [U-86](#)
 chtMultiRegionFoam, [U-87](#)
 chtMultiRegionSimpleFoam, [U-87](#)
 coalChemistryFoam, [U-87](#)
 coldEngineFoam, [U-86](#)
 compressibleInterDyMFoam, [U-85](#)
 compressibleInterFoam, [U-85](#)
 compressibleMultiphaseInterFoam, [U-85](#)
 dnsFoam, [U-86](#)
 dsmcFoam, [U-87](#)
 electrostaticFoam, [U-88](#)
 engineFoam, [U-86](#)
 financialFoam, [U-88](#)
 fireFoam, [U-86](#)
 icoFoam, [U-17, U-21, U-22, U-24, U-84](#)
 icoUncoupledKinematicParcelDyMFoam, [U-87](#)
 interDyMFoam, [U-85](#)
 interFoam, [U-85](#)
 interMixingFoam, [U-85](#)
 interPhaseChangeDyMFoam, [U-85](#)
 interPhaseChangeFoam, [U-85](#)
 laplacianFoam, [U-83](#)
 magneticFoam, [U-88](#)
 mdEquilibrationFoam, [U-87](#)
 mdFoam, [U-87](#)
 mhdFoam, [P-65, U-88](#)
 multiphaseEulerFoam, [U-85](#)
 multiphaseInterFoam, [U-86](#)
 nonNewtonianIcoFoam, [U-84](#)
 pimpleDyMFoam, [U-84](#)
 pimpleFoam, [U-84](#)
 pisoFoam, [U-17, U-84](#)
 porousInterFoam, [U-86](#)
 porousSimpleFoam, [U-84](#)
 potentialFreeSurfaceFoam, [U-86](#)
 potentialFoam, [P-42, U-83](#)
 reactingFoam, [U-86](#)
 reactingParcelFilmFoam, [U-87](#)
 reactingParcelFoam, [U-87](#)
 rhoCentralDyMFoam, [U-84](#)
 rhoCentralFoam, [U-84](#)
 rhoLTSPimpleFoam, [U-84](#)
 rhoPimpleFoam, [U-84](#)
 rhoPimplecFoam, [U-84](#)
 rhoReactingFoam, [U-86](#)
 rhoSimpleFoam, [U-84](#)
 rhoSimplecFoam, [U-84](#)
 rhoPorousSimpleFoam, [U-84](#)
 rhoReactingBuoyantFoam, [U-86](#)
 scalarTransportFoam, [U-83](#)
 settlingFoam, [U-86](#)
 shallowWaterFoam, [U-84](#)
 simpleReactingParcelFoam, [U-87](#)
 simpleFoam, [P-50, U-84](#)
 solidDisplacementFoam, [U-88](#)
 solidDisplacementFoam, [U-50](#)
 solidEquilibriumDisplacementFoam, [U-88](#)
 sonicDyMFoam, [U-84](#)
 sonicFoam, [P-56, U-84](#)
 sonicLiquidFoam, [P-59, U-85](#)

sprayEngineFoam, [U-87](#)
sprayFoam, [U-87](#)
thermoFoam, [U-87](#)
twoLiquidMixingFoam, [U-86](#)
twoPhaseEulerFoam, [U-86](#)
uncoupledKinematicParcelFoam, [U-87](#)
solver keyword, [U-52](#), [U-121](#)
solver relative tolerance, [U-121](#)
solver tolerance, [U-121](#)
solvers keyword, [U-120](#)
sonicDyMFoam solver, [U-84](#)
sonicFoam solver, [P-56](#), [U-84](#)
sonicLiquidFoam solver, [P-59](#), [U-85](#)
source, [P-33](#)
SpalartAllmaras model, [U-100](#), [U-101](#)
SpalartAllmarasDDES model, [U-101](#)
SpalartAllmarasIDDES model, [U-101](#)
specie
 library, [U-99](#)
specie keyword, [U-181](#)
specieThermo model, [U-99](#), [U-180](#)
spectEddyVisc model, [U-101](#)
spline keyword, [U-138](#)
splitCells utility, [U-91](#)
splitMesh utility, [U-91](#)
splitMeshRegions utility, [U-91](#)
spray
 library, [U-97](#)
sprayEngineFoam solver, [U-87](#)
sprayFoam solver, [U-87](#)
sqr
 tensor member function, [P-21](#)
sqrGradGrad
 fvc member function, [P-33](#)
SRFPimpleFoam solver, [U-84](#)
SRFSimpleFoam solver, [U-84](#)
star3ToFoam utility, [U-90](#)
star4ToFoam utility, [U-90](#)
startFace keyword, [U-130](#)
startFrom keyword, [U-22](#), [U-111](#)
starToFoam utility, [U-154](#)
startTime
 keyword entry, [U-22](#), [U-111](#)
startTime keyword, [U-22](#), [U-111](#)
steady flow
 turbulent, [P-49](#)
steadyParticleTracks utility, [U-93](#)
steadyState
 keyword entry, [U-119](#)
Stereolithography (STL), [U-145](#)
stitchMesh utility, [U-91](#)
stl
 keyword entry, [U-174](#)
stopAt keyword, [U-111](#)
strategy keyword, [U-80](#), [U-81](#)
streamFunction utility, [U-92](#)
stress analysis of plate with hole, [U-45](#)
stressComponents utility, [U-92](#)
Style window panel, [U-23](#), [U-166](#)
Su
 fvm member function, [P-33](#)
subsetMesh utility, [U-91](#)
summation convention, [P-13](#)
SUPERBEE differencing, [P-34](#)
supersonic flow, [P-55](#)
supersonic flow over forward step, [P-54](#)
supersonicFreeStream
 boundary condition, [U-137](#)
surfaceLambdaMuSmooth utility, [U-94](#)
surface mesh, [U-145](#)
surfaceAdd utility, [U-93](#)
surfaceAutoPatch utility, [U-93](#)
surfaceBooleanFeatures utility, [U-93](#)
surfaceCheck utility, [U-93](#)
surfaceClean utility, [U-93](#)
surfaceCoarsen utility, [U-93](#)
surfaceConvert utility, [U-93](#)
surfaceFeatureConvert utility, [U-93](#)
surfaceFeatureExtract utility, [U-93](#), [U-148](#)
surfaceField<Type> template class, [P-29](#)
surfaceFilmModels
 library, [U-102](#)
surfaceFind utility, [U-93](#)
surfaceFormat keyword, [U-174](#)
surfaceHookUp utility, [U-93](#)
surfaceInertia utility, [U-94](#)
surfaceMesh tools, [U-96](#)
surfaceMeshConvert utility, [U-94](#)

surfaceMeshConvertTesting utility, [U-94](#)
 surfaceMeshExport utility, [U-94](#)
 surfaceMeshImport utility, [U-94](#)
 surfaceMeshInfo utility, [U-94](#)
 surfaceMeshTriangulate utility, [U-94](#)
 surfaceNormalFixedValue
 boundary condition, [U-137](#)
 surfaceOrient utility, [U-94](#)
 surfacePointMerge utility, [U-94](#)
 surfaceRedistributePar utility, [U-94](#)
 surfaceRefineRedGreen utility, [U-94](#)
 surfaces keyword, [U-174](#)
 surfaceSplitByPatch utility, [U-94](#)
 surfaceSplitByTopology utility, [U-94](#)
 surfaceSplitNonManifolds utility, [U-94](#)
 surfaceSubset utility, [U-94](#)
 surfaceToPatch utility, [U-94](#)
 surfaceTransformPoints utility, [U-94](#)
 surfMesh
 library, [U-97](#)
 SuSp
 fvm member function, [P-33](#)
 sutherlandTransport model, [U-99](#), [U-180](#)
 symm
 tensor member function, [P-21](#)
 symmetryPlane
 boundary condition, [P-59](#), [U-134](#)
 symmetryPlane
 keyword entry, [U-135](#)
 symmTensorField class, [P-25](#)
 symmTensorThirdField class, [P-25](#)
 system directory, [P-46](#), [U-104](#)
 systemCall
 library, [U-96](#)

T

T()
 tensor member function, [P-21](#)
 Tcommon keyword, [U-182](#)
 template class
 GeometricBoundaryField, [P-28](#)
 fvMatrix, [P-29](#)
 dimensioned<Type>, [P-21](#)
 FieldField<Type>, [P-28](#)
 Field<Type>, [P-25](#)
 geometricField<Type>, [P-28](#)
 List<Type>, [P-25](#)
 pointField<Type>, [P-29](#)
 surfaceField<Type>, [P-29](#)
 volField<Type>, [P-29](#)
 temporal discretisation, [P-38](#)
 Crank Nicholson, [P-38](#)
 Euler implicit, [P-38](#)
 explicit, [P-38](#)
 in OpenFOAM, [P-39](#)
 temporalInterpolate utility, [U-93](#)
 tensor, [P-11](#)
 addition, [P-13](#)
 algebraic operations, [P-13](#)
 algebraic operations in OpenFOAM, [P-19](#)
 antisymmetric, [see](#) tensor, skew
 calculus, [P-23](#)
 classes in OpenFOAM, [P-19](#)
 cofactors, [P-18](#)
 component average, [P-16](#)
 component maximum, [P-16](#)
 component minimum, [P-16](#)
 determinant, [P-18](#)
 deviatoric, [P-17](#)
 diagonal, [P-17](#)
 dimension, [P-12](#)
 double inner product, [P-15](#)
 geometric transformation, [P-16](#)
 Hodge dual, [P-18](#)
 hydrostatic, [P-17](#)
 identities, [P-17](#)
 identity, [P-16](#)
 inner product, [P-14](#)
 inverse, [P-18](#)
 magnitude, [P-16](#)
 magnitude squared, [P-16](#)
 mathematics, [P-11](#)
 notation, [P-13](#)
 nth power, [P-16](#)
 outer product, [P-15](#)
 rank, [P-12](#)

rank 3, [P-12](#)
scalar division, [P-14](#)
scalar multiplication, [P-13](#)
scale function, [P-16](#)
second rank, [P-12](#)
skew, [P-17](#)
square of, [P-16](#)
subtraction, [P-13](#)
symmetric, [P-17](#)
symmetric rank 2, [P-12](#)
symmetric rank 3, [P-12](#)
trace, [P-17](#)
transformation, [P-16](#)
transpose, [P-12](#), [P-17](#)
triple inner product, [P-15](#)
vector cross product, [P-15](#)
tensor class, [P-19](#)
tensor member function
 *, [P-21](#)
 +, [P-21](#)
 -, [P-21](#)
 /, [P-21](#)
 &, [P-21](#)
 &&, [P-21](#)
 ^, [P-21](#)
 cmptAv, [P-21](#)
 cofactors, [P-21](#)
 det, [P-21](#)
 dev, [P-21](#)
 diag, [P-21](#)
 I, [P-21](#)
 inv, [P-21](#)
 mag, [P-21](#)
 magSqr, [P-21](#)
 max, [P-21](#)
 min, [P-21](#)
 pow, [P-21](#)
 scale, [P-21](#)
 skew, [P-21](#)
 sqr, [P-21](#)
 symm, [P-21](#)
 T(), [P-21](#)
 tr, [P-21](#)
 transform, [P-21](#)
tensorField class, [P-25](#)
tensorThirdField class, [P-25](#)
tetgenToFoam utility, [U-90](#)
text box
 Opacity, [U-167](#)
thermalProperties
 dictionary, [U-50](#)
thermodynamics keyword, [U-181](#)
thermoFoam solver, [U-87](#)
thermophysical
 library, [U-179](#)
thermophysicalFunctions
 library, [U-99](#)
thermophysicalProperties
 dictionary, [U-179](#)
thermoType keyword, [U-179](#)
Thigh keyword, [U-182](#)
time
 control, [U-111](#)
time derivative, [P-33](#)
 first, [P-35](#)
 second, [P-33](#), [P-35](#)
time step, [U-22](#)
timeFormat keyword, [U-112](#)
timePrecision keyword, [U-112](#)
timeScheme keyword, [U-114](#)
timeStamp
 keyword entry, [U-78](#)
timeStampMaster
 keyword entry, [U-78](#)
timeStep
 keyword entry, [U-22](#), [U-31](#), [U-111](#)
Tlow keyword, [U-182](#)
tolerance
 solver, [U-121](#)
 solver relative, [U-121](#)
tolerance keyword, [U-52](#), [U-121](#), [U-150](#)
Toolbars
 menu entry, [U-167](#)
tools
 algorithms, [U-96](#)
 cfdTools, [U-96](#)
 containers, [U-96](#)
 db, [U-96](#)

dimensionSet, [U-96](#)
dimensionedTypes, [U-96](#)
fields, [U-96](#)
finiteVolume, [U-96](#)
fvMatrices, [U-96](#)
fvMesh, [U-96](#)
global, [U-96](#)
graph, [U-96](#)
interpolations, [U-96](#)
interpolation, [U-96](#)
matrices, [U-96](#)
memory, [U-96](#)
meshes, [U-96](#)
primitives, [U-96](#)
surfaceMesh, [U-96](#)
volMesh, [U-96](#)
topoChangerFvMesh
 library, [U-97](#)
topoSet utility, [U-91](#)
topoSetSource keyword, [U-58](#)
totalPressure
 boundary condition, [U-137](#)
tr
 tensor member function, [P-21](#)
trace, [see](#) tensor, trace
traction keyword, [U-49](#)
transform
 tensor member function, [P-21](#)
transformPoints utility, [U-91](#)
transport keyword, [U-181](#)
transportProperties
 dictionary, [U-21](#), [U-38](#), [U-41](#)
transportProperties file, [U-58](#)
triple inner product, [P-15](#)
triSurface
 library, [U-97](#)
Ts keyword, [U-182](#)
turbulence
 dissipation, [U-39](#)
 kinetic energy, [U-39](#)
 length scale, [U-40](#)
turbulence
 library, [U-97](#)
turbulence keyword, [U-184](#)
turbulence model
 RAS, [U-39](#)
turbulenceProperties
 dictionary, [U-40](#), [U-59](#), [U-183](#)
turbulent flow
 steady, [P-49](#)
turbulentInlet
 boundary condition, [U-137](#)
tutorials
 breaking of a dam, [U-55](#)
 lid-driven cavity flow, [U-17](#)
 stress analysis of plate with hole, [U-45](#)
tutorials directory, [P-41](#), [U-17](#)
twoLiquidMixingFoam solver, [U-86](#)
twoPhaseEulerFoam solver, [U-86](#)
twoPhaseProperties
 library, [U-102](#)
type keyword, [U-132](#), [U-133](#)

U

U field, [U-22](#)
Ucomponents utility, [P-66](#)
UMIST
 keyword entry, [U-115](#)
uncompressed
 keyword entry, [U-112](#)
uncorrected
 keyword entry, [U-117](#), [U-118](#)
uncoupledKinematicParcelFoam solver, [U-87](#)
uniform keyword, [U-175](#)
units
 base, [U-107](#)
 of measurement, [P-21](#), [U-107](#)
 S.I. base, [P-21](#)
 SI, [U-107](#)
 Syst`eme International, [U-107](#)
 United States Customary System, [U-107](#)
 USCS, [U-107](#)
Update GUI button, [U-165](#)
uprime utility, [U-92](#)
upwind
 keyword entry, [U-116](#), [U-119](#)

upwind differencing, [P-34](#), [U-61](#)
USCS units, [U-107](#)
Use Parallel Projection button, [U-24](#)
Use parallel projection button, [U-167](#)
utility
 Co, [U-92](#)
 Lambda2, [U-92](#)
 Mach, [U-92](#)
 PDRMesh, [U-91](#)
 Pe, [U-92](#)
 Q, [U-92](#)
 R, [U-92](#)
 Ucomponents, [P-66](#)
 adiabaticFlameT, [U-95](#)
 ansysToFoam, [U-89](#)
 applyBoundaryLayer, [U-88](#)
 applyWallFunctionBoundaryCondition
 s,
 [U-88](#)
 attachMesh, [U-90](#)
 autoPatch, [U-90](#)
 autoRefineMesh, [U-91](#)
 blockMesh, [U-37](#), [U-89](#), [U-136](#)
 boxTurb, [U-88](#)
 ccm26ToFoam, [U-89](#)
 cfx4ToFoam, [U-89](#), [U-154](#)
 changeDictionary, [U-88](#)
 checkMesh, [U-90](#), [U-155](#)
 chemkinToFoam, [U-95](#)
 collapseEdges, [U-91](#)
 combinePatchFaces, [U-91](#)
 createBaffles, [U-90](#)
 createPatch, [U-90](#)
 createTurbulenceFields, [U-92](#)
 createExternalCoupledPatchGeometry,
 [U-88](#)
 datToFoam, [U-89](#)
 decomposePar, [U-79](#), [U-80](#), [U-95](#)
 deformedGeom, [U-90](#)
 dsmcFieldsCalc, [U-93](#)
 dsmcInitialise, [U-88](#)
 engineCompRatio, [U-93](#)
 engineSwirl, [U-88](#)
 ensight74FoamExec, [U-172](#)
 ensightFoamReader, [U-91](#)
 enstrophy, [U-92](#)
 equilibriumCO, [U-95](#)
 equilibriumFlameT, [U-95](#)
 execFlowFunctionObjects, [U-93](#)
 expandDictionary, [U-95](#)
 extrude2DMesh, [U-89](#)
 extrudeMesh, [U-89](#)
 extrudeToRegionMesh, [U-89](#)
 faceAgglomerate, [U-88](#)
 flattenMesh, [U-90](#)
 flowType, [U-92](#)
 fluent3DMeshToFoam, [U-89](#)
 fluentMeshToFoam, [U-89](#), [U-154](#)
 foamCalc, [U-33](#), [U-93](#)
 foamDataToFluent, [U-91](#), [U-170](#)
 foamDebugSwitches, [U-95](#)
 foamFormatConvert, [U-95](#)
 foamHelp, [U-95](#)
 foamInfoExec, [U-95](#)
 foamListTimes, [U-93](#)
 foamMeshToFluent, [U-89](#), [U-170](#)
 foamToEnsightParts, [U-91](#)
 foamToEnsight, [U-91](#)
 foamToGMV, [U-91](#)
 foamToStarMesh, [U-89](#)
 foamToSurface, [U-89](#)
 foamToTecplot360, [U-91](#)
 foamToVTK, [U-91](#)
 foamUpgradeCyclics, [U-88](#)
 foamUpgradeFvSolution, [U-88](#)
 foamyHexMesh, [U-89](#)
 foamyQuadMesh, [U-89](#)
 foamyHexMeshBackgroundMesh, [U-
 89](#)
 foamyHexMeshSurfaceSimplify, [U-89](#)
 gambitToFoam, [U-89](#), [U-154](#)
 gmshToFoam, [U-89](#)
 ideasToFoam, [U-154](#)
 ideasUnvToFoam, [U-89](#)
 insideCells, [U-90](#)
 kivaToFoam, [U-89](#)
 mapFields, [U-30](#), [U-37](#), [U-41](#), [U-54](#), [U-
 89](#),

U-160
mdInitialise, [U-89](#)
mergeMeshes, [U-90](#)
mergeOrSplitBaffles, [U-90](#)
mirrorMesh, [U-90](#)
mixtureAdiabaticFlameT, [U-95](#)
modifyMesh, [U-91](#)
moveDynamicMesh, [U-90](#)
moveEngineMesh, [U-90](#)
moveMesh, [U-90](#)
mshToFoam, [U-89](#)
netgenNeutralToFoam, [U-89](#)
objToVTK, [U-90](#)
orientFaceZone, [U-90](#)
pPrime2, [U-92](#)
particleTracks, [U-93](#)
patchAverage, [U-92](#)
patchIntegrate, [U-92](#)
patchSummary, [U-95](#)
pdfPlot, [U-93](#)
plot3dToFoam, [U-90](#)
polyDualMesh, [U-90](#)
postChannel, [U-93](#)
probeLocations, [U-93](#)
ptot, [U-93](#)
reconstructParMesh, [U-95](#)
reconstructPar, [U-83](#)
redistributePar, [U-95](#)
refineHexMesh, [U-91](#)
refineMesh, [U-90](#)
refineWallLayer, [U-91](#)
refinementLevel, [U-91](#)
removeFaces, [U-91](#)
renumberMesh, [U-90](#)
rotateMesh, [U-90](#)
sammToFoam, [U-90](#)
sample, [U-93](#), [U-173](#)
scalePoints, [U-157](#)
selectCells, [U-91](#)
setFields, [U-58](#), [U-89](#)
setSet, [U-90](#)
setsToZones, [U-90](#)
singleCellMesh, [U-91](#)
smapToFoam, [U-91](#)
snappyHexMesh, [U-89](#), [U-145](#)
splitCells, [U-91](#)
splitMeshRegions, [U-91](#)
splitMesh, [U-91](#)
star3ToFoam, [U-90](#)
star4ToFoam, [U-90](#)
starToFoam, [U-154](#)
steadyParticleTracks, [U-93](#)
stitchMesh, [U-91](#)
streamFunction, [U-92](#)
stressComponents, [U-92](#)
subsetMesh, [U-91](#)
surfaceLambdaMuSmooth, [U-94](#)
surfaceAdd, [U-93](#)
surfaceAutoPatch, [U-93](#)
surfaceBooleanFeatures, [U-93](#)
surfaceCheck, [U-93](#)
surfaceClean, [U-93](#)
surfaceCoarsen, [U-93](#)
surfaceConvert, [U-93](#)
surfaceFeatureConvert, [U-93](#)
surfaceFeatureExtract, [U-93](#), [U-148](#)
surfaceFind, [U-93](#)
surfaceHookUp, [U-93](#)
surfaceInertia, [U-94](#)
surfaceMeshConvertTesting, [U-94](#)
surfaceMeshConvert, [U-94](#)
surfaceMeshExport, [U-94](#)
surfaceMeshImport, [U-94](#)
surfaceMeshInfo, [U-94](#)
surfaceMeshTriangulate, [U-94](#)
surfaceOrient, [U-94](#)
surfacePointMerge, [U-94](#)
surfaceRedistributePar, [U-94](#)
surfaceRefineRedGreen, [U-94](#)
surfaceSplitByPatch, [U-94](#)
surfaceSplitByTopology, [U-94](#)
surfaceSplitNonManifolds, [U-94](#)
surfaceSubset, [U-94](#)
surfaceToPatch, [U-94](#)
surfaceTransformPoints, [U-94](#)
temporalInterpolate, [U-93](#)
tetgenToFoam, [U-90](#)
topoSet, [U-91](#)

transformPoints, [U-91](#)
 uprime, [U-92](#)
 viewFactorsGen, [U-89](#)
 vorticity, [U-92](#)
 vtkUnstructuredToFoam, [U-90](#)
 wallFunctionTable, [U-89](#)
 wallGradU, [U-92](#)
 wallHeatFlux, [U-92](#)
 wallShearStress, [U-92](#)
 wdot, [U-93](#)
 writeCellCentres, [U-93](#)
 writeMeshObj, [U-90](#)
 yPlusLES, [U-92](#)
 yPlusRAS, [U-92](#)
 zipUpMesh, [U-91](#)
 utilityFunctionObjects
 library, [U-96](#)

V

v2f model, [U-100](#)
 value keyword, [U-21](#), [U-136](#)
 valueFraction keyword, [U-136](#)
 van Leer differencing, [P-34](#)
 vanDriestDelta model, [U-101](#)
 vanLeer
 keyword entry, [U-116](#)
 VCR Controls menu, [U-25](#), [U-165](#)
 vector, [P-12](#)
 operator, [P-23](#)
 unit, [P-16](#)
 vector class, [P-19](#), [U-107](#)
 vector product, **see** tensor, vector cross
 product
 vectorField class, [P-25](#)
 version keyword, [U-105](#)
 vertices keyword, [U-20](#), [U-138](#), [U-139](#)
 veryInhomogeneousMixture model, [U-98](#),
[U-180](#)
 View menu, [U-167](#)
 View Settings
 menu entry, [U-24](#), [U-167](#)
 View Settings (Render View) window, [U-167](#)
 View Settings...

menu entry, [U-24](#)
 viewFactor
 library, [U-98](#)
 viewFactorsGen utility, [U-89](#)
 viscosity
 kinematic, [U-21](#), [U-41](#)
 volField<Type> template class, [P-29](#)
 volMesh tools, [U-96](#)
 vorticity utility, [U-92](#)
 vtk
 keyword entry, [U-174](#)
 vtkFoam
 library, [U-163](#)
 vtkPV3Foam
 library, [U-163](#)
 vtkUnstructuredToFoam utility, [U-90](#)

W

wall
 boundary condition, [P-59](#), [P-65](#), [U-57](#),
[U-134](#)
 wall
 keyword entry, [U-135](#)
 wallFunctionTable utility, [U-89](#)
 wallGradU utility, [U-92](#)
 wallHeatFlux utility, [U-92](#)
 Wallis
 library, [U-99](#)
 wallShearStress utility, [U-92](#)
 wclean script/alias, [U-73](#)
 wdot utility, [U-93](#)
 wedge
 boundary condition, [U-130](#), [U-135](#), [U-144](#)
 wedge
 keyword entry, [U-135](#)
 window
 Color Legend, [U-27](#)
 Options, [U-168](#)
 Pipeline Browser, [U-23](#), [U-164](#)
 Render View, [U-168](#)
 Seed, [U-169](#)
 View Settings (Render View), [U-167](#)
 window panel

Animations, [U-168](#)
Annotation, [U-24](#), [U-167](#)
Charts, [U-168](#)
Color Legend, [U-166](#)
Color Scale, [U-166](#)
Colors, [U-168](#)
Display, [U-23](#), [U-25](#), [U-164](#), [U-165](#)
General, [U-167](#), [U-168](#)
Information, [U-164](#)
Lights, [U-167](#)
Mesh Parts, [U-23](#)
Properties, [U-25](#), [U-164](#)
Render View, [U-168](#)
Style, [U-23](#), [U-166](#)
Wireframe
 menu entry, [U-166](#)
WM ARCH
 environment variable, [U-74](#)
WM ARCH OPTION
 environment variable, [U-74](#)
WM COMPILE OPTION
 environment variable, [U-74](#)
WM COMPILER
 environment variable, [U-74](#)
WM COMPILER BIN
 environment variable, [U-74](#)
WM COMPILER DIR
 environment variable, [U-74](#)
WM COMPILER LIB
 environment variable, [U-74](#)
WM DIR
 environment variable, [U-74](#)
WM MPLIB
 environment variable, [U-74](#)
WM OPTIONS
 environment variable, [U-74](#)
WM PRECISION OPTION
 environment variable, [U-74](#)
WM PROJECT
 environment variable, [U-74](#)
WM PROJECT DIR
 environment variable, [U-74](#)
WM PROJECT INST DIR
 environment variable, [U-74](#)

WM PROJECT USER DIR
 environment variable, [U-74](#)
WM PROJECT VERSION
 environment variable, [U-74](#)
wmake
 platforms, [U-70](#)
wmake script/alias, [U-69](#)
word class, [P-21](#), [P-27](#)
writeCellCentres utility, [U-93](#)
writeCompression keyword, [U-112](#)
writeControl
 keyword entry, [U-111](#)
writeControl keyword, [U-22](#), [U-60](#), [U-111](#)
writeFormat keyword, [U-54](#), [U-112](#)
writeInterval keyword, [U-22](#), [U-31](#), [U-111](#)
writeMeshObj utility, [U-90](#)
writeNow
 keyword entry, [U-111](#)
writePrecision keyword, [U-112](#)

X

x
 keyword entry, [U-175](#)
XiFoam solver, [U-86](#)
xmgr
 keyword entry, [U-112](#), [U-174](#)
xyz
 keyword entry, [U-175](#)

Y

y
 keyword entry, [U-175](#)
yPlusLES utility, [U-92](#)
yPlusRAS utility, [U-92](#)

Z

z
 keyword entry, [U-175](#)
zeroGradient
 boundary condition, [U-136](#)
zipUpMesh utility, [U-91](#)