OpenFOAM

The Open Source CFD Toolbox

Programmer's Guide

Version 2.2.2
28th September 2013

Copyright © 2011-2013 OpenFOAM Foundation.

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

Typeset in LaTeX.

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;

b. and, to Distribute and Publicly Perform the Work including as incorporated in Collections.

The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats, but otherwise you have no rights

to make Adaptations. Subject to 8(f), all rights not expressly granted by Licensor are hereby reserved, including but not limited to the rights set forth in Section 4(d).

4. Restrictions.

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

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

b. You may not exercise any of the rights granted to You in Section 3 above in any manner that is primarily intended for or directed toward commercial advantage or private monetary compensation. The exchange of the Work for other copyrighted works by means of digital file-sharing or otherwise shall not be considered to be intended for or directed toward commercial advantage or private monetary compensation, provided there is no payment of any monetary compensation in connection with the exchange of copyrighted works.

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

d. For the avoidance of doubt:

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, MERCHANTIBILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability.

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

7. Termination

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

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

this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

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

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

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

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

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

Trademarks

ANSYS is a registered trademark of ANSYS Inc.
CFX is a registered trademark of Ansys Inc.
CHEMKIN is a registered trademark of Reaction Design Corporation
EnSight is a registered trademark of Computational Engineering International Ltd.
Fieldview is a registered trademark of Intelligent Light
Fluent is a registered trademark of Ansys Inc.
GAMBIT is a registered trademark of Ansys Inc.
Icem-CFD is a registered trademark of Ansys Inc.
I-DEAS is a registered trademark of Structural Dynamics Research Corporation
JAVA is a registered trademark of Sun Microsystems Inc.
Linux is a registered trademark of Linus Torvalds
OpenFOAM is a registered trademark of SGI Corp.
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

Contents

Copyright Notice **P-2**
1. Definitions . . . P-2
2. Fair Dealing Rights. . . . P-3
3. License Grant. . . . P-3
4. Restrictions. . . . P-4
5. Representations, Warranties and Disclaimer . . . P-5
6. Limitation on Liability. . . . P-5
7. Termination . . . P-5
8. Miscellaneous . . . P-6

Trademarks **P-7**

Contents **P-9**

1 Tensor mathematics **P-11**
1.1 Coordinate system . . . P-11
1.2 Tensors . . . P-11
1.2.1 Tensor notation . . . P-13
1.3 Algebraic tensor operations . . . P-13
1.3.1 The inner product . . . P-14
1.3.2 The double inner product of two tensors . . . P-15
1.3.3 The triple inner product of two third rank tensors . . . P-15
1.3.4 The outer product . . . P-15
1.3.5 The cross product of two vectors . . . P-15
1.3.6 Other general tensor operations . . . P-16
1.3.7 Geometric transformation and the identity tensor . . . P-16
1.3.8 Useful tensor identities . . . P-17
1.3.9 Operations exclusive to tensors of rank 2 . . . P-17
1.3.10 Operations exclusive to scalars . . . P-18
1.4 OpenFOAM tensor classes . . . P-19
1.4.1 Algebraic tensor operations in OpenFOAM . . . P-19
1.5 Dimensional units . . . P-21

2 Discretisation procedures **P-23**
2.1 Differential operators . . . P-23
2.1.1 Gradient . . . P-23
2.1.2 Divergence . . . P-24
2.1.3 Curl . . . P-24
2.1.4 Laplacian . . . P-24
2.1.5 Temporal derivative . . . P-24
2.2 Overview of discretisation . . . P-25

2.2.1 OpenFOAM lists and fields . . . P-25
2.3 Discretisation of the solution domain . . . P-25
2.3.1 Defining a mesh in OpenFOAM . . . P-27
2.3.2 Defining a geometricField in OpenFOAM . . . P-28
2.4 Equation discretisation . . . P-29
2.4.1 The Laplacian term . . . P-34
2.4.2 The convection term . . . P-34
2.4.3 First time derivative . . . P-35
2.4.4 Second time derivative . . . P-35
2.4.5 Divergence . . . P-35
2.4.6 Gradient . . . P-36
2.4.7 Grad-grad squared . . . P-37
2.4.8 Curl . . . P-37
2.4.9 Source terms . . . P-37
2.4.10 Other explicit discretisation schemes . . . P-37
2.5 Temporal discretisation . . . P-38
2.5.1 Treatment of temporal discretisation in OpenFOAM . . . P-39
2.6 Boundary Conditions . . . P-39
2.6.1 Physical boundary conditions . . . P-40

3 Examples of the use of OpenFOAM P-41
3.1 Flow around a cylinder . . . P-41
3.1.1 Problem specification . . . P-42
3.1.2 Note on potentialFoam . . . P-43
3.1.3 Mesh generation . . . P-43
3.1.4 Boundary conditions and initial fields . . . P-45
3.1.5 Running the case . . . P-46
3.2 Steady turbulent flow over a backward-facing step . . . P-49
3.2.1 Problem specification . . . P-49
3.2.2 Mesh generation . . . P-50
3.2.3 Boundary conditions and initial fields . . . P-53
3.2.4 Case control . . . P-53
3.2.5 Running the case and post-processing . . . P-54
3.3 Supersonic flow over a forward-facing step . . . P-54
3.3.1 Problem specification . . . P-55
3.3.2 Mesh generation . . . P-56
3.3.3 Running the case . . . P-57
3.3.4 Exercise . . . P-57
3.4 Decompression of a tank internally pressurised with water . . . P-58
3.4.1 Problem specification . . . P-58
3.4.2 Mesh Generation . . . P-59
3.4.3 Preparing the Run . . . P-61
3.4.4 Running the case . . . P-62
3.4.5 Improving the solution by refining the mesh . . . P-63
3.5 Magnetohydrodynamic flow of a liquid . . . P-63
3.5.1 Problem specification . . . P-64
3.5.2 Mesh generation . . . P-65
3.5.3 Running the case . . . P-66

Index P-69

Chapter 1

Tensor mathematics

This Chapter describes tensors and their algebraic operations and how they are represented in mathematical text in this book. It then explains how tensors and tensor algebra are programmed in OpenFOAM.

1.1 Coordinate system

OpenFOAM is primarily designed to solve problems in continuum mechanics, *i.e.* the branch of mechanics concerned with the stresses in solids, liquids and gases and the deformation or flow of these materials. OpenFOAM is therefore based in 3 dimensional space and time and deals with physical entities described by tensors. The coordinate system used by OpenFOAM is the right-handed rectangular Cartesian axes as shown in Figure 1.1. This system of axes is constructed by defining an origin O from which three lines are drawn at right angles to each other, termed the Ox, Oy, Oz axes. A right-handed set of axes is defined such that to an observer looking down the Oz axis (with O nearest them), the arc from a point on the Ox axis to a point on the Oy axis is in a clockwise sense.

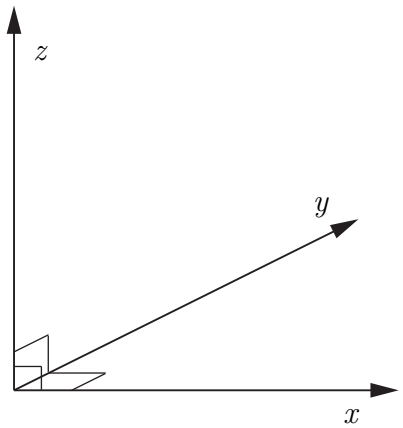


Figure 1.1: Right handed axes

1.2 Tensors

The term tensor describes an entity that belongs to a particular space and obeys certain mathematical rules. Briefly, tensors are represented by a set of *component values* relating to a set of unit base vectors; in OpenFOAM the unit base vectors $\mathbf{i}_x$, $\mathbf{i}_y$ and $\mathbf{i}_z$ are

aligned with the right-handed rectangular Cartesian axes x, y and z respectively. The base vectors are therefore orthogonal, *i.e.* at right-angles to one another. Every tensor has the following attributes:

Dimension d of the particular space to which they belong, *i.e.* $d = 3$ in OpenFOAM;

Rank An integer $r \geq 0$, such that the number of component values $= d^r$.

While OpenFOAM 1.x is set to 3 dimensions, it offers tensors of ranks 0 to 3 as standard while being written in such a way to allow this basic set of ranks to be extended indefinitely. Tensors of rank 0 and 1, better known as scalars and vectors, should be familiar to readers; tensors of rank 2 and 3 may not be so familiar. For completeness all ranks of tensor offered as standard in OpenFOAM 1.x are reviewed below.

Rank 0 'scalar' Any property which can be represented by a single real number, denoted by characters in italics, *e.g.* mass m, volume V, pressure p and viscosity μ.

Rank 1 'vector' An entity which can be represented physically by both magnitude and direction. In component form, the vector $\mathbf{a} = (a_1, a_2, a_3)$ relates to a set of Cartesian axes x, y, z respectively. The *index notation* presents the same vector as a_i, $i = 1, 2, 3$, although the list of indices $i = 1, 2, 3$ will be omitted in this book, as it is intuitive since we are always dealing with 3 dimensions.

Rank 2 'tensor' or second rank tensor, $\mathbf{T}$ has 9 components which can be expressed in array notation as:

$$\mathbf{T} = T_{ij} = \begin{pmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{pmatrix} \tag{1.1}$$

The components T_{ij} are now represented using 2 indices since $r = 2$ and the list of indices $i, j = 1, 2, 3$ is omitted as before. The components for which $i = j$ are referred to as the diagonal components, and those for which $i \neq j$ are referred to as the off-diagonal components. The *transpose* of $\mathbf{T}$ is produced by exchanging components across the diagonal such that

$$\mathbf{T}^{\mathrm{T}} = T_{ji} = \begin{pmatrix} T_{11} & T_{21} & T_{31} \\ T_{12} & T_{22} & T_{32} \\ T_{13} & T_{23} & T_{33} \end{pmatrix} \tag{1.2}$$

Note: a rank 2 tensor is often colloquially termed 'tensor' since the occurrence of higher order tensors is fairly rare.

Symmetric rank 2 The term 'symmetric' refers to components being symmetric about the diagonal, *i.e.* $T_{ij} = T_{ji}$. In this case, there are only 6 independent components since $T_{12} = T_{21}$, $T_{13} = T_{31}$ and $T_{23} = T_{32}$. OpenFOAM distinguishes between symmetric and non-symmetric tensors to save memory by storing 6 components rather than 9 if the tensor is symmetric. Most tensors encountered in continuum mechanics are symmetric.

Rank 3 has 27 components and is represented in index notation as P_{ijk} which is too long to represent in array notation as in Equation 1.1.

Symmetric rank 3 Symmetry of a rank 3 tensor is defined in OpenFOAM to mean that $P_{ijk} = P_{ikj} = P_{jik} = P_{jki} = P_{kij} = P_{kji}$ and therefore has 10 independent components. More specifically, it is formed by the outer product of 3 identical vectors, where the outer product operation is described in Section 1.3.4.

1.2.1 Tensor notation

This is a book on computational continuum mechanics that deals with problems involving complex PDEs in 3 spatial dimensions and in time. It is vital from the beginning to adopt a notation for the equations which is compact yet unambiguous. To make the equations easy to follow, we must use a notation that encapsulates the idea of a tensor as an entity in the own right, rather than a list of scalar components. Additionally, any tensor operation should be perceived as an operation on the entire tensor entity rather than a series of operations on its components.

Consequently, in this book the *tensor notation* is preferred in which any tensor of rank 1 and above, *i.e.* all tensors other than scalars, are represented by letters in bold face, *e.g.* $\mathbf{a}$. This actively promotes the concept of a tensor as a entity in its own right since it is denoted by a single symbol, and it is also extremely compact. The potential drawback is that the rank of a bold face symbol is not immediately apparent, although it is clearly not zero. However, in practice this presents no real problem since we are aware of the property each symbol represents and therefore intuitively know its rank, *e.g.* we know velocity $\mathbf{U}$ is a tensor of rank 1.

A further, more fundamental idea regarding the choice of notation is that the mathematical representation of a tensor should not change depending on our coordinate system, *i.e.* the vector $\mathbf{a}$is the same vector irrespective of where we view it from. The tensor notation supports this concept as it implies nothing about the coordinate system. However, other notations, *e.g.* a_i, expose the individual components of the tensor which naturally implies the choice of coordinate system. The unsatisfactory consequence of this is that the tensor is then represented by a set of values which are not unique — they depend on the coordinate system.

That said, the index notation, introduced in Section 1.2, is adopted from time to time in this book mainly to expand tensor operations into the constituent components. When using the index notation, we adopt the *summation convention* which states that whenever the same letter subscript occurs twice in a term, the that subscript is to be given all values, *i.e.* $1, 2, 3$, and the results added together, *e.g.*

$$a_i b_i = \sum_{i=1}^{3} a_i b_i = a_1 b_1 + a_2 b_2 + a_3 b_3 \tag{1.3}$$

In the remainder of the book the symbol $\sum$ is omitted since the repeated subscript indicates the summation.

1.3 Algebraic tensor operations

This section describes all the algebraic operations for tensors that are available in OpenFOAM. Let us first review the most simple tensor operations: addition, subtraction, and scalar multiplication and division. Addition and subtraction are both commutative and associative and are only valid between tensors of the same rank. The operations are performed by addition/subtraction of respective components of the tensors, *e.g.* the subtraction of two vectors $\mathbf{a}$ and $\mathbf{b}$ is

$$\mathbf{a} - \mathbf{b} = a_i - b_i = (a_1 - b_1, a_2 - b_2, a_3 - b_3) \tag{1.4}$$

Multiplication of any tensor $\mathbf{a}$ by a scalar s is also commutative and associative and is performed by multiplying all the tensor components by the scalar. For example,

$$s\mathbf{a} = sa_i = (sa_1, sa_2, sa_3) \tag{1.5}$$

Division between a tensor $\mathbf{a}$ and a scalar is only relevant when the scalar is the second argument of the operation, *i.e.*

$$\mathbf{a}/s = a_i/s = (a_1/s, a_2/s, a_3/s) \tag{1.6}$$

Following these operations are a set of more complex products between tensors of rank 1 and above, described in the following Sections.

1.3.1 The inner product

The inner product operates on any two tensors of rank r_1 and r_2 such that the rank of the result $r = r_1 + r_2 - 2$. Inner product operations with tensors up to rank 3 are described below:

- The inner product of two vectors $\mathbf{a}$ and $\mathbf{b}$ is commutative and produces a scalar $s = \mathbf{a} \bullet \mathbf{b}$ where

$$s = a_i b_i = a_1 b_1 + a_2 b_2 + a_3 b_3 \tag{1.7}$$

- The inner product of a tensor $\mathbf{T}$ and vector $\mathbf{a}$ produces a vector $\mathbf{b} = \mathbf{T} \bullet \mathbf{a}$, represented below as a column array for convenience

$$b_i = T_{ij} a_j = \begin{pmatrix} T_{11}a_1 + T_{12}a_2 + T_{13}a_3 \\ T_{21}a_1 + T_{22}a_2 + T_{23}a_3 \\ T_{31}a_1 + T_{32}a_2 + T_{33}a_3 \end{pmatrix} \tag{1.8}$$

  It is non-commutative if $\mathbf{T}$ is non-symmetric such that $\mathbf{b} = \mathbf{a} \bullet \mathbf{T} = \mathbf{T}^{\mathrm{T}} \bullet \mathbf{a}$ is

$$b_i = a_j T_{ji} = \begin{pmatrix} a_1 T_{11} + a_2 T_{21} + a_3 T_{31} \\ a_1 T_{12} + a_2 T_{22} + a_3 T_{32} \\ a_1 T_{13} + a_2 T_{23} + a_3 T_{33} \end{pmatrix} \tag{1.9}$$

- The inner product of two tensors $\mathbf{T}$ and $\mathbf{S}$ produces a tensor $\mathbf{P} = \mathbf{T} \bullet \mathbf{S}$ whose components are evaluated as:

$$P_{ij} = T_{ik} S_{kj} \tag{1.10}$$

  It is non-commutative such that $\mathbf{T} \bullet \mathbf{S} = \left(\mathbf{S}^{\mathrm{T}} \bullet \mathbf{T}^{\mathrm{T}}\right)^{\mathrm{T}}$

- The inner product of a vector $\mathbf{a}$ and third rank tensor $\mathbf{P}$ produces a second rank tensor $\mathbf{T} = \mathbf{a} \bullet \mathbf{P}$ whose components are

$$T_{ij} = a_k P_{kij} \tag{1.11}$$

  Again this is non-commutative so that $\mathbf{T} = \mathbf{P} \bullet \mathbf{a}$ is

$$T_{ij} = P_{ijk} a_k \tag{1.12}$$

- The inner product of a second rank tensor $\mathbf{T}$ and third rank tensor $\mathbf{P}$ produces a third rank tensor $\mathbf{Q} = \mathbf{T} \bullet \mathbf{P}$ whose components are

$$Q_{ijk} = T_{il} P_{ljk} \tag{1.13}$$

  Again this is non-commutative so that $\mathbf{Q} = \mathbf{P} \bullet \mathbf{T}$ is

$$Q_{ijk} = P_{ijl} T_{lk} \tag{1.14}$$

1.3.2 The double inner product of two tensors

The double inner product of two second-rank tensors $\mathbf{T}$ and $\mathbf{S}$ produces a scalar $s = \mathbf{T} : \mathbf{S}$ which can be evaluated as the sum of the 9 products of the tensor components

$$\begin{aligned} s = T_{ij}S_{ij} = \; & T_{11}S_{11} + T_{12}S_{12} + T_{13}S_{13} + \\ & T_{21}S_{21} + T_{22}S_{22} + T_{23}S_{23} + \\ & T_{31}S_{31} + T_{32}S_{32} + T_{33}S_{33} \end{aligned} \tag{1.15}$$

The double inner product between a second rank tensor $\mathbf{T}$ and third rank tensor $\mathbf{P}$ produces a vector $\mathbf{a} = \mathbf{T} : \mathbf{P}$ with components

$$a_i = T_{jk}P_{jki} \tag{1.16}$$

This is non-commutative so that $\mathbf{a} = \mathbf{P} : \mathbf{T}$ is

$$a_i = P_{ijk}T_{jk} \tag{1.17}$$

1.3.3 The triple inner product of two third rank tensors

The triple inner product of two third rank tensors $\mathbf{P}$ and $\mathbf{Q}$ produces a scalar $s = \mathbf{P} \vdots \mathbf{Q}$ which can be evaluated as the sum of the 27 products of the tensor components

$$s = P_{ijk}Q_{ijk} \tag{1.18}$$

1.3.4 The outer product

The outer product operates between vectors and tensors as follows:

- The outer product of two vectors $\mathbf{a}$ and $\mathbf{b}$ is non-commutative and produces a tensor $\mathbf{T} = \mathbf{ab} = (\mathbf{ba})^{\mathrm{T}}$ whose components are evaluated as:

$$T_{ij} = a_i b_j = \begin{pmatrix} a_1b_1 & a_1b_2 & a_1b_3 \\ a_2b_1 & a_2b_2 & a_2b_3 \\ a_3b_1 & a_3b_2 & a_3b_3 \end{pmatrix} \tag{1.19}$$

- An outer product of a vector $\mathbf{a}$ and second rank tensor $\mathbf{T}$ produces a third rank tensor $\mathbf{P} = \mathbf{aT}$ whose components are

$$P_{ijk} = a_i T_{jk} \tag{1.20}$$

This is non-commutative so that $\mathbf{P} = \mathbf{T}\,\mathbf{a}$ produces

$$P_{ijk} = T_{ij}a_k \tag{1.21}$$

1.3.5 The cross product of two vectors

The cross product operation is exclusive to vectors only. For two vectors $\mathbf{a}$ with $\mathbf{b}$, it produces a vector $\mathbf{c} = \mathbf{a} \times \mathbf{b}$ whose components are

$$c_i = e_{ijk}a_j b_k = (a_2b_3 - a_3b_2, a_3b_1 - a_1b_3, a_1b_2 - a_2b_1) \tag{1.22}$$

where the *permutation symbol* is defined by

$$e_{ijk} = \begin{cases} 0 & \text{when any two indices are equal} \\ +1 & \text{when } i,j,k \text{ are an even permutation of 1,2,3} \\ -1 & \text{when } i,j,k \text{ are an odd permutation of 1,2,3} \end{cases} \tag{1.23}$$

in which the even permutations are 123, 231 and 312 and the odd permutations are 132, 213 and 321.

1.3.6 Other general tensor operations

Some less common tensor operations and terminology used by OpenFOAM are described below.

Square of a tensor is defined as the outer product of the tensor with itself, *e.g.* for a vector $\mathbf{a}$, the square $\mathbf{a}^2 = \mathbf{aa}$.

***n*th power** of a tensor is evaluated by n outer products of the tensor, *e.g.* for a vector $\mathbf{a}$, the 3rd power $\mathbf{a}^3 = \mathbf{aaa}$.

Magnitude squared of a tensor is the rth inner product of the tensor of rank r with itself, to produce a scalar. For example, for a second rank tensor $\mathbf{T}$, $|\mathbf{T}|^2 = \mathbf{T} : \mathbf{T}$.

Magnitude is the square root of the magnitude squared, *e.g.* for a tensor $\mathbf{T}$, $|\mathbf{T}| = \sqrt{\mathbf{T} : \mathbf{T}}$. Vectors of unit magnitude are referred to as *unit vectors*.

Component maximum is the component of the tensor with greatest value, inclusive of sign, *i.e.* not the largest magnitude.

Component minimum is the component of the tensor with smallest value.

Component average is the mean of all components of a tensor.

Scale As the name suggests, the scale function is a tool for scaling the components of one tensor by the components of another tensor of the same rank. It is evaluated as the product of corresponding components of 2 tensors, *e.g.*, scaling vector $\mathbf{a}$ by vector $\mathbf{b}$ would produce vector $\mathbf{c}$ whose components are

$$c_i = \text{scale}(\mathbf{a}, \mathbf{b}) = (a_1 b_1, a_2 b_2, a_3 b_3) \tag{1.24}$$

1.3.7 Geometric transformation and the identity tensor

A second rank tensor $\mathbf{T}$ is strictly defined as a linear vector function, i.e. it is a function which associates an argument vector $\mathbf{a}$ to another vector $\mathbf{b}$ by the inner product $\mathbf{b} = \mathbf{T} \cdot \mathbf{a}$. The components of $\mathbf{T}$ can be chosen to perform a specific geometric transformation of a tensor from the x, y, z coordinate system to a new coordinate system x^*, y^*, z^*; $\mathbf{T}$ is then referred to as the *transformation tensor*. While a scalar remains unchanged under a transformation, the vector $\mathbf{a}$ is transformed to $\mathbf{a}^*$ by

$$\mathbf{a}^* = \mathbf{T} \cdot \mathbf{a} \tag{1.25}$$

A second rank tensor $\mathbf{S}$ is transformed to $\mathbf{S}^*$ according to

$$\mathbf{S}^* = \mathbf{T} \cdot \mathbf{S} \cdot \mathbf{T}^{\mathrm{T}} \tag{1.26}$$

The *identity tensor* $\mathbf{I}$ is defined by the requirement that it transforms another tensor onto itself. For all vectors $\mathbf{a}$

$$\mathbf{a} = \mathbf{I} \cdot \mathbf{a} \tag{1.27}$$

and therefore

$$\mathbf{I} = \delta_{ij} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{1.28}$$

where δ_{ij} is known as the *Kronecker delta* symbol.

1.3.8 Useful tensor identities

Several identities are listed below which can be verified by under the assumption that all the relevant derivatives exist and are continuous. The identities are expressed for scalar s and vector $\mathbf{a}$.

$$\begin{aligned}
&\nabla \bullet (\nabla \times \mathbf{a}) \equiv 0 \\
&\nabla \times (\nabla s) \equiv \mathbf{0} \\
&\nabla \bullet (s\mathbf{a}) \equiv s\nabla \bullet \mathbf{a} + \mathbf{a} \bullet \nabla s \\
&\nabla \times (s\mathbf{a}) \equiv s\nabla \times \mathbf{a} + \nabla s \times \mathbf{a} \\
&\nabla(\mathbf{a} \bullet \mathbf{b}) \equiv \mathbf{a} \times (\nabla \times \mathbf{b}) + \mathbf{b} \times (\nabla \times \mathbf{a}) + (\mathbf{a} \bullet \nabla)\mathbf{b} + (\mathbf{b} \bullet \nabla)\mathbf{a} \\
&\nabla \bullet (\mathbf{a} \times \mathbf{b}) \equiv \mathbf{b} \bullet (\nabla \times \mathbf{a}) - \mathbf{a} \bullet (\nabla \times \mathbf{b}) \\
&\nabla \times (\mathbf{a} \times \mathbf{b}) \equiv \mathbf{a}(\nabla \bullet \mathbf{b}) - \mathbf{b}(\nabla \bullet \mathbf{a}) + (\mathbf{b} \bullet \nabla)\mathbf{a} - (\mathbf{a} \bullet \nabla)\mathbf{b} \\
&\nabla \times (\nabla \times \mathbf{a}) \equiv \nabla(\nabla \bullet \mathbf{a}) - \nabla^2 \mathbf{a} \\
&(\nabla \times \mathbf{a}) \times \mathbf{a} \equiv \mathbf{a} \bullet (\nabla \mathbf{a}) - \nabla(\mathbf{a} \bullet \mathbf{a})
\end{aligned} \tag{1.29}$$

It is sometimes useful to know the $e - \delta$ identity to help to manipulate equations in index notation:

$$e_{ijk} e_{irs} = \delta_{jr}\delta_{ks} - \delta_{js}\delta_{kr} \tag{1.30}$$

1.3.9 Operations exclusive to tensors of rank 2

There are several operations that manipulate the components of tensors of rank 2 that are listed below:

Transpose of a tensor $\mathbf{T} = T_{ij}$ is $\mathbf{T}^{\mathrm{T}} = T_{ji}$ as described in Equation 1.2.

Symmetric and skew (antisymmetric) tensors As discussed in section 1.2, a tensor is said to be symmetric if its components are symmetric about the diagonal, i.e. $\mathbf{T} = \mathbf{T}^{\mathrm{T}}$. A skew or antisymmetric tensor has $\mathbf{T} = -\mathbf{T}^{\mathrm{T}}$ which intuitively implies that $T_{11} = T_{22} = T_{33} = 0$. Every second order tensor can be decomposed into symmetric and skew parts by

$$\mathbf{T} = \underbrace{\frac{1}{2}(\mathbf{T} + \mathbf{T}^{\mathrm{T}})}_{symmetric} + \underbrace{\frac{1}{2}(\mathbf{T} - \mathbf{T}^{\mathrm{T}})}_{skew} = \mathrm{symm}\,\mathbf{T} + \mathrm{skew}\,\mathbf{T} \tag{1.31}$$

Trace The trace of a tensor $\mathbf{T}$ is a scalar, evaluated by summing the diagonal components

$$\mathrm{tr}\,\mathbf{T} = T_{11} + T_{22} + T_{33} \tag{1.32}$$

Diagonal returns a vector whose components are the diagonal components of the second rank tensor $\mathbf{T}$

$$\mathrm{diag}\,\mathbf{T} = (T_{11}, T_{22}, T_{33}) \tag{1.33}$$

Deviatoric and hydrostatic tensors Every second rank tensor $\mathbf{T}$ can be decomposed into a deviatoric component, for which $\mathrm{tr}\,\mathbf{T} = 0$ and a hydrostatic component of the form $\mathbf{T} = s\mathbf{I}$ where s is a scalar. Every second rank tensor can be decomposed into deviatoric and hydrostatic parts as follows:

$$\mathbf{T} = \underbrace{\mathbf{T} - \frac{1}{3}(\mathrm{tr}\,\mathbf{T})\,\mathbf{I}}_{deviatoric} + \underbrace{\frac{1}{3}(\mathrm{tr}\,\mathbf{T})\,\mathbf{I}}_{hydrostatic} = \mathrm{dev}\,\mathbf{T} + \mathrm{hyd}\,\mathbf{T} \tag{1.34}$$

Determinant The determinant of a second rank tensor is evaluated by

$$\begin{aligned}\det \mathbf{T} = \begin{vmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{vmatrix} &= \begin{array}{l} T_{11}(T_{22}T_{33} - T_{23}T_{32}) - \\ T_{12}(T_{21}T_{33} - T_{23}T_{31}) + \\ T_{13}(T_{21}T_{32} - T_{22}T_{31}) \end{array} \\ &= \frac{1}{6} e_{ijk} e_{pqr} T_{ip} T_{jq} T_{kr}\end{aligned} \tag{1.35}$$

Cofactors The *minors* of a tensor are evaluated for each component by deleting the row and column in which the component is situated and evaluating the resulting entries as a 2×2 *determinant*. For example, the minor of T_{12} is

$$\begin{vmatrix} \cancel{T_{11}} & \cancel{T_{12}} & \cancel{T_{13}} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{vmatrix} = \begin{vmatrix} T_{21} & T_{23} \\ T_{31} & T_{33} \end{vmatrix} = T_{21}T_{33} - T_{23}T_{31} \tag{1.36}$$

The cofactors are *signed minors* where each minor is component is given a sign based on the rule

$$\begin{array}{l} +\text{ve if } i + j \text{ is even} \\ -\text{ve if } i + j \text{ is odd} \end{array} \tag{1.37}$$

The cofactors of $\mathbf{T}$ can be evaluated as

$$\operatorname{cof} \mathbf{T} = \frac{1}{2} e_{jkr} e_{ist} T_{sk} T_{tr} \tag{1.38}$$

Inverse The inverse of a tensor can be evaluated as

$$\operatorname{inv} \mathbf{T} = \frac{\operatorname{cof} \mathbf{T}^{\mathrm{T}}}{\det \mathbf{T}} \tag{1.39}$$

Hodge dual of a tensor is a vector whose components are

$$*\,\mathbf{T} = (T_{23}, -T_{13}, T_{12}) \tag{1.40}$$

1.3.10 Operations exclusive to scalars

OpenFOAM supports most of the well known functions that operate on scalars, *e.g.* square root, exponential, logarithm, sine, cosine *etc.*., a list of which can be found in Table 1.2. There are 3 additional functions defined within OpenFOAM that are described below:

Sign of a scalar s is

$$\operatorname{sgn}(s) = \begin{cases} 1 & \text{if } s \geq 0, \\ -1 & \text{if } s < 0. \end{cases} \tag{1.41}$$

Positive of a scalar s is

$$\operatorname{pos}(s) = \begin{cases} 1 & \text{if } s \geq 0, \\ 0 & \text{if } s < 0. \end{cases} \tag{1.42}$$

Limit of a scalar s by the scalar n

$$\operatorname{limit}(s, n) = \begin{cases} s & \text{if } s < n, \\ 0 & \text{if } s \geq n. \end{cases} \tag{1.43}$$

1.4 OpenFOAM tensor classes

OpenFOAM contains a C++ class library primitive that contains the classes for the tensor mathematics described so far. The basic tensor classes that are available as standard in OpenFOAM are listed in Table 1.1. The Table also lists the functions that allow the user to access individual components of a tensor, known as access functions.

Rank	Common name	Basic class	Access functions
0	Scalar	scalar	
1	Vector	vector	`x()`, `y()`, `z()`
2	Tensor	tensor	`xx()`, `xy()`, `xz()`...

Table 1.1: Basic tensor classes in OpenFOAM

We can declare the tensor

$$\mathbf{T} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \tag{1.44}$$

in OpenFOAM by the line:

```
tensor T(1, 2, 3, 4, 5, 6, 7, 8, 9);
```

We can then access the component T_{13}, or T_{xz} using the `xz()` access function. For instance the code

```
Info << ``Txz = '' << T.xz() << endl;
```

outputs to the screen:

```
Txz = 3
```

1.4.1 Algebraic tensor operations in OpenFOAM

The algebraic operations described in Section 1.3 are all available to the OpenFOAM tensor classes using syntax which closely mimics the notation used in written mathematics. Some functions are represented solely by descriptive functions, *e.g.*`symm()`, but others can also be executed using symbolic operators, *e.g.*`*`. All functions are listed in Table 1.2.

Operation	Comment	Mathematical Description	Description in OpenFOAM
Addition		$\mathbf{a}+\mathbf{b}$	`a + b`
Subtraction		$\mathbf{a}-\mathbf{b}$	`a - b`
Scalar multiplication		$s\mathbf{a}$	`s * a`
Scalar division		$\mathbf{a}/s$	`a / s`
Outer product	rank $\mathbf{a},\mathbf{b} >= 1$	$\mathbf{ab}$	`a * b`
Inner product	rank $\mathbf{a},\mathbf{b} >= 1$	$\mathbf{a}\cdot\mathbf{b}$	`a & b`
Double inner product	rank $\mathbf{a},\mathbf{b} >= 2$	$\mathbf{a}:\mathbf{b}$	`a && b`
Cross product	rank $\mathbf{a},\mathbf{b} = 1$	$\mathbf{a}\times\mathbf{b}$	`a ^ b`
Square		$\mathbf{a}^2$	`sqr(a)`

Continued on next page

Continued from previous page

Operation	Comment	Mathematical Description	Description in OpenFOAM
Magnitude squared		$\|\mathbf{a}\|^2$	`magSqr(a)`
Magnitude		$\|\mathbf{a}\|$	`mag(a)`
Power	$n = 0, 1, ..., 4$	$\mathbf{a}^n$	`pow(a,n)`
Component average	$i = 1, ..., N$	$\overline{a_i}$	`cmptAv(a)`
Component maximum	$i = 1, ..., N$	$\max(a_i)$	`max(a)`
Component minimum	$i = 1, ..., N$	$\min(a_i)$	`min(a)`
Scale		scale($\mathbf{a}$,$\mathbf{b}$)	`scale(a,b)`
Geometric transformation	transforms $\mathbf{a}$ using tensor $\mathbf{T}$		`transform(T,a)`

Operations exclusive to tensors of rank 2

Transpose		$\mathbf{T}^{\mathrm{T}}$	`T.T()`
Diagonal		diag $\mathbf{T}$	`diag(T)`
Trace		tr $\mathbf{T}$	`tr(T)`
Deviatoric component		dev $\mathbf{T}$	`dev(T)`
Symmetric component		symm $\mathbf{T}$	`symm(T)`
Skew-symmetric component		skew $\mathbf{T}$	`skew(T)`
Determinant		det $\mathbf{T}$	`det(T)`
Cofactors		cof $\mathbf{T}$	`cof(T)`
Inverse		inv $\mathbf{T}$	`inv(T)`
Hodge dual		$*\,\mathbf{T}$	`*T`

Operations exclusive to scalars

Sign (boolean)		sgn(s)	`sign(s)`
Positive (boolean)		$s >= 0$	`pos(s)`
Negative (boolean)		$s < 0$	`neg(s)`
Limit	n scalar	limit(s, n)	`limit(s,n)`
Square root		$\sqrt{s}$	`sqrt(s)`
Exponential		exp s	`exp(s)`
Natural logarithm		ln s	`log(s)`
Base 10 logarithm		$\log_{10} s$	`log10(s)`
Sine		sin s	`sin(s)`
Cosine		cos s	`cos(s)`
Tangent		tan s	`tan(s)`
Arc sine		asin s	`asin(s)`
Arc cosine		acos s	`acos(s)`
Arc tangent		atan s	`atan(s)`
Hyperbolic sine		sinh s	`sinh(s)`
Hyperbolic cosine		cosh s	`cosh(s)`
Hyperbolic tangent		tanh s	`tanh(s)`
Hyperbolic arc sine		asinh s	`asinh(s)`
Hyperbolic arc cosine		acosh s	`acosh(s)`
Hyperbolic arc tangent		atanh s	`atanh(s)`
Error function		erf s	`erf(s)`
Complement error function		erfc s	`erfc(s)`
Logarithm gamma function		ln Γs	`lgamma(s)`
Type 1 Bessel function of order 0		$\mathrm{J}_0\, s$	`j0(s)`
Type 1 Bessel function of order 1		$\mathrm{J}_1\, s$	`j1(s)`

Continued on next page

Continued from previous page

Operation	Comment	Mathematical Description	Description in OpenFOAM
Type 2 Bessel function of order 0		$Y_0\, s$	`y0(s)`
Type 2 Bessel function of order 1		$Y_1\, s$	`y1(s)`

a, **b** are tensors of arbitrary rank unless otherwise stated
s is a scalar, N is the number of tensor components

Table 1.2: Algebraic tensor operations in OpenFOAM

1.5 Dimensional units

In continuum mechanics, properties are represented in some chosen units, *e.g.* mass in kilograms (kg), volume in cubic metres (m^3), pressure in Pascals ($kg\,m\,s^{-2}$). Algebraic operations must be performed on these properties using consistent units of measurement; in particular, addition, subtraction and equality are only physically meaningful for properties of the same dimensional units. As a safeguard against implementing a meaningless operation, OpenFOAM encourages the user to attach dimensional units to any tensor and will then perform dimension checking of any tensor operation.

Units are defined using the dimensionSet class, *e.g.*

```
dimensionSet pressureDims(1, -1, -2, 0, 0, 0, 0);
```

No.	Property	Unit	Symbol
1	Mass	kilogram	k
2	Length	metre	m
3	Time	second	s
4	Temperature	Kelvin	K
5	Quantity	moles	mol
6	Current	ampere	A
7	Luminous intensity	candela	cd

Table 1.3: S.I. base units of measurement

where each of the values corresponds to the power of each of the S.I. base units of measurement listed in Table 1.3. The line of code declares `pressureDims` to be the dimensionSet for pressure $kg\,m\,s^{-2}$ since the first entry in the `pressureDims` array, 1, corresponds to k^1, the second entry, -1, corresponds to m^{-1} *etc.*. A tensor with units is defined using the dimensioned<Type> template class, the <Type> being scalar, vector, tensor, *etc.*. The dimensioned<Type> stores a variable name of class word,the value <Type> and a dimensionSet

```
dimensionedTensor sigma
    (
        "sigma",
        dimensionSet(1, -1, -2, 0, 0, 0, 0),
        tensor(1e6,0,0,0,1e6,0,0,0,1e6),
    );
```

creates a tensor with correct dimensions of pressure, or stress

$$\boldsymbol{\sigma} = \begin{pmatrix} 10^6 & 0 & 0 \\ 0 & 10^6 & 0 \\ 0 & 0 & 10^6 \end{pmatrix} \tag{1.45}$$

Chapter 2

Discretisation procedures

So far we have dealt with algebra of tensors at a point. The PDEs we wish to solve involve derivatives of tensors with respect to time and space. We therefore need to extend our description to a *tensor field*, *i.e.* a tensor that varies across time and spatial domains. In this Chapter we will first present a mathematical description of all the differential operators we may encounter. We will then show how a tensor field is constructed in OpenFOAM and how the derivatives of these fields are discretised into a set of algebraic equations.

2.1 Differential operators

Before defining the spatial derivatives we first introduce the nabla *vector operator* ∇, represented in index notation as ∂_i:

$$\nabla \equiv \ \partial_i \equiv \frac{\partial}{\partial x_i} \equiv \left(\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, \frac{\partial}{\partial x_3} \right) \tag{2.1}$$

The nabla operator is a useful notation that obeys the following rules:

- it operates on the tensors to its right and the conventional rules of a derivative of a product, *e.g.* $\partial_i ab = (\partial_i a)\, b + a\, (\partial_i b)$;
- otherwise the nabla operator behaves like any other vector in an algebraic operation.

2.1.1 Gradient

If a scalar field s is defined and continuously differentiable then the gradient of s, ∇s is a vector field

$$\nabla s = \partial_i s = \left(\frac{\partial s}{\partial x_1}, \frac{\partial s}{\partial x_2}, \frac{\partial s}{\partial x_3} \right) \tag{2.2}$$

The gradient can operate on any tensor field to produce a tensor field that is one rank higher. For example, the gradient of a vector field $\mathbf{a}$ is a second rank tensor field

$$\nabla \mathbf{a} = \partial_i a_j = \begin{pmatrix} \partial a_1/\partial x_1 & \partial a_2/\partial x_1 & \partial a_3/\partial x_1 \\ \partial a_1/\partial x_2 & \partial a_2/\partial x_2 & \partial a_3/\partial x_2 \\ \partial a_1/\partial x_3 & \partial a_2/\partial x_3 & \partial a_3/\partial x_3 \end{pmatrix} \tag{2.3}$$

2.1.2 Divergence

If a vector field $\mathbf{a}$ is defined and continuously differentiable then the divergence of $\mathbf{a}$ is a scalar field

$$\nabla \bullet \mathbf{a} = \partial_i a_i = \frac{\partial a_1}{\partial x_1} + \frac{\partial a_2}{\partial x_2} + \frac{\partial a_3}{\partial x_3} \tag{2.4}$$

The divergence can operate on any tensor field of rank 1 and above to produce a tensor that is one rank lower. For example the divergence of a second rank tensor field $\mathbf{T}$ is a vector field (expanding the vector as a column array for convenience)

$$\nabla \bullet \mathbf{T} = \partial_j T_{ji} = \begin{pmatrix} \partial T_{11}/\partial x_1 + \partial T_{21}/\partial x_2 + \partial T_{31}/\partial x_3 \\ \partial T_{12}/\partial x_1 + \partial T_{22}/\partial x_2 + \partial T_{32}/\partial x_3 \\ \partial T_{13}/\partial x_1 + \partial T_{23}/\partial x_2 + \partial T_{33}/\partial x_3 \end{pmatrix} \tag{2.5}$$

2.1.3 Curl

If a vector field $\mathbf{a}$ is defined and continuously differentiable then the curl of $\mathbf{a}$, $\nabla \times \mathbf{a}$ is a vector field

$$\nabla \times \mathbf{a} = e_{ijk}\partial_j a_k = \left(\frac{\partial a_3}{\partial x_2} - \frac{\partial a_2}{\partial x_3}, \frac{\partial a_1}{\partial x_3} - \frac{\partial a_3}{\partial x_1}, \frac{\partial a_2}{\partial x_1} - \frac{\partial a_1}{\partial x_2} \right) \tag{2.6}$$

The curl is related to the gradient by

$$\nabla \times \mathbf{a} = 2\,(*\,\text{skew}\,\nabla \mathbf{a}) \tag{2.7}$$

2.1.4 Laplacian

The Laplacian is an operation that can be defined mathematically by a combination of the divergence and gradient operators by $\nabla^2 \equiv \nabla \bullet \nabla$. However, the Laplacian should be considered as a single operation that transforms a tensor field into another tensor field of the same rank, rather than a combination of two operations, one which raises the rank by 1 and one which reduces the rank by 1.

In fact, the Laplacian is best defined as a *scalar operator*, just as we defined nabla as a vector operator, by

$$\nabla^2 \equiv \partial^2 \equiv \frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2} + \frac{\partial^2}{\partial x_3^2} \tag{2.8}$$

For example, the Laplacian of a scalar field s is the scalar field

$$\nabla^2 s = \partial^2 s = \frac{\partial^2 s}{\partial x_1^2} + \frac{\partial^2 s}{\partial x_2^2} + \frac{\partial^2 s}{\partial x_3^2} \tag{2.9}$$

2.1.5 Temporal derivative

There is more than one definition of temporal, or time, derivative of a tensor. To describe the temporal derivatives we must first recall that the tensor relates to a property of a volume of material that may be moving. If we track an infinitesimally small volume of material, or particle, as it moves and observe the change in the tensorial property ϕ in time, we have the *total*, or *material* time derivative denoted by

$$\frac{D\phi}{Dt} = \lim_{\Delta t \to 0} \frac{\Delta \phi}{\Delta t} \tag{2.10}$$

However in continuum mechanics, particularly fluid mechanics, we often observe the change of a ϕ in time at a fixed point in space as different particles move across that point. This change at a point in space is termed the *spatial* time derivative which is denoted by $\partial/\partial t$ and is related to the material derivative by:

$$\frac{D\phi}{Dt} = \frac{\partial \phi}{\partial t} + \mathbf{U} \bullet \nabla \phi \tag{2.11}$$

where $\mathbf{U}$ is the velocity field of property ϕ. The second term on the right is known as the convective rate of change of ϕ.

2.2 Overview of discretisation

The term discretisation means *approximation of a problem into discrete quantities*. The FV method and others, such as the finite element and finite difference methods, all discretise the problem as follows:

Spatial discretisation Defining the solution domain by a set of points that fill and bound a region of space when connected;

Temporal discretisation (For transient problems) dividing the time domain into into a finite number of time intervals, or steps;

Equation discretisation Generating a system of algebraic equations in terms of discrete quantities defined at specific locations in the domain, from the PDEs that characterise the problem.

2.2.1 OpenFOAM lists and fields

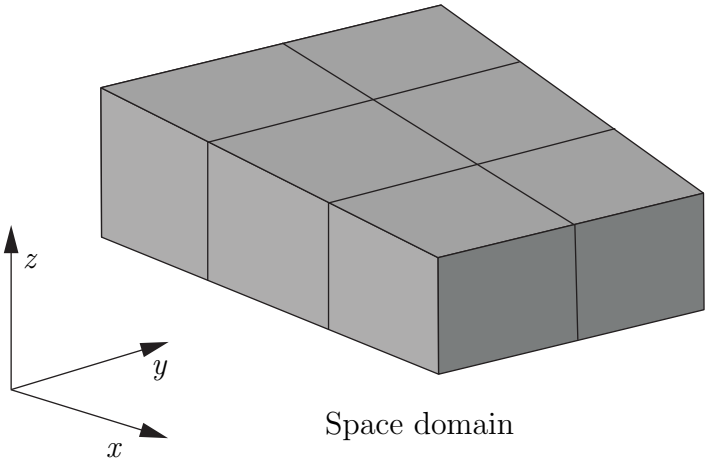
OpenFOAM frequently needs to store sets of data and perform functions, such as mathematical operations, on the data. OpenFOAM therefore provides an array template class List<Type>, making it possible to create a list of any object of class Type that inherits the functions of the Type. For example a List of vector is List<vector>.

Lists of the tensor classes are defined as standard in OpenFOAM by the template class Field<Type>. For better code legibility, all instances of Field<Type>, *e.g.*Field<vector>, are renamed using `typedef` declarations as scalarField, vectorField, tensorField, symmTensorField, tensorThirdField and symmTensorThirdField. Algebraic operations can be performed between Fields subject to obvious restrictions such as the fields having the same number of elements. OpenFOAM also supports operations between a field and single tensor, *e.g.* all values of a Field `U` can be multiplied by the scalar 2 with the operation `U = 2.0 * U`.

2.3 Discretisation of the solution domain

Discretisation of the solution domain is shown in Figure 2.1. The space domain is discretised into computational mesh on which the PDEs are subsequently discretised. Discretisation of time, if required, is simple: it is broken into a set of time steps Δt that may change during a numerical simulation, perhaps depending on some condition calculated during the simulation.

On a more detailed level, discretisation of space requires the subdivision of the domain into a number of cells, or control volumes. The cells are contiguous, *i.e.* they do not overlap one another and completely fill the domain. A typical cell is shown in Figure 2.2.



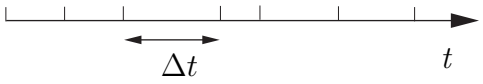


Figure 2.1: Discretisation of the solution domain

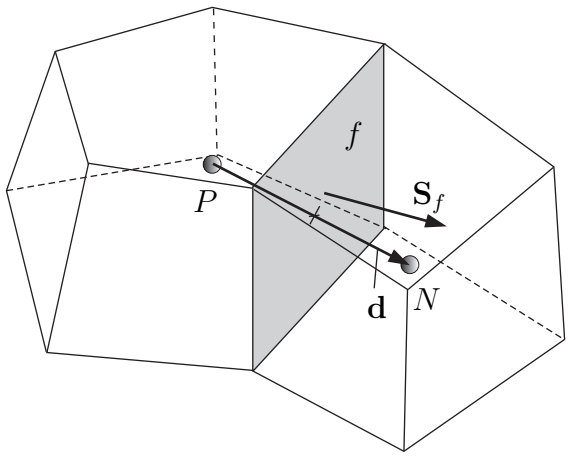


Figure 2.2: Parameters in finite volume discretisation

Dependent variables and other properties are principally stored at the cell centroid P although they may be stored on faces or vertices. The cell is bounded by a set of flat faces, given the generic label f. In OpenFOAM there is no limitation on the number of faces bounding each cell, nor any restriction on the alignment of each face. This kind of mesh is often referred to as "arbitrarily unstructured" to differentiate it from meshes in which the cell faces have a prescribed alignment, typically with the coordinate axes. Codes with arbitrarily unstructured meshes offer greater freedom in mesh generation and manipulation in particular when the geometry of the domain is complex or changes over time.

Whilst most properties are defined at the cell centroids, some are defined at cell faces. There are two types of cell face.

Internal faces Those faces that connect two cells (and it can never be more than two). For each internal face, OpenFOAM designates one adjoining cell to be the face *owner* and the other to be the *neighbour*;

Boundary faces Those belonging to one cell since they coincide with the boundary of the domain. These faces simply have an owner cell.

2.3.1 Defining a mesh in OpenFOAM

There are different levels of mesh description in OpenFOAM, beginning with the most basic mesh class, named polyMesh since it is based on polyhedra. A polyMesh is constructed using the minimum information required to define the mesh geometry described below and presented in Figure 2.3:

Points A list of cell vertex point coordinate vectors, *i.e.* a vectorField, that is renamed pointField using a `typedef` declaration;

Faces A list of cell faces List<face>, or faceList, where the face class is defined by a list of vertex numbers, corresponding to the pointField;

Cells a list of cells List<cell>, or cellList, where the cell class is defined by a list of face numbers, corresponding to the faceList described previously.

Boundary a polyBoundaryMesh decomposed into a list of patches, polyPatchList representing different regions of the boundary. The boundary is subdivided in this manner to allow different boundary conditions to be specified on different patches during a solution. All the faces of any polyPatch are stored as a single block of the faceList, so that its faces can be easily accessed using the slice class which stores references to the first and last face of the block. Each polyPatch is then constructed from

- a slice;
- a word to assign it a name.

FV discretisation uses specific data that is derived from the mesh geometry stored in polyMesh. OpenFOAM therefore extends the polyMesh class to fvMesh which stores the additional data needed for FV discretisation. fvMesh is constructed from polyMesh and stores the data in Table 2.1 which can be updated during runtime in cases where the mesh moves, is refined *etc.*.

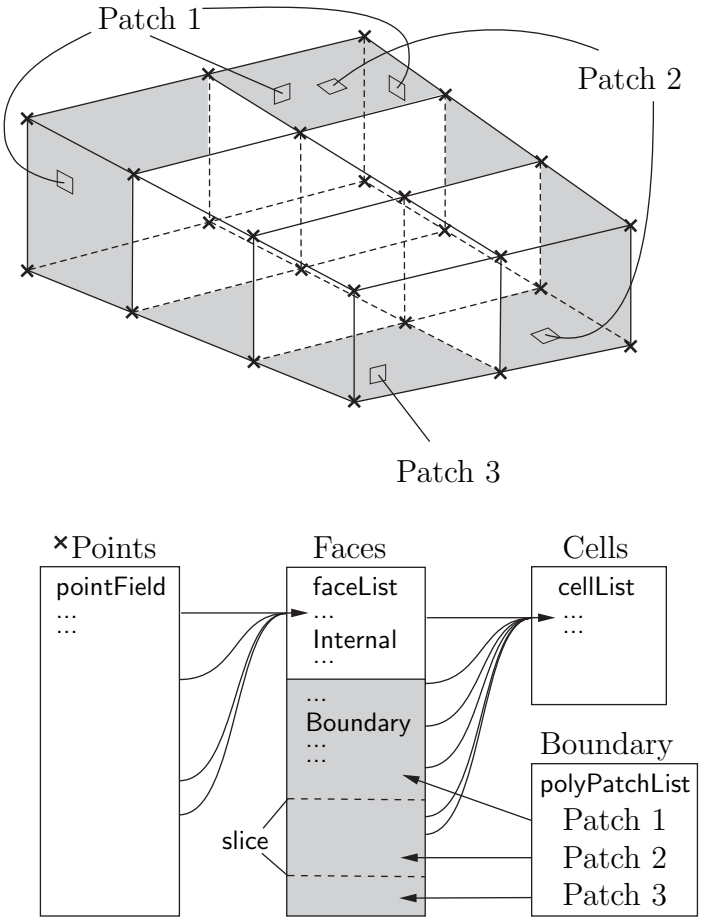


Figure 2.3: Schematic of the basic mesh description used in OpenFOAM

2.3.2 Defining a geometricField in OpenFOAM

So far we can define a field, *i.e.* a list of tensors, and a mesh. These can be combined to define a tensor field relating to discrete points in our domain, specified in OpenFOAM by the template class geometricField<Type>. The Field values are separated into those defined within the internal region of the domain, *e.g.* at the cell centres, and those defined on the domain boundary, *e.g.* on the boundary faces. The geometricField<Type> stores the following information:

Internal field This is simply a Field<Type>, described in Section 2.2.1;

BoundaryField This is a GeometricBoundaryField, in which a Field is defined for the faces of each patch and a Field is defined for the patches of the boundary. This is then a field of fields, stored within an object of the FieldField<Type> class. A reference to the fvBoundaryMesh is also stored [**].

Mesh A reference to an fvMesh, with some additional detail as to the whether the field is defined at cell centres, faces, *etc.*.

Dimensions A dimensionSet, described in Section 4.2.6.

Old values Discretisation of time derivatives requires field data from previous time steps. The geometricField<Type> will store references to stored fields from the previous, or old, time step and its previous, or old-old, time step where necessary.

Class	Description	Symbol	Access function
volScalarField	Cell volumes	V	`V()`
surfaceVectorField	Face area vectors	$\mathbf{S}_f$	`Sf()`
surfaceScalarField	Face area magnitudes	$\|\mathbf{S}_f\|$	`magSf()`
volVectorField	Cell centres	$\mathbf{C}$	`C()`
surfaceVectorField	Face centres	$\mathbf{C}_f$	`Cf()`
surfaceScalarField	Face motion fluxes **	ϕ_g	`phi()`

Table 2.1: fvMesh stored data.

Previous iteration values The iterative solution procedures can use under-relaxation which requires access to data from the previous iteration. Again, if required, geometricField<Type> stores a reference to the data from the previous iteration.

As discussed in Section 2.3, we principally define a property at the cell centres but quite often it is stored at the cell faces and on occasion it is defined on cell vertices. The geometricField<Type> is renamed using `typedef` declarations to indicate where the field variable is defined as follows:

volField<Type> A field defined at cell centres;

surfaceField<Type> A field defined on cell faces;

pointField<Type> A field defined on cell vertices.

These `typedef` field classes of geometricField<Type>are illustrated in Figure 2.4. A geometricField<Type> inherits all the tensor algebra of Field<Type> and has all operations subjected to dimension checking using the dimensionSet. It can also be subjected to the FV discretisation procedures described in the following Section. The class structure used to build geometricField<Type> is shown in Figure 2.5[1].

2.4 Equation discretisation

Equation discretisation converts the PDEs into a set of algebraic equations that are commonly expressed in matrix form as:

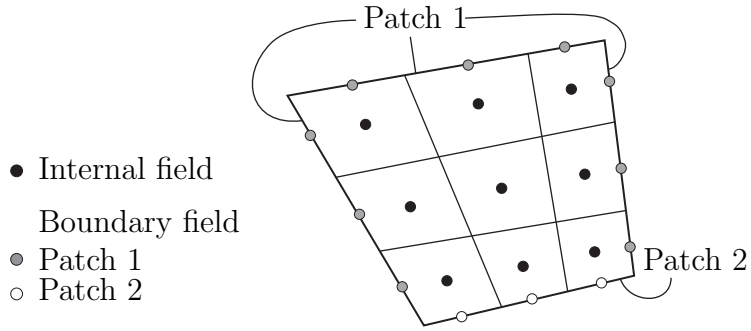
$$[A]\,[x] = [b] \tag{2.12}$$

where $[A]$ is a square matrix, $[x]$ is the column vector of dependent variable and $[b]$ is the source vector. The description of $[x]$ and $[b]$ as 'vectors' comes from matrix terminology rather than being a precise description of what they truly are: a list of values defined at locations in the geometry, *i.e.* a geometricField<Type>, or more specifically a volField<Type> when using FV discretisation.

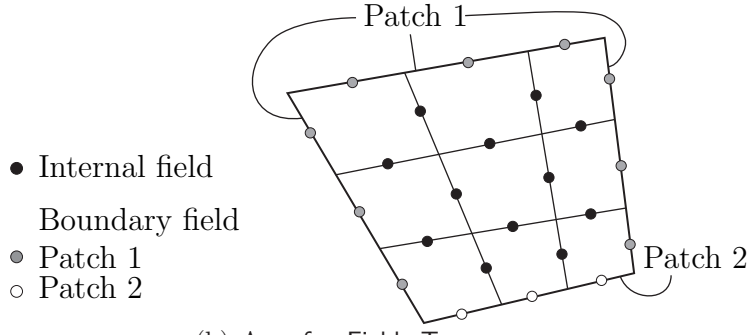
$[A]$ is a list of coefficients of a set of algebraic equations, and cannot be described as a geometricField<Type>. It is therefore given a class of its own: fvMatrix. fvMatrix<Type> is created through discretisation of a geometric<Type>Field and therefore inherits the <Type>. It supports many of the standard algebraic matrix operations of addition `+`, subtraction `-` and multiplication `*`.

Each term in a PDE is represented individually in OpenFOAM code using the classes of static functions finiteVolumeMethod and finiteVolumeCalculus, abbreviated by a `typedef`

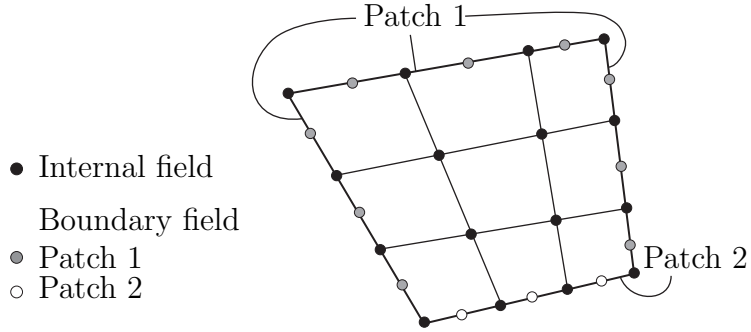
[1]The diagram is not an exact description of the class hierarchy, rather a representation of the general structure leading from some primitive classes to geometric<Type>Field.



(a) A volField<Type>



(b) A surfaceField<Type>



(c) A pointField<Type>

Figure 2.4: Types of geometricField<Type> defined on a mesh with 2 boundary patches (in 2 dimensions for simplicity)

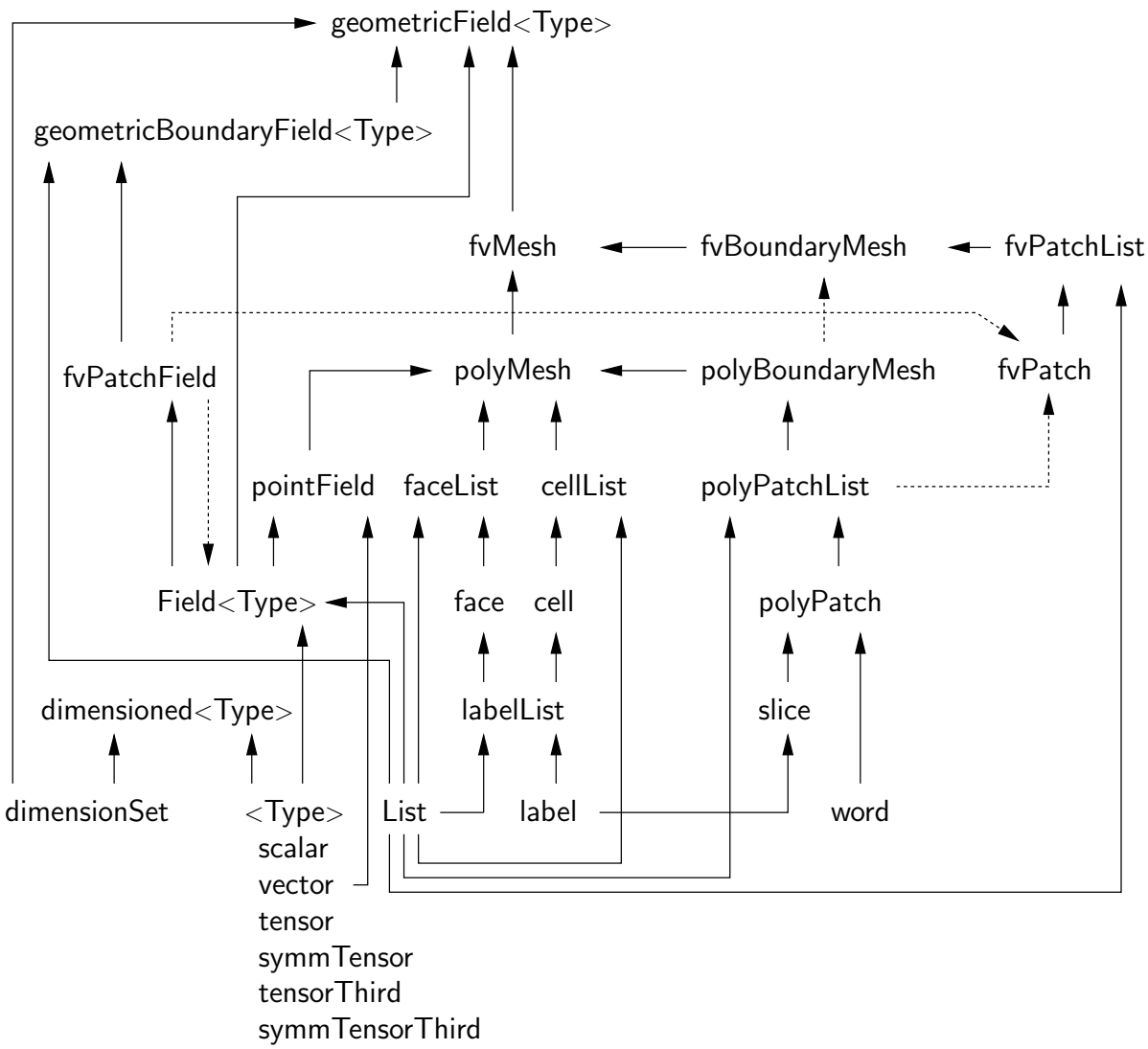


Figure 2.5: Basic class structure leading to geometricField<Type>

to fvm and fvc respectively. fvm and fvc contain static functions, representing differential operators, *e.g.* ∇^2, $\nabla\bullet$ and $\partial/\partial t$, that discretise geometricField<Type>s. The purpose of defining these functions within two classes, fvm and fvc, rather than one, is to distinguish:

- functions of fvm that calculate implicit derivatives of and return an fvMatrix<Type>
- some functions of fvc that calculate explicit derivatives and other explicit calculations, returning a geometricField<Type>.

Figure 2.6 shows a geometricField<Type> defined on a mesh with 2 boundary patches and illustrates the explicit operations merely transform one field to another and drawn in 2D for simplicity.

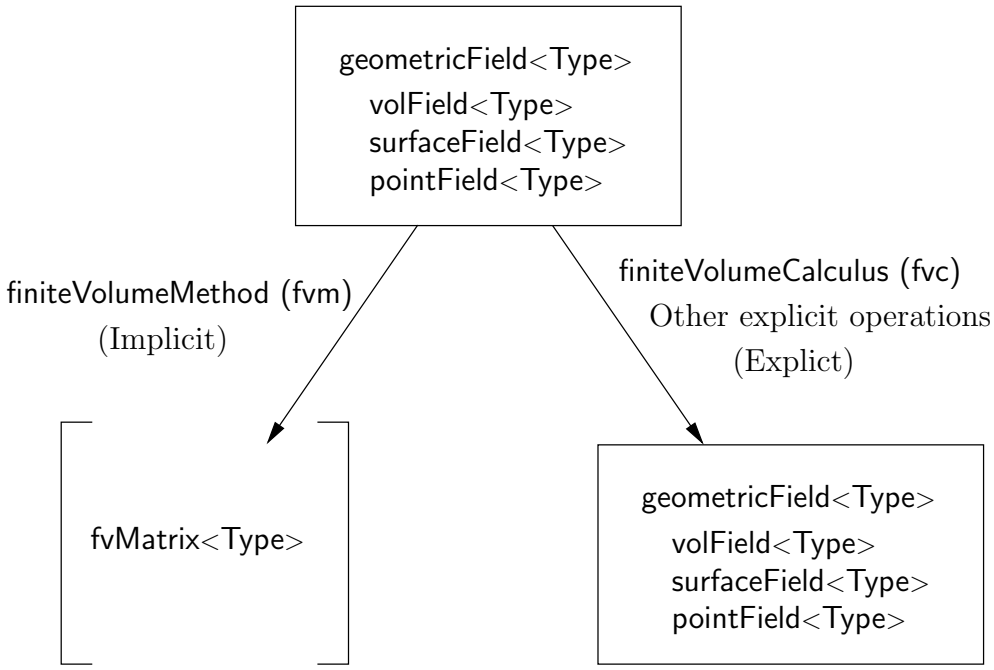


Figure 2.6: A geometricField<Type> and its operators

Table 2.2 lists the main functions that are available in fvm and fvc to discretise terms that may be found in a PDE. FV discretisation of each term is formulated by first integrating the term over a cell volume V. Most spatial derivative terms are then converted to integrals over the cell surface S bounding the volume using Gauss's theorem

$$\int_V \nabla \star \phi \; dV = \int_S d\mathbf{S} \star \phi \tag{2.13}$$

where $\mathbf{S}$ is the surface area vector, ϕ can represent any tensor field and the star notation $\star$ is used to represent any tensor product, *i.e.* inner, outer and cross and the respective derivatives: divergence $\nabla\bullet\phi$, gradient $\nabla\phi$ and $\nabla\times\phi$. Volume and surface integrals are then linearised using appropriate schemes which are described for each term in the following Sections. Some terms are always discretised using one scheme, a selection of schemes is offered in OpenFOAM for the discretisation of other terms. The choice of scheme is either made by a direct specification within the code or it can be read from an input file at job run-time and stored within an fvSchemes class object.

Term description	Implicit / Explicit	Text expression	fvm::/fvc:: functions
Laplacian	Imp/Exp	$\nabla^2\phi$	`laplacian(phi)`
		$\nabla \bullet \Gamma\nabla\phi$	`laplacian(Gamma, phi)`
Time derivative	Imp/Exp	$\frac{\partial\phi}{\partial t}$	`ddt(phi)`
		$\frac{\partial\rho\phi}{\partial t}$	`ddt(rho,phi)`
Second time derivative	Imp/Exp	$\frac{\partial}{\partial t}\left(\rho\frac{\partial\phi}{\partial t}\right)$	`d2dt2(rho, phi)`
Convection	Imp/Exp	$\nabla \bullet (\psi)$	`div(psi,scheme)`*
		$\nabla \bullet (\psi\phi)$	`div(psi, phi, word)`*
			`div(psi, phi)`
Divergence	Exp	$\nabla \bullet \chi$	`div(chi)`
Gradient	Exp	$\nabla\chi$	`grad(chi)`
		$\nabla\phi$	`gGrad(phi)`
			`lsGrad(phi)`
			`snGrad(phi)`
			`snGradCorrection(phi)`
Grad-grad squared	Exp	$\lvert\nabla\nabla\phi\rvert^2$	`sqrGradGrad(phi)`
Curl	Exp	$\nabla \times \phi$	`curl(phi)`
Source	Imp	$\rho\phi$	`Sp(rho,phi)`
	Imp/Exp†		`SuSp(rho,phi)`

†`fvm::SuSp` source is discretised implicit or explicit depending on the sign of `rho`.
†An explicit source can be introduced simply as a vol<Type>Field, *e.g.*`rho*phi`.
Function arguments can be of the following classes:
`phi`: vol<Type>Field
`Gamma`: scalar volScalarField, surfaceScalarField, volTensorField, surfaceTensorField.
`rho`: scalar, volScalarField
`psi`: surfaceScalarField.
`chi`: surface<Type>Field, vol<Type>Field.

Table 2.2: Discretisation of PDE terms in OpenFOAM

2.4.1 The Laplacian term

The Laplacian term is integrated over a control volume and linearised as follows:

$$\int_V \nabla \bullet (\Gamma \nabla \phi) \, dV = \int_S d\mathbf{S} \bullet (\Gamma \nabla \phi) = \sum_f \Gamma_f \mathbf{S}_f \bullet (\nabla \phi)_f \tag{2.14}$$

The face gradient discretisation is implicit when the length vector $\mathbf{d}$ between the centre of the cell of interest P and the centre of a neighbouring cell N is orthogonal to the face plane, *i.e.* parallel to $\mathbf{S}_f$:

$$\mathbf{S}_f \bullet (\nabla \phi)_f = |S_f| \frac{\phi_N - \phi_P}{|\mathbf{d}|} \tag{2.15}$$

In the case of non-orthogonal meshes, an additional explicit term is introduced which is evaluated by interpolating cell centre gradients, themselves calculated by central differencing cell centre values.

2.4.2 The convection term

The convection term is integrated over a control volume and linearised as follows:

$$\int_V \nabla \bullet (\rho \mathbf{U} \phi) \, dV = \int_S d\mathbf{S} \bullet (\rho \mathbf{U} \phi) = \sum_f \mathbf{S}_f \bullet (\rho \mathbf{U})_f \phi_f = \sum_f F \phi_f \tag{2.16}$$

The face field ϕ_f can be evaluated using a variety of schemes:

Central differencing (CD) is second-order accurate but unbounded

$$\phi_f = f_x \phi_P + (1 - f_x) \phi_N \tag{2.17}$$

where $f_x \equiv \overline{fN}/\overline{PN}$ where $\overline{fN}$ is the distance between f and cell centre N and $\overline{PN}$ is the distance between cell centres P and N.

Upwind differencing (UD) determines ϕ_f from the direction of flow and is bounded at the expense of accuracy

$$\phi_f = \begin{cases} \phi_P & \text{for } F \geq 0 \\ \phi_N & \text{for } F < 0 \end{cases} \tag{2.18}$$

Blended differencing (BD) schemes combine UD and CD in an attempt to preserve boundedness with reasonable accuracy,

$$\phi_f = (1 - \gamma) (\phi_f)_{UD} + \gamma (\phi_f)_{CD} \tag{2.19}$$

OpenFOAM has several implementations of the Gamma differencing scheme to select the blending coefficient γ but it offers other well-known schemes such as van Leer, SUPERBEE, MINMOD *etc.*.

2.4.3 First time derivative

The first time derivative $\partial/\partial t$ is integrated over a control volume as follows:

$$\frac{\partial}{\partial t}\int_V \rho\phi \; dV \tag{2.20}$$

The term is discretised by simple differencing in time using:

new values $\phi^n \equiv \phi(t + \Delta t)$ at the time step we are solving for;

old values $\phi^o \equiv \phi(t)$ that were stored from the previous time step;

old-old values $\phi^{oo} \equiv \phi(t - \Delta t)$ stored from a time step previous to the last.

One of two discretisation schemes can be declared using the `timeScheme` keyword in the appropriate input file, described in detail in section 4.4 of the User Guide.

Euler implicit scheme, `timeScheme EulerImplicit`, that is first order accurate in time:

$$\frac{\partial}{\partial t}\int_V \rho\phi \; dV = \frac{(\rho_P\phi_P V)^n - (\rho_P\phi_P V)^o}{\Delta t} \tag{2.21}$$

Backward differencing scheme, `timeScheme BackwardDifferencing`, that is second order accurate in time by storing the old-old values and therefore with a larger overhead in data storage than `EulerImplicit`:

$$\frac{\partial}{\partial t}\int_V \rho\phi \; dV = \frac{3\,(\rho_P\phi_P V)^n - 4\,(\rho_P\phi_P V)^o + (\rho_P\phi_P V)^{oo}}{2\Delta t} \tag{2.22}$$

2.4.4 Second time derivative

The second time derivative is integrated over a control volume and linearised as follows:

$$\frac{\partial}{\partial t}\int_V \rho\frac{\partial\phi}{\partial t} \; dV = \frac{(\rho_P\phi_P V)^n - 2\,(\rho_P\phi_P V)^o + (\rho_P\phi_P V)^{oo}}{\Delta t^2} \tag{2.23}$$

It is first order accurate in time.

2.4.5 Divergence

The divergence term described in this Section is strictly an explicit term that is distinguished from the convection term of Section 2.4.2, *i.e.* in that it is not the divergence of the product of a velocity and dependent variable. The term is integrated over a control volume and linearised as follows:

$$\int_V \nabla \bullet \phi \; dV = \int_S d\mathbf{S} \bullet \phi = \sum_f \mathbf{S}_f \bullet \phi_f \tag{2.24}$$

The `fvc::div` function can take as its argument either a surface<Type>Field, in which case ϕ_f is specified directly, or a vol<Type>Field which is interpolated to the face by central differencing as described in Section 2.4.10:

2.4.6 Gradient

The gradient term is an explicit term that can be evaluated in a variety of ways. The scheme can be evaluated either by selecting the particular grad function relevant to the discretisation scheme, *e.g.*`fvc::gGrad`, `fvc::lsGrad` *etc.*, or by using the `fvc::grad` function combined with the appropriate `gradScheme` keyword in an input file

Gauss integration is invoked using the `fvc::grad` function with `gradScheme Gauss` or directly using the `fvc::gGrad` function. The discretisation is performed using the standard method of applying Gauss's theorem to the volume integral:

$$\int_V \nabla\phi \; dV = \int_S d\mathbf{S}\, \phi = \sum_f \mathbf{S}_f \phi_f \tag{2.25}$$

As with the `fvc::div` function, the Gaussian integration `fvc::grad` function can take either a surfaceField<Type> or a volField<Type> as an argument.

Least squares method is based on the following idea:

1. a value at point P can be extrapolated to neighbouring point N using the gradient at P;
2. the extrapolated value at N can be compared to the actual value at N, the difference being the error;
3. if we now minimise the sum of the square of weighted errors at all neighbours of P with the respect to the gradient, then the gradient should be a good approximation.

Least squares is invoked using the `fvc::grad` function with `timeScheme leastSquares` or directly using the `fvc::lsGrad` function. The discretisation is performed as by first calculating the tensor $\mathbf{G}$ at every point P by summing over neighbours N:

$$\mathbf{G} = \sum_N w_N^2 \mathbf{d}\mathbf{d} \tag{2.26}$$

where $\mathbf{d}$ is the vector from P to N and the weighting function $w_N = 1/|\mathbf{d}|$. The gradient is then evaluated as:

$$(\nabla\phi)_P = \sum_N w_N^2 \mathbf{G}^{-1} \bullet \mathbf{d}\,(\phi_N - \phi_P) \tag{2.27}$$

Surface normal gradient The gradient normal to a surface $\mathbf{n}_f \bullet (\nabla\phi)_f$ can be evaluated at cell faces using the scheme

$$(\nabla\phi)_f = \frac{\phi_N - \phi_P}{|\mathbf{d}|} \tag{2.28}$$

This gradient is called by the function `fvc::snGrad` and returns a surfaceField<Type>. The scheme is directly analogous to that evaluated for the Laplacian discretisation scheme in Section 2.4.1, and in the same manner, a correction can be introduced to improve the accuracy of this face gradient in the case of non-orthogonal meshes. This correction is called using the function `fvc::snGradCorrection`.

2.4.7 Grad-grad squared

The grad-grad squared term is evaluated by: taking the gradient of the field; taking the gradient of the resulting gradient field; and then calculating the magnitude squared of the result. The mathematical expression for grad-grad squared of ϕ is $|\nabla(\nabla\phi)|^2$.

2.4.8 Curl

The curl is evaluated from the gradient term described in Section 2.4.6. First, the gradient is discretised and then the curl is evaluated using the relationship from Equation 2.7, repeated here for convenience

$$\nabla \times \phi = 2 \, *(\text{skew}\, \nabla\phi)$$

2.4.9 Source terms

Source terms can be specified in 3 ways

Explicit Every explicit term is a volField<Type>. Hence, an explicit source term can be incorporated into an equation simply as a field of values. For example if we wished to solve Poisson's equation $\nabla^2\phi = f$, we would define `phi` and `f` as volScalarField and then do

```
solve(fvm::laplacian(phi) == f)
```

Implicit An implicit source term is integrated over a control volume and linearised by

$$\int_V \rho\phi \, dV = \rho_P V_P \phi_P \tag{2.29}$$

Implicit/Explicit The implicit source term changes the coefficient of the diagonal of the matrix. Depending on the sign of the coefficient and matrix terms, this will either increase or decrease diagonal dominance of the matrix. Decreasing the diagonal dominance could cause instability during iterative solution of the matrix equation. Therefore OpenFOAM provides a mixed source discretisation procedure that is implicit when the coefficients that are greater than zero, and explicit for the coefficients less than zero. In mathematical terms the matrix coefficient for node P is $V_P \max(\rho_P, 0)$ and the source term is $V_P\phi_P \min(\rho_P, 0)$.

2.4.10 Other explicit discretisation schemes

There are some other discretisation procedures that convert volField<Type>s into surface<Type>Fields and visa versa.

Surface integral `fvc::surfaceIntegrate` performs a summation of surface<Type>Field face values bounding each cell and dividing by the cell volume, *i.e.* $(\sum_f \phi_f)/V_P$. It returns a volField<Type>.

Surface sum `fvc::surfaceSum` performs a summation of surface<Type>Field face values bounding each cell, *i.e.* $\sum_f \phi_f$ returning a volField<Type>.

Average `fvc::average` produces an area weighted average of surface<Type>Field face values, *i.e.* $(\sum_f S_f\phi_f)/\sum_f S_f$, and returns a volField<Type>.

Reconstruct

Face interpolate The geometric<Type>Field function `faceInterpolate()` interpolates volField<Type> cell centre values to cell faces using central differencing, returning a surface<Type>Field.

2.5 Temporal discretisation

Although we have described the discretisation of temporal derivatives in Sections 2.4.3 and 2.4.4, we need to consider how to treat the spatial derivatives in a transient problem. If we denote all the spatial terms as $\mathcal{A}\phi$ where $\mathcal{A}$ is any spatial operator, *e.g.* Laplacian, then we can express a transient PDE in integral form as

$$\int_t^{t+\Delta t} \left[\frac{\partial}{\partial t} \int_V \rho\phi \; dV + \int_V \mathcal{A}\phi \; dV \right] \; dt = 0 \tag{2.30}$$

Using the Euler implicit method of Equation 2.21, the first term can be expressed as

$$\begin{aligned} \int_t^{t+\Delta t} \left[\frac{\partial}{\partial t} \int_V \rho\phi \; dV \right] \; dt &= \int_t^{t+\Delta t} \frac{(\rho_P \phi_P V)^n - (\rho_P \phi_P V)^o}{\Delta t} \; dt \\ &= \frac{(\rho_P \phi_P V)^n - (\rho_P \phi_P V)^o}{\Delta t} \Delta t \end{aligned} \tag{2.31}$$

The second term can be expressed as

$$\int_t^{t+\Delta t} \left[\int_V \mathcal{A}\phi \; dV \right] \; dt = \int_t^{t+\Delta t} \mathcal{A}^*\phi \; dt \tag{2.32}$$

where $\mathcal{A}^*$ represents the spatial discretisation of $\mathcal{A}$. The time integral can be discretised in three ways:

Euler implicit uses implicit discretisation of the spatial terms, thereby taking current values ϕ^n.

$$\int_t^{t+\Delta t} \mathcal{A}^*\phi \; dt = \mathcal{A}^*\phi^n \Delta t \tag{2.33}$$

It is first order accurate in time, guarantees boundedness and is unconditionally stable.

Explicit uses explicit discretisation of the spatial terms, thereby taking old values ϕ^o.

$$\int_t^{t+\Delta t} \mathcal{A}^*\phi \; dt = \mathcal{A}^*\phi^o \Delta t \tag{2.34}$$

It is first order accurate in time and is unstable if the Courant number Co is greater than 1. The Courant number is defined as

$$Co = \frac{\mathbf{U}_f \bullet \mathbf{d}}{|\mathbf{d}|^2 \Delta t} \tag{2.35}$$

where $\mathbf{U}_f$ is a characteristic velocity, *e.g.* velocity of a wave front, velocity of flow.

Crank Nicholson uses the trapezoid rule to discretise the spatial terms, thereby taking a mean of current values ϕ^n and old values ϕ^o.

$$\int_t^{t+\Delta t} \mathcal{A}^*\phi \; dt = \mathcal{A}^* \left(\frac{\phi^n + \phi^o}{2} \right) \Delta t \tag{2.36}$$

It is second order accurate in time, is unconditionally stable but does not guarantee boundedness.

2.5.1 Treatment of temporal discretisation in OpenFOAM

At present the treatment of the temporal discretisation is controlled by the implementation of the spatial derivatives in the PDE we wish to solve. For example, let us say we wish to solve a transient diffusion equation

$$\frac{\partial \phi}{\partial t} = \kappa \nabla^2 \phi \tag{2.37}$$

An Euler implicit implementation of this would read

```
solve(fvm::ddt(phi) == kappa*fvm::laplacian(phi))
```

where we use the fvm class to discretise the `Laplacian` term implicitly. An explicit implementation would read

```
solve(fvm::ddt(phi) == kappa*fvc::laplacian(phi))
```

where we now use the fvc class to discretise the `Laplacian` term explicitly. The Crank Nicholson scheme can be implemented by the mean of implicit and explicit terms:

```
solve
    (
    fvm::ddt(phi)
    ==
    kappa*0.5*(fvm::laplacian(phi) + fvc::laplacian(phi))
    )
```

2.6 Boundary Conditions

Boundary conditions are required to complete the problem we wish to solve. We therefore need to specify boundary conditions on all our boundary faces. Boundary conditions can be divided into 2 types:

Dirichlet prescribes the value of the dependent variable on the boundary and is therefore termed 'fixed value' in this guide;

Neumann prescribes the gradient of the variable normal to the boundary and is therefore termed 'fixed gradient' in this guide.

When we perform discretisation of terms that include the sum over faces $\sum_f$, we need to consider what happens when one of the faces is a boundary face.

Fixed value We specify a fixed value at the boundary ϕ_b

- We can simply substitute ϕ_b in cases where the discretisation requires the value on a boundary face ϕ_f, *e.g.* in the convection term in Equation 2.16.
- In terms where the face gradient $(\nabla \phi)_f$ is required, *e.g.* Laplacian, it is calculated using the boundary face value and cell centre value,

$$\mathbf{S}_f \bullet (\nabla \phi)_f = |S_f| \frac{\phi_b - \phi_P}{|\mathbf{d}|} \tag{2.38}$$

Fixed gradient The fixed gradient boundary condition g_b is a specification on inner product of the gradient and unit normal to the boundary, or

$$g_b = \left(\frac{\mathbf{S}}{|\mathbf{S}|} \bullet \nabla\phi \right)_f \tag{2.39}$$

- When discretisation requires the value on a boundary face ϕ_f we must interpolate the cell centre value to the boundary by

$$\begin{aligned} \phi_f &= \phi_P + \mathbf{d} \bullet (\nabla\phi)_f \\ &= \phi_P + |\mathbf{d}|\, g_b \end{aligned} \tag{2.40}$$

- ϕ_b can be directly substituted in cases where the discretisation requires the face gradient to be evaluated,

$$\mathbf{S}_f \bullet (\nabla\phi)_f = |S_f|\, g_b \tag{2.41}$$

2.6.1 Physical boundary conditions

The specification of boundary conditions is usually an engineer's interpretation of the true behaviour. Real boundary conditions are generally defined by some physical attributes rather than the numerical description as described of the previous Section. In incompressible fluid flow there are the following physical boundaries

Inlet The velocity field at the inlet is supplied and, for consistency, the boundary condition on pressure is zero gradient.

Outlet The pressure field at the outlet is supplied and a zero gradient boundary condition on velocity is specified.

No-slip impermeable wall The velocity of the fluid is equal to that of the wall itself, *i.e.* a fixed value condition can be specified. The pressure is specified zero gradient since the flux through the wall is zero.

In a problem whose solution domain and boundary conditions are symmetric about a plane, we only need to model half the domain to one side of the symmetry plane. The boundary condition on the plane must be specified according to

Symmetry plane The symmetry plane condition specifies the component of the gradient normal to the plane should be zero.

Chapter 3

Examples of the use of OpenFOAM

In this section we shall describe several test cases supplied with the OpenFOAM distribution. The intention is to provide example cases, including those in the tutorials in chapter 2 of the User Guide, for every standard solver. The examples are designed to introduce certain tools and features of OpenFOAM, *e.g.* within pre-/post-processing, numerical schemes, algorithms. They also provide a means for validation of solvers although that is not their principal function.

Each example contains a description of the problem: the geometry, initial and boundary conditions, a brief description of the equations being solved, models used, and physical properties required. The solution domain is selected which may be a portion of the original geometry, *e.g.* if we introduce symmetry planes. The method of meshing, usually blockMesh, is specified; of course the user can simply view the mesh since every example is distributed with the *polyMesh* directory containing the data files that describe the mesh.

The examples coexist with the tutorials in the *tutorials* subdirectory of the OpenFOAM installation. They are organised into a set of subdirectories by solver, *e.g.* all the icoFoam cases are stored within a subdirectory *icoFoam*. Before running a particular example, the user is urged to copy it into their user account. We recommend that the user stores all OpenFOAM cases in a directory we recommend that the tutorials are copied into a directory *$FOAM_RUN*. If this directory structure has not yet been created in the user's account, it can be created with

```
mkdir -p $FOAM_RUN
```

The tutorials can then be copied into this directory with

```
cp -r $FOAM_TUTORIALS/* $FOAM_RUN
```

3.1 Flow around a cylinder

In this example we shall investigate potential flow around a cylinder using potentialFoam. This example introduces the following OpenFOAM features:

- non-orthogonal meshes;
- generating an analytical solution to a problem in OpenFOAM.

3.1.1 Problem specification

The problem is defined as follows:

Solution domain The domain is 2 dimensional and consists of a square domain with a cylinder collocated with the centre of the square as shown in Figure 3.1.

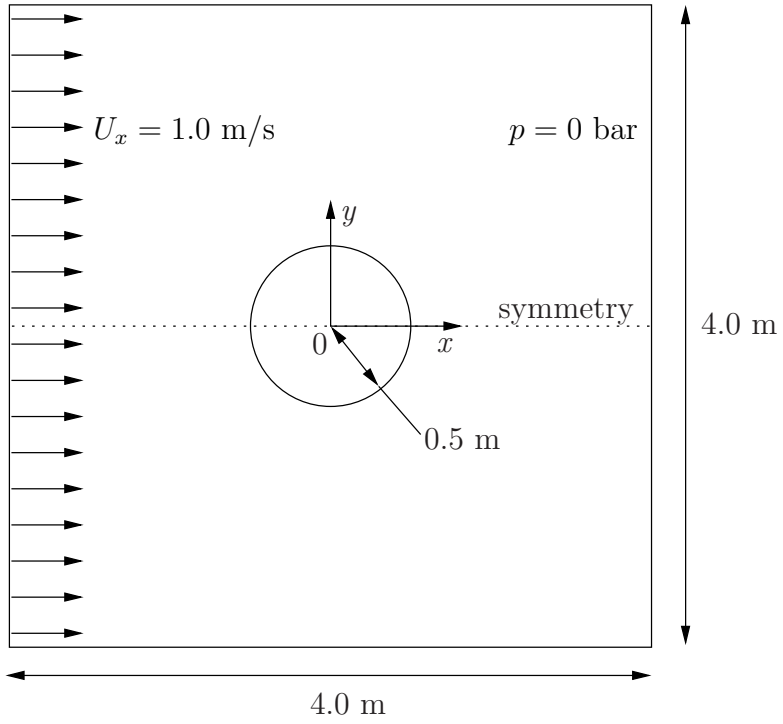


Figure 3.1: Geometry of flow round a cylinder

Governing equations

- Mass continuity for an incompressible fluid

$$\nabla \bullet \mathbf{U} = 0 \tag{3.1}$$

- Pressure equation for an incompressible, irrotational fluid assuming steady-state conditions

$$\nabla^2 p = 0 \tag{3.2}$$

Boundary conditions

- Inlet (left) with fixed velocity $\mathbf{U} = (1, 0, 0)$ m/s.
- Outlet (right) with a fixed pressure $p = 0$ Pa.
- No-slip wall (bottom);
- Symmetry plane (top).

Initial conditions $U = 0$ m/s, $p = 0$ Pa — required in OpenFOAM input files but not necessary for the solution since the problem is steady-state.

Solver name potentialFoam: a potential flow code, *i.e.* assumes the flow is incompressible, steady, irrotational, inviscid and it ignores gravity.

Case name *cylinder* case located in the *$FOAM_TUTORIALS/potentialFoam* directory.

3.1.2 Note on potentialFoam

potentialFoam is a useful solver to validate OpenFOAM since the assumptions of potential flow are such that an analytical solution exists for cases whose geometries are relatively simple. In this example of flow around a cylinder an analytical solution exists with which we can compare our numerical solution. potentialFoam can also be run more like a utility to provide a (reasonably) conservative initial **U** field for a problem. When running certain cases, this can useful for avoiding instabilities due to the initial field being unstable. In short, potentialFoam creates a conservative field from a non-conservative initial field supplied by the user.

3.1.3 Mesh generation

Mesh generation using blockMesh has been described in tutorials in the User Guide. In this case, the mesh consists of 10 blocks as shown in Figure 3.2. Remember that all

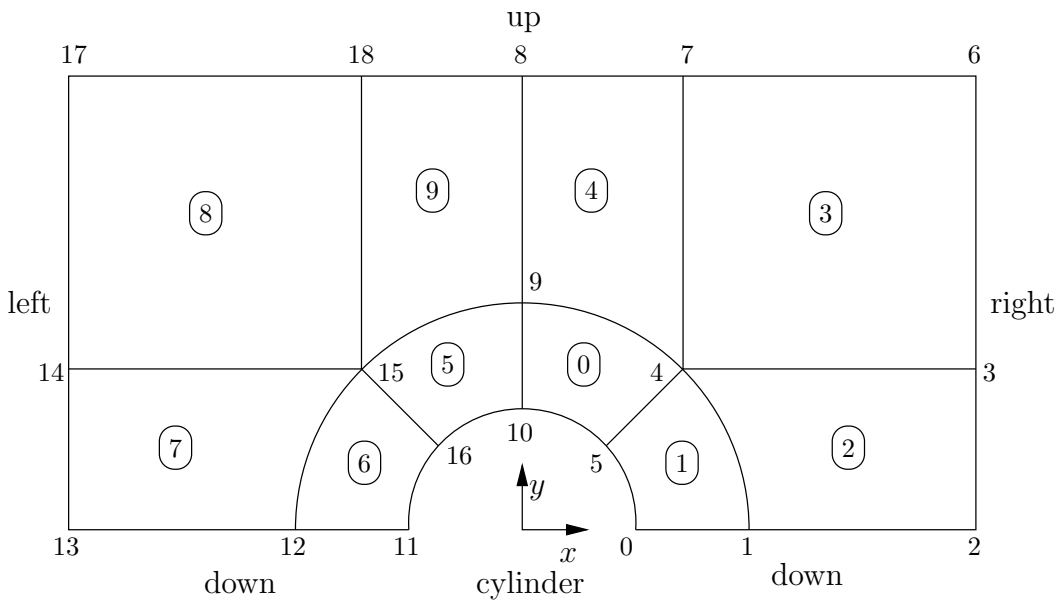


Figure 3.2: Blocks in cylinder geometry

meshes are treated as 3 dimensional in OpenFOAM. If we wish to solve a 2 dimensional problem, we must describe a 3 dimensional mesh that is only one cell thick in the third direction that is not solved. In Figure 3.2 we show only the back plane of the geometry, along $z = -0.5$, in which the vertex numbers are numbered 0-18. The other 19 vertices in the front plane, $z = +0.5$, are numbered in the same order as the back plane, as shown in the mesh description file below:

```
1  /*--------------------------------*- C++ -*----------------------------------*\
2  | =========                 |                                                 |
3  | \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
4  |  \\    /   O peration     | Version:  2.2.2                                 |
5  |   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
6  |    \\/     M anipulation  |                                                 |
7  \*---------------------------------------------------------------------------*/
8  FoamFile
9  {
10     version     2.0;
11     format      ascii;
12     class       dictionary;
13     object      blockMeshDict;
14 }
15 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
16
17 convertToMeters 1;
```

```
18
19  vertices #codeStream
20  {
21      codeInclude
22      #{
23          #include "pointField.H"
24      #};
25
26      code
27      #{
28          pointField points(19);
29          points[0]  = point(0.5, 0, -0.5);
30          points[1]  = point(1, 0, -0.5);
31          points[2]  = point(2, 0, -0.5);
32          points[3]  = point(2, 0.707107, -0.5);
33          points[4]  = point(0.707107, 0.707107, -0.5);
34          points[5]  = point(0.353553, 0.353553, -0.5);
35          points[6]  = point(2, 2, -0.5);
36          points[7]  = point(0.707107, 2, -0.5);
37          points[8]  = point(0, 2, -0.5);
38          points[9]  = point(0, 1, -0.5);
39          points[10] = point(0, 0.5, -0.5);
40          points[11] = point(-0.5, 0, -0.5);
41          points[12] = point(-1, 0, -0.5);
42          points[13] = point(-2, 0, -0.5);
43          points[14] = point(-2, 0.707107, -0.5);
44          points[15] = point(-0.707107, 0.707107, -0.5);
45          points[16] = point(-0.353553, 0.353553, -0.5);
46          points[17] = point(-2, 2, -0.5);
47          points[18] = point(-0.707107, 2, -0.5);
48
49          // Duplicate z points
50          label sz = points.size();
51          points.setSize(2*sz);
52          for (label i = 0; i < sz; i++)
53          {
54              const point& pt = points[i];
55              points[i+sz] = point(pt.x(), pt.y(), -pt.z());
56          }
57
58          os  << points;
59      #};
60  };
61
62
63  blocks
64  (
65      hex (5 4 9 10 24 23 28 29) (10 10 1) simpleGrading (1 1 1)
66      hex (0 1 4 5 19 20 23 24) (10 10 1) simpleGrading (1 1 1)
67      hex (1 2 3 4 20 21 22 23) (20 10 1) simpleGrading (1 1 1)
68      hex (4 3 6 7 23 22 25 26) (20 20 1) simpleGrading (1 1 1)
69      hex (9 4 7 8 28 23 26 27) (10 20 1) simpleGrading (1 1 1)
70      hex (15 16 10 9 34 35 29 28) (10 10 1) simpleGrading (1 1 1)
71      hex (12 11 16 15 31 30 35 34) (10 10 1) simpleGrading (1 1 1)
72      hex (13 12 15 14 32 31 34 33) (20 10 1) simpleGrading (1 1 1)
73      hex (14 15 18 17 33 34 37 36) (20 20 1) simpleGrading (1 1 1)
74      hex (15 9 8 18 34 28 27 37) (10 20 1) simpleGrading (1 1 1)
75  );
76
77  edges
78  (
79      arc 0 5 (0.469846 0.17101 -0.5)
80      arc 5 10 (0.17101 0.469846 -0.5)
81      arc 1 4 (0.939693 0.34202 -0.5)
82      arc 4 9 (0.34202 0.939693 -0.5)
83      arc 19 24 (0.469846 0.17101 0.5)
84      arc 24 29 (0.17101 0.469846 0.5)
85      arc 20 23 (0.939693 0.34202 0.5)
86      arc 23 28 (0.34202 0.939693 0.5)
87      arc 11 16 (-0.469846 0.17101 -0.5)
88      arc 16 10 (-0.17101 0.469846 -0.5)
89      arc 12 15 (-0.939693 0.34202 -0.5)
90      arc 15 9 (-0.34202 0.939693 -0.5)
91      arc 30 35 (-0.469846 0.17101 0.5)
92      arc 35 29 (-0.17101 0.469846 0.5)
93      arc 31 34 (-0.939693 0.34202 0.5)
94      arc 34 28 (-0.34202 0.939693 0.5)
95  );
96
```

```
97  boundary
98  (
99      down
100     {
101         type symmetryPlane;
102         faces
103         (
104             (0 1 20 19)
105             (1 2 21 20)
106             (12 11 30 31)
107             (13 12 31 32)
108         );
109     }
110     right
111     {
112         type patch;
113         faces
114         (
115             (2 3 22 21)
116             (3 6 25 22)
117         );
118     }
119     up
120     {
121         type symmetryPlane;
122         faces
123         (
124             (7 8 27 26)
125             (6 7 26 25)
126             (8 18 37 27)
127             (18 17 36 37)
128         );
129     }
130     left
131     {
132         type patch;
133         faces
134         (
135             (14 13 32 33)
136             (17 14 33 36)
137         );
138     }
139     cylinder
140     {
141         type symmetryPlane;
142         faces
143         (
144             (10 5 24 29)
145             (5 0 19 24)
146             (16 10 29 35)
147             (11 16 35 30)
148         );
149     }
150 );
151
152 mergePatchPairs
153 (
154 );
155
156 // ************************************************************************* //
```

3.1.4 Boundary conditions and initial fields

Using FoamX or editing case files by hand, set the boundary conditions in accordance with the problem description in Figure 3.1, *i.e.* the left boundary should be an Inlet, the right boundary should be an Outlet and the down and cylinder boundaries should be symmetryPlane. The top boundary conditions is chosen so that we can make the most genuine comparison with our analytical solution which uses the assumption that the domain is infinite in the y direction. The result is that the normal gradient of **U** is small along a plane coinciding with our boundary. We therefore impose the condition that the normal component is zero, *i.e.* specify the boundary as a symmetryPlane, thereby ensuring that the comparison with the analytical is reasonable.

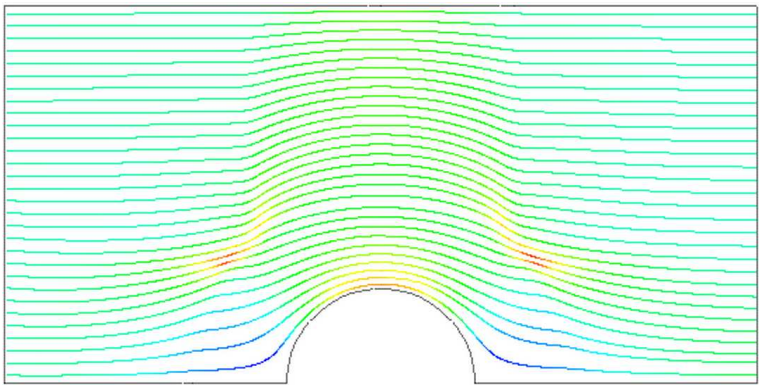
3.1.5 Running the case

No fluid properties need be specified in this problem since the flow is assumed to be incompressible and inviscid. In the *system* subdirectory, the *controlDict* specifies the control parameters for the run. Note that since we assume steady flow, we only run for 1 time step:

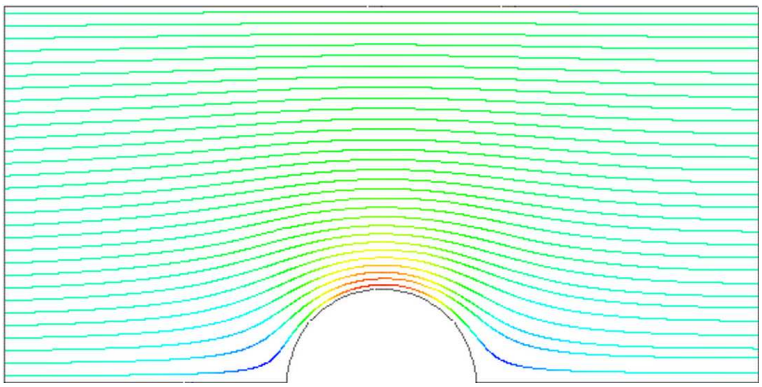
```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.2                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     potentialFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         1;

deltaT          1;

writeControl    timeStep;

writeInterval   1;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

functions
{
    difference
    {
        // Load the library containing the 'coded' functionObject
        functionObjectLibs ("libutilityFunctionObjects.so");
        type coded;
        // Name of on-the-fly generated functionObject
        redirectType error;
        code
        #{
            // Lookup U
            Info<< "Looking up field U\n" << endl;
            const volVectorField& U = mesh().lookupObject<volVectorField>("U");

            Info<< "Reading inlet velocity  uInfX\n" << endl;

            scalar ULeft = 0.0;
            label leftI = mesh().boundaryMesh().findPatchID("left");
            const fvPatchVectorField& fvp = U.boundaryField()[leftI];
            if (fvp.size())
            {
                ULeft = fvp[0].x();
            }
            reduce(ULeft, maxOp<scalar>());

            dimensionedScalar uInfX
```

```
75              (
76                  "uInfx",
77                  dimensionSet(0, 1, -1, 0, 0),
78                  ULeft
79              );
80
81              Info << "U at inlet = " << uInfX.value() << " m/s" << endl;
82
83
84              scalar magCylinder = 0.0;
85              label cylI = mesh().boundaryMesh().findPatchID("cylinder");
86              const fvPatchVectorField& cylFvp = mesh().C().boundaryField()[cylI];
87              if (cylFvp.size())
88              {
89                  magCylinder = mag(cylFvp[0]);
90              }
91              reduce(magCylinder, maxOp<scalar>());
92
93              dimensionedScalar radius
94              (
95                  "radius",
96                  dimensionSet(0, 1, 0, 0, 0),
97                  magCylinder
98              );
99
100             Info << "Cylinder radius = " << radius.value() << " m" << endl;
101
102             volVectorField UA
103             (
104                 IOobject
105                 (
106                     "UA",
107                     mesh().time().timeName(),
108                     U.mesh(),
109                     IOobject::NO_READ,
110                     IOobject::AUTO_WRITE
111                 ),
112                 U
113             );
114
115             Info<< "\nEvaluating analytical solution" << endl;
116
117             const volVectorField& centres = UA.mesh().C();
118             volScalarField magCentres(mag(centres));
119             volScalarField theta(acos((centres & vector(1,0,0))/magCentres));
120
121             volVectorField cs2theta
122             (
123                 cos(2*theta)*vector(1,0,0)
124               + sin(2*theta)*vector(0,1,0)
125             );
126
127             UA = uInfX*(dimensionedVector(vector(1,0,0))
128               - pow((radius/magCentres),2)*cs2theta);
129
130             // Force writing of UA (since time has not changed)
131             UA.write();
132
133             volScalarField error("error", mag(U-UA)/mag(UA));
134
135             Info<<"Writing relative error in U to " << error.objectPath()
136                 << endl;
137
138             error.write();
139         #};
140     }
141 }
142
143
144 // ************************************************************************* //
```

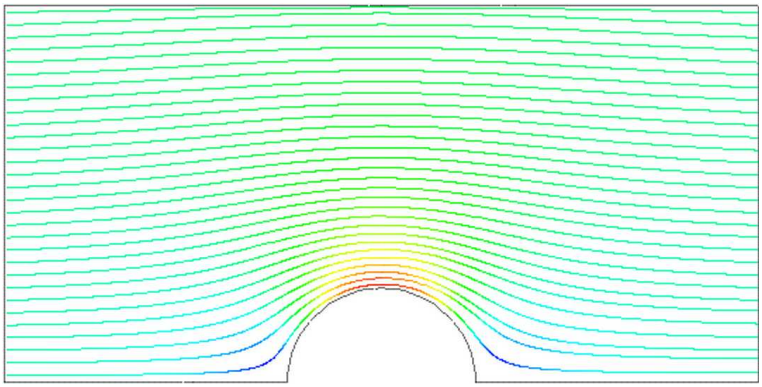
potentialFoam executes an iterative loop around the pressure equation which it solves in order that explicit terms relating to non-orthogonal correction in the Laplacian term may be updated in successive iterations. The number of iterations around the pressure equation is controlled by the `nNonOrthogonalCorrectors` keyword in *controlDict*. In the first instance we can set `nNonOrthogonalCorrectors` to 0 so that no loops are performed, *i.e.* the pressure equation is solved once, and there is no non-orthogonal correction. The solution is shown in Figure 3.3(a) (at $t = 1$, when the steady-state simulation is complete). We expect the solution to show smooth streamlines passing across the domain



(a) With no non-orthogonal correction



(b) With non-orthogonal correction



(c) Analytical solution

Figure 3.3: Streamlines of potential flow

as in the analytical solution in Figure 3.3(c), yet there is clearly some error in the regions where there is high non-orthogonality in the mesh, *e.g.* at the join of blocks 0, 1 and 3. The case can be run a second time with some non-orthogonal correction by setting `nNonOrthogonalCorrectors` to 3. The solution shows smooth streamlines with no significant error due to non-orthogonality as shown in Figure 3.3(b).

3.2 Steady turbulent flow over a backward-facing step

In this example we shall investigate steady turbulent flow over a backward-facing step. The problem description is taken from one used by Pitz and Daily in an experimental investigation [**] against which the computed solution can be compared. This example introduces the following OpenFOAM features for the first time:

- generation of a mesh using blockMesh using full mesh grading capability;
- steady turbulent flow.

3.2.1 Problem specification

The problem is defined as follows:

Solution domain The domain is 2 dimensional, consisting of a short inlet, a backward-facing step and converging nozzle at outlet as shown in Figure 3.4.

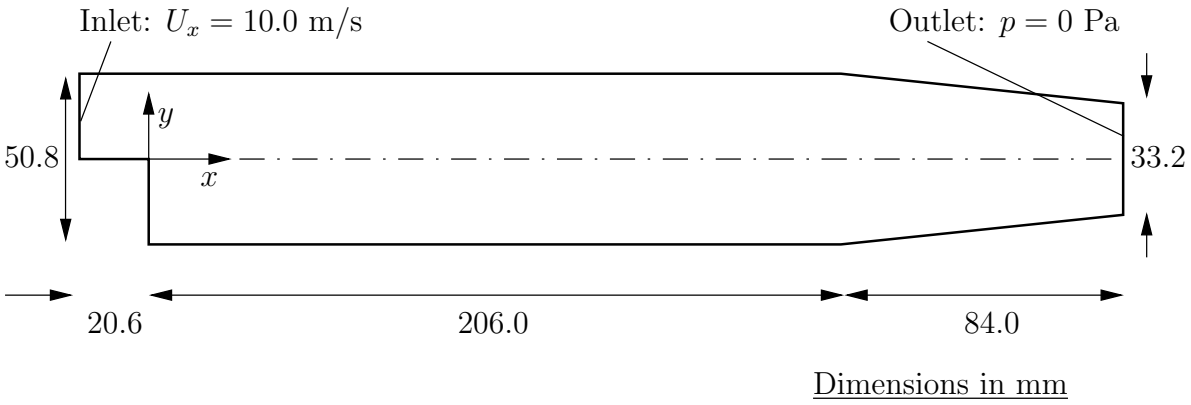


Figure 3.4: Geometry of backward-facing step

Governing equations

- Mass continuity for incompressible flow

$$\nabla \bullet \mathbf{U} = 0 \tag{3.3}$$

- Steady flow momentum equation

$$\nabla \bullet (\mathbf{U}\mathbf{U}) + \nabla \bullet \mathbf{R} = -\nabla p \tag{3.4}$$

where p is kinematic pressure and (in slightly over-simplistic terms) $\mathbf{R} = \nu_{eff}\nabla\mathbf{U}$ is the viscous stress term with an effective kinematic viscosity ν_{eff}, calculated from selected transport and turbulence models.

Initial conditions $U = 0$ m/s, $p = 0$ Pa — required in OpenFOAM input files but not necessary for the solution since the problem is steady-state.

Boundary conditions

- Inlet (left) with fixed velocity $\mathbf{U} = (10, 0, 0)$ m/s;
- Outlet (right) with fixed pressure $p = 0$ Pa;
- No-slip walls on other boundaries.

Transport properties

- Kinematic viscosity of air $\nu = \mu/\rho = 18.1 \times 10^{-6}/1.293 = 14.0$ μm^2/s

Turbulence model

- Standard $k - \epsilon$;
- Coefficients: $C_\mu = 0.09; C_1 = 1.44; C_2 = 1.92; \alpha_k = 1; \alpha_\epsilon = 0.76923$.

Solver name simpleFoam: an implementation for steady incompressible flow.

Case name *pitzDaily*, located in the *$FOAM_TUTORIALS/simpleFoam* directory.

The problem is solved using simpleFoam, so-called as it is an implementation for steady flow using the SIMPLE algorithm [**]. The solver has full access to all the turbulence models in the incompressibleTurbulenceModels library and the non-Newtonian models incompressibleTransportModels library of the standard OpenFOAM release.

3.2.2 Mesh generation

We expect that the flow in this problem is reasonably complex and an optimum solution will require grading of the mesh. In general, the regions of highest shear are particularly critical, requiring a finer mesh than in the regions of low shear. We can anticipate where high shear will occur by considering what the solution might be in advance of any calculation. At the inlet we have strong uniform flow in the x direction and, as it passes over the step, it generates shear on the fluid below, generating a vortex in the bottom half of the domain. The regions of high shear will therefore be close to the centreline of the domain and close to the walls.

The domain is subdivided into 12 blocks as shown in Figure 3.5.

The mesh is 3 dimensional, as always in OpenFOAM, so in Figure 3.5 we are viewing the back plane along $z = -0.5$. The full set of vertices and blocks are given in the mesh description file below:

```
1  /*--------------------------------*- C++ -*----------------------------------*\
2  | =========                 |                                                 |
3  | \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
4  |  \\    /   O peration     | Version:  2.2.2                                 |
5  |   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
6  |    \\/     M anipulation  |                                                 |
7  \*---------------------------------------------------------------------------*/
8  FoamFile
9  {
10     version     2.0;
11     format      ascii;
12     class       dictionary;
13     object      blockMeshDict;
14 }
15 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
16
17 convertToMeters 0.001;
18
```

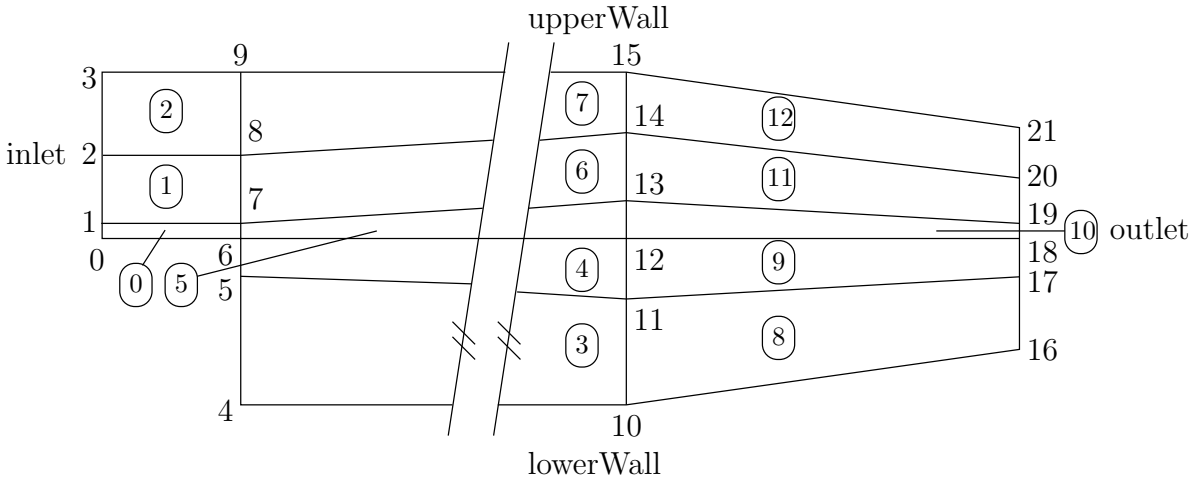


Figure 3.5: Blocks in backward-facing step

```
vertices
(
    (-20.6 0 -0.5)
    (-20.6 3 -0.5)
    (-20.6 12.7 -0.5)
    (-20.6 25.4 -0.5)
    (0 -25.4 -0.5)
    (0 -5 -0.5)
    (0 0 -0.5)
    (0 3 -0.5)
    (0 12.7 -0.5)
    (0 25.4 -0.5)
    (206 -25.4 -0.5)
    (206 -8.5 -0.5)
    (206 0 -0.5)
    (206 6.5 -0.5)
    (206 17 -0.5)
    (206 25.4 -0.5)
    (290 -16.6 -0.5)
    (290 -6.3 -0.5)
    (290 0 -0.5)
    (290 4.5 -0.5)
    (290 11 -0.5)
    (290 16.6 -0.5)
    (-20.6 0 0.5)
    (-20.6 3 0.5)
    (-20.6 12.7 0.5)
    (-20.6 25.4 0.5)
    (0 -25.4 0.5)
    (0 -5 0.5)
    (0 0 0.5)
    (0 3 0.5)
    (0 12.7 0.5)
    (0 25.4 0.5)
    (206 -25.4 0.5)
    (206 -8.5 0.5)
    (206 0 0.5)
    (206 6.5 0.5)
    (206 17 0.5)
    (206 25.4 0.5)
    (290 -16.6 0.5)
    (290 -6.3 0.5)
    (290 0 0.5)
    (290 4.5 0.5)
    (290 11 0.5)
    (290 16.6 0.5)
);

blocks
(
    hex (0 6 7 1 22 28 29 23) (18 7 1) simpleGrading (0.5 1.8 1)
    hex (1 7 8 2 23 29 30 24) (18 10 1) simpleGrading (0.5 4 1)
    hex (2 8 9 3 24 30 31 25) (18 13 1) simpleGrading (0.5 0.25 1)
    hex (4 10 11 5 26 32 33 27) (180 18 1) simpleGrading (4 1 1)
    hex (5 11 12 6 27 33 34 28) (180 9 1) edgeGrading (4 4 4 4 0.5 1 1 0.5 1 1 1 1)
    hex (6 12 13 7 28 34 35 29) (180 7 1) edgeGrading (4 4 4 4 1.8 1 1 1.8 1 1 1 1)
    hex (7 13 14 8 29 35 36 30) (180 10 1) edgeGrading (4 4 4 4 4 1 1 4 1 1 1 1)
    hex (8 14 15 9 30 36 37 31) (180 13 1) simpleGrading (4 0.25 1)
    hex (10 16 17 11 32 38 39 33) (25 18 1) simpleGrading (2.5 1 1)
```

```
78      hex (11 17 18 12 33 39 40 34) (25 9 1) simpleGrading (2.5 1 1)
79      hex (12 18 19 13 34 40 41 35) (25 7 1) simpleGrading (2.5 1 1)
80      hex (13 19 20 14 35 41 42 36) (25 10 1) simpleGrading (2.5 1 1)
81      hex (14 20 21 15 36 42 43 37) (25 13 1) simpleGrading (2.5 0.25 1)
82  );
83
84  edges
85  (
86  );
87
88  boundary
89  (
90      inlet
91      {
92          type patch;
93          faces
94          (
95              (0 22 23 1)
96              (1 23 24 2)
97              (2 24 25 3)
98          );
99      }
100     outlet
101     {
102         type patch;
103         faces
104         (
105             (16 17 39 38)
106             (17 18 40 39)
107             (18 19 41 40)
108             (19 20 42 41)
109             (20 21 43 42)
110         );
111     }
112     upperWall
113     {
114         type wall;
115         faces
116         (
117             (3 25 31 9)
118             (9 31 37 15)
119             (15 37 43 21)
120         );
121     }
122     lowerWall
123     {
124         type wall;
125         faces
126         (
127             (0 6 28 22)
128             (6 5 27 28)
129             (5 4 26 27)
130             (4 10 32 26)
131             (10 16 38 32)
132         );
133     }
134     frontAndBack
135     {
136         type empty;
137         faces
138         (
139             (22 28 29 23)
140             (23 29 30 24)
141             (24 30 31 25)
142             (26 32 33 27)
143             (27 33 34 28)
144             (28 34 35 29)
145             (29 35 36 30)
146             (30 36 37 31)
147             (32 38 39 33)
148             (33 39 40 34)
149             (34 40 41 35)
150             (35 41 42 36)
151             (36 42 43 37)
152             (0 1 7 6)
153             (1 2 8 7)
154             (2 3 9 8)
155             (4 5 11 10)
156             (5 6 12 11)
157             (6 7 13 12)
158             (7 8 14 13)
159             (8 9 15 14)
160             (10 11 17 16)
161             (11 12 18 17)
162             (12 13 19 18)
```

```
163                 (13 14 20 19)
164                 (14 15 21 20)
165             );
166         }
167     );
168
169     mergePatchPairs
170     (
171     );
172
173     // ************************************************************************* //
```

A major feature of this problem is the use of the full mesh grading capability of blockMesh that is described in section 5.3.1 of the User Guide. The user can see that blocks 4,5 and 6 use the full list of 12 expansion ratios. The expansion ratios correspond to each edge of the block, the first 4 to the edges aligned in the local x_1 direction, the second 4 to the edges in the local x_2 direction and the last 4 to the edges in the local x_3 direction. In blocks 4, 5, and 6, the ratios are equal for all edges in the local x_1 and x_3 directions but not for the edges in the x_2 direction that corresponds in all blocks to the global y. If we consider the ratios used in relation to the block definition in section 5.3.1 of the User Guide, we realize that different gradings have been prescribed along the left and right edges in blocks 4,5 and 6 in Figure 3.5. The purpose of this differential grading is to generate a fine mesh close to the most critical region of flow, the corner of the step, and allow it to expand into the rest of the domain.

The mesh can be generated using blockMesh from the command line or from within FoamX and viewed as described in previous examples.

3.2.3 Boundary conditions and initial fields

The case files can be viewed, or edited from within FoamX or by hand. In this case, we are required to set the initial and boundary fields for velocity **U**, pressure p, turbulent kinetic energy k and dissipation rate ε. The boundary conditions can be specified by setting the physical patch types in FoamX: the upper and lower walls are set to `Wall`, the left patch to `Inlet` and the right patch to `Outlet`. These physical boundary conditions require us to specify a fixedValue at the inlet on **U**, k and ε. **U** is given in the problem specification, but the values of k and ϵ must be chosen by the user in a similar manner to that described in section 2.1.8.1 of the User Guide. We assume that the inlet turbulence is isotropic and estimate the fluctuations to be 5% of **U** at the inlet. We have

$$U'_x = U'_y = U'_z = \frac{5}{100}10 = 0.5 \text{ m/s} \tag{3.5}$$

and

$$k = \frac{3}{2}(0.5)^2 = 0.375 \text{ m}^2/\text{s}^2 \tag{3.6}$$

If we estimate the turbulent length scale l to be 10% of the width of the inlet then

$$\varepsilon = \frac{C_\mu^{0.75} k^{1.5}}{l} = \frac{0.09^{0.75} 0.375^{1.5}}{0.1 \times 25.4 \times 10^{-3}} = 14.855 \,\text{m}^2/\text{s}^3 \tag{3.7}$$

At the outlet we need only specify the pressure $p = 0$Pa.

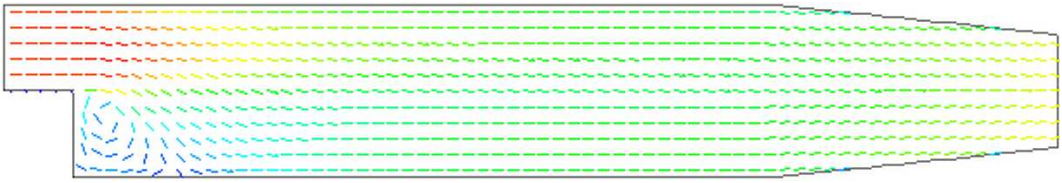
3.2.4 Case control

The choices of *fvSchemes* are as follows: the `timeScheme` should be `SteadyState`; the `gradScheme` and `laplacianScheme` should be set as default to `Gauss`; and, the `divScheme` should be set to `UD` to ensure boundedness.

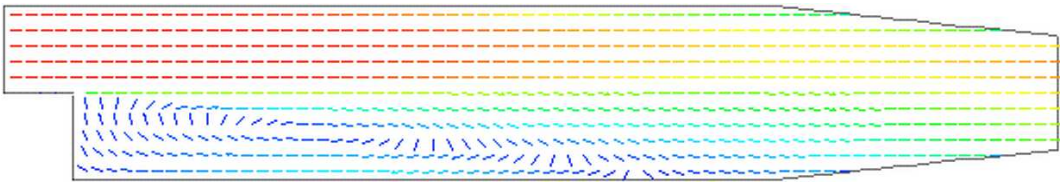
Special attention should be paid to the settings of *fvTolerances*. Although the top level simpleFoam code contains only equations for p and $\mathbf{U}$, the turbulent model solves equations for k, ε and $\mathbf{R}$, and tolerance settings are required for all 5 equations. A `solverTolerance` of 10^{-5} and `solverRelativeTolerance` of 0.1 are acceptable for all variables with the exception of p when 10^{-6} and 0.01 are recommended. Under-relaxation of the solution is required since the problem is steady. A `relaxationFactor` of 0.7 is acceptable for $\mathbf{U}$, k, ε and $\mathbf{R}$ but 0.3 is required for p to avoid numerical instability.

Finally, in *controlDict*, the time step `deltaT` should be set to 1 since in steady state cases such as this is effectively an iteration counter. With benefit of hindsight we know that the solution requires 1000 iterations reach reasonable convergence, hence `endTime` is set to 1000. Ensure that the `writeFrequency` is sufficiently high, *e.g.* 50, that you will not fill the hard disk with data during run time.

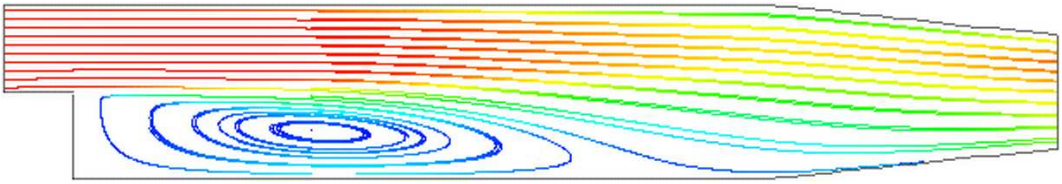
3.2.5 Running the case and post-processing



(a) Velocity vectors after 50 iterations



(b) Velocity vectors at 1000 iterations



(c) Streamlines at 1000 iterations

Figure 3.6: Development of a vortex in the backward-facing step.

Run the case and post-process the results. After a few iterations, *e.g.* 50, a vortex develops beneath the corner of the step that is the height of the step but narrow in the x-direction as shown by the vector plot of velocities is shown Figure 3.6(a). Over several iterations the vortex stretches in the x-direction from the step to the outlet until at 1000 iterations the system reaches a steady-state in which the vortex is fully developed as shown in Figure 3.6(b-c).

3.3 Supersonic flow over a forward-facing step

In this example we shall investigate supersonic flow over a forward-facing step. The problem description involves a flow of Mach 3 at an inlet to a rectangular geometry with a step near the inlet region that generates shock waves.

This example introduces the following OpenFOAM features for the first time:

- supersonic flow;

3.3.1 Problem specification

The problem is defined as follows:

Solution domain The domain is 2 dimensional and consists of a short inlet section followed by a forward-facing step of 20% the height of the section as shown in Figure 3.7

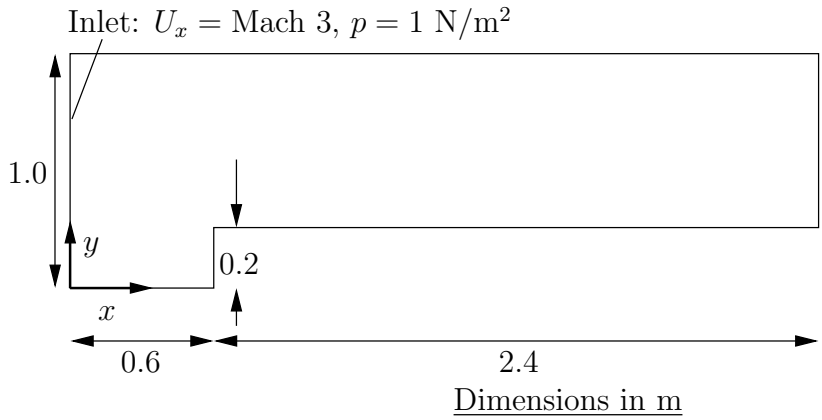


Figure 3.7: Geometry of the forward step geometry

Governing equations

- Mass continuity

$$\frac{\partial \rho}{\partial t} + \nabla \bullet (\rho \mathbf{U}) = 0 \tag{3.8}$$

- Ideal gas

$$p = \rho R T \tag{3.9}$$

- Momentum equation for Newtonian fluid

$$\frac{\partial \rho \mathbf{U}}{\partial t} + \nabla \bullet (\rho \mathbf{U}\mathbf{U}) - \nabla \bullet \mu \nabla \mathbf{U} = -\nabla p \tag{3.10}$$

- Energy equation for fluid (ignoring some viscous terms), $e = C_v T$, with Fourier's Law $\mathbf{q} = -k\nabla T$

$$\frac{\partial \rho e}{\partial t} + \nabla \bullet (\rho \mathbf{U} e) - \nabla \bullet \left(\frac{k}{C_v}\right) \nabla e = p \nabla \bullet \mathbf{U} \tag{3.11}$$

Initial conditions $U = 0$ m/s, $p = 1$ Pa, $T = 1$ K.

Boundary conditions

- Inlet (left) with fixedValue for velocity $U = 3$ m/s = Mach 3, pressure $p = 1$ Pa and temperature $T = 1$ K;
- Outlet (right) with zeroGradient on U, p and T;
- No-slip adiabatic wall (bottom);

- Symmetry plane (top).

Transport properties

- Dynamic viscosity of air $\mu = 18.1\mu\text{Pa s}$

Thermodynamic properties

- Specific heat at constant volume $C_v = 1.78571$ J/kg K
- Gas constant $R = 0.714286$ J/kg K
- Conductivity $k = 32.3$ μW/m K

Case name *forwardStep* case located in the *$FOAM_TUTORIALS/sonicFoam* directory.

Solver name sonicFoam: an implementation for compressible trans-sonic/supersonic laminar gas flow.

The case is designed such that the speed of sound of the gas $c = \sqrt{\gamma RT} = 1$ m/s, the consequence being that the velocities are directly equivalent to the Mach number, *e.g.* the inlet velocity of 3 m/s is equivalent to Mach 3. This speed of sound calculation can be verified using the relationship for a perfect gas, $C_p - Cv = R$, *i.e.* the ratio of specific heats

$$\gamma = C_p/C_v = \frac{R}{C_v} + 1 \tag{3.12}$$

3.3.2 Mesh generation

The mesh used in this case is relatively simple, specified with uniform rectangular cells of length 0.06 m in the x direction and 0.05 m in the y direction. The geometry can simply be divided into 3 blocks, one below the top of the step, and two above the step, one either side of the step front. The full set of vertices and blocks are given in the mesh description file below:

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.2                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 1;

vertices
(
    (0 0 -0.05)
    (0.6 0 -0.05)
    (0 0.2 -0.05)
    (0.6 0.2 -0.05)
    (3 0.2 -0.05)
    (0 1 -0.05)
    (0.6 1 -0.05)
    (3 1 -0.05)
    (0 0 0.05)
    (0.6 0 0.05)
    (0 0.2 0.05)
    (0.6 0.2 0.05)
    (3 0.2 0.05)
```

```
34      (0 1 0.05)
35      (0.6 1 0.05)
36      (3 1 0.05)
37  );
38
39  blocks
40  (
41      hex (0 1 3 2 8 9 11 10) (25 10 1) simpleGrading (1 1 1)
42      hex (2 3 6 5 10 11 14 13) (25 40 1) simpleGrading (1 1 1)
43      hex (3 4 7 6 11 12 15 14) (100 40 1) simpleGrading (1 1 1)
44  );
45
46  edges
47  (
48  );
49
50  boundary
51  (
52      inlet
53      {
54          type patch;
55          faces
56          (
57              (0 8 10 2)
58              (2 10 13 5)
59          );
60      }
61      outlet
62      {
63          type patch;
64          faces
65          (
66              (4 7 15 12)
67          );
68      }
69      bottom
70      {
71          type symmetryPlane;
72          faces
73          (
74              (0 1 9 8)
75          );
76      }
77      top
78      {
79          type symmetryPlane;
80          faces
81          (
82              (5 13 14 6)
83              (6 14 15 7)
84          );
85      }
86      obstacle
87      {
88          type patch;
89          faces
90          (
91              (1 3 11 9)
92              (3 4 12 11)
93          );
94      }
95  );
96
97  mergePatchPairs
98  (
99  );
100
101 // ************************************************************************* //
```

3.3.3 Running the case

The case approaches a steady-state at some time after 5 s. The results for pressure at 10 s are shown in Figure 3.8. The results clearly show discontinuities in pressure, *i.e.* shock waves, emanating from ahead of the base of the step.

3.3.4 Exercise

The user can examine the effect on the solution of increasing the inlet velocity.

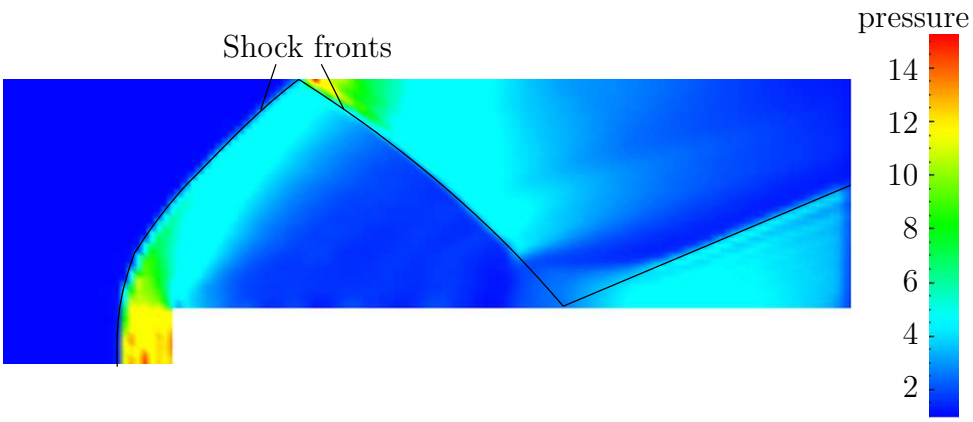


Figure 3.8: Shock fronts in the forward step problem

3.4 Decompression of a tank internally pressurised with water

In this example we shall investigate a problem of rapid opening of a pipe valve close to a pressurised liquid-filled tank. The prominent feature of the result in such cases is the propagation of pressure waves which must therefore be modelled as a compressible liquid.

This tutorial introduces the following OpenFOAM features for the first time:

- Mesh refinement
- Pressure waves in liquids

3.4.1 Problem specification

Solution domain The domain is 2 dimensional and consists of a tank with a small outflow pipe as shown in Figure 3.9

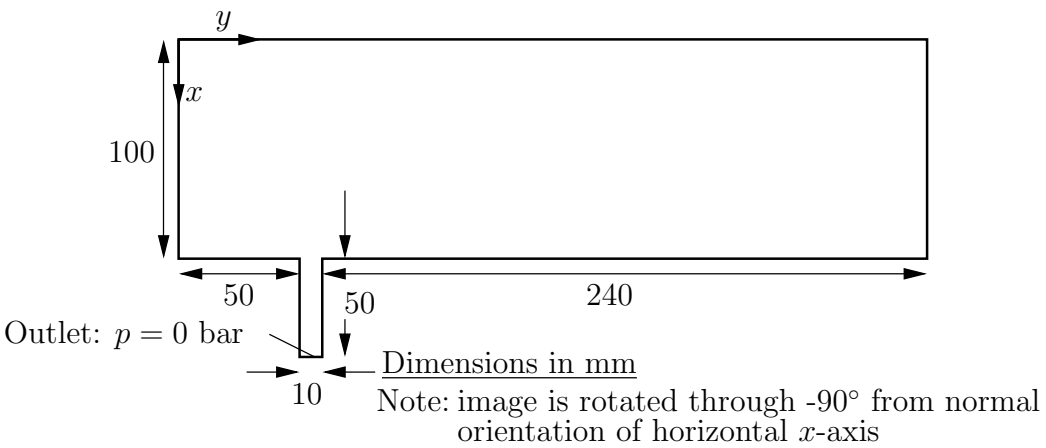


Figure 3.9: Geometry of a tank with outflow pipe

Governing equations This problem requires a model for compressibility ψ in the fluid in order to be able to resolve waves propagating at a finite speed. A barotropic relationship is used to relate density ρ and pressure p are related to ψ.

- Mass continuity

$$\frac{\partial \rho}{\partial t} + \nabla \bullet (\rho \mathbf{U}) = 0 \tag{3.13}$$

- The barotropic relationship

$$\frac{\partial \rho}{\partial p} = \frac{\rho}{K} = \psi \tag{3.14}$$

  where K is the bulk modulus

- Equation 3.14 is linearised as

$$\rho \approx \rho_0 + \psi\,(p - p_0) \tag{3.15}$$

  where ρ_0 and p_0 are the reference density and pressure respectively such that $\rho(p_0) = \rho_0$.

- Momentum equation for Newtonian fluid

$$\frac{\partial \rho \mathbf{U}}{\partial t} + \nabla \bullet (\rho \mathbf{U}\mathbf{U}) - \nabla \bullet \mu \nabla \mathbf{U} = -\nabla p \tag{3.16}$$

Boundary conditions Using FoamX the following physical boundary conditions can be set:

- `outerWall` is specified the wall condition;
- `axis` is specified as the symmetryPlane;
- `nozzle` is specified as a pressureOutlet where $p = 0$ bar.
- `front` and `back` boundaries are specified as empty.

Initial conditions $\mathbf{U} = \mathbf{0}$ m/s, $p = 100$ bar.

Transport properties

- Dynamic viscosity of water $\mu = 1.0$ mPa s

Thermodynamic properties

- Density of water $\rho = 1000$ kg/m^3
- Reference pressure $p_0 = 1$ bar
- Compressibility of water $\psi = 4.54 \times 10^{-7}$ s^2/m^2

Solver name sonicLiquidFoam: a compressible sonic laminar liquid flow code.

Case name *decompressionTank* case located in the *$FOAM_TUTORIALS/sonicLiquidFoam* directory.

3.4.2 Mesh Generation

The full geometry is modelled in this case; the set of vertices and blocks are given in the mesh description file below:

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.2                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
```

```
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 0.1;

vertices
(
    (0 0 -0.1)
    (1 0 -0.1)
    (0 0.5 -0.1)
    (1 0.5 -0.1)
    (1.5 0.5 -0.1)
    (0 0.6 -0.1)
    (1 0.6 -0.1)
    (1.5 0.6 -0.1)
    (0 3 -0.1)
    (1 3 -0.1)
    (0 0 0.1)
    (1 0 0.1)
    (0 0.5 0.1)
    (1 0.5 0.1)
    (1.5 0.5 0.1)
    (0 0.6 0.1)
    (1 0.6 0.1)
    (1.5 0.6 0.1)
    (0 3 0.1)
    (1 3 0.1)
);

blocks
(
    hex (0 1 3 2 10 11 13 12) (30 20 1) simpleGrading (1 1 1)
    hex (2 3 6 5 12 13 16 15) (30 5 1) simpleGrading (1 1 1)
    hex (3 4 7 6 13 14 17 16) (25 5 1) simpleGrading (1 1 1)
    hex (5 6 9 8 15 16 19 18) (30 95 1) simpleGrading (1 1 1)
);

edges
(
);

boundary
(
    outerWall
    {
        type wall;
        faces
        (
            (0 1 11 10)
            (1 3 13 11)
            (3 4 14 13)
            (7 6 16 17)
            (6 9 19 16)
            (9 8 18 19)
        );
    }
    axis
    {
        type symmetryPlane;
        faces
        (
            (0 10 12 2)
            (2 12 15 5)
            (5 15 18 8)
        );
    }
    nozzle
    {
        type patch;
        faces
        (
            (4 7 17 14)
        );
    }
    back
    {
        type empty;
        faces
        (
            (0 2 3 1)
            (2 5 6 3)
            (3 6 7 4)
```

```
            (5 8 9 6)
        );
    }
    front
    {
        type empty;
        faces
        (
            (10 11 13 12)
            (12 13 16 15)
            (13 14 17 16)
            (15 16 19 18)
        );
    }
);

mergePatchPairs
(
);

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

In order to improve the numerical accuracy, we shall use the reference level of 1 bar for the pressure field. Note that both the internal field level and the boundary conditions are offset by the reference level.

3.4.3 Preparing the Run

Before we commence the setup of the calculation, we need to consider the characteristic velocity of the phenomenon we are trying to capture. In the case under consideration, the fluid velocity will be very small, but the pressure wave will propagate with the speed of sound in water. The speed of sound is calculated as:

$$c = \sqrt{\frac{1}{\psi}} = \sqrt{\frac{1}{4.54 \times 10^{-7}}} = 1483.2\text{m/s}. \tag{3.17}$$

For the mesh described above, the characteristic mesh size is approximately 2 mm (note the scaling factor of 0.1 in the *blockMeshDict* file). Using

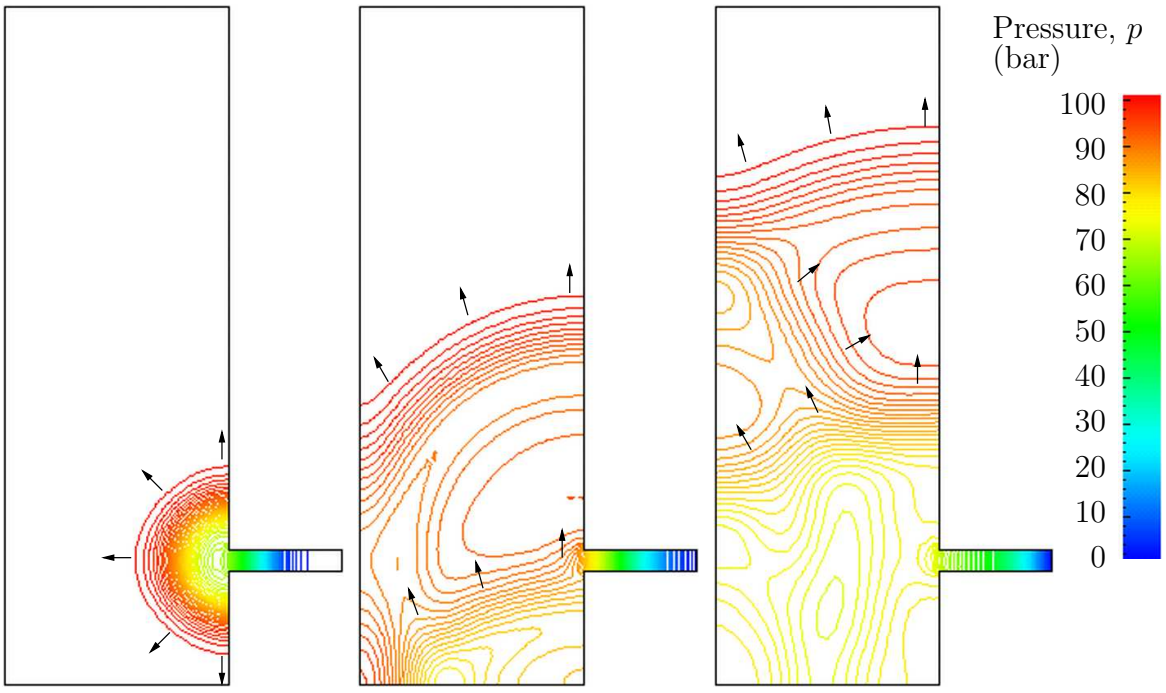
$$Co = \frac{U\,\Delta t}{\Delta x} \tag{3.18}$$

a reasonable time step is around $\Delta t = 5 \times 10^{-7}$s, giving the Co number of 0.35, based on the speed of sound. Also, note that the reported Co number by the code (associated with the convective velocity) will be two orders of magnitude smaller. As we are interested in the pressure wave propagation, we shall set the simulation time to 0.25 ms. For reference, the *controlDict* file is quoted below.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.2                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     sonicLiquidFoam;

startFrom       startTime;

```

```
22  startTime       0;
23
24  stopAt          endTime;
25
26  endTime         0.0001;
27
28  deltaT          5e-07;
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  // ************************************************************************* //
```

3.4.4 Running the case



(a) At $t = 50$ µs (b) At $t = 100$ µs (c) At $t = 150$ µs

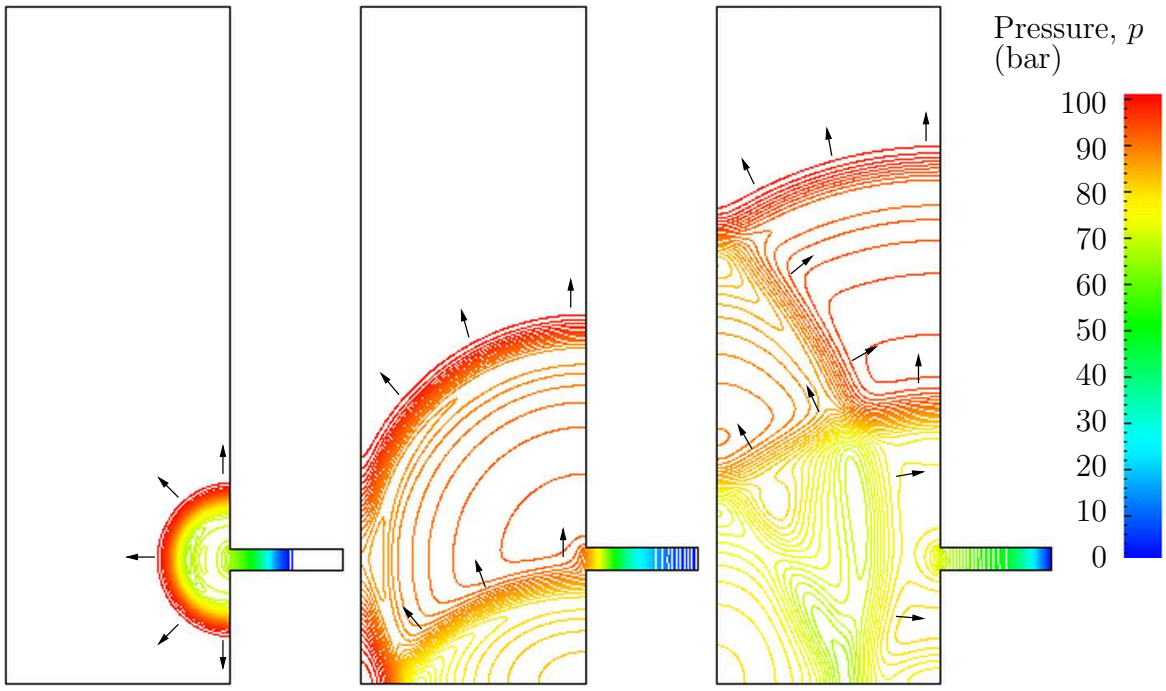
Figure 3.10: Propagation of pressure waves

The user can run the case and view results in dxFoam. The liquid flows out through the nozzle causing a wave to move along the nozzle. As it reaches the inlet to the tank, some of the wave is transmitted into the tank and some of it is reflected. While a wave is reflected up and down the inlet pipe, the waves transmitted into the tank expand and propagate through the tank. In Figure 3.10, the pressures are shown as contours so that the wave fronts are more clearly defined than if plotted as a normal isoline plot.

If the simulation is run for a long enough time for the reflected wave to return to the pipe, we can see that negative absolute pressure is detected. The modelling permits

this and has some physical basis since liquids can support tension, *i.e.* negative pressures. In reality, however, impurities or dissolved gases in liquids act as sites for cavitation, or vapourisation/boiling, of the liquid due to the low pressure. Therefore in practical situations, we generally do not observe pressures falling below the vapourisation pressure of the liquid; not at least for longer than it takes for the cavitation process to occur.

3.4.5 Improving the solution by refining the mesh



(a) At $t = 50$ µs (b) At $t = 100$ µs (c) At $t = 150$ µs

Figure 3.11: Propagation of pressure waves with refined mesh

Looking at the evolution of the resulting pressure field in time, we can clearly see the propagation of the pressure wave into the tank and numerous reflections from the inside walls. It is also obvious that the pressure wave is smeared over a number of cells. We shall now refine the mesh and reduce the time step to obtain a sharper front resolution. Simply edit the *blockMeshDict* and increase the number of cells by a factor of 4 in the x and y directions, *i.e.* block 0 becomes `(120 80 1)` from `(30 20 1)` and so on. Run blockMesh on this file. In addition, in order to maintain a Courant number below 1, the time step must be reduced accordingly to $\Delta t = 10^{-7}$ s. The second simulation gives considerably better resolution of the pressure waves as shown in Figure 3.11.

3.5 Magnetohydrodynamic flow of a liquid

In this example we shall investigate an flow of an electrically-conducting liquid through a magnetic field. The problem is one belonging to the branch of fluid dynamics known as magnetohydrodynamics (MHD) that uses mhdFoam.

3.5.1 Problem specification

The problem is known as the Hartmann problem, chosen as it contains an analytical solution with which mhdFoam can be validated. It is defined as follows:

Solution domain The domain is 2 dimensional and consists of flow along two parallel plates as shown in Fig. 3.12.

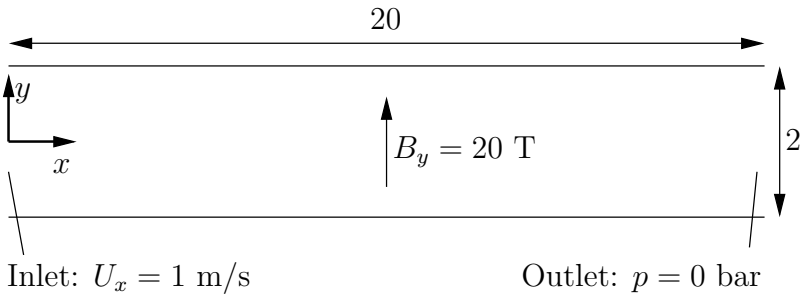


Figure 3.12: Geometry of the Hartmann problem

Governing equations

- Mass continuity for incompressible fluid

$$\nabla \bullet \mathbf{U} = 0 \tag{3.19}$$

- Momentum equation for incompressible fluid

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \bullet (\mathbf{U}\mathbf{U}) + \nabla \bullet (2\mathbf{B}\Gamma_{\mathbf{BU}}\mathbf{B}) + \nabla \bullet (\nu \mathbf{U}) + \nabla (\Gamma_{\mathbf{BU}}\mathbf{B} : \mathbf{B}) = -\nabla p \tag{3.20}$$

  where $\mathbf{B}$ is the magnetic flux density, $\Gamma_{\mathbf{BU}} = (2\mu\rho)^{-1}$.

- Maxwell's equations

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \tag{3.21}$$

  where $\mathbf{E}$ is the electric field strength.

$$\nabla \bullet \mathbf{B} = 0 \tag{3.22}$$

$$\nabla \times \mathbf{H} = \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t} = \mathbf{J} \tag{3.23}$$

  assuming $\partial \mathbf{D}/\partial t \ll \mathbf{J}$. Here, $\mathbf{H}$ is the magnetic field strength, $\mathbf{J}$ is the current density and $\mathbf{D}$ is the electric flux density.

- Charge continuity

$$\nabla \bullet \mathbf{J} = 0 \tag{3.24}$$

- Constitutive law

$$\mathbf{B} = \mu \mathbf{H} \tag{3.25}$$

- Ohm's law

$$\mathbf{J} = \sigma (\mathbf{E} + \mathbf{U} \times \mathbf{B}) \tag{3.26}$$

- Combining Equation 3.21, Equation 3.23, Equation 3.26, and taking the curl

$$\frac{\partial \mathbf{B}}{\partial t} + \nabla \bullet (\mathbf{UB}) - \nabla \bullet (\phi_{\mathbf{B}} \mathbf{U}) - \nabla \bullet (\Gamma_{\mathbf{B}} \mathbf{B}) = 0 \tag{3.27}$$

Boundary conditions

- `inlet` is specified the inlet condition with fixed velocity $\mathbf{U} = (1, 0, 0)$ m/s;
- `outlet` is specified as the outlet with with fixed pressure $p = 0$ Pa;
- `upperWall` is specified as a wall where $\mathbf{B} = (0, 20, 0)$ T.
- `lowerWall` is specified as a wall where $\mathbf{B} = (0, 20, 0)$ T.
- `front` and `back` boundaries are specified as empty.

Initial conditions $\mathbf{U} = \mathbf{0}$ m/s, $p = 100$ Pa, $\mathbf{B} = (0, 20, 0)$ T.

Transport properties

- Kinematic viscosity $\nu = 1$ Pa s
- Density $\rho = 1$ kg m/s
- Electrical conductivity $\sigma = 1\ (\Omega\,\mathrm{m})^{-1}$
- Permeability $\mu = 1$ H/m

Solver name mhdFoam: an incompressible laminar magneto-hydrodynamics code.

Case name *hartmann* case located in the *$FOAM_TUTORIALS/mhdFoam* directory.

3.5.2 Mesh generation

The geometry is simply modelled with 100 cells in the x-direction and 40 cells in the y-direction; the set of vertices and blocks are given in the mesh description file below:

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.2                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 1;

vertices
(
    (0 -1 0)
    (20 -1 0)
    (20 1 0)
    (0 1 0)
    (0 -1 0.1)
    (20 -1 0.1)
    (20 1 0.1)
    (0 1 0.1)
);

blocks
(
    hex (0 1 2 3 4 5 6 7) (100 40 1) simpleGrading (1 1 1)
);
```

```
35
36  edges
37  (
38  );
39
40  boundary
41  (
42      inlet
43      {
44          type patch;
45          faces
46          (
47              (0 4 7 3)
48          );
49      }
50      outlet
51      {
52          type patch;
53          faces
54          (
55              (2 6 5 1)
56          );
57      }
58      lowerWall
59      {
60          type patch;
61          faces
62          (
63              (1 5 4 0)
64          );
65      }
66      upperWall
67      {
68          type patch;
69          faces
70          (
71              (3 7 6 2)
72          );
73      }
74      frontAndBack
75      {
76          type empty;
77          faces
78          (
79              (0 3 2 1)
80              (4 5 6 7)
81          );
82      }
83  );
84
85  mergePatchPairs
86  (
87  );
88
89  // ************************************************************************* //
```

3.5.3 Running the case

The user can run the case and view results in dxFoam. It is also useful at this stage to run the Ucomponents utility to convert the **U** vector field into individual scalar components. MHD flow is governed by, amongst other things, the Hartmann number which is a measure of the ratio of electromagnetic body force to viscous force

$$M = BL\sqrt{\frac{\sigma}{\rho\nu}} \tag{3.28}$$

where L is the characteristic length scale. In this case with $B_y = 20$ T, $M = 20$ and the electromagnetic body forces dominate the viscous forces. Consequently with the flow fairly steady at $t = 2$ s the velocity profile is almost planar, viewed at a cross section midway along the domain $x = 10$ m. The user can plot a graph of the profile of U_x in dxFoam. Now the user should reduce the magnetic flux density **B** to 1 Tand re-run the code and Ucomponents. In this case, $M = 1$ and the electromagnetic body forces no longer dominate. The velocity profile consequently takes on the parabolic form, characteristic

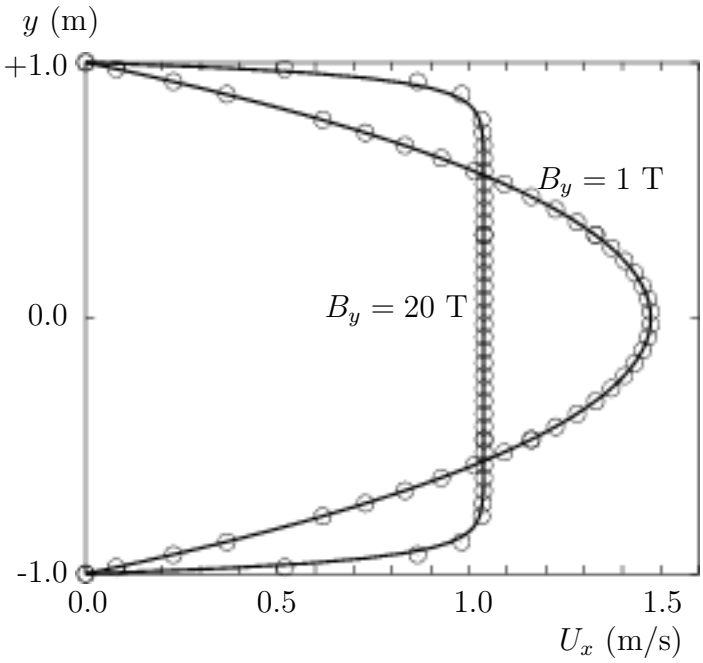


Figure 3.13: Velocity profile in the Hartmann problem for $B_y = 1$ T and $B_y = 20$ T.

of Poiseuille flow as shown in Figure 3.13. To validate the code the analytical solution for the velocity profile U_x is superimposed in Figure 3.13, given by:

$$\frac{U_x(y)}{U_x(0)} = \frac{\cosh M - \cosh M(y/L)}{\cosh M - 1} \tag{3.29}$$

where the characteristic length L is half the width of the domain, *i.e.* 1 m.

Index

Symbols Numbers 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

Symbols

* 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

Numbers

0 directory, U-104

A

access functions, P-19
addLayersControls keyword, U-146
adiabaticFlameT utility, U-94
adjointShapeOptimizationFoam solver, U-83
adjustableRunTime keyword entry, U-60, U-111
adjustTimeStep keyword, U-60, U-112
agglomerator keyword, U-123
algorithms tools, U-95
alphaContactAngle boundary condition, U-57
analytical solution, P-41
Animations window panel, U-168
anisotropicFilter model, U-99
Annotation window panel, U-24, U-167
ansysToFoam utility, U-89
APIfunctions model, U-98
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-89
Auto Accept button, U-168
autoMesh library, U-96
autoPatch utility, U-89
autoRefineMesh utility, U-90
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-97, U-180
basicSolidThermo
library, U-99
basicThermophysicalModels
library, U-97
binary
keyword entry, U-112
BirdCarreau model, U-100
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-36, U-88, 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
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-95
cfx4ToFoam utility, U-89, U-154
changeDictionary utility, U-88
Charts window panel, U-168
checkMesh utility, U-89, U-155
chemFoam solver, U-86
chemistryModel
library, U-98
chemistryModel model, U-98
chemistrySolver model, U-98
chemkinToFoam utility, U-94
Choose Preset button, U-166
chtMultiRegionFoam solver, U-86
chtMultiRegionSimpleFoam solver, U-86
Chung
library, U-98
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
cmptAv
tensor member function, P-21
Co utility, U-91
coalChemistryFoam solver, U-86
coalCombustion
library, U-96
cofactors
tensor member function, P-21
coldEngineFoam solver, U-86
collapseEdges utility, U-90
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
combinePatchFaces utility, U-90
comments, U-76
commsType keyword, U-78
compressed
keyword entry, U-112
compressibleInterDyMFoam solver, U-85
compressibleInterFoam solver, U-85
compressibleLESModels
library, U-100
compressibleRASModels
library, U-99
constant directory, U-104, U-179
constLaminarFlameSpeed model, U-98
constTransport model, U-98, U-180
containers tools, U-95
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-96
`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
createBaffles utility, U-89
createPatch utility, U-89
createTurbulenceFields utility, U-92
cross product, *see* tensor, vector cross product
`CrossPowerLaw`
keyword entry, U-59
CrossPowerLaw model, U-100
cubeRootVolDelta model, U-99
`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-182
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-95
`ddt`
fvc member function, P-33
fvm member function, P-33
DeardorffDiffStress model, U-100
`debug` keyword, U-146
decomposePar utility, U-79, U-80, U-94
decomposeParDict
dictionary, U-79
decomposition
of field, U-79
of mesh, U-79
decompositionMethods
library, U-96
decompression of a tank, P-58
`defaultFieldValues` keyword, U-58
deformedGeom utility, U-89
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-60, 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-184
dieselMixture model, U-97, U-180
dieselSpray
library, U-96
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-95
dimensions keyword, U-20, U-108
dimensionSet class, P-21, P-28, P-29
dimensionSet tools, U-95
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-96
distributed keyword, U-81, U-82
distributionModels
library, U-96
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
dsmc
library, U-96
dsmcFieldsCalc utility, U-92
dsmcFoam solver, U-87
dsmcInitialise utility, U-88
dx
keyword entry, U-174
dynamicFvMesh
library, U-96
dynamicMesh
library, U-96
dynLagrangian model, U-100
dynMixedSmagorinsky model, U-100
dynOneEqEddy model, U-100
dynSmagorinsky model, U-100

E

eConstThermo model, U-98, U-179
edgeGrading keyword, U-140
edgeMesh
library, U-96
edges keyword, U-138
Edit menu, U-167, U-168
Edit Color Map button, U-165
egrMixture model, U-97, U-180
electrostaticFoam solver, U-87
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-96
engineCompRatio utility, U-92
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-91
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
ePsiThermo model, U-97, U-180
equationOfState keyword, U-181
equilibriumCO utility, U-94
equilibriumFlameT utility, U-94
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-92
expandDictionary utility, U-94
expansionRatio keyword, U-152
explicit
temporal discretisation, P-38
extrude2DMesh utility, U-88
extrudeMesh utility, U-88
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-95
fields, P-25
mapping, U-160
fields tools, U-95
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-59
file format, U-104
fileFormats
library, U-96
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-95
finiteVolume tools, U-95
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-89
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-91
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-32, U-92
foamCalcFunctions
library, U-95
foamCorrectVrt script/alias, U-158
foamDataToFluent utility, U-91, U-170
foamDebugSwitches utility, U-94
`FoamFile` keyword, U-105
`foamFile`
keyword entry, U-174
foamFormatConvert utility, U-94
foamInfoExec utility, U-94
foamJob script/alias, U-177
foamListTimes utility, U-92
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
forces
library, U-95
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
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-95
fvMatrix template class, P-29
fvMesh class, P-27
fvMesh tools, U-95
fvMotionSolvers
library, U-96
fvSchemes
dictionary, U-60, 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-96
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-95
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-95
graphFormat keyword, U-112
GuldersEGRLaminarFlameSpeed model, U-98
GuldersLaminarFlameSpeed model, U-98

H

hConstThermo model, U-98, U-179
Help menu, U-167
HerschelBulkley model, U-100
Hf keyword, U-181, U-182
hhuMixtureThermo model, U-97, U-180
hierarchical
keyword entry, U-80, U-81
highCpCoeffs keyword, U-182
homogenousDynSmagorinsky model, U-100
homogeneousMixture model, U-97, U-180
hPolynomialThermo model, U-98, U-179
hPsiMixtureThermo model, U-97, U-180
hPsiThermo model, U-97, U-180
hRhoMixtureThermo model, U-97, U-180
hRhoThermo model, U-97, U-180
hsPsiMixtureThermo model, U-97, U-180
hsPsiThermo model, U-97, U-180
hsRhoMixtureThermo model, U-97, U-180
hsRhoThermo model, U-97, U-180

I

I
tensor member function, P-21
icoFoam solver, U-17, U-21, U-22, U-24, U-84
icoPolynomial model, U-98, U-179
icoUncoupledKinematicParcelDyMFoam solver, U-86
icoUncoupledKinematicParcelFoam solver, U-87
ideasToFoam utility, U-154
ideasUnvToFoam utility, U-89
identities, *see* tensor, identities
identity, *see* tensor, identity
incompressibleLESModels
library, U-100
incompressibleRASModels
library, U-99
incompressibleTransportModels
library, P-50, U-100
incompressibleTurbulenceModels
library, P-50
index
notation, P-12, P-13
Information window panel, U-164
inhomogeneousMixture model, U-97, 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-89
interDyMFoam solver, U-85
interfaceProperties
library, U-100
interfaceProperties model, U-100
interFoam solver, U-85
interMixingFoam solver, U-85
internalField keyword, U-21, U-108
interPhaseChangeFoam solver, U-85
interpolation tools, U-95
interpolationScheme keyword, U-174
interpolations tools, U-95
interpolationSchemes keyword, U-114
inv
tensor member function, P-21
iterations
maximum, U-121

J

janafThermo model, U-98, U-179
jobControl
library, U-95
jplot
keyword entry, U-112, U-174

K

kEpsilon model, U-99
keyword
As, U-182
Cp, U-181

Cv, U-182
FoamFile, U-105
Hf, U-181, U-182
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-125
pRefValue, U-23, U-125
p_rhgRefCell, U-125
p_rhgRefValue, U-125
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-184
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-59
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-184
MGridGen, U-123
MUSCL, U-116
Newtonian, U-59
PBiCG, U-121
PCG, U-121
QUICK, U-119
RASModel, U-40, U-184

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-184
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
kOmega model, U-99
kOmegaSST model, U-99
kOmegaSSTSAS model, U-100
Kronecker delta, P-16

L

lagrangian
library, U-96
lagrangianIntermediate
library, U-96
Lambda2 utility, U-91
LamBremhorstKE model, U-99
laminar model, U-99
laminar
keyword entry, U-40, U-184
laminarFlameSpeedModels
library, U-98

laplaceFilter model, U-99
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-99
LaunderSharmaKE model, U-99
layers keyword, U-152
leastSquares
    keyword entry, U-117
leastSquares keyword, U-51
LESdeltas
    library, U-99
LESfilters
    library, U-99
LESModel
    keyword entry, U-40, U-184
LESModel keyword, U-184
LESProperties
    dictionary, U-184
levels keyword, U-150
libraries, U-67
library
    Chung, U-98
    LESdeltas, U-99
    LESfilters, U-99
    MGridGenGAMGAgglomeration, U-96
    ODE, U-96
    OSspecific, U-96
    OpenFOAM, U-95
    P1, U-98
    PV3FoamReader, U-163
    PVFoamReader, U-163
    SLGThermo, U-99
    Wallis, U-98
    autoMesh, U-96
    barotropicCompressibilityModels, U-98
    basicSolidThermo, U-99
    basicThermophysicalModels, U-97
    blockMesh, U-96
    chemistryModel, U-98
    coalCombustion, U-96
    compressibleLESModels, U-100
    compressibleRASModels, U-99
    conversion, U-96
    decompositionMethods, U-96
    dieselSpray, U-96
    distributionModels, U-96
    dsmc, U-96
    dynamicFvMesh, U-96
    dynamicMesh, U-96
    edgeMesh, U-96
    engine, U-96
    fieldFunctionObjects, U-95
    fileFormats, U-96
    finiteVolume, U-95
    foamCalcFunctions, U-95
    forces, U-95
    fvDOM, U-98
    fvMotionSolvers, U-96
    genericFvPatchField, U-96
    incompressibleLESModels, U-100
    incompressibleRASModels, U-99
    incompressibleTransportModels, P-50, U-100
    incompressibleTurbulenceModels, P-50
    interfaceProperties, U-100
    jobControl, U-95
    lagrangianIntermediate, U-96
    lagrangian, U-96
    laminarFlameSpeedModels, U-98
    linear, U-98
    liquidMixtureProperties, U-99
    liquidProperties, U-98
    meshTools, U-96
    molecularMeasurements, U-96
    molecule, U-96
    pairPatchAgglomeration, U-96
    postCalc, U-95
    potential, U-96
    primitive, P-19
    radiationModels, U-98
    randomProcesses, U-96
    reactionThermophysicalModels, U-97
    sampling, U-95
    solidMixtureProperties, U-99
    solidParticle, U-96
    solidProperties, U-99
    solid, U-99
    specie, U-98
    surfMesh, U-96
    surfaceFilmModels, U-101
    systemCall, U-96
    thermalPorousZone, U-99
    thermophysicalFunctions, U-98
    thermophysical, U-179
    topoChangerFvMesh, U-96
    triSurface, U-96
    twoPhaseInterfaceProperties, U-100
    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-99
LienCubicKELowRe model, U-99
LienLeschzinerLowRe model, U-99
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-98
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-98
lists, P-25
List<Type> template class, P-25
localEuler
    keyword entry, U-119
location keyword, U-105
locationInMesh keyword, U-148, U-149
locDynOneEqEddy model, U-100
lowCpCoeffs keyword, U-182
lowReOneEqEddy model, U-100
LRDDiffStress model, U-100
LRR model, U-99
lsGrad
    fvc member function, P-33
LTSInterFoam solver, U-85
LTSReactingParcelFoam solver, U-87

M

Mach utility, U-91
mag
    tensor member function, P-21
magneticFoam solver, U-87
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-88, U-160
mapping
    fields, U-160
Marker Style menu, U-34
matrices tools, U-95
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-100
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-88
mechanicalProperties
    dictionary, U-49
memory tools, U-95
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-33
    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-95
meshQualityControls keyword, U-146
meshTools
    library, U-96
message passing interface
    openMPI, U-81
method keyword, U-81
metis
    keyword entry, U-81
MGridGenGAMGAgglomeration
    library, U-96
MGridGen
    keyword entry, U-123
mhdFoam solver, P-65, U-87
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-100
mixtureAdiabaticFlameT utility, U-94
mode keyword, U-149
model
    APIfunctions, U-98
    BirdCarreau, U-100
    CrossPowerLaw, U-100
    DeardorffDiffStress, U-100
    GuldersEGRLaminarFlameSpeed, U-98
    GuldersLaminarFlameSpeed, U-98
    HerschelBulkley, U-100
    LRDDiffStress, U-100
    LRR, U-99
    LamBremhorstKE, U-99
    LaunderGibsonRSTM, U-99
    LaunderSharmaKE, U-99
    LienCubicKELowRe, U-99
    LienCubicKE, U-99
    LienLeschzinerLowRe, U-99
    NSRDSfunctions, U-98
    Newtonian, U-100
    NonlinearKEShih, U-99
    PrandtlDelta, U-99
    RNGkEpsilon, U-99
    Smagorinsky2, U-100
    Smagorinsky, U-100
    SpalartAllmarasDDES, U-100
    SpalartAllmarasIDDES, U-100
    SpalartAllmaras, U-99, U-100
    anisotropicFilter, U-99
    basicMultiComponentMixture, U-97, U-180
    chemistryModel, U-98
    chemistrySolver, U-98
    constLaminarFlameSpeed, U-98
    constTransport, U-98, U-180
    cubeRootVolDelta, U-99
    dieselMixture, U-97, U-180
    distributed, U-96
    dynLagrangian, U-100
    dynMixedSmagorinsky, U-100
    dynOneEqEddy, U-100
    dynSmagorinsky, U-100
    eConstThermo, U-98, U-179
    ePsiThermo, U-97, U-180
    egrMixture, U-97, U-180
    hConstThermo, U-98, U-179
    hPolynomialThermo, U-98, U-179

hPsiMixtureThermo, U-97, U-180
hPsiThermo, U-97, U-180
hRhoMixtureThermo, U-97, U-180
hRhoThermo, U-97, U-180
hhuMixtureThermo, U-97, U-180
homogenousDynSmagorinsky, U-100
homogeneousMixture, U-97, U-180
hsPsiMixtureThermo, U-97, U-180
hsPsiThermo, U-97, U-180
hsRhoMixtureThermo, U-97, U-180
hsRhoThermo, U-97, U-180
icoPolynomial, U-98, U-179
inhomogeneousMixture, U-97, U-180
interfaceProperties, U-100
janafThermo, U-98, U-179
kEpsilon, U-99
kOmegaSSTSAS, U-100
kOmegaSST, U-99
kOmega, U-99
laminar, U-99
laplaceFilter, U-99
locDynOneEqEddy, U-100
lowReOneEqEddy, U-100
maxDeltaxyz, U-100
mixedSmagorinsky, U-100
multiComponentMixture, U-97, U-180
oneEqEddy, U-100
perfectGas, U-98, U-179
polynomialTransport, U-98, U-180
powerLaw, U-100
ptsotchDecomp, U-97
pureMixture, U-97, U-180
qZeta, U-99
reactingMixture, U-97, U-180
realizableKE, U-99
reconstruct, U-96
scaleSimilarity, U-100
scotchDecomp, U-97
simpleFilter, U-99
smoothDelta, U-100
specieThermo, U-98, U-180
spectEddyVisc, U-100
sutherlandTransport, U-98, U-180
veryInhomogeneousMixture, U-97, U-180
modifyMesh utility, U-90
molecularMeasurements
library, U-96
molecule
library, U-96
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-97, U-180
multigrid
geometric-algebraic, U-122
multiphaseEulerFoam solver, U-85
multiphaseInterFoam solver, U-85
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-59
Newtonian model, U-100
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-99
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-98
null
keyword entry, U-174
numberOfSubdomains keyword, U-81

O

object keyword, U-105
objToVTK utility, U-90
ODE
library, U-96
oneEqEddy model, U-100
Opacity text box, U-167
OpenFOAM
applications, U-67
file format, U-104
libraries, U-67
OpenFOAM
library, U-95
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
OSspecific
library, U-96
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-125
p_rhgRefValue keyword, U-125
pairPatchAgglomeration
library, U-96
paraFoam, U-23, U-163
parallel
running, U-79
partialSlip
boundary condition, U-137
particleTracks utility, U-92
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-94
PBiCG
keyword entry, U-121
PCG
keyword entry, U-121
pdfPlot utility, U-92
PDRFoam solver, U-86
PDRMesh utility, U-90
Pe utility, U-91
perfectGas model, U-98, 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-33
plot3dToFoam utility, U-89
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-98, U-180
polyPatch class, P-27
polyPatchList class, P-27
polySpline
keyword entry, U-139
porousInterFoam solver, U-85
porousSimpleFoam solver, U-84
post-processing, U-163
post-processing
paraFoam, U-163
postCalc
library, U-95
postChannel utility, U-92

potential
library, U-96
potentialFoam solver, P-42, U-83
potentialFreeSurfaceFoam solver, U-84
pow
tensor member function, P-21
powerLaw model, U-100
pPrime2 utility, U-91
Pr keyword, U-182
PrandtlDelta model, U-99
preconditioner keyword, U-121, U-122
pRefCell keyword, U-23, U-125
pRefValue keyword, U-23, U-125
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-95
printCoeffs keyword, U-41, U-184
processorWeights keyword, U-80
probeLocations utility, U-92
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
ptot utility, U-92
ptsotchDecomp model, U-97
pureMixture model, U-97, U-180
purgeWrite keyword, U-112
PV3FoamReader
library, U-163
PVFoamReader
library, U-163

Q

Q utility, U-91
QUICK
keyword entry, U-119
qZeta model, U-99

R

R utility, U-92
radiationModels
library, U-98
randomProcesses
library, U-96
RASModel
keyword entry, U-40, U-184
RASModel keyword, U-184
raw
keyword entry, U-112, U-174
reactingFoam solver, U-86
reactingMixture model, U-97, U-180
reactingParcelFilmFoam solver, U-87
reactingParcelFoam solver, U-87
reactionThermophysicalModels
library, U-97
realizableKE model, U-99
reconstruct model, U-96
reconstructPar utility, U-83, U-94
reconstructParMesh utility, U-94
refGradient keyword, U-136
refineHexMesh utility, U-90
refinementRegions keyword, U-149
refinementLevel utility, U-90
refinementRegions keyword, U-148, U-150
refinementSurfaces keyword, U-148
refineMesh utility, U-90
refineWallLayer utility, U-90
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-90
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
rhoCentralDyMFoam solver, U-84
rhoCentralFoam solver, U-84
rhoLTSPimpleFoam solver, U-84
rhoPimpleFoam solver, U-84
rhoPimplecFoam solver, U-84
rhoReactingFoam solver, U-86
rhoSimpleFoam solver, U-84
rhoSimplecFoam solver, U-84
rmdepall script/alias, U-74

RNGkEpsilon model, U-99
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-89
sample utility, U-92, U-173
sampling
    library, U-95
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-100
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-90
Set Ambient Color button, U-166
setFields utility, U-58, U-88
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
simple
    keyword entry, U-80, U-81
simpleFilter model, U-99
simpleFoam solver, P-50, U-84
simpleGrading keyword, U-140
simpleReactingParcelFoam solver, U-87
simpleSpline
    keyword entry, U-139
simulationType keyword, U-40, U-59, U-184
singleCellMesh utility, U-90
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-100
Smagorinsky2 model, U-100
smapToFoam utility, U-91
smoothDelta model, U-100
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
    library, U-99
Solid Color

menu entry, U-166
solidDisplacementFoam solver, U-87
solidDisplacementFoam solver, U-50
solidEquilibriumDisplacementFoam solver, U-87
solidMixtureProperties
library, U-99
solidParticle
library, U-96
solidProperties
library, U-99
solver
LTSInterFoam, U-85
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-86
chtMultiRegionSimpleFoam, U-86
coalChemistryFoam, U-86
coldEngineFoam, U-86
compressibleInterDyMFoam, U-85
compressibleInterFoam, U-85
dnsFoam, U-86
dsmcFoam, U-87
electrostaticFoam, U-87
engineFoam, U-86
financialFoam, U-88
fireFoam, U-86
icoFoam, U-17, U-21, U-22, U-24, U-84
icoUncoupledKinematicParcelDyMFoam, U-86
icoUncoupledKinematicParcelFoam, U-87
interDyMFoam, U-85
interFoam, U-85
interMixingFoam, U-85
interPhaseChangeFoam, U-85
laplacianFoam, U-83
magneticFoam, U-87
mdEquilibrationFoam, U-87
mdFoam, U-87
mhdFoam, P-65, U-87
multiphaseEulerFoam, U-85
multiphaseInterFoam, U-85
nonNewtonianIcoFoam, U-84
pimpleDyMFoam, U-84
pimpleFoam, U-84
pisoFoam, U-17, U-84
porousInterFoam, U-85
porousSimpleFoam, U-84
potentialFoam, P-42, U-83
potentialFreeSurfaceFoam, U-84
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
scalarTransportFoam, U-83
settlingFoam, U-86
shallowWaterFoam, U-84
simpleFoam, P-50, U-84
simpleReactingParcelFoam, U-87
solidDisplacementFoam, U-87
solidDisplacementFoam, U-50
solidEquilibriumDisplacementFoam, U-87
sonicDyMFoam, U-84
sonicFoam, P-56, U-85
sonicLiquidFoam, P-59, U-85
sprayEngineFoam, U-87
sprayFoam, 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-85
sonicLiquidFoam solver, P-59, U-85
source, P-33
SpalartAllmaras model, U-99, U-100
SpalartAllmarasDDES model, U-100
SpalartAllmarasIDDES model, U-100
specie
library, U-98
`specie` keyword, U-181
specieThermo model, U-98, U-180
spectEddyVisc model, U-100

spline keyword, U-138
splitCells utility, U-91
splitMesh utility, U-90
splitMeshRegions utility, U-90
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-89
star4ToFoam utility, U-89
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-92
steadyState
    keyword entry, U-119
Stereolithography (STL), U-145
stitchMesh utility, U-90
stl
    keyword entry, U-174
stopAt keyword, U-111
strategy keyword, U-80, U-81
streamFunction utility, U-91
stress analysis of plate with hole, U-45
stressComponents utility, U-91
Style window panel, U-23, U-166
Su
    fvm member function, P-33
subsetMesh utility, U-90
summation convention, P-13
SUPERBEE differencing, P-34
supersonic flow, P-55
supersonic flow over forward step, P-54
supersonicFreeStream
    boundary condition, U-137
surface mesh, U-145
surfaceAdd utility, U-93
surfaceAutoPatch 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-101
surfaceFind utility, U-93
surfaceFormat keyword, U-174
surfaceInertia utility, U-93
surfaceMesh tools, U-95
surfaceMeshConvert utility, U-93
surfaceMeshConvertTesting utility, U-93
surfaceMeshExport utility, U-93
surfaceMeshImport utility, U-93
surfaceMeshInfo utility, U-93
surfaceMeshTriangulate utility, U-93
surfaceNormalFixedValue
    boundary condition, U-137
surfaceOrient utility, U-93
surfacePointMerge utility, U-93
surfaceRedistributePar utility, U-93
surfaceRefineRedGreen utility, U-93
surfaces keyword, U-174
surfaceSmooth utility, U-93
surfaceSplitByPatch utility, U-93
surfaceSplitNonManifolds utility, U-94
surfaceSubset utility, U-94
surfaceToPatch utility, U-94
surfaceTransformPoints utility, U-94
surfMesh
    library, U-96
SuSp
    fvm member function, P-33
sutherlandTransport model, U-98, 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
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
*n*th 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-89
text box
Opacity, U-167
thermalPorousZone
library, U-99
thermalProperties
dictionary, U-50
thermodynamics keyword, U-181
thermophysical
library, U-179
thermophysicalFunctions
library, U-98
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-95
cfdTools, U-95
containers, U-95
db, U-95
dimensionSet, U-95
dimensionedTypes, U-95
fields, U-95
finiteVolume, U-95
fvMatrices, U-95
fvMesh, U-95
global, U-95
graph, U-95
interpolations, U-95
interpolation, U-95
matrices, U-95
memory, U-95
meshes, U-95
primitives, U-95
surfaceMesh, U-95
volMesh, U-95
topoChangerFvMesh
library, U-96
topoSet utility, U-90
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-90
transport keyword, U-181
transportProperties
dictionary, U-21, U-38, U-41
transportProperties file, U-59
triple inner product, P-15
triSurface
library, U-96
Ts keyword, U-182
turbulence
dissipation, U-39
kinetic energy, U-39
length scale, U-40
turbulence keyword, U-184
turbulence model
RAS, U-39
turbulenceProperties
dictionary, U-40, U-59, U-184
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
twoPhaseInterfaceProperties
library, U-100
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
Système International, U-107
United States Customary System, U-107
USCS, U-107
Update GUI button, U-165
uprime utility, U-91
upwind
keyword entry, U-116, U-119
upwind differencing, P-34, U-60
USCS units, U-107
Use Parallel Projection button, U-24
Use parallel projection button, U-167
utility

Co, U-91
Lambda2, U-91
Mach, U-91
PDRMesh, U-90
Pe, U-91
Q, U-91
R, U-92
Ucomponents, P-66
adiabaticFlameT, U-94
ansysToFoam, U-89
applyBoundaryLayer, U-88
applyWallFunctionBoundaryConditions, U-88
attachMesh, U-89
autoPatch, U-89
autoRefineMesh, U-90
blockMesh, U-36, U-88, U-136
boxTurb, U-88
cfx4ToFoam, U-89, U-154
changeDictionary, U-88
checkMesh, U-89, U-155
chemkinToFoam, U-94
collapseEdges, U-90
combinePatchFaces, U-90
createBaffles, U-89
createPatch, U-89
createTurbulenceFields, U-92
datToFoam, U-89
decomposePar, U-79, U-80, U-94
deformedGeom, U-89
dsmcFieldsCalc, U-92
dsmcInitialise, U-88
engineCompRatio, U-92
engineSwirl, U-88
ensight74FoamExec, U-172
ensightFoamReader, U-91
enstrophy, U-91
equilibriumCO, U-94
equilibriumFlameT, U-94
execFlowFunctionObjects, U-92
expandDictionary, U-94
extrude2DMesh, U-88
extrudeMesh, U-88
extrudeToRegionMesh, U-89
faceAgglomerate, U-88
flattenMesh, U-89
flowType, U-91
fluent3DMeshToFoam, U-89
fluentMeshToFoam, U-89, U-154
foamCalc, U-32, U-92
foamDataToFluent, U-91, U-170
foamDebugSwitches, U-94
foamFormatConvert, U-94
foamInfoExec, U-94
foamListTimes, U-92
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
gambitToFoam, U-89, U-154
gmshToFoam, U-89
ideasToFoam, U-154
ideasUnvToFoam, U-89
insideCells, U-89
kivaToFoam, U-89
mapFields, U-30, U-37, U-41, U-54, U-88, U-160
mdInitialise, U-88
mergeMeshes, U-90
mergeOrSplitBaffles, U-90
mirrorMesh, U-90
mixtureAdiabaticFlameT, U-94
modifyMesh, U-90
moveDynamicMesh, U-90
moveEngineMesh, U-90
moveMesh, U-90
mshToFoam, U-89
netgenNeutralToFoam, U-89
objToVTK, U-90
pPrime2, U-91
particleTracks, U-92
patchAverage, U-92
patchIntegrate, U-92
patchSummary, U-94
pdfPlot, U-92
plot3dToFoam, U-89
polyDualMesh, U-90
postChannel, U-92
probeLocations, U-92
ptot, U-92
reconstructParMesh, U-94
reconstructPar, U-83, U-94
refineHexMesh, U-90
refineMesh, U-90
refineWallLayer, U-90
refinementLevel, U-90
removeFaces, U-90
renumberMesh, U-90
rotateMesh, U-90
sammToFoam, U-89
sample, U-92, U-173
scalePoints, U-157

selectCells, U-90
setFields, U-58, U-88
setSet, U-90
setsToZones, U-90
singleCellMesh, U-90
smapToFoam, U-91
snappyHexMesh, U-89, U-145
splitCells, U-91
splitMeshRegions, U-90
splitMesh, U-90
star3ToFoam, U-89
star4ToFoam, U-89
starToFoam, U-154
steadyParticleTracks, U-92
stitchMesh, U-90
streamFunction, U-91
stressComponents, U-91
subsetMesh, U-90
surfaceAdd, U-93
surfaceAutoPatch, 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
surfaceInertia, U-93
surfaceMeshConvertTesting, U-93
surfaceMeshConvert, U-93
surfaceMeshExport, U-93
surfaceMeshImport, U-93
surfaceMeshInfo, U-93
surfaceMeshTriangulate, U-93
surfaceOrient, U-93
surfacePointMerge, U-93
surfaceRedistributePar, U-93
surfaceRefineRedGreen, U-93
surfaceSmooth, U-93
surfaceSplitByPatch, U-93
surfaceSplitNonManifolds, U-94
surfaceSubset, U-94
surfaceToPatch, U-94
surfaceTransformPoints, U-94
tetgenToFoam, U-89
topoSet, U-90
transformPoints, U-90
uprime, U-91
viewFactorsGen, U-88
vorticity, U-91
wallFunctionTable, U-88
wallGradU, U-91
wallHeatFlux, U-91
wallShearStress, U-92
wdot, U-92
writeCellCentres, U-92
writeMeshObj, U-89
yPlusLES, U-92
yPlusRAS, U-92
zipUpMesh, U-90
utilityFunctionObjects
library, U-96

V

`value` keyword, U-21, U-136
`valueFraction` keyword, U-136
van Leer differencing, P-34
`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-97, 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-88
viscosity
kinematic, U-21, U-41
volField<Type> template class, P-29
volMesh tools, U-95
vorticity utility, U-91
`vtk`
keyword entry, U-174
vtkFoam
library, U-163
vtkPV3Foam
library, U-163

W

wall
boundary condition, P-59, P-65, U-57, U-134
`wall`
keyword entry, U-135
wallFunctionTable utility, U-88
wallGradU utility, U-91

wallHeatFlux utility, U-91
Wallis
    library, U-98
wallShearStress utility, U-92
wclean script/alias, U-73
wdot utility, U-92
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-92
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-89
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-90