HopeFOAM

(High Order Parallel Extensible CFD Software)

Programmer's Guide

Version 0.1

The Exercise Group
Innovation Institute for Defence Science and Technology,
the Academy of Military Science (AMS), China.
September 2017

contents

License

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

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

1. Definitions

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

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

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

d. **"Licensor"** means the individual, individuals, entity or entities that offer(s) the Work under the terms of this License.

e. **"Original Author"** means, in the case of a literary or artistic work, the individual, individuals, entity or entities who created the Work or if no individual or entity can be identified, the publisher; and in addition (i) in the case of a performance the actors,

singers, musicians, dancers, and other persons who act, sing, deliver, declaim, play in, interpret or otherwise perform literary or artistic works or expressions of folklore; (ii) in the case of a phonogram the producer being the person or legal entity who first fixes the sounds of a performance or other sounds; and, (iii) in the case of broadcasts, the organization that transmits the broadcast.

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

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

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

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

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

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

a. to Reproduce the Work, to incorporate the Work into one or more Collections, and to Reproduce the Work as incorporated in the Collections; and,

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

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

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

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

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

c. If You Distribute, or Publicly Perform the Work or Collections, You must, unless a request has been made pursuant to Section 4(a), keep intact all copyright notices for the Work and provide, reasonable to the medium or means You are utilizing: (i) the name of the Original Author (or pseudonym, if applicable) if supplied, and/or if the Original Author and/or Licensor designate another party or parties (e.g., a sponsor

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

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

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

    iii. **Voluntary License Schemes**. The Licensor reserves the right to collect royalties, whether individually or, in the event that the Licensor is a member of a collecting society that administers voluntary licensing schemes, via that society, from any exercise by You of the rights granted under this License that is for a purpose or use which is otherwise than noncommercial as permitted under Section 4(b).

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

5. Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, 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. This License may not be modified without the mutual written agreement of the Licensor and You.

e. The rights granted under, and the subject matter referenced, in this License were drafted utilizing the terminology of the Berne Convention for the Protection of Literary and Artistic Works (as amended on September 28, 1979), the Rome Convention of 1961, the WIPO Copyright Treaty of 1996, the WIPO Performances and Phonograms Treaty of 1996 and the Universal Copyright Convention (as revised on July 24, 1971). These rights and subject matter take effect in the relevant jurisdiction in which the License terms are sought to be enforced according to the corresponding provisions of the implementation of those treaty provisions in the applicable national law. If the standard suite of rights granted under applicable copyright law includes additional rights not granted under this License, such additional rights are deemed to be included in the License; this License is not intended to restrict the license of any rights under applicable law.

About HopeFOAM

HopeFOAM is a major extension of OpenFOAM to provide higher order finite element method and other numerical methods for computational mechanics. It is developed by Exercise Group, Innovation Institute for Defense Science and Technology, the Academy of Military Science (AMS), China. The Group aims at developing open source software packages for large scale computational science and engineering. HopeFOAM has following features:

- **High Order**：In addition to the finite volume discretization method used in OpenFOAM, HopeFOAM aims at integrating high-order discretization methods into the computational mechanics Toolbox, among which DGM（Discontinuous Galerkin Method）is the first one.
- **Parallel**：In order to improve the performance and scalability of parallel computing, parallel computational toolkits/software are integrated into HopeFOAM to accelerate the discretization and computational procedures.
- **Extensible**：By incorporating with the high order discretization and efficient parallel computing, HopeFOAM provides an extensible software framework for further development of application module and easy-to-use interfaces for developers.
- **FOAM**：The current version of HopeFOAM is a major extension of the OpenFOAM-4.0 released by the OpenFOAM Foundation on the 28th of June, 2016.

HopeFOAM-0.1 is the first publicly released version of HopeFOAM and developed by a major extension of OpenFOAM-4.0. The well-known high-order discretization method, Discontinuous Galerkin Method (DGM) is implemented in HopeFOAM-0.x. There are copious references about the method in Hesthaven and Warbuton's book:【Hesthaven J S, Warburton T. Nodal discontinuous Galerkin methods: algorithms, analysis, and applications[M]. Springer Science & Business Media, 2007.】

HopeFOAM-0.1 provides 2D-DGM and related support. The major components include data structure, DGM discretization, solvers and related tools. PETSc is used for solving of linear systems of equations. 3D applications will be supported in a new release in a few months.

The guiding principle in the development of HopeFOAM-0.1 is to reuse the primitive

data structure of OpenFOAM-4.0 as much as possible and keep it consistent with the user interfaces. Thus, users of OpenFOAM could implement and adopt corresponding high order DGM solvers in a relatively straightforward way. The original OpenFOAM applications can be use normally.

This manual is the programmer’s guide of version 0.1 of the High Order Parallel Extensible Open Source CFD software package (HopeFOAM) .

1 Code structure

1.1 Software architecture

HopeFOAM-0.1 is developed on the basis of the original finite volume method (FVM) framework of the OpenFOAM-4.0 as shown in figure 1.1.

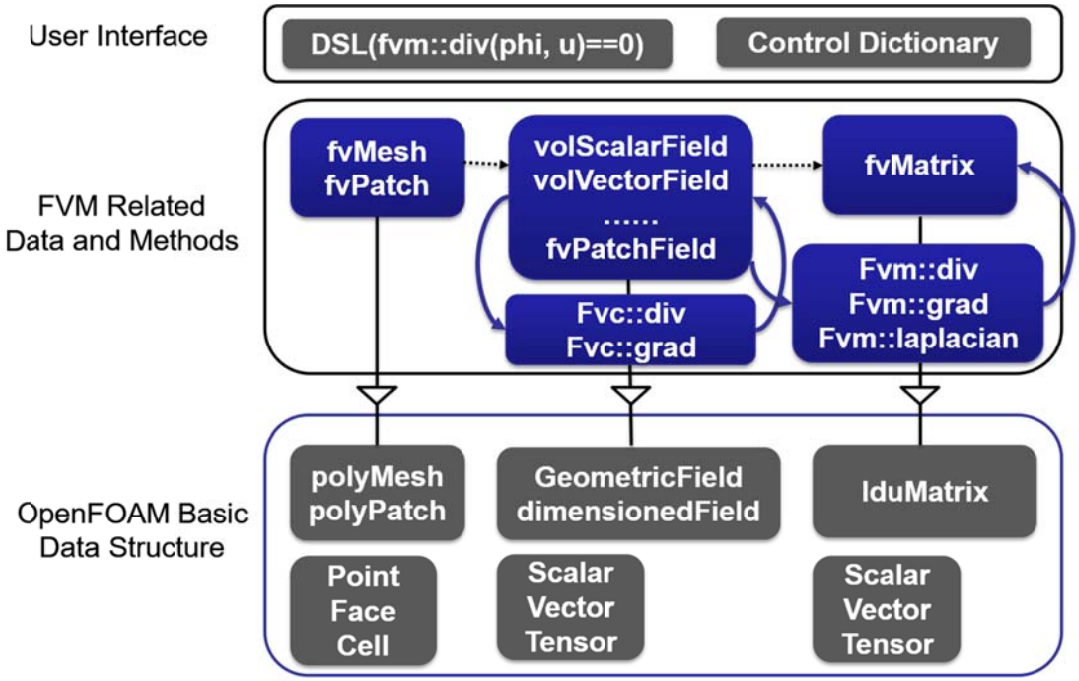

Fig1.1 The origin FVM framework of OpenFOAM-4.0

The above framework can be mainly divided into three layers from bottom to top:

- Basic Data Structure of OpenFOAM: The code of this layer mainly implements the underlying data structure of CFD calculation, including mesh, field data, linear algebra system. The main purpose is to support the implementation of universally applicable numerical discretization methods. Basic data structure is intended to be independent of specific discrete methods. But in practice, its implementation is more or less tend to match FVM discretization.
- FVM Related Data and Methods: FVM related data and methods are developed based on low-level OpenFOAM basic data structure, accordingly implements data structure of mesh, field data, linear algebra system. In addition, it develops the FVM discrete operator.

- User Interface: This layer provides the CFD users with an interface support DSL language features.

The software architecture of HopeFOAM-0.1 is a major extension of original framework of OpenFOAM, as is shown in figure 1.2. Compared with the OpenFOAM-4.0, this framework includes an additional Discontinuous Galerkin(DG) module. The design of the software architecture follows these principles:

- Reuse the framework and data structure
- Keep consistent with the user interface
- Inherit the pre and post processing tools
- Add a separate DG source package
- High order DG discretization
- High performance
- High extensibility, runtime selection

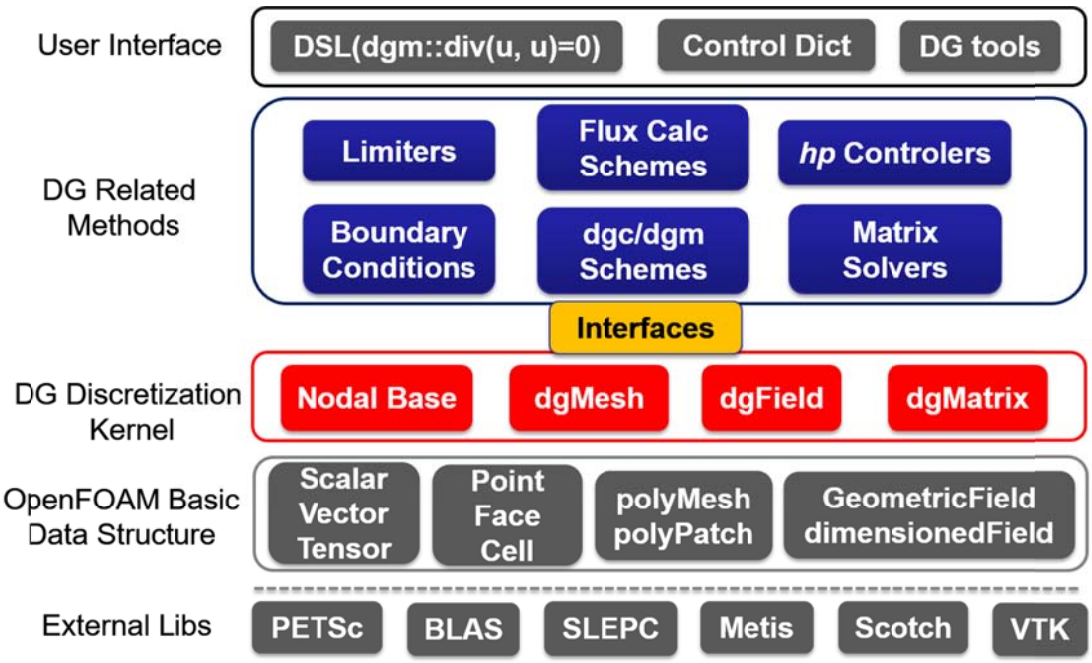

Fig1.2 The Framework of HopeFOAM-0.1

HopeFOAM keeps the original module of OpenFOAM intact while adding a large amount of DG modules:

- External Libs: To improve the computational performance and create an integrated software ecosystem, a large amount of third-party libraries are imported. PETSc, BLAS, and SLEPC are used for efficient parallel solution of

Algebraic Linear Systems. Metis and Scotch are used to support mesh decomposition in parallel. VTK is used for data visualization.

- Basic data structure of OpenFOAM: The code of this layer is preserved and reused.
- DG Discretization Kernel: This is one instance of the core implementation of DG method in HopeFOAM-0.1 , which mainly provides data structure such as basis functions, mesh, field data, and linear algebra system for the DG method.
- DG Related Data and Methods: This module mainly implements the boundary conditions, explicit/implicit discrete operator, hp-adaptive, flux calculation and limiters of DG method etc.
- User Interface：This module mainly provides the DG calculation interface for the CFD users. In addition to conventional differential operators, DG controller, which is used for solver parameter configuration, and the DG tools are added.

1.2 Tree structure of code directory

The main code structure and the corresponding functions of the DG method in HopeFOAM-0.1 are as follows:

```
|-- HopeFOAM-0.1
    |-- src
        --|-- DG: All DG modules
            |    |-- convectionSchemes : convection Schemes
            |    |    |-- convectionScheme: Abstract base for Convection Scheme
            |    |    `-- defaultConvectionScheme: default Convection Scheme
            |    |-- ddtSchemes: ddt Schemes
            |    |    |-- EulerDdtScheme: Euler Ddt Scheme
            |    |    `-- ddtScheme: Abstract base for ddt Scheme
            |    |-- dg: Namespace for Discontinuous Galerkin Method
            |    |-- dgSchemes: Selector class for DG Method differencing schemes
            |    |-- dgSolution: Selector class for DG solution solution
            |    |-- dgc: Namespace of functions to calculate explicit derivatives
            |    |-- dgm: Namespace of functions to implicit derivatives returning a matrix
            |    |-- divSchemes: class for div schemes
            |    |    |-- defaultDivScheme: Basic second-order div using face-gradients
            |    |    `-- divScheme: Abstract base class for div schemes
            |    |-- godunovFlux: Generic Godunov flux class
            |    |    |-- fluxSchemes: Abstract base class for flux Calculation schemes
            |    |    |    `-- scheme:
```

```
|   |   |       `-- RoeFlux: Roe flux scheme class
|   |   `-- limiteSchemes: Abstract base class for limiter
|   |       `-- scheme
|   |           `-- Trianglelimite: limite is only support triangle mesh
|   |-- gradSchemes: class for grad Schemes
|   |   |-- defaultGrad: class for default grad Schemes
|   |   `-- gradScheme: Abstract base class for gradient schemes
|   |-- laplacianSchemes: Basic second-order laplacian
|   |   |-- defaultLaplacianScheme: second-order laplacian using face-gradients
|   |   `-- laplacianScheme: Abstract base class for laplacian schemes
|   `-- simpleFlux: flux calculation schemes
|       |-- fluxCalcScheme: Abstract base class for flux calculation schemes
|       `-- schemes
|           |-- LFFlux: LF flux scheme class
|           |-- averageFlux: average flux scheme class
|           `-- noneFlux: zero flux scheme class
|-- Equation: composite design pattern to construct an equation
|-- Make
|-- cfdtools: include files for solvers
|   `-- general
|       `-- include
|-- dgMatrices: linear algebraic system
|   |-- denseMatrix: small dense matrix for the calculation of base functions
|   |-- dgLduMatrix: a general matrix class stored with LDU format
|   |-- dgMatrix: A special matrix type and solver
|   `-- dgScalarMatrix: A scalar instance of dgMatrix
|-- dgMesh: Mesh data for DG
|   |-- dgBoundaryMesh: Boundary Mesh for DG
|   |-- dgPatches: patch classes for DG
|   |   |-- basic: basic patchs classes for DG
|   |   |   |-- coupled: An abstract base class for patches that couple regions
|   |   |   `-- generic: DG variant of the genericPolyPatch
|   |   |-- constraint:
|   |   |   |-- arc: A arc patch
|   |   |   |-- empty: A patch which will not exist in the dgMesh
|   |   |   `-- processor: Processor patch
|   |   |-- derived
|   |   |   `-- wall: wall patch
|   |   `-- dgPatch: A DG patch using a polyPatch and a dgBoundaryMesh
|   |-- dgTree: A template class, the information is organized into a tree structure
|   `-- polyPatchs:
|       `-- constraint
```

```
|                 `-- arc: arc front and back plane patch
|-- element
|   |-- baseFunctions
|   |   |-- baseFunction: Abstract class for Legendre base Function.
|   |   `-- straightBaseFunctions
|   |       |-- lineBaseFunction: standard Line Element
|   |       |-- quadrilateralBaseFunction:  standard quadrilateral Element
|   |       |-- tetrahedralBaseFunction：standard Tetrahedral Element
|   |       `-- triangleBaseFunction：  standard Triangle Element
|   |-- dofAddressings
|   |   |-- dgDofAddressing：Addressing rule for dof in dg method.
|   |   `-- dofAddressing：Abstract class indicating the addressing rule for dof.
|   |-- gaussIntegration
|   |   |-- gaussIntegration：Abstract class for gaussIntegration.
|   |   |-- gaussLineIntegration：gauss interpolation for Line Element
|   |   |-- gaussQuadrilateralIntegration：  gaussQuadrilateralIntegration
|   |   |-- gaussTetrahedralIntegration:  standard Tetrahedral Element
|   |   `-- gaussTriangleIntegration: standard Triangle Element
|   |-- physicalElementData: Contains the information for critical cell discretisation
|   |-- polynomials:
|   |   `-- Legendre: Abstract base class for standard Element.
|   |-- stdElement: including baseFunction and gauss integration Method
|   `-- stdElementSets: Hash Table from names to all the stdElemets
`-- fields: fields data for DG
    |-- GeometricDofField: GeometricField with Dof Addresing supported
    |   |-- GeometricDofSphericalTensorField: Spherical Tensor specific part
    |   |-- GeometricDofSymmTensorField: SymmTensor specific part
    |   `-- GeometricDofTensorField: Tensor specific part
    |-- dgFields: fields data for DG
    |-- dgGaussField: Field to store the data intepolated from GeometricDofField with gauss base
    `-- dgPatchFields: classes for DG Patch Fields
        |-- basic
        |   |-- calculated: calculated DgPatch Field
        |   |-- coupled: Abstract base class for coupled patches.
        |   |-- fixedGradient: supplies a fixed gradient condition
        |   |-- fixedValue: supplies a fixed value condition
        |   |-- mixed: class for 'mixed' type boundary
        |   |-- reflective: reflective boundary condition
        |   `-- zeroGradient: zero-gradient condition
        |-- constraint: classes for constraint DG Patch Fields
        |       |-- cyclic: enforces a cyclic condition between a pair of boundaries.
```

```
|   |-- empty:
|   |-- processor: processor communication across patches
|   `-- slip: the vector is symmetry to the normal of the patch
|-- derived: classes for derived DG Patch Fields
|   |-- freestream: free-stream condition
|   |-- freestreamPressure: free-stream condition for pressure
|   |-- inletOutlet: generic outflow condition
|   `-- noSlip: fixes the velocity to zero at walls
`-- dgPatchField
```

2 Key data structure

2.1 Mesh: dgMesh

DG discrete method is based on the mesh data structure dgMesh. Figure 2.1 shows the referencing relationship of mesh data structure. The mesh geometry information is stored in dgPolyMesh. The information of configuration files with the same name in case directory are stored in dgScheme and dgSolution. The basis functions and coefficient matrices required for DG discretization are stored in DG Discret physicalElementData.

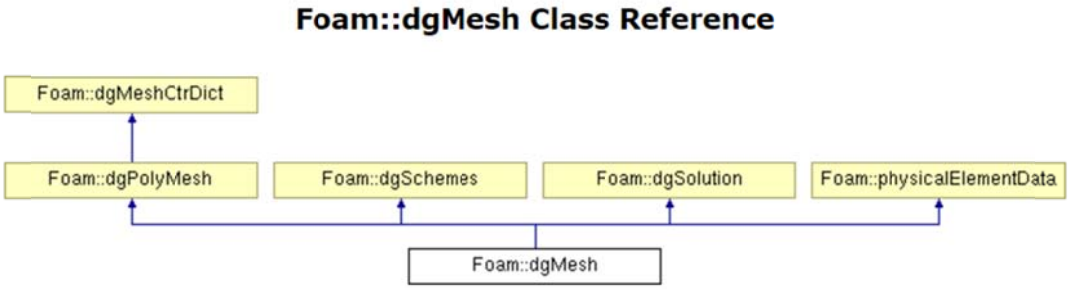

Fig2.1 Class diagram of mesh data structure

dgMeshCtrDict & dgPolyMesh: dgMeshCtrDict and dgPolyMesh read the configuration from file dgSolution in the case directory. The mesh spatial dimensions and the discretization order are defined in *dgSolution*. dgPolyMesh, which is inherited from the base mesh data structure polyMesh in OpenFOAM, is combined with spatial dimension to create DG discretization tree structure mesh. The mesh information of polyMesh is shown in figure 2.2. Mesh information of dgpolyMesh is constructed into dgTree <Type> data structure as shown in figure2.3. Following this pattern, Cells and Faces in dgPolyMesh are constructed into this form, corresponding to member variables cellTree_ and faceTree_ , respectively. dgTree <Type> is an array of pointers, and each pointer points to a dgTreeUnit <Type> object. The mesh information is stored by loading Cell and Face into dgTreeUnit <Type>.

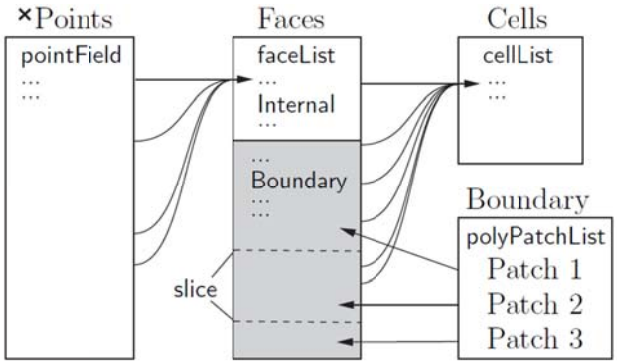

Fig2.2 Mesh Geometry Information of polyMesh

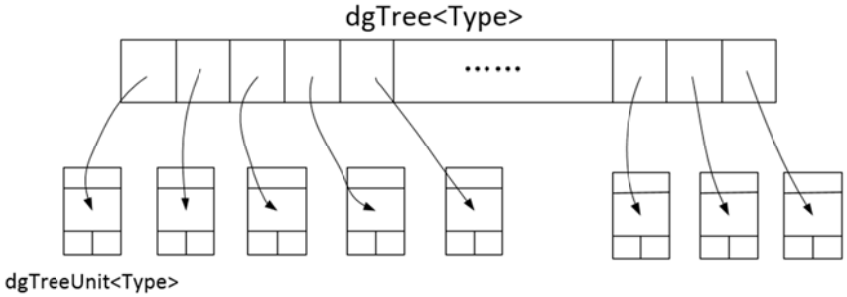

Fig2.3 DG mesh tree structure

dgScheme & dgSolution: dgSolution is used to select DG solution methods, while dgScheme completes the selection of DG discretization methods and discrete operators. As a subclass, dgMesh guarantees that all the field operations is able to obtain relevant information.

physicalElementData: the numerical values of DG discretization are stored in physicalElementData, and detailed information will be described in Section 2.3.

2.2 Field data structure: GeometricDofField

GeometricDofField is a template class that stores the field data based on dgMesh. This template class is inherited from the OpenFOAM data structure GeometricField, and its parameters are described as follows:

- Type: Basic types of field data, such as label, vector, etc.
- PatchField: Data types of boundary field, this parameter can only be dgPatchField <Type> in GeometricDofField. dgPatchField is the boundary field data structure of the DG method, which is used to save field value of discrete points on the boundary surface. Its implementation is similar to the one of fvPatchField in OpenFOAM.
- GeoMesh: Types of discrete grid, and this parameter can only be dgMesh in

GeometricDofField

GeometricDofField contains the following member variables:

- boundaryField_: Where the field values of discrete points on the boundary surface are stored. The data structure of boundaryField is defined by the Boundary class in GeometricField, which can be obtained by the boundaryField() and boundaryFieldRef() member functions.
- timeIndex_: It is the label data type used to store index value of time step in solving process. Index value indicates whether the stored field value is calculated at the current time step or not, then determines whether the member variable field0Ptr_ needs to be updated or not.
- field0Ptr_: A pointer of GeometricDofField class which stores filed data of the previous time step.
- fieldPrevIterPtr_: A pointer of GeometricDofField class which stores filed data of the previous iteration step.
- gaussField_: A pointer of dgGaussField class which points to the Gaussian field based on GeometricDofField, GaussField_ stores the field values of Gaussian points generated by the original discrete points and field values.

2.3 Basis function data structure: physicalElementData

Basis functions are the foundation of DG numerical discretization, and the discrete process consists of three main steps: a) Creating the interpolation points to form high-order mesh based on geometric information of mesh and discretization order in dgPolyMesh. b) Mapping the geometry elements to standard elements. c) Implementation of the numerical discretization based on the standard element basis functions.

High-order discretization is achieved by higher order polynomials fitting of field values in flow field. This process will increase high-order points to store more flow field information. Taking two-dimensional triangular element as an example, figure 2.4 shows the process of obtaining DG third-order approximation mesh through interpolating high-order points on dgPolyMesh mesh. After interpolating high-order points, Face and Cell in OpenFOAM mesh correspond to physicalFaceElement

class and the physicalCellElement class in DG. The mesh information of high-order discretization are stored in tree structure dgTree <Type>. The faceElementsTree_ and cellElementsTree_ in the physicalElementData are the face tree and cell tree, respectively.

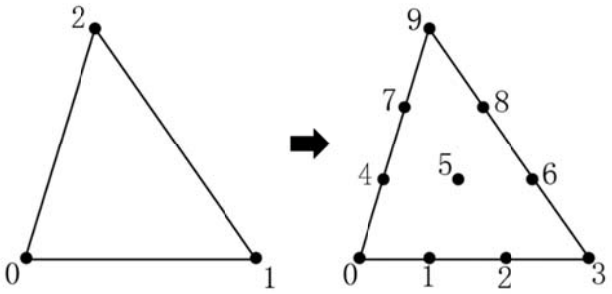

Fig2.4 Process of interpolating high-order points

A mapping relation between the general straight-sided triangle and the standard triangle in a two-dimensional space is shown in figure 2.5. All general elements are mapped to standard elements by coordinate transformation, with each element having a different mapping, Ψ. The integration is realized on standard elements, for detailed information, please refer to reference book[1]. The mapping information of elements are stored in physicalCellElement class.

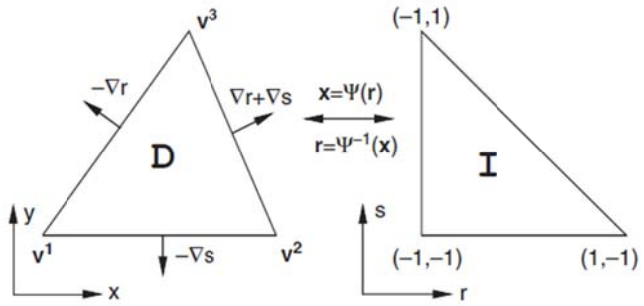

Fig2.5 Mapping between two triangle elements

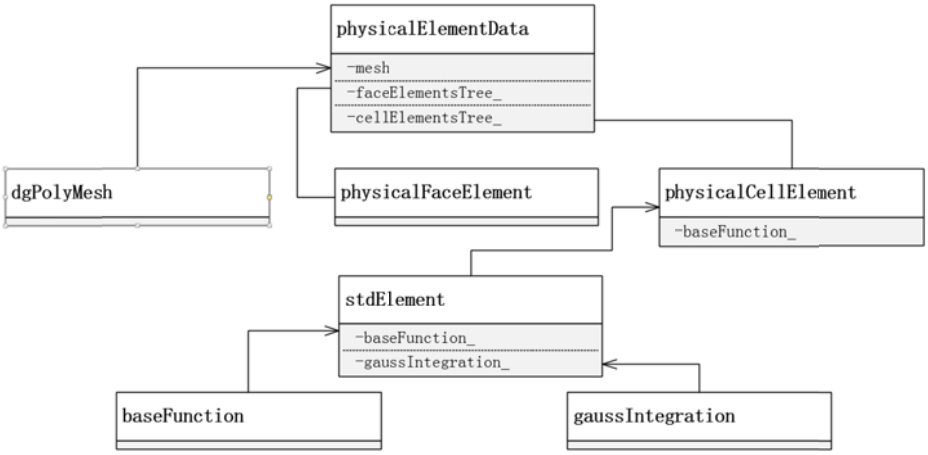

Fig2.6 Class diagram of basis functions

Basis functions of standard elements are accessed through baseFuntion_ in physicalElementCell. The class diagram of basis functions is shown in figure2.6.

2.4 Matrix data structure: dgMatrix

Partial differential equations are discretized into a linear system of equations, whose storage and solution are managed by dgMatrix class. Figure 2.7 demonstrates the class diagram of matrix data structure. The data structure is determined by properties of the coefficient matrices constituting the linear system of equations. Coefficient matrices are sparse matrices. denseMatrix class is used to store small, dense blocks in sparse matrix. The dgLduMatrix class assembles all dense blocks into the final coefficient matrix, and stored in mat_ of the dgMatrix class. ksp_ and mat_ are the data structure for solving linear system of equations with external dependency package PETSc.

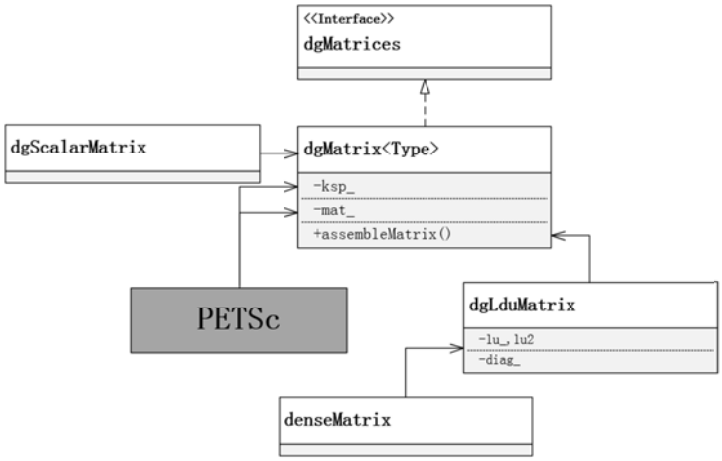

Fig.7 Class diagram of matrix

3 Typical process of discretization and solution

3.1 DG numerical principle

This section take the solution of a simple partial differential equation to illustrate the principle of DG numerical discretization. DG is a high-order finite element method. Compared with other high-order finite element methods, the most important characteristic is the decoupling of the elements. Since the residual integration of the finite-element method is treated as the local calculation of each element, we can express the local solution u as a polynomial of order N on arbitrary element D_k:

$$u_h^k = \sum_{i=1}^{N} \hat{u}_i^k \cdot \varphi_i^k \tag{1}$$

Considering the equation:

$$\frac{\partial u}{\partial x} = 0 \tag{2}$$

Since the residuals are orthogonal to any basis functions $\varphi_j^k (j = 1, 2, \cdots, N)$, do integral on the both side of Eq.(2) in D_k,

$$\begin{aligned} 0 &= \int_{D_k} \frac{\partial u_h^k}{\partial x} \cdot \varphi_j^k dx \\ &= -\int_{D_k} u_h^k \cdot \partial \varphi_j^k dx + \oint_{D_k} \vec{\boldsymbol{n}} \cdot u_h^{k*} \cdot \varphi_j^k dx, \ (j = 1, 2, \cdots, N) \end{aligned} \tag{3}$$

where $u_h^{k*} = f(u^{k-}, u^{k+})$, thus the equation is transformed into the sum of volume integral and surface integral.

The polynomial of order N is then substituted into Eq.(3), and express the coefficients with matrices:

$$\begin{aligned} \int_{D_k} u_h^k \cdot \varphi_j^k dx &= \int_{D_k} \sum_{i=1}^{N} \hat{u}_i^k \cdot \varphi_i^k \cdot \varphi_j^k dx \\ &= \sum_{i=1}^{N} \hat{u}_i^k \int_{D_k} \varphi_i^k \cdot \varphi_j^k dx, \ (j = 1, 2, \cdots, N) \end{aligned} \tag{4}$$

$$\int_{D_k} u_h^k \cdot \partial\varphi_j^k dx = \int_{D_k} \sum_{i=1}^{N} \hat{u}_i^k \cdot \varphi_i^k \cdot \partial\varphi_j^k dx$$

$$= \sum_{i=1}^{N} \hat{u}_i^k \int_{D_k} \varphi_i^k \cdot \partial\varphi_j^k dx, \ (j = 1, 2, \cdots, N) \tag{5}$$

Substituting Eq.(4) and Eq.(5) into Eq.(3) obtains a set of linear system of equations, where $\hat{u}_i^k \ (i = 1, 2, \cdots, N)$ are unknown variables to be calculated. At this point, the DG numerical discretization of Eq.(2) is completed.

3.2 Discrete operator

The discrete operators are obtained by applying the DG numerical discretization method to differential operators in partial differential equation. The discrete operators provided by HopeFOAM-0.1 are listed in Tab.1

Tab.1 Discretization of PDE Terms in DG of HopeFOAM-0.1

Term description	Implicit/ Explicit	Text Expression	dgm::/dgc:: function
Time derivative	Imp/Exp	$\frac{\partial \phi}{\partial t}$	ddt(phi)
Convection	Imp/Exp	$\nabla \cdot (U\phi)$	div(U,phi)
Divergence	Exp	$\nabla \cdot \phi$	div(phi)
Gradient	Exp	$\nabla \phi$	grad(phi)
Laplacian	Imp	$\nabla^2 \phi$	laplacian(phi)
Source	Imp	ϕ	Sp(phi)

phi: dg<Type>Field

U: Field of velocity

3.3 Architecture of discrete system

The top layer discrete operator interfaces in the DG discrete system are similar to OpenFOAM interfaces, as is shown in figure 3.1. dgm and dgc are respectively the implicit and explicit discrete operator interfaces, and users need to determine which

discrete operator interface to choose according to discrete scheme, and then access the discrete operator.

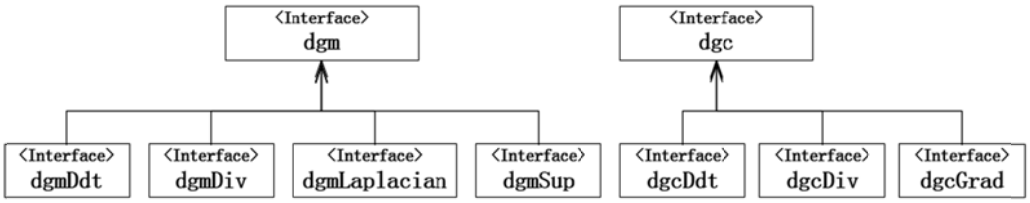

Fig3.1 Structure of top layer discrete operator interfaces in the discrete system

Functions of discrete operator are accessed through discrete operator interfaces. Because the DG meshes are organized by elements, the numerical discretization is constructed in the same way. The Equation class is used to store intermediate results of discrete process; the EquationAdd class provides the addition function for Equation; the EquationMul class provides the function to multiply Equation by a constant and the EquationEqual class implements operation of two Equations are equal.

The organizational structure of each discrete operator has the similar form, in this paragraph, the first-order time discrete operator ddt is used as an example to introduce organizational structure. Figure 3.2 demonstrates the class diagram of ddt. Member function ddt() of explicit interface dgcDdt and implicit interface dgmDdt access dgcNew() function and dgmNew() function in EquationDdtSchemeScheme<Type> class, which instantiates objects of EquationEulerDgcDdtScheme<Type> class and EquationEulerDgmDdtScheme<Type> class. The calculateCell () function is used to call the dgcDdtCalculateCell () function and dgmDdtCalculateCell () function to complete the first-order time discretization of each element.

DG flux calculation codes are stored in the simpleFlux and godunovFlux directories. The simpleFlux directory contains three simple flux calculation methods, namely, averageFlux, LFflux and noneFlux. The godunovFlux directory includes complex flux calculation methods, *limiter* folder in this directory serves the function of limiter. HopeFOAM-0.1 only provides Roe flux calculation method and the first-order limiter at present.

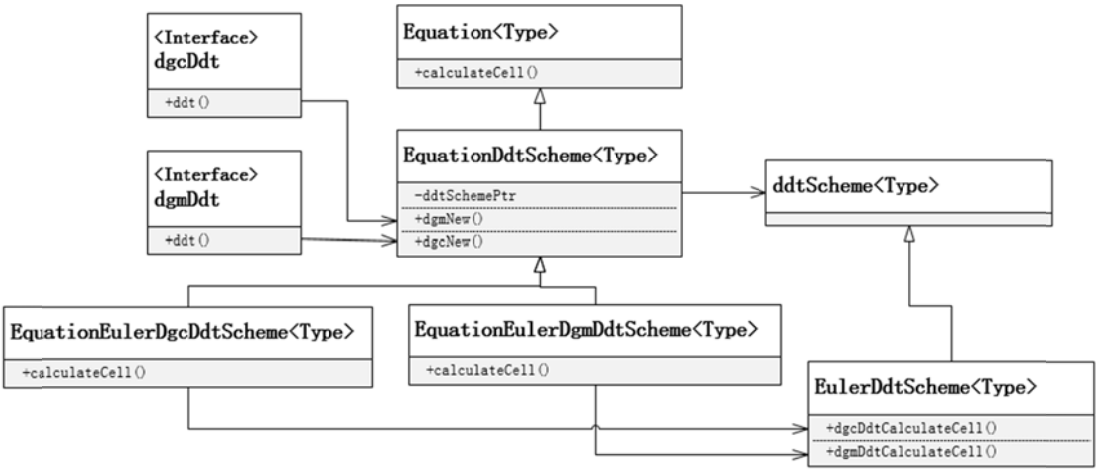

Fig3.2 Class diagram of ddt data structure

4 Example of the program of HopeFOAM: flow around cylinder

4.1 Problem description

The flow around cylinder is a time-dependent two-dimensional flow through a channel, with a circular obstacle in the channel. The cylinder is placed off the centerline of the channel, so it begins to shed vortices as soon as the parabolic inflow is ramped up from its zero initial condition. This test case was suggested as a benchmark in the mid-1990s. The problem domain is shown in Figure 4.1

Fig4.1 Domain of flow around cylinder

The simulation time is 8s, initial time t=0, and the inlet boundary conditions for this case are given by a parabolic inflow profile, modulated by a sine function depending on time as

$$\begin{aligned} u(x,y,t) &= 0.41^{-2}\sin(\frac{\pi t}{8})6(y+0.2)(0.21-y), \\ v(x,y,t) &= 0. \end{aligned} \tag{6}$$

4.2 The development of dgChorinFoam solver

4.2.1 Declaring dependencies of DG core

For DG discrete method, the solver needs to declare header file dgCFD.H, and the following header files:

- setRootCase.H: Initialize system parameters.
- createTime.H: Create a time control object runTime.
- createField.H: A user-defined hear file which define field data and other data that need to be used in solution.

- createMesh.H: Build mesh object of numerical simulation. If users use DG method, dgMesh object needs to be constructed.
- createTimeControl.H: Allow users to adjust runTime by defining configuration parameters in system/conrolDict in case directory.

4.2.2 Field data initialization and boundary field data update

OpenFOAM provides users to initialize field data and update boundary field data by setting boundary type in configuration files in case directory. Currently, HopeFOAM-0.1 does not serves enough boundary types. Since the case has complex initial values and time-dependent boundary conditions, users need to write code in solver.

In this case, initialize field data is that velocity field of inlet boundary is initialized in *setNonUniformInlet.H*, boundary field data is updated in *setBoundaryValues.H*, which will be called at the end of each timestep.

4.2.3 Physics equation description

The physics equation description of DG follows the original style of OpenFOAM except the discrete namespace. For example dgc is implicit discrete namespace, dgm is explicit discrete namespace. Take Eq.(7) as an example

$$U^{n+1} + \nabla p^n = U^n \tag{7}$$

The superscript n represents the last time step, n+1 represents the current time step. So U^{n+1} is unknown, p^n and U^n are known values of the last time step. In the solver, Eq.(7) is expressed as

$$dgm :: Sp(U) + dgc :: grad(p) == U \tag{8}$$

dg::solveEquation() function is called to solve equation in solver

$$dg :: solveEquation(dgm :: Sp(U) + dgc :: grad(p) == U) \tag{9}$$

References

[1] Hesthaven J S, Warburton T. Nodal discontinuous Galerkin methods: algorithms, analysis, and applications[M]. Springer Science & Business Media, 2007.