User's Guide for dugksFoam

An OpenFOAM solver for Boltzmann model equation

by Lianhua Zhu

Document Version 1.0

December 14, 2015

Preface

The dugksFoam is an OpenFOAM solver for the Boltzmann equation with the Shakhov collision model. The numerical method behind it is the discrete unified gas kinetic scheme (DUGKS, see Ref. [2]). The DUGKS discretizes the governing equation in both physical space and velocity space. It solves the partial differential equations of the discrete velocity distribution functions in a finite volume framework. In DUGKS, the fluxes of distribution functions are constructed from the local characteristic solution of the governing equation itself. This feature makes DUGKS very efficient for simulating near continuum flows.

The OpenFOAM is one of the most popular open-source general CFD toolkits. The biggest feature of it is that it allows users to develop their own solvers in a very high level. The OpenFOAM provides the solver developers varies ready-to-use major components of numerical solving of PDE (mainly for finite-volume discretization), such as the arbitrary unstructured mesh representation, spatial discretization operator, time integration schemes, boundary condition types and message passing interface (MPI) based parallelization. In the development of a typical OpenFOAM solver, the developer spends most of the time to define the solving procedure, i.e., writing the Field Operation And Manipulation expressions. Besides these basic components, OpenFOAM also provides a branch of general utilities for pre-processing, post-processing, parallel computing, job control etc.

By implementing the DUGKS into an OpenFOAM solver, we can take many advantages of the OpenFOAM toolkit. Such as the easy pre and post processing, parallelization, solving control and parameter configurations. We expect it can be a convenient tool for study non-equilibrium gas flow and heat transfer problem in complex geometries. In addition, it can serve as a reference for developing other kinetic type equations such as the phonon transport equation, semiconductor equation etc., because solving kinetic type equation in OpenFOAM is not so that common compared with those macro-filed based solvers the OpenFOAM provides. The only kinetic type equation solver appears in official OpenFOAM distribution is the discrete ordinates model (DOM) for thermal radiation computation.

In this documentation, we present the installation, usages, demo cases of the dugksFoam. For the detailed information about the DUGKS, one can

refer to the papers by Guo et al[1, 2]. For the detailed of implementation of the DUGKS in unstructured mesh and the configuration of demo cases in this documentation, one can refer the paper post on arxiv.org by the author[3].

Contents

Chapter 1

Installation

Before the installation of dugksFoam, you should have installed the OpenFOAM together with the ThirdParty tools on your Linux machine. The download address of OpenFOAM and the detailed installation instructions can found in the official web site of OpenFOAM and the OpenFOAM wiki. The OpenFOAM versions I have tested is 2.2.1, 2.3.0 and 2.4.0. But the dugksFoam should also works on the latest release of OpenFOAM (Ver. 3.0.0 or above).

The detailed installation instructions are as follows.

1. Load the OpenFOAM environment: Type the command `ofxxx` where `xxx` is the three digits of the OpenFOAM version you installed, if you have followed the official installation instructions of OpenFOAM. For example, `of240` or `of230`.

2. Create your own solvers installation location, and cd to it :

   ```
   mkdir -p $FOAM_RUN/../applications
   cd $FOAM_RUN/../applications
   ```

3. Get the source code using git (see below) from dugksFoam repository or download it as a ZIP package by clicking here.

   - If using git:

     ```
     git clone git@github.com:zhulianhua/dugksFoam.git
     cd dugksFoam/src
     ```

   - If installing by ZIP package, move the ZIP package (`dugksFoam-master.zip`) to `$FOAM_RUN/../applications`. Then unzip it by

     ```
     unzip dugksFoam-master.zip
     mv dugksFoam-master dugksFoam
     cd dugksFoam/src
     ```

4. For OpenFOAM release older than 2.4.0, there is a compatible issue in the make file options about `meshTool`. If you are using OpenFOAM older than 2.4.0, fix it by this command:

   ```
   git apply PatchMeshToolIssue
   ```

5. Compile the dugksFoam by:

   ```
   ./Allwmake
   ```

6. Check if the compilation is OK:

   ```
   which dugksFoam
   ```

   It should tell you where the compiled executable `dugksFoam` is.

Chapter 2

Usage

2.1 Overview

Besides the standard `system/controlDict`, the mesh files in `constant/polyMesh` and the initial fields (`0` directory), you should also provide the following additional configuration files,

- `constant/Xis` : the discrete velocity set in 1D;
- `constant/weights` : the weight coefficients corresponding to the discrete velocity set;
- `constant/DVMparameters` : sets the gas parameters and discrete velocity information;
- `system/fvSchemes` : sets the discrete scheme for the gradient evaluation.
- `system/fvSolution` : dummy file (sets nothing).

The formats for these configuration files are described in the following subsections. It is recommoned to always start a new case by copy-and-modifying an existing case, such as the the demo cases provided with the dugksFoam source code (see Sec. 3.1).

2.2 Step by step guide

2.2.1 Prepare the physical meshes

Generate the mesh either by using `blockMesh/snappyHexMesh` provided by OpenFOAM or using a third party mesh generation softwares, such as Gambit, pointwise, ICEM CFD or gmsh. Note that some of the third party softwares in recent release support exporting mesh files in OpenFOAM internal format directly, such as the pointwise. Nevertheless, you can always convert a

mesh in other formats into a OpenFOAM internal format by the mesh-conversing tools provided by OpenFOAM, such as the `fluentMeshToFoam` or `gmshToFoam`.

I personally use the Gambit a lot. The general procedure is :

1. Build the geometry and meshing it in Gambit.
2. Export fluent mesh file in Gambit, say `demo.msh`.
3. Copy the fluent mesh (`demo.msh`) to your case directory.
4. Convert the fluent mesh to OpenFOAM mesh by

```
fluentMeshToFoam demo.msh
```

2.2.2 Set initial macro variable field and boundary conditions

The initial macro field files in the `0` directory required are,

- `0/rho` : the initial density field;
- `0/U` : the initial velocity field;
- `0/T` : the initial temperature field.

Set the initial values (usually uniform) for the three macro fields. Modify the boundary condition types for each boundary patch in the three files according to the physical boundary condition types. The solver supports several commonly seen physical boundary condition types. There are

- diffusive wall boundary;
- specular reflection wall boundary
- far field boundary;

Their usages are presented in the following.

Diffusive wall boundary

For the diffusive wall boundary, you need to set the corresponding boundary patch in `0/rho` to `calculatedMaxwell` together with a dummy uniform value. Below is an example:

```
boundaryField
{
     ...
     topWall
```

```
    {
        type            calculatedMaxwell;
        value           uniform 1.0;
    }
    ...
}
```

You should also set the corresponding boundary patch in `0/U` and `0/T` as `fixedValue` type and provide the wall's moving velocity and temperature.

Specular reflection wall boundary (symmetric boundary)

Physically, the specular reflection wall boundary is identical to the symmetry boundary. Currently, the dugksFoam supports only symmetry boundaries aligned in the X/Y/Z directions. To specify a symmetric boundary, set the corresponding boundary patch type as `symmetryPlane` in all of the three initial fields, i.e., `0/rho`, `0/U`, `0/T`. Note that you should also change the basic boundary patch type in `constant/polyMesh/boundary` to `symmetryPlane`.

Far field boundary

This boundary type is exclusively used for the outer boundary in the simulation of supersonic extern flows past objects. The physical interpretation is that the particles comes into the computational domain with the far field equilibrium velocity distribution. To specify such a boundary type, just set the boundary types as `fixedValue`, and provide the free-stream flow condition as the boundary values in `0/rho`, `0/U` and `0/T`.

2.2.3 Prepare the discrete velocity set

The dugksFoam currently only support Cartesian grids in velocity space. And the grid points are identical in each direction for 2D or 3D problems, i.e., $\xi_{ix} = \xi_{iy} = \xi_{iz} \equiv \xi_i$. For 1D and 2D problems, it uses the dimensional reduction technique in the velocity space, which improves its efficiency considerably. The discrete velocity set ξ_i and the corresponding weights w_i are provided by two files, the `constant/Xis` and `constant/weights`. Each of the files represents a 1D list. Below are examples of `constant/Xis` and `constant/weights` files corresponding to a 28 point discrete velocity grid.

- `constant/Xis`:

```
FoamFile
{
    version     2.0;
    format      ascii;
```

```
    class       scalarList;
    location    "constant";
    object      Xis;
}
// ************************************************ //
28
(
-1.7664856627356885e+03
-1.5149431219611315e+03
-1.3067831284809952e+03
 //...
 //... skip of 22 lines
 //...
 1.3067831284809952e+03
 1.5149431219611315e+03
 1.7664856627356885e+03
);
// ************************************************ //
```

- constant/weights:

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       scalarList;
    location    "constant";
    object      weights;
}
// ************************************************ //
28
(
 2.9037294321023950e+02
 2.2430119141828482e+02
 1.9483915929202675e+02
 //...
 //... skip of 22 lines
 //...
 1.9483915929202675e+02
 2.2430119141828482e+02
 2.9037294321023950e+02
);
// ************************************************ //
```

For Newton-Cotes quadrature and half-range Gauss-Hermite quadrature, we provide the script `setDV.py` to modify the two files conveniently. The script is located in `src/scripts`. If you followed the installation steps in Sec. 1, you should be able to run `setDv.py` and thus set those files directly. For example, to set a 28-points Gauss-Hermite discrete velocity set, simply run

```
setDV.py GH 408.16 28
```

where `GH` stands for Gauss-Hermite, `408.16` is the most probable molecular speed, and `28` is the number of discrete velocities. Or to set a 81-points compound Newton-Cotes rule discrete velocity set, run

```
setDV.py NC 2000.0 81
```

where `NC` stands for Newton-Cotes, `2000.0` stands for the max discrete velocity, and `81` is the number of discrete velocities

2.2.4 Set gradient evaluation schemes

There are two schemes available to evaluate the gradients of distribution during the reconstruction step. They are `leastSquares` and `Gauss linear`. Both of them can be modified with a limiter function by the keyword `cellLimited`. The strength of the limiter can be controlled by a scalar parameter s, $0 < s < 1$. $s = 0$ means don't limit, while $s = 1$ means full limiting. These options are input in the section of `gradSchemes` of the file `system/fvSchemes`. Below is an example. Note that `divSchemes` and `laplacianSchemes` are always set to `none` as dugksFoam doesn't use those operators.

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// ************************************************ //
gradSchemes
{
    default         leastSquares;
//  default         cellLimited Gauss linear 1.0;
//  default         cellLimited leastSquares 0.5;
}

divSchemes
```

```
{
    default         none;
}

laplacianSchemes
{
    default         none;
}
// ************************************************* //
```

2.2.5 The dummy fvSolution file

The `system/fvSolution` should always be provided as follows

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// ************************************************* //
solvers
{
}
// ************************************************* //
```

Nothing is set in that file, since the dugksFoam is an explicit solver.

2.2.6 Set solving control parameters

The main control parameters for the solver running, such as when to stop, how often to dump immediate macro fields, time step size etc. are set in the file `system/controlDict` as like other OpenFOAM solvers. Note that you can turn on the `adjustTimeStep` option and provide a maximal CFL number by the `maxCo` keyword. Below is an example of the `system/controlDict` file

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
```

```
// ************************************************* //
application     dugksFoam;
startFrom       latestTime;
startTime       0;
stopAt          endTime;
endTime         2.0e2;
deltaT          4.00e-4;
//writeControl    adjustableRunTime;
writeControl    timeStep;
writeInterval   200;
purgeWrite      0;
writeFormat     ascii;
writePrecision  16;
writeCompression off;
timeFormat      general;
timePrecision   9;
runTimeModifiable true;
adjustTimeStep  yes;
maxCo           0.8;
maxDeltaT       1;
// ************************************************* //
```

2.2.7 Set gas properties

The gas properties are set in the `gasProperties` section of file `constant/DVMProperties`. You should specify the specific gas constant R, the viscosity-temperature relation exponent ω, the reference temperature T_{ref}, the reference viscosity μ_{ref} at T_{ref}, and the Prandtl number Pr. Note that you should also provide the maximum and minimum discrete velocity and the number of discrete velocity in the `fvDVMparas` section of this file. They should be consistent with `constant/Xis` files. An example of the `constant/DVMProperties` file is shown below,

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      DVMProperties;
}
// ************************************************* //

fvDVMparas
```

```
{
    xiMax       xiMax [0 1 -1 0 0 0 0]  2.138251518302359e+03;
    xiMin       xiMin [0 1 -1 0 0 0 0] -2.138251518302359e+03;
    nDV               28;  // Number of discrete velocity
}

gasProperties
{
    R            R [0 2 -2 -1 0 0 0] 208.244343891;
    omega        0.81; // VHS viscosity ~ Temperature index
    Tref         Tref [0 0 0 1 0 0 0] 400.0;
    muRef        muRef [1 -1 -1 0 0 0 0] 1.94014016536e-05;
    Pr           0.666666666666667; // Prantl number
}
// ************************************************* //
```

2.2.8 Run in serial or parallel

If all of the files and parameters described in the previous subsections are prepared, you can finally run the `dugksFoam` solver. To run `dugksFoam` serially,

```
dugksFoam
```

To run `dugksFoam` in parallel with `mpirun`, you should firstly decompose the computational domain use the tool `decomposePar`. Refer the OpenFOAM official User'S Guide to see how. After the domain decomposition, you can run `dugksFoam` in parallel by

```
mpirun  -np 8 dugksFoam -parallel
```

where 8 means to use MPI processes.

2.2.9 Post processing

After running the solver, you can see the immediate results directories. If you run in parallel, you have to reconstruct the results files by `reconstructPar` first. To view the results, you can use either ParaView or Tecplot.

- To use ParaView : create an empty file in the case directory by

  ```
  touch a.foam
  ```

  Then use the ParaView to read the `a.foam`.

- To use Tecplot : use the tool FoamToTecplot360 to convert the results to Tecplot format results, then use Tecplot. Or use the latest releases of Tecplot360 which support reading the OpenFOAM result files directly.

Chapter 3

Demo cases

3.1 2D cavity flow at Kn = 0.075

This demonstrational case is provided in the `demos` subdirectory of the dugksFoam source code package. This case is a popular benchmark problem for validating numerical method for micro or rarefied gas flows. It has been studied in Ref. [3] using this solver, where you can find the detailed description of this problem. We only mention some setting that need special attention for a new user. The mesh file and setting have already been prepared in the case directory. So you can run the dugksFoam directly.

The flow configuration is illustrated in Fig. 3.1. The walls are diffusive boundaries. For such a simple geometry, you can use the `blockMesh` shipped with the OpenFOAM to generate the structured mesh. Refer to the cavity flow tenurial case in the *OpenFOAM User's Guide* for the detailed usage of `blockMesh`. The initial temperautre filed is uniform 273K, and the wall temperature is also 273K. In this case, the Knudsen number Kn is 0.075 based on the initial density filed and the cavity width L. So the mean free path is 0.075m. The initial density field input in the `0/rho` file should be calculated from the mean free path provided the argon gas properties. Refer to [3] for the related formulations. We also provide a simple Python script named `para.py` in the case's directory to compute the related parameters. You can run it by `python para.py`.

The discrete velocities used are 28×28 half-range Gauss-Hermite quadrature points. The files `constant/Xis` and `constant/weights` can be generated by

```
setDV.py GH 337.196399395 28
```

where 28 is the number of discrete velocity in each direction, and 337.196399395 stands for the most probable speed of argon gas molecular at $T = 273$K. Refer to Sec. 2.2.3 for more details about settings of discrete velocities.

Fig. 3.2 show some of the results of this case. You can also compare the results with those in [3] in detail.

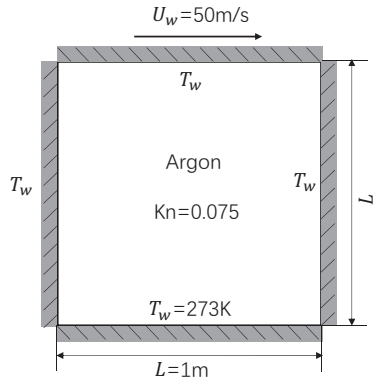

Figure 3.1: Lid-driven cavity flow

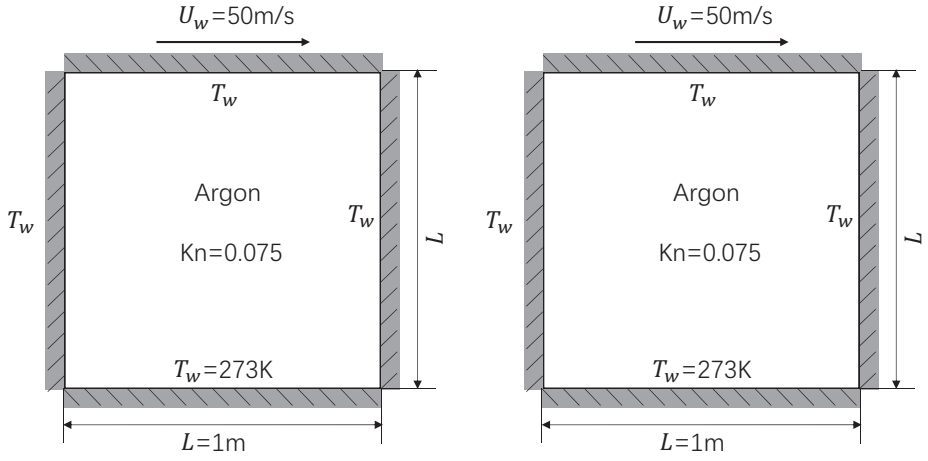

Figure 3.2: Results of the cavity flow case. (a) Temperature contours and heat flux. (b) Velocity magnitude and streamlines.

Bibliography

[1] Z.L. Guo, K. Xu, R.J. Wang, Discrete unified gas kinetic scheme for all Knudsen number flows: low-speed isothermal case, Phys. Rev. E, 88 (2013) 033305.

[2] Z.L. Guo, R.J. Wang, K. Xu, Discrete unified gas kinetic scheme for all Knudsen number flows. II. Thermal compressible case, Phys. Rev. E, 91(2015) 033313.

[3] L.H. Zhu, Z.L. Guo, K. Xu, Discrete unified gas kinetic scheme on unstructured meshes, arXiv preprint arXiv:1503.07374, (2015).