COUPLED CFD-DEM MODEL USER GUIDE

Jacob R. Madden *

Viterbi School of Engineering, University of Southern California

ASTE 499 - Dr. Lubos Brieda

May 13, 2020

*jrmadden@usc.edu

Contents

1 Software **3**

2 Meshing **3**
2.1 Preliminary Geometry Work 3
2.2 Tet Meshing in Salome 4
2.3 Hex-Tet Meshing in OpenFOAM 4
2.4 Mesh Quality 5

3 CFD Sub-Model **6**
3.1 Setting Up Your Directory 6
3.2 controlDict 6
3.3 fvSolution 6
3.4 fvSchemes 7

4 DEM Sub-Model **7**
4.1 DEM Domain 7
4.2 Input Script 7

5 Running The Simulation **7**

Pneumatic conveying systems are used to facilitate bulk movement of granular or powder materials in many industries (mining, chemical, agricultural, etc), and several numerical modeling techniques exist to characterize their behavior. This user guide describes a coupled CFD-DEM model used to predict bulk particle movement due to gas expansion into a hard vacuum, assuming cohesive, homogeneous particles and isentropic, incompressible, turbulent gas flow. The model was developed using open source modeling software (OpenFOAM, LIGGHTS, CFDEMcoupling), and detailed explanations are provided for modifying model parameters and assumptions for the fluid, particle, and coupling states. This low-fidelity model is designed to validate hardware testing and accelerate pneumatic system design workflow by reducing the physical testing required to iterate on designs.

Note that this user guide includes general instructions and information regarding meshing, the OpenFOAM CFD applications, and the LIGGGHTS DEM applications, to facilitate the development of a CFD-DEM case from scratch. However you may skip to the **Running the Simulation** section for directions on running and post-processing data for an existing CFD-DEM case, independent of the other sections.

1 Software

Use of the CFD-DEM simulation model described in this user guide requires an installation of OpenFOAM (CFD), LIGGGHTS (DEM), and CFDEMcoupling. A detailed explanation of the installation and compilation can be found at [1]. Note that the OpenFOAM version is specific to this coupling package. For post-processing and visualization, a working installation of VTK (version 5.8 or higher) and Paraview are also required. Details can also be found at [1].

In this report, OpenFOAM version 5.x (build -538044ac05c4) was installed and compiled, with LIGGGHTS version 3.8.0 and CFDEMcoupling version 3.8.1. VTK version 6.3.0 and Paraview 5.4.0 were used for visualization.

2 Meshing

An accurate CFD simulation begins with a quality mesh. The overarching workflow includes pre-processing of the fluid geometry to prepare for meshing, generation of the mesh itself using external meshing software such as Salome or internal OpenFOAM meshing utilities, and defining boundaries and groups on the completed mesh for the CFD simulation. Here, the pre-processing and base mesh generation will be completed in Salome, with further refinement completed using OpenFOAM utilities.

2.1 Preliminary Geometry Work

To begin with, the base geometry should be simplified and knit together to create water-tight boundaries at the inlet, outlet, and any connecting features. These can be maintained as separate entities for boundary condition purposes. Sharp corners and edges should be filleted and external features should be eliminated completely (assuming internal pipe flow). The Salome GEOM (Geometry) tools such as *Sewing*, *Hole/face Suppression*, and *Fusing*

can be used here to ensure a watertight solid. The geometry can be analyzed using the inspection tools (*Check shape*, *Get Compound of Blocks*, *Detect self-intersections*) to verify the composition of the geometry (ie. solids, compounds, number of bodies) and ensure that the geometry isn't poorly defined. Once the base geometry has been fully simplified and fused, move onto the next step.

While still in the GEOM module, use the *Create Groups* function to generate sets of faces including (at a minimum) the inlet, outlet and walls. Additional groups can be generated as needed (ie. for refinement surfaces). At this point, every face should be included in one of the defined groups. Make sure that there are no missing faces, or the meshing tool may run into issues. At this point, to generate a full-tetrahedron mesh, move onto the **Meshing in Salome** step. Otherwise, to generate a hex-tet mesh using the OpenFOAM meshing utilities, move onto the **Meshing in OpenFOAM** step.

2.2 Tet Meshing in Salome

Now, activate the MESH (Meshing) module. For the base mesh, a fully-tetrahedron mesh will work well. Use Create Mesh and select the final watertight object from the GEOM module. Then, under **Algorithm**, select *Netgen 1D-2D-3D*, and under **Hypothesis**, select the *Netgen 3D Parameters* icon. Here, define the minimum and maximum cell size, depending on your object scale, and the *Fineness*. Baselines options are (1e-3, 1e-4) and Fine. The click Apply and Close, and generate the mesh using the Compute Mesh selection. After it finishes, create groups of faces as before, but this time, select the corresponding group from the GEOM module using the Group on geometry option. This will associate the correct boundaries for the mesh. For example, create a wall group, and then select the entire wall group from the GEOM module. The mesh will need to be re-computed afterwards. Then, export the full mesh as a .UNV file and move it into the OpenFOAM case directory. Convert this to a useable OpenFOAM mesh by using the *ideastoUnv* utility (**$ ideasToUnv meshName.unv**). The tetrahedral mesh is now ready for CFD simulation.

2.3 Hex-Tet Meshing in OpenFOAM

Now, export the full geometry solid as an .STL file. Also select each group of faces and export as .STL, using the same group name for the .STL file. Make sure that the ASCII option is checked (not binary). This can be verified by opening up the resultant .STL file - it should consist of legible "loops", "vertices", and "facets". If the file instead consists of columns of nonsensical numbers and letters, it has been exported as binary.

Then, copy these .STL files into the OpenfOAM case, under the **caseName/constant/triSurface** sub-directory. It is important that these sub-directories are names exactly as described. Otherwise, the OpenFOAM meshing utilities won't be able to find them. (Check the next section on Setting Up Your CFD Case Directory for more details).

There are three primary dictionaries (as defined by OpenFOAM) that are used for mesh generation: *blockMeshDict*, *snappyHexDict*, and *surfaceFeatureExtractDict*. Examples for base dictionary settings can be found in the tutorials (use **$ find . -name "snappyHexDict"**, etc to find these cases). A brief overview of each, with basic settings will now be presented.

The *blockHexDict* defines the base hexahedral domain for the final tet-hex mesh. For internal flow, this domain can be smaller than the volume of interest (if only part of the domain should be hex meshed - details can be found at [2]). Otherwise, for external flow, the domain should be larger than the object of interest, to ensure that flow is accurately captures. The remainder of this tutorial assumes internal pipe flow. In this dictionary, the vertices define the faces of the hex mesh domain (see [3] for a detailed explanation). Then the blocks define the grading of the hex mesh. Here, use a simpleGrading approach (constant interval hex blocks along each axis). The mergePatchPairs entry is only necessary for more complicated mesh definitions - ignore it here.

The *snappyHexMeshDict* dictionary provides a myriad of options to refine the hex-tet mesh parameters. One a select few will be discussed here - see [4] for more detail. In the geometry entry, write out the *inlet*, *outlet*, *walls*, *volume*, etc groups, pointing to the corresponding .STL file. To allow for surface refinement (ie. rounded corners and edges), write out the appropriate **.eMesh* file, specifying the appropriate level (5-6 provides adequate resolution edges). Follow the same process for the refinementSurfaces entry, specifying a refinement level, ie. (5,6), for any groups that require more detailed meshing. The locationInMesh entry will tell the what "side" of the mesh to keep (ie. internal or external). For internal flow here, specify a point inside the volume, not on a face. Other mesh quality controls can be kept at default levels.

The *surfaceFeatureExtractDict* defines the files that features should be extracted from (ie. edges and surfaces). Specify each **.STL* here, using the default options to maintain adequate mesh resolution. For geometries with sharp edges and corners, ensure that the includedAngle entry is set to 150-180, to adequately capture these features.

Finally, to generate the mesh, move into the main case directory. Execute the **$ blockMesh** command. This generates the base hex mesh, which can be visualized using the **$ paraFoam** command. Then, run the **$ surfaceFeatureExtract** command to extract surface features and generate the **.eMesh* files found inside the **constant/triSurface** directory and **.obj* feature files in the **constant/extendedFeatureEdgeMesh** directory. Finally, run the **$ snappyHexMesh -overwrite** command, to generate the final tet-hex mesh using the base hex domain and extracted features. The final mesh data can be found in the **constant/polyMesh** directory. Visualize the final mesh using the **$ paraFoam** command. The hext-tet mesh is now ready for CFD simulation.

Note that a script inside the CFD directory, *Meshrun*, has been written to automate this procedure using MPI processes, also decomposing the domain for use in the final CFD-DEM simulation. Just modify the dictionary files as desired, move the correct *.STL* files into the **constant/triSurface** folder, and run the script to generate the corresponding hex-tet mesh. Log files are also written for each process, to allow troubleshooting in case of failure. Further, the *Meshclean* script will automatically clear out any old polyMesh and processor data, if a new run is required..

2.4 Mesh Quality

Mesh quality can be verified using the **$ checkMesh** command from the main case directory. This will output quality info, such as max skewness, non-orthogonality of cells, boundary openness, patch topology, cell type, and more. This is an extremely useful tool to provide a

quantitative measure of mesh quality. In general, lower non-orthogonality and skewness will lead to easier-to-converge cases.

3 CFD Sub-Model

3.1 Setting Up Your Directory

To run the CFD sub-model, the OpenFOAM case directory must be set up exactly as described below. Otherwise, the utilities won't be able to locate files. For a basic simulation, start by copying over one of the base tutorials, such as **$HOME/OpenFOAM-X.x/tutorials/incompressible/simpleFoam/pitzDaily**, into your run directory **$HOME-/userName/run/...** and modifying the case name to distinguish it.

Verify that the case includes the system, constant, and 0 folders. The system folder includes the main simulation control files which dictate the case time interval, step, and output file properties (*controlDict*), finite volume discretization method and parameters (*fvSchemes*), and equations solvers and algorithms for each fluid property (*fvSolution*). The constant folder holds dictionaries with parameters for turbulence, thermophysical, or transport models, as well as meshing data. It also includes the *couplingProperties* dictionary, which defines the force models for particle-fluid interaction forces, and model parameters. Finally, time folders (0, 0.1, 0.2, ...) hold initial conditions and boundary field conditions for the domain, and results from each time interval. Note that the remainder of this section assumes the mesh is ready for simulation, after following either **Meshing in Salome** or **Meshing in OpenFOAM**, above.

3.2 controlDict

The three required dictionaries for the system sub-directory are the *controlDict*, *fvSchemes*, and *fvSolution*. The *controlDict* file is straightforward and easy to modify - the base solver is given by the application entry, the start and end times correspond to simulation time, with the *deltaT* parameter dictating the CFD timestep (as bounded by the Courant number), and the IO controls (*writeControl*, *writeInterval*, *writeFormat*, etc) defining the output time files structure (ie. 0.1, 0.2, 0.3, ...). The *deltaT* entry is the only parameter which affects solver stability in this dictionary.

3.3 fvSolution

The *fvSolution* dictionary dictates the matrix solver algorithm, numerical procedure parameters for PISO/SIMPLE/PIMPLE algorithms, and relaxation factors. These have been tuned for computational speed and accuracy, and should be left as is for stability purposes. If computational time is not an issue, the relaxation factors can be tightened up to promote higher accuracy during each time step.

3.4 fvSchemes

The solver discretization schemes can be selected in the *fvSchemes* dictionary, however the default options (Crank-Nicolson and standard finite volume discretization with linear interpolation from cell center to cell faces) are recommended. The only non-standard scheme that may be modified here is the discretization of the divergence of the $k-\epsilon$ terms, currently set to an upwind scheme to promote convergence at the cost of accuracy. Depending on the turbulence estimates, this may need to be modified.

4 DEM Sub-Model

4.1 DEM Domain

The DEM sub-model is much simpler to set up than the CFD case. There are only two main scripts (*in.liggghts_init* and *in.liggghts_run*) which determine the DEM sub-model. Besides modification of these two scripts, which reside in the main DEM directory, there is a **DEM/post** sub-directory, which houses a restart script and will hold post-processed data, and the **DEM/meshes** sub-directory, which houses an *.STL* with the bounding box for the DEM domain and and *.STL* with the initial particle insertion bounding box. These can be left alone for now - only the input scripts will be discussed here.

4.2 Input Script

The *in.liggghts_** text files are known as input scripts, as they define all the necessary DEM parameters. During each DEM step, these scripts are parsed and executed line-by-line. First, the particle atom type is defined - leave this as granular so that the mechanical properties and models can be correctly defined. Next, the **fix property/global ...** set defines the particle's mechanical properties (Young's Modulus, Poisson's Ratio, etc). Then, the granular model is defined with **pair_style gran model ...**, which determines the inter-particle interaction model - the base Hertzian model should be used for preliminary simulations. The DEM domain is defined by using the *.STL* mesh file, coupling force model is defined, and the screen output is described. The coupling force model can be explicit or implicit, and this choice may affect solver stability. Finally, the data dump file is defined, for use in post-processing.

5 Running The Simulation

An *Allrun.sh* script exists to automatically run the CFD-DEM case and process data. It first checks for a CFD mesh, using the existing mesh if possible or creating a new hex-tet mesh with the provided *.STL* geometry if not. Then, it will check for a DEM restart file (indicating that particles have been inserted into the domain) and use that if it exists - if not, it will run the *in.liggghts_init* script to insert particles into the region defined by the DEM/meshes *.STL* file. Then, the post-processing data is cleaned out from previous runs, and the parallel CFD-DEM run starts. This continues until the CFD *endTime* is reached.

Then, the program waits for user input (**¡Enter¿**), and will automatically run the *Post-proc* script, to post-process both CFD and DEM data. One the CFD side, it will clean out the decomposed processor directories and create a **.foam* file for visualization of the decomposed CFD data. On the DEM side, it runs the *lpp.py* script to convert the DEM dump data into *.VTK* files. Then, it runs one more *vtkToVtp.py* script, which converts the DEM *.VTK* data files into *.VTP* files, generates a *.PVD* file with the compiled *.VTK* data, and organizes the DEM/post directory. Finally, it launches Paraview with prompts indicating where the final post-processed data is located. All CFD and DEM data is automatically time-stamped and synced, to facilitate easy visualization.

An additional script in the main CFD-DEM directory, *Allclean*, can be used to clear out all CFD and DEM data from previous runs. By default, the CFD mesh and DEM restart files are saved, to reduce unnecessary computational overhead during the CFD-DEM simulation. However, this can be easily modified by changing the **keepCFDmesh** or **keepDEMrestart** options in the script body.

References

[1] DCS Computing. CFDEM®coupling Documentation. `https://www.cfdem.com/media/CFDEM/docu/CFDEMcoupling_Manual.html#id2`. [Online; accessed 13-May-2020].

[2] Elia Agnani. snappyWiki - Cylinder Case - snappyWiki. `https://sites.google.com/site/snappywiki/snappyhexmesh/cylinder-case`. [Online; accessed 12-May-2020].

[3] The OpenFOAM Foundation. *OpenFOAM v7 User Guide*. "https://cfd.direct/openfoam/user-guide/", 2019.

[4] Elia Agnani. snappyWiki - snappyHexMeshDict. `https://sites.google.com/site/snappywiki/snappyhexmesh/snappyhexmeshdict`. [Online; accessed 12-May-2020].