Escoamentos Multifásicos – FEN/PPG-EM/UERJ

Prof° Gustavo Rabello – 2° período 2014 – projeto numérico – 10/11/2014

This numerical project must be developed and presented by the students at the end of the Multiphase Flows course. At first, there is no restriction on the programming language used, however, due to its complexity, I strongly suggest the students to use any script programming language such as Matlab and Python due to the ease of use. If the student is convinced that a compiled programming language is feasible for the code, there will be no benefit for his final grade.

1. Steady laminar fully-developed single-phase channel flow

   - Develop a routine to compute the velocity in a laminar steady fully-developed single-phase channel flow with variable viscosity, using any numerical method. Assume that the viscosity profile is a prescribed input to the routine. Consider three possible wall boundary conditions: (i) prescribed velocity at the wall (Dirichlet), (ii) prescribed velocity-gradient at the wall (Neumann), and (iii) prescribed relation between the shear-stress at the wall and the velocity field. Consider two possible global boundary conditions: (i) prescribed pressure-gradient and (ii) prescribed flow-rate.

2. Turbulence models - Steady turbulent fully-developed single-phase channel flow:

   - eddy-viscosity Prandtl mixing-length model with prescribed funcionts for the mixing-length l_c.
   - $k - \epsilon$ model (optional).
   - $k - \omega$ model (optional).

3. Wall functions:

   - use log-law wall-function as the prescribed relation between the shear-stress at the wall and the velocity-field. Consider definition of smooth and rough walls.

4. Unsteady turbulent fully-developed single-phase channel flow

   - develop a routine for unsteady flow. Assume that the turbulence models and wall functions are the same as for the steady situation, but considering the instantaneous velocity field ("quasi-steady equilibrium"). Consider two possible global boundary conditions: (i) pressure-gradient a prescribed function of time, and (ii) flow-rate a prescribed function of time.

5. Particle tracking (Euler-Lagrange)

   - Develop a routine to compute the 1D velocity and trajectory of a group of particles in a channel. Assume that the unsteady velocity field of the continuous phase is known. Consider that the particles can be subject to different forces (gravity, drag, virtual mass and lift) prescribed as an input to the routine. Assume periodic boundary conditions in the direction parallel to the wall. Consider two possible wall boundary conditions for the particles: (i) smooth walls with elastic bouncing (specular reflection), and (ii) perfectly absorbing walls.

The following items are optional and must be implemented in 2-dimensional codes.

6. RANS Turbulence

    - Develop a routine to compute the "RANS turbulence"(characteristic velocity, time-scale, length-scale and acceleration) at the position of a particle with a known trajectory and velocity (i.e., the "turbulence seen by the particle"). The particle moves in a 2D channel with a known fully-developed turbulent flow (i.e., the mean velocity field is a known function of the distance to the wall and of the time), and periodic boundary conditions in the direction parallel to the walls. Consider two possible types of models: (i) a discrete-eddy model, and (ii) a Langevin model. For the continuous-phase use the models developed in the previous tasks (i.e., an eddy-viscosity Prandtl mixing- length model and/or a k-epsilon model).

7. Particle-laden turbulent channel code

    - Combine the previous tasks into a generic particle-laden fully-developed turbulent channel flow code. The code should be able to deal with an arbitrary inclination with respect to gravity, the different boundary conditions considered in the previous tasks (for both the continuous and the dispersed phases), and statistically-steady or unsteady flow. Neglect the influence of the dispersed phase on the continuous phase and inter-particle interactions (i.e. assume one-way coupling). The code should be able to deal with arbitrary initial conditions for the particles (for both the position and the velocity), prescribed as an input.