Documentation of the code

Adolfo Vázquez-Quesada and Marco Ellero

1 Introduction

In this paper, the documentation for the code used to simulate concentrated spherical particles in a Newtonian or a bi-viscous fluid is presented. The suspension is evolved by considering only the lubrication force between particles and with the walls. For further details, the reader may refer to [6], where the model is described. In this code, the solvent is not explicitly simulated as in [3, 8, 10], but rather through short-range interparticle forces. As a result, this code is faster than the cited ones but is valid only for simulating concentrated suspensions.

2 The code

The code is written in Fortran in an Object-Oriented Programming style (see [4]). The classes (class* files) are programmed using modules and type definitions. Each class has its subroutines (inc* files) included in its own files. The main program is written in main.f90.

To compile the code, the gfortran program must be installed. Other fortran compilers can be used if the makefile files are edited. To compile the code type the following in the shell (within the program directory):

```
make

make link
```

To run the program, modify the *input* file and then type:

```
./biviscous suspension
```

For a clean recompilation, first type:

```
make clean
```

The input variables are read from the *input* file. In the next section, the meaning of each variable in the input file is explained.

2.1 Input Variables

An input file can be found in the ‘src‘ directory. The meanings of the inputs are indicated below:

- **Nsteps**: Number of steps.
- **dt**: Time step.

- **explicit**: `.TRUE.` if the explicit scheme is used; `.FALSE.` if the semi-implicit scheme [3, 8, 10] is used.
- **sweep_tol**: Tolerance for the semi-implicit scheme [3, 8, 10]. Used only if **explicit** = `.FALSE.`.
- **N_sweep_max**: Maximum allowed number of sweeps. Used only if **explicit** = `.FALSE.`.
- **dim**: Number of dimensions (2 or 3).
- **L**: Box size (2 or 3 real numbers, corresponding to the length in the x, y, and z directions).
- **eta0**: Shear viscosity η_0 of the solvent.
- **eta1**: When the biviscous model is considered [9], **eta1** is the shear viscosity η_1 at high shear rates. Used only if **biviscous** = `.TRUE.`.
- **gamma_dot**: The imposed shear rate $\dot{\gamma}$.
- **biviscous**: `.FALSE.` if the solvent is Newtonian; `.TRUE.` if it is biviscous.
- **d_bulk**: Distance (in units of L_z) from the walls where we consider as bulk.
- **N**: Number of particles. Used only if **read_pos** = `.FALSE.`.
- **R**: Radius of particles.
- **mass**: Mass of particles.
- **read_pos**: `.FALSE.` if the program generates the positions of the particles (only possible for low concentrations); `.TRUE.` if the positions of the particles are read from a file.
- **file_pos**: Route of the file to read the positions of the particles. Used only if **read_pos** = `.TRUE.`.
- **read_vel**: `.TRUE.` if the velocities are read from a file; `.FALSE.` if the program generates the velocities of the particles. The velocity imposed initially on the particles corresponds to the velocity profile of shear rate $\dot{\gamma}$.
- **file_vel**: Route of the file to read the velocities of the particles. Used only if **read_vel** = `.TRUE.`. If **read_vel** = `.FALSE.`, all particle velocities are set to zero.
- **rcut**: Cutoff radius for interactions between particles.
- **rcut_on**: Cut-on radius for interactions between particles.
- **rlist**: Value greater than 1 (and close to 1) that allows calculation of the skin around **rcut** to find neighbors. The radius of the sphere to search for neighbors will be **rlist** × **rcut** (see the section *The Verlet neighbor list* in [1]).
- **tau**: The parameter τ of the repulsive force [3, 8, 10].
- **F0**: The parameter F_0 of the repulsive force [3, 8, 10].

- **fixed_seed**: `.TRUE.` if the seed is selected by the user; `.FALSE.` if the seed is generated automatically.
- **seed**: Seed for the random number generator. Used only if **fixed_seed** = `.TRUE.`.
- **dir_output**: Route of the directory where the program will output data. If the directory does not exist, it will be created by the program.
- **freq_write**: Frequency to output all data except particle data.
- **freq_write_part**: Frequency to output particle data.

2.2 Classes

- **Class computational**: Defines the precision of the real numbers.
- **Class input**: Class to read the input variables from a file.
- **Class read input**: Class with general tools to read variables of many different types from files.
- **Class system**: Contains all the necessary variables for the simulation.
- **Class particle**: Class for particle objects.
- **Class random**: Class for the random number generator.
- **Class output**: Class for output files.
- **Class cell**: Class for cell objects used to search for neighbors (see [1]).
- **Class wall**: Class for wall objects.
- **Class comp_time**: Class for calculating computation times.
- **Class files_utilities**: Collection of several utilities to handle files.
- **Class file**: Class for file objects.
- **Class functions_utilities**: Collection of utilities related to functions.

Aquí está el texto revisado en formato LaTeX:

2.3 Key for Output Files

- *info.dat*: General information about the simulation. The meaning is self-explanatory.
- *input*: A copy of the input file.
- *comp_time.dat*: File with data about the time taken by the code to perform tasks. The columns are:
    1. Time step.
    2. Total computational time.

3. Neighbour searching computational time.
4. Semi-implicit scheme computational time.
5. Velocity-Verlet computational time.

- Initial positions file: If the initial positions have been read from a file, this file will be copied into the output directory.
- Initial velocities file: If the initial velocities have been read from a file, this file will be copied into the output directory.
- *walls.dat*: File with data from walls. The columns in 3D are (in 2D, the columns are fewer):
  1. Time step.
  2. x force on the bottom wall.
  3. y force on the bottom wall.
  4. z force on the bottom wall.
  5. x force on the top wall.
  6. y force on the top wall.
  7. z force on the top wall.
- *stress.dat*: File with the stress tensor $\boldsymbol{S}_{pp}$ calculated from the bulk. The columns in 3D are (in 2D, the columns are fewer):
  1. Time step.
  2. S_{pp}^{xx}.
  3. S_{pp}^{xy}.
  4. S_{pp}^{xz}.
  5. S_{pp}^{yx}.
  6. S_{pp}^{yy}.
  7. S_{pp}^{yz}.
  8. S_{pp}^{zx}.
  9. S_{pp}^{zy}.
  10. S_{pp}^{zz}.
- *particles....dat*: Files with data about particles. The columns in 3D are (in 2D, the columns are fewer):
  1. Particle identity (an integer).
  2. Particle x position.
  3. Particle y position.
  4. Particle z position.
  5. Particle x velocity.
  6. Particle y velocity.
  7. Particle z velocity.

8. Particle radius.

- *shear_rate.dat*: File with the calculated shear rate (which may be smaller than the input shear rate).

  1. Time step.
  2. Calculated shear rate.

2.4 Before Simulating

To create the initial configuration of particles, the algorithm of the code randomly fills the space with non-overlapping particles at random positions. This approach works well for low concentrations, but for concentrated cases, it may not achieve a valid configuration. In such cases, the initial configuration should be provided by the user to the program as input so that the program can read the initial configuration.

The tools to build the initial configuration are located in the directory *to_prepare_initial_configuration*. A Monte Carlo simulation is used for this task. For more information, please refer to the *readme* file in that directory.

3 The model

The model is described in [6].

3.1 Some Calculations

3.1.1 Viscosity from the Wall

Given that the walls are oriented in the z direction, only the components σ_{zx}, σ_{zy}, and σ_{zz} of the stress can be calculated:

$$\begin{aligned}
\sigma_{zx} &= \frac{F^x_{\text{wall}}}{L_x L_y} \\
\sigma_{zy} &= \frac{F^y_{\text{wall}}}{L_x L_y} \\
\sigma_{zz} &= \frac{F^z_{\text{wall}}}{L_x L_y} \qquad (1)
\end{aligned}$$

The viscosity of the suspension is calculated as

$$\eta = \frac{\sigma_{zx}}{\dot{\gamma}} \qquad (2)$$

Note that, in general, the shear rate generated by the wall is not exactly the input shear rate, due to some slip with the walls (see Fig. 1).

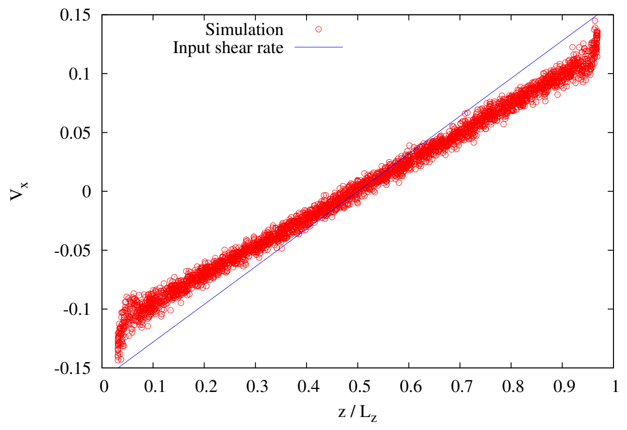

Figure 1: Slip in the simulations for a suspension with concentration $\phi = 0.4$.

3.1.2 Viscosity from the Irving-Kirkwood Method

The stress tensor can also be calculated via the Irving-Kirkwood method [5]. Following this method, the stress tensor is calculated from the positions and forces between particles as [2, 7]:

$$\boldsymbol{\sigma} = \frac{1}{V}\left[\sum_i \boldsymbol{v}_i \boldsymbol{v}_i + \frac{1}{2}\sum_i \sum_{j\neq i} \boldsymbol{r}_{ij}\boldsymbol{F}_{ij}\right] \tag{3}$$

From the stress tensor, the viscosity can be calculated using Eq. (2).

3.1.3 Comparison of Both Methods

Results obtained with both methods are compared in this section. The system consists of a suspension with concentration $\phi = 0.4$ in a shear flow between planar walls. Figure 2 shows the evolution of the ratio $S_{zx}^{\text{wall}}/S_{zx}^{\text{IK}}$ for three different time steps. Although convergence is achieved for $dt = 10^{-4}$ when the stress is calculated from the wall (result not shown here), convergence with the Irving-Kirkwood method requires a time step that is ten times smaller ($dt = 10^{-5}$).

3.1.4 Faster Way to Obtain Irving-Kirkwood Results

It has been observed that the Irving-Kirkwood results for the stress tensor are accurate only for very small time steps. To obtain results more quickly using the Irving-Kirkwood approach, one can run a simulation with a relatively large time step dt. Afterwards, in post-processing, the simulation can be continued for a few steps starting from the collected data until the Irving-Kirkwood result reaches a quasi-steady state. Results from such a post-processing calculation have been compared with those from simulations with smaller time steps, showing almost identical results, as depicted in Figure 3.

To perform the post-processing calculation of the stress tensor, use the script *script_stress.sh* in the scripts directory.

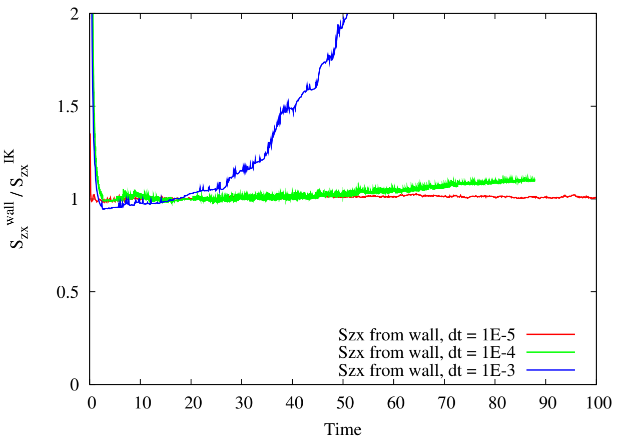

Figure 2: Comparison of the ratio of the component S_{zx} calculated with the wall and the component S_{zx} calculated via the Irving-Kirkwood method.

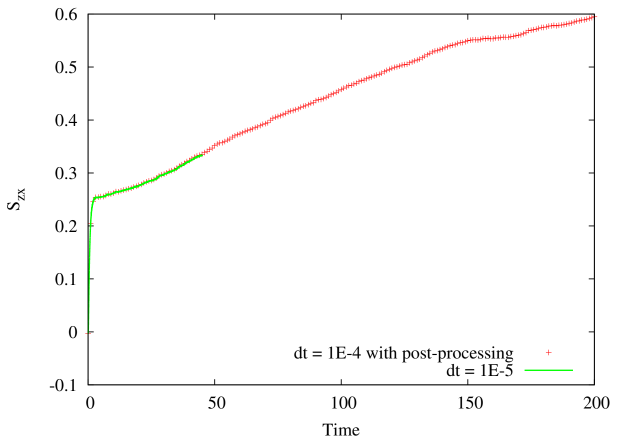

Figure 3: Comparison of the temporal evolution of the S_{zx} component of the stress calculated with the proposed post-processing method and another simulation with a smaller time step.

3.1.5 Calculation of the Normal Stress Differences

The normal stress differences are calculated from the stress tensor as

$$\begin{aligned} N_1 &= \frac{\sigma_{xx} - \sigma_{zz}}{\eta_0 \dot{\gamma}} \\ N_2 &= \frac{\sigma_{zz} - \sigma_{yy}}{\eta_0 \dot{\gamma}} \end{aligned} \tag{4}$$

3.1.6 Viscosity and Normal Stress Differences Versus Concentration

The dependency of viscosity and normal stress differences on the concentration ϕ is shown in Figure 4. The normal stress differences are defined as

$$\begin{aligned} N_1 &= \frac{S_{xx} - S_{zz}}{\eta_0 \dot{\gamma}} \\ N_2 &= \frac{S_{zz} - S_{yy}}{\eta_0 \dot{\gamma}} \end{aligned} \tag{5}$$

Although the agreement is good for the viscosity of the suspension and for N_2, the magnitude of N_1 is significantly larger than that obtained by Bertevas et al. It has been verified that the results for N_1 and N_2 are not sensitive to changes in the cutoff radius and the size of the simulation box (results for N_1 and N_2 shown in the figures are from a simulation box of size $L = 32 \times 32 \times 32$).

References

[1] Mike P Allen and Dominic J Tildesley. *Computer simulation of liquids*. Oxford university press, 1989.

[2] Erwan Bertevas, Xijun Fan, and Roger I Tanner. Simulation of the rheological properties of suspensions of oblate spheroidal particles in a newtonian fluid. *Rheologica acta*, 49(1):53–73, 2010.

[3] Xin Bian and Marco Ellero. A splitting integration scheme for the sph simulation of concentrated particle suspensions. *Computer Physics Communications*, 185(1):53–62, 2014.

[4] Viktor K Decyk, Charles D Norton, and Boleslaw K Szymanski. Introduction to object-oriented concepts using fortran90. 1996.

[5] JH Irving and John G Kirkwood. The statistical mechanical theory of transport processes. iv. the equations of hydrodynamics. *The Journal of chemical physics*, 18(6):817–829, 1950.

[6] SS Prasanna Kumar, A Vázquez-Quesada, and M Ellero. Numerical investigation of the rheological behavior of a dense particle suspension in a biviscous matrix using a lubrication dynamics method. *Journal of Non-Newtonian Fluid Mechanics*, 281:104312, 2020.

[7] N Phan-Thien, N Mai-Duy, and BC Khoo. A spring model for suspended particles in dissipative particle dynamics. *Journal of Rheology (1978-present)*, 58(4):839–867, 2014.

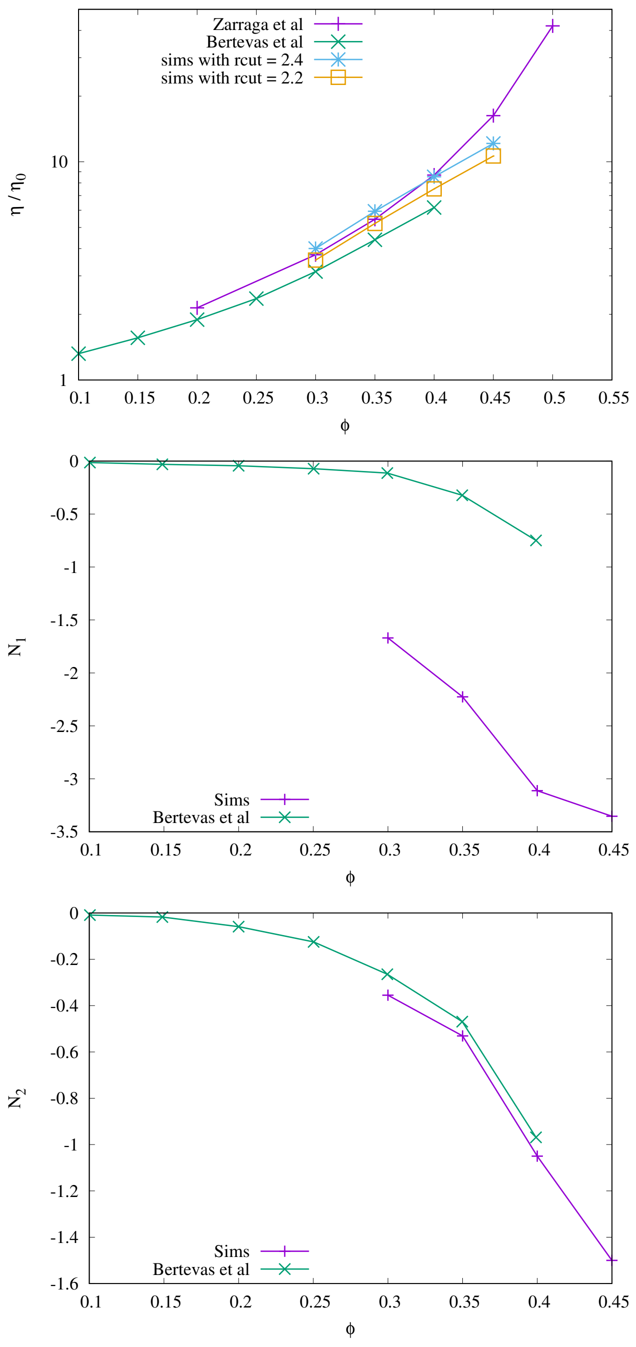

Figure 4: Dependency of viscosity and normal stress differences N_1 and N_2 on the concentration ϕ.

[8] Adolfo Vázquez-Quesada, Xin Bian, and Marco Ellero. Three-dimensional simulations of dilute and concentrated suspensions using smoothed particle hydrodynamics. *Computational Particle Mechanics*, 3(2):167–178, 2016.

[9] Adolfo Vázquez-Quesada and Marco Ellero. Analytical solution for the lubrication force between two spheres in a bi-viscous fluid. *Physics of Fluids*, 28(7):073101, 2016.

[10] Adolfo Vázquez-Quesada and Marco Ellero. Rheology and microstructure of non-colloidal suspensions under shear studied with smoothed particle hydrodynamics. *Journal of Non-Newtonian Fluid Mechanics*, 233:37–47, 2016.