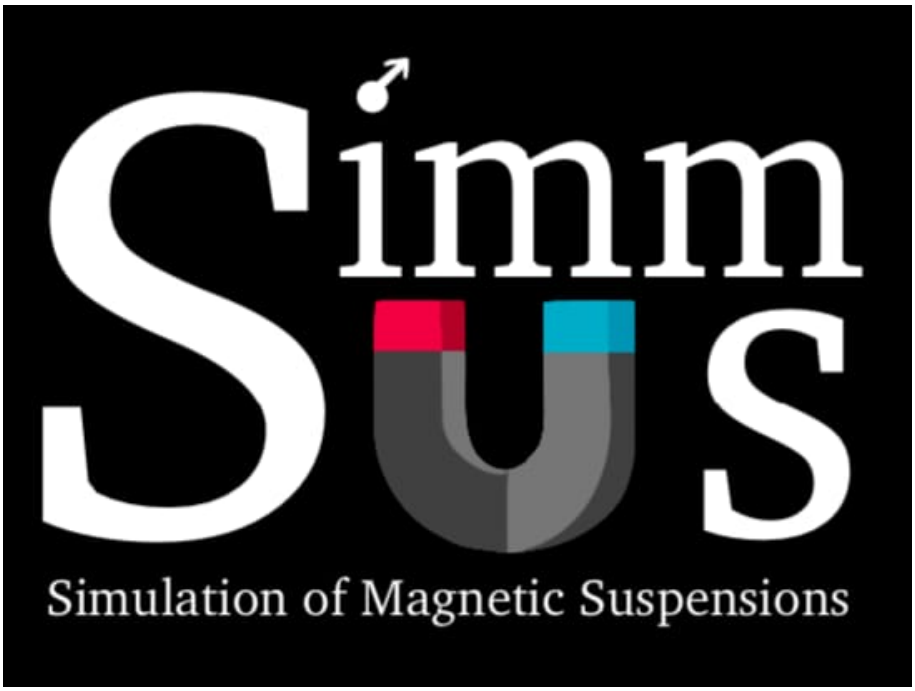
SIMMSUS
Simulation of magnetic suspensions

Developers Manual

August 15, 2023



Contents

1 What is SIMMSUS? 3

2 Why FORTRAN? 4

3 SIMMSUS files 4

3.1 simmsus.f90 . . . 6
3.2 input.f90 . . . 7
3.3 main.f90 . . . 8
3.4 subroutines.f90 . . . 9
3.4.1 Periodic calculations . . . 10
3.4.2 Random numbers . . . 13
3.4.3 Calculation of forces and torques . . . 15
3.4.4 Translational and rotational inertia . . . 16
3.4.5 Initial condition . . . 19
3.4.6 External field excitations . . . 20
3.5 statistics.f90 . . . 22
3.6 variables.f90 . . . 25
3.7 makefile . . . 25
3.8 simconfig.dat . . . 26

4 How to run a simulation? 37

5 Simulation output files 38

6 How to visualize a simulation? 41

7 The equations 45

8 Validation 46

9 Gallery 48

10 Things to improve 51

11 References 52

1 What is SIMMSUS?

SIMMSUS is a research code written in FORTRAN that simulates the motion of a system of interacting particles. These particles can be simulated in different scenarios and may interact through different physical mechanisms. The code was initially developed for studying the physics of magnetic spherical particles in suspensions in order to better understand the properties of magnetic fluids (ferrofluids).

Since the beggining of the development, the code was designed to consider the effects of Brownian motion, long range dipole-dipole and hydrodynamic interactions. The first physics simulated through SIMMSUS was the study of the alignment of the particles in the direction of an applied steady-state magnetic field to see wether the code was capable of capturing the behavior of the equilibrium magnetization predicted by theoretical asymptotic models available on the literature (Ivanov and Kusnetsova, 2001). The comparison between the predictions of SIMMSUS and the asymptotic models served as a first validation of the numerical code.

After this initial validation, we have implemented several additional features on the code and have used SIMMSUS to explore different physical situations regarding the behavior of magnetic fluids. Many of the studies conducted through SIMMSUS have been published in prestigious academic Journals, such as Physics of Fluids, Journal of Magnetism and Magnetic Materials, Powder Technology and Mechanics Research Communications.

In its present version SIMMSUS is capable of simulating Brownian and non-Brownian suspensions and the user can turn on/off specific physical mechanisms in order to build a customized scenario. For example the user can mix magnetic particles with non-magnetic particles, turn on/off the gravitational field, apply an oscillatory or steady-state magnetic field over the particles, apply a simple or oscillatory shear over the particles, simulate different initial configurations (ordered, random or spherical distributions), simulate mono or polydisperse particles (with the same radius or with different radius) and the list goes on.

Regarding the application of time-dependent magnetic fields, we have conducted a rigorous study on the dynamical susceptibility response of ferrofluids using SIMMSUS and have validated the dynamical solution of the rotational motion of the dipole moments of the particles provided by the code by comparing the numerical solution with an asymptotic theoretical model (Berkov et al, 2009). The code seems to predict with excellent precision the dynamical behavior of the internal structure of ferrofluids.

2 Why FORTRAN?

Nowadays few people program in FORTRAN and the fact that SIMMSUS has been written in FORTRAN may sound strange to many young programmers that are more familiar to popular languages like Python. However, it is not a secret that Physicists and Engineers have been using FORTRAN for a long time. FORTRAN is a compilable, intermediary level language that has been evolving with the processors we use to perform our calculations since the 50s. When it comes to solve problems that demand great computational effort FORTRAN seems to be one of the preferred languages among scientists. SIMMSUS deals with a great amount of heavy calculations in order to simultaneously compute multiple simulations with many particles through several time-steps solving highly complex equations. Therefore, we have chosen to write SIMMSUS in FORTRAN because we believe in the power of this language for this kind of scientific applications. Also, SIMMSUS was developed in an academic enviroment. So it was created around a culture where people really like FORTRAN. Since FORTRAN is a compilable language, the performance of the solver is strongly affected by the quality of the compiler used to build the executable file (program). Therefore, we strongly recommend you to compile SIMMSUS with Intel Fortran Compiler in Linux enviroment, since this was the enviroment in which SIMMSUS was programmed and tested so far.

3 SIMMSUS files

The code contains 8 files, from which 7 are necessary to produce the executable file (program) after the compilation and 1 is a configuration file. These are the following files:

1. simmsus.f90
2. input.f90
3. main.f90
4. subroutines.f90
5. statistics.f90
6. variables.f90
7. makefile

8. simconfig.dat

In order to compile the source-code you must open a Linux terminal and run the make command. For the first time you should run the make command twice. In the first time the compiler will produce the modules from the files variables.f90 and subroutines.f90 and the second run will use these recently created modules to produce the executable file simmsus.ex. Since the origins of its earlier development, SIMMSUS has been tested and compiled exclusively on Linux using the Intel Fortran Compiler and up to this version of the manual we have no idea if it compiles or runs using other compilers or operational systems.

In order to run a simulation, the user will need only 2 files: the executable file (simmsus.ex) and the input configuration file simconfig.dat. The user can drag these two files to a new folder to run a new set of numerical simulations defined on the configuration file simconfig.dat. The configuration file is a text file with several questions that the user should answer in order to direct the path that SIMMSUS should cross in order to produce a specific set of simulations for the intended physics. We will talk about this file later. Figure (1) shows how the source-code files are related to each other.

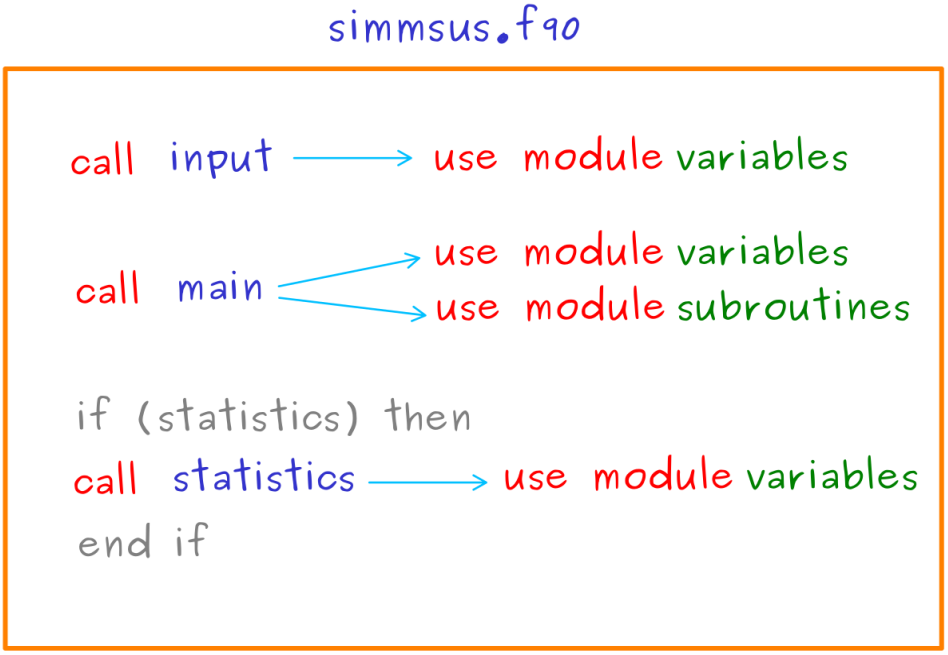


Figure 1: Source files structure

The file simmsus.f90 basically calls other files from the source code struc-

ture in a straightfoward order. The first call is related to the reading of the simulation data. The file input.f90 reads the configuration file simconfig.dat and storage in logical, real and integer variables all the information necessary to run a set of simultaneous configurations. SIMMSUS may run a single simulation or perform several simultaneous numerical experiments varying the initial configuration of the particles and the set of random numbers used to emulate Brownian forces and torques. Therefore, the use may configure in the simconfig.dat file all the informations regarding all the simultaneous numerical experiments. We will talk about the simconfig.dat file later on this manual. It is important to notice that the input.f90 file uses the variables module, written in the variables.f90 file. We will also talk about this module later on this manual.

After reading the configuration file, simmsus.f90 calls the main.f90. This file calls the necessary subroutines in a specific order in order to run the simulations. Here, the dynamical variables are allocated in the computer memory, the initial condition is generated and the main loop occurs. Here the physics is processed and several output files are created along the simulation process. The main.f90 file uses the variables and subroutine modules. The last one contains all the subroutines used on the code.

After the processing stage, simmsus.f90 checks if the user has chosen to perform a statistical analysis of the calculated results and if so calls the statistics.f90 file, which also uses the variables module.

The makefile contains the compilation instructions. Here the name of the source code files, the compilation command (and consequently the compiler) is specified and any optional compilation flags are also defined.

We will talk about the general logic and structure of each of these files in the next subsections of this manual.

3.1 simmsus.f90

This file simply calls the program subroutines in a specific order in order to execute a set of numerical simulations defined by the user in the configuration file (simconfig.dat). Besides calling these subroutine this file also counts the total simulation time and write a simple header that appears on the terminal when the user executes the solver. The header of the release version of the program is shown in figure (2). We ask to the developers to update the date of the last implementations and to add their names as colaborators in the header of simmsus.f90 whenever nem implementations are made. New developers should feel free to alter the esthetics of the header and include new information they feel on the header of the program.

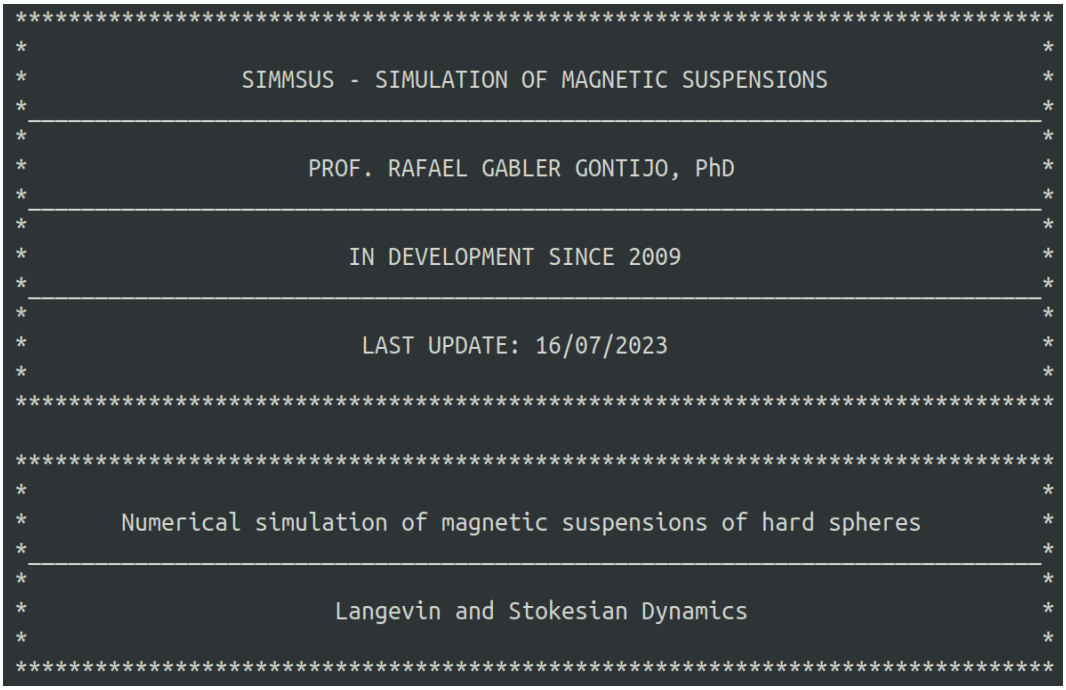


Figure 2: Header of simmsus.f90

3.2 input.f90

This subroutine reads the file simconfig.dat and storage the information in logical, integer and real varaibles. The variables presented in simconfig.dat must respect the formats defined in file input.f90. These formats are defined through the following commands:

505 FORMAT(1X,A40,1X,E11.4E2)
507 FORMAT(1X,A40,1X,I6)
508 FORMAT(1X,A40,L10).

Here the numbers 505, 507, 508 are simply shortcuts used to specify these formats. The format 505 defines a single keyboard space (1X) followed by 40 letters (A40) a single space (1X) and a real variables containing 11 digits expressed in scientific notation with 4 digits after the comma and 2 digits after the symbol E. Here is an example of the 505 format expressed in simconfig.dat:

VOLUME FRACTION OF PARTICLES.....: 1.0000E-02

The other formats, 507 and 508, are related to integer and logical variables respectively. New implementations should be added as options in the simconfig.dat file and altered on the input.f90 file also. The user is free to modify the structure of the menus on the configuration file simconfig.dat, but it is important to change the strucutre of the file input.f90 accordingly.

3.3 main.f90

The main.f90 file sets the order in which calculations are performed in order to achieve a specific set of numerical simulations. In this file the subroutines are sequentially called from the subroutines.f90 module. The heavy calculations are programmed in subroutines.f90.

The sequence of procedures called by main.f90 are presented bellow.

1. Definition of constant internal numerical parameters;
2. Allocation of the matrices in the memory;
3. Definition of number formats for the output files;
4. Cleaning all important variables;
5. Determination of particle size distribution;
6. Calculation of the simulation box size;
7. Creation of output files;
8. Calculation of Green-functions and Lattice indexes structure;
9. Distribution of dipole moments;
10. Generation of the initial condition;
11. Calculation of field excitations;
12. Execution of the main simulation loop;
13. Deallocation of matrices from memory;

The main simulation loop is where the simulation occurs. This loop is composed by the following sequence:

1. Calculation of each force acting on each particle;
2. Solution of the linear momentum equation;

3. Evolution of the position of each particle;

4. Calculation of each torque acting on each particle;

5. Solution of the angular momentum equation;

6. Evolution of the orientation of each particle;

This loop goes on until the total simulation time is reached. It is important to notice that position, velocity and dipole orientation are storaged in large matrices with the following indeces notation:

$$x(i,j,k) \rightarrow i = [1,...,N_{rea}], \;\; j = [1,...,N_{part}], \;\; k = [1,2,3], \tag{1}$$

where N_{rea} denotes the number of numerical experiments (realizations) and N_{part} the number of particles. The last column related to the index k represents the number of spacial dimensions. This way, the number of variable $X(12, 1234, 2)$ represents the position of particle 1234 in the 12^{th} numerical experiment in direction 2. Directions $(1, 2, 3)$ are respectively (x, y, z). Since SIMMSUS also deals with the modeling of Brownian systems, in which a stochastic forcing is always present, the code was built to perform several simultaneous numerical experiments so a meaningful statistics could be obtained for a large number of independent numerical experiments. However, the user is also free to perfom a single numerical experiment. The number of numerical experiments as well as the number of particles in each simulation set is configured in the simconfig.dat file. The loops regarding the execution of all the calculations with respect to each numerical experiment are paralelized through openMP. For this purpose the user must set the flag -qopenmp in the makefile (if using INTEL FORTRAN COMPILER).

3.4 subroutines.f90

The inteligence of SIMMSUS is programmed in this module. Here, it is where all the heavy calculations are really performed. This file contains all the calculation routines implemented in the code, which are called by other files, such as main.f90 and statistics.f90. We won't detail all the subroutines contained in this file here, since many of them are really simple and intuitive and all important details regarding SIMMSUS subroutines are commented in the code. However, some subroutines are more complex than others and we believe some discussion regarding some of them are important for the developer to understand some aspects of the logic of SIMMSUS.

3.4.1 Periodic calculations

In order to understand the logic of SIMMSUS it is important to understand a few things. First of all, the calculations of forces, torques and particle's velocities can be perfomed in a periodic or non-periodic structure. Here, we are not referring to the boundary conditions of the simulation cell, which are always periodic, but to the way in which these quantities are calculated. From the physics of multibody interacting systems it is known that long-range particle interactions have a certain decay. For example, the gravitational force between two masses decay with $1/r^2$, where r is the distance between the center of the bodies. A typical $1/r^n$ decay is said to be slow if $n \leq 3$. In this sense slow decays are felt from a great distance from the source which generates the disturbances in the surroundings. These slow decays lead to a famous convergence problem of the calculation of average properties of these systems, which demands a huge amount of particles to produce stable, convergent values of the transport properties of these suspensions. One way to solve this problem is to emulate an infinite suspension of particles using a periodic representation of the particulate system strucute called Lattice. A Lattice is a representation of a periodically replicated strucutre containing the central cell (the simulation box itself) and imaginary surrounding cells that mimics the configuration of the central cell. A visual representation of a Lattice strucutre is shown in figure (3).

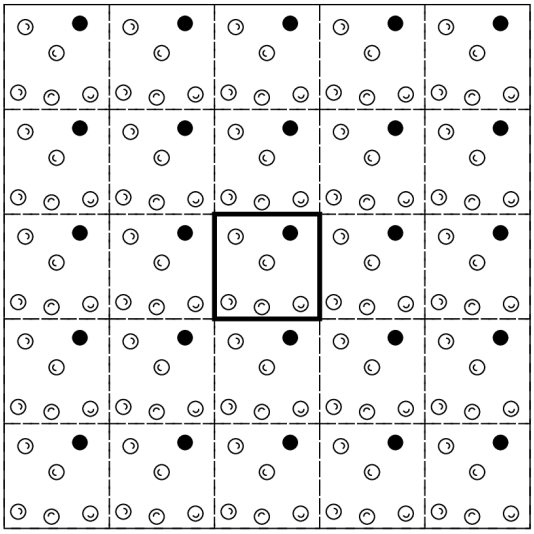


Figure 3: Visual representation of a Lattice structure

Using a Lattice representation it is possible to imagine what the black particle shown in figure (3) would see when looking to any direction: an

infinite system with no beggining, no end and no walls. The use of a Lattice structure leads to a convergence of the system transport properties for long-range slow decays, but increases substantially the computational time, since now each particle on the central cell must interact with their neighbours in the real simulation box and with all the particles in the imaginary boxes surrounding the real one in the Lattice structure. Also, when we consider particle-particle interactions in a system with N_{part}, interacting forces or torques are generally expressed in terms of a series in the form:

$$\boldsymbol{F}^i = \sum_{i \neq j} \frac{f(r_i, r_j)}{r_{ij}^n}, \tag{2}$$

where r_{ij} denotes the distance between particles i and j and $f(r_i, r_j)$ is a function of the system configuration at the instant of time in which the calculation is being performed. These functions differ from each kind of physical mechanism involved and are generally known as Green functions. These functions arive from physical principles and have to be modified using a sophisticated mathematical technique known as the Ewald (1921) summation to be expressed in a Lattice structure.

In SIMMSUS the user can define wether the periodic calculation is being performed for long-range dipolar interactions for both forces and torques. This definition occurs on the configuration file simconfig.dat. Generally, these periodic calculations are only necessary in non-dilute conditions. Usually for particle volume fractions above $\phi \geq 10\%$, but the user should consult the following reference to understand the limits in which non-periodic calculations of magnetic forces and torques should produce precise results.

- GONTIJO, R.G.; CUNHA, F.R. . Dynamic numerical simulations of magnetically interacting suspensions in creeping flow. Powder Technology (Print), v. 279, p. 146-165, 2015;

SIMMSUS also considers the effect of hydrodynamic interactions in Creeping flow, which are calculated using a mobility formulation where the particles velocities are directly linked with the non-hydrodynamic forces acting on each of them through equation (3).

$$\begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \vdots \\ \boldsymbol{u}_N \end{pmatrix} = \begin{pmatrix} \boldsymbol{M}^{11} & \boldsymbol{M}^{12} & \cdots & \boldsymbol{M}^{1N} \\ \boldsymbol{M}^{21} & \boldsymbol{M}^{22} & \cdots & \boldsymbol{M}^{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{M}^{N1} & \boldsymbol{M}^{N2} & \cdots & \boldsymbol{M}^{NN} \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \\ \vdots \\ \boldsymbol{f}_N \end{pmatrix}, \tag{3}$$

where $\boldsymbol{u}_1$, $\boldsymbol{u}_2$, ..., $\boldsymbol{u}_N$ denotes the velocities of particles $1, 2, \ldots, N$. The tensors $\boldsymbol{M}^{ij}$, for $i = 1, \ldots, N$ and $j = 1, \ldots, N$, are second rank mobility tensors or square matrices that depend on the suspension configuration. These tensors couple the motion of particles given how the forces acting on a particle j change the velocities of a particle i. Here, $\boldsymbol{f}_i$ represents the sum of non-hydrodynamic forces acting on a particle i. Hence, when the user sets that hydrodynamic interactions must be computed in simconfig.dat, the calculations of the particle's velocities are automatically performed in a period way. For more details regarding the mobility formulation used in SIMMSUS the reader should consult the following reference:

- Gontijo, R. G.; CUNHA, F. R. . Numerical simulations of magnetic suspensions with hydrodynamic and dipole-dipole magnetic interactions. PHYSICS OF FLUIDS, v. 29, p. 062004, 2017.

Whenever the user defines one or more of the following variables as true

- ACCOUNT HYDRODYNAMIC INTERACTIONS;
- PERIODIC MAGNETIC TORQUES;
- PERIODIC MAGNETIC FORCES,

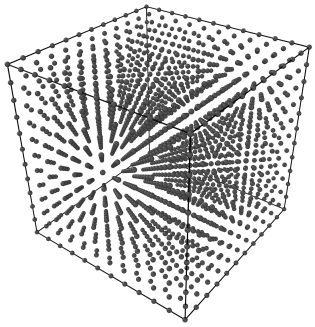
SIMMSUS ativates the following subroutines related to periodic computations:

- tabelagreen $\rightarrow$ computes the necessary Green functions and storages them in large pre-calculated tables;
- estrutura_periodica $\rightarrow$ creates the indexes of cells in the Lattice strucutre;
- periodic_interactions $\rightarrow$ computes all periodic interactions between the particles at a given time-step (this is the most expensive calculation performed by SIMMSUS);

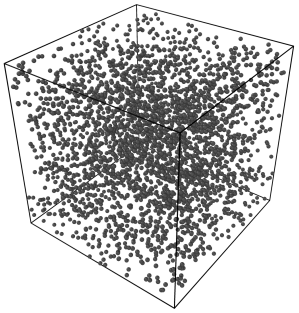
It is important to mention that when the user enables **account hydrodynamic interactions** in the file simconfig.dat SIMMSUS automatically considers a mobility formulation, which neglects the effect of particle inertia. Therefore, when considering hydrodynamic interactions we recommend the user to disable variable **particle inertia** in simconfig.dat.

3.4.2 Random numbers

In order to run the particle simulations random numbers must be generated. These random numbers are important to create random independent initial conditions and to compute Brownian forces and torques. Figure (4) shows typical initial particle distributions that can be produced by SIMMSUS. Figure (4a) shows an initial ordered distribution of particles while figure (4b) illustrates a random initial distribution of particles.



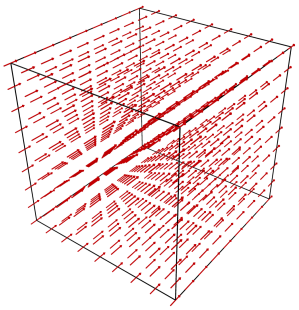
(a) ordered particle distribution



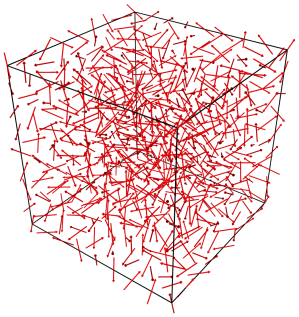
(b) random particle distribution

Figure 4: Typical initial particle distributions generated by SIMMSUS

Not only the initial particle distribution can be ordered or random, but also the initial dipole orientation, as shown in figure (5).



(a) ordered dipole orientation distribution



(b) random dipole orientation distribution

Figure 5: Typical initial dipole orientation distribution generated by SIMMSUS

As we have mentioned, random numbers must also be generated in order

to simulate the typical random walks produced by a particle in Brownian motion, as shown in figure (6).

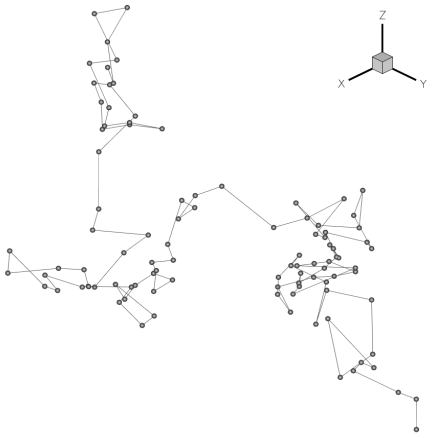


Figure 6: Typical random walk of a particle in Brownian motion

The subroutine responsible for generating the sequences of random numbers used in a typical simulation is called **randomica**. This subroutine is defined in terms of the following arguments:

randomica(a,b,c,n,d), where a,b,c,n,d denote

- a,b $\rightarrow$ the range of the random sequence;
- c $\rightarrow$ is a vector containing the random sequence generated;
- n $\rightarrow$ is the number of elements in vector c;
- d $\rightarrow$ is an integer used to produce a random seed for the random sequence that is being produced;

We have opted for implementing a customized subroutine for generating the random numbers used in our simulations instead of using the native random_number FORTRAN subroutine. We have chosen this option due to several statistical tests performed in the beggining of the development of SIMMSUS used to validate the implementation of Brownian forces and torques. We have compared the ensemble average of the mean square displacement of a single Brownian particle as a function of time with Einstein's theory and have found an excellent agreement using the implemented random number generation routine in SIMMSUS. For more details the user can consult the following reference:

- Micromechanics and microhydrodynamics of magnetic suspensions, PhD Thesis, University of Brasília, Rafael Gabler Gontijo (in portuguese).

3.4.3 Calculation of forces and torques

In SIMMSUS each type of force and torque is calculated using a specific subroutine. The following forces are considered in SIMMSUS:

- Long-range dipolar forces between the particles;
- Short-range repulsive forces between approaching particles;
- Contact forces for slightly overlapped particles;
- Brownian and gravitational forces;

With respect to the torques acting on the particles, SIMMSUS considers

- Long-range dipolar torques between the particles;
- Field-dipole torques between the particles;
- Brownian torques;

Forces regarding field-dipole interaction are implemented in SIMMSUS through subroutine campo_externo, but are disabled in the present version of the code. It is known that the magnetic force acting on a dipole is proportional to the magnetic field gradient and since we have been using SIMMSUS to study problems where the simulation box is a continuum volume in which magnetic field gradients are very small we have been neglecting the force acting on a dipole due to an external magnetic field. Anyway, the user is free to alter this option in the source-code of the program whenever he/she wants.

The subroutines responsible for computing magnetic, repulsive, contact and gravitational forces are respectively

- forca_magnetica;
- respulsion;
- gravity;

Subroutine forca_magnetica computes long-range dipolar forces between the particles in a non-periodic way. Subroutine repulsion computes both short-range repulsive and contact forces between the particles. Brownian forces and torques are both computed by the same subroutine, namely brownian. Magnetic torques between the particles are computed in a non-periodic way by subroutine torque_magnetico. Periodic magnetic forces and torques are computed inside subroutine periodic_interactions. For details regarding the expressions used to compute long-range periodic magnetic forces and torques the reader should consult the following reference:

- GONTIJO, R.G.; CUNHA, F.R. . Dynamic numerical simulations of magnetically interacting suspensions in creeping flow. Powder Technology (Print), v. 279, p. 146-165, 2015;

3.4.4 Translational and rotational inertia

When hydrodynamic interactions are neglected, SIMMSUS gives two possibilities for computing the velocity of the particles: the first one considers particle inertia, while the second one neglects it. The physical principle behind the equation used to solve the velocities of the particles is Newton's second law, for both the linear and angular velocity of each particle. In other words, the equations used for the solution of the motion of the particles are:

$$m\frac{d\boldsymbol{v}}{dt} = \sum \boldsymbol{F}, \tag{4}$$

$$I\frac{d\boldsymbol{\omega}}{dt} = \sum \boldsymbol{T}, \tag{5}$$

where m and I denotes respectively the mass and the inertia moment of a single particle, $\boldsymbol{v}$ and $\boldsymbol{\omega}$ represent the linear and angular velocities of the particle, t denotes time and $\boldsymbol{F}$ and $\boldsymbol{T}$ represent respectively the forces and torques acting on a single particle. In the abscence of hydrodynamic interactions SIMMSUS considers a simple Stokes drag as the hydrodynamic force acting on each particle, this force is given in dimensional terms as $\boldsymbol{F_D} = -6\pi\eta a$, where η and a represent the fluid's viscosity and the radius of a single particle. It is important to keep in mind that the Stokes drag is only valid in Creeping-flow regime ($Re \ll 1$) and that is a basic premisse behind the mathematical equation of SIMMSUS. Now, equations (6) and (7) are solved in SIMMSUS in their nondimensional version, given by

$$St\frac{d\boldsymbol{v^*}}{dt^*} = \sum \boldsymbol{F^*}, \tag{6}$$

$$St_r \frac{d\boldsymbol{\omega}^*}{dt^*} = \sum \boldsymbol{T}^*, \tag{7}$$

where the asterisks denotes nondimensional variables and St and St_r denote respectively the Stokes number and its correspondent rotational version. These physical parameters are defined as:

$$St = \frac{mU_s}{6\pi\eta a^2} \qquad \text{and} \qquad St_r = \frac{IU_s}{8\pi\eta a^4}, \tag{8}$$

where U_s denotes the Stokes velocity of an isolated particle. Equations (6) and (7) were obtained considering typical velocities and timescales related to the Stokes velocity, which seems to be a good choise for sedimentation problem. Other possible scales generate different versions of the nondimensional equations solved by the code. SIMMSUS has three possible choices of nondimensional equations implemented. Anyway, the important fact here is that the Stokes number provides a relation between the relaxation timescale of the particle with respect to a viscous dissipation timescale. For really small particles it is valid to assume $St \ll 1$, which leads us to neglect the effect of particle inertia. For inertialess (non-massive) particles in Creeping flow it is possible to obtain the velocity of a single particle by isolating $\boldsymbol{v}$ from the expression of $\boldsymbol{F_D}$ and balancing this hydrodynamic force with other non-hydrodynamic forces. This is the context assumed by SIMMSUS when the user sets variable **particle inertia** as **false** in the configuration file **simconfig.dat**. When neglecting particle inertia and hydrodynamic interactions we assume that hydrodynamic forces will balance non-hydrodynamic forces. Since the $\boldsymbol{F_D} \sim \boldsymbol{v}$, in this context $\boldsymbol{v}$ is calculated in a dimensional form through:

$$\boldsymbol{v} = \left(\frac{\boldsymbol{I}}{6\pi\eta a} \right) \cdot \boldsymbol{F_{NH}}, \tag{9}$$

where $\boldsymbol{I}$ is the second-rank identity tensor and $\boldsymbol{F_{NH}}$ denotes the sum of non-hydrodynamic forces. These non-hydrodynamic forces are long-range dipolar forces, Brownian, gravitational, repulsive and contact forces. It is interesting to notice here that expression (9) could be written in a compact form as

$$\boldsymbol{v} = \boldsymbol{M_s} \cdot \boldsymbol{F_{NH}}, \tag{10}$$

where the self-mobility matrix $\boldsymbol{M_s}$ is simply $\boldsymbol{I}/(6\pi\eta a)$. It is interesting to mention that even though we could be interested in simulating the behavior of really small particles, which could be considered as inertialess particles, the consideration of a small effect of particle inertia could be interesting for numerical purposes, since a non-null Stokes number produces a certain response time of the particles with respect to the forces acting on them.

This small effect of particle inertia may be useful in order to reduce the noise provenient from Brownian motion and produce a cleaner numerical behavior. Previous tests have shown that this seems to be the case for the rotational movement of the particles. Therefore, the solution of the rotational motion of the particles in SIMMSUS always considers a small effect of particle inertia. A good value of the rotational Stokes number is $1.0E-01$ which can be found on the beggining of file main.f90. For more information regarding the role of rotational inertia on the numerical behavior of magnetic suspensions using SIMMSUS we suggest the following reference:

- GONTIJO, R.G.. A numerical perspective on the relation between particle rotational inertia and the equilibrium magnetization of a ferrofluid. Journal of Magnetism and Magnetic Materials, v. 434, p. 91-99, 2017.

The subroutines responsible for solving the velocities of massive (with inertia) particles, their positions, angular velocities and evolving their dipole moments are respectively:

- resvel;
- respos;
- resomega;
- evoldip;

The arguments in each of these subroutines are:

resvel(a,b,c,d), where **a** is the velocity component of a given particle in a given numerical experiment, **b** denotes the numerical time-step, **c** represents the Stokes number of the particle and **d** is the sum of all forces acting on the particle in a given direction;

respos(a,b,c), where **a** is the position in a given direction of a given particle in a given numerical experiment, **b** denotes the numerical time-step and **c** is the associated particle velocity;

resomega(a,b,c,d), where **a** is the rotational velocity component of a given particle in a given numerical experiment, **b** denotes the numerical time-step, **c** represents the rotational Stokes number of the particle and **d** is the sum of all torques acting on the particle in a given direction;

evoldip(a,b,c,d,e,f), where **a,b,c** are the particles dipole moments in each direction, **d** and **e** are the angular velocities of the particle in the directions of the dipole moments of the same diretions as the dipole moments **b** and **c** and **f** is the numerical time-step;

The evolution of the dipole moment of the particles is obtained through the solution of the simple kinematic equation:

$$\frac{d}{dt}(\hat{d}) = \boldsymbol{\omega} \times \hat{d}, \tag{11}$$

where $\hat{d}$ is the direction of the particle's dipole moment. Hence, SIMMSUS always assumes that the dipole moment of the particle is fixed on the particle.

3.4.5 Initial condition

SIMMSUS works with three different types of initial particle distributions: random, ordered or spherical aggregates. Random and ordered distributions have been shown in figure (4) of this manual. The initial spherical aggregate condition is shown in figure (7).

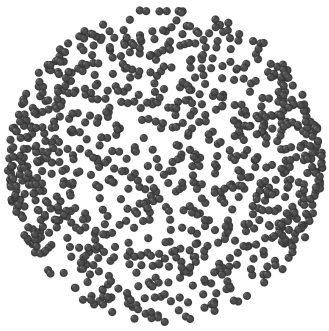


Figure 7: Initial spherical aggregate particle distribution

The subroutines responsible for creating the initial condition are:

- particle_distribution;
- box_size;
- distribui_dipolo;
- condicao_inicial;

3.4.6 External field excitations

In SIMMSUS the user can choose different ways of applying an external excitation on the multibody magnetic system of particles. When we refer here to *external field excitations* we are refering to an external applied magnetic field and/or an external applied shear rate on the system. The user can combine both kinds of excitations, which leads to very interesting dynamical responses. These excitations are calculated in the subroutines **field_excitations**, **rotating_field** and **torque_externo**. Each of these subroutines are responsible for a specific set of possible excitations as follows:

- **field_excitations**: is used for implementing a nonlinear applied magnetic field based on the solution of the Duffing's harmonic oscillator, a double frequency excitation and a dynamical sweep of a single frequency excitation where the system periodically increases its excitation frequency. The last alternative is also implemented for the shear rate within this sub-routine;
- **rotating_field**: is used for implementing a rotating 2D magnetic field;
- **torque_externo**: is used to calculate the magnetic torques on the particles arising from an external field. This subroutine is used for all the 1D applied fields, which include a steady-state and a simple oscillatory field;

Bellow we list the equations responsible for each of the implemented field excitations on SIMMSUS.

1. Oscillatory field $\rightarrow \boldsymbol{H}(t) = H_0 \sin(\omega t)\hat{e}_z$;
2. Double period oscillatory field $\rightarrow \boldsymbol{H}(t) = H_0 \left[\sin(\omega_1 t) + \sin(\omega_2 t)\right] \hat{e}_z$;
3. Rotating field $\rightarrow \boldsymbol{H}(t) = H_0 \sin(\omega t)\hat{e}_y + \cos(\omega t)\hat{e}_z$;
4. Nonlinear Duffing oscillator $\rightarrow \boldsymbol{H}(t) = f(t)\hat{e}_z$ where $f(t)$ is the solution of the following ordinary differential equation (ODE):

$$\frac{d^2 f}{dt^2} + C_1 \frac{df}{dt} + C_2 f + C_3 f^3 = C_4 \cos(\omega t); \tag{12}$$

Variables ω, ω_1, ω_2, C_1, C_2, C_3 and C_4 are all defined by the user in the configuration file **simconfig.dat**.

Figure (8) shows the excitation signals for an oscillatory (8a), double period oscillatory (8b) and nonlinear Duffing field excitation for different

values of parameter C_1 (8c) and (8d). Figure (9) illustrates the excitation signal in the case of a dynamical sweep of the fields frequency, which can also be enabled and configured by the user in **simconfig.dat** file.

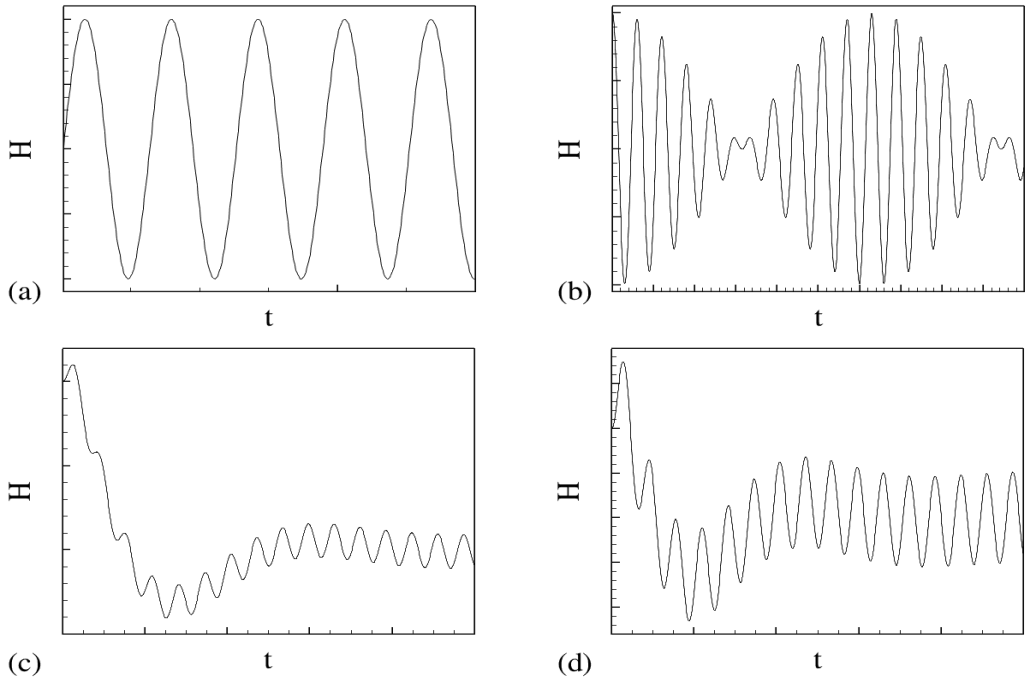


Figure 8: Figure (a) illustrates a simple oscillatory magnetic field. Figure (b) shows a double period excitation; Figures (c) and (d) show a nonlinear Duffing oscillator excitation with different constants.

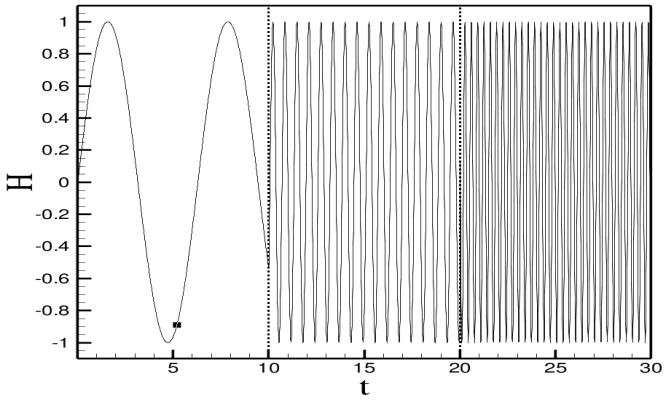


Figure 9: Illustration of the fields dynamical frequency sweep.

3.5 statistics.f90

SIMMSUS has also a statistical analysis module implemented on file **statistics.f90**. This file is called whenever the user enables **statistical analysis** as **true** on file **simconfig.dat**. In order for SIMMSUS to perform this statistical analysis the user must enable **record velocity on file** as true on **simconfig.dat**. The statistical analysis module basically reads all the velocity components of all particles in all realizations from the output files named **velocidade___.plt** and calculates several important statistical quantities based on ensemble averages of relevant physical variables. These calculations produce new output files containing the calculated variables. The statistical quantitites calculated are listed bellow.

- **Average velocity**: this variable is saved on file **velocidade_media.plt** and represents an ensemble average of all velocities of all particles in all realizations at all the different time-steps. This quantity can be defined as:

$$\overline{v_x}(t) = \frac{1}{N_{part}} \frac{1}{N_{rea}} \sum_{j=1}^{N_{part}} \sum_{i=1}^{N_{rea}} v_x(i,j,t) \tag{13}$$

$$\overline{v_y}(t) = \frac{1}{N_{part}} \frac{1}{N_{rea}} \sum_{j=1}^{N_{part}} \sum_{i=1}^{N_{rea}} v_y(i,j,t) \tag{14}$$

$$\overline{v_z}(t) = \frac{1}{N_{part}} \frac{1}{N_{rea}} \sum_{j=1}^{N_{part}} \sum_{i=1}^{N_{rea}} v_z(i,j,t), \tag{15}$$

  where N_{part} denotes the number of particles and N_{rea} the number of numerical experiments. Figure (10) shows a typical behavior of the average velocity of a suspension of sedimenting Brownian particles calculated by SIMMSUS:

- **Variance (Reynolds stress) tensor**: this variable is saved on file **variancia.plt** and represents . This quantity can be defined as:

$$\sigma_{xx}(t) = \frac{1}{N_{part}} \frac{1}{N_{rea}} \sum_{j=1}^{N_{part}} \sum_{i=1}^{N_{rea}} \left[v_x(i,j,t) - \overline{v_x}(t)\right] \left[v_x(i,j,t) - \overline{v_x}(t)\right] \tag{16}$$

$$\sigma_{xy}(t) = \frac{1}{N_{part}} \frac{1}{N_{rea}} \sum_{j=1}^{N_{part}} \sum_{i=1}^{N_{rea}} \left[v_x(i,j,t) - \overline{v_x}(t)\right] \left[v_y(i,j,t) - \overline{v_y}(t)\right] \tag{17}$$

$$\sigma_{xz}(t) = \frac{1}{N_{part}} \frac{1}{N_{rea}} \sum_{j=1}^{N_{part}} \sum_{i=1}^{N_{rea}} \left[v_x(i,j,t) - \overline{v_x}(t)\right] \left[v_z(i,j,t) - \overline{v_z}(t)\right] \tag{18}$$

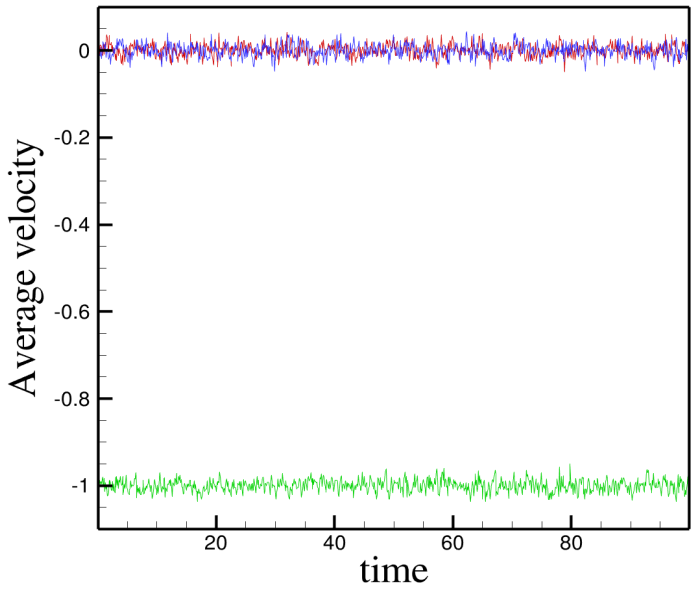


Figure 10: Average velocity of a suspension of 500 sedimenting Brownian particles taken over 100 independent numerical experiments. The green line denotes the component on the direction of gravity while the red and blue lines represent the transversal components.

$$\sigma_{yy}(t) = \frac{1}{N_{part}} \frac{1}{N_{rea}} \sum_{j=1}^{N_{part}} \sum_{i=1}^{N_{rea}} \left[v_y(i,j,t) - \overline{v_y}(t)\right] \left[v_y(i,j,t) - \overline{v_y}(t)\right] \quad (19)$$

$$\sigma_{yz}(t) = \frac{1}{N_{part}} \frac{1}{N_{rea}} \sum_{j=1}^{N_{part}} \sum_{i=1}^{N_{rea}} \left[v_y(i,j,t) - \overline{v_y}(t)\right] \left[v_z(i,j,t) - \overline{v_z}(t)\right] \quad (20)$$

$$\sigma_{zz}(t) = \frac{1}{N_{part}} \frac{1}{N_{rea}} \sum_{j=1}^{N_{part}} \sum_{i=1}^{N_{rea}} \left[v_z(i,j,t) - \overline{v_z}(t)\right] \left[v_z(i,j,t) - \overline{v_z}(t)\right], \quad (21)$$

note that each component of the symmetric second-rank tensor $\boldsymbol{\sigma}$ is the component of the particle Reynolds stress tensor $\sigma_{ij} = v_i^{'} v_j^{'}$, where $v^{'}$ denotes the velocity fluctuation of the system at a given instant of time. Figure (11) shows a typical behavior of the variance of a suspension of sedimenting Brownian particles calculated by SIMMSUS:

- **Time correlation function:** This variable is saved on file **autocorrelacao.plt** and can be defined as:

$$R_x(\tau) = \frac{\left\langle v_x^{'}(t) v_x^{'}(t+\tau) \right\rangle}{\left\langle v_x^{'}(t) v_x^{'}(t) \right\rangle}, \quad (22)$$

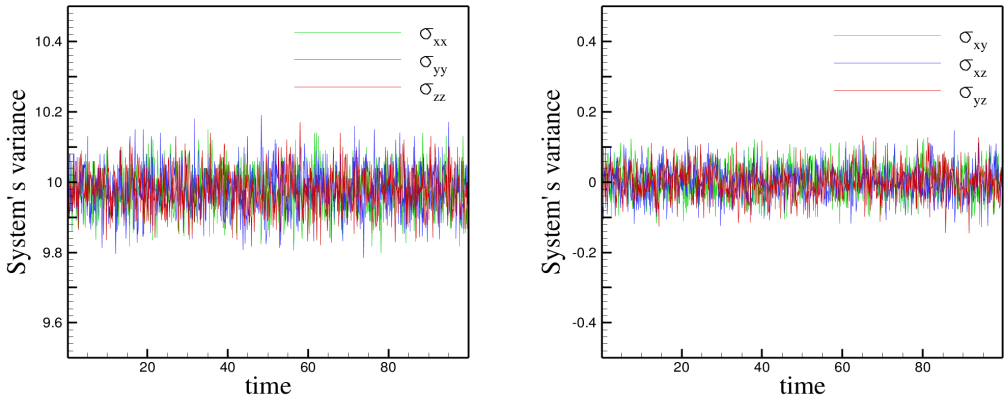


Figure 11: System's variance of a suspension of 500 sedimenting Brownian particles taken over 100 independent numerical experiments. Left side represents the diagonal components of the particle Stress Reynolds while the right side denote the non-diagonal components.

$$R_y(\tau) = \frac{\left\langle v_y'(t) v_y'(t+\tau) \right\rangle}{\left\langle v_y'(t) v_y'(t) \right\rangle}, \tag{23}$$

$$R_z(\tau) = \frac{\left\langle v_z'(t) v_z'(t+\tau) \right\rangle}{\left\langle v_z'(t) v_z'(t) \right\rangle}, \tag{24}$$

where $\langle \quad \rangle$ denotes ensemble averages over all realizations. The time correlaction function measures the correlation degreee between the particles velocity fluctuations through their dynamical evolution process;

- **System's correlation time:** This variable provides the correlation time of the system. It is saved on file **tempo_correlacao.plt** and can be defined as:

$$t_x^c = \int_0^{t_{sim}} R_x(\tau) d\tau \tag{25}$$

$$t_y^c = \int_0^{t_{sim}} R_y(\tau) d\tau \tag{26}$$

$$t_z^c = \int_0^{t_{sim}} R_z(\tau) d\tau, \tag{27}$$

where t_{sim} denotes the simulation time. Figure (12) shows a typical behavior of the time correlation function and system's correlation time of a suspension of sedimenting Brownian particles calculated by SIMMSUS:

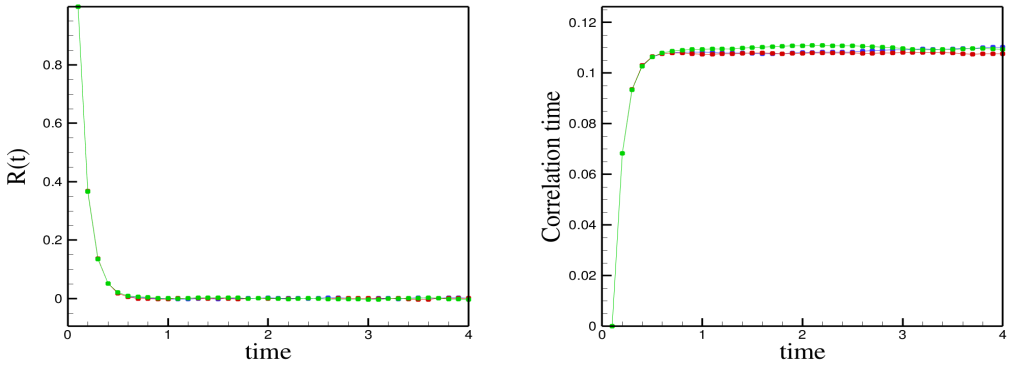


Figure 12: Time correlation function (left) and system's time correlation (right) of a suspension of 500 sedimenting Brownian particles taken over 100 independent numerical experiments. The green line denotes the component on the direction of gravity while the red and blue lines represent the transversal components.

3.6 variables.f90

This file defines the global variables used in SIMMSUS. We ask for the developers to maintain an organized version of this file including a legend for each variable in the form of comments bellow the definition of the global variables of the problem.

3.7 makefile

This file is responsible for the source code compilation. Historically, throughout SIMMSUS development we have tested it using the Intel Fortran Compiler for Linux. Therefore, compilation directives for SIMMSUS in its present version automatically consider that the user is compiling SIMMSUS using Intel Fortran Compiler. Moreover, we use parallelized loops for the simultaneous numerical experiments. This paralelization strategy considers the use of OpenMP, therefore the user may notice the -qopenmp flag no the makefile. We also consider that the code will be run in 64 bits processors and this information must appear on a specif flag on the makefile also. Figure (13) shows a typical example of the makefile structure necessary for compiling SIMMSUS files. In order to compile the code and produce an executable file the user must open a terminal in the folder where the files are located and type make and press enter twice. In the first time an error message will appear due to the inexistence of the module files, which are created after the user press make and type enter for the first time. In the second compilation

attempt the error message will no long appear and the user will notice the creation of an executable file named simmsus.ex.

```
#-----------------------------------------------------#
#       COMPILATION INSTRUCTIONS FOR SIMMSUS          #
#                                                     #
#               Written initially by                  #
#               Rafael Gabler Gontijo                 #
#                                                     #
# Goal: to compile a given version of SIMMSUS         #
#-----------------------------------------------------#
#
# Code source files (Linux)
#
SRC-LNX = subroutines.f90 variables.f90 input.f90  \
          main.f90 statistics.f90 simmsus.f90  \

 OBJ-LNX = $(SRC-LNX:.f90=.o)

# Compiler definition and flags

ENGINE-LNX = ifort
FLAGS-LNX = -m64 -O2 -qopenmp

# Executable file generation (Linux)

simmsus.ex : $(SRC-LNX)
        $(ENGINE-LNX) $(FLAGS-LNX) -o simmsus.ex $(SRC-LNX)

# Cleaning command
clean :
        rm simmsus.ex
```

Figure 13: Make file structure

3.8 simconfig.dat

This is the configuration file in which the user will define what scenario should be simulated by changing the data displayed in this file. Here we will pass through each variable displayed in the configuration file. The header of **simconfig.dat** is shown in figure (14). We ask for developers to maintain the structure of the header. We remind that any modifications on the structure of the configuration file should be coupled with modifications on file **input.f90**.

```
************************************************
*                                              *
*  SIMMSUS:  SIMULATION CONFIGURATION FILE     *
*                                              *
************************************************
```

Figure 14: simconfig.dat header

Figure (15) shows the first round of questions displayed in the configuration file.

```
- SIMULATION LOGICAL VARIABLES

SOLVE TORQUE EQUATION.................: FALSE
MAGNETIC PARTICLES....................: FALSE
STATIC (T) OR DYNAMIC (F) SIMULATION...: FALSE
SEDIMENTATION.........................: TRUE
FLUCTUATION MODE......................: FALSE
BROWNIAN MOTION.......................: TRUE
ACCOUNT HYDRODYNAMIC INTERACTIONS.....: FALSE
CONTINUE AN OLD SIMULATION............: FALSE
PARTICLE INERTIA......................: TRUE
```

Figure 15: Simulation logical variables menu on simconfig.dat

Bellow we detail what each variable means.

- **SOLVE TORQUE EQUATION:** if **TRUE** then SIMMSUS solves the rotational motion of the particles, if **FALSE** then the particles are assumed to be torque free;
- **MAGNETIC PARTICLES:** if **TRUE** then SIMMSUS considers the presence of magnetic forces and/or torques between the particles, if **FALSE** then the particles are assumed to be non-magnetic;
- **STATIC (T) OF DYNAMIC (F) SIMULATION:** if **TRUE** then SIMMSUS considers a unique time-step evolution of the particles (useful for Monte-Carlo simulations), if **FALSE** then the particles evolve dynamically until the end of the simulation time;
- **SEDIMENTATION:** if **TRUE** then SIMMSUS considers the gravitational force, if **FALSE** then the particles are assumed to be neutrally-buoyant;
- **BROWNIAN MOTION:** if **TRUE** then SIMMSUS considers the presence of Brownian forces and/or torques, if **FALSE** then SIMMSUS simulate a non-colloidal suspension of particles;
- **ACCOUNT HYDRODYNAMIC INTERACTIONS:** if **TRUE** then SIMMSUS considers the presence of hydrodynamic interactions between the particles through a mobility formulation as described before in subsection 3.4.1, if **FALSE** then the only hydrodynamic force considered is the Stokes drag around each particle;

- **CONTINUE AN OLD SIMULATION:** if **TRUE** then SIMMSUS continues and old simulation by reading the last data printed on position and velocity files and uses them as a new initial condition in order to continue and old simulation. This is useful when the user runs heavy simulations in a place where the electric power supply is unstable. In order to continue an old simulation the user must enable the record data options that will be explained later. If this variable is set to be **FALSE** then SIMMSUS start new simulations from zero;

- **PARTICLE INERTIA:** if **TRUE** then SIMMSUS considers the effect of particle inertia as described in subsection 3.4.4. If this variable is set to be **FALSE** then SIMMSUS neglects the effect of particle inertia;

Figure (16) shows the dipolar interaction calculation menu in the configuration file.

```
- DIPOLAR INTERACTION CALCULATIONS

PERIODIC MAGNETIC TORQUES.............: FALSE
PERIODIC MAGNETIC FORCES..............: FALSE
```

Figure 16: Dipolar interaction calculations menu on simconfig.dat

Bellow we detail what each variable means.

- **PERIODIC MAGNETIC TORQUES:** If **TRUE** then SIMMSUS computes long-range magnetic torques between the particles periodically according to subsection 3.4.1, if **FALSE** these torques are computed in a non-periodic way;

- **PERIODIC MAGNETIC FORCES:** If **TRUE** then SIMMSUS computes long-range magnetic forces between the particles periodically according to subsection 3.4.1, if **FALSE** these forces are computed in a non-periodic way;

Figure (17) shows the record data option menu in the configuration file. Bellow we detail what each variable means.

- **RECORD POSITION IN FILE:** If **TRUE** then SIMMSUS generate large data files named **posicao___.plt** or **posicao___.xyz** depending on the output format defined by the user containing the position of each individual particle in each realization if **FALSE** these files are not created. This option must be enabled if the user wants to visualize animations resulting from the simulations;

```
- RECORD DATA OPTIONS

RECORD POSITION IN FILE.................: FALSE
RECORD VELOCITY IN FILE.................: TRUE
RECORD DIPOLE IN FILE...................: FALSE
OVITO(T) OR TECPLOT(F) OUTPUT FORMAT...: TRUE
```

Figure 17: Record data options menu on simconfig.dat

- **RECORD VELOCITY IN FILE:** If **TRUE** then SIMMSUS generate large data files named **velocidade___.plt** containing the velocities of each individual particle in each realization if **FALSE** these files are not created. This option must be enabled if the user wants to perform statistical analysis from the simulation data;
- **RECORD DIPOLE IN FILE:** If **TRUE** then SIMMSUS prints the orientation of each individual dipole moment of each particle with their positions in the **posicao___.plt** or **posicao___.xyz** files if **FALSE** these columns are not printed. This option must be enabled if the user wants to visualize the evolution of the particles dipole moments in the animations;
- **OVITO(T) OR TECPLOT(F) OUTPUT FORMAT:** If **TRUE** then SIMMSUS prints the position and dipole orientation of particles in an appropriate format to be read by OVITO with the extension **posicao___.xyz** if **FALSE** these files are printed with the **posicao___.plt** extension and minor synthax modifications are implemented for the files to be post-processed by Tecplot.

Figure (18) shows the simulation quantities menu in the configuration file.

```
- SIMULATION QUANTITIES

NUMBER OF PARTICLES.....................: 00500
NUMBER OF REALIZATIONS..................: 00100
```

Figure 18: Simulation quantities menu on simconfig.dat

Bellow we detail what each variable means.

- **NUMBER OF PARTICLES:** Represents the number of particles in each numerical experimental. Must be an integer with the same format shown in figure (18);

- **NUMBER OF REALIZATIONS:** Represents the number of simultaneous numerical experiments. Must be an integer with the same format shown in figure (18);

Figure (19) shows the initial configuration menu in the configuration file.

```
- INITIAL CONFIGURATION INFORMATION

ORDERED (T) OR RANDOM (F) ARRANGEMENT..: FALSE
MIX MAGNETIC AND NON MAGNETIC PARTICLES: FALSE
INITIAL SPHERICAL AGGREGATE............: FALSE
ORDERED DIPOLES........................: FALSE
```

Figure 19: Initial configuration menu on simconfig.dat

Bellow we detail what each variable means.

- **ORDERED (T) OR RANDOM (F) ARRANGEMENT:** If **TRUE** SIMMSUS creates an initial ordered configuration. If the user sets this variable to be true, than the number o particles must have a real cubic root. If this variable is set to be **FALSE** then SIMMSUS creates a different random initial configuration for each numerical experiment;
- **MIX MAGNETIC AND NON MAGNETIC PARTICLES:** Here the user can choose to mix magnetic with non-magnetic particles;
- **INITIAL SPHERICAL AGGREGATE:** If this option is set to be **TRUE** then SIMMSUS puts all the particles inside a boundary sphere in the form of an initial spherical aggregate whose internal volume fraction is set on the particle distribution data menu on simconfig.dat;
- **ORDERED DIPOLES:** If **TRUE** then all dipoles are set to be aligned in the same direction in the initial condition;

Figure (20) shows the particle distribution data menu in the configuration file.

Bellow we detail what each variable means.

- **MONO (F) OR POLIDISPERSITY (T)** If **FALSE** SIMMSUS considers different particle diameters, if **TRUE** then all particles have the same diameters;
- **VOLUME FRACTION OF PARTICLES:** Is a real format variable that specifies the volume fraction of particles. Hence, a 5% concentration should be expressed as 5.0000E-02;

```
- PARTICLE DISTRIBUTION DATA

MONO (F) OR POLIDISPERSITY (T).........: FALSE
VOLUME FRACTION OF PARTICLES...........: 5.0000E-02
BOX ASPECT RATIO.......................: 00001
PERCENTAGE OF NON-MAGNETIC PARTICLES...: 5.0000E-01
```

Figure 20: Particle distribution data menu on simconfig.dat

- **BOX ASPECT RATIO:** Is an integer variable that specifies the aspect ratio of the simulation box. Therefore 00001 denotes a cubic box;
- **PERCENTAGE OF NON-MAGNETIC PARTICLES:** If **MIX MAGNETIC AND NON MAGNETIC PARTICLES** is set to be **TRUE** in the initial configuration information menu, here the user must specify using real format the percentage of non-magnetic particles used in the simulations;

Figure (21) shows the magnetic field information menu in the configuration file.

```
- MAGNETIC FIELD INFORMATION

APPLY AN EXTERNAL MAGNETIC FIELD.......: TRUE
POSITION OF THE EXTERNAL FIELD.........: 00002
OSCILLATORY FIELD......................: TRUE
ROTATING FIELD.........................: FALSE
NON-LINEAR DUFFING FIELD EXCITATION....: FALSE
DOUBLE FREQUENCY FIELD EXCITATION......: FALSE
DYNAMICAL INCREASE OF FIELD FREQUENCY..: TRUE
FREQUENCY 1 OF THE MAGNETIC FIELD......: 1.0000E+00
FREQUENCY 2 OF THE MAGNETIC FIELD......: 1.1000E+01
C1 PARAMETER FOR DUFFING EXCITATION....: 1.0000E+00
C2 PARAMETER FOR DUFFING EXCITATION....: 1.0000E+00
C3 PARAMETER FOR DUFFING EXCITATION....: 1.0000E+00
C4 PARAMETER FOR DUFFING EXCITATION....: 5.0000E+01
MAX FREQUENCY FOR DYNAMICAL INCREASE...: 1.0000E+01
NUMBER OF INTERVALS FOR DYN. INCREASE..: 00010
```

Figure 21: Magnetic field information menu on simconfig.dat

Bellow we detail what each variable means.

- **APPLY AN EXTERNAL MAGNETIC FIELD** If **TRUE** SIMMSUS considers the application of an external magnetic field in the direction specified in the **POSITION OF THE EXTERNAL FIELD** variable in the same menu;

- **POSITION OF THE EXTERNAL FIELD:** Is an integer format variable that defines the position of the external magnetic field according to the following scheme:

  1. Field applied on the lower wall;
  2. Field applied on the upper wall;
  3. Field applied on the right wall;
  4. Field applied on the left wall;

  Front and back wall fields are not implemented in the current version of SIMMSUS;

- **OSCILLATORY FIELD:** If **TRUE** then SIMMSUS considers the application of an oscillatory field;

- **ROTATING FIELD:** If **TRUE** then SIMMSUS considers the application of a rotating magnetic field;

- **NON-LINEAR DUFFING FIELD EXCITATION:** If **TRUE** then SIMMSUS considers the application of a magnetic field based on the solution of the nonlinear harmonic Duffing oscillator based on a nonlinear ordinary differential equation whose coefficients are defined in this menu;

- **DOUBLE FREQUENCY FIELD EXCITATION:** If **TRUE** then SIMMSUS considers the application of an oscillatory magnetic field with two independent frequencies;

- **DYNAMICAL INCREASE OF FIELD FREQUENCY:** If **TRUE** then SIMMSUS considers that the frequency of the oscillatory field will be increased during the simulation in regular intervals from a minimum to a maximum value defined by the user in this menu;

- **FREQUENCY 1 OF THE MAGNETIC FIELD:** Is a real format variable that specifies the first (or only) frequency of the field. For oscillatory, rotating or nonlinear Duffing field excitation this is the frequency of the field;

- **FREQUENCY 2 OF THE MAGNETIC FIELD:** Is a real format variable that specifies the second frequency of the field for a double frequency field excitation;

- **C1,C2,C3 and C4 PARAMETERS FOR DUFFING EXCITATION:** Are real format variables used to assemble the nonlinear ordinary differential equation (12) related to the field excitation. More information about this excitation can be found on subsection 3.4.6 of this manual;

- **MAX FREQUENCY FOR DYNAMICAL INCREASE:** Is a real format variable that specifies the maximum frequency of the field considered for the dynamical sweep (if **TRUE**);

- **NUMBER OF INTERVALS FOR DYN. INCREASE:** Is an integer format variable that specifies the number of different frequencies (intervals) considered for the dynamical sweep (if **TRUE**);

Figure (22) shows the shear rate information menu in the configuration file.

```
- SHEAR RATE INFORMATION

 TURN ON SHEAR RATE.....................: FALSE
 OSCILLATORY SHEAR......................: FALSE
 DYNAMICAL INCREASE OF SHEAR RATE.......: FALSE
 DIMENSIONLESS SHEAR RATE...............: 0.0000E+00
 FREQUENCY FOR THE OSCILLATORY SHEAR....: 0.0000E+00
```

Figure 22: Shear rate information menu on simconfig.dat

Bellow we detail what each variable means.

- **TURN ON SHEAR RATE** If **TRUE** SIMMSUS considers the application of a shear rate over the suspension space;

- **OSCILLATORY SHEAR:** If **TRUE** the shear rate is considered to be oscillatory;

- **DYNAMICAL INCREASE OF SHEAR RATE:** If **TRUE** then SIMMSUS considers that the frequency of the oscillatory shear will be increased during the simulation in regular intervals from a minimum to a maximum value. Here the maximum value for the frequency of the oscillatory shear is the one defined in the previous menu (magnetic field information): MAX FREQUENCY FOR DYNAMICAL INCREASE;

- **DIMENSIONLESS SHEAR RATE:** Is a real format variable that specifies the intensity of the nondimensional applied shear rate;

- **FREQUENCY FOR THE OSCILLATORY SHEAR:** Is a real format variable that specifies the oscillatory shear, which in the case of a dynamical increase of the shear is equivalent to the minimum frequency adopted on the oscillatory shear ramp;

Figure (23) shows the physical parameters menu in the configuration file.

```
- PHYSICAL PARAMETERS

LAMBDA ..............................: 0.0000E+00
ALPHA ...............................: 1.0000E+01
BROWNIAN PECLET NUMBER...............: 1.0000E+00
TRANSLATIONAL STOKES NUMBER..........: 1.0000E-01
```

Figure 23: Physical parameters menu on simconfig.dat

Bellow we detail what each variable means.

- **LAMBDA:** Is a real format variable that specifies the ratio between the interacting energy of the dipole moments of the particles and the energy of thermal fluctuations arising from Brownian motion. This parameter can be defined as:

$$\lambda = \frac{\mu_0 m_d^2}{24\pi k_B T a^3}, \tag{28}$$

  where μ_0 is the magnetic permeability of free space, m_d is the intensity of the particles dipole moments, k_B is the Boltzmann constant, T is the absolute temperature of the fluid and a is the particle radius;

- **ALPHA:** Is a real format variable that specifies the ratio between the energy of the applied field and the energy of thermal fluctuations arising from Brownian motion. This parameter can be defined as:

$$\alpha = \frac{\mu_0 m_d H_0}{k_B T}, \tag{29}$$

  where H_0 is the magnitude of the applied magnetic field;

- **BROWNIAN PECLET NUMBER:** Is a real format variable that specifies the ratio between the Brownian timescale $a^2/\mathcal{D}$ and a sedimentation timescale a/U_s. Here $\mathcal{D}$ denotes the Stokes-Einstein diffusion coefficient and U_s is the Stokes velocity of an isolated particle. Hence, for $Pe < 1$ the particle is not capable to sediment due to the action of the Brownian motion. Small values of Peclet imply in a highly Brownian regime. This quantity only makes sense when variable **SEDIMENTATION** is set to be **TRUE**;

- **TRANSLATIONAL STOKES NUMBER:** Is a real format variable that specifies the relaxation time of the particles. This quantity is related to the mass of the particles. For more information regarding the physical meaning of the Stokes number please consult subsection 3.4.4;

Figure (24) shows the numerical data menu in the configuration file.

```
- NUMERICAL DATA

SIMULATION TIME.....................: 1.0000E+02
NUMERICAL TIME-STEP.................: 1.0000E-02
STEP FOR STORING THE RESULTS........: 00010
CONTINUE FROM ITERACTION NUMBER.....: 10000
```

Figure 24: Numerical data menu on simconfig.dat

Bellow we detail what each variable means.

- **SIMULATION TIME:** Is a real format variable that specifies the total simulation time;
- **NUMERICAL TIME-STEP:** Is a real format variable that specifies the simulation time-step;
- **STEP FOR STORING THE RESULTS:** Is an integer format variable that specifies the number of time-steps considered between two consecutive data register. This option is useful to avoid huge output files that can fill up your storage disk;
- **CONTINUE FROM ITERACTION NUMBER:** Is an integer format variable that specifies the last iteraction number when the user wants to contine and old simulation;

Figure (25) shows the optional data treatment menu in the configuration file.

```
- OPTIONAL DATA TREATMENT

STATISTICAL ANALYSIS................: TRUE
CALCULATE THE STRUCTURE FACTOR.....: FALSE
PRINT LOCAL MAPS OF PHI.............: FALSE
```

Figure 25: Optional data treatment menu on simconfig.dat

Bellow we detail what each variable means.

- **STATISTICAL ANALYSIS:** Is a logical variable that if set to be **TRUE** will calculate the statistical quantities of the simulations described in subsection 3.5
- **CALCULATE THE STRUCTURE FACTOR:** Is a logical variable responsible for calculating the structure factor of the suspension;
- **PRINT LOCAL MAPS OF PHI:** If true, SIMMSUS divides the suspension space in 1000 smaller cells and compute the local volume fraction of particles inside each cell. This generates a text file with the 3D local volume fraction of particles within the suspension space;

4 How to run a simulation?

First, download the files in simmsus repository (https://github.com/rafaelgabler/simmsus), put then in a folder and from a terminal run the make command twice to compile. The first run will create the modules and the second one will create the executable file simmsus.ex.

In order to run a simulation the user needs to put the following files in a new simulation folder

1. simconfig.dat
2. simmsus.ex

and run ./simmsus.ex. The initial terminal screen is displayed in figure (26)

Figure 26: Simmsus execution screen

The simulation progress will appear in the terminal (figure 27). After calculating all particles positions, velocities and orientations in all parallel

realizations, SIMMSUS calculates automatically some statistics of the particulate system (if this option is enable on the configuration file) and prints additional terminal messages to finish the simulations (figures 28 and 29).

```
SIMULATION PROGRESS:  0.1000100     %
SIMULATION PROGRESS:  0.2000200     %
SIMULATION PROGRESS:  0.3000300     %
SIMULATION PROGRESS:  0.4000400     %
SIMULATION PROGRESS:  0.5000500     %
SIMULATION PROGRESS:  0.6000600     %
SIMULATION PROGRESS:  0.7000700     %
SIMULATION PROGRESS:  0.8000801     %
SIMULATION PROGRESS:  0.9000900     %
SIMULATION PROGRESS:   1.000100     %
SIMULATION PROGRESS:   1.100110     %
SIMULATION PROGRESS:   1.200120     %
SIMULATION PROGRESS:   1.300130     %
SIMULATION PROGRESS:   1.400140     %
SIMULATION PROGRESS:   1.500150     %
SIMULATION PROGRESS:   1.600160     %
SIMULATION PROGRESS:   1.700170     %
SIMULATION PROGRESS:   1.800180     %
SIMULATION PROGRESS:   1.900190     %
SIMULATION PROGRESS:   2.000200     %
```

Figure 27: Simulation progress screen

```
Statistical analysis over the average velocity - OK
Statistical analysis over the variance - OK
Statistical analysis over the self-correlation function- OK
Statistical analysis over the diffusion coefficient - OK
Generation of output files - OK
```

Figure 28: Statistical analysis final terminal screen

```
Deallocating the variables used in the statistics module...

Deallocating matrix 1 - OK
Deallocating matrix 2 - OK
Deallocating matrix 3 - OK
Deallocating matrix 4 - OK
Deallocating matrix 5 - OK
Deallocating matrix 6 - OK
Deallocating matrix 7 - OK
Deallocating matrix 8 - OK
Deallocating matrix 9 - OK
Deallocating matrix 10 - OK
Deallocating matrix 11- OK
Deallocating matrix 12- OK
Deallocating matrix 13- OK
Deallocating matrix 14- OK
Deallocating matrix 15- OK
TOTAL SIMULATION TIME:   8.932895     SECONDS
```

Figure 29: Deallocating variables screen

5 Simulation output files

You will notice that after running a simulation several data files appear in the original folder. Depending on the specific simulated physics and on the way the user configured the simulations in simconfig.dat the following additional files may appear.

- **test_particle.plt**: represents the XYZ positions of an arbitrary test particle within the suspension space. This is useful to plot typical trajectories of a test particle subjected to a specific physics;

- **magnetization.plt**: This file contains the average orientation (magnetization) of all the particles in all simultaneous numerical experiments in each direction (Mx, My, Mz), the field excitation components and its derivatives as a function of time. This is a very useful file to plot the dynamical behavior of the suspension magnetization, perform general time-series analysis on this quantity and plot hysteresis curves of the system;

- **position x.plt** or **posicao x.xyz**: These files contain the position in each direction (X,Y,Z) of each particle (each line) for several time-steps (separated by zones). The letter "x" in the name of the file is an index that represents the number of that particular numerical experiment. These files will be created if the user sets **RECORD POSITION IN FILE** equal TRUE in the configuration file. If the option "RECORD DIPOLE IN FILE" is also enabled the orientations of each magnetic moment will be included in the "posicao x.plt" files. Only these files are necessary to make animations of the motion of the particles using post-processing softwares such as Tecplot and Ovito. The plt files are apprppriate to be used with Tecplot and the xyz files with OVITO;

- **velocity x.plt** If the option "RECORD VELOCITY IN FILE" is set to be true, the solver will create several files named "velocidade x.plt" where "x" is an index that represents the number of that particular numerical experiment. These files are not necessary for particle animation, but are required to perform post-processing statistical analysis;

- **local_phi.plt**: If the option "PRINT LOCAL MAPS OF PHI" is enabled on the configuration file, SIMMSUS will create this file with the local values of the volume fraction of particles inside the simulation box. The algorithm responsible for building this file divided the simulation box in 1000 sub-cells and counts the number of particles in each

sub-cell. This procedure is done twice, after the generation of the initial condition and at the end of the simulation. This file is interesting to plot a 3D color map that shows how the microstructure of the system is altered through the simulation process;

- **structure_factor.plt**: Suspension's structure factor calculated values;

- **average_velocity.plt**: This file is generated if the "STATISTICAL ANALYSIS" option is enabled on the configuration file. It represents the average velocity of the suspension in each direction based on an ensemble average over all the simultaneous numerical experiments with the respective error-bars as a function of time;

- **variance.plt**: This file is generated if "STATISTICAL ANALYSIS" is enabled on the configuration file. It represents the variance of the velocity fluctuations of the suspension based on an ensemble average over all the simultaneous numerical experiments with the respective error-bars as a function of time. This quantity denotes the components of the equivalent Reynolds stresses based on particle velocity fluctuations. The sum of its diagonal denotes the particle pressure of the system;

- **self_correlation.plt**: This file is generated if the "STATISTICAL ANALYSIS" option is enabled on the configuration file and represents the normalized self-correlation function of velocity fluctuations in each direction;

- **correlation_time.plt**: This file is generated if the "STATISTICAL ANALYSIS" option is enabled on the configuration file and represents the correlation time of the system, which is basically the integral of the normalized self-correlaction function with respect to time.

6 How to visualize a simulation?

Here we will teach how to visualize beautiful simulations using an open source tool named OVITO. In order to download and install OVITO please visit https://www.ovito.org/. OVITO is a very simple to use and really powerful tool for visualization and analysis of numerical simulations of systems of particles.

OVITO installation is very simple and the program runs automatically from an executable file in the bin folder of the downloaded files in OVITO's website. So just copy any posicao***.xyz file (from a simulation performed by SIMMSUS) to the same file of OVITO's executable file and open OVITO with ./ovito. The initial screen of OVITO is shown in figure (30).

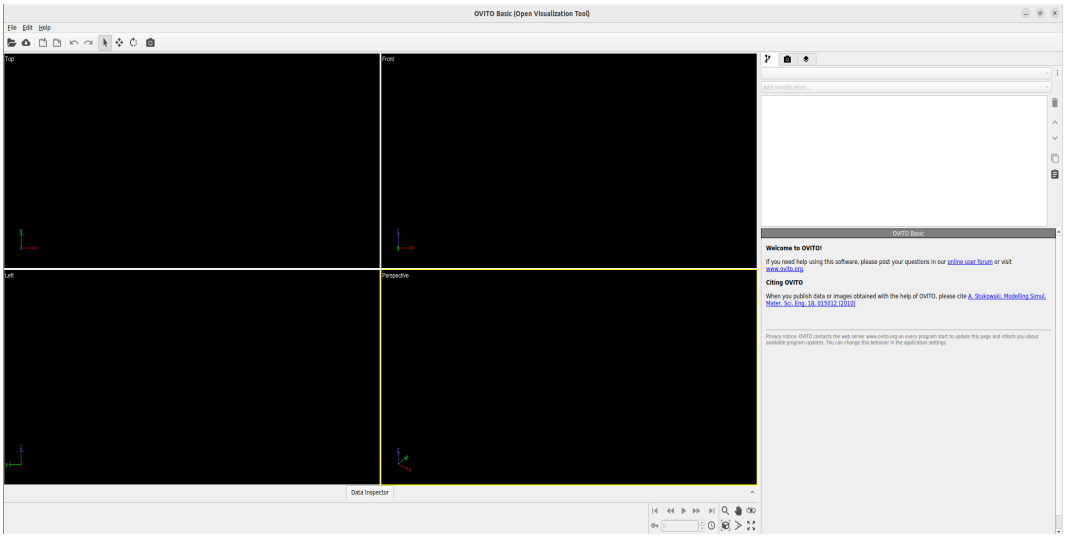


Figure 30: OVITO's initial screen

After opening OVITO you must load the simulation file in File - Load file. After choosing the appropriate xyz file you must define the variables in each column according to figure (31). After clicking in OK you will see the initial condition of the simulation in multiple frames just like figure (32).

In order to see the dipole moments of the particles you must mark the checkbox associated with the dipoles on the right side of the animation screen (figure 33). After enabling the dipole checkbox you should see the initial condition of the system with additional arrows in each particle indicating their dipole orientation. Figure (34) shows the initial configuration of the system with the indication of the particles dipole moments in red arrows. The user may configure the colors and sizes of the particles and arrows by clicking on Particles or Dipoles and altering their information on the same

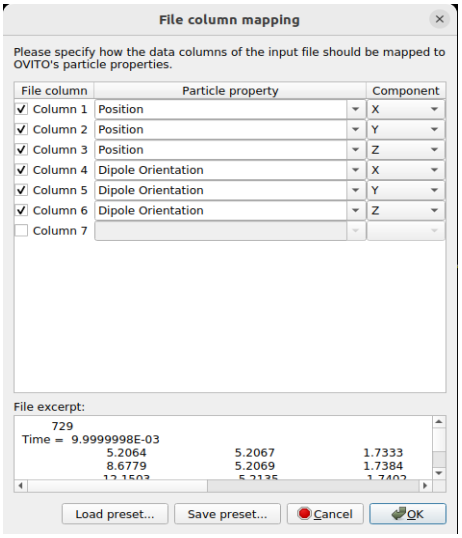


Figure 31: Assigning variables to each column in OVITO

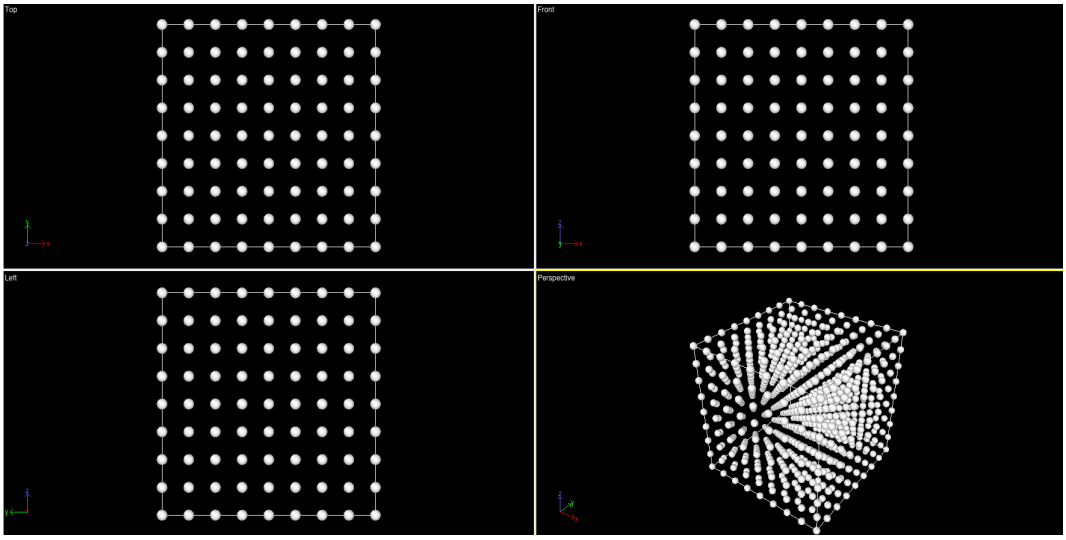


Figure 32: Initial configuration of the simulation loaded by OVITO

screen right under the checkboxes. The user may run the animation on the screen by clicking on the play button at the bottom of the page. You may also export mp4 files of each individual simulation frame by clicking on the camera button at the right side of the control screen and enabling the options shown in figure (35).

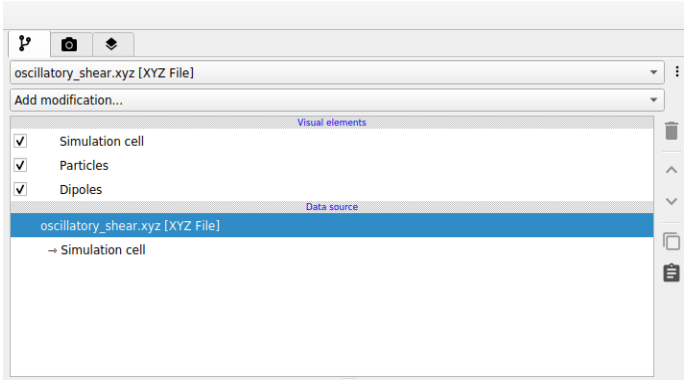


Figure 33: Setting to show the dipole moments of the particles in OVITO

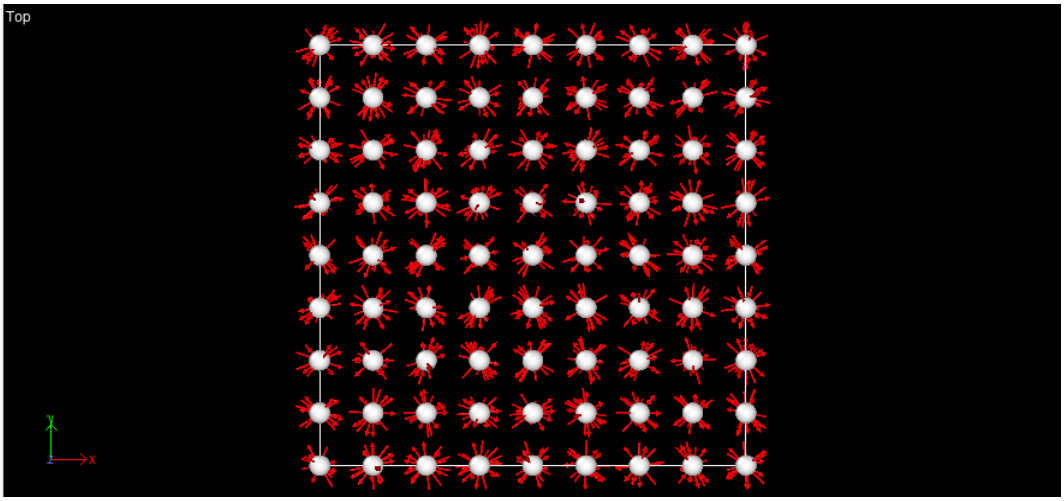


Figure 34: Initial configuration of the simulation loaded by OVITO after enabling Dipoles

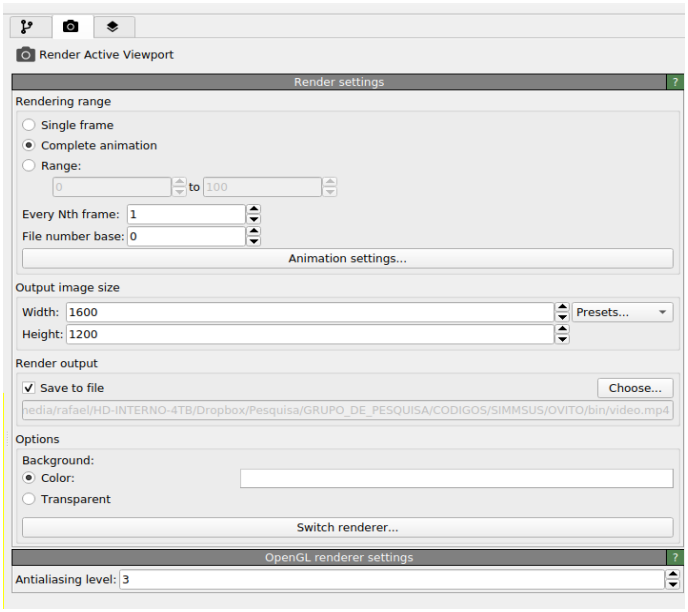


Figure 35: Exporting you mp4 animation through OVITO

7 The equations

The equations solved by SIMMSUS depend on the options defined by the user in the configuration file. Therefore, there are several paths in which the simulation can follow and each path will lead to a different set of mathematical equations responsible for computing the specif physics defined by the user. We won't enter in details here about the equations solved by SIMMSUS, since they have been extensively documented in previous publications. We recommend that users and potential developers interested in knowing more about the mathematical modeling adopted by SIMMSUS should consult the following references.

1. GONTIJO, R.G.; CUNHA, F.R. . Dynamic numerical simulations of magnetically interacting suspensions in creeping flow. Powder Technology (Print), v. 279, p. 146-165, 2015.

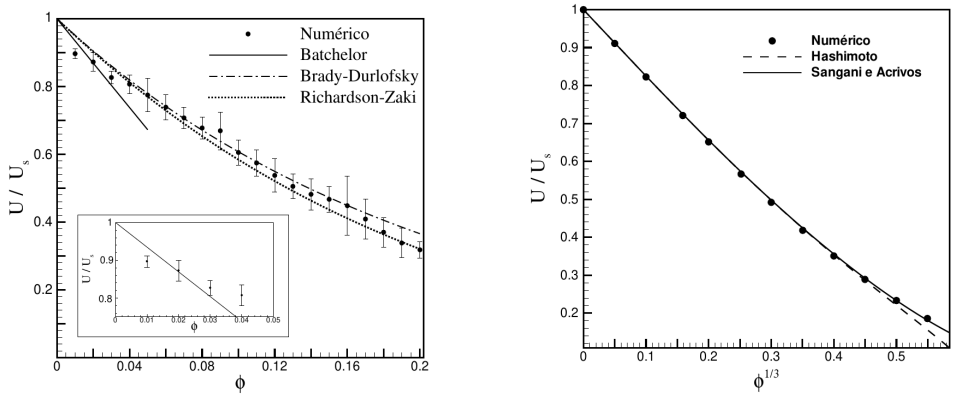
2. Gontijo, R. G.; CUNHA, F. R. . Numerical simulations of magnetic suspensions with hydrodynamic and dipole-dipole magnetic interactions. PHYSICS OF FLUIDS, v. 29, p. 062004, 2017.

3. GUIMARÃES, A. B. ; CUNHA, F. R. ; Gontijo, R. G. . The influence of hydrodynamic effects on the complex susceptibility response of magnetic fluids undergoing oscillatory fields: New insights for magnetic hyperthermia. PHYSICS OF FLUIDS, v. 32, p. 012008-012008-17, 2020.

8 Validation

SIMMSUS has been tested in different physical scenarios and using different database for validation purposes. Here we show some quantitative simulation results documented in previous publications that shows the capacity of the code to capture relevant aspects on the physics of magnetic and non-magnetic suspensions.

Figure (36) shows the average sedimentation velocity of non-Brownian and non-magnetic particles subjected to hydrodynamic interactions. The initial condition of the system, considered for the plot shown in figure (36b) is shown in figure (37).



(a) Random particle distribution

(b) Ordered particle distribution

Figure 36: Average sedimentation velocity of a system of non-magnetic, non-Brownian particles, subjected to hydrodynamic interactions. Figure (a) shows the behavior of the system when a random initial condition is considered and figure (b) displays the results obtained for an ordered array of particles. In both plots the numerical results are represented by the symbols, while continous, dashed and dash-dotted lines represents different asymptotic and consolidated empirical correlations in the context of sedimentation.

In the context of Brownian, magnetic suspensions, we present figure (38). In these plots we show the equilbrium magnetization of a suspension subjected to a steady state magnetic field (38a) and the temperature derivative of a suspension of magnetic Brownian particles subjected to an oscillatory field with respect to the frequency of the field. In both plots the symbols denote numerical results obtained by SIMMSUS and the lines represent consolidated asymptotic theories. For more details regarding these plots we recommend the reader to consult the references displayed in this manual.

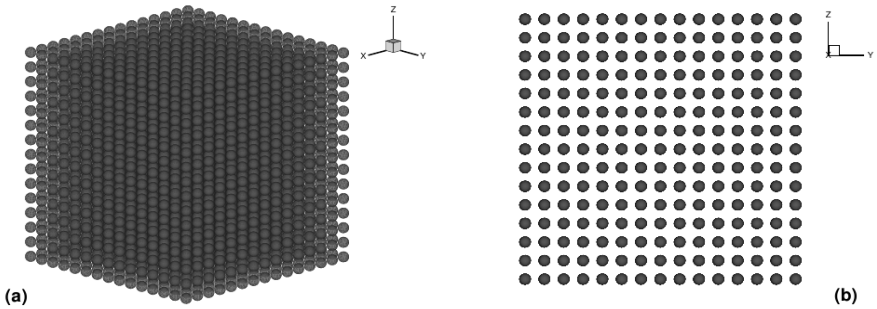
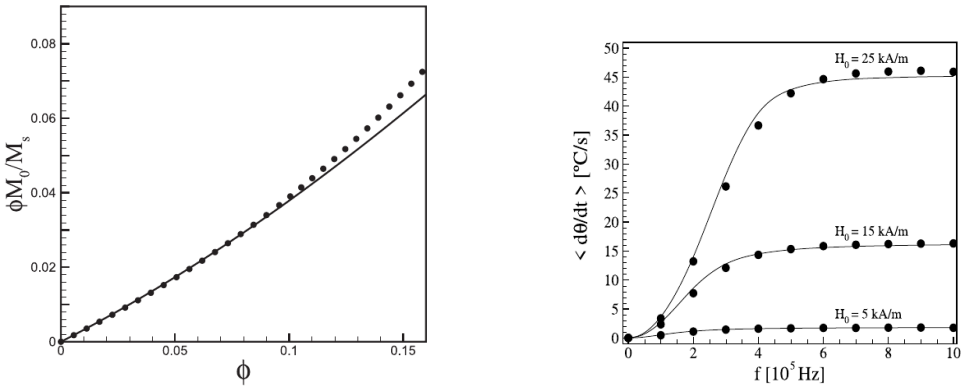


Figure 37: Ordered initial condition considered for the results shown in figure (36b).



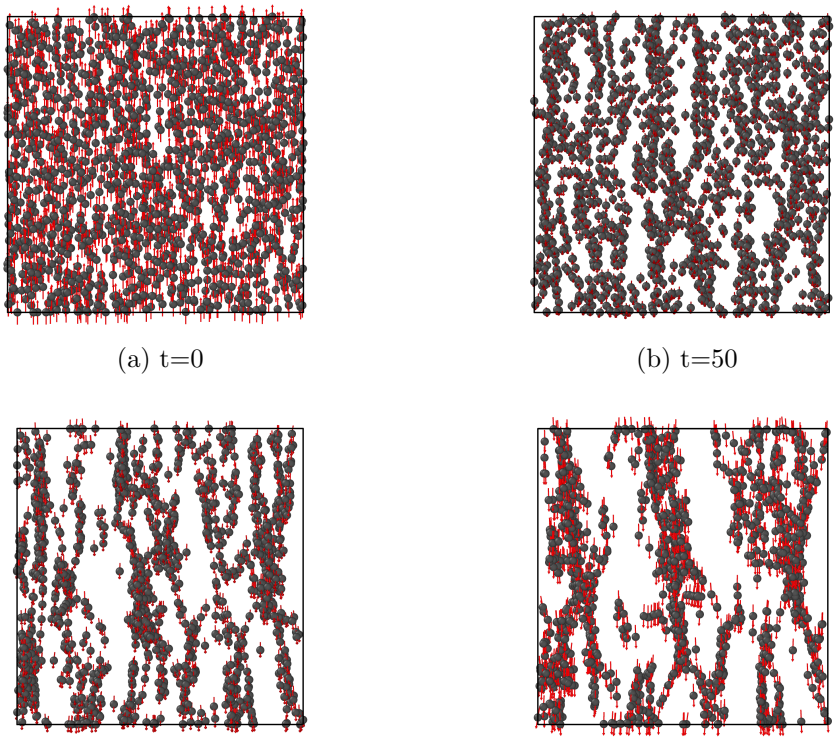
(a) Steady state magnetic field

(b) Oscillatory magnetic field

Figure 38: Equilbrium magnetization of a suspension subjected to a steady state magnetic field with respect to the volume fraction of particles (a). Time derivative of a suspension of magnetic particles subjected to an oscillatory magnetic field in the context of magnetic hyperthermia with respect to the frequency of the applied field.

9 Gallery

Here we show some pictures of numerical simulations performed by SIMMSUS. Figure (39) illustrates the formation of long chains of magnetic particles due to the action of a magnetic rotating field. This plot was obtained using OVITO.



(a) t=0 (b) t=50

(c) t=150 (d) t=200

Figure 39: Chain formation due to the action of a rotating magnetic field applied over a suspension of non-Brownian magnetic particles

Figure (40) shows a local volume fraction map inside the suspension space considering a slice on the 3D simulation box at its center in the z plane. This plot illustrates the idea of the PRINT LOCAL MAPS OF PHI information in the configuration file simconfig.dat. It is possible to clearly see the anisotropic chain formation behavior due to the action of an applied rotating magnetic field. Figure (41) shows an initial random distribution generated by SIMMSUS.

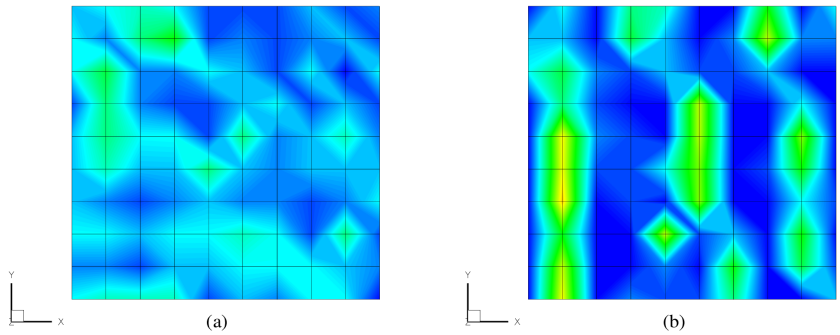
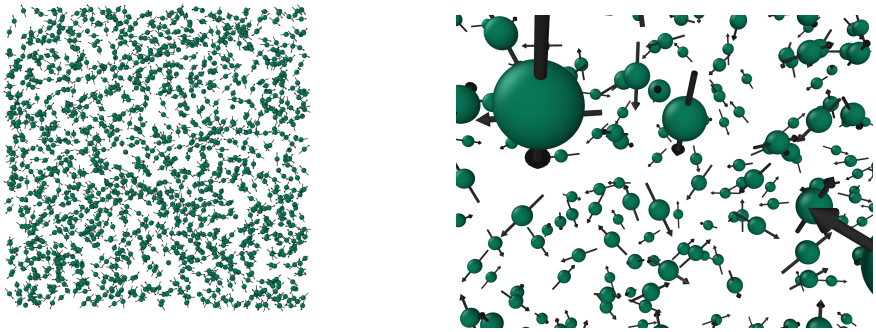


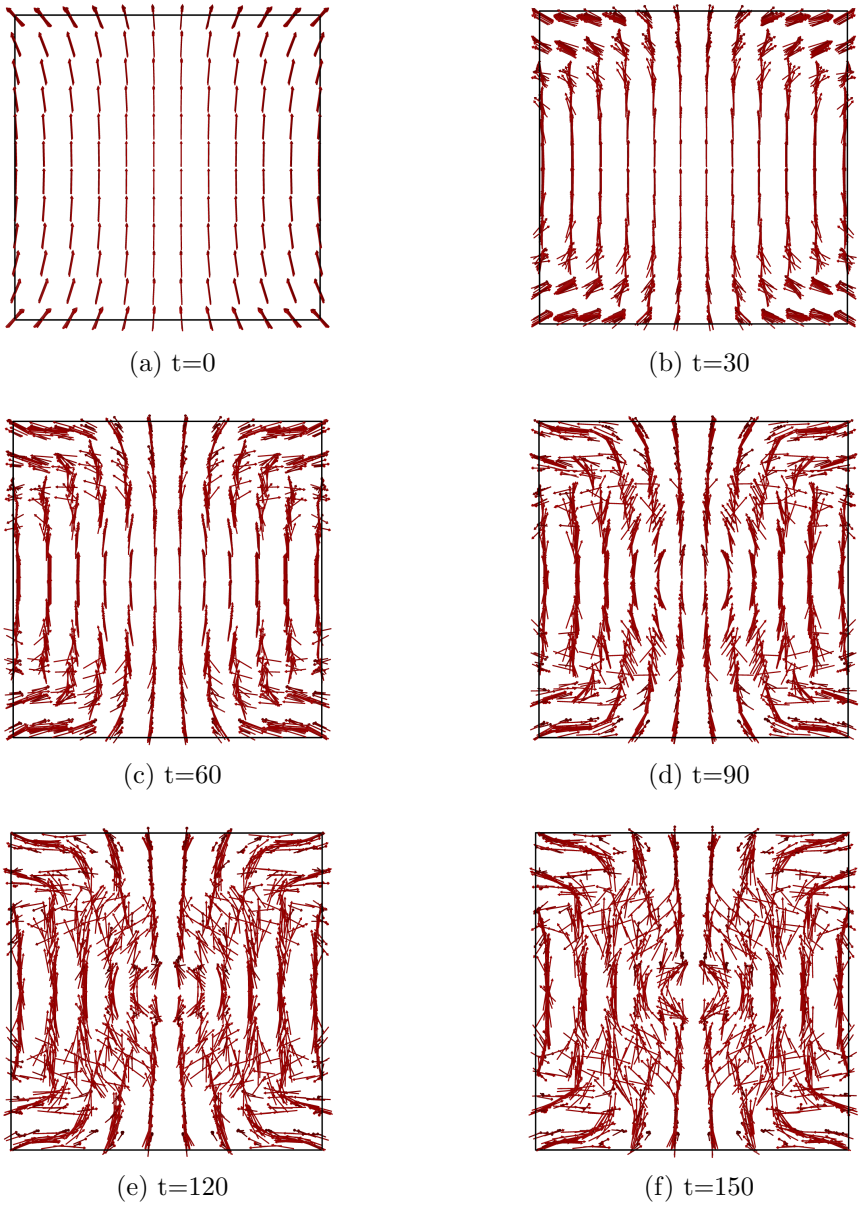
Figure 40: Local concentration maps showing the initial (a) and final (b) distribution of particles in the middle of the 3D box for the time-evolution sequence shown in figure (39).



(a) 2D view

(b) 3D view inside the suspension space

Figure 41: Initial random particle distribution



(a) t=0 (b) t=30

(c) t=60 (d) t=90

(e) t=120 (f) t=150

Figure 42: Time evolution of magnetic dipoles of initially ordered neutrally buoyant particles in suspension.

10 Things to improve

- Implementation of statistical distributions for particle size distribution: in the current version of SIMMSUS, whenever the POLIDISPERSITY option is enable the diameters of the particles follow a uniform distribution in a given interval. It would be interesting to let the user choose the statistical distribution of these quantities considering for example a normal and log-normal distributions, which seems to be more realistic in the context of polydispersions;

- Making a dimensional version of SIMMSUS: this would be a very important implementation that could make SIMMSUS more suitable for industrial and technological applications. In the present version SIMMSUS deals with non-dimensional equations based on physical parameters that are familiar to scientists and researchers. A dimensional version of SIMMSUS could make more people interested in using the code to simulate real life problems easier;

- Building a graphical user interface: it would be interesting to replace the configuration file by widget menus in which the user could fill a form defining the simulation options in a graphical way. A very good option for this implementation would be to use DISLIN, a high level plotting library that displays data in the form of curves, polar plots, 3D-color plots, surfaces, contours, maps and is capable of building widget programs using FORTRAN;

- Automatic construction of Gnuplot scripts: an interesting and rather simple implementation for future versions would be to include in the configuration file an option to build customized Gnoplot scripts based on the output files of a specific simulation set. Depending on the options selected by the user in the configuration file differente output files are created. This implementation would create additional text files with the *.gnu extension that would be Gnuplot scripts designed to plot the relevant curves for that specific simulation. These plots could be also automatically exported to EPS format to be included in Latex documents;

- Improve the implementation of the structure factor calculation and check if the current subroutine fator_estrutura is properly computing this property. We have not done enough checks for this implementation;

11 References

1. GONTIJO, R.G.; CUNHA, F.R. . Dynamic numerical simulations of magnetically interacting suspensions in creeping flow. Powder Technology (Print), v. 279, p. 146-165, 2015.

2. GONTIJO, R.G.; Malvar, S. ; CUNHA, F.R. . Magnetic particulate suspensions from the perspective of a dynamical system. Powder Technology (Print), v. 297, p. 165-182, 2016.

3. GONTIJO, R.G.. A numerical perspective on the relation between particle rotational inertia and the equilibrium magnetization of a ferrofluid. Journal of Magnetism and Magnetic Materials, v. 434, p. 91-99, 2017.

4. Gontijo, R. G.; CUNHA, F. R. . Numerical simulations of magnetic suspensions with hydrodynamic and dipole-dipole magnetic interactions. PHYSICS OF FLUIDS, v. 29, p. 062004, 2017.

5. GONTIJO, R.G.; Malvar, S. . Microstructural transition in an ordered set of magnetic spheres immersed in a carrier liquid. Mechanics Research Communications, v. 83, p. 12-17, 2017.

6. GONTIJO, R.G.. Heat transfer increase for a laminar pipe flow of a magnetic fluid subjected to constant heat flux: an initial theoretical approach. MECHANICS RESEARCH COMMUNICATIONS, v. 91, p. 27-32, 2018.

7. DE CARVALHO, D D ; GONTIJO, R G . Reconstructing a continuous magnetization field based on local vorticity cells, CFD and Langevin dynamics: a new numerical scheme. JOURNAL OF MAGNETISM AND MAGNETIC MATERIALS, v. 514, p. 167135, 2020.

8. GUIMARÃES, A. B. ; CUNHA, F. R. ; Gontijo, R. G. . The influence of hydrodynamic effects on the complex susceptibility response of magnetic fluids undergoing oscillatory fields: New insights for magnetic hyperthermia. PHYSICS OF FLUIDS, v. 32, p. 012008-012008-17, 2020.