An Efficient Parallel Strategy for the Simulation of Particle Sedimentation

S. Fourmanoit[a], F. Bertrand[b] and R. Roy[a]
{Sylvain.Fourmanoit,Francois.Bertrand,Robert.Roy}@PolyMtl.CA

[a]Dept. of Computer Engineering, École Polytechnique de Montréal – P.O Box 6079, Stn. CV, Montréal, QC, Canada H3C 3A7

[b]Dept. of Chemical Engineering, École Polytechnique de Montréal – P.O Box 6079, Stn. CV, Montréal, QC, Canada H3C 3A7

Many packages have emerged over the years to provide efficient implementations of a large spectrum of numerical methods, from linear algebra to complex finite element solvers. In this paper, we will show how carefully selected open source packages can be used as toolboxes to prototype in just a few days efficient parallel simulation strategies for the sedimentation of a large number of particles.

De nombreux outils logiciels ont déjà vu le jour pour permettre l'implémentation efficace d'un large éventail de méthodes numériques, depuis l'algèbre linéaire jusqu'aux solveurs complexes à éléments finis. Dans cet article, nous montrerons comment la selection judicieuse d'outils logiciels à code source libre permet, en quelques jours seulement, la production de simulations parallèles efficaces pour le calcul de la sédimentation d'un grand nombre de particules.

Introduction

Multiphase flows are very common in fluid mechanics applications. However, due to their complex nature, large scale simulation of such flows is still an area of active research in CFD. As a result, many commercial and non-commercial pieces of software of high quality have been produced, so much that it is generally easy to find a fair to very good open source implementation of any method older than two years referenced in the literature.

Consequently, the problem a researcher is facing nowadays is usually not to program from scratch a good code that fits specific needs, but to find a way to combine stable and well-tested independent pieces of code in order to exploit their strengths without running into their known limitations and weaknesses. Of course, the main reason for integrating open source pieces of code is economical.

In this paper, we are interested in simulating the behavior of a large number (typically from 10K to 1 million) analytically-shaped (ellipsoidal) solid particles immersed in a viscous, Newtonian fluid at $\mathrm{Re} \leq 20$. The numerical strategy will be based on an immersed boundary technique developed by Fogelson and Peskin [1], the characteristics of which lead to an easy parallelization. After briefly presenting this technique, we will explain how a simulation strategy based upon it can be prototyped very easily using freely available code. Finally, simulation results will be shown to illustrate the accuracy and efficiency of the proposed method.

1 Immersed Boundary Method

The technique that was used to simulate the sedimentation of a large number of solid rigid particles immersed in a continuous phase will now be described.

1.1 Fluid behavior

The Navier-Stokes equations describe the motion of a Newtonian fluid:

$$\rho\frac{\partial \mathbf{v}}{\partial t} + \rho(\mathbf{v}\cdot\nabla)\mathbf{v} = -\nabla p + \eta\nabla^2\mathbf{v} + \mathbf{f}\ , \qquad (1)$$

where ρ and η are respectively the fluid density and viscosity, and where $\mathbf{v}$ and p are the velocity and pressure fields. $\mathbf{f}$ stands for the body force and accounts in this work for the buoyant force and a force inherent to the immersed boundary technique, as will be described later.

We will also suppose that the fluid is incompressible ($\nabla\cdot\mathbf{v} = 0$). It should be noted that incompressibility is not required for the overall simulation strategy, but is only a simplifying hypothesis.

A MAC-based (*staggered marker and cell*) second order finite difference method in space [2] is used to discretize equation (1). The incompressibility constraint is accounted for using a fractional step method:

$$\rho\frac{\mathbf{v^*} - \mathbf{v^n}}{\Delta t} = -\rho(\mathbf{v^n}\cdot\nabla)\mathbf{v^n} + \eta\nabla^2\mathbf{v^n} + \mathbf{f^n}\ , \qquad (2)$$

$$\rho\frac{\mathbf{v^{n+1}} - \mathbf{v^*}}{\Delta t} = -\nabla p^{n+1}\ , \qquad (3)$$

where n represents the time iteration. If we apply the divergence operator on this last equation, we get, since $\nabla\cdot\mathbf{v^{n+1}} = 0$:

$$\nabla^2 p^{n+1} = \frac{\rho}{\Delta t}\nabla\cdot\mathbf{v}^*\ . \qquad (4)$$

This step can then be viewed as a projection of the velocity field onto a divergence-free subspace. Of course, initial and boundary conditions must be imposed for mathematical well-posedness. For the sedimentation problem, two

sets of boundary conditions for the velocity where considered:

- no slip boundary conditions everywhere;
- periodic boundary conditions everywhere.

Moreover, the Neumann conditions for equation (4), which are obtained by projecting (3) onto $\hat{n}$, the normal unit vector, are

$$\frac{\rho}{\Delta t}(v_{\perp}^{n+1} - v_{\perp}^{*}) = -(\hat{n} \cdot \nabla)p^{n+1} \ , \tag{5}$$

and can be shown to vanish. To summarize, the computation of a time iteration consists in the following steps:

1. Given $\mathbf{v^n}$, solve explicit equation (2) for $\mathbf{v}^*$.
2. Given $\mathbf{v}^*$, solve Poisson problem (4) for p^{n+1} with boundary conditions (5). In this work, this problem is solved using a multigrid solver.
3. Given p^{n+1}, solve explicit equation (3) for $\mathbf{v^{n+1}}$.

We want to emphasize here that, in the case of the sedimentation problem, one must solve at each time iteration a Poisson equation on a structured Cartesian 3D mesh, a task that can be done efficiently in parallel. Finally, since (2) and (3) are explicit schemes, the step-size must satisfy a CFL stability criterion.

1.2 Fluid-particle interactions

In this work, the sedimentation of solid particles is modeled through what is referred to as the immersed boundary method. In a nutshell, the advantage of this method lies in the fact that it relies on one single mesh for the entire simulation. The particles and their position in the suspending fluid are rather taken into account by means of a space- and time-dependent force term in equation (1). Another possibility would have been to resort to the fictitious domain method as proposed by Glowinski et al. [3] and used by one of the authors of this work to simulate fluid flow in complex mixing systems [4].

The whole idea is to express the coupling between the solid and the liquid. To realize this, we define n_i nodes (also called control points) that discretize particle i in a frame of reference with the particle center of gravity as the origin. The position of these control points can be expressed as:

$$\mathbf{x_{ij}^r} = \mathbf{x_i(t)} + \mathbf{O_i(t)} \cdot \mathbf{r_{ij}^r} \ , \tag{6}$$

where $\mathbf{x_i(t)}$ is the position of the center of gravity of particle i, $\mathbf{O_i(t)}$ an orientation matrix and $\mathbf{r_{ij}^r}$ the position of control point j in the (local) frame of reference of the particle.

For each control point discretizing a particle, a corresponding marker $\mathbf{x_{ij}^m}$ can be defined to take into account the relative fluid motion:

$$\dot{\mathbf{x}}_{\mathbf{ij}}^{\mathbf{m}} = \mathbf{v}(\mathbf{x_{ij}^m}) \ . \tag{7}$$

It is then possible to calculate the distance $\xi_{\mathbf{ij}} = \mathbf{x_{ij}^m} - \mathbf{x_{ij}^r}$ between control point j of particle i and its associated fluid marker. One way to compel the control point to stick to the fluid marker is to introduce into equation (1) the following Hookean force:

$$\mathbf{f_{ij}} = (-k\,\xi_{\mathbf{ij}} - 2\gamma\,\dot{\xi}_{\mathbf{ij}})\,\delta(\mathbf{x} - \mathbf{x_{ij}^m}) \ , \tag{8}$$

where k is the Hooke's coefficient, γ the damping constant and δ the Dirac distribution. With this explicit technique, k has to be large enough so that $\max \|\xi_{\mathbf{ij}}\| \ll h$ at all times. Similarly, γ has to be negligible compared to other energy dissipation sources in the system. Following (8), $\mathbf{f} = \sum_{i,j} \mathbf{f}_{ij}$ is then added to the body force of equation (1).

Values for k and γ in the immersed boundary method must be determined. It can be showed [1] that suitable values for k must satisfy:

$$k \propto \frac{M}{n_i(\Delta t)^2} \ , \tag{9}$$

where M is related to the mass of the largest particle in the system. As for γ, our tests in 3D showed that setting $\gamma = 0$ does not affect significantly the results.

1.3 Particle behavior

Following Newton's third law of motion, the driving forces $\mathbf{f_{ij}}$ introduced to enforce a quasi-rigid fluid motion also apply on the particles but in the opposite direction:

$$\mathbf{F_{ij}} = k\,\xi_{\mathbf{ij}} + 2\gamma\,\dot{\xi}_{\mathbf{ij}} \ . \tag{10}$$

The corresponding torque, $\tau_{\mathbf{ij}}$, on particle i is given by

$$\tau_{\mathbf{ij}} = (\mathbf{x_{ij}^r} - \mathbf{x_i}) \wedge \mathbf{F_{ij}} \ . \tag{11}$$

Moreover the buoyant force reads as

$$\mathbf{F_i^g} = (\rho - \rho_p)\,V_i\,\mathbf{g} \ , \tag{12}$$

where V_i stands for the volume of particle i.

In this work, it is assumed that particles do not collide, which in the case of the free and even hindered settling of particles can be justified by the relative importance of the lubrification forces between neighboring particles. Of course, in the case of the consolidation of settling particles on a flat surface, for instance, the situation would be different; one would then have to model particle collision using, for example, the discrete element method (DEM). In our proposed strategy, particles collisions are prevented by introducing for two neighboring particles i and k a repulsive force $\mathbf{F_{ik}^p}$ of module inversely proportional to the distance

1. **Prediction** Obtain the positions of the particles and that of their associated markers based on previously computed values.
2. **Force calculation** Compute the $\mathbf{f_{ij}}$ corresponding to the new positions of the particles and markers.
3. **Navier-Stokes solution** Solve equation (1) using a MAC grid and a multigrid solver, as explained in subsection 1.1.
4. **Correction** From the velocity computed in step 3, get new positions of the particles and markers and go back to 2 if needed.

Figure 1. Overall Simulation strategy

between these two particles. Finally, the Brownian effects are neglected, which is a reasonable hypothesis if, for instance, the particles are not too small.

The total force $\mathbf{F_i}$ and total torque τ_i on particle i are given by

$$\mathbf{F_i} = \mathbf{F_i^g} + \sum_k \mathbf{F_{ik}^p} + \sum_j \mathbf{F_{ij}} \, , \tag{13}$$

$$\tau_i = \sum_j \tau_{ij} \, . \tag{14}$$

Finally, a predictor/corrector scheme is used to integrate

$$\left\{ \begin{array}{lcl} \mathbf{\ddot{r}_i} & = & \mathbf{F_i}/M_i \\ \mathbf{\ddot{O}_i} & = & \tau_i/I_i \end{array} \right. \tag{15}$$

and obtain the position and orientation of particle i over time.The simulation strategy is summarized on figure 1.

2 Implementation strategy

From an algorithmic point of view, the immersed boundary technique described above is a good candidate for parallelization. A closer look at this method also shows that only a few statically-sized data structures are required to implement it:

- **Particles data structures**
  For analytically-described particles, the position of the control points will be inferred from their centers of gravity and orientation, so that only the markers have to be stored explicitly.
- **Fluid data structures**
  Two velocity fields are needed, since the computation of $\mathbf{v}^*$ requires $\mathbf{v^n}$, and that of $\mathbf{v^{n+1}}$ requires $\mathbf{v}^*$.

Other quantities such as $\mathbf{f}$, for instance, are computed on the fly.

For a given particle concentration (also called solids content), the size of the particles and fluid data structures varies linearly with the number of grid nodes. Overall, the memory needed to store the data structures related to the particles is much less that that for the fluid data structures. Algorithmically speaking, the prediction, force collocation and correction steps of the overall strategy (Fig.1) are $\mathcal{O}(n)$ (n being the number of particles). Consequently, step 3 is the most consuming both in terms of computational time and storage.

2.1 Parallel multigrid solver

As explained in subsection 1.1, fluid flow is predicted by solving at each time step a Poisson problem. Fortunately, well-designed octree-like domain decompositions can be used in conjunction with multigrid solvers [5]. A 2D quadtree-like multilevel decomposition is given in Fig.2 as an illustration of this partitioning technique.

We recall that a multigrid solver (see [6] for instance) is an iterative solver that combines solutions obtained with other iterative methods (Jacobi in this work) using overlapping grids of different mesh sizes so that long-range perturbations propagate with high speed. Multigrid exhibits a convergence rate that is independent of the number of unknowns in the discretized system, and is optimal in the sense that the number of operations required varies linearly with it.

The domain partitioning scheme in Fig.2 was extended in 3D. The multigrid solver comes from `http://www.mgnet.org/`. Since all the nodes of a mesh also belong to the finer ones, inter-domain communication is required. To avoid sending many small messages between processes (latency would then be the bottleneck), it is more efficient that each process store not only the data for its corresponding sub-grid, but also a small portion of data, the so-called ghost layers, from its neighbors. To fit the specific nature of our domain partitioning scheme, we designed a multilevel ghost layer that is sent asynchronously to the relevant neighbors at each step of the multigrid method. This layer contains not only the connected nodes on the finest grid, but those on coarser grids as well (see Fig.3).

This multilevel technique combined with this domain partitioning scheme yields impressive parallel characteristics:

- **Predictable behavior**
  - **Static memory use** The exact size of the problem is always known before the simulation begins.
  - **Almost perfect load balancing** Processes have similar loads; small differences can only result from a non-uniform particle distribution.

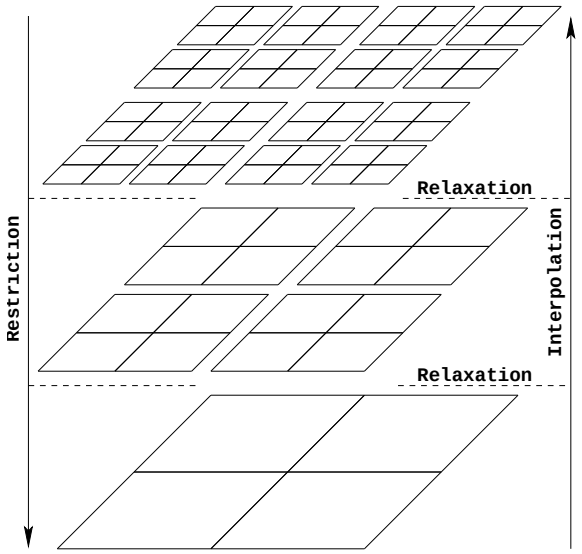

Figure 2. Parallel Multilevel domain decomposition of a square

- **Coarse granularity** Reduced amount of communication. The MAC grid associated to a given process never changes, and data at different grid resolutions overlap or belong to the multilevel ghost layer. As a result, the algorithms can be implemented efficiently on loosely-coupled parallel machines like Beowulf clusters or massively parallel architectures.

- **Good scalability** The overall solution strategy including the immersed boundary technique and the multigrid solver has a time complexity that is a linear function of the problem size. It has been shown [5] that the scalability of the solver layout (i.e. $\lim_{p\to\infty} T(n,1)/T(pn,p)$, where $T(n,p)$ is the time to solve a system of n unknowns on p processors) is $\mathcal{O}(\mathbf{1})$, including communications, provided that some simple constraints are respected regarding domain partitioning. Further experimentations with our prototype will be needed to assert this.

2.2 Other aspects

Two other pieces of code were needed to implement our solution strategy: Gear's predictor/corrector method from Numerical recipes (`http://www.nr.com/`), and a driver written in Objective ML (`http://caml.inria.fr/`) to initiate and pilot the simulations. This driver binds all the pieces of code while storing the particles data, the fluid data being of course kept inside the multigrid solver program (see Fig.4).

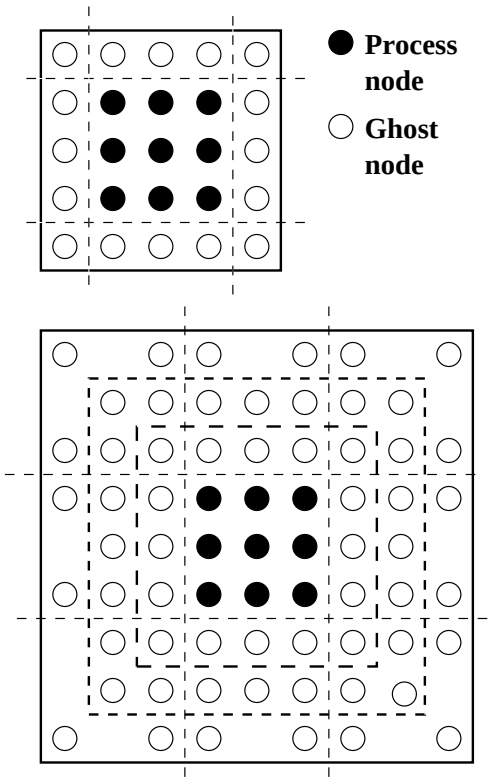

Figure 3. Normal and multilevel ghost layers on a 2D domain

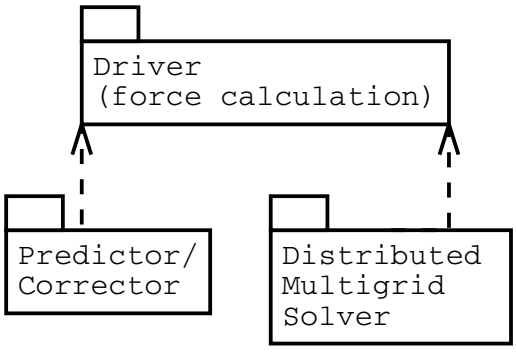

Figure 4. Software packages

3 Numerical results

The terminal velocity of spherical particles settling in a viscous Newtonian fluid with respect to their concentration ϕ is well described by an empirical law called the Richardson and Zaki correlation [7]. It stipulates that the terminal velocity varies as $v_s = v_0(1-\phi)^n$, where v_0 is the terminal velocity of a single particle and $n \in [5,6]$ an index that depends on the Reynolds number. In our simulations, particles radius, a, was chosen such that

$$Re = \frac{2}{9}\frac{a^3}{\eta^2}\rho(\rho_p - \rho)g \approx 0.1\ . \tag{16}$$

The computational domain consists of a cube of size $12a \times 12a \times 12a$, This geometry was discretized using a grid size $h = a/2$. We recall that, with the immersed boundary method, the particles are not meshed as such, but are rather taken into account through a series of control points. In this work, $4\pi a^2/h = 8\pi a$ control points were used and located along the surface of the particles. The mean terminal velocity was computed as:

$$v_s = \frac{1}{N}\sum_{i=1}^{N} \mathbf{v_i} \cdot \hat{z}\ , \tag{17}$$

where $\hat{z}$ is a unit vector pointing in the settling direction. The values of v_s versus ϕ (see Fig.5) agree very well with the correlation of Richardson and Zaki. The slight underestimation of v_s at large concentration values is not really significant. In fact, we observed that it disappears almost completely if the exponent n, a very sensible parameter in the correlation, is slightly modified.

4 Conclusion

The objective of this work was to show how open source packages can be harnessed to prototype and test out a solution strategy for the simulation of the sedimentation of a large number of particles. The whole software process required about two weeks for programming and led to an efficient parallel implementation of the immersed boundary method that proved to be very flexible and accurate.

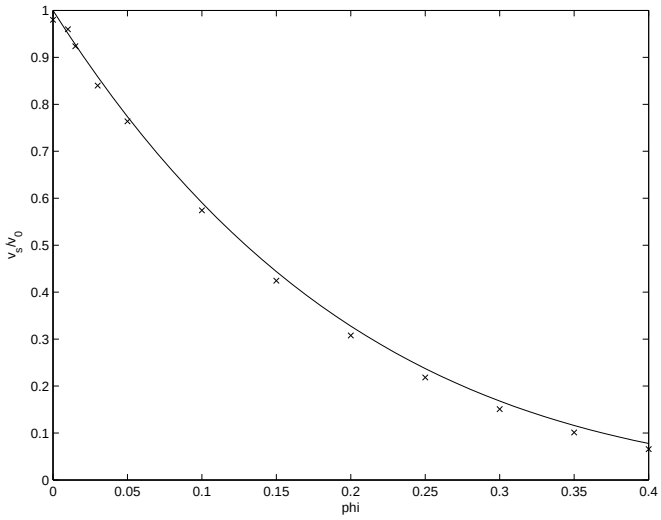

Figure 5. Comparison with the Richardson and Zaki correlation

References

1. A. L. Fogelson and C. S. Peskin. A fast numerical method for solving the three-dimensional Stokes equations in the presence of suspended particles. *Journal of Computational Physics*, pages 50–69, 1988.
2. S. Popinet and S. Zaleski. A front-tracking algorithm for accurate representation of surface tension. *International Journal for Numerical Methods in Fluids*, pages 775–793, 1999.
3. R. Glowinski, T. W. Pan, T. I. Hesla, D. D. Joseph, and J. Periaux. A fictitious domain method with distributed Lagrange multipliers for the numerical simulation of particulate flow. In *Domain Decomposition Methods*, volume 10, pages 121–137, Providence, RI, 1998.
4. P. A. Tanguy F. Bertrand and F. Thibault. A 3D fictitious domain method for incompressible fluid flow problems. *International Journal for Numerical Methods in Fluids*, pages 719–736, 1997.
5. R. D. Falgout and J. E. Jones. Multigrid on massively parallel architectures. In *Multigrid Methods VI*, volume 14, pages 101–107, Berlin, 2000.
6. P. Wesseling. *An Introduction to Multigrid Methods*. Wiley, New York, 1991.
7. E. Guyon, J. P. Hulin, and L. Petit. *Hydrodynamique physique*. CNRS Editions, Paris, 1994.