Hybrid grid–particle methods and Penalization: A Sherman–Morrison–Woodbury approach to compute 3D viscous flows using FFT

Robin Chatelin, Philippe Poncet

Hybrid grid-particle methods and Penalization: a Sherman-Morrison-Woodbury approach to compute 3D viscous flows using FFT

Robin Chatelin[a,b], Philippe Poncet[c]

[a] *Toulouse Mathematics Institute, UMR CNRS 5219, Team MIP, F-31077 Toulouse, France*
[b] *Université de Toulouse, INSA, Dept. GMM, 135 avenue de Rangueil, F-31077 Toulouse, France*
[c] *Laboratoire de Mathmatiques et de leurs Applications, UMR CNRS 5142, IPRA, Université de Pau et des Pays de l'Adour, Avenue de l'Université, BP 1155, F-64013, Pau, France*

Abstract

Particle methods are very convenient to compute transport equations in fluid mechanics as their computational cost is linear and they are not limited by convection stability conditions. To achieve large 3D computations the method must be coupled to efficient algorithms for velocity computations, including a good treatment of non-homogeneities and complex moving geometries. The Penalization method enables to consider moving bodies interaction by adding a term in the conservation of momentum equation. This work introduces a new computational algorithm to solve implicitly in the same step the Penalization term and the Laplace operators, since explicit computations are limited by stability issues, especially at low Reynolds number. This computational algorithm is based on the Sherman-Morrison-Woodbury formula coupled to a GMRES iterative method to reduce the computations to a sequence of Poisson problems: this allows to formulate a penalized Poisson equation as a large perturbation of a standard Poisson, by means of algebraic relations. A direct consequence is the possibility to use fast solvers based on Fast Fourier Transforms for this problem with good efficiency from both the computational and the memory consumption point of views, since these solvers are recursive and they do not perform any matrix assembling. The resulting fluid mechanics computations are very fast and they consume a small amount of memory, compared to a reference solver or a linear system resolution. The present applications focus mainly on a coupling between transport equation and 3D Stokes equations, for studying biological organisms motion in a highly viscous flows with variable viscosity.

Keywords: Three-dimensional computational fluid dynamics, Particle methods, Complex geometry, Penalization, Sherman-Morrison-Woodbury formula, Krylov methods, Non-homogeneous Stokes flow, Biological flows.

1. Introduction

Particle methods have been widely studied in the last decades to compute transport equations in fluid mechanics. In their Lagrangian form the associated convective terms are vanishing with the corresponding CFL stability condition [15, 4, 2], so large time steps can be performed. Moreover Lagrangian methods are linearly scaling, robust and accurate [18, 31]. These features are particularly interesting for large 3D computations. Neverthess, efficient numerical methods are required to compute the related velocity field, in order to keep the Lagrangian treatment of the convection beneficial.

Indeed, on the one hand, when considering vortex methods based on vorticity, Poisson equations are involved in the velocity computation. Such Poisson equations can be solved using FFT-based or finite differences based solvers [14], or kernel methods improved by multipole development techniques [12]. FFT-based solvers provide excellent efficiency even for very large flows [6], particularly for unbounded or periodic domains with no obstacles [7]. Kernel methods (such as Biot-Savard laws for the Poisson equation or Stokeslets for the Stokes equation) are usually less efficient, but they can handle boundaries naturally using integral equations on surfaces [32], and are well fitted for infinite domain problems [30]. Their main drawback is that the expression of the kernel for variable flow parameters (such as viscosity) does not exist, and therefore these methods cannot be applied.

On the other hand, whether a Poisson equation or another elliptic model is involved for the velocity computation, standard discretization methods, such as finite elements or finite volumes methods [17], are leading to large non-standard linear systems [38], whose assembling and resolution become prohibitive for large 3D flows. Furthermore, these problems become especially challenging when the fluid is interacting with immersed moving objects and when the fluid inner composition is not homogeneous.

The Penalization method introduced by Angot et al. [3] is an interesting way to handle the fluid-structure interaction by adding a term in the conservation of momentum equation. Moving obstacles are considered through their characteristic function so a single mesh is used and no remeshing is needed. A splitting of Navier-Stokes equations enables to compute separately the convection (with Lagrangian methods), the diffusion and the Penalization. For high Reynolds numbers diffusion does not dominate so the associated stability condition does not restrict the method: computations can be explicit [23, 13]. When the Reynolds number becomes smaller, this CFL condition imposes smaller time steps and implicit computations are required to keep using Lagrangian methods efficiency (with large time steps). It becomes critical when the Reynolds number is so small that the flow is governed by quasi-static Stokes equations: both the Penalization and the diffusion have to be computed together.

This article presents a novel approach to compute implicitly this Penalization term, which is crucial for the stability and the convergence of algorithms. The present method is based on the Sherman-Morrison-Woodbury (SMW) formula (1), which allows to compute the inverse of a perturbed matrix $A + E^T CE$ with respect to the original inverse matrix A^{-1}, by means of the following relation:

$$\left(A + E^T CE\right)^{-1} = A^{-1} - A^{-1}E^T \left(C^{-1} + EA^{-1}E^T\right)^{-1} EA^{-1} \tag{1}$$

One can refer for instance to [25] for a review of history, various uses and extensions of this formula.

In this equation A and C are square matrices of size n and p respectively while E is a p by n matrix. p is the rank of the perturbation and is assumed to be smaller than n. This is an algebraic relation, which leads to two features: on the one hand no approximation is made to get this formula (and consequently no error is added); on the other hand this allows coefficients of C to be as large as needed, hence the method fits very well for Penalization coefficients which jump from zero to $1/\varepsilon$.

This new method aims at being cheap from a computational point of view so matrix assembling and linear system resolution are avoided. The Penalization term is treated as a small rank perturbation and the SMW formula is exploited coupled to a GMRES approach to get a numerical algorithm which involves only Poisson problems resolution. In this way fast solvers based on Fast Fourier Transforms are used and the computational cost stays reasonable even for large 3D problems. It is shown that the computational cost of the problem computed with this new numerical method is not depending on the complexity of the geometry. This is a good alternative to Kernel methods such as Stokeslets (whose computational time is directly linked to the number and size of obstacles) and linear system assembling (since the condition number grows dramatically with the number of penalized points). This paper shows that this novel method enhanced computational performances: less memory is consumed and computations are faster than using direct multigrid solvers or linear system resolutions.

In the section 2, the Penalization method is presented for the Poisson equation. We present our novel methodology and the computational benefits on this simple problem before introducing the method in the context of fluid mechanics equations. In the section 3, the numerical algorithm to solve the Stokes problem with variable viscosity in a complex geometry is presented coupled to a Lagrangian method for transport equations. The chosen splitting makes appear exactly penalized Poisson problem so the methodology developed in section 2 is straightly applied. In section 4, three numerical fluid mechanics illustrations, including applications to biomedical computing, are presented. Finally in section 5 the computational performances are discussed for these different fluid mechanics simulations.

2. Penalized problems and use of Sherman-Morrison-Woodbury (SMW) formula

2.1. Problem setup for penalized Poisson equation

The method is first presented for a Poisson problem and generalized in the following for fluid mechanics equations. Let u be the solution to a penalized Poisson equation in a domain Ω, with homogeneous Dirichlet boundary conditions

imposed on $\partial\Omega$:

$$\begin{cases} -\Delta u + \dfrac{\chi}{\varepsilon}(u - \overline{u}) = f & \text{in } \Omega \\ u = 0 & \text{on } \partial\Omega \end{cases} \tag{2}$$

whith χ the characteristic function of obstacles and $\varepsilon \ll 1$ the Penalization parameter.

This problem can be rewritten in a more general formulation as

$$\begin{cases} -\Delta u + cu = f & \text{in } \Omega \\ u = 0 & \text{on } \partial\Omega \end{cases} \tag{3}$$

where c is a compact supported function (possibly not continuous with large variations) on the domain $\mathcal{B} \subset \Omega$ (where f is updated from the previous equation).

Let us introduce a discretization $\{x_i\}_{i=1..N}$ of the domain Ω, containing K points inside $\mathcal{B}$. One can then define a discrete mapping ϕ of $1..K$ such that $\{\phi(k)\}_{k=1..K}$ are the index of the points inside $\mathcal{B}$.

This allows to define the restriction matrix E of the points in Ω localized inside $\mathcal{B}$. This is a $K \times N$ matrix:

$$E_{ik} = \delta_{\phi(i),k}$$

where $\delta_{p,q}$ is the Kronecker symbol (equal to 1 if $p = q$ and 0 otherwise). Its transposition E^T is the $N \times K$ inclusion matrix of the points in $\mathcal{B}$ into Ω, because $\{x_{\phi(k)}\}_{k=1..K}$ describes the subset of points x_i of Ω inside $\mathcal{B}$.

The problem (3) can be difficult to solve when the function c exhibits large variations or discontinuities, leading to numerical instabilities and thus requiring artificial smoothing or a large number of iterations when using iterative methods (such as fixed point algorithm). Instead, we introduce the pure Poisson equation:

$$\begin{cases} -\Delta u = f & \text{in } \Omega \\ u = 0 & \text{on } \partial\Omega \end{cases} \tag{4}$$

for which we assume to have a fast solver, such as Fast Fourier Transform (FFT), diagonalizing the Laplace operator. This means that on discretization $\{x_i\}_{i=1..N}$, this Poisson problem can be written as a linear system $AU = B$, which can be solved quickly by means of a fast solver such as FishPack [36], MudPack [37, 1] or FFTW [21]. We can write, for convenience, $U = A^{-1}B$, meaning that a call to one of these fast solver is required.

Now we can consider the initial problem (3): it is the problem (4) perturbed by an extra-term cu (possibly large), on a compact domain $\mathcal{B}$ of Ω, also called the *solid domain*. At a discrete level, this means that using K discretization points inside $\mathcal{B}$ allows to build the diagonal $K \times K$ perturbation matrix C defined by $C_{ii} = c(x_i)$ (and 0 otherwise). Using restriction and inclusion matrices E and E^T, the discretization of problem (3) is given by:

$$\left(A + E^T C E\right) U = F \tag{5}$$

which, by means of Sherman-Morrison-Woodbury formula (1), can be written

$$U = A^{-1}F - A^{-1}E^T \left(C^{-1} + EA^{-1}E^T\right)^{-1} E\, A^{-1}F \tag{6}$$

where every product between A^{-1} and a vector means a call to a fast solver of the standard Poisson equation (4). It only requires the inversion of C, which is a small diagonal matrix (compared to A), as long as $\mathcal{B}$ contains a small number of points (compared to the number of discretization points of Ω).

This formula is algebraic, which means that no approximation is made, thus no error is introduced and the method can be used for large perturbation functions c, including Penalization term $1/\varepsilon$ over an arbitrarily shaped body $\mathcal{B}$.

In the next section a reformulation of the algorithm is presented to use solvers based on Fast Fourier Transforms (FFT). This numerical method is not requiring any matrix assembling or matrix inversion, even for the term $\left(C^{-1} + EA^{-1}E^T\right)^{-1}$, which leads to fast computations with a small memory consumption.

MudPack	SMW-GMRES and FishPack	Penalized Laplacian matrix
$31N + 180N^{2/3} + 360N^{1/3}$	$N + 15N^{1/3} + (2 + N_{Kry})K$	$8N + O(N^2)$

Table 1: Memory needed to compute the solution of (5) with MudPack, SMW-GMRES algorithm with FishPack and a standard matrix assembling using Gauss reduction method. N is the number of grid points, so the number of double precision number to store (64 bit per number).

2.2. GMRES approach combined with fast FFT-based solvers

Let us write again the algorithm using the same notations as in the previous section with A and $E^T CE$ matrices resulting from the discretization of the Laplace operator and the perturbation cu on the compact domain $\mathcal{B}$ respectively.

The goal is to make appear only linear system resolutions and to replace each linear system involving the matrix A by a FFT-based solver resolution. Indeed, as soon as matrix

$$Q = C^{-1} + EA^{-1}E^T \tag{7}$$

is known (where C is a diagonal matrix in the present context), getting the solution of the perturbed system (6) requires only to solve two Poisson problems: $AW = F$ and $AU = E^T Q^{-1} E\, W$. It remains to subtract these vectors to get the solution to (6).

The linear system $AU = E^T Q^{-1} E\, W$ can be written $AU = E^T Z$ where Z is the solution to $QZ = EW$. The resolution of this problem is not straightforward without the computation of the inverse of A. It is nevertheless possible to solve it using a GMRES method [35] on the problem $QZ = EW$. Indeed the GMRES method is minimizing the residual $|QZ - EW|$ and only the product QZ is computed. Thus it is done implicitly without any matrix assembling or inversion. The product evaluation $Z \to QZ$ is performed by two sub-steps, which are computed at each Krylov iteration. As p is supposed to be small, the orthogonalization of Krylov vectors is not deteriorating the cost of the algorithm.

The global algorithm reads in four steps:

1. Solve the Poisson problem $AW = F$
2. Solve $QZ = EW$ with $Q = C^{-1} + EA^{-1}E^T$. This is performed by means of the GMRES method, that requires the product evaluation $Z \to QZ$ as follows:
   (a) Solve the Poisson problem $A\Theta = E^T Z$
   (b) Set $QZ = C^{-1}Z + E\Theta$
3. Solve a last Poisson problem $AY = E^T Z$
4. Update the solution $U = W - Y$

Steps 2a and 2b are computed at each Krylov iteration so the total number of Poisson problems to solve is $2 + N_{Kry}$ with N_{Kry} the typical size of the Krylov space. From a computational point of view, the other steps of our algorithm are negligible: since C is a diagonal matrix its inverse computation is immediate and the matrix E is not assembled since it can be deduced from χ through the mapping ϕ.

Moreover, the algorithm was presented for homogeneous Dirichlet boundary conditions but it is possible to generalize to periodic and non homogeneous (Neumann or Dirichlet) boundary conditions. For periodic boundary conditions there is to compute each Poisson problem with periodic boundary conditions. For non homogeneous boundary conditions, Poisson problem of step 1 is computed with these non homogeneous boundary conditions but Poisson problems of steps 2a and 3 are computed with homogeneous boundary conditions of the same type. This ensures the compatibility between boundary values of the solution to the step 4 and the boundary conditions of the initial problem. Any suitable combination of these boundary conditions can then be used.

2.3. Memory consumption and efficiency

Let us compare in this section the computational efficiency of this algorithm with the solver MudPack to compute the solution to (5). This solver uses multigrid algorithms to compute the solution to elliptic problems with variable coefficients. Thus it can used directly on this Penalization problem and compared with the SMW-GMRES algorithm using FishPack solver to compute the solution to each Poisson problems.

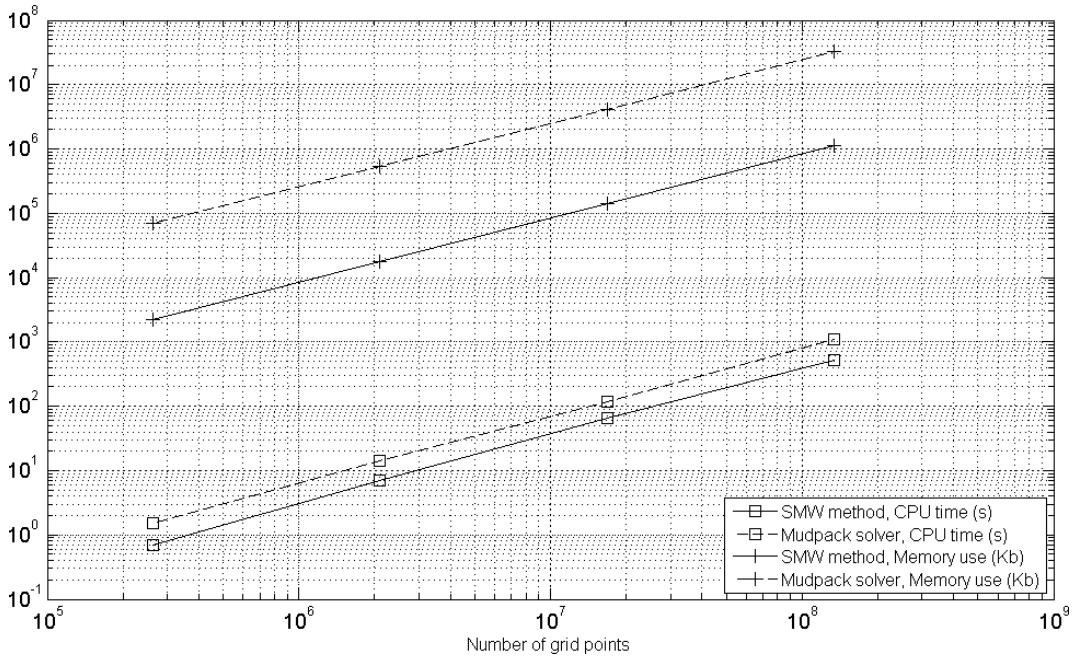

Figure 1: Computational time elapsed and memory required to solve problem (2) with $\overline{u} = 1$ and $f = 0$ for different grid refinements. This figure compares the direct resolution of penalized problem with MudPack solver and the present algorithm based on SMW and GMRES methods (Krylov dimension set to 50), using FishPack for solving the Poisson equations.

To compare the memory consumption, let us use same notations as in the previous section. Computations are performed with a regular 3D cubic grid with N discretization points (which means with $N^{1/3}$ grid points in each direction). For instance the size of variable U is N, and the number of penalized points is K, as well as the size of variables Z or EU. In the following, the variable F (of size N) is not counted since it is the entry data, common to all methods.

To compute the solution to (5) with MudPack we need to store variable U in a different memory location from F, to the opposite of FFT-based solvers. The coefficients of the elliptic equation are computed in a separate routine so they are not stored in the virtual memory. Finally MudPack needs a temporary vector of size $30N + 180N^{2/3} + 360N^{1/3}$ (plus lower powers of N).

With FishPack solver, the solution to the Poisson problem is stored directly in the RHS (since FFT computations are recursive), so we can save one variable of size N. Hence in the step 1 there is only to store the variable W, until the step 3. To compute step 2 we need two arrays of size K (the variable Z and $E\Theta$) and one array of size N which contains successively E^TZ, Θ, E^TZ and Y. Orthogonalization onto the Krylov basis requires to store the N_{Kry} successive values of $QZ - EW$, which means a $N_{Kry} \times K$ array, whose size is at most K^2 when a full GMRES orthogonalization is performed. The final solution is stored in W by means of $U = W - Y$. Finally FishPack also needs a temporary vector, but (it is also a consequence of recursive FFT computations) its size is much smaller than for MudPack: $15N^{1/3}$ (plus lower powers of N).

Finally the assembling of penalized Laplacian matrix with second order finite difference stencils (or Q_2 finite elements on the same regular grid) leads to a sparse $N \times N$ matrix with seven non-zero coefficients per line. The resolution of a linear system usually needs temporary arrays of size $O(N^2)$ (for LU or more elaborated factorization methods). Finally there is also to count the solution U and the RHS F.

The memory consumption of these algorithms is given in the table 1, which compares the resolution of problem (5) for a 3D scalar field (described in next paragraph). The SMW-GMRES algorithm uses 31 times less memory than MudPack solver and 8 times less than than a matrix storage (since we assume $K << N$). Furthermore, the linear system resolution using matrix assembly imposes to use much bigger data sets.

In order to test the computational cost of the algorithm, a simple numerical example is set up. Problem (2) is

solved with $\overline{u} = 1$ and $f = 0$ in a 3D cubic domain with $N^{1/3} = 2^r$ (the refinement in each direction), for different values of r. The computational box is the unity-cube and the solid domain $\mathcal{B}$ is a sphere centered in the middle of this box, its radius is 0.1. The computational time is measured to compare MudPack solver and SMW-GMRES method. The results are presented on figure 1. Results are given for sequential computations on an Intel Xeon X5450. By using our algorithm instead of MudPack solver, we save 50% of computational time.

The computational time was not compared with linear system solving since for 3D simulations, linear system assembling and resolution needs parallel computations. Here FishPack and MudPack solvers were used in a sequential way, so comparison would not have been relevant. Anyway it was already observed that the Penalization terms can deteriorate the condition number, particularly in 3D [26].

This demonstrates that the SMW-GMRES algorithm, combined with FishPack solver, saves both memory and computational times. It is possible to perform large 3D computations in a sequential way. In the next section this method is combined to other powerful computational algorithms to compute fluid mechanics problems.

2.4. A few cases of Penalization method in computational fluid dynamics

The Penalization method [3] was introduced to take into account immersed obstacles in a viscous flow. By considering their characteristic functions there is no need to get an accurate mesh of the interface between the fluid and the obstacles (on the contrary to discretization methods based on conforming grids). Consequently a single mesh is considered, it does not have to move (with Arbitrary Lagrangian Eulerian methods for example) and a periodic regeneration to fit obstacles is not required. This saves long computations for large 3D problems. This method is particularly interesting to combine with a Cartesian grid and corresponding efficient solvers.

The fluid structure interaction is achieved by adding a term in the conservation of momentum equation. It forces a solid velocity $\overline{u}$ inside the obstacle region $\mathcal{B}$ of characteristic function χ. The solid velocity $\overline{u}$ can be prescribed as a data, it is the so called 1-way fluid-structure interaction, when the forces exerted by the fluid on the solid are neglected. On the contrary, the 2-ways coupling do not neglect these forces. This was successfully implemented with a penalization method for rigid bodies by computing these forces and imposing a rigid motion for obstacles [13, 34]. The case of elastic bodies was presented in [23] with self propelled swimmer simulations: the swimming deformation is imposed and the forces are computed to modify the velocity field $\overline{u}$.

In the following the methodology presented in the previous section for the Poisson problem is extended to the 1-way fluid-structure interaction in computational fluid dynamics. The methodology can be generalized to the 2-ways coupling by using the appropriate update of the solid velocity field $\overline{u}$ described in the previous references (and references therein).

2.4.1. Navier-Stokes equations in velocity-pressure formulation

The Penalized Navier-Stokes momentum equation reads:

$$\rho\frac{\partial u}{\partial t} + \rho u\cdot\nabla u - \mathrm{div}(2\mu D(u)) + \frac{\chi}{\varepsilon}(u-\overline{u}) = f - \nabla p \tag{8}$$

where u is the flow velocity, p the pressure, $D(u) = (\nabla u + \nabla u^T)/2$ the strain tensor and μ the viscosity. Eventually ε is the Penalization parameter satisfying $\varepsilon \ll 1$.

Using an explicit time discretization of convection (Lax-Wendroff for example), the penalized Navier-Stokes equations can be written as follows:

$$u^{n+1} - \mu\delta t\Delta u^{n+1} + \chi\varepsilon^{-1}\delta t\, u^{n+1} = RHS \text{ in } \Omega \tag{9}$$

with adequate outflow boundary conditions. In this case, the perturbation c is defined by $c(x) = (\mu\varepsilon)^{-1} + (\mu\delta t)^{-1}$. This formulation is also compatible with an implicit discretization of convection, or with its Lagrangian formulation (as a substep of a Particle-in-Cell or Smoothed-Particle-Hydrodynamics method), each scheme leading to a different expression of the equation (9).

This formulation is compatible with the SMW-GMRES algorithm developed in the previous section since equation (9) can be rewritten as the prototype equation (3).

2.4.2. Navier-Stokes equations in velocity-vorticity formulation

The Penalization method has also been intensively used in Vortex in Cell community. By taking the curl of equation (8) the pressure and external forces f (supposed to derive from a potential) are vanishing and unknowns of the problem are velocity and vorticity linked by the relation $\omega = \text{curl } u$. The reconstruction of u from ω is performed by solving Poisson problems thought the stream function.

By taking the curl of equation (8), the Penalization term is split in two terms:

$$\text{curl}\left(\frac{\chi}{\varepsilon}(u-\overline{u})\right) = \frac{\chi}{\varepsilon}(\omega-\overline{\omega}) + \frac{\delta_\Gamma}{\varepsilon}(u-\overline{u})$$

where δ_Γ is the Dirac function of surface Γ (the solid region boundary).

A semi-implicit treatment of the term curl $\left(\chi\varepsilon^{-1}(u-\overline{u})\right)$ was successfully implemented in [13, 34] with a splitting decomposition. A full implicit numerical computation of this equation is also possible by handling the Dirac term carefully. It gives good results with an adequate splitting, when the treatment of the Dirac is followed by a diffusion sub-step [18]. In this case, the same discretization can be achieved and an equation similar to (3) is solved. Hence the SMW-GMRES algorithm is compatible with the velocity-vorticity formulation.

2.4.3. Stokes equations for highly viscous flows

In this paper an alternative is proposed to solve the penalized term in the same step as the diffusion. In fact when the Reynolds number of the flow is decreasing it is not possible to compute the diffusion explicitly. This situation becomes critical when the Reynolds number is so small that the Navier-Stokes equations are reduced to the Stokes problem:

$$-\text{div}(2\mu D(u)) + \frac{\chi}{\varepsilon}(u-\overline{u}) = f - \nabla p \quad \text{and} \quad \text{div}u = 0 \tag{10}$$

In the following the algorithm is presented in this context of highly viscous flows but, as presented above, the methods can be extended to the Navier-Stokes frameworks.

3. Application: splitting strategies for highly viscous flows in 3D

As the Penalization method is very convenient to combine with a regular grid, it was widely used in computational fluid dynamics to model solid-fluid interactions. This section is presenting a splitting algorithm using the SMW formula to solve the problem of a solid immersed in a highly viscous flow. In this context the Reynolds number is very small and Navier-Stokes equations are reduced to quasi-static Stokes equations. To take into account the influence of the solid on the fluid a Penalization term is added in the conservation of momentum equation.

Eventually μ and u are coupled through a diffusion-transport equation. This arises if the viscosity is a function of a mass fraction α, which also follows a diffusion-transport equation (as a consequence of mass conservation). Biological fluids such as sperm or mucus are non-homogeneous and the use of a variable viscosity flow is justified by their inner composition. Essentially composed of water, these biological liquids are non-homogeneous: it depends on the concentration of seminal fluid (for semen) and mucins (for mucus). In addition amino acids, enzymes and many other different elements compose these fluids, these different compositions influence directly the viscosity. Moreover these components are not at the same ratio considering a healthy person or not, and pathogen situations lead to dramatic changes in viscosity properties. Eventually spermatozoa are great travelers and the swimming medium viscosity is different in seminal fluid and in genital mucus.

$$\begin{cases} -\text{div}(2\mu D(u)) + \dfrac{\chi}{\varepsilon}(u-\overline{u}) = f - \nabla p \\ \text{div}u = 0 \\ \dfrac{\partial\alpha}{\partial t} + u\cdot\nabla\alpha - \eta\Delta\alpha = 0 \\ \mu = \phi(\alpha) \end{cases} \tag{11}$$

Notations are the same as in the previous section. The solid velocity $\overline{u}$ is assumed to be known at each time and not depending on the fluid velocity: it is the 1-way fluid structure interaction. This system has to be completed by suitable boundary conditions.

The convection-diffusion equation is integrated with an explicit scheme, which is particularly interested to combine with particle methods. Thus using a time discretization $t_n = n\delta t$ (with δt the time step), the computation of $\alpha(t_n)$ only requires the knowledge of $u(t_{n-1})$, so the third equation of (11) is decoupled from the Stokes problem (first and second equations), which is solved at each computation of the convection, to transport particles. In the following the presentation of the resolution of convection-diffusion equation and Stokes problem are separated, since, with this integration, u can be seen as a data in the third equation. The Stokes conservation of momentum equation and the incompressibility equation are also decoupled using a fractional step algorithm: a projection method adapted to this quasi-static regime which is briefly detailed bellow.

3.1. Particle methods

The transport part of the third equation of (11) is computed using a Lagrangian method and the diffusion part using an Eulerian method on a Cartesian grid. A splitting strategy is developed to separate both phenomena. Second order splitting is possible with Strang formula (Trotter's permutations are giving higher order's) and an extension enables to mix transport and diffusion inside Rung Kutta sub-steps [10]. In this way the diffusion induces a regularization of α after each convection sub-step, and consequently a regularization of μ.

The diffusion is solved explicitly on a Cartesian grid: it is a standard heat equation with a small diffusion coefficient, so explicit computations are not limited by the stability condition.

For the resulting homogeneous transport equation a Particle-In-Cell method introduced for Navier-Stokes equations is adapted. It consists in the discretization of unknowns on a particle cloud:

$$\mu = \sum_{p=1}^{N} \mu_p \delta_{\xi_p} v_p$$

where $\delta_{\xi_p}(\varphi) = \varphi(\xi_p)$, for any admissible test function φ, is the Dirac function at location ξ_p.

Featuring particles the transport part of the splitting algorithm (meaning the third partial differential equation of (11) without diffusion) is reduced to a set of ordinary differential equations:

$$\frac{\mathrm{d}\mu_p}{\mathrm{d}t} = 0, \quad \frac{\mathrm{d}\xi_p}{\mathrm{d}t} = u(\xi_p), \quad \frac{\mathrm{d}v_p}{\mathrm{d}t} = 0 \tag{12}$$

for particles $p = 1, \ldots, N$, with conservation of particles volume v_p due to the divergence-free velocity field (solution to the invicid Stokes problem).

Then the key point is to transfer unknowns from particles to the grid. The choice of a high order interpolation kernel with a compact support gives good computational performances. In the present work M4' [29] (third order) was used but other kernels presented recently in the literature have a higher accuracy for a similar computational cost [28, 5] (since the FFT-based solver used for computations has second order accuracy a third order interpolation kernel is suitable in this work). The compact support of this kernel implies a linear scalability of the complete method.

3.2. Stokes solver

The projection method introduced by Chorin [11] is a fractional step algorithm to compute separately velocity field and pressure. It separates the resolution of the conservation of momentum equation (first equation of (11)) and the incompressibility (second equation of (11)).

An adaptation of this method to the quasi-static case is used. It has been introduced and validated with details in [9]. The idea is first to estimate the velocity by solving the conservation of momentum equation without the pressure term. This intermediate solution is called u^*. Then this solution is projected on divergence free fields:

$$u = \Pi(u^*) = u^* - \nabla\zeta \tag{13}$$

where the projector ζ is the solution to the following Poisson problem

$$\begin{cases} -\Delta\zeta = -\mathrm{div} u^* \\ \dfrac{\partial\zeta}{\partial n} = u^* \cdot n \quad \text{on boundaries} \end{cases} \tag{14}$$

Cilia radius	$1\mu m$
Computational box size	$15 \times 15 \times 15 \mu m^3$
Spinning frequency	$1Hz$
Mass fraction diffusion	$0\ \mu m^2 s^{-1}$
Reynolds number	$\sim 10^{-6}$

Table 2: Parameters used for the spinning sphere simulation

As reported in [24] the projection of u^* generates an error in a boundary layer. In fact the correction (13) is also performed on the boundary of the computational domain where $\nabla\zeta = 0$ only for the normal component. Thus the tangential components of the corrected velocity field do not verify the original boundary conditions. This problem becomes critical for quasi-static flows since an incremental computation of the projector is not possible, on the contrary to Navier-Stokes equations, and the boundary layer fills the whole domain. This problem is also present in the penalized domain where $\nabla\zeta \neq 0$.

To handle this issue, an implicit formulation of both the boundary conditions and the penalization term is proposed and solved iteratively in order to recover an accurate solution. The idea of this process is to add at each estimation sub-step what is expected to be removed during the projection (13). With this iterative process, terms from non-homogeneous viscosity are also computed explicitly in the same iteration since $\mathrm{div}(2\mu D(u)) = \mu\Delta u + \mu\nabla \mathrm{div} u + 2D(u)\nabla\mu$ and $\mathrm{div} u = 0$. This estimation sub-step reads:

$$\begin{cases} -\mu\Delta u^*_{k+1} + \dfrac{\chi}{\varepsilon}(u^*_{k+1} - \nabla\zeta_k - \bar{u}) = f + 2D(u_k)\nabla\mu + \mathrm{div} u^*_k \nabla\mu \\ u^*_{k+1} = g + \nabla\zeta_k \quad \text{on boundaries} \end{cases} \tag{15}$$

The last term of the first equation of (15) is coming from the substitution of $u = \Pi(u^*)$ back in (11). With:

$$G(u^*_k) = \left(f + 2D(u_k)\nabla\mu + \mathrm{div} u^*_k \nabla\mu + \frac{\chi}{\varepsilon}(\nabla\zeta_k + \bar{u}) \right) / \mu \tag{16}$$

the problem (15) is transformed into the fixed-point problem:

$$\begin{cases} -\Delta u^*_{k+1} + \dfrac{\chi}{\mu\varepsilon} u^*_{k+1} = G(u^*_k) \\ u^*_{k+1} = g - (\Pi(u^*_k) - u^*_k) \quad \text{on boundaries} \end{cases} \tag{17}$$

This problem is solved using the SMW formula coupled with GMRES approach as presented in the previous section. All the Poisson equations of the algorithm (in both the SMW-GMRES coupling and the projection) are solved with FishPack solver. Finally the explicit treatment of $2D(u_k)\nabla\mu + \mathrm{div} u^*_k \nabla\mu$ is decoupling the velocity components computations.

The computational method summarized in this section was validated with several benchmark in [9] and the use of the SMW-GMRES method instead of MudPack do not change these results.

In the next section accuracy and computational performances of SMW based algorithm is discussed for fluid mechanics computations.

4. Numerical simulations of highly viscous flows

In this section three numerical simulations are presented in this context of fluid-structure interaction with a highly non-homogeneous viscous flow. The flow configurations are described, listing the flow parameters: immersed geometries, initial mass fraction (and viscosity) profiles, boundary conditions...

4.1. A sphere mixing a non-homogeneous flow

The first example is a sphere spinning at constant velocity in a Stokes flow with variable viscosity. This simulation was used as a benchmark to validate the conservation of the transported mass fraction in the previous fractional step algorithm [9].

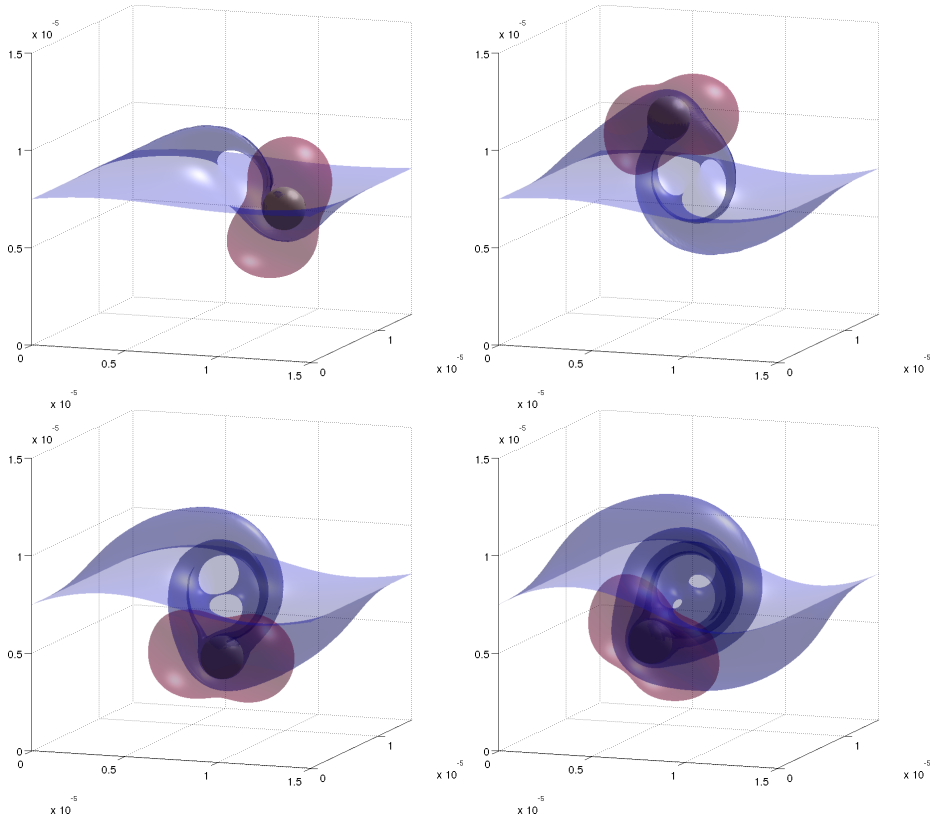

Figure 2: A sphere spinning in a non-homogeneous viscosity fluid. At initialization velocity profile is defined by equation (18) with $H_z = 15\ \mu m$ the height of the computational box. Snapshots present the flow at $t = 0.55s$ (top left), $t = 1.2s$ (top right), $t = 1.75s$ (bottom left) and $t = 2.85s$ (bottom right). In blue is an isosurface of viscosity of level $1.5 \times 10^{-3} Pa.s$ and in red is an isosurface of velocity magnitude of level $5\ \mu m/s$. The grid resolution is $256 \times 256 \times 256$.

At initialization the mass fraction α is stratified according to a linear profile (with H_z the height of computational box) and μ is a linear function of α:

$$\alpha(x, y, z, t = 0) = \alpha_0(x, y, z) = (1 + z/H_z) \quad \mu(x, y, z, t = 0) = \mu_0(x, y, z) = \mu_{water}\alpha_0 \tag{18}$$

Thus the fluid is two times more viscous at the top than at the bottom. It is then transported by the computed Stokes flow. In this simulation α is following a pure convection equation ($\eta = 0$). Even if it is easier to implement since no splitting is required for the conservation of mass, the absence of diffusion is not giving any regularization to viscosity so the gradient is growing and the RHS of equation (17) is less regular. The following equation gives analytically the sphere velocity $\overline{u}$ and its position $\overline{X}$ used for computations at any time t, with H_z the height of the computational box.

$$\overline{X}(t) = \left(\frac{H_z}{4}\cos 2\pi t, 0, \frac{H_z}{4}\sin 2\pi t\right) \quad \text{and} \quad \overline{u}(t) = \left(-\frac{\pi H_z}{2}\sin 2\pi t, 0, \frac{\pi H_z}{2}\cos 2\pi t\right) \tag{19}$$

To be under the Stokes assumption the sphere is microscopic with a diameter $R = 1\ \mu m$ and the computational box is $(15\ \mu m)^3$. Water's viscosity is $10^{-3} Pa.s$ and the Penalization parameter is set to 10^{-13}. This simulation parameters are gathered in the table 2. The boundary conditions are periodic in the y direction and an homogen Dirichlet boundary condition is imposed on the other faces of the computational box.

Figure 2 presents snapshots of the flow created by this spinning sphere. Isosurfaces of velocity magnitude of level $5\ \mu m/s$ and viscosity of level $1.5 \times 10^{-3} Pa.s$ are presented. In a homogeneous Stokes flow passing a sphere

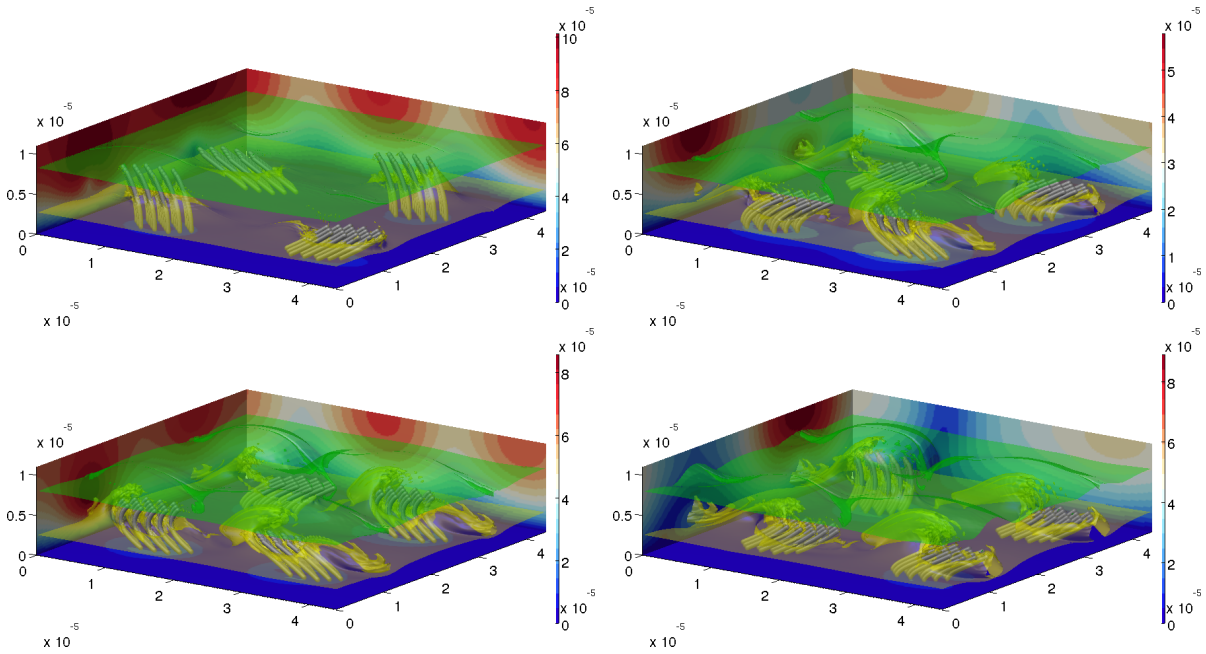

Figure 3: Different snapshots of four epithelium cells beating at $4Hz$ in a pulmonary mucus at $t = 0.187\ s$ (top left), $t = 0.812\ s$ (top right), $t = 1.104\ s$ (bottom left) and $t = 1.25\ s$ (bottom right). The viscosity was stratified at initialization. The simulation resolution is $512 \times 512 \times 64$. In yellow an isosurface of viscosity of level $1.25 \times 10^{-3} Pa.s$ is shown. In green is an isosurface of viscosity of level $1.75 \times 10^{-3} Pa.s$. Colors of the boundary are quantifying the velocity norm.

Cilia length	$7\mu m$
Cilia radius	$0.3\mu m$
Computational box size	$60 \times 15 \times 15 \mu m^3$
Beating frequency	$4 - 20Hz$
Mass fraction diffusion	$10^{-3} \mu m^2 s^{-1}$
Reynolds number	$\sim 5 \times 10^{-4}$

Table 3: Parameters used for the mucus flow simulation

isosurfaces of velocity magnitude have a characteristic almond-shape. Here this almond is slightly deformed by the boundary conditions and the non linearity induced by the viscosity convection.

4.2. Mucus film in human lung

The second application investigates mucociliary clearance: the natural phenomenon which transports mucus in the lung from distal to proximal airways. This mucus flows as a film which protects bronchial walls from inhaled pathogens, pollution particles, dust... That is why it is continuously evacuated to prevent pathogens proliferation. To propel the mucus, bronchus are covered by a few micrometers cilia oscillating at frequencies about $4 - 20Hz$ in the fluid.

In [9] we presented a simulation of a single cell involving a few cilia beating in a synchronous way. Here we show that the SMW approach fits very well for a more complex problem: the simulation considers 4 cells of 19 cilia beating in an asynchronous way so 76 cilia are modeled in this simulation.

Each cilium is modeled with a parametric curve. Let $P(\zeta, t)$ be the curve position describing cilium at time t at a distance l from the bronchial wall: $P :]0, 1[\longrightarrow \mathbb{R}^3$. Let L be the derivative of P with respect to ζ. The cilium beating is produced by solving a first order hyperbolic partial differential equation set on L:

$$\frac{\partial L}{\partial t} + v(t)\frac{\partial L}{\partial \zeta} = 0 \tag{20}$$

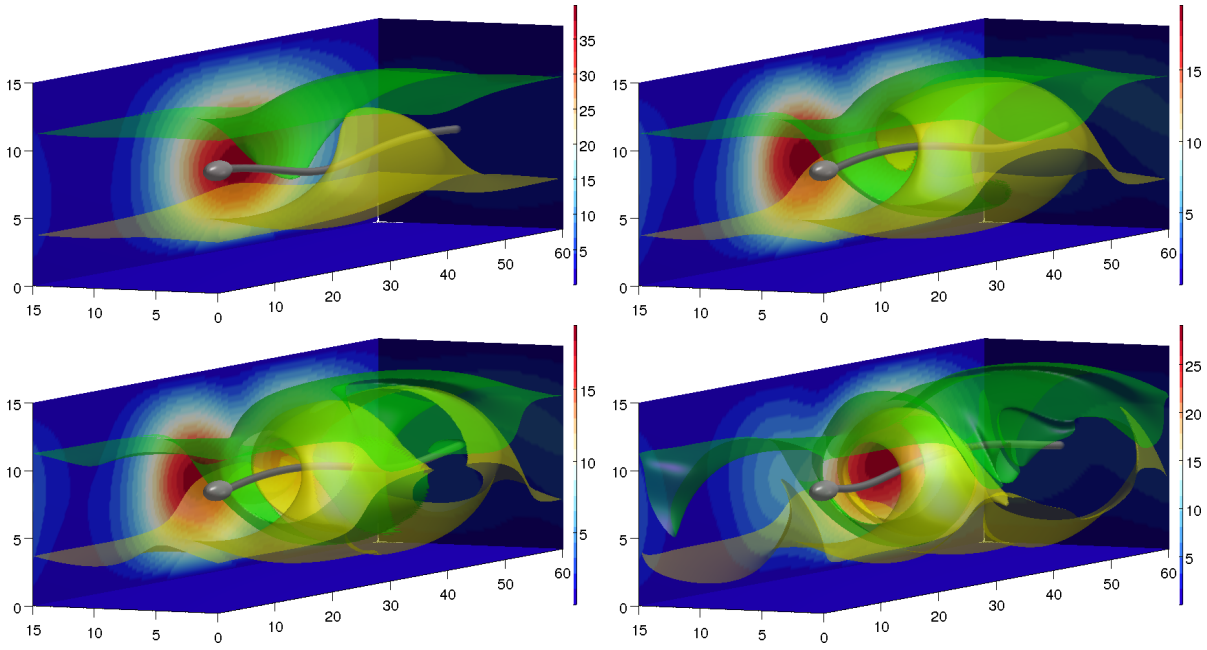

Figure 4: Different snapshots of the spermatozoon helical beating in a non-homogeneous Stokes fluid. The simulation resolution is $512 \times 64 \times 64$. In green an isosurface of viscosity of level $1.75 \times 10^{-3} Pa.s$ is shown and in yellow a second isosurface of viscosity of level $1.25 \times 10^{-3} Pa.s$. Colors of the boundary are quantifying the velocity norm.

The solution to this first order hyperbolic equation is also solution to the following damped wave equation:

$$\frac{\partial^2 L}{\partial t^2} - v^2(t)\frac{\partial^2 L}{\partial \zeta^2} = -v'(t)\frac{\partial L}{\partial \zeta} \tag{21}$$

With v^2 a Young modulus, quantifying the elasticity. The boundary conditions of equation (20) are producing the beating: $P(0,t)$ remains constant to ensure that the cilium is hooked up to the bronchial wall and $L(0,t) = \partial_\zeta P(0,t) = g(t)$ is producing the cilium beating (see [8] for details). Eventually T is the beating period and a decay is added to g to get an asynchronous beating between cells. Finally $P(1,t)$ is free to evolve because no boundary condition is required for the outer boundary of a transport equation if the transport field is positive. This 1D cilium model is very cheap to solve numerically even with implicit methods.

For this simulation the same initial mass fraction and relation linking μ and α as for micrometrix sphere computation is used (see equation (18)). The other simulation parameters are listed in the table 3. The boundary conditions are periodic in the x and y direction, an homogeneous Dirichlet boundary conditions ensures a no-slip boundary conditions at the bottom of the computational box (since it models the bronchial wall) and a tangential velocity is authorized at the top of the computational box, assuming a flat interface between air and mucus. This reads:

$$\frac{\partial u_x}{\partial n} = \frac{\partial u_y}{\partial n} = u_z = 0 \qquad \text{on the top face of the computational box} \tag{22}$$

Figure 3 presents two isosurfaces of viscosity at different times of the beating. It also shows velocity magnitude on the boundaries of computational box since periodic boundary conditions were used in x and y direction. The motion of cilia is transporting the mucus film in the positive x direction, it means mucociliary clearance is effective and results are compatible with medical observations [9].

4.3. A microswimmer

A third numerical illustration is presenting the computation of a microswimmer. At microscopic scales swimmers have to develop strategies to break their motion symmetry to move. Indeed at these scales fluid dynamics is governed

Flagellum length	$40\mu m$
Flagellum radius	$0.3\mu m$
Computational box size	$44 \times 44 \times 11\mu m^3$
Beating frequency	$10Hz$
Mass fraction diffusion	$10^{-3}\mu m^2 s^{-1}$
Reynolds number	$\sim 10^{-2}$

Table 4: Parameters used for the spermatozoon simulation

by Stokes equations which are reversible in time, as a consequence of Purcell's scallop theorem [33]. Various microswimmer behaviors have been investigated in the literature, most of them use one or several flagella to propel and many different beating patterns can be observed. Some of them are straightly deforming their body to swim without using any flagella. A complete review of microswimming agents was recently published [27].

One of the most studied microswimmer of the literature is the spermatozoon. Two recent reviews are summarizing the work of many authors [22, 20] with both experimental and numerical studies and discussions.

Depending on the mammalian species studied various features are observed (different size, beating patterns, ...). In this section a simple model of a spermatozoon is proposed to illustrate a possible application of the previous algorithm. The spermatozoon is described with an ovoid head (which contains the genetic material) and a beating flagellum. The beating pattern is helical and dimensions are chosen to get a reasonable approximation of a human spermatozoon swimming in a liquid with stratified viscosity.

Computations are performed in the spermatozoon frame of reference: the head is not moving with respect to the computational box. It is modeled by a revolution ellipsoid with a major axis of $4\mu m$ and minor axises of $1.5\mu m$. Flagellum is described as a cilium, with the same beating model as previously, but parameters are changed: v is constant and boundary conditions are changed to produce an helical beating: $P(0,t) = (0,0,0)$ ensures that flagellum is hooked up to the head and $L(0,t) = \partial_\xi P(0,t) = (\cos(\theta), \sin(\theta)\sin(2\pi t/T), \sin(\theta)\cos(2\pi t/T))$ produces the helical beating inside a right circular cone with an axis oriented in the positive x direction and a half top angle $\theta = \pi/25$. Flagellum is $40\mu m$ long so it is a good approximation for a human spermatozoon.

For this simulation the same initial mass fraction (and relation linking μ and α) as for the micrometrix sphere simulation is used (see equation (18)). Boundary conditions are also identical (see equation (22) and comments in the text). The other simulation parameters are listed in the table 4.

On the figure 4 snapshots are presented. Two isosurfaces of viscosity at different times of the beating are drawn and the velocity magnitude on the boundaries is also presented. Flagella is creating a motion in the fluid which is mixing convected mass. Viscosity is transported in the positive x direction. It means spermatozoon performs an efficient swim in the opposite direction. The mean velocity of the fluid has been computed during the simulation and the result is $0.47\mu m/s$. This model reproduces good spermatozoon features, compatible with literature observations [22], even if it is a very simplified model (particularly since the 2-ways coupling is neglected). It is nevertheless a very good example to test the efficiency of our novel algorithm on a complex geometry.

5. Results: computational performances of the SMW-GMRES algorithm for highly viscous flow simulations

In this section the computational performances of the SMW-GMRES algorithm are presented for the 3D fluid mechanics applications described in previous paragraphs. The discussion focuses on the geometry dependency and computational time, comparing the algorithm with the straight use of MudPack. The memory footprint of the algorithm is not discussed in this section since the same comments savings as in section 2.3 are observed.

To discuss the performances of this novel computational method, the matrix solved with the GMRES algorithm is explicitly computed to investigate its eigenvalues distribution and conditioning number. Figure 5 presents two plots of these eigenvalues for the three different numerical simulations presented previously, using different refinements.

For all these geometries, the eigenvalues are decreasing logarithmically and the conditioning numbers remains small. It remains small for all the geometries and grows slowly with the resolution. Hence the method is not very sensitive to obstacles geometry, to the opposite of kernel methods like Stokeslets, where the number of source points

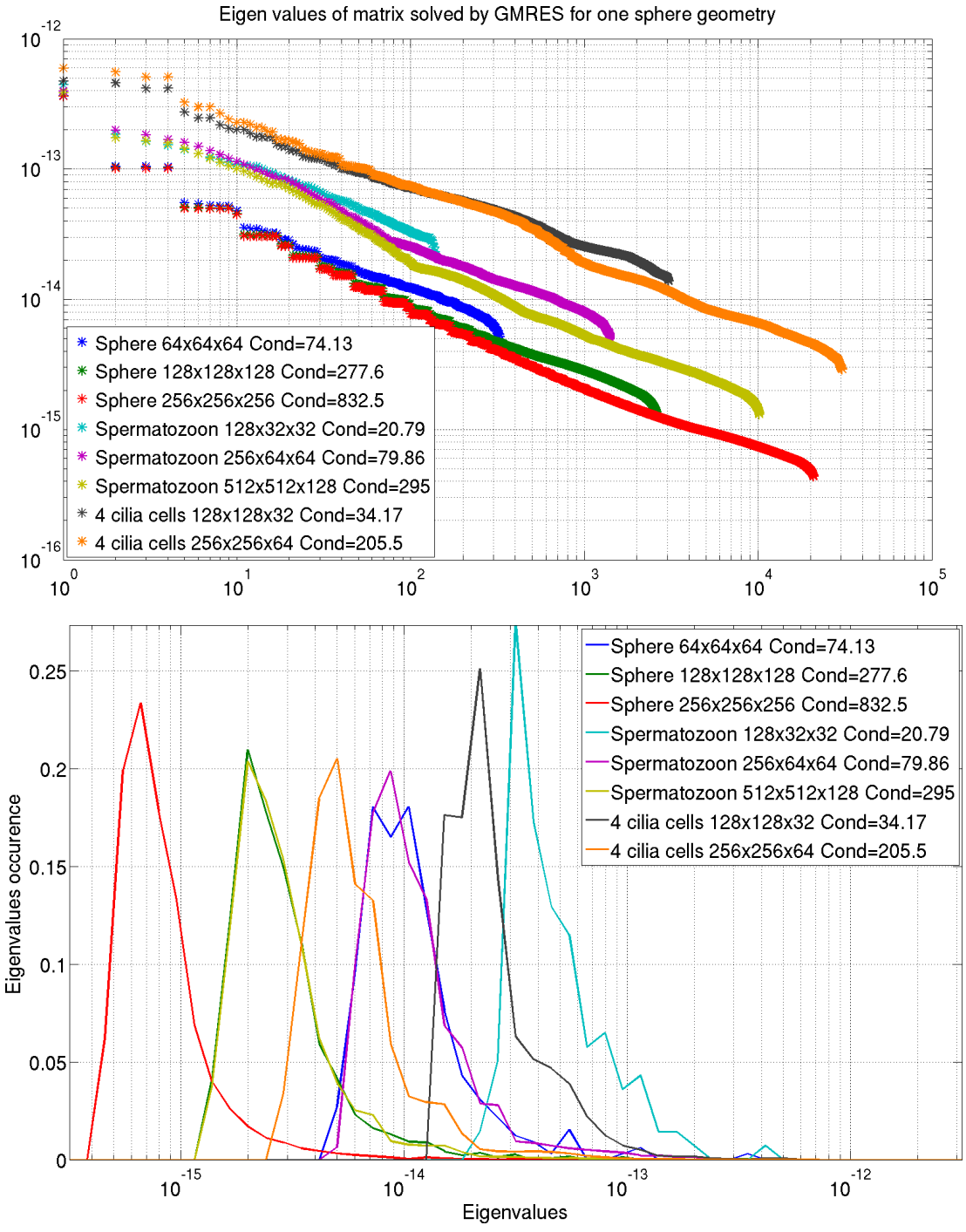

Figure 5: Eigenvalues of the matrix solved by the GMRES method (raw plot on top picture and their distribution on bottom picture). Results are presented for the three simulations of the section 4, using various resolutions.

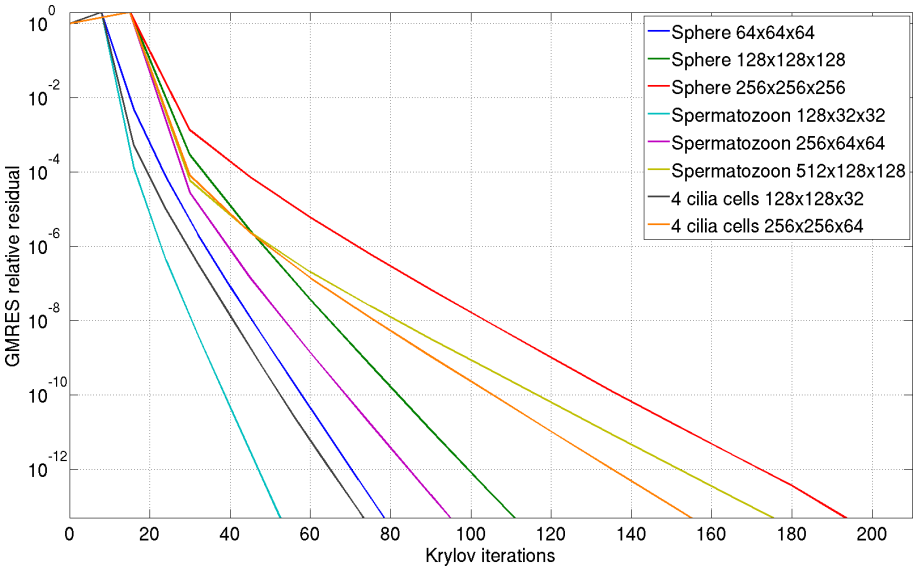

Figure 6: Residuals of the iterative GMRES process for the three flow configurations using different refinements. The process is restarted after 8 (for small matrices) or 15 (for the others) Krylov iterations.

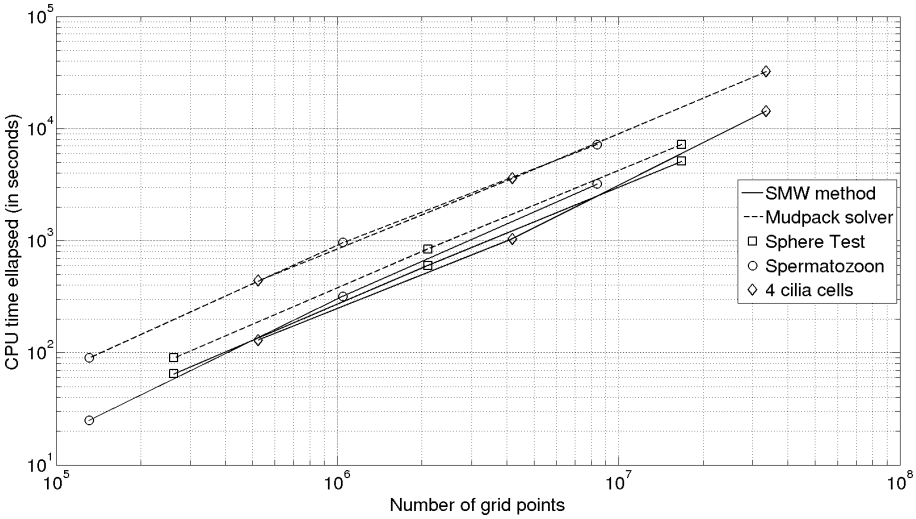

Figure 7: Computational time elapsed to solve one step of RK2 transport algorithm. It involves two calls of the tokes solver (using 8 fixed point iterations) plus Lagrangian transport. Results compare the efficiency of Mudpack solver (solving directly Helmholtz problem with multigrid method) and the SMW method with GMRES approach. Sequential computations performed on an Intel Xeon X5450.

(which is directly linked to the computational time) needs to grow with both the geometry complexity (number of obstacles) and the refinement.

This result insures fast computations with GMRES in the step 2 of the algorithm (described in section 2.2), since the residuals (presented on the figure 6) are decreasing quickly. For these simulations the GMRES method was restarted after 8 (for small matrices) or 15 (for the others) Krylov iterations. This figure shows that, for all the tested geometries, this number of Krylov iterations needs to grow slower than the refinement. For example, using 64 times more grid points (meaning 4 times more in each direction), the number of Krylov iterations is only multiplied by 2.5 to 3.

From the computational time point of view a comparison between the use of the multigrid solver MudPack and the present SMW-GMRES approach at each iteration of Stokes solver is presented on the figure 7, for the three previous simulations. Results were obtained for sequential computations on an Intel Xeon X5450. Computational times are presented for one step of RK2, meaning two sub steps of Lagrangian transport on particles (presented in section 3.1) and two executions of the iterative Stokes solver on the grid (presented in section 3.2), plus interpolations between the grid and particles.

For the sphere spinning test, the computational time with the present algorithm is 30% cheaper than with the multigrid solver (it is a bit smaller than the result of section 2.3 since MudPack handles variable viscosity slightly better than FishPack). The result becomes much more interesting (comparable to the result of section 2.3) when the geometry is more complex. For instance on the four cilia cells and spermatozoon computations the SMW-GMRES approach is 50% faster than the multigrid solver. Indeed, in order to handle jumping coefficients in the Helmholtz problem, the multigrid solver has to perform more multigrid cycles so computations are longer.

6. Conclusions and perspectives

In computational fluid mechanics, the Penalization method is a convenient way to handle a fluid-structure interaction on a regular grid. This paper introduced a novel algorithm to solve implicitly in the same step both the Penalization and the diffusion terms. The problem is decomposed in a series of Poisson problems using the Sherman-Morrison-Woodbury formula coupled to a GMRES approach. Each matrix-vector product in this Krylov algorithm is replaced by a call to a FFT-based solver for the Poisson problem, so no matrix assembling or linear system inversion is required. The eigenvalues of the operator solved by the GMRES method are well distributed, so the corresponding condition number remains small, even when using a fine refinement: this guarantees the use of a small dimension for the Krylov space. The FFT-based solvers are used to get fast computations of these Poisson equations, it is particularly interesting for large 3D problems.

This algorithm has been implemented to compute highly viscous flows interacting with immersed obstacles. Using this novel numerical method, the computation of the fluid velocity, solution to the penalized Stokes problem, is reduced to a sequence of Poisson problems. This problem is coupled to a convection-diffusion equation, describing the evolution of the viscosity, which is discretised by a Lagrangian method. This hybrid grid-particle method have a fast computational efficiency for 3D problems. This is notably a consequence of the lack of stability condition of the Lagrangian methods which enables to perform large time steps, compatible with implicit computations of Penalization and diffusion, hence unnecessary velocity resolutions are avoided.

Different flow configurations were simulated with this numerical method: first a simple "toy" Poisson problem, then a more complex fluid dynamics application with a sphere spinning in a stratified viscous flow. The other applications are concerning biomedical computing: the mucus transport by beating epithelium cells in the lung and the swim of a spermatozoon. For all these applications, computations were compared with the use of a multigrid solver.

The present numerical method was shown to be more efficient from both the computational time and the memory load. Moreover, this algorithm has a small dependency with respect to the geometry (number and shape of obstacles). Hence it fits very well for 3D computations of complex flows.

A parallel extension of the SMW-GMRES algorithm is a possible research perspective. The most expensive part in the resolution is the inversion of the matrix Q (see equation (7)) solved by the GMRES algorithm. The parallelization of the GMRES method has been widely studied by several authors (see for example [19, 16] and references therein) who identified two expensive steps: the matrix vector products and the Arnoldi orthogonalization. In the present methodology, each matrix-vector product can be parallelized by using a parallel FFT-based solver such

as FFTW [21] (this also enables the parallel computation of steps 1 and 3 of the algorithm). The parallelization of the Arnoldi orthogonalization is more subtle, since in this process each vector is computed orthogonal with respect to the previously computed, which is possible only in a sequential way. These authors have proposed several solutions, the most popular is an independent parallel computation of a base of non-orthogonalized vectors: they are then orthogonalized by a parallelizable process (a *QR* factorization for example). The rest of the algorithm is negligible from a computational point of view: the restriction and exclusion operators are not assembled since they can be immediately deduced from the characteristic function of the penalized domain.

Another extension will be its implementation for the Navier-Stokes equations: as presented in section 2 the methodology would not change very much for the Navier-Stokes system if a good splitting is chosen. Its implementation with a 2-ways fluid-structure interaction was also briefly described and might also be considered. Eventually as this algorithm fits very well for sequential computations a massive use with parallel simulations will be investigated for data assimilation.

Acknowledgments

This work is supported by the ANR Grant BioFiReaDy, under the contract number ANR-2010-JCJC-0113-01.
This work was granted access to the HPC resources of CALMIP under the allocation P0314.

References

[1] John C. Adams. mudpack: Multigrid portable fortran software for the efficient solution of linear elliptic partial differential equations. *Applied Mathematics and Computation*, 34(2, Part 2):113–146, November 1989.
[2] Christopher Anderson and Claude Greengard. On vortex methods. *SIAM Journal on Numerical Analysis*, 22(3):413–440, June 1985.
[3] Philippe Angot, Charles-Henri Bruneau, and Pierre Fabrie. A penalization method to take into account obstacles in incompressible viscous flows. *Numerische Mathematik*, 81(4):497–520, 1999.
[4] J. Thomas Beale and Andrew Majda. Rates of convergence for viscous splitting of the navier-stokes equations. *Mathematics of Computation*, 37(156):243–259, October 1981.
[5] Michael Bergdorf and Petros Koumoutsakos. A lagrangian Particle–Wavelet method. *Multiscale Modeling & Simulation*, 5(3):980–995, January 2006.
[6] Philippe Chatelain, Alessandro Curioni, Michael Bergdorf, Diego Rossinelli, Wanda Andreoni, and Petros Koumoutsakos. Billion vortex particle direct numerical simulations of aircraft wakes. *Computer Methods in Applied Mechanics and Engineering*, 197(13–16):1296 – 1304, 2008.
[7] Philippe Chatelain and Petros Koumoutsakos. A fourier-based elliptic solver for vortical flows with periodic and unbounded directions. *Journal of Computational Physics*, 229(7):2425–2431, April 2010.
[8] Robin Chatelin. *Méthodes numériques pour l'écoulement de Stokes 3D: fluides à viscosité variable en géométrie complexe mobile ; application aux fluides biologiques*. PhD thesis, Université Toulouse 3 Paul Sabatier, 2013.
[9] Robin Chatelin and Philippe Poncet. A hybrid grid-particle method for moving bodies in 3D stokes flow with variable viscosity. *SIAM Journal on Scientific Computing*, 35(4):B925–B949, August 2013.
[10] Robin Chatelin, Philippe Poncet, Alain Didier, Marlène Murris-Espin, Dominique Anne-Archard, and Marc Thiriet. Mucus and ciliated cells of human lung : Splitting strategies for particle methods and 3d stokes flows. In *IUTAM Symposium on Particle Methods in Fluid Mechanics*, 2012.
[11] Alexandre Joel Chorin. Numerical solution of the navier-stokes equations. *Mathematics of Computation*, 22(104):745–762, October 1968.
[12] Roger Cocle, Groire Winckelmans, and Goic Daeninck. Combining the vortex-in-cell and parallel fast multipole methods for efficient domain decomposition simulations. *Journal of Computational Physics*, 227(21):9091–9120, November 2008.
[13] M. Coquerelle and G.-H. Cottet. A vortex level set method for the two-way coupling of an incompressible fluid with colliding rigid bodies. *Journal of Computational Physics*, 227(21):9121–9137, November 2008.
[14] Georges-Henri Cottet and Petros D. Koumoutsakos. *Vortex Methods: Theory and Practice*. Cambridge University Press, March 2000.
[15] Benoit Couet, Oscar Buneman, and Anthony Leonard. Simulation of three-dimensional incompressible flows with a vortex-in-cell method. *Journal of Computational Physics*, 39(2):305–328, February 1981.
[16] Rudnei Dias da Cunha and Tim Hopkins. A parallel implementation of the restarted GMRES iterative algorithm for nonsymmetric systems of linear equations. *Advances in Computational Mathematics*, 2(3):261–277, April 1994.
[17] A. Decoene, S. Martin, and B. Maury. Microscopic modelling of active bacterial suspensions. *Mathematical Modelling of Natural Phenomena*, 6(05):98–129, 2011.
[18] M. El Ossmani and P. Poncet. Efficiency of multiscale hybrid grid-particle vortex methods. *Multiscale Modeling & Simulation*, 8(5):1671–1690, January 2010.
[19] Jocelyne Erhel. A parallel GMRES version for general sparse matrices. *Electronic Transactions on Numerical Analysis*, 3(12):160–176, 1995.
[20] Lisa J. Fauci and Robert Dillon. Biofluidmechanics of reproduction. *Annual Review of Fluid Mechanics*, 38(1):371–394, 2006.
[21] M. Frigo and S.G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005.

[22] E.A. Gaffney, H. Gadêlha, D.J. Smith, J.R. Blake, and J.C. Kirkman-Brown. Mammalian sperm motility: Observation and theory. *Annual Review of Fluid Mechanics*, 43(1):501–528, 2011.

[23] Mattia Gazzola, Philippe Chatelain, Wim M. van Rees, and Petros Koumoutsakos. Simulations of single and multiple swimmers with non-divergence free deforming geometries. *Journal of Computational Physics*, 230(19):7093–7114, August 2011.

[24] J.L. Guermond, P. Minev, and Jie Shen. An overview of projection methods for incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 195:6011–6045, September 2006.

[25] W. W. Hager. Updating the inverse of a matrix. *SIAM Rev.*, 31(2):221–239, June 1989.

[26] J. Janela, A. Lefebvre, and B. Maury. A penalty method for the simulation of fluid-rigid body interaction. In *ESAIM: Proceedings*, volume 14, pages 115–123, 2005.

[27] Eric Lauga and Thomas R Powers. The hydrodynamics of swimming microorganisms. *Reports on Progress in Physics*, 72(9):096601.1–096601.36, September 2009.

[28] Adrien Magni and Georges-Henri Cottet. Accurate, non-oscillatory, remeshing schemes for particle methods. *Journal of Computational Physics*, 231(1):152–172, January 2012.

[29] J.J Monaghan. Extrapolating b splines for interpolation. *Journal of Computational Physics*, 60(2):253–262, September 1985.

[30] P Ploumhans and G.S Winckelmans. Vortex methods for high-resolution simulations of viscous flow past bluff bodies of general geometry. *Journal of Computational Physics*, 165(2):354–406, December 2000.

[31] Philippe Poncet. Topological aspects of three-dimensional wakes behind rotary oscillating cylinders. *Journal of Fluid Mechanics*, 517:27–53, 2004.

[32] Philippe Poncet. Analysis of an immersed boundary method for three-dimensional flows in vorticity formulation. *Journal of Computational Physics*, 228(19):7268–7288, October 2009.

[33] EM Purcell. Life at low reynolds number. *American Journal of Physics*, 45(1):3–11, 1977.

[34] Johannes Tophj Rasmussen, Georges-Henri Cottet, and Jens Honor Walther. A multiresolution remeshed vortex-in-cell algorithm using patches. *Journal of Computational Physics*, 230(17):6742–6755, July 2011.

[35] Y Saad and MH Schultz. GMRES: a generalized minimal residual method for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869, 1986.

[36] Paul Swarztrauber and Roland Sweet. Efficient FORTRAN subprograms for the solution of elliptic partial differential equations (abstract). *SIGNUM Newsl.*, 10(4), December 1975.

[37] Roland A. Sweet. A parallel and vector variant of the cyclic reduction algorithm. *SIAM Journal on Scientific and Statistical Computing*, 9(4):761–765, July 1988.

[38] Tayfun E. Tezduyar. Finite element methods for flow problems with moving boundaries and interfaces. *Archives of Computational Methods in Engineering*, 8(2):83–130, 2001.