Direct numerical simulation of complex viscoelastic flows via fast lattice-Boltzmann solution of the Fokker-Planck equation

L. Bergamasco[a,*], S. Izquierdo[a], A. Ammar[b]

[a] *Instituto Tecnológico de Aragón (ITA) - María de Luna 8, 50018 Zaragoza, Spain*
[b] *Arts et Métiers ParisTech, Boulevard du Ronceray, BP 93525, F-49035 Angers cedex 01, France*

Abstract

Micro-macro simulations of polymeric solutions rely on the coupling between macroscopic conservation equations for the fluid flow and stochastic differential equations for kinetic viscoelastic models at the microscopic scale. In the present work we introduce a novel micro-macro numerical approach, where the macroscopic equations are solved by a finite-volume method and the viscoelastic equation by a lattice-Boltzmann one. The kinetic model is given by molecular analogy with a finitely extensible non-linear elastic (FENE) dumbbell and is deterministically solved through an equivalent Fokker-Planck equation. The key features of the proposed approach are: (i) a fast solution of the microscopic scale equation; (ii) the transport of stresses in Eulerian framework, thanks to an operator-splitting procedure on the Fokker-Planck equation. This latter feature allows application of the proposed method to non-homogeneous flow conditions. The model optimization is achieved through an extensive analysis of the lattice Boltzmann solution, which provides control on the numerical error in the domain and on the computational time. The resulting micro-macro model is validated against the two-dimensional benchmark problem of a viscoelastic flow past a confined cylinder. As a further improvement, three acceleration strategies for the lattice Boltzmann solution on graphic processing units are introduced and the relative speed-up discussed.

*Corresponding author
Email addresses: lbergamasco@ita.es (L. Bergamasco), sizquierdo@ita.es (S. Izquierdo), amine.ammar@ensam.eu (A. Ammar)

Keywords: Multi-scale, finite volume method, lattice Boltzmann method, FENE kinetic model, GPU computing.

1. Introduction

One of the most commonly adopted practices for the simulation of dilute polymeric suspensions relies on macroscopic constitutive equations, derived from molecular models and solved via well-established numerical methods [1]. The advantage of this approach is the low computational cost associated, the drawback is that some kinetic models does not have a closed-form continuous counterpart. With regards to the finitely extensible non-linear elastic (FENE) model for example, a rheological law can only be derived under closure approximations (i.e. FENE-P, FENE-LS) [2]. The resulting models are then able to phenomenologically describe the basic flow features but the underlying theoretical assumptions can hinder accuracy in the retrieval of relevant viscoelastic phenomena.
In a more general modeling strategy, the kinetic origin of the molecular models is retained [3]. Methods using this approach are generally described as micro-macro models, due to the separated solution of the micro and macro scales. Continuity and momentum equations are solved using continuous equations (macro-scale) and kinetic equations are solved by stochastic or deterministic methods (micro-scale) [4]. In this framework, one of the most popular methodologies is the CONNFFESSIT approach, where a finite element solution of the macroscopic equations is combined with stochastic simulations for the dumbbell configuration [5]. One of the major issues concerned with this approach is the high computational expense and the embedded statistical noise, which can be filtered using variance reduction techniques [6]. Another similar and commonly used approach is the Brownian configuration field method [7]. This method already embeds efficient variance reduction, as long as individual molecules are clustered in continuous configuration fields according to their initial configuration and applied force, but the computational cost of the stochastic simulation is anyway a limit.
An alternative approach for noise reduction and faster computations consists in the solution of an equivalent Fokker-Planck equation for the probability density of the dumbbell configuration. However, a literature review reveals that due to the dimensionality of the problem and the lack of efficient numerical methods to solve the Fokker-Planck equation, little progress has been

done in this framework [4] and no method prevail. Relevant recent work about the direct solution of the Fokker-Planck equation for complex flows relies on a Galerkin spectral element technique for 2D [8] and the relative extension to 3D [9].
Another group of promising methods are those that approximate the solution of the Fokker-Planck equation reducing the dimensionality of the problem. This order-reduction can be done *a priori*, like in the lattice-Fokker-Planck method [10], or *a posteriori* like in the proper generalized decomposition [11]. Either technique aims to systematically reduce the degrees of freedom and therefore the computational expense.
In this work we focus on direct deterministic numerical methods, therefore no approximation occurs beyond mesh resolution. The proposed approach relies on a previous work by Ammar [12] about a lattice Boltzmann solution of the Fokker-Planck equation for homogeneous flows. Recently this method has been also theoretically analyzed [13] and applied for the solution of a population balance equation (a kinetic-like equation similar to the Fokker-Planck) [14]. However, none of the previous works [12, 13, 14] deals with the coupling of the kinetic solution with macroscopic fields, thus we investigate efficient ways to exploit it in multi-scale simulations. In the proposed micro-macro model, the macroscopic equations are solved by a finite-volume method using the commercial solver ANSYS Fluent v14.0, while the microscopic equation is solved by a lattice-Boltzmann method. The Fokker-Planck equation is solved using a time-splitting procedure, which allows the transport of stresses in a full Eulerian framework and thus the application to non-homogeneous flows.
The outlines of the paper are as follows: the governing equations for the polymeric suspension and a derivation of the stochastic equation for the FENE dumbbell model are firstly introduced; successively, the equivalent Fokker-Planck equation is derived and the solution strategy explained (Section 2). In the following (Section 3), the numerical approaches for the continuum and the kinetic equations are discussed, together with the algorithm for the coupled solution. Section 4 comprises the numerical analysis of the sub-grid solution, the validation of the coupled model and its optimization. The last section 5 is dedicated to the acceleration on graphic card and to the relative coupling with the macroscopic solver. A brief summary of the results obtained and an outlook on further developments concludes the paper (Section 6).

2. Theoretical model

2.1. Hydrodynamic system

Let us consider a polymeric solution of a Newtonian and a viscoelastic fluid. Assuming the flow to be incompressible and isothermal, mass and momentum conservation reads:

$$\nabla_x \cdot \mathbf{v} = 0; \tag{1}$$

$$\rho \frac{\partial \mathbf{v}}{\partial t} + \rho \nabla_x \cdot (\mathbf{v} \otimes \mathbf{v}) = -\nabla_x p + \nabla_x \cdot \boldsymbol{\sigma}; \tag{2}$$

where ρ is the density, p the pressure, $\mathbf{v}$ the velocity vector and the subscript x denotes operators in the physical space. For the sake of clarity, we remark that the transient and the advective terms are reported for generality, but in the following we drop them as long as in this paper we are concerned with steady-state Stokes flow. The total stress tensor $\boldsymbol{\sigma}$, embeds contributions from both the Newtonian solvent $\boldsymbol{\sigma}_s$ and the polymeric solute $\boldsymbol{\sigma}_p$, therefore $\boldsymbol{\sigma} = \boldsymbol{\sigma}_s + \boldsymbol{\sigma}_p$. Denoting by μ_s the dynamic viscosity of the solvent, $\boldsymbol{\sigma}_s$ is given by Newton's law as:

$$\boldsymbol{\sigma}_s = \mu_s \left(\nabla_x \mathbf{v} + (\nabla_x \mathbf{v})^\dagger \right) = \mu_s \dot{\boldsymbol{\gamma}}; \tag{3}$$

having indicated with $\dot{\boldsymbol{\gamma}}$ the rate of strain tensor. In order to close the hydrodynamic system, an additional equation must be solved for the viscoelastic contribution $\boldsymbol{\sigma}_p$.

2.2. Viscoelastic model

In the simplest micro-mechanical approach for polymer rheology, molecular chains are modeled by two beads and a spring connector, that is by a non-rigid dumbbell immersed in a fluid. A general kinetic model can then be derived considering the equations of motion of the beads in the dumbbell, namely the equilibrium of inertial, frictional, Brownian and connector forces [15]. For a j-th bead located in $\mathbf{r}_i$, the equilibrium yields the so called *Langevin* equation:

$$m_j \frac{d}{dt} \left(\frac{d\mathbf{r}_j}{dt} - \mathbf{v}\left(\mathbf{r}_j\right) \right) = \zeta_j \left(\frac{d\mathbf{r}_j}{dt} - \mathbf{v}\left(\mathbf{r}_j\right) \right) + \sigma \frac{d\mathbf{W}_j}{dt} + \mathbf{F}_j{}^c; \tag{4}$$

with m being the mass of the bead, ζ a drag coefficient, σ a coefficient for the standard Wiener process $\mathbf{W}$ and $\mathbf{F}^c$ the connector force. Indicating with k_B the Boltzmann constant and T the absolute temperature, $\sigma = \sqrt{2k_B\zeta T}$ from the principle of equipartition of energy [1]. Assuming high friction regime and thus over-dumped Brownian dynamics [16], the inertial term on the left-hand side can be dropped and, indicating with $\boldsymbol{\xi} = \mathbf{r}_2 - \mathbf{r}_1$ the end-to-end vector of a dumbbell, yields the following (Itô) stochastic differential equation:

$$\frac{d}{dt}\boldsymbol{\xi} = \boldsymbol{\kappa}\cdot\boldsymbol{\xi} - \frac{2}{\zeta}\mathbf{F}^c\left(\boldsymbol{\xi}\right) + \sqrt{\frac{4k_BT}{\zeta}}\frac{d}{dt}\mathbf{W}; \tag{5}$$

where $\mathbf{W}$ is a standard Brownian motion $\left(\mathbf{W}_2 - \mathbf{W}_1\right)/\sqrt{2}$ and the symbol $\boldsymbol{\kappa}$ has been adopted for the transpose of the velocity gradient tensor $\left(\nabla_x\mathbf{v}\right)^\dagger$. The peculiarity of the dumbbell model lies in the expression of the connector force law $\mathbf{F}^c\left(\boldsymbol{\xi}\right)$. In this work we are concerned with the finitely extensible non-linear elastic model, therefore indicating with h the spring constant and ξ_0 a finite extensibility parameter, the connector force reads:

$$\mathbf{F}^c\left(\boldsymbol{\xi}\right) = \frac{h}{1 - \left|\left|\boldsymbol{\xi}\right|\right|^2/{\xi_0}^2}\boldsymbol{\xi}; \tag{6}$$

with $||\cdot||$ indicating vector norm. This entropic force law, originally proposed by Warner [17], exhibits linear behavior for small extensions and the finite length ξ_0 in the limit of an infinite force. In a stochastic approach, Eq. (5) should then be stochastically solved for the dumbbell configurations in the random process $\mathbf{W}$ with the spring force law (6).
The ordinary stochastic differential equation (5), can be associated with a partial differential equation for a probability density function (PDF), which can then be deterministically solved in place of a large ensemble of realizations for the Brownian driver. In this case the related probability density function $\psi\left(\mathbf{x}, \boldsymbol{\xi}, t\right)$ satisfies the Fokker-Planck equation [18]:

$$\frac{\partial\psi}{\partial t} + \mathbf{v}\cdot\nabla_x\psi + \nabla_\xi\cdot\left[\left(\boldsymbol{\kappa}\cdot\boldsymbol{\xi} - \frac{2}{\zeta}\mathbf{F}^c\left(\boldsymbol{\xi}\right)\right)\psi\right] = \frac{2k_BT}{\zeta}\nabla_\xi^2\psi; \tag{7}$$

which is also called *Smoluchowski* equation in polymer science. Index ξ on operators indicates that they act in the configuration space. Due to its definition in two spaces and the dimensionality, the solution of Eq. (7) is non-trivial and we proceed as detailed in the next section.

2.3. Solution strategy

In order to solve the Fokker-Planck equation directly, we consider a time-splitting procedure similar to that proposed by Lozinski [8]. Following this idea, the linear operators acting in the configuration space are separated from those acting in the physical space. In this way Eq. (7) can be firstly solved in the only configuration space for an intermediate distribution function $\tilde{\psi}$, which is then used for the solution in the only physical space. We adopt a mixed explicit/implicit framework for the two stages, thus:

$$\frac{\tilde{\psi}^n - \psi^n}{\Delta t} = -\nabla_\xi \cdot \left[\left(\boldsymbol{\kappa} \cdot \boldsymbol{\xi} - \frac{2}{\zeta} \mathbf{F}^c \left(\boldsymbol{\xi} \right) \right) \psi^n \right] + \frac{2 k_B T}{\zeta} \nabla_\xi^2 \psi^n; \tag{8}$$

$$\frac{\psi^{n+1} - \tilde{\psi}^n}{\Delta t} + \mathbf{v} \cdot \nabla_x \psi^{n+1} = 0; \tag{9}$$

thus Eq. (7) reduces to an advection-diffusion equation in the configuration space (8) and an advection equation in physical space (9). Let us now firstly focus on the first stage for Eq. (8): the space scaling is achieved considering a relaxation time $\theta = \zeta / 4h$ and a dimensionless finite extensibility parameter $b = \xi_0^2 h / k_B T$, therefore $\boldsymbol{\xi}$ is made dimensionless with $\sqrt{k_B T / h}$, $\boldsymbol{\kappa}$ with $1/\theta$ and time with θ, thus the resulting dimensionless equation reads:

$$\frac{\tilde{\psi}^n - \psi^n}{\Delta \breve{t}} = -\nabla_{\breve{\xi}} \cdot \left[\left(\breve{\boldsymbol{\kappa}} \cdot \breve{\boldsymbol{\xi}} - \frac{1}{2} H(\breve{\xi}) \breve{\boldsymbol{\xi}} \right) \psi^n \right] + \breve{\alpha} \nabla_{\breve{\xi}}^2 \psi^n; \tag{10}$$

with the diffusion coefficient being $\breve{\alpha} = 1/2$. From now on, the convection vector of ψ in the configuration space (terms in round brackets on right-hand side of Eq. (10)) will be indicated with $\mathbf{u}$ for convenience. The reader should notice that the scaling of the velocity gradient tensor $\boldsymbol{\kappa}$ represents the link between the physical velocity field and the convection vector $\mathbf{u}$ through the relaxation time θ of the polymer. On the basis of this consideration, we define a microscopic (or local) Weissenberg number that will be used later, based on the second invariant of the rate of strain tensor as:

$$Wi_m = \theta \sqrt{\frac{1}{2} \dot{\boldsymbol{\gamma}} : \dot{\boldsymbol{\gamma}}}. \tag{11}$$

Using the same scaling parameters as above, the connector force law $\mathbf{F}^c \left(\boldsymbol{\xi} \right)$ in Eq. (6) is also made dimensionless as:

$$H(\breve{\xi}) = \frac{1}{1 - \left|\left|\breve{\boldsymbol{\xi}}\right|\right|^2 / b}. \tag{12}$$

Equation (10) with the connector force law (12) is therefore the final dimensionless equation to be solved in the configuration space. In this work we assume the dumbbells to be always laying in the same plane, therefore the configuration space is two-dimensional and the dumbbell extensibility domain (support of the PDF) results in a disc of radius $\sqrt{b}$. In this first stage, Eq. (10) is solved for an equilibrium solution of $\tilde{\psi}^n$ for the local convection vector $\mathbf{u}$. The details on the numerical method together with its optimization will be extensively discussed later.
At this point, the obtained intermediate $\tilde{\psi}^n$ should be convected in physical space by Eq. (9) according to the second stage of the time-splitting procedure. However, we note that the convection of the full PDF in an Eulerian framework would require a prohibitively amount of data to be retained and transported. The problem can be sensibly reduced considering that the final target for the hydrodynamic system is the viscoelastic stress tensor. Therefore we proceed by computing an intermediate stress tensor, which is convected in physical space in place of the PDF. Indicating then with $\langle\langle\cdot\rangle\rangle$ the ensemble averaging operator, the intermediate dimensionless viscoelastic stress tensor $\tilde{\breve{\boldsymbol{\sigma}}}_p^n$ is calculated from $\tilde{\psi}^n$ using the Kramers expression [3]:

$$\tilde{\breve{\boldsymbol{\sigma}}}_p^n = \langle\langle H(\breve{\xi})\breve{\boldsymbol{\xi}} \otimes \breve{\boldsymbol{\xi}}\rangle\rangle - \mathbf{I} = \int_{||\breve{\boldsymbol{\xi}}||^2 < b} \tilde{\psi}^n (H(\breve{\xi})\breve{\boldsymbol{\xi}} \otimes \breve{\boldsymbol{\xi}}) d\breve{\boldsymbol{\xi}} - \mathbf{I}; \tag{13}$$

and convected according to Eq. (9) in place of the distribution function:

$$\frac{\breve{\boldsymbol{\sigma}}_p^{n+1} - \tilde{\breve{\boldsymbol{\sigma}}}_p^n}{\Delta t} + \mathbf{v} \cdot \nabla \tilde{\breve{\boldsymbol{\sigma}}}_p^{n+1} = 0. \tag{14}$$

In the iterative solution adopted in this work, this procedure is formally equivalent to the convection of the PDF before computing stresses (the numerical procedure will be discussed in the next section). Furthermore, the conservation of stresses is analogous to the conservation of the second order moment of the distribution, which is actually the only needed information for the solution. The advantage of this approach is that the second stage for the solution of the Fokker-Planck equation (9), reduces to the convective

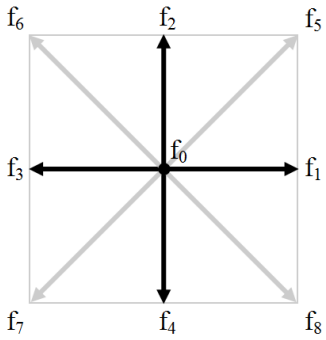


Figure 1: (Color online) Lattice stencils and relative discrete distribution functions: five links for D2Q5 (black color) and four additional links for D2Q9 (gray diagonals).

transport of three scalar quantities, one for each component of the symmetric stress tensor. In this case the choice of the time step Δt has no theoretical justification and is chosen to achieve convergence of the global solution.
Finally, the dimensionless stress tensor is scaled-up to its corresponding in the physical space, to serve as volumetric source term in the momentum equation (2). Indicating with n_c the number of polymer chains per unit volume, an equivalent polymer viscosity can be defined as $\mu_p = \theta n_c k_B T$ and the extra stress is scaled as [19]:

$$\boldsymbol{\sigma}_p = \frac{\mu_P}{\theta} \breve{\boldsymbol{\sigma}}_p^{n+1}. \tag{15}$$

3. Numerical methods

3.1. Finite Volume Method

The macroscopic governing equations (1) and (2) and the transport of stresses (14) are solved by finite volume method (FVM). In this approach, transport equations are numerically solved on a discretized computational domain (mesh) and the conserved variables are calculated at cell centers. Partial differential equations are therefore converted to algebraic equations by integration about the cells (or control volumes), for example Eq. (2):

$$\int_{V_c} \rho \frac{\partial \mathbf{v}}{\partial t} dV + \oint \rho \nabla_x \cdot (\mathbf{v} \otimes \mathbf{v})\, d\mathbf{A} = \oint (-\nabla_x p + \nabla_x \cdot \sigma)\, d\mathbf{A}. \tag{16}$$

Equation (16) is then applied to each control volume and its neighboring cells in the domain, resulting in a system of algebraic equations with sparse coefficient matrix to be solved. Fluxes at cell faces, which are required for convective terms, can then be interpolated using several different numerical schemes: in this paper we adopt a third order quadratic upwind scheme (QUICK) for momentum (2) and transport of stresses (14) and a second order scheme for pressure interpolation. For the sake of clarity, we remark that despite the hyperbolic nature of Eq. (14), the solution is sufficiently smooth to be solved with a third order scheme for highly-convective transport. The interested reader can refer for example to [20] for details on the methods.

3.2. Lattice Boltzmann Method

The advection-diffusion equation for the FENE model (10) is solved by lattice Boltzmann method (LBM). This mesoscopic approach relies on the Boltzmann transport equation, whose discrete form in the Bhatnagar-Gross-Krook (BGK) approximation of the collision operator, reads as [21]:

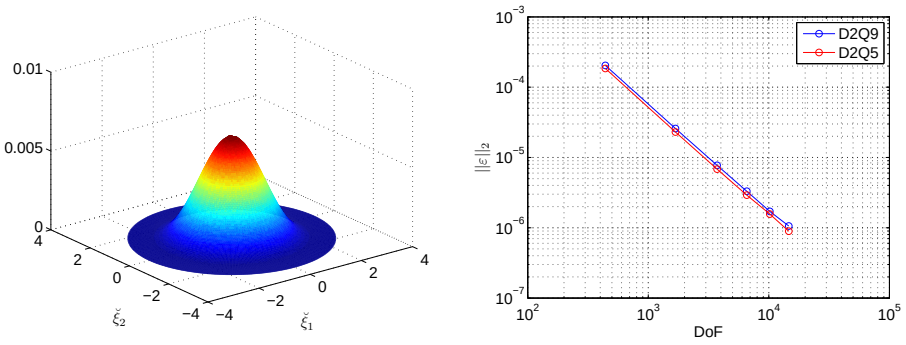
$$f_i\left(\breve{\boldsymbol{\xi}}+\mathbf{c}_i\delta\breve{t},\breve{t}+\delta\breve{t}\right)-f_i\left(\breve{\boldsymbol{\xi}},\breve{t}\right)=-\frac{1}{\tau}\left(f_i\left(\breve{\boldsymbol{\xi}},\breve{t}\right)-f_i^{eq}\left(\breve{\boldsymbol{\xi}},\breve{t}\right)\right); \tag{17}$$

with $\delta\breve{t}$ being the time step, f_i the discrete particle distribution functions and $\mathbf{c}_i$ the associated microscopic velocity vectors. The equilibrium distribution function f_i^{eq} can be derived, for example, via second-order Taylor expansion in the Mach number of the Maxwell-Boltzmann equilibrium [22]:

$$f_i^{eq}=\left(1+\frac{\mathbf{c}_i\mathbf{u}}{c_s^2}+\frac{\left(\mathbf{c}_i\mathbf{u}\right)^2}{2c_s^4}-\frac{||\mathbf{u}||^2}{2c_s^2}\right)\omega_i\psi; \tag{18}$$

where c_s is the lattice speed of sound that, indicating with $\delta\breve{\xi}$ the lattice spacing and thus $c=\delta\breve{\xi}/\delta\breve{t}$ the lattice speed, is defined as $c_s=c/\sqrt{3}$. The reader should notice that in this case we retain the tilde notation for space and time for analogy with the equation being solved (10), but rigorously we should consider dimensionless lattice units. Macroscopic quantities can be recovered from the moments of the distribution function:

$$\psi=\sum_i f_i=\sum_i f_i^{eq}; \tag{19}$$



(a) (b)

Figure 2: (Color online) Shaded surface (in polar coordinates) of the analytical equilibrium PDF (Eq. 23) on a 1,681 DoF lattice (a) and ℓ_2-*norm* convergence of ψ with $\tau = 0.55$ (b).

$$\psi \mathbf{u} = \sum_i \mathbf{c}_i f_i^{eq}; \tag{20}$$

$$\psi \left(\mathbf{uu} + c_s^2 \mathbf{I} \right) = \sum_i \mathbf{c}_i \mathbf{c}_i f_i^{eq}; \tag{21}$$

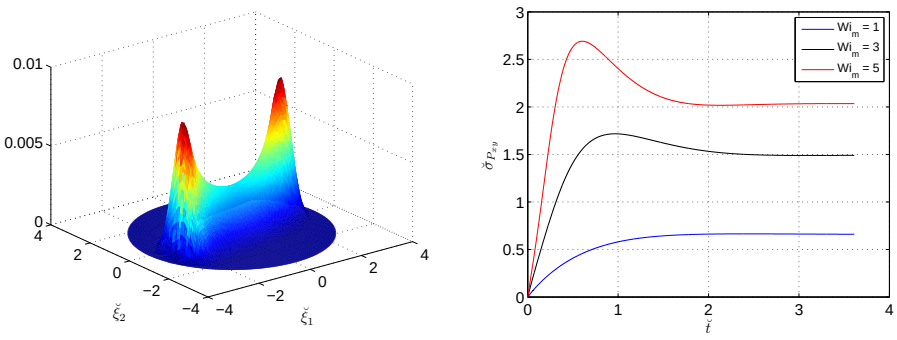
which also allow to recover the macroscopic equation (10) by multi-scale expansion and thus the following expression for the relaxation time (see Appendix A for details):

$$\tau = \frac{\breve{\alpha}}{\delta \breve{t} {c_s}^2} + \frac{1}{2}. \tag{22}$$

Given the advective-diffusive nature of Eq. (10), the numerical solution can be carried out on two lattice topologies, D2Q9 and D2Q5 (Fig. 1). The lattice constants for both stencils can be found in Appendix B. The domain length l is imposed to be 20 percent larger then the domain of existence of the PDF, therefore indicating with N the number of nodes, the lattice spacing $\delta\breve{\xi}$ is given by l/N.

3.3. Coupled numerical algorithm

The numerical solution of the coupled model has been carried out using the commercial CFD code ANSYS Fluent v14.0. The lattice Boltzmann solution is called at cell centers as a sub-grid routine via compiled-C user



(a) (b)

Figure 3: (Color online) Start-up plane Couette flow: shaded surface (in polar coordinates) of the equilibrium PDF for $Wi_m = 5$ on a 1,681 DoF lattice (a) and dimensionless shear stress evolution $\check{\sigma}_{P_{xy}}$ for different Wi_m on a D2Q9 lattice with 3,721 DoF and $\tau = 0.55$ (b).

defined function (UDF). The numerical procedure can be summarized as follows:

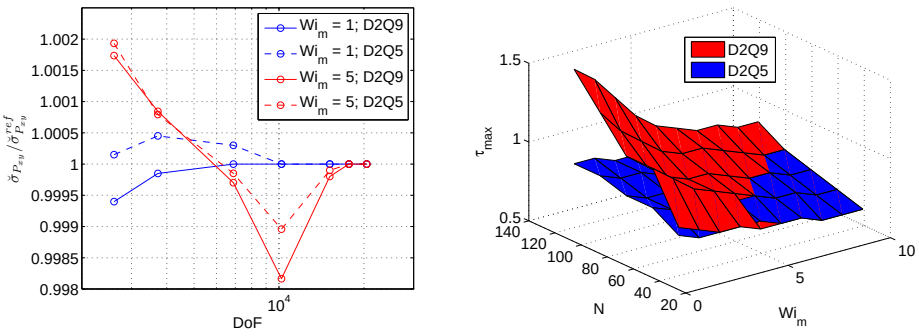
1. solution of the governing equations by finite volume method: Eq. (1) and (2);
2. sub-grid lattice Boltzmann solution of the FENE kinetic equation: Eq. (10);
3. computation of the local viscoelastic stress tensor: Eq. (13);
4. convective transport of the viscoelastic stresses: Eq. (14);
5. addition of the extra-stress to the momentum equation: (Eq. 2).

The procedure is iteratively repeated until global convergence. The convergence criterion for the FVM iterations is a 10^{-8} residual while for global convergence is 10^{-4}. For numerical stability, a segregated algorithm (SIMPLE) is used for the pressure-velocity coupling and the sub-grid solution is computed at the end of the FVM iteration.

4. Results and discussion

4.1. Sub-grid solution analysis

In this section the tilde notation and superscripts introduced in section 2.3 are omitted for readability, but they should be kept in mind in the following



(a) (b)

Figure 4: (Color online) Error convergence for $Wi_m = 1$ and 5 on the two lattices (a) and stability map for D2Q9 and D2Q5 lattices (b).

analysis. The relaxation of the probability density $\psi(\breve{\boldsymbol{\xi}}, \breve{t})$ to equilibrium is tested considering that for null velocity gradient $\breve{\boldsymbol{\kappa}}$, an analytical solution for Eq. (10) can be found in the form [12]:

$$\psi_{eq} = \frac{H(\breve{\xi})^{-b/2}}{\int H(\breve{\xi})^{-b/2} d\breve{\boldsymbol{\xi}}}; \tag{23}$$

which for a dimensionless dumbbell extensibility $b = 10$ (constant in the paper), yields the equilibrium distribution shown in Fig. 2(a). Given then an initial distribution function ψ_0 (constant in this case), satisfying the normality condition $\int \psi(\breve{\boldsymbol{\xi}}) d\breve{\boldsymbol{\xi}} = 1$, the relaxation rate and error convergence are analyzed by an ℓ_2-*norm* with respect to the reference solution defined as:

$$||\varepsilon||_2 = \frac{1}{N} \sum_{k=1}^{N} \sqrt{\psi_{eq}^2 - \psi_t^2}; \tag{24}$$

being ψ_t the distribution function at time t. The convergence criterion for relaxation is a 10^{-8} residual calculated as backward finite difference on the norm. The analysis for the two lattices shows that the error of the 8-neighbors lattice is slightly larger than that of the 4-neighbors one (see Fig. 2(b)).
Passing now to the analysis of non-null $\breve{\boldsymbol{\kappa}}$ gradient, we examine the time evolution of the shear stress $\breve{\sigma}_{p_{xy}}$ for a start-up planar Couette flow $[0, \breve{\kappa}_{xy}; 0, 0]$. The initial distribution function is in this case (as in the rest of the paper)

given as Eq. (23). According to its definition (11), in this case the local Weissenberg number corresponds to the magnitude of the component $\breve{\kappa}_{xy}$ itself. The obtained equilibrium PDF for $Wi_m = 5$ is shown in Fig. 3(a), while the stress evolution for varying Wi_m in Fig. 3(b). In this case the relaxation to equilibrium is checked by the same residual criterion as above, but applied on the shear stress. The results obtained are consistent with concerning literature (see for example the results obtained by Leonenko and Phillips [23] using a reduced basis approximation). The error analysis has been carried out for $Wi_m = 1$ and $Wi_m = 5$ and is shown in Fig. 4(a). To allow a proper visualization of the comparison, the shear stress has been normalized using the value obtained with the highest number of nodes $\breve{\sigma}_{p_{xy}}/\breve{\sigma}^{ref}_{p_{xy}}$ for each case. Notice as for a higher Wi_m the solution requires a higher number of nodes to converge, in particular for the D2Q9 lattice. This behavior can be associated with the shape of the equilibrium distribution function, that for lower Wi_m is closer to the initial condition (Fig. 2(a)) than for a higher Wi_m (Fig. 3(a)).

Table 1: Comparison of the computational time [s] and relative numerical error [%] for the two lattices for $\tau = 0.55$ and $\tau = \tau_{max}$ (start-up plane Couette flow at $Wi_m = 5$).

stencil	**DoF**	**1,681**	**3,721**	**6,561**	**10,201**	**14,641**
D2Q9	$\tau = 0.55$	0.98	4.88	15.24	36.86	80.54
	$\tau = \tau_{max}$	0.33	0.87	2.48	4.63	8.43
	speed-up	3.0	5.6	6.1	8.0	9.6
	error	0.5357	-0.1933	-0.1538	-0.1690	-0.1092
D2Q5	$\tau = 0.55$	0.62	3.11	6.45	9.93	14.24
	$\tau = \tau_{max}$	0.2	0.63	1.93	3.92	7.03
	speed-up	3.1	4.9	3.4	2.5	2.0
	error	0.0764	-0.0594	-0.0297	-0.0347	-0.0198

An analysis of the stability domains for the two tested lattices has been also carried out for Wi_m in the range 1 to 10. The results show that the stability range of the D2Q9 is larger than that of the D2Q5 lattice in the region of low Wi_m and high DoF (Fig. 4(b)). Despite the increased stability, the error of the 8-neighbors lattice is also slightly larger than that of the 4-neighbors one (Fig. 4(a)).

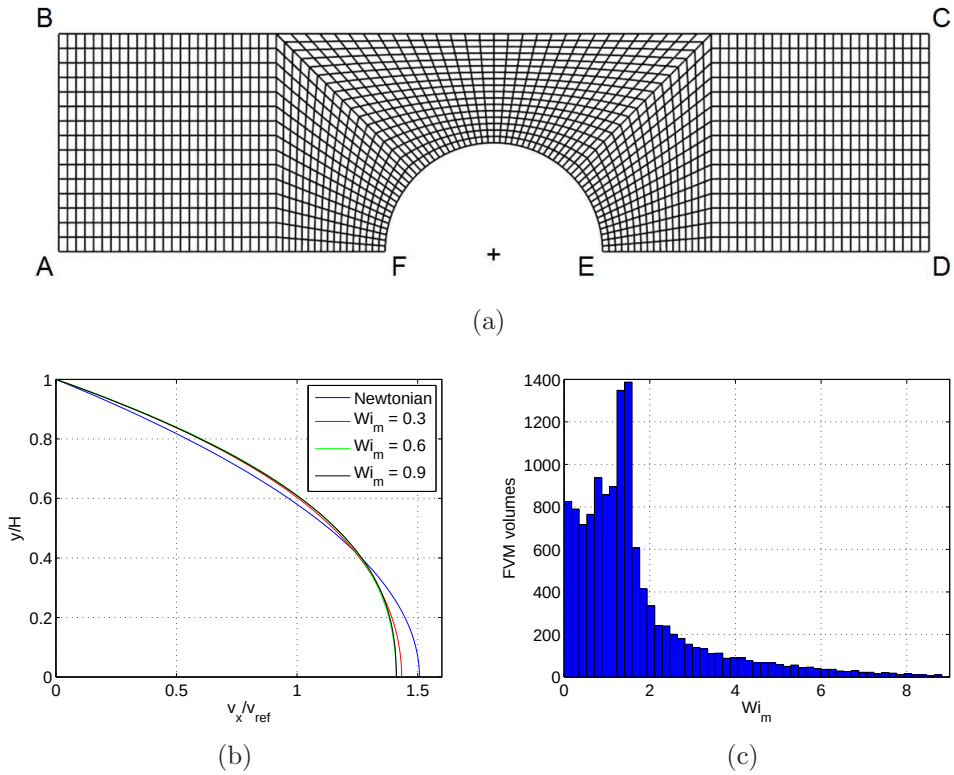


Figure 5: (Color online) (a) Mesh layout for the computational domain (1,770 cells displayed) and (b) discrete distribution of Wi_m in the physical domain (mesh M1, number of bins 50).

Table 1 shows the comparison of the computational time required by the two lattices to converge to equilibrium for the start-up plane Couette flow at $Wi_m = 5$, using the minimum relaxation time $\tau = 0.55$ and the maximum stable allowed on the basis of of the stability map. The tested CPU is an Intel Xeon X5650 2.67GHz. The D2Q5 lattice requires less computational time due to the reduced number of links and therefore of computational operations, however the speed-up for the D2Q9, when moving from τ_{min} to τ_{max}, is greater due to the larger stability range. The relative numerical error introduced increasing the relaxation time for the two lattices is anyway always lower than 1% and the maximum speed-up achievable is nearly ten times for the D2Q9 lattice.

4.2. Coupled model

In order to validate our model, we decided to adopt a commonly used benchmark problem: a two-dimensional viscoelastic flow around a cylinder confined between two parallel plates [1]. The computational domain is shown in Fig. 5(a), where only 1,770 cells are displayed to allow a proper visualization of the mesh layout. In order to save in computational time, only half of the domain is studied and symmetry conditions are applied on the lower boundaries. The domain extent is 4 m length (L), 0.5 m height (H) and the hole is 0.25 radius (R) centered in the origin. The ratio of the radius of the cylinder to the half-width of the channel has been chosen to be $\Lambda = 0.5$ and the ratio of the solvent to the total viscosity is $\beta = \mu_s/(\mu_s + \mu_p) = 0.59$ [1, 9]. The other boundary conditions are: stream-wise periodicity between inlet (AB) and outflow (CD); no-slip for momentum and homogeneous Neumann for convection of stresses on the hole and upper boundary (BC). Indicating with $\langle\cdot\rangle$ the volume-averaging operator, we define for this problem a macroscopic (or global) Weissenberg and Reynolds number based on the average velocity at inlet (or outlet):

$$Wi_M = \frac{\langle \mathbf{v} \rangle}{R}\theta = \langle \dot{\boldsymbol{\gamma}} \rangle \, \theta; \qquad Re_M = \frac{\rho \, \langle \mathbf{v} \rangle \, R}{\mu}. \tag{25}$$

The Reynolds number is kept constant to 10^{-3} (creeping flow) in all cases. In order to test the FVM mesh independence, the solution has been carried out on two different grids, respectively of 12,500 (M1) and 17,500 (M2). The number of nodes and relaxation time for the sub-grid solution are $N = 81$ and $\tau = 0.55$ (D2Q9). The profiles of dimensionless viscoelastic stresses on the symmetry plane and on the cylinder surface for $Wi_M = 0.6$ are consistent with those obtained by Chauvière and Lozinski [9] with a Galerkin spectral element method (Fig. 7). The dimensionless dumbbell elongations in the domain for $Wi_M = 0.9$ (Fig. 7), are also qualitatively consistent with the spectral method for the 3D case (see [9] for the comparison between 2D and 3D solution). As further validation we also compare the drag factor, which is defined as follows:

$$C_D = \frac{F_x}{4\pi R(\mu_s + \mu_p) \, \langle \mathbf{v} \rangle}; \tag{26}$$

where F_x is the drag force on the cylinder surface (with polar angle ϑ), including pressure, viscous and viscoelastic contributions:

$$F_x = 2\int_0^{\pi}\left[\left(-p + 2\mu_s\frac{\partial v_x}{\partial x} + \sigma_{pxx}\right)\cos\vartheta + \left(\mu_s\left(\frac{\partial v_y}{\partial x} + \frac{\partial v_x}{\partial y}\right) + \sigma_{pxy}\right)\sin\vartheta\right]Rd\vartheta. \tag{27}$$

The obtained values of the drag factor for different Weissenberg numbers are reported in table 2. A final assessment of the numerical solution is given by comparison of the velocity profiles obtained in Fig. (5(b)).
The sub-grid solution has been distributed on 8 cores Intel Xeon X5650 2.67GHz, reducing 8 times the computational time which results in 6.5h/iteration for mesh M1. An important advantage of this approach indeed, is that it is linearly scalable on computing nodes (i.e. clusters), hence this time can be reduced according to the computational power available.

Table 2: Comparison of the calculated drag factor with the results obtained by Chauvière and Lozinski [9] for different Weissenberg number.

Wi	0.6	0.9	1.1
C_D			
C_D [9]	8.8925	8.5521	8.4056

Let us now focus on a more detailed analysis of the case: solving the macroscopic fields for $Wi_M = 0$, yields the relative physical velocity gradient $\nabla_x \mathbf{v}$ that can be used to extrapolate the corresponding Wi_m (Eq. 11) in the domain for a given Wi_M. For the sake of clarity, we remark that strictly this procedure is only valid for a preliminary analysis, as long as the streamlines modifies with the converging solution. However, in the tested range of Wi_M $(0.1 \div 0.9)$ for Stokes flow, the velocity gradient modifications are small [24] and it provides a good approximation.
The discrete distribution of local Weissenberg number for $Wi_M = 0.6$ on mesh M1 is shown in Fig. 5(c). For this case, the local Weissenberg ranges between 0 and 9, with the highest frequencies between 0 and 2 and tail between 2 and 9. The parameters for the sub-grid solution (N and τ) can then be chosen according to the following criteria:

1. Minimize numerical error: high lattice resolution and minimum relaxation time are to be used. This approach assures converged solution throughout

the domain but non-homogeneous numerical error. The computational cost is high due to excessive number of nodes in $low - Wi_m$ regions.

2. Maximize computational speed: coarse lattice resolution and maximum stable relaxation time. The lattice parameters are chosen according to the maximum value of Wi_m in the domain, that is the coarsest allowed lattice and the maximum stable relaxation time. This approach does not guarantee constant nor converged numerical error. The maximum error depends on the choice of the lattice size (Fig. 4(a)).

3. Locally-adaptive: lattice resolution based on local Wi_m. The lattice parameters are dynamically adapted according to the local Wi_m (Fig. 5(c)). Therefore, coarser lattices are used in $low - Wi_m$ regions and finer lattices in $high - Wi_m$ ones. This approach represents a trade-off between the two above discussed ones and allows to optimize the computational speed, providing control on the error. The number of different lattices to use can be chosen on the basis of an expanded analysis such as that in Fig. 4(a), according to the desirable degree of speed-up/error control.

An overview of the three approaches is reported in Table 3. The parameters for the comparison of the achievable speed-up are: $N_{min} = 41$, $N_{max} = 121$, $\tau_{min} = 0.55$ and τ_{max} the maximum local stable value for the locally-adaptive approach (Fig. 4(b)) and the maximum stable value for $Wi_m = 9$ for optimizing the computational speed (0.6 for D2Q9 and 0.55 for D2Q5). In order to compare the advantage of the locally-adaptive approach here we use two lattice sizes, namely $N = 81$ for $Wi_m = 1 \div 5$ and $N = 121$ for $Wi_m = 5 \div 9$. We remark that this choice is made to illustrate the methodology but the number of lattice resolutions is arbitrary.

5. GPU acceleration and coupling

In this section we present and discuss the methodology that progressively lead us to the faster implementation for Graphic Processing Unit (GPU). Going into the details of coding goes beyond the purpose of the present work, therefore we provide a methodological description for each strategy. A slightly more detailed description is given for the faster implementation achieved. The available GPU is an NVIDIA Quadro 600 1GB DRAM

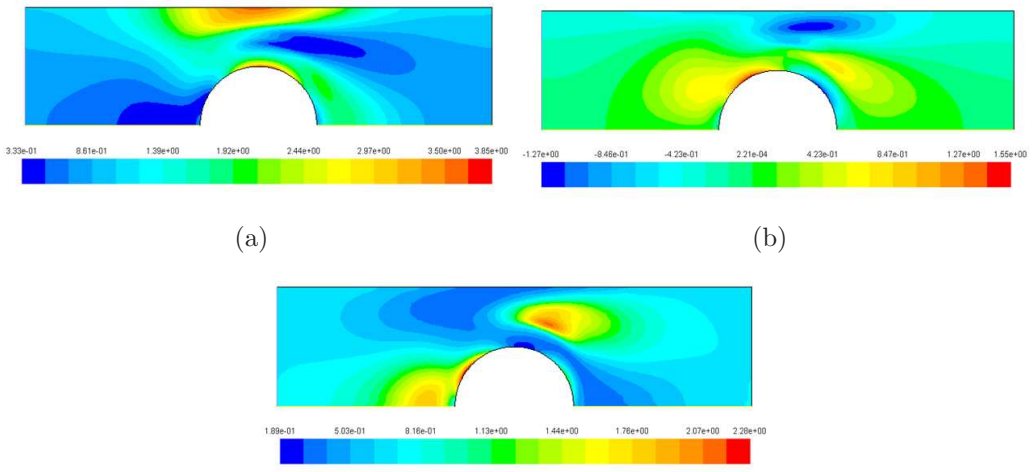


Figure 6: (Color online) Contours of dimensionless molecular elongations (configuration tensor) for $Wi_M = 0.6$: (a) $\langle\langle\breve{\xi}_x\breve{\xi}_x\rangle\rangle$ (b) $\langle\langle\breve{\xi}_x\breve{\xi}_y\rangle\rangle$ (c) $\langle\langle\breve{\xi}_y\breve{\xi}_y\rangle\rangle$

DDR3 96 cores and all the tests are performed in single-precision floating point operations.

5.1. Sailfish implementation

Sailfish is an open-source code for computational fluid dynamics based on lattice Boltzmann method and optimized for NVIDIA graphic cards [25]. The structure of the code is non-trivial as it makes use of different scripting languages. The highest level code is Python, where the simulation parameters are set-up (LB-mesh size, methods, initial and boundary conditions). The code generation passes then through *Mako* templates, which generate CUDA C or OpenCL optimized code. The generated code is then compiled on-the-fly and the resulting binary is run on graphic card. The code in its current version (0.3) is primarily designed to solve Navier-Stokes equations, however small changes in the lowest level kernels allow us to tailor it for our purposes, namely to solve an advection-diffusion equation. The speed-up achieved with respect to CPU reaches nearly 60x (see Fig. 8(a)).
The coupling with Fluent is then realized by means of serial dynamical calls to *Sailfish* from compiled-C user defined function. Despite the simplicity of implementation, this approach has proven not to be computationally

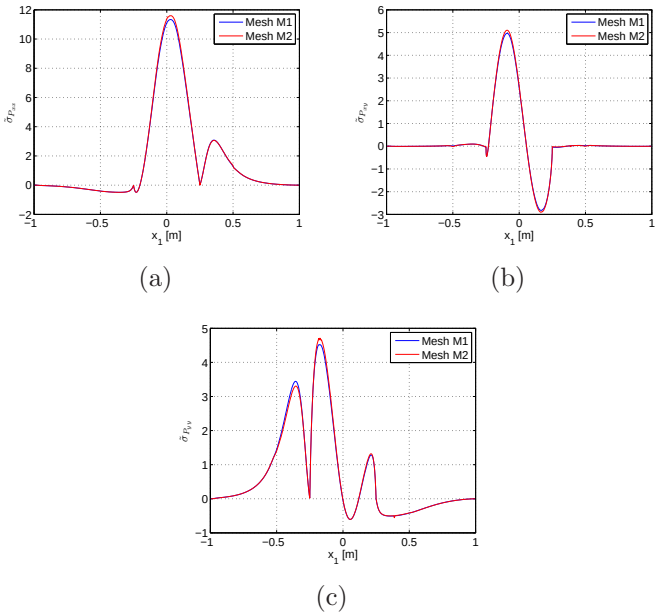


Figure 7: (Color online) Profiles of the dimensionless viscoelastic stresses on the symmetries and on the cylinder for $Wi_M = 0.6$. Results for the two tested FVM meshes: (a) $\breve{\sigma}_{P_{xx}}$ (b) $\breve{\sigma}_{P_{xy}}$ (c) $\breve{\sigma}_{P_{yy}}$.

efficient due to the time required by *Sailfish* for the GPU code generation and compilation for each call. The total time (or *wall time*) for a single call is indeed significantly higher than the effective computational time on the GPU cores (*kernel time*). A profiling of timing and GPU utilization is reported in Fig. (8(b)). We conclude that *Sailfish* in its current version cannot be straightforwardly exploited for our purposes and an *ad-hoc* implementation for GPU has to be developed.

5.2. CUDA implementation 1: texture memory

The Compute Unified Device Architecture (CUDA) is a parallel computing platform and coding environment developed by NVIDIA [26], which enables to exploit the power of graphic processing units for scientific applications (GPGPU). A GPU-oriented implementation for CUDA-enabled cards can be all the way written in C or C++, taking advantage of the language extensions (API) provided (see the CUDA Programming Guide [27]).
The general layout of a code, comprises a *host function* running on CPU and

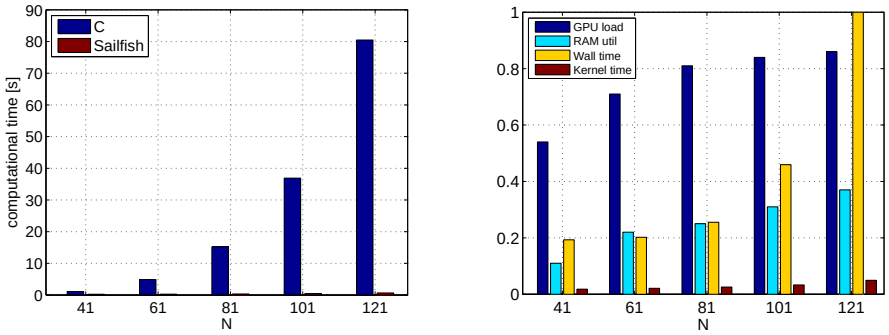
Table 3: Summary table of the three sub-grid solution strategies: ME (minimize error), SA (strain-adaptive) and MS (maximize speed). The comparison of the computational speed-up per FVM iteration refers to different approaches on the same stencil (results for mesh M1).

stencil	approach	N	τ	error	speed-up
D2Q9	(ME) *minimize error*	N_{max}	τ_{min}	variable	13.9 (ME/SA) 17.6 (SA/MS)
	(SA) *strain-adaptive*	$f(Wi_m)$	τ_{max}	controlled	
	(MS) *maximize speed*	N_{min}	τ_{max}	variable	
D2Q5	(ME) *minimize error*	N_{max}	τ_{min}	variable	3.1 (ME/SA) 23.3 (SA/MS)
	(SA) *strain-adaptive*	$f(Wi_m)$	τ_{max}	controlled	
	(MS) *maximize speed*	N_{min}	τ_{max}	variable	

kernel functions running on the GPU. The kernels therefore take advantage of the parallelization on the graphic card. Due to the large computing power, the bottleneck resides in the memory read/write operations. As a general rule, a proper implementation should then reduce memory accesses as much as possible and/or use faster memories available on the graphic card. In this implementation for example we use *texture memory*, which is an read-only memory that is spatially cached, making faster the data retrieval in the device memory [27].
The implementation relies on two kernels, one for the collision step and one for the streaming. The LB-mesh is hierarchically divided into grid, blocks and threads. Typically each node is assigned a thread, which is then handled by a processor. Data is allocated and passed from CPU to GPU and vice-versa, respectively at the beginning and at the end of the solution process. The data exchange between the two kernels is optimized using textures, however this data passage represent a bottleneck in the code. By means of this approach indeed the maximum speed-up achieved is around 25x with respect to the CPU.

5.3. CUDA implementation 2: shared memory

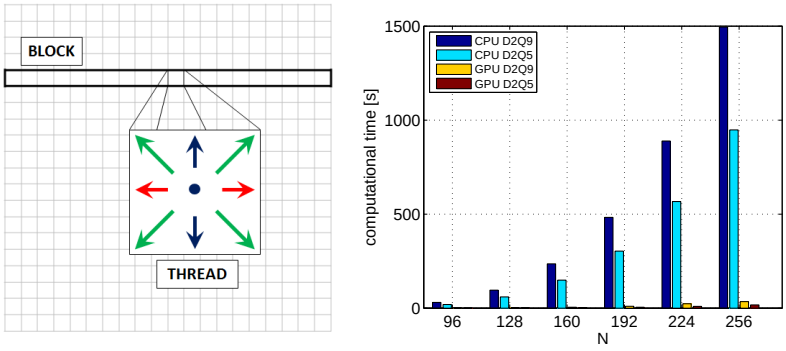
In order to overcome the limit of the previous code, in this version we make use of *shared memory*, an extremely fast on-chip memory. The main issues to take into account for the implementation of a lattice Boltzmann method



(a) (b)

Figure 8: (Color online) Comparison of the computational time for compiled C and SAILFISH (D2Q9 lattice with $\tau = 0.55$) on the available hardware (a) and GPU profiling for the coupling (normalized quantities).

is that it is of limited size (for our card is 48 KB) and that data is shared only between threads belonging to the same block. A proper implementation of a lattice Boltzmann method using shared memory was originally proposed by Tölke [28]. We also refer ther reader to compare [29] for more details on hardware and code excerpts. Here we propose a similar strategy, but tailored for small LB-meshes (our case). The code relies on a single kernel for collision and propagation and the mesh is processed by rows, therefore each block corresponds to one mesh row and the number of threads to the mesh width (Fig. 9(a)). In this way the size of the data loaded into shared memory per block is limited to that required to process one row. Collision and propagation are then performed on shared memory, where the horizontal propagation of the distributions is straightforward within the block, while the vertical one is achieved with a correct alignment between global and shared memory. The coalescence of global memory accesses and the lack of bank conflicts on shared memory has been checked using the CUDA visual profiler [30] (we do not go into the details of these issues, the interested reader can refer to the CUDA Programming Guide [27]). It is important to remark that this implementation is tailored for small meshes and does not apply in other cases. Fixing the block size equal to the mesh width indeed, sets a constraint on the choice of the number of threads (which plays a key-role for performance). This approach represent a tentative to properly exploit



(a) (b)

Figure 9: (Color online) Scheme of the CUDA implementation using shared memory (a) and comparison of the computational time for compiled C and the CUDA implementation using shared memory (D2Q9 and D2Q5 lattices with $\tau = 0.55$) (b).

the GPU for the resolution of small meshes, which is not normally done but relevant for our application.
The developed code is finally compiled with the CUDA compiler (nvcc) and dynamically called from a Fluent user defined function. The sub-grid simulation is driven by passage and retrieval of the required variables between the two compiled codes through a stream process.
With this implementation the speed-up reaches nearly 60x with respect to the CPU, as shown in Fig. 9(b) and Table 4). The computational time per VFM iteration for the D2Q5 lattice with $N = 128$ mesh and $\tau = 0.55$ for example, results in 4,5h on a single GPU. Local-adaption and utilization of multiple GPUs still hold, thus providing the possibility of customization or acceleration based on the methodology presented in Section 4.

6. Conclusions

In this work, a novel micro-macro model for dilute polymeric solutions in Stokes flow has been presented. The proposed approach rely on a coupled numerical solution for the macro and microscopic scales: a finite-volume method for the fluid-flow equations and a lattice-Boltzmann method for the kinetic viscoelastic model. The validity of the introduced model has been proven against a complex benchmark problem of two-dimensional flow past a confined cylinder. Various optimization strategies have been also discussed

Table 4: Comparison of the computational time for C and CUDA (shared memory implementation).

N	**96**	**128**	**160**	**192**	**224**	**256**
D2Q9 CPU	30.4	95.5	235.8	483.0	889.0	1495.0
D2Q9 GPU	0.8	2.0	4.7	10.1	23.3	34.7
speed-up	38	48	50	48	38	43
D2Q5 CPU	19.0	60.0	149.0	303.5	567	948
D2Q5 GPU	0.4	1.3	2.5	5.0	9.6	7.1
speed-up	48	46	60	60	60	55

on the basis of a systematical analysis of the test case and of the numerical methods. Advantages of the proposed approach are: (i) control of the trade-off between numerical accuracy and computational cost; (ii) linear scalability on distributed computing units.
Further optimization has been achieved with a proper implementation of the sub-grid solution for graphic cards and an efficient coupling strategy. It has been shown that the speed-up reaches nearly 60x with respect to a high-level CPU. In this work we used a single graphic card, but the solution can still be distributed on multiple units, thus further reducing the computational time.
For the sake of completeness we remark that in this work we proposed the coupling with a finite volume method solver, but the accelerated sub-grid solution can be easily called from other solvers (i.e. FEM-LBM or LBM-LBM solutions).
Finally, the results obtained suggest that a direct numerical method together with proper hardware implementation, may deserve attention in the framework of numerical methods for complex fluids.

Appendix A. Asymptotic analysis

In this appendix we briefly report the procedure to recover the FENE equivalent Fokker-Planck equation (10) from the lattice-BGK equation (17) via asymptotic expansion (Chapman-Enskog procedure) [12]. Tilde notation is omitted for readability.
Let us consider the 2-nd order Taylor expansion of the post-collision term (first term on the left-hand side) in Eq. (17):

$$\Omega_i(f) \approx (\partial_t + \nabla \cdot \mathbf{c}_i) f_i + \frac{1}{2}\left(\partial_t^2 + 2\partial_t \nabla \cdot \mathbf{c}_i + \nabla\nabla : \mathbf{c}_i\mathbf{c}_i\right) f_i; \tag{A.1}$$

and the following expansions of the time derivative ∂_t and distribution function f_i in terms of a small formal number ϵ (spatial derivative is not expanded):

$$\partial_t = \epsilon\partial_{t1} + \epsilon^2\partial_{t2} + \mathcal{O}\left(\epsilon^3\right); \tag{A.2}$$

$$f_i = {f_i}^{eq} + \epsilon {f_i}^{(1)} + \epsilon^2 {f_i}^{(2)} + \mathcal{O}\left(\epsilon^3\right). \tag{A.3}$$

Applying (A.2) and (A.3) in eq. (A.1) yields the scale-separated form (A.4) and (A.5). Combining to get rid of higher order derivatives yields (A.6).

$$(\partial_{t1} + \nabla \cdot \mathbf{c}_i) {f_i}^{eq} = -\frac{1}{\tau\delta t} {f_i}^{(1)}; \tag{A.4}$$

$$\partial_{t2} {f_i}^{eq} + (\partial_{t1} + \nabla \cdot \mathbf{c}_i) {f_i}^{(1)} + \frac{\delta t}{2} (\partial_{t1} + \nabla \cdot \mathbf{c}_i)^2 {f_i}^{eq} = -\frac{1}{\tau\delta t} {f_i}^{(2)}; \tag{A.5}$$

$$\partial_{t2} {f_i}^{eq} + \left(1 - \frac{1}{2\tau}\right) (\partial_{t1} + \nabla \cdot \mathbf{c}_i) {f_i}^{(1)} = -\frac{1}{\tau\delta t} {f_i}^{(2)}. \tag{A.6}$$

Using now the 0-th order moment (19) and the condition (A.7) on the non-equilibrium distribution functions, yields (A.8) and (A.9):

$$\sum_i {f_i}^{noneq} = \sum_i {f_i}^{(1,2)} = 0; \tag{A.7}$$

$$\partial_{t1}\psi + \nabla \cdot (\psi \mathbf{u}) = 0; \tag{A.8}$$

$$\partial_{t2}\psi + \left(1 - \frac{1}{2\tau}\right) \nabla \cdot \sum_i \mathbf{c}_i {f_i}^{(1)} = 0; \tag{A.9}$$

Recovering ${f_i}^{(1)}$ from eq. (A.4), the sum in (A.9) becomes (A.10), rearranging (A.11):

$$\sum_i \mathbf{c}_i {f_i}^{(1)} = -\tau\delta t \sum_i \mathbf{c}_i \left(\partial_{t1} + \nabla\cdot\mathbf{c}_i\right) {f_i}^{eq} = -\tau\delta t \left(\partial_{t1}\left(\psi\mathbf{u}\right) + \nabla\cdot\left[\psi\left(\mathbf{uu} + c_s^2\mathbf{I}\right)\right]\right); \tag{A.10}$$

$$\sum_i \mathbf{c}_i {f_i}^{(1)} = -\tau\delta t \left(\mathbf{u}\left(\partial_{t1}\psi + \nabla\cdot\left(\psi\mathbf{u}\right)\right) + {c_s}^2\nabla\psi\right). \tag{A.11}$$

Finally using (A.11) into (A.9) and reassembling scales, yields the final macroscopic equation:

$$\frac{\partial\psi}{\partial t} = -\nabla_\xi\cdot\left(\mathbf{u}\psi\right) + \delta t\left(\tau - \frac{1}{2}\right){c_s}^2\nabla_\xi^2\psi; \tag{A.12}$$

that from the comparison with (7), gives the following expression for the lattice relaxation time:

$$\tau = \frac{\alpha}{\delta t {c_s}^2} + \frac{1}{2}. \tag{A.13}$$

Appendix B. Lattice constants

The discrete velocities $\mathbf{c}_i$ and weights ω_i for the D2Q9 lattice are:

$$\mathbf{c}_i = \begin{cases} (0,0) & i = 0 \\ (\pm c, 0), (0, \pm c) & i = 1,2,3,4 \\ (\pm c, \pm c) & i = 5,6,7,8 \end{cases}$$

$$\omega_i = \begin{cases} 4/9 & i = 0 \\ 1/9 & i = 1,2,3,4 \\ 1/36 & i = 5,6,7,8 \end{cases}$$

and for the D2Q5 lattice:

$$\mathbf{c}_i = \begin{cases} (0,0) & i = 0 \\ (\pm c, 0), (0, \pm c) & i = 1,2,3,4 \end{cases}$$

$$\omega_i = \begin{cases} 1/3 & i = 0 \\ 1/6 & i = 1, 2, 3, 4 \end{cases}$$

Acknowledgments

Referees are gratefully acknowledged for their help in improving the quality of the manuscript.

References

[1] R. G. Owens and T. N. Phillips. *Computational Rheology*. Imperial College Press, 2002.

[2] R. B. Bird, R.C. Armstrong, and O. Hassager. *Dynamics of Polymeric Liquids Vol. 1 (Fluid Mechanics)*. Wiley-Interscience, 1987.

[3] R. B. Bird, C. F. Curtiss, R. C. Armstrong, and O. Hassager. *Dynamics of Polymeric Liquids Vol. 2 (Kinetic Theory)*. Wiley-Interscience, 1987.

[4] R. Keunings. Micro-macro methods for the multiscale simulation of viscoelastic flow using molecular models of kinetic theory. *Rheology Reviews (British Society of Rheology)*, pages 67–98, 2004.

[5] M. Laso and H.C. Öttinger. Calculation of viscoelastic flow using molecular models: the CONNFFESSIT approach. *Journal of Non-Newtonian Fluid Mechanics*, 47(0):1–20, 1993.

[6] B. Jourdain, C. Le Bris, and T. Lelièvre. On a variance reduction technique for micro-macro simulations of polymeric fluids. *Journal of Non-Newtonian Fluid Mechanics*, 122(1-3):91–106, 2004.

[7] M.A. Hulsen, A.P.G. van Heel, and B.H.A.A. van den Brule. Simulation of viscoelastic flows using brownian configuration fields. *Journal of Non-Newtonian Fluid Mechanics*, 70(1-2):79–101, 1997.

[8] C. Lozinski, A.and Chauvière. A fast solver for Fokker-Planck equation applied to viscoelastic flows calculations: 2D FENE model. *Journal of Computational Physics*, 189(2):607–625, 2003.

[9] C. Chauvière and A. Lozinski. Simulation of complex viscoelastic flows using the Fokker-Planck equation: 3D FENE model. *Journal of Non-Newtonian Fluid Mechanics*, 122(1-3):201–214, 2004.

[10] D. Moroni, B. Rotenberg, J.P. Hansen, S. Succi, and S. Melchionna. Solving the Fokker-Planck kinetic equation on a lattice. *Phys. Rev. E*, 73:066707, 2006.

[11] A. Ammar, B. Mokdad, F. Chinesta, and R. Keunings. A new family of solvers for some classes of multidimensional partial differential equations encountered in kinetic theory modeling of complex fluids. *Journal of Non-Newtonian Fluid Mechanics*, 139(3):153 – 176, 2006.

[12] A. Ammar. Lattice Boltzmann method for polymer kinetic theory. *Journal of Non-Newtonian Fluid Mechanics*, 165(19-20):1082–1092, 2010.

[13] F. Wu, W. Shi, and F. Liu. A lattice Boltzmann model for the Fokker-Planck equation. *Communications in Nonlinear Science and Numerical Simulation*, 17(7):2776 – 2790, 2012.

[14] A. Majumder, V. Kariwala, S. Ansumali, and A. Rajendran. Lattice Boltzmann method for multi-dimensional population balance models in crystallization. *Chemical Engineering Science*, 70(0):121 – 134, 2012.

[15] T. Li. Mathematical analysis of multi-scale models of complex fluids. *Communications in Mathematical Sciences*, 5(1):1–51, 2007.

[16] A. Onuki. *Phase Transition Dynamics*. Cambridge University Press, 2002.

[17] H. R. Warner. Kinetic theory and rheology of dilute suspensions of finitely extendible dumbbells. *Industrial & Engineering Chemistry Fundamentals*, 11(3):379–387, 1972.

[18] J. Barrett and E. Süli. Existence of global weak solutions to some regularized kinetic models for dilute polymers. *Multiscale Modeling & Simulation*, 6(2):506–546, 2007.

[19] R.O. Vargas, O. Manero, and T.N. Phillips. Viscoelastic flow past confined objects using a micro-macro approach. *Rheologica Acta*, 48:373–395, 2009.

[20] H. Versteeg and W. Malalasekra. *An Introduction to Computational Fluid Dynamics: The Finite Volume Method Approach*. Prentice Hall, 1996.

[21] S. Succi. *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond*. Oxford University Press, 2001.

[22] S. Chen and G. D. Doolen. Lattice Boltzmann method for fluid flows. *Annual Review of Fluid Mechanics*, 30(1):329–364, 1998.

[23] G. M. Leonenko and T. N. Phillips. The prediction of plane Couette flow for a FENE fluid using a reduced basis approximation of the Fokker-Planck equation. *International Journal for Multiscale Computational Engineering*, 9(1):73–88, 2011.

[24] X. Hu, Z. Ding, and L. J. Lee. Simulation of 2D transient viscoelastic flow using the CONNFFESSIT approach. *Journal of Non-Newtonian Fluid Mechanics*, 127(2-3):107–122, 2005.

[25] Sailfish reference manual and repository, 2012. URL `http://sailfish.us.edu.pl/`.

[26] Nvidia developer zone, 2012. URL `https://developer.nvidia.com/category/zone/cuda-zone`.

[27] Cuda toolkit downloads, 2012. URL `https://developer.nvidia.com/cuda-downloads`.

[28] Jonas Tölke. Implementation of a lattice Boltzmann kernel using the Compute Unified Device Architecture developed by nVIDIA. *Computing and Visualization in Science*, 13(1):29–39, 2010.

[29] F. Kuznik, C. Obrecht, G. Rusaouen, and J.J. Roux. LBM based flow simulation using GPU computing processor. *Computers & Mathematics with Applications*, 59(7):2380–2392, 2010.

[30] Cuda visual profiler, 2012. URL `https://developer.nvidia.com/nvidia-visual-profiler`.