Highlights

• A sharp-interface Cartesian grid method for viscoelastic fluid flow.
• Accurately resolved the viscoelastic stress on the cylinder surface.
• Oldroyd-B fluid flow past a pair of cylinders.
• Comparison between the parallel SICG and the parallel SIIB method.
A sharp-interface Cartesian grid method for viscoelastic fluid flow in complex geometry

Wei Yi\textsuperscript{a,b,*}, Daniel Corbett\textsuperscript{a}, Xue-Feng Yuan\textsuperscript{a,c}

\textsuperscript{a}Manchester Institute of Biotechnology, The School of Chemical Engineering and Analytical Science, The University of Manchester, 131 Princess Street, Manchester M1 7DN, United Kingdom
\textsuperscript{b}State key Laboratory of High Performance Computing, The School of Computer, National University of Defense Technology, Changsha, P. R. China
\textsuperscript{c}National Supercomputer Centre in Guangzhou, Research Institute on Application of High Performance Computing, Sun Yat-Sen University, Guangzhou, P. R. China

Abstract

Computational methods based on Cartesian mesh are popular in simulating fluid flow with moving boundaries of complex geometry. In this paper, a sharp-interface Cartesian grid method is proposed for simulating viscoelastic fluid flow. We implement a finite volume numerical scheme with an improved Rhie-Chow interpolation on the open-source toolbox OpenFOAM. In the benchmark test of flow past a stationary cylinder, the velocity for Newtonian fluid flow is found to be second-order accurate with linear/bi-linear fitting functions for local reconstruction and third-order accurate with quadratic fitting functions. Only first-order accuracy is achieved with current solver for the Oldroyd-B fluid flows due to the difficulty in handling the extra stress near the boundary. However, our sharp-interface Cartesian grid method has been verified to correctly predict the extra stress on the surface of the cylinder. Simulation results of confined Oldroyd-B fluid flow past a pair of cylinders are also reported. A comparison between the proposed sharp-interface Cartesian grid method and a smoothed-interface immersed boundary method is carried out with respect to accuracy and efficiency.

Keywords: sharp interface, local reconstruction, OpenFOAM, viscoelastic fluid flow

\textsuperscript{*}Corresponding author. Address: State key Laboratory of High Performance Computing, The School of Computer, National University of Defense Technology, Changsha, P. R. China, Tel: +86-13574101298 Email:ywei@nudt.edu.cn
1. Introduction

Developing numerical methods for simulating viscoelastic fluid flow with stationary or moving boundaries of complex geometry is critical for the study of many particulate flow problems. Interesting topics previously studied include: behavior of a single particle in a micro-channel flow of complex fluid with rotation [1, 2] and cross-flow migration [3, 4, 5, 6, 7, 8], flow-induced evolution of microstructure in a non-Newtonian suspension matrix [9, 10, 11, 12, 13, 14, 15, 16] and self-propulsion of micro-swimmers in a bio-fluid [17, 18, 19].

Previous simulations of particulate flows focused on approximating the flow with potential flow, Stokes flow, or point-type-particle flow. Important features, such as viscosity, inertia or the particle orientation were neglected. In direct simulation of particulate flow with fully resolved hydrodynamics, either a body-fitted mesh that conforms to the solid-fluid interface, or a Cartesian mesh with extra handling at the interface is employed. The body-fitted mesh can resolve the boundary field accurately but with high expense in re-meshing. The Cartesian mesh requires less math operations in the discretisation of differential equations, and its regularity is more promising for the optimization of load-balancing on parallel machines.

Early direct numerical simulation of particulate flow with a viscoelastic medium used a finite element method on a body-fitted mesh, with fluid motion described by the Navier-Stokes equations and solid motion described by the Newton-Euler equations. At each time iteration, the simulation includes three steps: an automatic re-meshing step according to new positions of particles, a projection step mapping the flow field from the old mesh to the new mesh, and the solving of governing equations on the new mesh [20]. Huang and Feng first used this method for simulating a viscoelastic fluid flow around a stationary cylinder [21]. Later on, Feng et al. reported a study of moving boundary case using the same method [22]. Huang et al. improved the numerical scheme by introducing the ALE moving mesh technique for calculating the motion of particles [23, 24]. The ALE method allows the mesh inside the computational domain to move arbitrarily, allowing precise tracking of interfaces in a fluid-structure system. D’Avino et al. used an ALE-based finite element method to study cross-flow migration of particles subjected to a Giesekus fluid shear flow in 2D [4]. They reduced the re-meshing cost by constricting the re-meshing region to a narrow channel. Code verification was done by a comparison with the result using a fictitious domain method. The fictitious domain method was found to require a calculation time about 10 times higher than the ALE method, in which a body-fitted mesh was used to resolve the boundary flow. Such body-fitted mesh was more efficient than the uniform Cartesian mesh used in the fictitious domain method. However, it is worth to notice while simulating a system of multiple particles, the Cartesian mesh would be much more efficient. D’Avino et al. later presented a 3D simulation of the dynamics of a particle suspended in a Giesekus fluid under confined shear using the same method [3]. To reduce the cost in re-meshing, they avoided updating the particle position in the main flow direction by assigning the grid with a corresponding velocity instead. The same method
had been adopted for a series of studies on the dynamics of a spherical particle with ignorable inertial effects in confined viscoelastic fluid flow including i) the formation of a separatrix that distinguishes the cross-stream migration direction [6], ii) the design rule of viscoelasticity induced single-line focusing equipment [25], iii) the effect of shear-thinning and the effect of secondary flow on the cross-stream migration [7]. The re-meshing step becomes the bottleneck of performance in large scale simulations thereafter, as it introduces a substantial computational complexity.

In a Cartesian grid method, there are several grid cells that are partly covered by the immersed object. They should be interpreted as in a two-phase state. These Eulerian cells are not physically well-defined, when the surface of the immersed object is rigid. The volume-of-fluid (VOF) method [26] introduces a new parameter "volume fraction of fluid" to the computational model, so that the solid can be treated as a general fluid with extremely high viscosity. The VOF method is commonly adopted in solving multi-phase flow with fluid-fluid interface or free surface [27].

The cutting-cell method uses an embedded body-fitted mesh near the immersed boundary based on a Cartesian mesh for resolving the boundary flow [28, 29, 30]. The cutting cell method calculates the intersections between the Lagrangian grid which represents the moving boundary and the stationary Cartesian grid. These intersections in conjunction with the original vertexes in the fluid part form new irregular units. The boundary conditions therefore can be exactly enforced on the boundary. The difficulties in applying the method lie in a) the complexity of computing the intersections, especially in 3D; b) the possible generation of computational units with small volume, where cell combination is necessary to avoid instability.

The immersed boundary method [31] tracks moving boundaries with a moving Lagrangian grid but solves the fluid flow on a stationary Eulerian grid. The exchange of information between the Lagrangian grid and the Eulerian grid is accomplished by interpolation operations with a smoothed-delta function. This method can be quickly implemented based on any existing numerical solvers and then extended to simulate viscoelastic fluid flow. Goyal and Derksen demonstrated its capability through the simulation of spherical particles sedimenting in a FENE-CR fluid flow [32]. The smoothed delta function would result in a smooth transition of velocity across the sharp solid-fluid interface thus the stress on the interface becomes much less accurate than a body-fitted mesh method, even non-physical. In addition, an empirical hydrodynamic radius was suggested to make a better prediction of drag force because the original method gave a over-predicted drag [33, 32].

The fictitious domain method is a popular method for simulating particulate flow on the structured mesh. A common feature of this method and the immersed boundary method is the involvement of the solid part in solving the fluid flow. The solid-fluid motion is handled implicitly and there are no explicit steps to calculate the force and torque on the particles. Glowinski et al. proposed a distributed Lagrange multiplier/fictitious domain method with a finite element scheme [34]. The motion of the entire computational domain was described by a combined equation [34, 35, 36]. The rigid body motion constraint in the solid part was particularly enforced by
adding an appropriate distributed Lagrangian multiplier term into that equation. Singh et al. developed the viscoelastic version of the method for the simulation of particle sedimentation in a 2D Oldroyd-B fluid flow [37]. Yu et al. investigated the sedimentation of 102 circular particles in a shear-thinning fluid in 2D [38] and the sedimentation of a single spherical particle in a Bingham fluid [39].

However, both the immersed boundary method and the fictitious domain method are difficult to resolve the flow field at the solid-fluid interface, where high shear rate or extensional rate might result in a high viscoelastic stress in viscoelastic fluid flows. In this study, we proposed a sharp-interface Cartesian grid method for simulating viscoelastic fluid flow based on the methods reported in [40, 41]. The flow field in the vicinity of the solid-fluid interface is reconstructed using a polynomial fitting function, with which the Dirichlet or the Neumann boundary conditions can be well-imposed. Our numerical scheme is distinguished from previous studies in [40, 41] in the following aspects: a) our implementation considers an improved Rhie-Chow interpolation [42, 43], which is introduced to avoid the chequer-board effect associated with co-located grid; b) the solver is capable of resolving the viscoelastic stress at the solid-fluid interface accurately in a viscoelastic fluid flow; c) the code developed is fully compatible for parallel simulation on MPI-based clusters. New aspects of this study also include: a) a new study of Oldroyd-B fluid flow past a pair of cylinders with a blockage of 50%; b) a comparison of the proposed sharp-interface Cartesian grid method with the smoothed-interface immersed boundary method in terms of accuracy and parallel efficiency.

The rest of this paper is organized as follows, Part 2 presents the detail of numerical scheme in solving the governing equations. Part 3 evaluates the order of accuracy in space and verifies the correctness of the implementation for simulating Newtonian fluid flow. Part 4 evaluates the order of accuracy in space and verifies the capability of the method for simulating a confined Oldroyd-B fluid flow past a stationary cylinder with a blockage of 50%. To highlight the advantage of the sharp-interface Cartesian grid method, Part 4 also describes the simulation of flow past two closely positioned cylinders. Part 5 compares the proposed sharp-interface Cartesian method with a smoothed-interface immersed boundary method. Part 6 concludes the paper.

2. Numerical method

2.1. Governing equations

The incompressible fluid flow is governed by the Navier-Stokes equations composed of a momentum equation and a continuity equation.

\[ \partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \frac{\nabla \cdot \sigma}{\rho_f} + \nabla p_0 \]  

(1)

\[ \nabla \cdot \mathbf{u} = 0 \]  

(2)
where $\sigma$ is the total stress, $\rho_f$ is the density of the fluid, $\partial_t$ indicates a partial derivative with respect to time, $\nabla p_0$ has a constant component in the streamwise direction, which represents a constant pressure gradient used for driving the flow, $p$ is the corrected kinematic pressure. $\sigma$ has the solvent viscous part $\sigma_s$ and the viscoelastic part $\sigma_p$, and is given by $\sigma = \sigma_s + \sigma_p$. The viscous part is dependent on the strain rate tensor, $\sigma_s = \eta_s (\nabla u + \nabla u^T)$, where $\eta_s$ is the viscosity of the solvent. The viscoelastic part is obtained by solving a constitutive equation for the viscoelastic fluid.

The Oldroyd-B model is applied for the validation of the proposed numerical method in this paper.

\[
\sigma_p + \lambda \left( \frac{\partial \sigma_p}{\partial t} + u \cdot \nabla \sigma_p \right) = \eta_p (\nabla u + \nabla u^T) + \lambda (\sigma_p \cdot \nabla u + \nabla^T u \cdot \sigma_p) \tag{3}
\]

where $\eta_p$ is the polymer contribution to the viscosity, $\lambda$ is the relaxation time of the viscoelastic fluid.

2.2. Numerical schemes

2.2.1. Momentum equation

The momentum equation is discretised with a co-located grid finite volume method in OpenFOAM. Face-centre variables are linearly interpolated from cell-centre variables. Face-centre gradient are computed using the central difference scheme. Gauss’s theorem is applied in computing the integral of a gradient or divergence term of a finite volume unit. It is straightforward to obtain a discretisation of the momentum equation with second-order accuracy in space with a uniform Cartesian grid. The convective term and the viscous term are computed with the Adam-Bashforth scheme and the Crank-Nicholson scheme, respectively. The extra stress is computed with a forward Euler scheme. Eqn. 4 describes the temporal discretisation scheme.

\[
\frac{u^{n+1} - u^n}{\Delta t} = -\frac{3}{2} C[u^n] + \frac{1}{2} C[u^{n-1}] - \nabla p^n + \frac{\eta_p}{2\rho_f} (\nabla^2 u^n + \nabla^2 u^{n+1}) + \frac{1}{\rho_f} \nabla \cdot \sigma_p^n + \nabla p_0 \tag{4}
\]

where $C$ is the convection operator that $C[u] = u \cdot \nabla u$, and the superscript is an indicator of time step.
2.2.2. Pressure equation

For clarity, we reformulate the discretised momentum equation (Eqn. 4) in the following form:

\[ a_P u_P^{n+1} = H_P - (\nabla p^{n+1})_P \]

where \( a_P \) is the diagonal coefficient of the linear equations, \( H_P \) is a combination of the implicit off-diagonal terms, the source terms and the explicit terms excluding the kinematic pressure terms. The superscript of the pressure term \( \nabla p^{n+1} \) is different in Eqn. 4. The pressure from the previous time step is used for predicting the velocity by solving the momentum equation. The pressure in this equation is used for applying the divergence free constraint to the velocity.

By taking the divergence on both sides of Eqn. 5, the Poisson equation of the pressure (Eqn. 7) is derived based on the continuity condition \( \nabla \cdot u_P^{n+1} = 0 \). Our scheme for solving the pressure field is the same as in [44].

\[ \nabla \cdot u_P^{n+1} = \nabla \cdot H_P a_P - \nabla \cdot \left( \frac{\nabla p^{n+1}}{a_P} \right) \]

(6)

\[ \frac{\nabla^2 p^{n+1}}{a_P} = \nabla \cdot \left( \frac{H_P}{a_P} \right) \]

(7)

The discretisation of the momentum equation and the Poisson equation on a co-located grid is vulnerable to nonphysical numerical oscillations (chequer-board effect), when a central difference scheme is used. This is due to the decoupling of velocity and pressure on odd and even grid. Rhie-Chow interpolation [45] was introduced to amend this defect by adding a dissipation term while computing the flux. With the original Rhie-Chow interpolation, Choi[42] found that the steady state solution was dependent on the time step. Yu et al. [43] suggested that the chequer-board effect could still appear with a small time step. An improved Rhie-Chow interpolation [42, 43] to resolve this issue is utilized in our implementation. We define an intermediate velocity \( u^* \) that satisfies \( u^*_P = \frac{H_P}{a_P} \) for a cell centre ‘P’, and apply the divergence theorem to calculate the volume integral of the r.h.s of Eqn. 7,

\[ \int_V \nabla \cdot \frac{H_P}{a_P} dV = \int_{dV} u^*_j \cdot \eta dS = \sum u^*_j \cdot S_j \]

(8)

where \( \eta \) is the surface normal, \( S_j \) is the vector area of the surface, \( u^*_j \) is the surface intermediate velocity from an interpolation of cell-centre variable \( u^* \).

Considering a face centre \( e \) connecting two adjacent grid cells \( P \) and \( E \) in a uniform grid, the intermediate velocity \( u^*_e \) can be interpolated from \( u^*_P \) and \( u^*_E \). A standard Rhie-Chow interpolations is given by

\[ u^*_e = \frac{1}{2}(u^*_E + u^*_P) \]

(9)
while an improved Rhie-Chow interpolation is given by

\[ u^*_{n+1} = \frac{1}{2}(u^*_{E} + u^*_{P}) + \frac{u^n_{e}}{a_{e}\Delta t} - \frac{1}{2\Delta t} \left( \frac{u^n_{P}}{a_{P}} + \frac{u^n_{E}}{a_{E}} \right) \]  

(10)

where \( a_E, a_P \) and \( a_e \) are the same except for boundary grid cells. The formulation of the improved Rhie-Chow interpolation is different from that in [42]. Choi [42] solved the Poisson equation for a pressure correction, thus their intermediate velocity includes a pressure contribution. It is straightforward to prove that their formulation [42] is equivalent to the formulation in Eqn. 10. After solving the pressure, the cell-centre and face-centre velocity can therefore be computed as follows,

\[ u^{n+1}_p = u^*_p - \frac{\nabla p^{n+1}}{a_p} \]  

(11)

\[ u^{n+1}_f = u^*_f - \frac{\nabla p^{n+1}}{a_f} \]  

(12)

2.2.3. Oldroyd-B constitutive equation

The Oldroyd-B constitutive equation is solved based on the prediction of velocity at the end of each time iteration. All terms on the l.h.s of Eqn. 3 are discretised implicitly while the terms on the r.h.s are computed explicitly.

\[ \sigma^{n+1}_{pp} + \lambda \left( \frac{\sigma^n_{pp}}{\Delta t} + C[\sigma^{n+1}] \right) = \eta_p \left\{ \nabla u^{n+1} + (\nabla u^{n+1})^T \right\} + \lambda (\sigma^n_{pp} \cdot \nabla u^{n+1} + \nabla (u^{n+1})^T \cdot \sigma^n_{pp}) \]  

(13)

The convective term is discretised with the MINMOD scheme for stability reasons. [46] found the MINMOD scheme was about order 1.8 \(-\) 1.9 accurate with an investigation of the drag coefficient for flow past a stationary cylinder. The divergence of extra stress \( \nabla \cdot \sigma_{pp} \) is introduced into the momentum equation as an explicit source term for the prediction of velocity at the next time step.

2.3. Boundary local reconstruction

The fluid flow is solved on a Cartesian mesh while the position of the immersed boundary is tracked by a detached Lagrangian mesh. Figure 1 shows the categories of grid cells on a uniform Cartesian mesh with a 2D cylindrical immersed boundary. The grid cells are categorized into four types: forcing cells, ghost cells, normal fluid cells and normal solid cells. Forcing cells have at least one neighbour with its centre inside the cylinder. Ghost cells have centres inside the cylinder and at least one forcing-cell neighbour. Cells with centres outside the cylinder except forcing cells are normal fluid cells. Cells with centres inside the cylinder except ghost cells are normal solid cells, and are not involved in the computation. The application of boundary conditions at the fluid-solid interface is challenging since Lagrangian points are not necessarily overlapping with
Figure 1: Categorization of grid cells on a uniform Cartesian mesh with a cylindrical immersed boundary. The circles are ghost cells, minuses are forcing cells, and pluses are normal fluid cells.

cell-centres of the Cartesian grid. In this work, the flow field at forcing cells of the Cartesian grid are locally reconstructed to meet the boundary conditions enforced on the Lagrangian grid using polynomial fitting functions. Specifically, variables at forcing cells are not solved in the linear system of equations, but are explicitly reconstructed based on variables at normal fluid cells in an iterative manner. The reconstruction at ghost cells is important in suppressing the nonphysical oscillation when simulating moving boundaries [40]. The reconstruction at ghost cells is similar to that at forcing cells.

The Dirichlet boundary condition for the velocity field, and the Neumann boundary condition for the pressure field can be well-imposed with fitting functions. Due to the hyperbolic nature of the viscoelastic constitutive equations, in principle there is no need to enforce a boundary condition at the cylinder wall. However, we have found that directly solving the constitutive equation at the cylinder wall causes strong non-physical oscillations and quickly lead to divergence, which was also reported in [47]. Linear extrapolation and Neumann boundary condition for the extra stress on the cylinder wall are therefore considered in our simulations.

2.3.1. Velocity field

The velocity of an immersed boundary is either pre-specified or solved in a system that couples the velocity of the immersed object with the flow field, e.g. Newton-Euler equations for a rigid solid body. The velocity is the same for the fluid and the solid at the interface, where no-slip boundary condition is applied. A local reconstruction method assumes that the velocity profile in the vicinity of the immersed boundary can be approximated by polynomial fitting functions.

Linear reconstruction. Three points are required to get the fitting coefficients $b_0 \sim b_2$ for a two-dimensional linear function $u = b_0 + b_1 x + b_2 y$. The coefficients can be solved with the following
Figure 2: A linear fitting function stencil example for forcing cells at a cylindrical immersed boundary. The circles are ghost cells, minuses are forcing cells, and pluses are normal fluid cells. $O$ is the cylinder centre, $A$ is the cell centre of a forcing cell, $Q$ is a point on the extension from $O$ to $A$ inside the forcing cell, and $C$ is the intersection between the segment $OA$ and the circumference. The square stencil $AEFD$ are the nearest $2 \times 2$ cell-centres for $Q$, and the triangular stencil $CDE$ is the reconstruction stencil for the forcing cell $A$ for a two-dimensional linear fitting function: $u = b_0 + b_1x + b_2y$. Similarly, the triangular stencil $C' D' E'$ is the reconstruction stencil for the ghost cell $A'$.

Figure 2 illustrates candidate cells used for solving coefficients of a linear function. Specifically, for the forcing cell centred at $A$, the connection $OA$ between the cylinder centre and the cell-centre intersects with the cylinder circumference at $C$. The intersection $C$ in conjunction with cell-centres of two neighbours of the forcing cell are used for the reconstruction. A $2 \times 2$ stencil that encompasses the forcing cell and three neighbours (two nearest neighbours and one diagonal neighbour) is chosen based on $OA$. By extending $OA$ slightly to a point $Q$ and finding the nearest $2 \times 2$ stencil, it is guaranteed that there are either two or three normal fluid cells in the stencil. When three normal fluid cells (two nearest neighbours and one diagonal neighbour) are in the stencil, the two nearest neighbours and the point on the boundary makes a more compact stencil.

Bi-linear reconstruction. Four points are required to get the coefficients $b_0 \sim b_3$ for a two-dimensional bi-linear function $u = b_0 + b_1x + b_2y$, $b_3xy$. The coefficients can be solved with the system of equations,

$$\begin{bmatrix} 1 & x_0 & y_0 \\ 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} u_0 \\ u_1 \\ u_2 \end{bmatrix}$$

(14)
the following system of equations,

\[
\begin{bmatrix}
1 & x_0 & y_0 & x_0y_0 \\
1 & x_1 & y_1 & x_1y_1 \\
1 & x_2 & y_2 & x_2y_2 \\
1 & x_3 & y_3 & x_3y_3 \\
\end{bmatrix}
\begin{bmatrix}
b_0 \\
b_1 \\
b_2 \\
b_3 \\
\end{bmatrix}
= 
\begin{bmatrix}
u_0 \\
u_1 \\
u_2 \\
u_3 \\
\end{bmatrix}
\]

(15)

The local reconstruction with a bi-linear function is much different from that with a linear function. A fitting function of an image point, instead of a forcing-cell centre, is constructed. Consider a forcing-cell with centre \(A\) (see Figure 3), firstly the intersection point \(C\) between \(OA\) and the cylinder circumference is calculated. An image point \(I\) which satisfies \(CA = AI\) is obtained. The nearest \(2 \times 2\) stencil of the image point can then be determined. Since the image point is further away from the cylinder centre than the forcing-cell centre, cells in the stencil are either forcing cells or normal fluid cells. If the forcing cell belongs to the stencil, point \(C\) on the circumference serves as the fourth point for constructing the bi-linear fitting function. Otherwise the four points in the \(2 \times 2\) stencil are sufficient to determine the fitting coefficients. In particular, unlike the linear fitting function case, cells used to determine the bi-linear fitting function might be forcing cells. Therefore the velocity at forcing cells are mutually dependent. An embedded iteration becomes necessary in the local reconstruction step. Our simulation shows that the number of iterations is less than 10, even when a very low tolerance is set. As only forcing cells are involved in the iteration, the computational cost is almost negligible. The velocity at the forcing cell is computed according to the mid-point rule as follows,

\[
u_A = \frac{u_I + u_C}{2} + O(\Delta l^2)
\]

(16)

The mid-point interpolation introduces a second-order error \(O(\Delta l^2)\), where \(\Delta l\) is the distance between the image point and the forcing-cell centre. The local reconstruction of velocity at the image point \(u_I\) with a bi-linear fitting function has an error of \(O(\Delta x^2)\), where \(\Delta x\) is the Cartesian grid spacing [41]. Therefore the total error is \(O(\Delta x^2 + \Delta l^2)\). As \(\Delta l\) is in the same scale of \(\Delta x\), the overall order of accuracy in space becomes second-order.

**Quadratic reconstruction.** Six neighbouring points are required to get coefficients \(b_0 \sim b_5\) of a quadratic fitting function in the form of \(u = b_0 + b_1x + b_2y + b_3xy + b_4x^2 + b_5y^2\). The coefficients
Figure 3: A bi-linear fitting function stencil example for forcing cells at a cylindrical immersed boundary. The circles are ghost cells, minuses are forcing cells, and pluses are normal fluid cells. $O$ is the cylinder centre, $A$ is the cell centre of a forcing cell, $C$ is the intersection between the segment $OA$ and the circumference and $I$ is the image point that satisfies $CA = AI$. The square stencil $AEFD$ are the nearest $2 \times 2$ cell-centres of $I$, and the quadrangular stencil $CDFE$ is the reconstruction stencil for the image point $I$ for a two-dimensional bi-linear fitting function $u = b_0 + b_1 x + b_2 y + b_3 xy$. Similarly, the quadrangular stencil $G'E'F'D'$ is the reconstruction stencil for the image point $I'$, which satisfies $A'C' = C'I'$, of the ghost cell $A'$.

The system of equations can be solved with the following system of equations:

$$
\begin{bmatrix}
1 & x_0 & y_0 & x_0^2 & y_0^2 \\
1 & x_1 & y_1 & x_1^2 & y_1^2 \\
1 & x_2 & y_2 & x_2^2 & y_2^2 \\
1 & x_3 & y_3 & x_3^2 & y_3^2 \\
1 & x_4 & y_4 & x_4^2 & y_4^2 \\
1 & x_5 & y_5 & x_5^2 & y_5^2
\end{bmatrix}
\begin{bmatrix}
b_0 \\
b_1 \\
b_2 \\
b_3 \\
b_4 \\
b_5
\end{bmatrix}
= 
\begin{bmatrix}
u_0 \\
u_1 \\
u_2 \\
u_3 \\
u_4 \\
u_5
\end{bmatrix}
$$

(17)

Consider a forcing cell with cell-centre at $A$, as shown in Figure 4, by slightly extending the vector $OA$ of a small amount of $0.01 \Delta x$ to $A'$. A $4 \times 4$ subgrid which includes the nearest 16 cell-centres to $A'$ are obtained (the $4 \times 4$ dashed line stencil in Figure 4). This subgrid provides candidate cells for the reconstruction. Only normal fluid cells in the subgrid are selected and sorted by their distance to $A'$. Points $D$, $E$, $F$, $G$ and $H$ are the nearest five point to point $A'$. Candidates are chosen to guarantee that no four points are on the same line to avoid the singularity of the coefficient matrix. Only normal fluid cells are employed for the reconstruction of a forcing cell, thus there are no mutual dependency between forcing cells and iteration is unnecessary.

Linear, bi-linear and quadratic reconstructions require the invert of a $3 \times 3$, $4 \times 4$ and $6 \times 6$ matrix. These can be done with the help of an open source package ALGLIB [48].
Figure 4: A quadratic fitting function stencil example for forcing cells at a cylindrical immersed boundary. The circles are ghost cells, minuses are forcing cells, and pluses are normal fluid cells. $O$ is the cylinder centre, $A$ is the cell centre of a forcing cell, $C$ is the intersection between the segment $OA$ and the circumference, and $A'$ satisfies $AA' = 0.01\Delta x OA / |OA|$. The dashed line stencil are the nearest $4 \times 4$ cell-centres of $A'$. The five nearest centres of normal fluid cells in the stencil to $A'$ in conjunction with point $C$ on the boundary forms the reconstruction stencil of the forcing point $A$ for a two-dimensional quadratic fitting function $u = b_0 + b_1 x + b_2 y + b_3 xy + b_4 x^2 + b_5 y^2$.

2.3.2. Pressure field

Neumann boundary condition is applied for the pressure. The pressure gradient along the surface normal direction is derived from the momentum equation, written as,

$$\frac{dp}{dn} = - \frac{D u}{Dt} \cdot n + \frac{\eta \nabla^2 u \cdot n}{\rho_f} + \frac{\nabla \cdot \sigma_p \cdot n}{\rho_f} + \nabla p_0 \cdot n$$

(18)

where $n = (n_x, n_y)$ is the surface normal, $\frac{Du}{Dt}$ is the material derivative of velocity on a Lagrangian point, which is zero for a stationary boundary. $\nabla u$ and $\nabla \sigma_p$ can be calculated with fitting functions of $u$ and $\sigma_p$ respectively. The viscous term $\nabla^2 u$ is non-zero only when quadratic fitting functions are applied for the velocity. Assuming a linear fitting function for the pressure $p = b_0 + b_1 x + b_2 y$,

$$\frac{dp}{dx} = b_1, \quad \frac{dp}{dy} = b_2$$

therefore

$$\frac{dp}{dn} = (\frac{dp}{dx} \cdot \frac{dp}{dy}) \cdot (n_x, n_y) = b_1 n_x + b_2 n_y$$

(19)

The coefficients of the fitting function can be obtained by solving:

$$\begin{pmatrix} 0 & n_x & n_y \\ 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \end{pmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} dp/dn \\ p_1 \\ p_2 \end{bmatrix}$$

(20)
Using bi-linear fitting functions for the pressure:

\[ p = b_0 + b_1 x + b_2 y + b_3 xy, \]

\[ \frac{dp}{dn} = b_1 n_x + b_2 n_y + b_3 (y n_x + x n_y) \]  

\[ \text{(21)} \]

Similarly, using quadratic fitting functions for the pressure:

\[ p = b_0 + b_1 x + b_2 y + b_3 xy + b_4 x^2 + b_5 y^2, \]

\[ \frac{dp}{dn} = b_1 n_x + b_2 n_y + b_3 (y n_x + x n_y) + 2b_4 x n_x + 2b_5 y n_y \]

\[ \text{(22)} \]

The same method for selecting candidate cells in the reconstruction of velocity can also be applied in the reconstruction of the pressure.

2.3.3. Extra stress field

**Solving directly.** The constitutive equation for the Oldroyd-B fluid is shown in the following,

\[ \sigma_p + \lambda \left( \frac{\partial \sigma_p}{\partial t} + u \cdot \nabla \sigma_p \right) = \eta_p \left( \nabla u + \nabla u^T \right) + \lambda \left( \sigma_p \cdot \nabla u + \nabla u^T \cdot \sigma_p \right) \]

\[ \text{(23)} \]

The value of extra stress on the boundary is required for the discretisation of the convective term \( \nabla \sigma_p \), and additionally for computing the divergence term \( \nabla \cdot \sigma_p \) in the momentum equation. When the boundary is stationary (\( u = 0 \)), the convective term vanishes. As a result, the extra stress is only a function of the strain rate tensor. Theoretically the extra stress can be solved in the absence of boundary conditions. Using the backward Euler scheme, the constitutive equation is given by:

\[ (1 + \frac{\lambda}{\Delta t}) \sigma_p^{n+1} - \lambda \left( \sigma_p^{n+1} \cdot \nabla u + \nabla u^T \cdot \sigma_p^{n+1} \right) = \frac{\lambda}{\Delta t} \sigma_p^n + \eta_p \left( \nabla u + \nabla u^T \right) \]

\[ \text{(24)} \]

where the left hand side includes all the implicit terms of the extra stress.

The extra stress on the particle wall can therefore be used for reconstructing the extra stress at forcing/ghost cells. Solving the constitutive equation at the wall was suggested by Oliveira et al. [47] for simulating viscoelastic fluid flow with the finite volume on a body-fitted mesh. However, the solution relies highly on the prediction of the strain rate tensor. Our simulation results show that the strain rate tensor on the cylinder wall predicted using a Cartesian mesh method has non-physical oscillations. These oscillations quickly lead to the divergence of simulations.

**Neumann boundary condition.** Neumann boundary condition \( \nabla \sigma_p \cdot n = 0 \), where \( n \) is the surface normal, can be applied for the extra stress on the surface. The error of applying the Neumann boundary condition is similar to that of the upwind interpolation scheme, where a first-order error is presented. Considering the Taylor series expansion for a stress component \( \sigma_{p,xx} \) in 1D:
\[
\sigma_{p,xx}(y + \Delta y) = \sigma_{p,xx}(y) + \frac{\partial \sigma_{p,xx}}{\partial y} (y) \Delta y + O(\Delta^2 y). 
\]
The first derivative of the stress component is non-zero on the surface, hence a first-order error is introduced into the solution by enforcing the Neumann boundary condition. The analytical solution for the Poiseuille flow of an Oldroyd-B fluid was reported in [49, 50]. At steady state, the \(xx\)-component of the extra stress is a quadratic function of the \(y\)-coordinate, while the \(xy\)-component is a linear function of the \(y\)-coordinate. Although the Neumann boundary condition is first-order accurate, it is highly stable.

**Linear extrapolation.** Linear extrapolation schemes are implemented for calculating \(\sigma_p\) at forcing cells and ghost cells. Two types of extrapolation schemes are proposed. In scheme (a), for the extrapolation of the extra stress at a forcing cell, the compact stencil close to the forcing cell includes three cells, of which are either normal fluid cells or forcing cells. Iteration is required because the forcing cells might be dependent on each other. In scheme (b), the extrapolation stencil is less compact without mutual dependency. The stress at forcing cells is extrapolated only from the stress at normal fluid cells. These two schemes are found to produce similar results. Scheme (b) is preferable because it requires no internal iterations. The methods for selecting the cells in the stencil are described in the following.

(a) Linear extrapolation with a compact stencil: In Figure 5, \(\sigma_p\) at a forcing cell \(A\) can be determined by computing a linear fitting function with \(\sigma_p\)s at \(D, F\) and \(E\). The triangular stencil \(DFE\) is made of three nearest cell-centres of \(Q\). \(Q\) is obtained by slightly extending the vector, which begins at the cylinder centre \(O\) and ends at \(A\), with a small amount of \(0.01\Delta x\). Similarly, the extrapolation stencil for a ghost cell \(A'\) is \(D'F'E'\). Since the value at a forcing/ghost cell might rely on other forcing/ghost cells, iteration is required.

(b) Linear extrapolation without mutual dependency: In Figure 6, \(\sigma_p\) at a forcing cell \(A\) can be determined by computing a linear fitting function with \(\sigma_p\)s at \(B, C\) and \(D\). The \(3 \times 3\) subgrid includes the nine nearest cell-centres for \(Q\). \(Q\) is obtained by extending the vector, which begins at the cylinder centre \(O\) and ends at \(A\), with an amount of \((\sqrt{2} + 0.01)\Delta x\). The subgrid provides candidate cells for the extrapolation. Only normal fluid cells in the subgrid are selected and sorted by the distance to \(A'\). \(A'\) is obtained by extending the vector, which begins at the cylinder centre \(O\) and ends at \(A\), with a small amount of \(0.01\Delta x\). Points \(B, C\) and \(D\) are the nearest three to \(A'\). If the cell-centres of the three selected cells are on the same line, the third cell is replaced by the cell which ranks the fourth.

### 2.4. Iterative algorithm

We have discussed the discretisation scheme for each individual equation in Section 2.2 and the local reconstruction at the sharp interface in Section 2.3. The solution of each governing equation involves a step for solving the discretised linear equations, a subsequent step for local reconstruction at forcing cells, and an iterative loop which encompasses the previous two steps. The velocity \(u\), pressure \(p\), and extra stress \(\sigma_p\) at forcing cells are initialized with the value from the previous time step in the first iteration. Additionally, the velocity at face-centres is required in
Figure 5: A linear extrapolation fitting function stencil example for forcing/ghost cells at a cylindrical immersed boundary. The circles are ghost cells, minuses are forcing cells and pluses are normal fluid cells, $O$ is the cylinder centre, $A$ is the cell centre of a forcing cell, $A'$ satisfies $\Delta x A A' = 0.01 \Delta x O A / |O A|$, and $Q$ satisfies $\Delta x Q A = 0.01 \Delta x Q A / |Q A|$. Points $D$, $E$ and $F$ are the nearest three cell-centres to $Q$. The stencil $DFE$ makes up the reconstruction stencil for the forcing cell. Similarly, the triangular stencil $D'E'F'$ makes up the reconstruction stencil for the ghost cell $A'$.

Figure 6: A linear extrapolation fitting function stencil example for forcing cells at a cylindrical immersed boundary. The circles are ghost cells, minuses are forcing cells, pluses are normal fluid cells, $O$ is the cylinder centre, $A$ is the cell centre of a forcing cell, $A'$ satisfies $\Delta x A A' = 0.01 \Delta x O A / |O A|$, and $Q$ satisfies $\Delta x Q A = (\sqrt{2} + 0.01) \Delta x O A / |O A|$. $Q$ determines $3 \times 3$ candidates used for the extrapolation. Points $B$, $C$ and $D$ are the nearest three candidates $A'$. The stencil $BCD$ makes up the reconstruction stencil for the forcing cell.
Table 1: Iterative solvers for sparse linear system of equations

<table>
<thead>
<tr>
<th>variable</th>
<th>solver specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>(u)</td>
<td>PCG(^1) preconditioner: DIC(^4)</td>
</tr>
<tr>
<td>(\sigma_p, p)</td>
<td>PBiCG(^2) preconditioner: DILU(^5)</td>
</tr>
<tr>
<td>(p)</td>
<td>GAMG(^3) smoother: GaussSeidel</td>
</tr>
</tbody>
</table>

\(^1\) PCG: preconditioned conjugated gradient;  
\(^2\) PBiCG: preconditioned bi-conjugate gradient;  
\(^3\) GAMG: geometric-algebraic multi-grid;  
\(^4\) DIC: diagonal incomplete-Cholesky;  
\(^5\) DILU: diagonal incomplete-LU.

the discretisation of all three governing equations (the momentum equation, the pressure equation and the Oldroyd-B constitutive equation). Thus an update of velocity or flux at face centres is vitally important following the local reconstruction of cell-centre variables at forcing cells, in order to correctly describe the flow behaviour near the boundary. The iterative solvers for sparse linear systems of equations in this work has been summarised in Table 1. Both the PCG solver and GAMG solver can be used to solve the pressure equation, however the GAMG solver is found to be more efficient. Details for solvers and preconditioners refers to the user guide for OpenFOAM [51].

A comprehensive iterative algorithm is given in Figure 7. The iterative algorithm solves for the velocity field, the pressure field and the viscoelastic field sequentially. The variables at forcing cells are not solved simultaneously with normal fluid cells but kept at their original values and updated with a subsequent local reconstruction step. The algorithm is not necessary to solve the internal part of the immersed boundary. The number of iterations for velocity, pressure and extra stress are fixed to 2, 3 and 2 respectively for the simulation in this work, unless otherwise specified. More iterations are found to have negligible influence on the steady state solution.

2.5. Calculation of drag force

Lagrangian points are spread over the surface of the immersed boundary for the evaluation of the drag force on the immersed object. A Lagrangian point could be either near a forcing cell or a ghost cell. Fitting functions for local reconstruction at forcing/ghost cells can therefore be used to calculate the stress and pressure on Lagrangian points. For instance, the total force on a cylinder can be evaluated by

\[
F = -\int pndS + \int (\sigma_S + \sigma_P) \cdot ndS + \int \nabla p_0 dV
\]  

The distance between neighbouring Lagrangian points is usually the same as the Cartesian grid spacing in the vicinity of the sharp interface.

3. Newtonian fluid flow

3.1. Convergence

A cylinder with radius \(R\) is placed at the centre \((L/2, H/2)\) in a square domain with \(L = H = 4R\), as illustrated in Figure 8. The flow is driven by a fixed pressure difference between the
Fluid-solid coupling iteration starts

- Solve the momentum equation at normal fluid cells [Eqn. 4]
- Reconstruct the velocity \( u \) at forcing cells
- Compute the change in velocity field \( \Delta u \)
- Compute the mass flux across faces [Eqn. 10]
- Solve the pressure equation at normal fluid cells [Eqn. 7]
- Reconstruct the pressure \( p \) at forcing cells
- Correct the flux between neighbouring normal fluid cells [Eqn. 12]
- Compute the changes in pressure field \( \Delta p \)

\[ |\Delta u| > \varepsilon_u \]
\[ |\Delta p| > \varepsilon_p \]

|\Delta u| \leq \varepsilon_u
|\Delta p| \leq \varepsilon_p

Fluid-solid coupling Iteration ends

- Update the motion and position of immersed objects
- Compute the force and torque on immersed objects [Eqn. 25]
- Reconstruct the velocity, pressure and extra stress at ghost cells
- Compute the changes in extra stress field \( \Delta \sigma_P \)
- Reconstruct the extra stress \( \sigma_P \) at forcing cells
- Solve the constitutive equation at normal fluid cells [Eqn. 13]
- Compute the strain rate tensor
- Correct the velocity at normal fluid cells [Eqn. 11]

\[ |\Delta u| \leq \varepsilon_u \]
\[ |\Delta p| \leq \varepsilon_p \]

Figure 7: Iterative algorithm of the SICG method for fluid-solid coupling.
inlet at $x = 0$ and $x = L$ (by setting $\nabla p_0$ in Eqn. 1). Periodic boundary conditions are imposed for both the velocity and the pressure at the inlet and the outlet. Dirichlet boundary conditions $u = (0, 0)$ are applied on the side walls at $y = 0$ and $y = H$ for the velocity. Neumann boundary conditions $\partial p/\partial y = 0$ is applied on the side walls for the pressure. The Reynolds number is defined as $Re = \rho_f D u_\infty / \eta$, where $D$ is the diameter of the cylinder, and $u_\infty$ is the mean flow rate, which is computed by simulation on the $600 \times 600$ mesh. The Reynolds number for the convergence test is around 84. Since there is no analytical solution for the flow, simulation results on the mesh of size $600 \times 600$ ($\Delta x = D/300$) are taken as the reference solution for calculating the error on different meshes. The maximum Courant number is kept at about $0.14$ for this test. We carried out simulation on five meshes sized at $60 \times 60$, $100 \times 100$, $120 \times 120$, $200 \times 200$, $600 \times 600$. The simulation result on the $600 \times 600$ mesh is mapped to coarse meshes with a second-order interpolation scheme. The error of the flow field from the simulation can be quantitatively evaluated by $L_\infty = \max |\varepsilon(i)|$, $L_1 = \sum |\varepsilon(i)| \Delta V$ and $L_2 = \sqrt{\sum \varepsilon^2(i) \Delta V}$, where $\Delta V$ is the normalized volume of a mesh cell, and $\varepsilon(i)$ is the error at the cell centre. $L_\infty$ is a good indicator of the error at the boundary, while $L_1$ and $L_2$ represent the error in the entire computational domain. The steady state is investigated.

The order of convergence is tested with different reconstruction schemes for the pressure and the velocity, including ‘LuLp’, ‘BuBp’, ‘QuLp’ and ‘QuQu’, where ‘L’ indicates linear reconstruction, ‘B’ indicates bi-linear reconstruction, and ‘Q’ indicates quadratic reconstruction, ‘u’ represents the velocity and ‘p’ represents the pressure. For example, ‘LuLp’ means using linear reconstruction for both the velocity and the pressure. Figure 9 illustrates the normalised error of velocity components when using different reconstruction schemes.

When linear fitting functions are used for reconstructing the velocity and the pressure at forcing cells, as shown in Figure 9(a) and 9(b), the order of accuracy with $L_\infty$ is slightly below second-order, while the order of accuracy with $L_1$ and $L_2$ are second-order. A similar convergence is achieved with bi-linear fitting functions.

Using a quadratic fitting function for the velocity at forcing cells and a linear fitting function for the pressure, the order of accuracy with $L_\infty$ is slightly below third-order, while the order of
Figure 9: Normalised error (normalised by $u_\infty$) of velocity components for simulating flow past a cylinder with different grid spacings when $Re = 84$. The slope of the dotted line is 1, the slope of the solid line is 2, and the slope of dashed line is 3. $LuLp$: linear reconstruction for velocity and pressure; $BuBp$: bi-linear reconstruction for velocity and pressure; $QuLp$: quadratic reconstruction for velocity and linear reconstruction for pressure; $QuQp$: quadratic reconstruction for velocity and pressure.
Accuracy with $L_1$ and $L_2$ are third-order, as shown in Figure 9(c) and 9(d). With a quadratic fitting function for the pressure at forcing cells, the numerical error of the velocity is further decreased. However, the measured order of accuracy is slightly lower than that with the QuLp scheme, but also close to third-order.

Significant improvement of accuracy with quadratic fitting functions can also be observed in the calculation of the order of accuracy for the strain rate tensor $\nabla u$, as shown in Figure 10. Linear fitting functions for the velocity field result in a first-order accuracy with $L_\infty$, while quadratic fitting functions result in a second-order accuracy with $L_\infty$, when computing the gradient of velocity. The accuracy of the strain rate tensor directly affects the computation of lift and drag force on the cylinder (see Eqn. 25), as the viscous stress $\sigma_v$ is computed by $\eta_s (\nabla u + (\nabla u)^T)$. In Figure 11, the profile of $(\nabla u)_{xx}$ and $(\nabla u)_{yx}$ along a half circumference of the cylinder in the clockwise direction are shown. Wiggles appear when a coarse mesh is used for the simulation. Such nonphysical oscillations could be the reason why when the QuQp solver is used, the four $L_\infty$ error values with different grid spacings are not aligned (Figure 10(b)). Using a linear fitting function for the velocity, even with the finest mesh, there are still strong wiggles near peak points of $(\nabla u)_{xx}$. Similar wiggles exist near peak points of $(\nabla u)_{yx}$, but are much less significant. In comparison, using a quadratic fitting function for the velocity, profiles of $(\nabla u)_{xx}$ and $(\nabla u)_{yx}$ become smoother and more accurate, as shown in Figure 11(c) and Figure 11(d).

3.2. Validation

The correctness of the code for Newtonian fluid flows is validated with simulation of the benchmark case, unbounded flow past a stationary cylinder. As shown in Figure 12, a cylinder with radius $R$ is placed in a square domain $H \times H$, where $R = 0.15$ and $H = 16$. The centre of the cylinder is at $(6.85, 8)$. No-slip boundary conditions are enforced at the sidewalls ($y = 0$ and $y = H - 0$) and the top and bottom surfaces ($x = 0$ and $x = H - 0$). The fluid is assumed to be incompressible with density $\rho = 1$ and viscosity $\mu = 1$.

![Figure 10: Normalised error (normalised by $u_\infty/H$) of $xx$-component and $yx$-component of the strain rate tensor $\nabla u$ for simulating flow past a cylinder with different grid spacings when $Re = 84$. The slope of the dotted line is 1, the slope of the solid line is 2, and the slope of dashed line is 3.](image)

(a) $xx$-component

(b) $yx$-component

Figure 10: Normalised error (normalised by $u_\infty/H$) of $xx$-component and $yx$-component of the strain rate tensor $\nabla u$ for simulating flow past a cylinder with different grid spacings when $Re = 84$. The slope of the dotted line is 1, the slope of the solid line is 2, and the slope of dashed line is 3.
Figure 11: Normalised components (normalised by $u_\infty/H$) of strain rate tensor for simulating flow past a cylinder with different grid spacings when $Re = 84$.

Figure 12: Computational geometry for unbounded flow past a stationary cylinder in 2D.
and $y = 16$) and the inlet $(x = 0)$ with $u = (u_\infty, 0)$. At the outlet $(x = 16)$, the gradient in the surface normal direction is enforced as zero, i.e. $\partial u / \partial x = 0$. On the sidewalls and at the inlet, zero gradient boundary condition is also enforced for the pressure ($\partial p / \partial n = 0$). At the outlet, Dirichlet boundary condition $p = 0$ is imposed. The four boundaries of the computational domain are about 25$D$ away from the cylinder, thus the wall effects on the flow near the cylinder is insignificant and the flow further away from the cylinder is less important. A locally refined mesh employed in this problem could save a large amount of computational expense without losing much accuracy. The prediction of the flow in the vicinity of the cylinder is as accurate as that with a uniform grid. We use 4-layer locally refined meshes for the simulation. The three layers on top of the base uniform mesh are with domains [6.0, 12.2] × [8.3, 8.7], [6.4, 9.0] × [7.5, 8.5], [6.6, 7.4] × [7.6, 8.4]. Two meshes with minimal grid spacings of $D/60$ and $D/120$ are employed in this study. The time step is chosen so that the Courant number is around 0.4 for all simulations. Table 2 shows that the prediction of drag coefficients are consistent with those reported in literature. The mesh-dependent variation with the QuQp solver is much smaller comparing to that with the LuLp solver. Using the drag coefficient in the QuQp-D/120 group as reference values, we can calculate the error of the LuLp solver with grid spacings of $D/60$ and $D/120$. The former is approximately twice of the latter for both $Re = 20$ and $Re = 40$, demonstrating that the LuLp solver is first-order accurate in computing the drag force, which is consistent with our previous convergence tests for $\nabla u$. For a higher Reynolds number flow, one characteristic phenomenon is the periodical generation of a Karman vortex street, subjected to a small perturbation. A comparison of lift coefficient $C_l = F_y / (\rho u_\infty^2 R)$, drag coefficient $C_d = F_x / (\rho u_\infty^2 R)$, and the Strouhal number $S_l = f_s D / u_\infty$, where $f_s$ is the shedding frequency of the Karman vortex, are shown in Table 3 for $Re = 100$ when the frequency of the vortex becomes steady.

<table>
<thead>
<tr>
<th>$Re$</th>
<th>$C_d^{LuLp-D/60}$</th>
<th>$C_d^{LuLp-D/120}$</th>
<th>$C_d^{QuQp-D/60}$</th>
<th>$C_d^{QuQp-D/120}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>2.06</td>
<td>2.02</td>
<td>2.07</td>
<td>2.057</td>
</tr>
<tr>
<td>40</td>
<td>1.54</td>
<td>1.52</td>
<td>1.53</td>
<td>1.54</td>
</tr>
</tbody>
</table>

Table 2: Drag coefficient for unbounded flow past a stationary cylinder

References:
- Linnick and Fasel (2005)
- Choi et al. (2007)
- Mittal et al. (2008)
- Yang et al. (2009)
- Liu et al. (1998)

<table>
<thead>
<tr>
<th>$Re$</th>
<th>$C_f^{LuLp-D/60}$</th>
<th>$C_f^{LuLp-D/120}$</th>
<th>$C_f^{QuQp-D/60}$</th>
<th>$C_f^{QuQp-D/120}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>1.34 ±0.011</td>
<td>1.34 ±0.011</td>
<td>1.29 ±0.011</td>
<td>1.29 ±0.011</td>
</tr>
<tr>
<td>40</td>
<td>1.34 ±0.009</td>
<td>1.34 ±0.009</td>
<td>1.29 ±0.009</td>
<td>1.29 ±0.009</td>
</tr>
</tbody>
</table>

Table 3: Drag and lift coefficient for flow past a stationary cylinder

References:
- Linnick and Fasel (2005)
- Choi et al. (2007)
- Mittal et al. (2008)
- Yang et al. (2009)
- Liu et al. (1998)
4. Oldroyd-B fluid flow

4.1. Convergence

The convergence test for the viscoelastic solver uses the same flow geometry and meshes as the Newtonian case, as shown in Figure 8. The boundary conditions for the velocity and pressure are the same as the Newtonian fluid flow case. Periodic boundary conditions are applied for the extra stress at the inlet and the outlet. Dirichlet boundary conditions with values from linear extrapolation are applied for the extra stress on the sidewalls. For consistency with the definition in the literature for viscoelastic fluid flow simulation, the Reynolds number is redefined with the cylinder radius as \( Re = \rho f u_\infty R/\eta \). The Weissenberg number, which characterizes the ratio of viscoelastic effect to viscous effect, is defined as \( Wi = \lambda u_\infty / R \). The convergence for an Oldroyd-B fluid flow is analysed when \( Re = 0.30 \) and \( Wi = 0.40 \).

We have investigated linear extrapolation and Neumann boundary condition \( \partial \sigma_{ij} / \partial n = 0 \). The schemes with linear extrapolation for the extra stress at forcing/ghost cells are marked with a ‘*’, e.g. QuLp*. The time step decreases quadratically with decreasing grid spacing (a fixed \( \Delta t / \Delta x^2 \)), and the maximum Courant number with the grid spacing \( \Delta x = D/300 \) is set as 0.04.

Figure 13 shows the normalised error of velocity components at steady state. When Neumann boundary conditions are applied for the extra stress on the surface of the cylinder, the velocity is first-order accurate with \( L_\infty \), for both the LuLp solver and the QuQp solver. When linear extrapolation is used for computing the extra stress at forcing cells, the error of the velocity components are smaller than that with Neumann boundary conditions. The order of accuracy is 1.4 for \( u_x \) and 1.3 for \( u_y \) with \( L_\infty \).

We previously showed that the velocity is above second-order accurate with \( L_\infty \) when quadratic fitting functions are used to locally reconstruct the velocity field. Therefore the first-order error should be caused by the treatment of the extra stress. Further simulations with bi-linear extrapolation or higher order extrapolations show strong oscillations of the extra stress, which make simulations diverge. Stabilized high order extrapolation scheme could be developed as part of future work to improve the accuracy of the viscoelastic solver.

4.2. Validation

We consider the benchmark case of a confined Oldroyd-B fluid flow past a stationary cylinder to verify the correctness of the code for viscoelastic fluid flows. Figure 14 illustrates the geometry of the flow. The cylinder is kept stationary with the centre at \((12R, 2R)\) in a rectangular channel of size \(L \times H\), with \(L = 14\), \(H = 2\) and \(R = 0.5\). This flow with a blockage of 50% has been widely used to validate numerical schemes for viscoelastic fluid flow simulations [32, 46, 56]. The front and rear part of the channel are sufficiently long, so that when periodic boundary conditions are applied between the inlet and outlet, the effect of the cyclic image of the cylinder is negligible. The flow is driven by a fixed pressure gradient \( \nabla p_0 \). Periodic boundary conditions are applied at
Figure 13: Normalised error of velocity components for simulating flow past a cylinder with different grid spacings when $Re = 0.30$ and $Wi = 0.40$. The slope of the dotted line is 1, the slope of solid line is 2, the slope of the dash line in (a) and (b) are 1.4 and 1.3 respectively. Neumann boundary conditions are applied for the extra stress on the surface of the cylinder for $LuLP$ and $QuQp$, while linear extrapolation is used for computing the extra stress at forcing cells for $QuLp^*$. 

Figure 14: Computational geometry for confined flow past a stationary cylinder in 2D.
the inlet ($x = -6$) and outlet ($x = 8$) for the velocity, the pressure and the extra stress. On the
sidewalls ($y = -1$ and $y = 1$), no-slip boundary conditions $u = (0, 0)$ are applied for the velocity,
Neumann boundary conditions $\partial p/\partial y = 0$ are applied for the pressure, and Dirichlet boundary
conditions with values from linear extrapolation are applied for the extra stress.

Theoretically, there is difficulty for time-dependent simulation of Newtonian or viscoelastic
fluid flow for small Reynolds number flow. With an explicit or semi-explicit discretization for the
viscous term, a restrictive viscous stability condition $\Delta t \leq \Delta x^2 Re/(2n)$, where $n$ is the number of
spacial dimensions, was proposed by Ferziger and Perić [57]. The time step is required to decrease
quadratically with the grid spacing (i.e. $\Delta t/\Delta x^2$ remains constant). If only steady-state simulation
results are of interest, the convective term in the momentum equation can be omitted. As a result,
the temporal term ($\rho \partial u/\partial t$) serves as an under-relaxation mechanism for solving the momentum
equation at steady-state. The time step is therefore only restricted by the CFL condition (the
CFL number remains constant if $\Delta t/\Delta x$ remains constant). Creeping flow with $Wi$ within the
range of 0 and 1 is investigated.

A comparison study between our simulation result and that in [46] demonstrates the accuracy
of our solver. Alves et al. [46] used a second-order finite volume scheme on a conformed mesh
in their work. They imposed a uniform flow at the entrance of the channel so their flow rate
is exact and the mean-flow-rate correlated Weissenberg number is exact. As a consequence, the
simulation needed a larger computational domain to decrease the boundary effects. The flow in
our simulation is driven by a constant pressure difference between the inlet and outlet instead, and
the mean-flow-rate is computed by post-processing of the simulation results on the finest mesh.

The simulations are mainly carried out on two 4-layer locally refined meshes, with minimal
grid spacing of $D/320$ or $D/640$ in the vicinity of the surface of the cylinder, with a total
number of 250624 or 894976 cells respectively. Three layers of grid on top of a uniform
Cartesian mesh are generated in domains $[-0.6, 0.6] \times [-0.6, 0.6] \cup [0.6, -0.1] \times [1.6, 0.1], [-0.8, 0.8] \times
[-0.8, 0.8] \cup [0.8, -0.2] \times [1.8, 0.2]$ and $[-1.0, 1.0] \times [1.0, 1.0] \cup [1.0, -0.4] \times [2.0, 0.4]$. An additional
mesh with minimal grid spacing of $D/1280$, with a total number of 1281028 cells, is generated by
a further layer of mesh near the surface of the cylinder (cells with cell-centres in a distance shorter
than $0.1R$ to the surface are further refined), for getting more accuracy in several tests. Alves et al.
used a half-domain body-fitted mesh with minimal grid spacings of $\Delta r = D/800$ and $\Delta s = D/240$,
where $\Delta r$ is the grid spacing in the radial direction, and $\Delta s$ is the grid spacing in the tangential
direction. They have showed that the prediction of drag coefficient is much more sensitive to $\Delta r$
[46]. A recent study of this benchmark case with full-domain meshes that are similar to [46] has
been carried out on OpenFOAM [58].

The drag coefficient is defined as $C_d = F_x/(\eta u_\infty)$, where $u_\infty$ is the mean flow rate. The
simulations have been carried out with four solvers LuLp, QuQp, LuLp*, QuLp*. The solvers
LuLp and QuQp apply Neumann boundary conditions for the extra stress on the surface of the
cylinder, while the solvers LuLp* and QuLp* use linear extrapolation to compute the extra stress at
forcing/ghost cells. The Courant number is between 0.3 and 0.4 for all simulations. 256 Lagrangian points are uniformly distributed on the surface of the cylinder for the measurement of the drag force.

Table 4 shows the drag coefficient at steady state. The value in the bracket is an indicator for the amplitude of fluctuation, which reflects the stability of the numerical scheme. The drag coefficients with the solvers $LuLp^*$ and $QuLp^*$ show significant fluctuations, comparing to that with the solvers $LuLp$ and $QuQp$. The amplitude of fluctuation grows as $Wi$ increases. The effect of mesh refinement on the amplitude of fluctuation is not clear in Table 4. The fluctuation of the drag makes it difficult to compare the accuracy between applying Neumann boundary condition and using linear extrapolation to compute the extra stress. When $Wi = 0.92$, the solver $LuLp^*$ and $QuLp^*$ diverge on the mesh with minimal grid spacing of $D/320$, even with much small time steps. Notice that the maximum amplitude of fluctuations is below 0.2% in Table 4.

When Neumann boundary conditions are applied for the extra stress on the surface, the drag predicted with the $QuQp$ solver is much more accurate than that with the $LuLp$ solver. The variation of drag with $\Delta x = D/640$ and $\Delta x = D/1280$ for the $QuQp$ solver is below 0.05% (
Figure 16: Profiles of normalised $x$-component of velocity along the centreline of the channel for flow past a cylinder.

Table 4: Drag coefficients for creeping flow past a stationary cylinder

<table>
<thead>
<tr>
<th>$Wi$</th>
<th>$D/320$</th>
<th>$LuLp^*$</th>
<th>$D/640$</th>
<th>$D/1280$</th>
<th>$D/320$</th>
<th>$LuLp^*$</th>
<th>$D/640$</th>
<th>$D/1280$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.20</td>
<td>126.41</td>
<td>126.58</td>
<td>126.68</td>
<td>126.64</td>
<td>126.70</td>
<td>120.74</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.30</td>
<td>122.84 (+0.02)</td>
<td>123.02 (+0.02)</td>
<td>123.14 (+0.02)</td>
<td>123.15 (+0.06)</td>
<td>123.21 (+0.06)</td>
<td>123.21 (+0.03)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.41</td>
<td>120.16 (+0.06)</td>
<td>120.32 (+0.03)</td>
<td>120.43 (+0.03)</td>
<td>120.55 (+0.06)</td>
<td>120.51 (+0.05)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.51</td>
<td>118.42 (+0.07)</td>
<td>118.54 (+0.04)</td>
<td>118.65 (+0.05)</td>
<td>118.78 (+0.10)</td>
<td>118.73 (+0.11)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.62</td>
<td>117.42 (+0.08)</td>
<td>117.58 (+0.07)</td>
<td>117.63 (+0.08)</td>
<td>117.80 (+0.12)</td>
<td>117.68 (+0.16)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.72</td>
<td>117.19 (+0.06)</td>
<td>117.19 (+0.15)</td>
<td>117.25 (+0.11)</td>
<td>117.48 (+0.14)</td>
<td>117.34 (+0.20)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.82</td>
<td>117.16 (+0.09)</td>
<td>117.34 (+0.17)</td>
<td>117.36 (+0.12)</td>
<td>117.71 (+0.15)</td>
<td>117.44 (+0.23)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.92</td>
<td>diverge</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1 the value in the bracket indicates the amplitude of oscillation due to numerical instability;
2 * indicates using linear extrapolation for the stress at forcing cells;
3 – represents untested.
Figure 17: Profiles of normalised $xx$-component and $xy$-component of the extra stress on the surface of the cylinder when Neumann boundary conditions are applied for $\sigma_p$. The figure is plotted in the clockwise direction against the length of the arc starting from the upstream point to the downstream point. Dashed lines represent the results with $\Delta x = D/320$, dots represent the results with $\Delta x = D/640$, and solid lines represent the results with $\Delta x = D/1280$. Black: $Wi = 0.20$, red: $Wi = 0.41$, green: $Wi = 0.62$, blue: $Wi = 0.82$. 
Figure 18: The maximum normalised $\sigma_{p,xx}/(\eta u_\infty/R)$ component of the extra stress on the surface of the cylinder with the $QuQp$ scheme.
Figure 19: Profiles of normalised $xx$-component and $xy$-component of the extra stress on the surface of the cylinder when using linear extrapolation to compute $\sigma_p$ at forcing/ghost cells. The figure is plotted in the clockwise direction against the length of the arc starting from the upstream point to the downstream point. Dashed lines represent the results with $\Delta x = D/320$, dots represent the results with $\Delta x = D/640$, and solid lines represent the results with $\Delta x = D/1280$. Black: $Wi = 0.20$, red: $Wi = 0.41$, green: $Wi = 0.62$, blue: $Wi = 0.82$. 
Table 4). Figure 15 demonstrates the accuracy of the solver $LuLp$ and $QuQp$ with the predicted drag coefficients in comparison with the result from Alves et al. [46]. The mean value is plotted in cases where fluctuations appear. A minimal drag coefficient is found between $Wi = 0.6$ and $Wi = 0.8$, which is consistent with the drag reduction results in literature (Figure 14 in [46]). A profile of normalised $x$-direction velocity with the $QuQp$ solver is plotted along the centreline of the channel, as shown in Figure 16. The front and rear profiles are symmetric with vanishing $Wi$. The increasing of $Wi$ expands the area where the velocity of a channel flow is influenced.

Additionally, the extra stress around the cylinder is investigated for further verification. Figure 17 and 19 compare the normalised profile of $\sigma_{p,xx}$ and $\sigma_{p,xy}$ on the surface of the cylinder. The figured is plotted starting from the upstream point to the downstream point against the arc length clockwise. The largest $\sigma_{p,xx}$ appears in the midway between the two ending points. The dashed lines, dots and solid lines represent the prediction with $\Delta x = D/320$, $D/640$ and $D/1280$ respectively.

The stress profile is smooth when Neumann boundary conditions are applied for $\sigma_p$ on the surface of the cylinder, as shown in Figure 17. The profiles with the $QuQp$ solver are found to be almost overlapping on that with the $LuLp$ solver. $\sigma_{p,xx}$ on the surface of the cylinder is shown bounded as the increment of $\sigma_{p,xx}$ is decreasing as $Wi$ increases. The mesh dependency of the stress is apparent. Figure 18 shows the maximum $\sigma_{p,xx}$ predicted with the $QuQp$ scheme. $\sigma_{p,xx}$ are consistent with that reported in [46] with M120, which has a grid spacing of $D/800$ in the radial direction.

The mesh-dependency of the extra stress on the boundary is attenuated by using linear extrapolation, as shown in Figure 19. The $LuLp^*$ solver and the $QuLp^*$ solver give better prediction of $xy$-component of the extra stress. However, a better prediction of the $xx$-component is with linear extrapolation limited at relatively low $Wi (\leq 0.62)$. Wiggles are shown near the peak point of the $xx$-component as $Wi$ increases. The $xx$-component of the extra stress at the rear of the cylinder along the centreline $y = 0$ is shown in Figure 20. The stress at the rear was reported to be unbounded [46]. Similar to the results observed by Alves et al. [46], the stress profile at the rear of the cylinder is not sensitive to the treatment at the surface of the cylinder. Simulations with a grid spacing of $D/640$ can predict the stress accurately up to $Wi = 0.62$ in this work. The accuracy can be further improved using a Cartesian mesh refinement algorithm.

4.3. Flow past a pair of cylinders

Flow past a pair of cylinders lying along the main flow direction of an Oldroyd-B fluid flow is investigated with the $QuQp$ solver. The computational geometry (Figure 21) and boundary conditions are the same as that in the single cylinder case in Section 4.2 except a longer channel of size of $[-12, 16] \times [-1, 1]$. The two cylinders are separated with a gap of $R/2$ or $R$. The cylinder close to the inlet is positioned at $(-2, 0)$ (named as the upstream cylinder), the other is positioned at $(-0.75, 0)$ when the gap is $R/2$, or $(-0.5, 0)$ when the gap is $R$ (named as the downstream
Figure 20: Profiles of $x_x$-component of the extra stress along the centreline at the rear of the cylinder. Dashed lines represent the results with $\Delta x = D/320$, and dots represent the results with $\Delta x = D/640$.

Figure 21: Computational geometry for confined flow past a pair of stationary cylinders in 2D.
Table 5: Drag coefficients for flow past a pair of cylinders with separations of R/2 and R.

<table>
<thead>
<tr>
<th>Wi</th>
<th>( \Delta x = D/320 )</th>
<th>( \Delta x = D/640 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Delta x = D/320 )</td>
<td>( \Delta x = D/640 )</td>
<td></td>
</tr>
<tr>
<td>0.21</td>
<td>120.74</td>
<td>120.80</td>
</tr>
<tr>
<td>0.31</td>
<td>118.51</td>
<td>118.95</td>
</tr>
<tr>
<td>0.42</td>
<td>117.19</td>
<td>116.92</td>
</tr>
<tr>
<td>0.52</td>
<td>116.68</td>
<td>114.78</td>
</tr>
<tr>
<td>0.63</td>
<td>116.82</td>
<td>112.71</td>
</tr>
<tr>
<td>0.74</td>
<td>117.45</td>
<td>110.81</td>
</tr>
<tr>
<td>0.85</td>
<td>118.49</td>
<td>109.12</td>
</tr>
<tr>
<td>0.95</td>
<td>119.90 (+0.01)</td>
<td>107.70 (+0.03)</td>
</tr>
</tbody>
</table>

Table 5: Drag coefficients for flow past a pair of cylinders with separations of R/2 and R.

Separation: R/2

<table>
<thead>
<tr>
<th>Wi</th>
<th>( \Delta x = D/320 )</th>
<th>( \Delta x = D/640 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Delta x = D/320 )</td>
<td>( \Delta x = D/640 )</td>
<td></td>
</tr>
<tr>
<td>0.20</td>
<td>123.75</td>
<td>123.91</td>
</tr>
<tr>
<td>0.30</td>
<td>120.95</td>
<td>122.28</td>
</tr>
<tr>
<td>0.41</td>
<td>118.95</td>
<td>121.09</td>
</tr>
<tr>
<td>0.51</td>
<td>117.74</td>
<td>120.02</td>
</tr>
<tr>
<td>0.62</td>
<td>117.14</td>
<td>119.02</td>
</tr>
<tr>
<td>0.72</td>
<td>117.05</td>
<td>118.12</td>
</tr>
<tr>
<td>0.83</td>
<td>117.35</td>
<td>117.39</td>
</tr>
<tr>
<td>0.94</td>
<td>118.02 (+0.01)</td>
<td>116.89</td>
</tr>
</tbody>
</table>

cylinder). Two 4-layer meshes with minimal grid spacings of \( D/320 \) and \( D/640 \) are generated for the simulation. Three square layers cover areas with side length of \( 2R + 0.2, 2R + 0.6 \) and \( 2R + 1.0 \) from top to down. The maximum Courant number for all simulations is between 0.3 and 0.4.

Figure 22 shows the streamline with coloured contours of normalised y-component velocity from three simulations: \( Wi = 0.21 \) or \( Wi = 0.63 \) at a separation of R/2, \( Wi = 0.62 \) at a separation of R. The streamline with separation R/2 and R has a substantially different pattern in the wake between the two cylinders, while the effect of different Weissenberg number is not obvious. As shown in Figure 23, for small \( Wi \), the drag on the downstream cylinder (solid symbols) is slightly stronger than that on the upstream one (empty symbols). The drag on the upstream cylinder becomes larger than that on the downstream one as \( Wi \) increases. When the separation is R/2, the minimal drag on the upstream cylinder is observed when \( Wi \) is between 0.42 and 0.63 (circles). When the separation is R, the minimal drag on the upstream cylinder is observed when \( Wi \) is between 0.62 and 0.83 (triangles). The drag on the downstream cylinder decreases almost linearly with increasing \( Wi \). Beyond the range of \( Wi \) tested, the drag coefficient of the downstream cylinder is still decreasing as \( Wi \) increases. As the viscoelastic effect becomes stronger, the drag on the upstream cylinder are much larger than that of the downstream cylinder, with a difference of about 11% for \( Wi = 0.95 \) when the separation is R/2. Figure 23 illustrates the variation of drag coefficients of the cylinder pair with respect to the Weissenberg number. Simulations show that the peak value of \( \sigma_{p,xx}, \sigma_{p,xy} \) on the downstream cylinder is always larger than that of the upstream one.
(a) $Wi = 0.21$, separation: $R/2$

(b) $Wi = 0.63$, separation: $R/2$

(c) $Wi = 0.62$, separation: $R$

Figure 22: Streamlines of flow past a pair of cylinders with contours representing the normalised $y$-component (normalised by $u_\infty$) of the velocity field.
Figure 23: Drag coefficients of creeping Oldroyd-B fluid flow past a single/pair of cylinders with $\Delta x = D/640$. (a) The empty and solid circles represent drag coefficients for the upstream and downstream cylinders respectively when the separation is $R/2$. (b) The empty and solid triangles represent drag coefficients for the upstream and downstream cylinder respectively when the separation is $R$. (c) The empty squares represent drag coefficients for flow past a single cylinder.
5. Comparison with a smoothed-interface immersed boundary method

5.1. Accuracy

In the discussion of a Cartesian grid method, there remains a question that whether the smoothed-interface immersed boundary method (SIIB) [32] or the proposed sharp-interface Cartesian grid method (SICG) is more suitable for simulating viscoelastic fluid flow with rigid immersed boundaries. For simulating Newtonian fluid flow, the former is first-order accurate in space, as a smoothed delta function [59] is used for exchanging information between the Lagrangian and Eulerian grid, while the latter is above second-order accurate, due to a linear, bi-linear or quadratic fitting function for local reconstruction of the boundary field. Nevertheless, for simulating a viscoelastic fluid flow, both methods are first-order accurate in space, as shown in Section 4.1 with sufficient evidence. In addition to the SICG method, we have also implemented a smoothed-interface immersed boundary method on OpenFOAM, with an improved Rhie-Chow interpolation and similar discretisation schemes in this paper, i.e., Adam-Bashforth scheme for convective terms, Crank-Nicholson scheme for diffusion terms. The simulation of the solver is briefly discussed here as a comparison to that of the SICG solver.

Figure 24 shows the prediction of drag coefficients, in the simulation of a creeping flow for an Oldroyd-B fluid flow past a stationary cylinder, using the same computing settings as in Section 4.2. The same meshes and time steps are used for both solvers. The error is far below 1% in both simulations. With the SIIB method, the drag is slightly over-predicted, in consistency with literatures [32, 33]. For the evaluation of viscoelastic stress on the surface of the cylinder, we consider two approaches to post-process the data from the SIIB method. In the first approach, the extra stress at a Lagrangian point is interpolated from the nearest cell-centre and three face vertices (the ‘cellPoint’ scheme) using a tool ‘sample’ in OpenFOAM. In the second approach, the extra stress is computed with a local reconstruction as in the SICG method. Figure 25 compares the normalised \( \sigma_{xx} \)-component and \( \sigma_{xy} \)-component of the stress on the mesh with a minimal grid spacing of \( D/640 \). The SICG solver gives a much more accurate prediction of the extra stress on the surface.

Since the velocity and pressure profiles are smeared out at the solid-fluid interface with a SIIB method, for simulating an oscillating cylinder in free stream, the resulting lift and drag force profiles are much smoother than their SICG counterparts. The benchmark case of Newtonian fluid flow past an oscillating cylinder is simulated with both methods. The cylinder is moving in the direction perpendicular to the main flow direction: \( y(t) = y_0 + A \sin(2\pi f_e t) \), where \( y(t) \) is the \( y \)-coordinate of the cylinder centre, \( y_0 \) is the initial \( y \)-coordinate, \( A \) is the amplitude, \( f_e \) is the oscillation frequency, \( A = 0.2D \) and \( f_e = 0.52 \). We choose the same Reynolds number \( Re = 185 \) as in [60]. \( f_e \) is 0.8 times the natural shedding frequency \( f_s \) at \( Re = 185 \). The Courant number for all simulations is kept around 0.4. Figure 26 illustrates the time evolution of drag and lift coefficients in our simulations. The smoothed delta function used for the SIIB method is IB4 [59].
An apparent oscillation is observed in the drag profile predicted with the SICG method. This oscillation arises from the discontinuity of the sudden change of a computational unit from solid to fluid or vice versa between two succeeding time steps [61]. Although the field-extension technique [40] for attenuating the oscillation has been implemented, the oscillation is still strong comparing with the prediction from a SIIB method. The nonphysical oscillation can be reduced by mesh refinement.

5.2. Efficiency

The efficiency of a numerical solver is significantly important when the solver is extended for simulating flow with multiple immersed particles in parallel environment. We carry out a strong scaling test (fix the mesh size but change the number of processing cores). There is a significant difference between the SIIB method and the SICG method. The former solves the governing equations in the internal part of the solid while the latter does not. A performance study is carried out by running simulations on a computing node with 64 cores, i.e. four AMD Opteron(TM) 6272 processors, each including 16 cores with main frequency of 2.1 GHz. In this test, PCG solver in conjunction with the DIC preconditioner is used to solve the discretised
Figure 25: Profiles of normalised components of the viscoelastic stress on the surface of the cylinder for $Wi = 0.62$. The figure is plotted in the clockwise direction against the length of the arc starting from the upstream point of the cylinder.

Figure 26: Drag and lift coefficients of Newtonian fluid flow past an oscillatory cylinder.
Figure 27: Running time on a 64-core computing node. The oscillating case considers a Newtonian fluid flow when $Re = 185$, while the stationary case considers an Oldroyd-B fluid flow when $Re = 0$ and $Wi = 0.62$.

momentum equation, $BiCG$ solver in conjunction with the $DILU$ preconditioner is used to solve the discretised viscoelastic constitutive equation. The pressure equation are solved using the $PCG$ solver in conjunction with the $DIC$ preconditioner. The convergence tolerance for all variables is $10^{-8}$. The performances of the two problems in Section 5.1 is investigated. The performance test uses 4-layer meshes with 214858 cells for the oscillating cylinder case, 223744 cells for the stationary cylinder case. We run the simulation of the oscillating cylinder in free stream with $Re = 185$ for 5000 time steps, and the Oldroyd-B fluid flow past a stationary cylinder with $Wi = 0.62$ and $Re = 0$ for 20000 time steps. The Courant number for all simulations is kept around 0.4. The number of velocity iterations and pressure iterations for both methods are 2 and 3, respectively.

Figure 27 illustrates the running time on 8, 16, 32, 64 cores. The minimal grid spacing for the oscillating case and the stationary case are $D/80$ and $D/320$ respectively. The results show that the running times of the two solvers for simulating the oscillating cylinder in a free stream are similar, when the number of processing cores is the same. However, for the stationary case, the
simulation time with the SICG method is almost half of that with the SIIB method. With further investigation into solving the linear equations, we find that the major time difference is caused by the number of iterations in solving the pressure equations. The number of iterations for solving the Poisson equation with the SICG method is almost half of that with the SIIB method. Thus it is due to the involvement of the equations for the internal part of the cylinder with the SIIB method. The number of normal fluid cells for the oscillating case and the stationary case are 211863/214858, 142384/223744, respectively, which explains the difference in performance between the oscillating case and the stationary case. Considering the speedup of both methods, the running time for simulations on 8 cores is halved when the number of cores is doubled to 16. The same reduction of running time is observed when the number of cores is doubled to 32. This demonstrates satisfactory level of parallelism of our implementation for both methods in OpenFOAM. The simulation on 64 cores gains nearly zero speedup, comparing with the simulation on 32 cores, which suggests that the inter-processor communication cost has become dominant. The number of mesh cells on each core when 32 cores are used is around 6700. It is also worth to notice that the influence of the number of Lagrangian points (SIIB) or the number of forcing cells (SICG) on the parallel performance is insignificant in the present 2D simulation.

6. Conclusion

A new numerical scheme for simulations of viscoelastic fluid flow with boundaries of complex geometry on the Cartesian mesh is proposed in this paper. For the simulation of a Newtonian fluid flow, the solver shows second-order accuracy of the velocity when linear/bilinear fitting functions are used, and approaches third-order of accuracy when quadratic fitting functions are used. For the simulation of a viscoelastic fluid flow, the velocity is first-order accurate when applying Neumann boundary condition for the extra stress, and can be slightly improved to order 1.3 ~ 1.4 when linear extrapolation are used for computing the extra stress at forcing/ghost cells. However, linear extrapolation is found to be unstable for high Weissenberg number flow. The correctness of the code is validated with the benchmark case, flow past a stationary cylinder. The measured drag coefficients and stress profiles are comparable with that from a finite volume method on a body-fitted mesh using a similar mesh density. A further study of the Oldroyd-B fluid flow past a pair of cylinders demonstrates the advantage of the current Cartesian mesh method in comparison to conventional body-fitted mesh methods. The comparison between the sharp-interface Cartesian grid method and the smoothed-interface immersed boundary method shows that the former is better at resolving the surface stresses, and is more efficient because it avoids solving linear equations in the solid part.

In the simulation of an Oldroyd-B fluid flow with $Wi < 1$, the correct prediction of drag coefficients and extra stress on the immersed boundary shows the effectiveness of the proposed method. The current solver could be further improved by using available stability techniques such
as the discrete elastic-viscous stress splitting (DEVSS) technique\cite{62}, matrix logarithms technique\cite{63, 64}, for simulations of high Weissenberg number viscoelastic fluid flow.

**Acknowledgement**

Wei Yi’s PhD studentship is funded by the University of Manchester and the Chinese Scholarship Council. The authors thank Ying Chen and Alfredo Lanzaro for suggestions in writing the paper. The authors thank N8 HPC facility in Leeds and the National supercomputer centre in Guangzhou for providing computing resources. The authors thank the OpenFOAM team in National University of Defense Technology for helpful discussions. The authors also thank the referees for valuable comments on the submitted manuscript.

**References**


