Theor. Comput. Fluid Dyn. (2023) 37:661–680
https://doi.org/10.1007/s00162-023-00674-x

ORIGINAL ARTICLE

Ivonne Leonor Medina Lino · Mariana Carrasco-Teja · Ian Frigaard

GPU computing of yield stress fluid flows in narrow gaps

Received: 5 December 2022 / Accepted: 15 September 2023 / Published online: 19 October 2023

Abstract We present a Graphic Processing Units (GPU) implementation of non-Newtonian Hele-Shaw flow that models the displacement of Herschel-Bulkley fluids along narrow eccentric annuli. This flow is characteristic of many long-thin flows that require extensive calculation due to an inherent nonlinearity in the constitutive law. A common method of dealing with such flows is via an augmented Lagrangian algorithm, which is often painfully slow. Here we show that such algorithms, although involving slow iterations, can often be accelerated via parallel implementation on GPUs. Indeed, such algorithms explicitly solve the nonlinear aspects only locally on each mesh cell (or node), which makes them ideal candidates for GPUs. Combined with other advances, the optimized GPU implementation takes $\approx 2.5\%$ of the time of the original algorithm.

Keywords Yield stress fluid · Algorithm improvement · Parallelization · Non-Newtonian fluid

1 Introduction

The aim of this paper is the study the displacement of yield stress fluids in narrow annular gaps, developing an implementation in parallel using Graphic Processing Unit (GPU) technology. Yield stress fluids are non-Newtonian fluids that occur commonly in natural and industrial settings [1]. These fluids do not deform unless the deviatoric stress exceeds a particular threshold, the yield stress. Often, yielding behaviour is accompanied by other rheological behaviours. However, here we consider only simple yield stress fluids [2], as typified by rheological models such as the Bingham, Casson and Herschel-Bulkley fluids.

A yield stress rheology introduces nonlinearity in the deviatoric stress tensor and typically a singularity in the effective viscosity. One often wants to resolve the yielding behaviour exactly. Methods for doing so were first pioneered by Glowinski in the 1970 s [3], as outlined in the reviews [4,5]. Although there have been many advances in meshing/discretization, implementations and applications (e.g. [6–8]), the basic method for computing the exact yield stress flow retains its original framework, as follows. First, the flow problem on each time step is rewritten as a velocity ($\mathbf{v}$) minimization. The yield stress means that the minimization is non-differentiable for flows that contain unyielded regions. To work around this, the strain rate tensor is replaced by a new variable ($\mathbf{q}$). The constraint arising from this relaxation is then enforced using a Lagrange multiplier (λ), so that the original convex minimization is now a Lagrangian saddle point problem. For numerical stability the Lagrangian is augmented with a penalty term that vanishes at the solution. The augmented Lagrangian problem is solved using an iterative Uzawa algorithm, which has 3 steps, repeated iteratively until convergence. Step 1

Communicated by Teodor Burghelea.

I. Frigaard (✉)· I. L. Medina Lino · M. Carrasco-Teja
Department of Mechanical Engineering, University of British Columbia, Vancouver, BC V6T 1Z4, Canada
E-mail: frigaard@mail.ubc.ca

I. L. Medina Lino
E-mail: ivonneleonor1@gmail.com

is an elliptic linear problem for $\mathbf{v}$, solved over the whole flow domain. The second step is a nonlinear problem for $\mathbf{q}$, solved locally on each element. The third step is a simple projection/update of λ.

The inherent interest for GPU parallelization comes in the structure of the basic algorithm described above. In solving a problem on N cells, the dominant cost might be thought to arise from the linear algebra problem of step 1. However, this same linear problem is solved repeatedly within the Uzawa iteration. For a moderate sized problem, one can use e.g. a QR decomposition, after which the step 1 problem requires $O(N)$ operations at each iteration. For very large problems the linear step 1 might require an iterative solution but good initial guesses and preconditioners are likely available to accelerate. Thus, although step 1 can be a significant cost timewise, it is a standard problem and not the focus here.

In step 2 the problem is more interesting. In a very simple application such as a Bingham fluid, this nonlinear problem is a quadratic equation solved algebraically. In such cases step 2 is computationally trivial. However, for more complex rheological laws and/or for modeling techniques common in long-thin geometries, as we treat here, the local nonlinear problem in step 2 must itself be solved numerically. We then enter a regime where step 2 is (by far) the dominant cost computationally. However, the beauty of the underlying mathematical framework is that step 2 can be solved locally on each cell, making this the perfect problem for a highly parallel method such as GPU programming, although the ideal GPU efficiency is likely still limited by data transfer to/from the CPU. In this paper, we indeed demonstrate the efficacy of such an implementation.

The class of flows that we consider are flows with small aspect ratios. These occur widely, in both thin film flows and Hele-Shaw cell applications. Ours is a version of the latter and it is motivated by the industrial process of primary cementing, used to complete oil & gas wells prior to production [9]. In the process a sequence of fluids (mainly yield stress fluids) are pumped upwards along a narrow eccentric annulus formed by the borehole and the outside of a steel casing inserted into the well. In practice the geometry varies due to changes in eccentricity, that are a result of both geomechanical strains on the borehole and due to geological and drilling induced defects (e.g. washouts). Thus, we have both a complex geometry and a complex rheology to contend with.

To illustrate that quite different applications that lead to similar mathematical structure, see for example the food processing flows modelled in [10], or some of the ice-sheet flows reviewed in [11], where analogous "yielding" comes from frictional behaviour at the base of the ice sheet. Indeed, disregarding yield stress applications, it is a common modelling technique to use asymptotic methods to reduce the momentum balance to a simplified shear flow, so that more complex physical features can be added and resolved. This can include multi-phase scenarios, heat transfer, species transport etc., as well as certain boundary layer flows. The main point is that the *narrow* dimension has a physical model that should be solved (numerically) prior to being able to compute the flow in the other 2 dimensions.

1.1 Primary cementing displacement flows

An example of the above is given by our work on cementing displacement flows. The model considered here was derived by [12], and we outline it below. It assumes the flow is laminar, uses scaling arguments to reduce the Navier–Stokes equations to a local shear flow, and then averages across the narrow annular gap to derive an elliptic equation for the stream function (essentially a Hele-Shaw approach). The mathematical foundation of this model was developed by [13], who put the model into its variational framework and established key properties of the functions involved. Ideal cementing flows involve a steady state displacement, in which a displacement front advances along the well at a uniform pumping speed. That such solutions exist and how to predict them and incorporate them into process designs for vertical wells was studied by [13, 14]. Using the same variational formulation, [15] developed an augmented Lagrangian algorithm for the cementing displacement flow model. This numerical solution is used to study the effects of increasing eccentricity, yield stress and how the interface can become unsteady.

The early 2000's also saw the rise of horizontal wells as a dominant architecture for oil and gas wells in many parts of the world. The change in well orientation causes the annulus to be more eccentric and buoyancy effects, due to fluid density differences, promote stratification. Carrasco-Teja et al. developed analyses, based on the Bittleston et al. model [12], specifically designed for horizontal well cementing [16]. They were able to show that steady displacements may still arise in horizontal wells, even with significant density differences, and that the axial length of the interface would grow as the density difference increased [16].

Although these results are reassuring, in many situations a long slumping interface is simply not desirable, e.g. because such stratified interfaces tend to destabilize. An industry practice developed to aid horizontal well displacements has been to rotate the casing during displacement, with the idea that the induced Couette component will drag the fluids around in the azimuthal direction. This was first modelled by [17,18]. Although it is relatively easy to model for Newtonian fluids [17], for non-Newtonian fluids the nonlinearity results in a local closure problem that must be resolved as a 2D flow, i.e. the Poiseuille and Couette components of the flow do not decouple. When yield stress fluid are present this local closure problem again is computed using an augmented Lagrangian method, becoming painfully slow to compute.

More recently, this modeling approach has been extended to more complex cementing scenarios. For example, turbulent, transitional and mixed regimes are modelled [19–22]. Such models must cover all flow regimes, which means that the yield stress dominated behaviour at slow flow rates is augmented by other nonlinear behaviours at high flow rates, eventually becoming turbulent. The augmented Lagrangian algorithm is again used computationally. Both the rotating casing scenarios and dealing with mixed flow regimes require progressively expensive models for the local closure problem, i.e. due to transitions between regimes or other parametric variations. In all cases where the local closure problem is expensive, a GPU implementation is progressively attractive.

This article is divided as follows. In Sect. 2 we briefly describe the theoretical framework of the annular displacement flow model. In the next section we describe the software and hardware that we used to parallelize our model and the criteria that we used to choose them. The fourth section is related to the modification of the original algorithm that we developed to optimize the code, as well as the new parallel code. We explain in detail this mathematical modification. We also show the pseudocode with its three variants. We included a subsection with convergence analysis. The fifth section is devoted to show our results in serial and in parallel and the corresponding comparison. We included error analysis and physically relevant examples to validate the algorithm's change and the parallelization.

2 Model framework

In the primary cementing operation, 100's or 1000's of metres of an oil/gas wells are cemented by pumping fluids along a narrow, eccentric annular gap of typical width of 2–3 cm; see Fig. 1a. This enormous disparity in length-scales makes solution of the full Navier–Stokes equations unfeasible on the scale of the full wellbore. Primarily for this reason, 2D gap-averaged approaches have evolved following [12]. The approach uses scaling arguments to reduce the Navier–Stokes equations locally to a shear flow. The main consideration is that

$$\text{annular gap} \ll \text{annular circumference} \ll \text{axial length-scale}, \tag{1}$$

together with some requirements that the geometry vary slowly along the well. The reduced equations are then averaged across the annular gap, in the same way that a Hele-Shaw model is derived.

The geometry of the model is shown in Fig. 1, and the parameters are listed in Table 1. Dimensional parameters are denoted by having a "hat" accent $(\hat{\cdot})$ throughout the paper, and dimensionless parameters without. The dimensionless spatial coordinates are $(\phi, \xi) \in (0, 1) \times (0, Z)$ where $\phi = \theta/\pi$ is the azimuthal coordinate and ξ measures axial distance, upwards from the bottom of the well. For simplicity, we assume that the flow is symmetric azimuthally and therefore only half of the annulus is considered.
The annular gap half-width $H(\phi, \xi)$ is given by:

$$H(\phi, \xi) = \bar{H}(\xi)[1 + e(\xi) cos \pi \xi] \tag{2}$$

where $e(\xi) \in [0, 1]$ is the eccentricity. When the annulus is concentric $e(\xi) = 0$ and when the casing contacts the wellbore wall $e(\xi) = 1$; see Fig. 2. The inclination from vertical is denoted by $\beta(\xi)$, and the mean annulus radius at each depth is denoted by $r_a(\xi)$. The gap-averaged velocities in the azimuthal and the axial directions are respectively $\bar{v}$ and $\bar{w}$. Since the flows are considered incompressible, and assuming no-slip conditions at the wall, these satisfy:

$$\frac{\partial}{\partial \phi}[H\bar{v}] + \frac{\partial}{\partial \xi}[H r_a \bar{w}] = 0. \tag{3}$$

As the gap-averaged flow is two-dimensional, the velocity components are best represented via a stream function, automatically satisfying the incompressibility condition above.

$$\frac{\partial \Psi}{\partial \phi} = H r_a \bar{w}, \quad \frac{\partial \Psi}{\partial \xi} = -H\bar{v}. \tag{4}$$

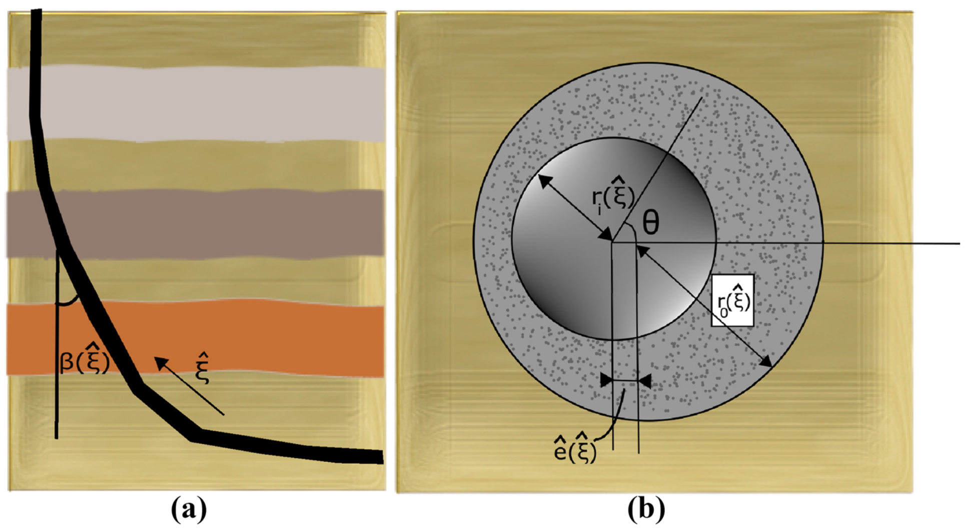

Fig. 1 **a** Schematic of the transversal view of a well. **b** Schematic of a cross-section of the well that describes the eccentric annular geometry

Table 1 List of parameters

Parameter	Symbol
Eccentricity	$e \in [0, 1)$
Angle of inclination	β
Outer radii of the annulus	$\hat{r}_0$ (m)
Inner radii of the annulus	$\hat{r}_i$ (m)
Mean velocity scale	$\hat{w}^*$ (m/s)
Length of the casing	$\hat{l}$ (m)
Imposed flow rate	$\hat{Q} = \pi(\hat{r}_0^2 - \hat{r}_i^2)\hat{w}^*$ (m^3/s)
Time	$\hat{t} = \frac{\pi \hat{r}_a^*}{\hat{w}^*} t$ (s)
Fluid j power-law index	n_j
Fluid yield stress	$\tau_{Y,j} = \hat{\tau}_{Y,j}/\hat{\tau}*$ (Pa)
Viscous stress scale	$\hat{\tau}* = max_{k=1,2}\{\hat{\tau}_{k,Y} + \hat{\kappa}_k[\hat{\dot{\gamma}}*]^{n_k}\}$ (Pa)
Strain rate	$\hat{\dot{\gamma}}^* = 2\hat{w}^*/(\hat{r}_0 - \hat{r}_i)$ (s^{-1})
Fluid j consistency	$\kappa_j = \hat{\kappa}_j[\hat{\dot{\gamma}}]^{n_k}/\hat{\tau}*$
Fluid j density	$\rho_j = \hat{\rho}_j/max_{k=1,2}\{\hat{\rho}_j\}$
Dimensional density	$\hat{\rho}_j$ (kg/m^3)
Stokes number	$St^* = [2\hat{\tau}^*]/[max_{k=1,2}\{\hat{\rho}_k\}\hat{g}(\hat{r}_0 - \hat{r}_i)]$

Determining a field equation for Ψ is addressed in detail in [12]. It starts with the observation that the streamlines are parallel to the modified pressure gradient of the flow, i.e. the pressure gradient minus the mean static pressure gradient. This allows for a local re-orientation of the coordinates along the streamline. In these local coordinates the flow is basically a Poiseuille flow through a plane channel. Cementing fluids are characterized as Herschel-Bulkley fluids, with 3 parameters: yield stress τ_Y, consistency κ and power-law index n. For a plane Poiseuille flow the relationship between the modified pressure gradient G, and the area flow rate $|\nabla_a \Psi|$, is given by:

$$|\nabla_a \Psi| = \begin{cases} 0, & \chi \leqslant 0, \\ \dfrac{H^{2+1/n}}{\kappa^{1/n}(2+1/n} \dfrac{\chi^{1+1/n}}{(\chi + \tau_Y/H)^2}\left[\chi + \dfrac{(2+1/n)\tau_Y}{(1+1/n)H}\right], & \chi > 0. \end{cases} \qquad (5)$$

The variable χ measures the excess of the pressure gradient G over the critical value τ_Y/H at which we transition from flow to no-flow. The operator ∇_a is defined as:

$$\nabla_a = \left(\frac{1}{r_a}\frac{\partial}{\partial \phi}, \frac{\partial}{\partial \xi}\right).$$

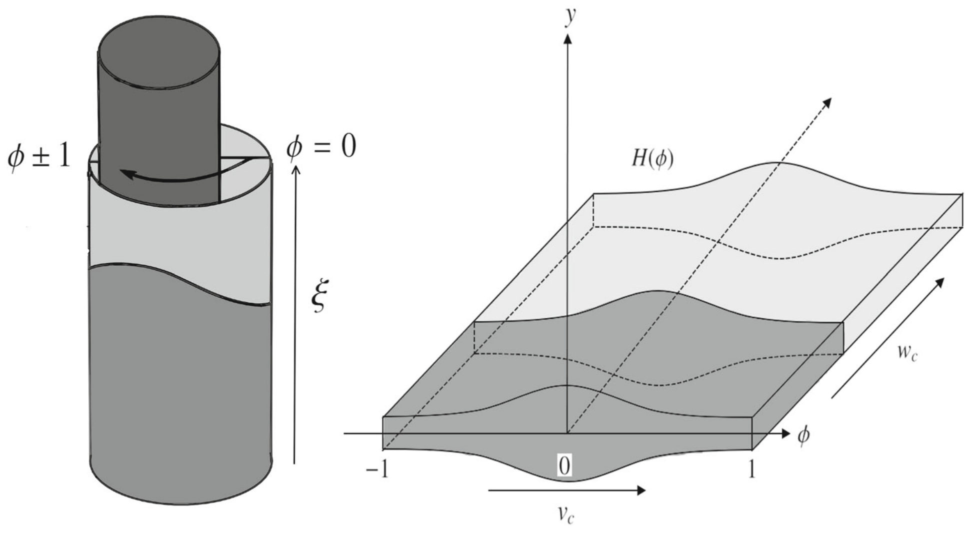

Fig. 2 **a** Schematic of the narrow eccentric annulus (**b**) Annulus unwrapped as a Hele-Shaw cell

To understand (5), note that GH gives the wall shear stress in the annular gap. If $GH \leq \tau_Y$ then the fluid does not deform and consequently there is no flow, i.e. $|\nabla_a \Psi| = 0$. When $\chi = G - \tau_Y/H > 0$, there is a flow and, due to the nonlinearity of the fluid, the flow rate is no longer linearly related to the pressure gradient. This nonlinearity is described by the second part of (5), which is an implicit equation for $\chi(|\nabla_a \Psi|)$. The properties of $\chi(|\nabla_a \Psi|)$ determine the function space in which a weak solution may be found, as discussed in [13]. Note that when the rheological parameters are: $n = 1$, $\tau_Y = 0$, we recover a Newtonian fluid behaviour.

The flow law (5) can be used to either define an equation for the pressure or for the stream function. As the pressure gradient is not determined in regions where $|\nabla_a \Psi| = 0$, the latter is the preferred formulation. Indeed determining regions where $|\nabla_a \Psi| = 0$ has practical relevance for the cementing process, often indicating defective cementing. The stream function satisfies

$$\nabla_a \cdot \mathbf{S} = -f, \tag{6}$$

where

$$\begin{cases} \mathbf{S} = \left[\dfrac{r_a \chi(|\nabla_a \psi|)}{|\nabla_a \psi|} + \dfrac{r_a \tau_Y}{H|\nabla_a \psi|}\right] \nabla_a \psi, & \Leftrightarrow |\mathbf{S}| > \dfrac{r_a \tau_Y}{H}, \\ |\nabla_a \psi| = 0, & \Leftrightarrow |\mathbf{S}| \leq \dfrac{r_a \tau_Y}{H}. \end{cases} \tag{7}$$

The term f represents the influence of buoyancy forces:

$$f = \nabla_a \cdot \left(\frac{r_a \rho(\bar{\mathbf{c}}) \cos \beta}{St^*}, \frac{r_a \rho(\bar{\mathbf{c}}) \sin \beta \sin \pi \phi}{St^*} \right) = \nabla_a \cdot \tilde{\mathbf{f}}, \tag{8}$$

where St^* is the Stokes number, representing the balance of viscous to buoyancy stresses, as defined in Table 1; see also [12]. This term arises from cross-differentiating to eliminate the pressure, which results in taking derivatives of the static pressure gradient components.

Equation (6) is solved for Ψ at each time, in order to derive the velocity field. The flows considered are however displacement flows. To model the different fluids a concentration vector $\bar{\mathbf{c}}$ is introduced, representing the gap-averaged volume fraction of each fluid. Each individual fluid component $\bar{c}_k$ satisfies

$$\frac{\partial}{\partial t}[H r_a \bar{c}_k] + \frac{\partial}{\partial \phi}[H \bar{v} \bar{c}_k] + \frac{\partial}{\partial \xi}[H r_a \bar{w} \bar{c}_k] = 0, \quad k = 1, 2, .. \tag{9}$$

Here we assume only 2 fluids, with fluid 2 displacing fluid 1. Conservation of volume means that $\bar{c}_1 + \bar{c}_2 = 1$. Note that in some formulations, the above equation is supplemented by a diffusion term on the right-hand side which can represent both molecular diffusivity and dispersion effects.

The fluid concentrations advance in time. Time dependency may also enter in the boundary conditions, e.g. the amount of each fluid that is pumped into the annulus. Coupling of $\bar{\mathbf{c}}$ and Ψ is two-way. The concentrations are used to define the (mixture of) rheological properties τ_Y, κ and n, from those of the individual fluids. The density $\rho(\bar{\mathbf{c}})$ also depends on the fluids, and hence the buoyancy comes from gradients in $\rho(\bar{\mathbf{c}})$. In the other direction, the fluid velocities determine transport of $\bar{\mathbf{c}}$.

We impose the boundary conditions of symmetry at the wide and narrow sides for the concentration equation:

$$\frac{\partial \bar{c_k}}{\partial \phi} = 0, \quad \phi = 0, 1. \tag{10}$$

At the bottom of the annulus, inflow concentrations are specified and at the top an outflow condition is used. For the elliptic second-order Eq. (6), on the wide side of the annulus we impose:

$$\Psi(0, \xi, t) = 0, \tag{11}$$

and on the narrow side:

$$\Psi(1, \xi, t) = Q(t), \tag{12}$$

where $Q(t)$ is an appropriately scaled flow rate. In the axial direction we impose Dirichlet conditions given by:

$$\Psi(\phi, 0, t) = \Psi_0(\phi, t), \quad \Psi(\phi, Z, t) = \Psi_Z(\phi, t), \tag{13}$$

with Ψ_0 and Ψ_Z specified. These boundary conditions can vary for different operational settings, or for slight variants of the model. For example, when the full annulus is modelled, ϕ is extended to [0, 2] and all variables are made periodic in ϕ.

3 Computational overview

We now describe what is the typical computational challenge of simulating a primary cementing displacement flow. The model (4)–(13) is solved over a spatial domain of size $1 \times Z$. Since oil and gas wells tend to be very long, $Z \sim 5 \times 10^2 - 10^4$, and even $Z \sim 10^2$ for a laboratory experiment; see Eq. (1). As the aim is to displace the drilling mud and replace it with the cement slurry, in any simulation the fluids must travel the length of the well Z, and the transport Eq. (9) is advanced over a total time $T \sim Z$, given that the velocities have been scaled to be $O(1)$. At each time step both the stream function Eq. (6) and the transport Eq. (9) must be solved. Equation (9) imposes a CFL constraint on the time step used, but the computational time taken per timestep scales linearly with the number of mesh cells. In more complex models of the mass transport diffusive effects may also be included (molecular and turbulent diffusivities, Taylor dispersion, etc.; see [19]). Nevertheless, in the dimensionless setting the (physically scaled) diffusivities remains small enough that an explicit method can be used without compromising the time step restrictions, i.e. the CFL condition can be considered the main time step restriction.

The above represent the unmovable constraints of the computational problem, in terms of number of time steps that need to be executed, for any given spatial discretization. The spatial discretization is determined by accuracy and resolution requirements. Here we will assume a simple rectangular mesh. Although one can elongate the mesh spacing in ξ to help reduce the impact of $Z \gg 1$, this comes with a loss of resolution. Regarding azimuthal resolution, the point of the annular Hele-Shaw approach is to resolve 2D effects around the displacement front (secondary flows), so that $N_{phi} = 10 - 100$ azimuthal meshpoints would be sensible. Assuming a mesh $d\phi = 1/N_{phi}$ in the azimuthal direction, we will frequently want $d\xi \sim d\phi$, to avoid distorting the mesh too much. This might also be a reason for not selecting N_{phi} too large. Other factors in selecting $d\xi$ include the fact that for industrial settings, the eccentricity, mean radius and gap width may all vary considerably along a well and one wishes to capture these variations. On the other hand, as the Hele-Shaw approach involves averaging across the annular gap, resolving flow features on the gap length-scale is the limit of model validity. Thus having balanced all consideration, the number of meshpoints is $N \sim O(ZN_{phi}^2)$, with potentially a large constant of proportionality (reflecting the mesh aspect ratio).

The second-order Eq. (6) is elliptic and nonlinear. Broadly speaking, at each time step we must solve this nonlinear problem of size N, and the usual methods involve repeatedly resolving a sequence of linear problems, each of size N. Although of size N, these elliptic problems are sparse and the matrices are generally banded with width $3N_\phi$. For now lets assume that (6) is solved in a time $\propto A_{ALG}N^{a_{ALG}}$ on each time step, where A_{ALG} & $a_{ALG} (\geq 1)$ will depend on the algorithm used and its implementation. Lets now consider the total time required to simulate a displacement, T_{comp}. We suppose an end time $T \sim Z$ and that a CFL condition is imposed, which lead to:

$$T_{comp} \propto [A_{ALG} N^{a_{ALG}} + O(N)] Z N_{phi}. \tag{14}$$

The $O(N)$ term comes from the cost of solving (9), assumed of size N and relatively inexpensive. The main cost at each time step is in resolving (6), which we discuss below and focus on improving in this paper. Optimistically, we might be able to ensure that $a_{ALG} = 1$, as we will discuss, which leaves A_{ALG} to be improved upon. Nevertheless, we see that at best we end up with:

$$T_{comp} \propto A_{ALG} Z^2 N_{phi}^3. \tag{15}$$

Although we work on reducing A_{ALG} in this paper, the other terms above remain immovable and $Z \gg 1$ is a serious constraint to rapid computations in realistic well geometries. Simply put, the aim of the parallelization is to reduce the dominant per-timestep cost ($A_{ALG}N$), of solving (6), ideally by dividing by the number of parallel GPU threads.

3.1 Finding the streamfunction

Although elliptic, the streamfunction Eq. (6) is not defined in regions where the velocity is zero, i.e. where the gradient of Ψ vanishes. Instead of the classical formulation, we adopt the weak formulation of (6) as a variational inequality, as is discussed at length in [13]. This leads naturally to a minimization problem for the streamfunction (at each time step), formally written as:

$$\min_{\Phi \in \bar{V}_0(\Omega)} \{J(\Phi)\}, \tag{16}$$

where $\Omega = (0, 1) \times (0, Z)$. The functional only depends on the gradient of the streamfunction: $\mathbf{q} = \nabla_a \Phi$, and can be split as follows.

$$J(\mathbf{q}) = J_0(\mathbf{q}) + J_1(\mathbf{q}), \quad \mathbf{q} \in X, \text{ where} \tag{17}$$

$$J_0(\mathbf{q}) = \int_\Omega \left(\frac{1}{2} \int_0^{|\nabla_a \Psi^* + \mathbf{q}|^2} \frac{\chi(s^{1/2})}{s^{1/2}} \, \mathrm{d}s + \mathbf{q} \cdot \tilde{f} \right) \mathrm{d}\Omega, \quad \mathbf{q} \in X, \tag{18}$$

$$J_1(\mathbf{q}) = \int_\Omega \frac{\tau_Y}{H} |\nabla_a \Psi^* + \mathbf{q}| \, \mathrm{d}\Omega, \quad \mathbf{q} \in X. \tag{19}$$

We have homogenized the boundary conditions for Ψ, along $\phi = 0, 1$, and at the ends $\xi = 0, Z$, by setting

$$\Psi = \Psi^* + \Phi, \ \Psi^* \in \bar{V}(\Omega), \ \Phi \in \bar{V}_0(\Omega), \tag{20}$$

with suitably constructed Ψ^*. In the continuous setting we take

$$V_0(\Omega) = \{\Phi \in C^\infty(\Omega) : \ \Phi(0, \xi) = 0, \ \Phi(1, \xi) = 0\},$$

$$V(\Omega) = \{\Phi \in C^\infty(\Omega) : \ \Phi(0, \xi) = 0, \ \Phi(1, \xi) = 1\},$$

and the test spaces $\bar{V}_0(\Omega)$ and $\bar{V}(\Omega)$ are the closures of $V_0(\Omega)$ and $V(\Omega)$, respectively, with respect to the $W^{1,1+n}(\Omega)$ norm. Hence, $X = L^{1+n}(\Omega)$, and n would be the smallest value of n in Ω, (which may change with time if the fluid mix). Nevertheless, practically speaking we work with finite dimensional spaces for the numerical solution (when $n = 1$ is equivalent).

The functional $J_1(\boldsymbol{q})$ is non-differentiable in regions where $\nabla_a(\Psi^* + \Phi) = 0$, which correspond to stationary parts of the flow. One approach is to regularize the problem to remove the non-differentiability, which physically means that the fluids may move very slowly in these regions. However, from the industrial perspective, it is of interest to determine which regions are truly stationary. Therefore, we fully resolve the static regions by dealing directly with the non-differentiability. This is done using the algorithm developed in [15] that employs the augmented Lagrangian algorithm of [23].

This method replaces the minimization with a saddle point problem, that results from introducing an additional variable for the gradient of the streamfunction. The Lagrangian saddle point problem is also typically augmented with a stabilizing term. Thus, the augmented Lagrangian problem concerns the functional L_r:

$$\mathbf{L}_r(\Phi, \mathbf{q}, \mu) = J(\mathbf{q}) + \int_\Omega \mu \cdot (\nabla_a \Phi - \mathbf{q}) d\Omega + \frac{r}{2} \int_\Omega |\nabla_a \Phi - \mathbf{q}|^2 d\Omega \tag{21}$$

The saddle point of $\mathbf{L}_r$, $\forall r > 0$, is found using an iterative Uzawa algorithm, in which successive iterates solve for one variable.

For example, holding $\{\mathbf{p}^{k-1}, \lambda^{k-1}\}$ fixed, we solve for Φ^k. The problem for Φ^k minimizes $\mathbf{L}_r(v, \mathbf{p}^{k-1}, \lambda^{k-1})$ with respect to v, which we see is a linear elliptic problem (a Poisson equation). The discrete matrix does not change on each iteration, which means that this problem can generally be solved cheaply.

The second problem, to define $\mathbf{p}^k$, holds $\{\Phi^k, \lambda^{k-1}\}$ fixed, and minimizes $\mathbf{L}_r(\Phi^k, \mathbf{q}, \lambda^{k-1})$ with respect to $\mathbf{q}$. After some algebra, we find that $\nabla_a \Psi^* + \mathbf{q}$ should be parallel to the vector $(\lambda^{k-1} + r\nabla_a(\Psi^* + \Phi^k) - \tilde{\mathbf{f}})$. If $x = |\lambda^{k-1} + r\nabla_a(\Psi^* + \Phi^k) - \tilde{\mathbf{f}}|$ then $(\nabla_a \Psi^* + \mathbf{q})$ has magnitude θx, where $\theta \geq 0$ minimizes

$$M(\theta) = \frac{1}{2} \int_0^{(\theta x)^2} \frac{\chi(s^{1/2})}{s^{1/2}} ds + \frac{\tau_Y}{H} |\theta| x + \frac{r}{2} (\theta x)^2 - \theta x^2. \tag{22}$$

This local minimization has two cases. First, if $x \leq \tau_Y / H$ then $M(\theta)$ is minimized by $\theta = 0$, with solution $\mathbf{q}^k = -\nabla \Psi^*$, that represents an unyielded flow. The other case represents yielded flow, when $x > \tau_Y / H$. In this second case we simply differentiate $M(\theta)$ and equate to zero to obtain the minimum:

$$\chi(\theta x) + \frac{\tau_Y}{H} + r\theta x - x = 0 \tag{23}$$

Recall that χ is defined by (5), Which is an implicit definition of χ as a function of the areal flow rate $|\nabla_a \Psi|$. Thus, the problem for $\mathbf{q}^k$ at each iteration is time consuming, i.e. we must solve (23) iteratively and each evaluation of χ would require the implicit function χ in (5) to be inverted. This is a significant contribution to the total computational time.

3.2 Computational improvements

We refer to the algorithm explained above as the serial algorithm. We explore the performance of this code, which is predictably slow, and improve it. In order to accelerate the code, we have implemented two strategies: GPU parallelization and algorithmic improvements.

3.2.1 GPU parallelization

As mentioned in the introduction, our code was parallelized using CUDA [24], and specifically CUDA Fortran [25]. The CPU is an Intel® Core^{TM} i7-3770 CPU @ 3.40 GHz, with speed 3411 MHz and with 32768 MB (DDR# 1333 MHz). The hardware added to our computer was a NVIDIA® graphic card, model GeForce RTX 2080 Ti. All examples presented later are run using the same computer, with only specific functions translated to CUDA and run on the GPU. All data passes through the CPU also. In Fig. 3 we present a test of computational performance in Gigafloating-point operations per second (GFLOPS) for that particular model of GPU vs. our CPU. We see an increasing difference between the GPU and the CPU performance with the size of problem.

Generally, we have to apply two criteria to decide if an application is suitable to be parallelized via GPU:

1. It is computationally intensive. The time spends on transferring data to a GPU memory is significantly less than the time in computation.
2. The computation can be broken down into hundreds/thousands of independent units of work, which means that they can be massively parallelized.

If the program does not satisfy both criteria, it may run slower on a GPU. A GPU is especially well-suited to solve problems with data-parallel computation, i.e. the same operation or function is executed at the same time or in parallel. This allows the memory access latency to be hidden with high arithmetic intensity instead of the big data cache. Applications that process large data sets can use a data-parallel programming model to speed

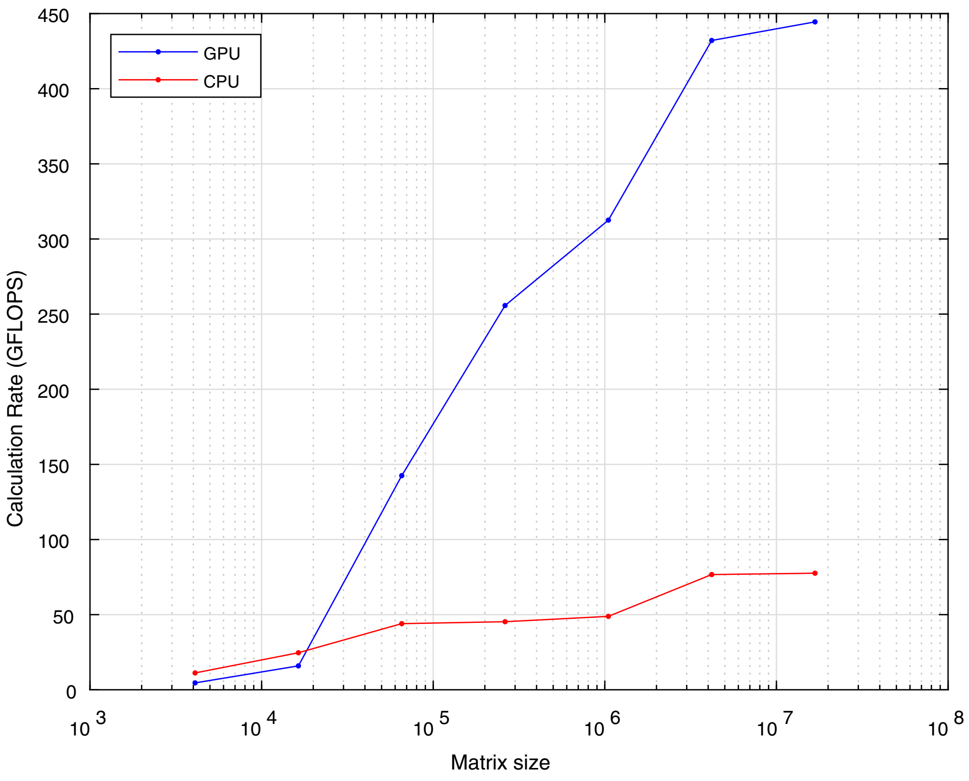

Fig. 3 Performance comparison between GPU and CPU using GeForce RTX 2080 Ti, using a standard matrix multiplication test

up computations. A multi-threaded program is partitioned into a block of threads that execute independently of each other. Therefore, a GPU with more CUDA cores should execute code in less time than a GPU with fewer multiprocessors, unless memory bound.

Turning to our annular model, the mathematical operations are varied. We will provide a profile of the main computational costs below, but overall the main cost comes from the augmented Lagrangian algorithm outlined in Sect. 3.1. The key point is that the most expensive step is solved locally (i.e. independently) on each finite volume cell, allowing us to identify each with one thread on the GPU and making the parallelization process very straightforward. Other parts of the algorithm are either not suitable for parallelization, or are not costly enough to warrant this attention.

3.2.2 Algorithmic improvements

In addition to the code parallelization, we modified the way the nonlinear problem (23) is solved. Briefly, note that the physical definition of χ is as the excess of the pressure gradient over its limiting value, given by

$$\chi = G - \frac{\tau_Y}{H}. \tag{24}$$

Thus, when $\chi > 0$ the fluid is flowing locally, meaning that $|\nabla_a \Psi^* + \mathbf{q}^k rvert > 0$. Also note that the function $\chi(\theta x)$ increases monotonically, where $\theta x = |\nabla_a \Psi^* + \mathbf{q}^k|$. With a simple change of variable, we can rewrite (23) not as an equation for θ, but rather as an equation for χ, i.e.

$$f(\chi) = \chi + r\frac{H^{2+1/n}}{\kappa^{1/n}(2+1/n)}\frac{\chi^{1+1/n}}{(\chi + \tau_Y/H)^2}\left[\chi + \frac{(2+1/n)\tau_Y}{(1+1/n)H}\right] - \left(x - \frac{\tau_Y}{H}\right) = 0. \tag{25}$$

The function $f(\chi)$ is evidently monotonically increasing, and considering that $x > \frac{\tau_Y}{H}$ we also see that $f(0) < 0$. It is not hard to establish that (25) has a single zero, which needs to be found numerically. Note

Table 2 Percentage of time by function

Function	$Time(sec.)on124x256mesh$	%
Advancep	376.50	97.2
Advancelambda	9.41	2.42
AdvanceOneTimeStep	1.34	0.34
Advanceu	0.13158	0.035

however that this eliminates one level of nested iteration in defining χ. Having found the solution to (25), we can use (5) to find θx explicitly, and as $|\nabla\psi^* + \mathbf{q}^k| = \theta x$, then $\mathbf{q}^k$ is found.

There are many options that can be used to solve (25), e.g. bisection method, regula falsi, Ridder's method. Although nonlinear, the properties of (5) are analyzed in depth in [15], which makes it relatively easy to find bounds for the solution and iterate reliably. We shall refer to using (25) in place of (23) as the optimized algorithm. For all results presented in the paper the bisection method has been used.

4 Results

Before presenting results of representative cases, we give an overview of the relative computational costs. We have measured the runtime of the entire code and profiled runtimes of the different parts of the code. Apart from the initialization steps in the code, the main subroutines that are called on each time step are as follows:

- `advanceonetimestep`: this calculates the concentration of each fluid inside the annulus, advancing one time step using an explicit (flux corrected transport) method.
- `advanceu`: this calculates a new value of Φ from the saddle point problem.
- `advancep`: this calculates a new value of $\mathbf{q}$ from the saddle point problem.
- `advancelambda`: this calculates a new value of λ.

As we have noted in the prior discussion, `advanceonetimestep`, `advancep` and `advancelambda` represent operations that naturally scale linearly with the size of the mesh. In general, `advanceu` involves solving a linear Poisson problem on the mesh. However, the Poisson problem remains unchanged for successive time steps. Thus, here we have performed a QR decomposition at the start of the computation and used this repeatedly on each time step to solve the Poisson problem. This ensures that `advanceu` will scale linearly with the size of the mesh. This method is suitable for the moderate sized meshes that we use, but some form of iterative method would be needed for very large problems.

Table 2 shows the percentage of computational time taken by each function for a typical computation using yield stress fluids and with a mesh size `124x256`. We see that `advancep` represents 97% of the runtime. Although each function should scale linearly with time, the nested nonlinear iteration performed on each mesh cell is the dominant cost. This explains our motivation for targeting our GPU implementation for this operation.

4.1 Case studies

We have developed and analyzed six different cases to illustrate typical rheological parameters and operational variables. Our cases are greatly simplified compared to the ranges found in operations, by having only 2 fluids present and in that the annulus is uniform. The fluid properties are specified in Table 3. All cases consider inner and outer radii: $\hat{r}_i = 0.15$m and $\hat{r}_o = 0.175$m, and a length $\hat{l} = 1000$m. Each annulus has a constant standoff (eccentricity). The annuli are all vertical, which would relate to a surface or intermediate casing operation. For simplicity the flow rate is fixed at $\hat{Q} = 0.002525$ m^3/s. Even though simplified, each case is described by 15 parameters in the dimensional setting, which can be reduced to 11 dimensionless parameters by scaling the equations and variables. The dimensional parameters are described and discussed in [19].

In Table 4 we listed the dimensionless parameters used in our experiments. Note that $\rho_2 = 1$ in all the cases due to scaling, and the scaled length of the well is $Z = 1958.86$. We use $N_\xi = 1024$ meshpoints in the axial direction throughout. Cases A-C use $N_\phi = 64$ and the remainder use $N_\phi = 32$. All computations start with the annulus completely full with displaced fluid (colored yellow below), and with the interface at the bottom ($\xi = 0$) of the annulus.

Table 3 Dimensional fluid properties: $\hat{r}_i = 0.15$m, $\hat{r}_o = 0.175$m, $\hat{l} = 1000$m, $\beta = 0$, $\hat{Q} = 0.002525$ m^3/s. Fluid 1 (displaced fluid), Fluid 2 (displacing fluid)

Case	e	$\hat{\kappa}_1$=1.0 (Pa.s^n)	$\hat{\kappa}_2$=1.0 (Pa.s^n)	n_1	n_2	$\hat{\tau}_{Y,1}$ (Pa)	$\hat{\tau}_{Y,2}$ (Pa)	$\hat{\rho}_1$ (kg/m^3)	$\hat{\rho}_2$ (kg/m^3)
A	0.2	1.0	1.0	1.0	1.0	0	0	1200	1200
B	0.5	1.0	1.0	1.0	1.0	0	0	1200	1200
C	0.5	1.0	1.0	0.6	0.5	0	0	1200	1200
D	0.5	0.02	0.01	0.7	1.0	7.05	2.35	1440	1800
E	0.5	0.02	0.03	0.7	1.0	4.79	2.35	1440	1800
F	0.4	1.0	1.0	1.0	0.6	0.3	1.0	1200	1300

Table 4 Dimensionless parameters: $\rho_2 = 1$, $Z = 1958.86$. Fluid 1 (displaced fluid), Fluid 2 (displacing fluid)

Case	St^*	e	κ_1	κ_2	n_1	n_2	$\tau_{Y,1}$	$\tau_{Y,2}$	ρ_1
A	0.233	0.2	1.0	1.0	1.0	1.0	0.0	0.0	1.0
B	0.233	0.5	1.0	1.0	1.0	1.0	0.0	0.0	1.0
C	0.135	0.5	1.0	0.8123	0.6	0.5	0.0	0.0	1.0
D	0.143	0.5	0.0120	0.0336	0.7	1.0	0.9880	0.3290	0.8
E	0.143	0.5	0.0176	0.0492	0.7	1.0	0.6705	0.3290	0.8
F	0.0806	0.4	0.0773	0.0336	1.0	0.6	0.2899	0.9664	0.9

Cases A & B are Newtonian. The densities are the same for both fluids. In principle the augmented-Lagrangian procedure is not needed for these linear cases. These are included for bench-marking purposes. In Fig. 4 we show results from case A at 4 successive times through the displacement flow. The streamlines (white) are plotted over the colormap of the fluid concentrations. For this base case, the only effect is due to the eccentricity ($e = 0.2$), which causes the fluid to flow faster in the wider part of the annulus. From an initially horizontal interface we observe that the interface elongates along the wide side of the annulus and exits. Case B is a similar flow but with larger eccentricity ($e = 0.5$); see Fig. 5. The behaviour is qualitatively the same as in Fig. 4, but as we increase the eccentricity the fluid in the wider part of the annulus increases and that in the narrower part decelerates, so that the interface elongates in each frame.

Note that for these two cases there is no density difference. The 2 fluids rheologies are identical, and we are viewing a pure dispersion problem. The degree of spreading of the concentration profile at the interface is entirely indicative of numerical diffusion/dispersion in solving the concentration equation (9). Unlike in more complex situations, the streamlines are entirely parallel and there is no secondary flow close to the interface. It can be observed that there is relatively little spreading of the interface with the current algorithm. Note that since the fluids upstream and downstream are identical, the streamlines should remain parallel across the interface.

Case C represents the first displacement flow; see Fig. 6. The fluids are now shear-thinning ($n_1 = 0.5$ and $n_2 = 0.6$) and the eccentricity is still $e = 0.5$. Recall that the flow rate imposed is identical for all cases, to allow comparison. Due to the shear-thinning effect both fluids flow faster on the wide side, relative to the Newtonian case B, i.e. faster flow shears the fluid more, reducing its effective viscosity. We observe this in that the streamlines are now denser on the wide side than in case B. The consequence of the faster velocities is that the interfaces advances faster on the wide side, making the displacement less effective than that of case B. Note too that fluid 2 has smaller n and hence is effectively less viscous than fluid 1, which may contribute to the finger-like phenomenon observed. The streamlines now distort slightly as they cross the displacement front, only becoming parallel upstream and downstream of the front. We see that the wide side interface develops a shock-like profile. Distortion of the streamlines from vertical becomes more apparent as we consider fluids with larger physical differences below.

A different non-Newtonian phenomenon is introduced in case D, where both fluids are shear thinning, $n_1 < n_2 < 1$, and both have a yield stress, that is larger in the displaced fluid, as is often the case for drilling mud and cement slurry; see Fig. 7. The eccentricity is still $e = 0.5$, but there is a significant density difference, $\rho_2 > \rho_1$, with the density of the displacing fluid being approximately that of cement. Similar to the previous example, the shear-thinning of the fluids leads to a faster flow in the wide side of the annulus, but this effect is countered by the buoyancy force coming from the density difference. The result is a more efficient displacement. The distortion around the interface is more visible, and we see significant dispersion ahead of the front close to $\phi = 0$, the wide side of the annulus. It has been shown in Figs. 4 and 5 that the

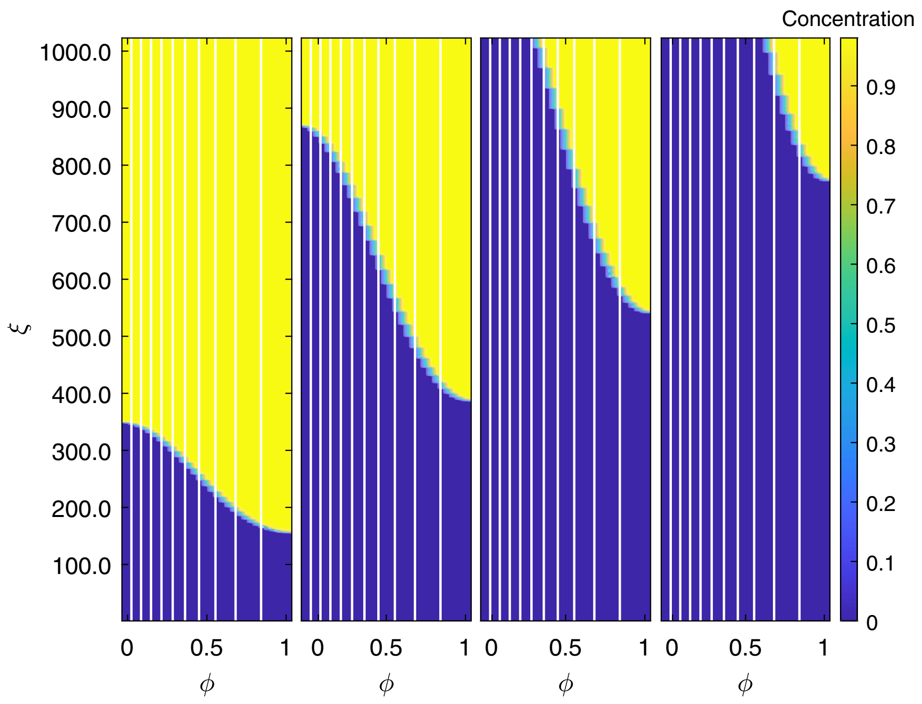

Fig. 4 Case A concentration colormap, computed in parallel. Sequence of times: 2042.14 s, 5105.26 s, 7147.33 s, 1010.59 s, from left to right. The white lines show the streamlines $\Psi = 0,\ 0.1,\ 0.2,\ \ldots, 1$

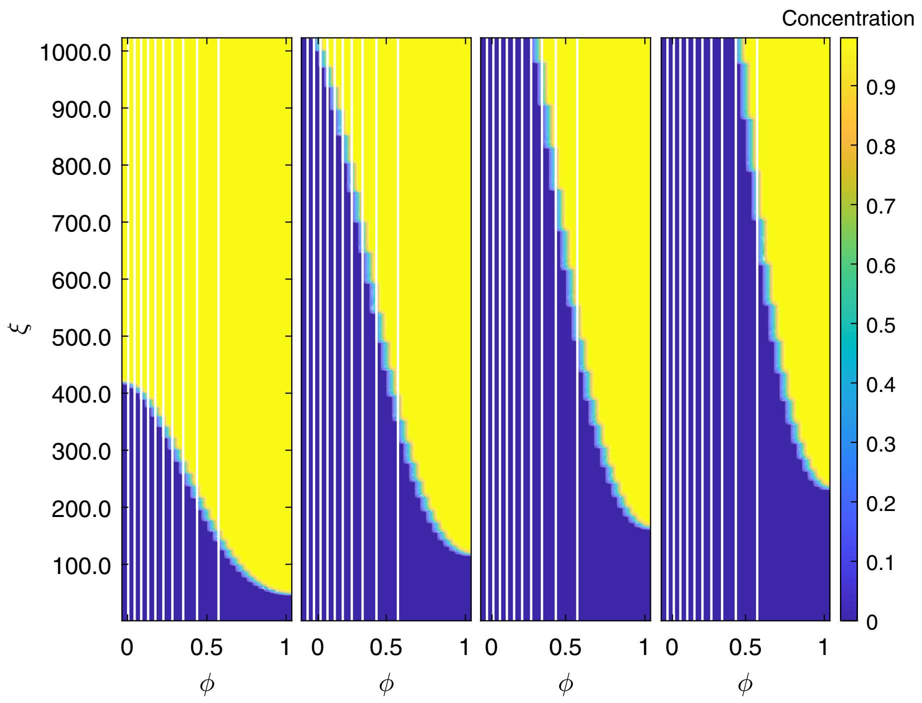

Fig. 5 Case B concentration colormap, computed in parallel. Sequence of times: 2042.14 s, 5105.26 s, 7147.33 s, 1010.59 s, from left to right. The white lines show the streamlines $\Psi = 0,\ 0.1,\ 0.2,\ \ldots, 1$

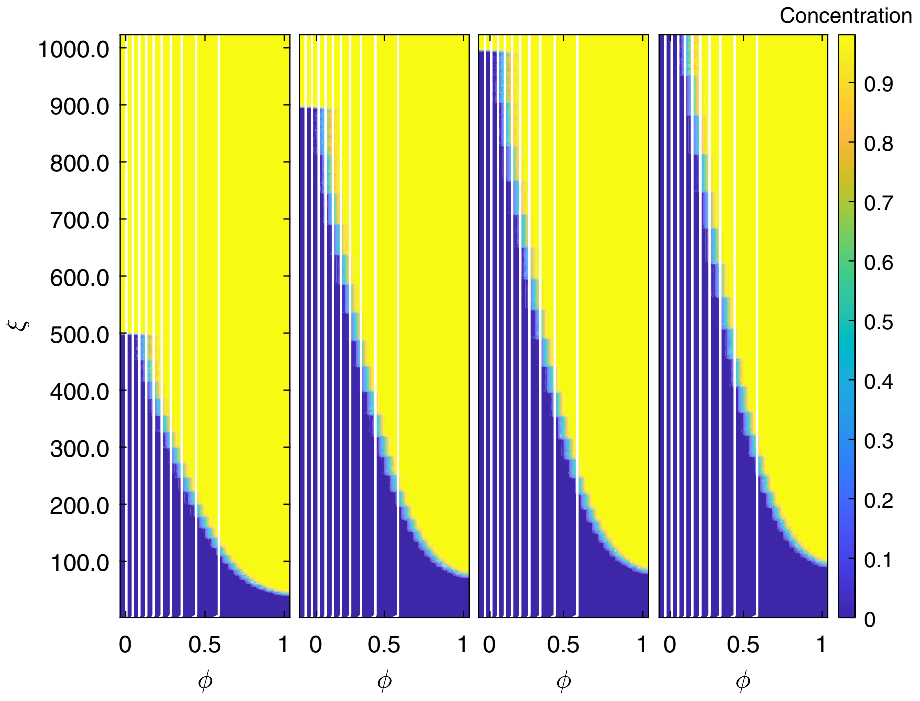

Fig. 6 Case C concentration colormap, computed in parallel. Sequence of times: 2042.14 s, 5105.26 s, 7147.33 s, 1010.59 s, from left to right. The white lines show the streamlines $\Psi = 0,\ 0.1,\ 0.2,\ \ldots, 1$

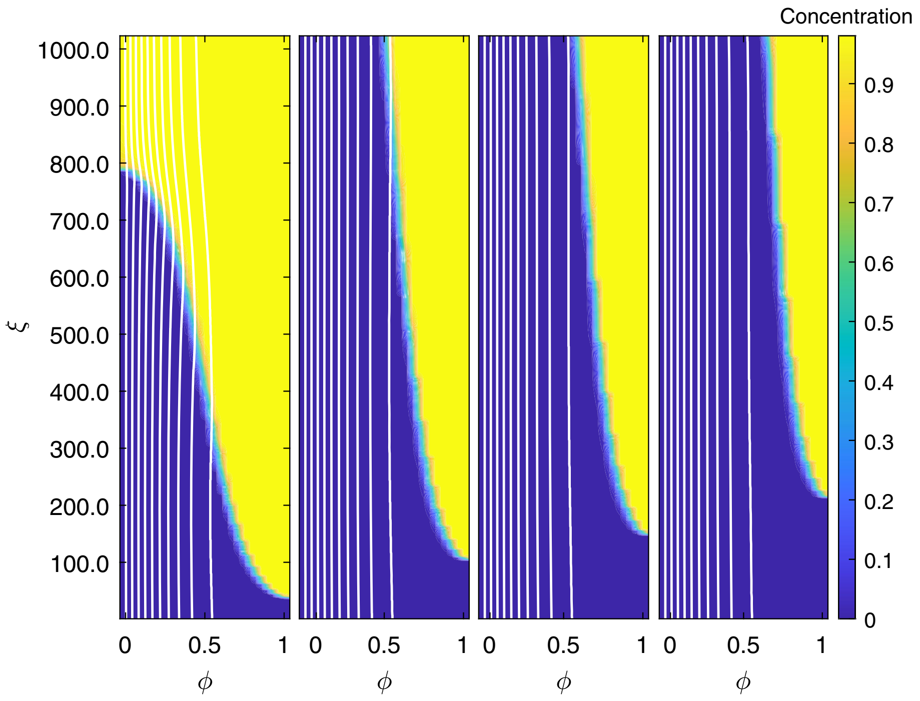

Fig. 7 Case D concentration colormap, computed in parallel. Sequence of times: 2042.14 s, 5105.26 s, 7147.33 s, 1010.59 s, from left to right. The white lines show the streamlines $\Psi = 0,\ 0.1,\ 0.2,\ \ldots, 1$

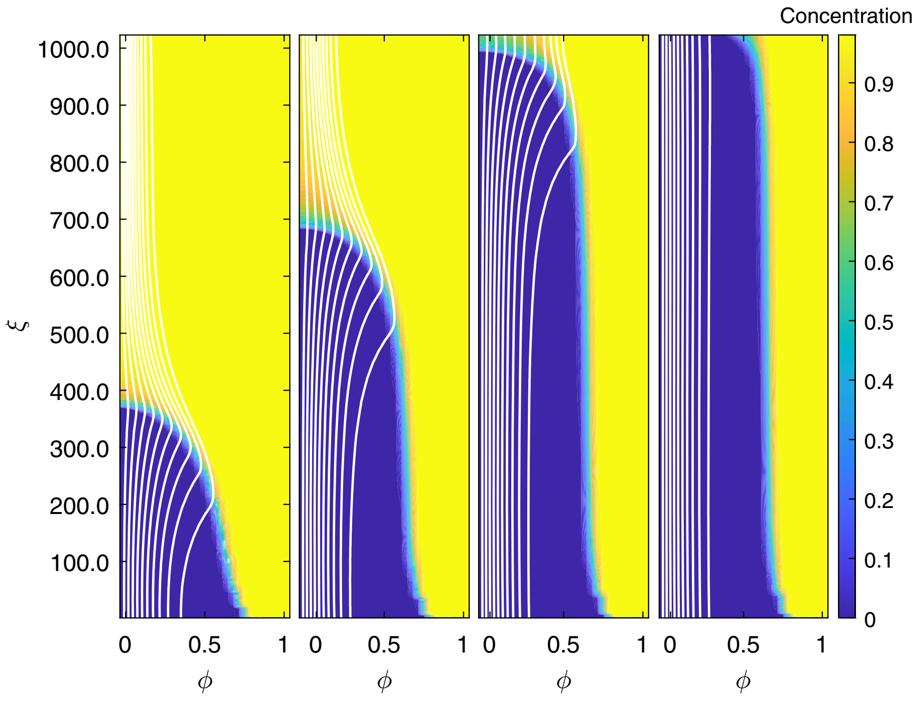

Fig. 8 Case E concentration colormap, computed in parallel. Sequence of times: 2042.14 s, 5105.26 s, 7147.33 s, 1010.59 s, from left to right. The white lines show the streamlines $\Psi = 0,\ 0.1,\ 0.2,\ \ldots, 1$

numerical method is able to preserve the sharpness of the front, so the dispersion is due to secondary flows that emerge. On the wide side, near $\phi = 0$, there is flow ahead of the front and on the narrow side, near $\phi = 1$, the secondary flow is behind the front. This secondary flow, which acts to flatten the interface, is due to buoyancy.

Note that ahead of the front, in the displaced fluid, in the narrowest part of the annulus, there are no streamlines, $|\nabla_a \Psi| \sim 0$; see (5). Interestingly, far behind the front, when fully in fluid 2, there is also static fluid on the narrow side. However, buoyancy induces weak secondary flows on the narrow side, sufficient to move fluid 1 away from $\phi = 1$ in fluid 1 and towards $\phi = 1$ in fluid 2. This allows the front to move along the narrow side, albeit slowly. It is interesting to compare with case C, where there is also poor narrow side displacement. The slow movement of the front on the narrow side is comparable between cases C and D. However, case C has no density difference and no yield stress, thus the fluids are always moving on the narrow side, and not just close to the front. Evidently, both cases C and D are poor displacements. In the preceding figures the final time shown corresponds to approximately 1 annular volume pumped. During a cementing operation, an excess of $\sim$20% annular volume may be pumped, but continuous pumping of cement is simply not possible, due to environmental waste disposal and cost concerns. Thus, displacement simulations such as cases C and D show wells that will have very poor cement coverage over significant lengths, likely leading to leakage and giving concern for corrosion, i.e. the cement does not protect the narrow side of the casing.

Case E is similar to case D, but with a reduced yield stress in fluid 1 and an increased consistency in fluid 2. Both effects lead to a better displacement, albeit still poor, as we see that there remains a static channel along the narrow side of the annulus; see Fig. 8. The residual channel of fluid 1 is narrower than in case D. In order to displace a yield stress fluid fully, one needs a combination of features: reduced yield stress, increased viscosity on fluid 2, significant buoyancy and reduced eccentricity. Finding the parametric ranges in which displacements can be effective is a key engineering challenge, more so when the well geometries are not as simple as those here. Case F illustrates a situation where the yield stress fluid 1 is fully displaced.

4.2 Benchmark comparisons

Cases A and B involve only passive advection, i.e. fluids 1 and 2 are identical. This situation admits an analytical solution for the streamfunction and hence velocity; see [14]. The computed concentration may then

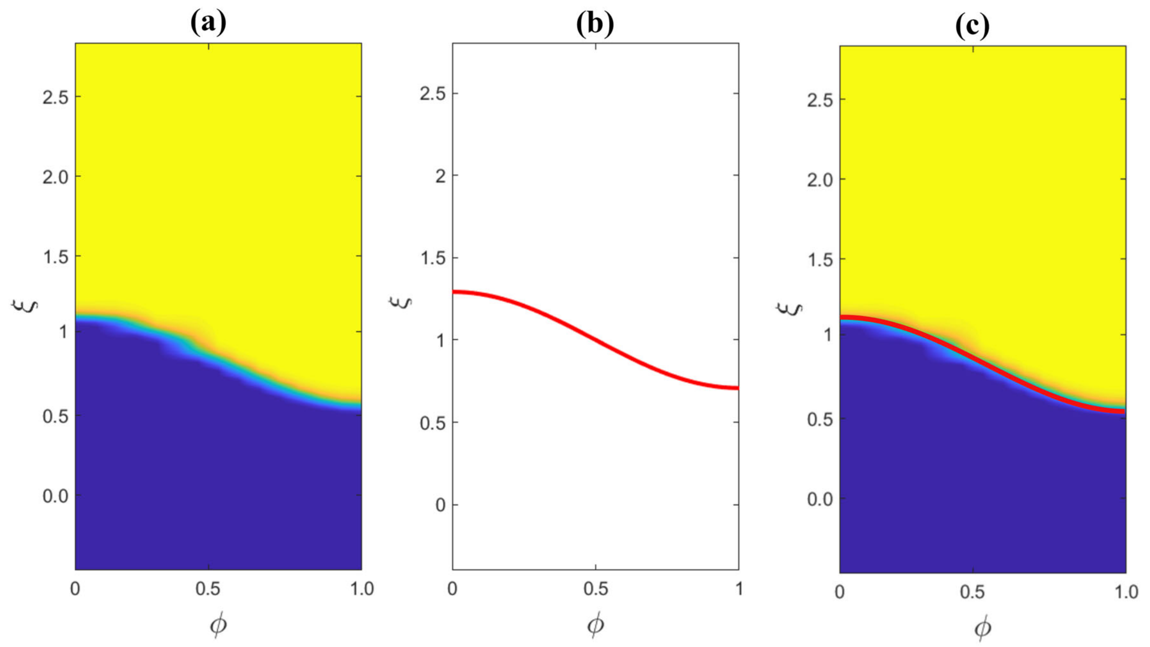

Fig. 9 Case A: **a** computed solution; **b** analytical advected interface; **c** comparison

be compared with the advected interface. For identical Newtonian fluids the velocity depends only on ϕ and is given by:

$$\bar{w}(\phi) = \frac{H^2(\phi)}{\int_0^1 H^3(\phi)\, d\phi} \tag{26}$$

with $H(\phi) = 1 + e \cos \pi\phi$. This integral is readily evaluated and the (analytical) position of the interface is simply advected from an initially flat initial condition. The comparison is made in Fig. 9 for case A. Note that the blocky/blurring near the interface is a consequence of mesh resolution and possibly the shading graphics. Figure 9 corresponds to early in case A, which is depicted in Fig. 4 over the full 1000m of annulus. For this comparison only the first few metres are shown.

Figure 10 shows comparisons of computed results using both the non-optimized serial and optimized parallel codes, for cases E and F. The results are very close. Displacement flows are advective, hence small differences in solution will flow downstream and can be amplified in doing so. Such differences in numerical solution remain small in cases where there is some underlying stability to the solution, as here. However, this might not always hold as many cementing displacement flows are unstable.

To give a quantitative comparison of the difference in solutions, we calculated the L^2 norm between the results from the non-optimized serial code and optimized parallel code, for meshes: 8x8, 8x16, 16x16, 16x32, 32x32, 32x64 64x64 and 64x128. We chose these specific size of meshes because the grids in the parallel version of the code should be a power of 2, to match the number of threads in the GPU that run the functions in parallel.

As we can verify in Figs. 11a and 12a, the difference is less than 10^{-13} for all the meshes, for the Newtonian fluid computations. For the non-Newtonian cases this increases to $\lesssim 10^{-11}$, as shown in Fig. 13a for case E. Although the algorithm is the same for Newtonian and non-Newtonian cases, in general the non-Newtonian flows require more iteration, which allows for larger error in convergence. As commented with regard to Fig. 10, small numerical differences are transported with the displacement flow and hence the discrepancies between methods will grow with time in most simulations, and more so in response to the flow complexity.

4.3 Computational times and scaling

For the same 3 cases, A, B and E, Figs. 11b, 12b and 13b show the computing time taken for various meshes. The red line denotes the non-optimized serial code and the blue line is the optimized parallel code. Both codes

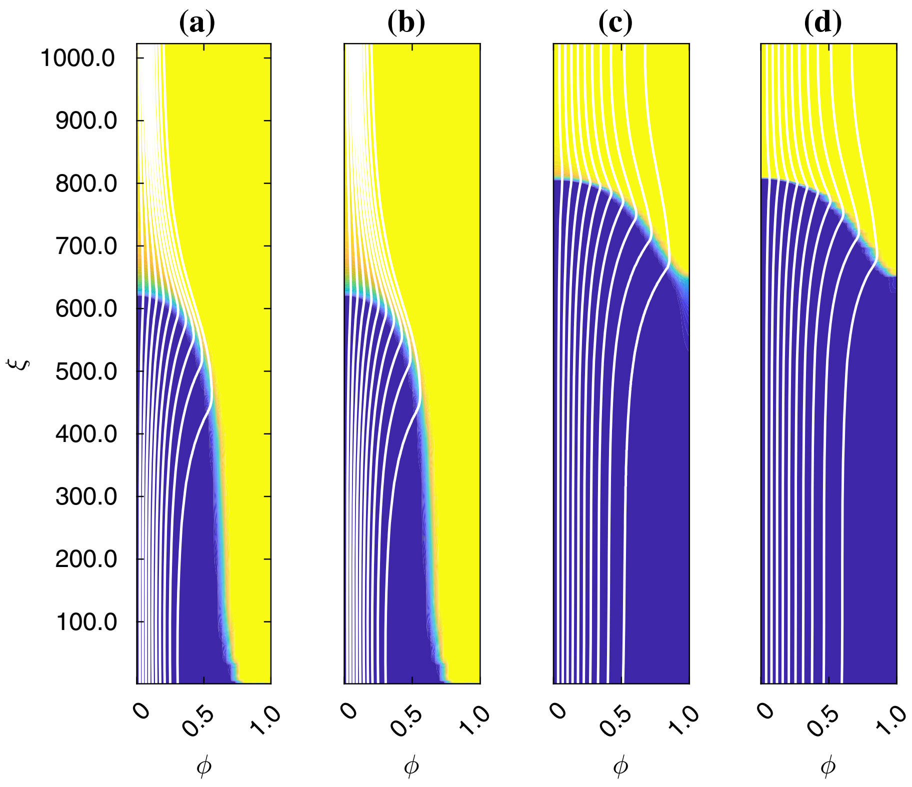

Fig. 10 **a** Case E in optimized parallel at t = 4333; **b** Case E in nonoptimized serial at t = 4333; **c** Case F in optimized parallel at t = 4982; **d** Case F in non-optimized serial at t = 4982

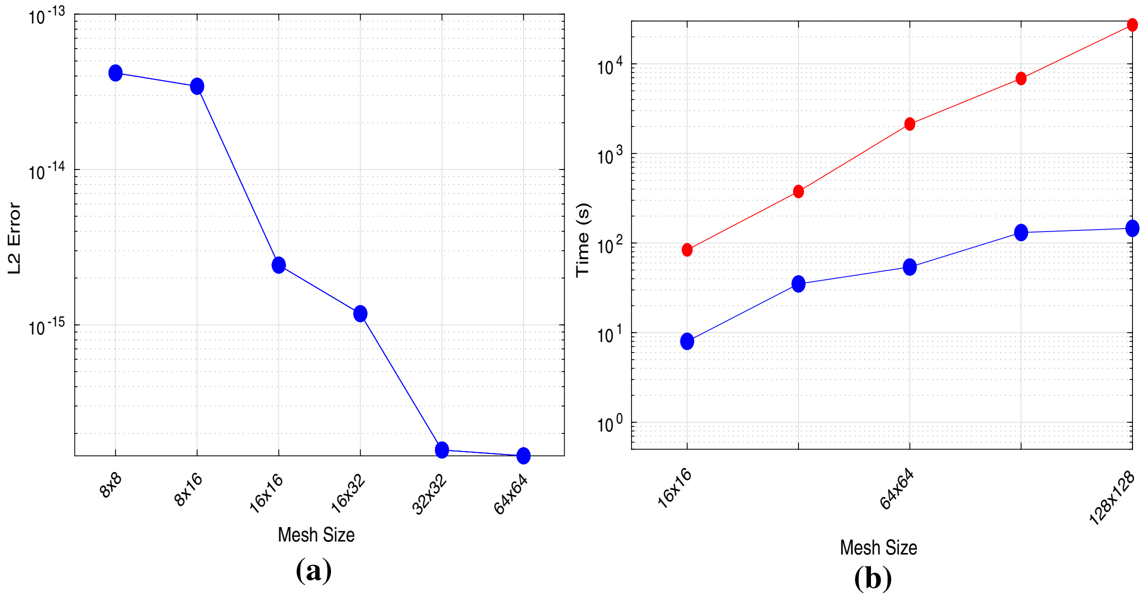

Fig. 11 Case A: **a** L^2 error between optimized parallel and non-optimized serial codes; **b** annular timing benchmark; red line: non-optimized serial code; blue line: optimized parallel code

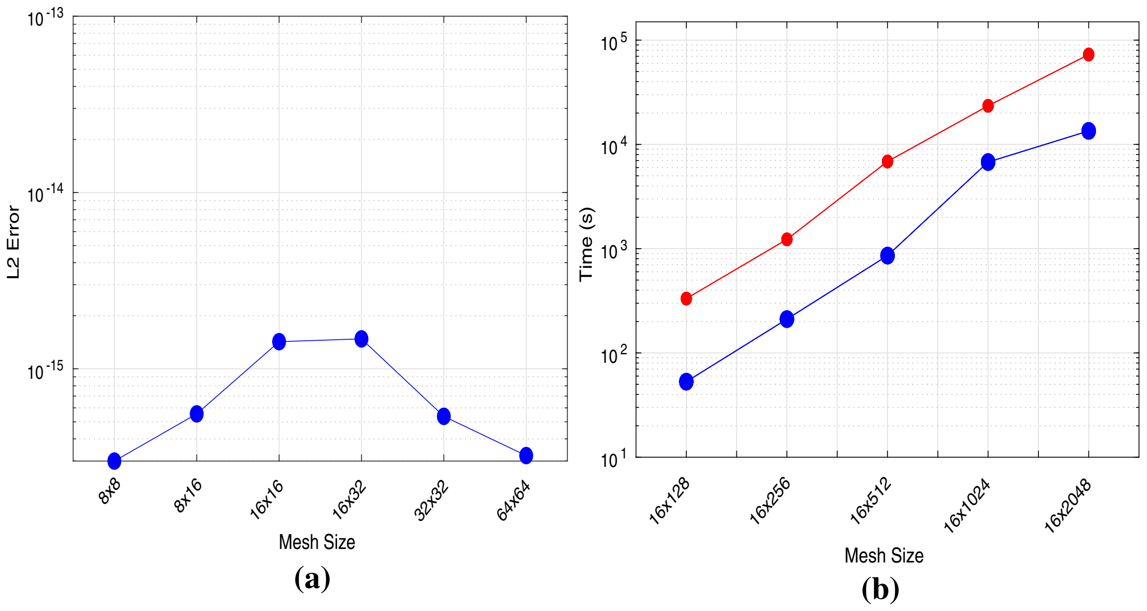

Fig. 12 Case B: **a** L^2 error between optimized parallel and non-optimized serial codes; **b** annular timing benchmark; red line: non-optimized serial code; blue line: optimized parallel code

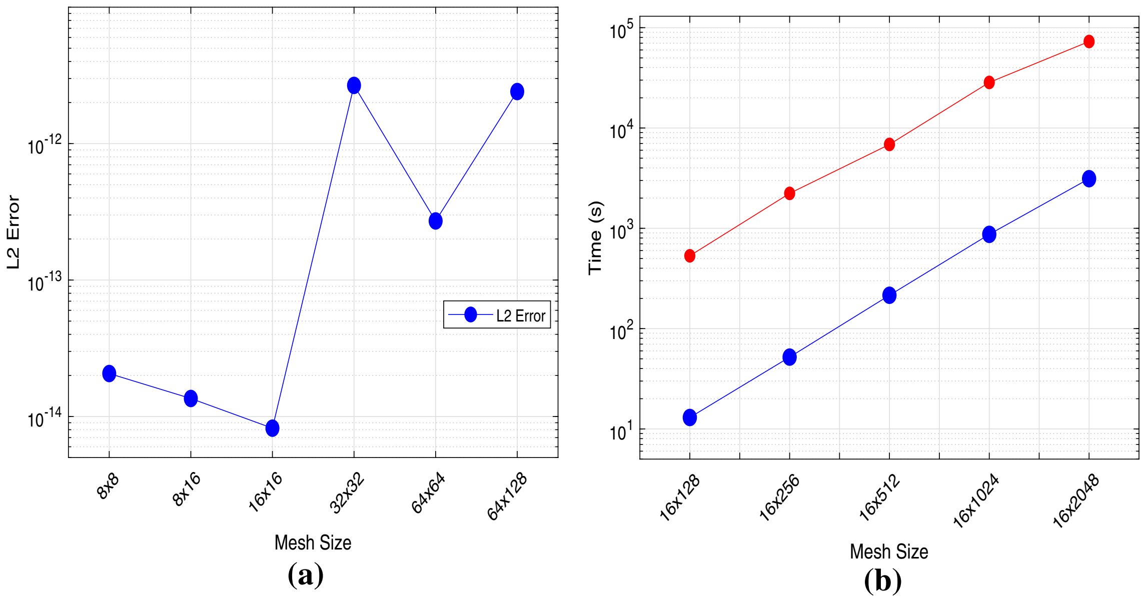

Fig. 13 Case E: **a** L^2 error between optimized parallel and non-optimized serial codes; **b** annular timing benchmark; red line: non-optimized serial code; blue line: optimized parallel code

runtime increase with the meshsize. The direct observation is that the blue line is taking less time than the red line across all the meshes in all cases. The calculations in the GPU are intrinsically very fast and take very little time, but only parts of the code are run on the GPU, i.e. those parts of the algorithm that have been parallelised.

Firstly, note that the decrease in computing time using the optimised parallel code appears to be more significant for the non-Newtonian simulations (Fig. 13b), compared to the Newtonian simulations (Figs. 11b and 12b). This is to be expected. In theory the augmented Lagrangian procedure should converge within a few iterations when the problem is linear. Possibly it could be designed to converge in 1 iteration, but we have not done so. In any case, this means that the GPU can not reach its full potential for the linear cases.

Table 5 Benchmark for cases: in parallel and in serial

Benchmark mesh 32 × 1024	Serial	Parallel
Case A	150 min 55 s	86 m 29 s
Case B	189 min 61 s	70 m 8 s
Case C	1197 min 46 s	78 m 49 s
Case D	2160 min 20 s	162 m 25 s
Case E	5755 min 30 s	177 min 1 s
Case F	6324 min 52 s	101 min 6 s

Table 6 Benchmark for cases: original algorithm and optimised algorithm

Benchmarktime mesh 32 × 1024	Optimized algorithm serial	Optimized algorithm parallel
Case A	140 min 3 s	46 min 43 s
Case B	139 min 5 s	46 min 45 s
Case C	140 min 29 s	61 min 54 s
Case D	589 min 36 s	67 min 17 s
Case E	593 min 5 s	71 min 58 s
Case F	360 min 5 s	69 min 58 s

In Table 5 we show the computational times taken for all the cases ran in serial and in parallel with a 32X1024 mesh. In Table 6 we show the time of the optimized algorithm in serial and the optimized algorithm in parallel. We notice the time reduction for all cases for the parallel version. We can also see that the time of the optimized serial algorithm is lower than the original serial algorithm. The best results are obtained with the code with both the optimized algorithm and in parallel.

To quantify the improvement we compared the average times of these cases, over the 3 different improvement strategies. We calculated the percentage of the time taken with respect to the original serial algorithm. The time in parallel is $\approx$ 13% of the time in serial. The optimized algorithm in serial takes $\approx$ 4.45% of the time of the original code. The optimized algorithm in parallel takes $\approx$ 2.5% of the original code. These specific numerical improvements will also depend on the mesh size, rheological and process parameters.

Noticeable here is that the very long runtimes (cases E & F, non-Newtonian fluids) improved the most. As we know, Non-Newtonian simulations require more computational resources. Hence, the optimization becomes more relevant and valuable in the light of the real-time simulations. Running 10–15 simulations of 1-hour duration to enhance the design of a well is feasible, but not when each takes 100 h. The most important point is that the improvements bring the simulations into the range of a single desktop machine with a good GPU, as is available to practicing engineers.

A very general perspective on expected speed-up can be given in the sense of the theoretical concept of scalability. The first definition of scalability is related to the size of the problem and the time it takes to be executed. There are many possibilities to make this calculation. When we want to analyze the size of the problem, the number of processors that are used to calculate the algorithm become relevant. The parallelization in a GPU uses different kinds of memories, like share memory, global memory, etc., and the arithmetic processor units are set in different levels, (threads, blocks, warps, SM, etc). A simple expression we can use to anticipate the scalability of our application it is Amdahl's law that relates the number of processors, to the proportion of the code in parallel and forecasts an approximation of the speed up S:

$$S = \frac{1}{f + (1 - f)/p}, \tag{27}$$

where f is the part of the program running in serial and p is the number of processors. Looking at Table 2, we see that the most expensive part of the code takes 97.2% of the (serial) computing time. It is this part that we have coded on the GPU. Thus, if we were to crudely take: $f = 2.8\%$, and $p = 4532$, we find that $S = 35.4$. This speed up is similar to that we have observed. Although demonstrating consistency of our results, the analysis is rather crude.

Although Table 2 resulted from profiling a relevant computation, different physical parameters would result in variability of the profile values, which has not been explored (there are at least 10 relevant dimensionless physical parameters). Specifying f in this way is also too simplistic. The CPU is used in the parallel codes and there is always some limiting in passing data and allocating to the GPU threads. Such features would change with different CPU and GPU pairings, as they are related to the architecture, but would not be eliminated. The focus of our paper is algorithmic and not hardware.

The other aspect to consider is that the best performance of our code comes from 2 independent improvements: algorithmic and GPU. Checking Tables 5 and 6 for the cases D-F (non-Newtonian and requiring the most iteration), illustrates the effects of the 2 improvements. Comparing the 1st column between the 2 tables indicate that the algorithmic improvement is a speed up in the range of 5-20 times. Comparing the 2 columns in each table, the GPU parts seem to speed the code by 12-60 times in the non-optimised cases and by a factor of 5-10 after the algorithmic improvement. Thus, the speed ups of the two improvements are comparable.

In terms of a general scaling of computational times with the size of the problem N, the reader is referred to the start of Sect. 3 for general comments regarding the expectations for the original serial code, while (27) gives an idealised view of the speed-up possible from the GPU implementation. As we have seen, the actual speed-ups are a combination of both improvements made. This brings us to the final point regarding scaling and a key difficulty in anticipating: the physical nature of the problem. Different choices of fluid change the nonlinearity and hence the computational times. This is also a time evolving problem (a displacement flow), which means that the proportion of the annulus filled with fluid 1 or fluid 2 (each with different nonlinearities) changes during the flow as different fluids are replaced.

5 Conclusions

In this study, we implemented a faster version of the annular displacement flows simulations first presented in [26]. We parallelized the original code using a GPU, and modified the algorithm where it had the most expensive computational cost. This resulted in an average reduction of the computing time to around 2.5% of the original cost. The cost reduction appears to vary significantly with the fluid rheologies: perhaps unsurprising as these change the nature of the non-linearities.

The study serves as an example of a class of fluid flow problems that may be amenable to significant GPU acceleration. It is fairly commonplace to use scaling arguments to reduce 3D Navier–Stokes problems to simpler situations where the flow occurs in a thin film, slowly varying channel or similar geometry. Such models are amenable to analysis and often are fast enough to compute, such that they may serve as a digital twin for process design or control. In these models there is typically a reduced direction in which the equations are integrated analytically (across a boundary layer, or thin film, etc). In dealing with complex fluids, these reduced models are not always analytically tractable and their computation can quickly become the main cost. Here we have shown that both GPU and smart consideration of the reduced models, can both lead to significant savings in computational cost.

Acknowledgements This research has been carried out at the University of British Columbia, supported financially by Schlumberger and NSERC through CRD project No. 514472-17. This support is gratefully acknowledged. I.M. gratefully acknowledges support from the University of British Columbia via a university FYF award.

Author contribution A. Fedorov performed theoretical analysis and wrote the main manuscript text. N. Palchekovskaya performed numerical simulation and prepared figures. All authors reviewed the manuscript.

Data Availability By contacting the corresponding author.

Code availability Downloadable code is at: https://github.com/ivonneleonor/ViscoplasticFluids_with_NewAlgorithm_using_GPU_V2.0.

Declarations

Conflict of interest The author declares that they have no conflict of interest.

Ethical approval Not applicable.

References

1. Balmforth, N.J., Frigaard, I.A., Ovarlez, G.: Recent developments in viscoplastic fluid mechanics. Annu. Rev. Fluid Mech. **46**, 121–146 (2014)

2. Frigaard, Ian: Simple yield stress fluids. Curr. Opin. Colloid Interface Sci. **43**, 80–93 (2019)
3. Trémolières, R., Lions, J.-L., and Glowinski, R.: Numerical analysis of variational inequalities. Elsevier (1976)
4. Glowinski, R., and Wachs, A.: On the numerical simulation of viscoplastic fluid flow. In Handbook of Numerical Analysis, vol. 16, pp. 483–717. Elsevier (2011)
5. Saramito, P., Wachs, A.: Progress in numerical simulation of yield stress fluid flows. Rheol. Acta **56**(3), 211–230 (2017)
6. Saramito, P.: A damped Newton algorithm for computing viscoplastic fluid flows. J. Nonnewton. Fluid Mech. **238**, 6–15 (2016)
7. Dimakopoulos, Y., Makrigiorgos, C., Georgios, C., Tsamopoulos, J.: The PAL (Penalized Augmented Lagrangian) method for computing viscoplastic flows: a new fast converging scheme. J. Non-Newtonian Fluid Mech. **256**, 23–41 (2018)
8. Treskatis, T., Roustaei, A., Frigaard, I., Wachs, A.: Practical guidelines for fast, efficient and robust simulations of yield-stress flows without regularisation: a study of accelerated proximal gradient and augmented Lagrangian methods. J. Nonnewton. Fluid Mech. **262**, 149–164 (2018)
9. Nelson, E.B.: Well cementing. Schlumberger Educational Services (1990)
10. Fitt, A.D., Please, C.P.: Asymptotic analysis of the flow of shear-thinning foodstuffs in annular scraped heat exchangers. J. Eng. Math. **39**(1), 345–366 (2001)
11. Schoof, C., Hewitt, I.: Ice-sheet dynamics. Annu. Rev. Fluid Mech. **45**, 217–239 (2013)
12. Bittleston, S., Ferguson, J., Frigaard, I.: Mud removal and cement placement during primary cementing of an oil well-laminar non-Newtonian displacements in an eccentric annular Hele-Shaw cell. J. Eng. Math. **43**, 229–253 (2002)
13. Pelipenko, S., Frigaard, I.A.: On steady state displacements in primary cementing of an oil well. J. Eng. Math. **46**, 1–26 (2004)
14. Pelipenko, S., Frigaard, I.: Visco-plastic fluid displacements in near-vertical narrow eccentric annuli: prediction of travelling-wave solutions and interfacial instability. J. Fluid Mech. **520**, 343–377 (2004)
15. Pelipenko, S., Frigaard, I.: Two-dimensional computational simulation of eccentric annular cementing displacements. IMA J. Appl. Math. **69**(6), 557–583 (2004)
16. Carrasco-Teja, M., Frigaard, I., Seymour, B., Storey, S.: Viscoplastic fluid displacements in horizontal narrow eccentric annuli: stratification and travelling wave solutions. J. Fluid Mech. **605**, 293 (2008)
17. Carrasco-Teja, M., Frigaard, I.A.: Displacement flows in horizontal, narrow, eccentric annuli with a moving inner cylinder. Phys. Fluids **21**(7), 073102 (2009)
18. Carrasco-Teja, M., Frigaard, I.A.: Non-Newtonian fluid displacements in horizontal narrow eccentric annuli: effects of slow motion of the inner cylinder. J. Fluid Mech. **653**, 137 (2010)
19. Maleki, A., Frigaard, I.: Primary cementing of oil and gas wells in turbulent and mixed regimes. J. Eng. Math. **107**(1), 201–230 (2017)
20. Maleki, A., Frigaard, I.A.: Turbulent displacement flows in primary cementing of oil and gas wells. Phys. Fluids **30**(12), 123101 (2018)
21. Maleki, A., and Frigaard, I.A.: Rapid classification of primary cementing flows. Chem. Eng. Sci. p. 115506 (2020)
22. Renteria, A., Maleki, A., Frigaard, I., Lund, B., Taghipour, A., and Ytrehus, J.-D.: Displacement efficiency for primary cementing of washout sections in highly deviated wells. In SPE Asia Pacific oil and gas conference and exhibition, pp. 137. Society of Petroleum Engineers (2018)
23. Glowinski, R.: Lectures on Numerical Methods for Non-linear Variational Problems. Springer, Cham (2008)
24. NVIDIA°. CUDA C programming guide, version 10.1. NVIDIA° Corp, (2019)
25. NVIDIA°. PGI compiler user's guide, version 2019. NVIDIA° Corp, (2019)
26. Pelipenko, S., and Frigaard, IA.: Mud removal and cement placement during primary cementing of an oil well–part 2; steady-state displacements. J. Eng. Math. 48(1): 1–26, (2004d)