Multiscale Simulation of Polymer Flooding with Shear Effects

Sindre T. Hilden · Olav Møyner · Knut-Andreas Lie · Kai Bao

Abstract Multiscale methods have been developed as an alternative approach to upscaling and to accelerate reservoir simulation. The key idea of all these methods is to construct a set of prolongation operators that map between unknowns associated with cells in a fine grid holding petrophysical properties and unknowns on a coarser grid used for dynamic simulation. Herein, we extend one such method—the multiscale restricted-smoothed basis (MsRSB) method—to polymer flooding including shear thinning (and thickening) effects, which gives highly nonlinear fluid models that are challenging to simulate. To this end, we first formulate a sequentially-implicit solution procedure for polymer models with non-Newtonian rheology. By treating the implicit velocity dependence of the viscosities in an inner iteration loop, we obtain a formulation that appears to be more robust and stable than the standard fully-implicit approach. We then use a general algebraic multiscale framework to formulate an efficient and versatile multiscale solver. The unique feature of the MsRSB method is how the prolongation operators are constructed. By using restricted smoothing much in the same way as in smoothed aggregation multigrid methods, one gets a robust and flexible method that enables coarse partitions and prolongation operators to be constructed in an semi-automated manner even for highly complex geo-cellular models with high media contrasts and unstructured cell connections. By setting iterative tolerances appropriately, the resulting iterative multiscale solver can be set to compute mass-conservative approximations to the sequential or fully-implicit solution to arbitrary accuracy, and hence be used to trade accuracy for efficiency.

We first verify the sequential solution procedure and multiscale solver against a well-established commercial simulator on a test case with simple geometry, highly heterogeneous media properties, and strongly nonlinear fluid behavior. Next, the sequential fine-scale and multiscale solvers are validated on a synthetic simulation model of a shallow-marine reservoir. Here, the computational time is dominated by the pressure solves, and 5–8 times speedup is observed when replacing the fine-scale pressure solver by the iterative multiscale method. We also demonstrate the flexibility of the method by applying it to model with unstructured polyhedral cells that adapt to well positions and faults.

1 Introduction

Geological reservoir models are often made with much higher spatial resolution than what can be utilized in a conventional reservoir simulator. Some form of model upscaling is therefore necessary, which
inevitably introduces two forms of errors: (i) *homogenization errors* induced by neglecting subscale heterogeneities, and (ii) *truncation errors* induced by lower resolution in the numerical discretization. Various types of multiscale methods have been proposed over the past two decades to reduce homogenization errors and to overcome the limitations of traditional upscaling methods, see [17, 4, 8, 18, 29, 12]. Some of these methods have been developed to a point where they can handle realistic reservoir models [22, 40, 2, 33, 34, 42, 20, 11]. The key idea of such multiscale methods is to construct a set of prolongation operators (or basis functions) that map between unknowns associated with cells in a grid that represents the petrophysical properties of the reservoir and unknowns on a coarser grid used for dynamic simulation. The prolongation operators are computed numerically by solving localized flow problems, much in the same way as for flow-based upscaling methods, and can be used to construct a reduced coarse-scale system of flow equations that describe the macro-scale displacement driven by global forces. Unlike effective parameters, the multiscale basis functions have subscale resolution which ensures that fine-scale heterogeneity is correctly accounted for in a systematic manner. Linear combinations of basis functions will only resolve a linearized elliptic part of the system, and most multiscale methods therefore employ some kind of residual iteration to eliminate localization errors and account for nonlinear parabolic flow effects.

Multiscale methods have so far predominantly been used to study single-phase flow or secondary recovery methods. The latter are characterized by a combination of spreading and self-sharpening waves and are hence somewhat forgiving towards numerical truncation errors. The simulation of EOR methods, on the other hand, is much more sensitive to grid resolution. The reason for this is a complex interplay between the transport of chemical substances and how these substances affect the flow by changing the properties of the fluids and the surrounding rock. The transport of chemical substances is largely a linear process and thus strongly affected by numerical diffusion. The effects chemicals have on fluid and rock properties are generally highly nonlinear and highly sensitive to threshold parameters that determine sharp transitions between regions of very different behavior. For polymer flooding, in particular, the polymer fronts are not self-sharpening like water fronts. High numerical resolution is therefore required to limit numerical diffusion which would otherwise biases or deteriorates simulation results.

The high grid resolution needed to accurately account for polymer effects, will in most cases make field-scale simulation of offshore polymer flooding computationally infeasible by traditional reservoir simulators. To overcome this problem, one can either reduce the computational time through massive parallelization, use dynamic gridding strategies [16], and/or reduce the overall computational cost by exploiting weak couplings in the flow equations to formulate efficient sequential solution procedures. One example is streamline simulation [3, 10], another example is methods based on optimal cell ordering and identification of connected components in the transport equations [39], which may provide order-of-magnitude speedup compared with standard Newton–Raphson type solvers for the saturation-concentration part of the problem, see [27, 28]. In many cases, resolving the pressure component of the system equation is the part that contributes most to the overall computational cost. Nested gridding [13, 5] solves the pressure system on a coarse grid and has previously been applied to polymer flooding [6]. However, in contrast to the multiscale methods, nested gridding does not offer a consistent way of reducing the fine-scale residual. Herein, we will therefore investigate to what extent one can utilize a state-of-the art multiscale method, the multiscale restricted-smoothed basis (MsRSB) method [36, 38, 37], to reduce homogenization errors and accelerate the simulation of polymer flooding with shear-thinning and thickening effects. Truncation errors will be reduced by solving the saturation-concentration part of the problem on the underlying geological grid.

To enable the use of the multiscale method, we first formulate a sequential solution procedure for flow equations describing polymer flooding with shear thinning/thickening effects. The crucial part of this algorithm is to solve the pressure equation using a nested iteration approach in which the highly nonlinear effects of the non-Newton fluid rheology are accounted for by an inner Newton iteration. The resulting method appears to be more stable and robust than the standard fully-implicit approach. We can then easily adapt the iterative MsRSB method to the outer pressure equation, even though this has a different type of nonlinearity than what has previously been studied with multiscale methods. By using the multiscale solver along with an inexpensive smoother to efficiently reduce the fine-scale error, we can compute an approximate pressure field that satisfies a prescribed tolerance, and subsequently reconstruct a set of mass-conservative fluxes that can be used to evolve the saturation and polymer concentration on the original fine grid. The result is a flexible method that can either be used as a robust upscaling-downscaling method, or as an accelerated iterative solver that is guaranteed to produce a conservative flux field irregardless of whether the fine-scale
residual is reduced to machine precision (exact solver) or the iterations are terminated at a prescribed tolerance (approximate solver). To the best of our knowledge, the method presented herein is the first multiscale method that has been applied to account for non-Newtonian fluid rheology.

Through a series of numerical experiments, we verify and validate the multiscale method. First, we verify the correctness of the sequential formulation and our multiscale solver against Schlumberger’s ECLIPSE 100 for idealized test cases with heterogeneity sampled from the SPE 10 benchmark [9]. Then, the multiscale solver is validated on a model of a shallow-marine reservoir by showing that it can effectively resolve the highly nonlinear flow physics of a shear-thinning polymer flooding on a model with industry-standard complexity. Last, but not least, we demonstrate the geometrical flexibility of the MsRSB method by applying it to a test case with general polyhedral cells that adapt to wells and faults. Through the different examples, we show that the multiscale method can be set to reproduce the results of a fully-implicit simulator, but also can be used to trade accuracy for computational efficiency by computing approximate flux fields that are mass-conservative and satisfy a prescribed residual tolerance.

2 Model Description

As our basic flow model, we consider an immiscible, two-phase, three-component, black-oil system that includes the flow effects offered in contemporary commercial simulators. This includes adsorption of polymer onto the reservoir rock, fluid and rock compressibilities, permeability and porosity reduction, a Todd–Longstaff model for the mixing of polymer into water, and non-Newtonian fluid rheology for the diluted polymer. Understanding the rheological behavior of the polymer-water mixture on a reservoir scale is particularly important to successfully design and evaluate an offshore polymer flooding project. In most cases, the diluted polymer solution is a pseudoplastic (non-Newtonian) fluid, whose apparent viscosity decreases when subjected to sufficiently high shear rates, which causes the polymer molecules to uncoil, untangle, and align and elongate with the flow direction. Shear-thinning behavior is beneficial in that it improves injectivity by lowering the viscosity in the near-well zone where shear rates are high. As the injected fluids move away from the well, shear rates decline and viscosity increases, gradually establishing the desired mobility control in terms of a more stable and sharper displacement front. The disadvantage of a shear-thinning fluid is that it has a higher tendency to flow in high-permeable regions, which may impair sweep efficiency and hence reduce attainable recovery and/or lead to significantly higher operation and processing costs. Diluted polymer solutions may also exhibit shear-thickening (pseudodilatant) behavior, which can improve sweep efficiency but reduces injectivity.

The primary physical variables in the mathematical model are the oil and water pressures, \( p_o \) and \( p_w \), the oil and water saturations, \( S_o \) and \( S_w \), and the polymer concentration \( C_p \). The concentration is given in units of mass per volume of water. It is assumed that the presence of polymer changes the properties of the water, but does not influence the properties of the oil phase. The conservation equations for oil, water, and water with polymer are

\[
\frac{\partial}{\partial t} (\phi_b S_o) + \nabla \cdot (b_o v_o) = b_o q_o, \tag{1a}
\]

\[
\frac{\partial}{\partial t} (\phi_w S_w) + \nabla \cdot (b_w v_w) = b_w q_w, \tag{1b}
\]

\[
\frac{\partial}{\partial t} (\phi b_w (1 - S_{ipw}) S_w C_p) + \nabla \cdot (b_w C_p v_w) = b_w q_p, \tag{1c}
\]

with the closing relation \( S_w + S_o = 1 \). The Darcy velocities are given by the multiphase extension of Darcy’s law

\[
v_o = -\frac{k_{ro}}{\mu_o} K (\nabla p_o - g \rho_o \nabla z), \tag{2a}
\]

\[
v_w = -\frac{k_{rw}}{\mu_{w,ct} R_b} K (\nabla p_w - g \rho_w \nabla z), \tag{2b}
\]

\[
v_p = -\frac{k_{pw}}{\mu_{p,ct} R_b} K (\nabla p_w - g \rho_w \nabla z). \tag{2c}
\]

The porosity \( \phi \), absolute permeability \( K \), phase pressures \( p_{\alpha} \), fluid densities \( \rho_{\alpha} \), and inverse formation volume factors \( b_{\alpha} = 1/B_{\alpha} \) which relate surface and reservoir volumes of fluid phase \( \alpha \) are familiar
from the two-phase black-oil equations, where \( \alpha = \{ o, w \} \). The inclusion of polymer in the reservoir has a number of different effects on the flow, which is modeled through a number of parameters related to the polymer: \( R_k \) is the permeability reduction factor, \( S_{I.p} \) denotes the inaccessible pore volume, \( C_a \) is the amount of polymer adsorbed on the rock surface, \( \rho_r \) is the rock density, and \( \phi_{ref} \) a reference porosity. The details of these parameters, and the physical effects they model, are described in more detail below.

**Viscosity influence.** Adding polymer to the water results in an increased viscosity of the solution. The effective viscosities of water and polymer are denoted by \( \mu_{w,eff} \) and \( \mu_{p,eff} \), respectively. These effective viscosities are defined using the Todd–Longstaff mixing model \([44]\). The viscosity of a fully mixed solution of water and polymer is denoted by \( \mu_m(C_p) \), and the effective polymer viscosity is given by

\[
\mu_{p,eff} = \mu_m(C_p) \omega \mu_p^{1-\omega},
\]

where \( C_{p,max} \) is the maximum possible polymer concentration, \( \mu_p = \mu_m(C_{p,max}) \), and \( \omega \in [0, 1] \) is the Todd–Longstaff mixing parameter. For \( \omega = 1 \), the polymer solution and the water are fully mixed, whereas for \( \omega = 0 \) if the polymer solution is completely segregated from the pure water. Values of \( \omega \) in between 0 and 1 model partial mixing. The viscosity of partially mixed water is defined as

\[
\mu_{w,e} = \mu_m(C_p) \omega \mu_w^{1-\omega}.
\]

The effective water viscosity \( \mu_{w,eff} \) is then given by

\[
\frac{1}{\mu_{w,eff}} = \frac{1 - C_p/C_{p,max}}{\mu_{w,e}} + \frac{C_p/C_{p,max}}{\mu_{p,eff}}.
\]

We also define a viscosity multiplier \( m_\mu \), such that

\[
m_\mu(C_p) = m_{\mu}(C_p) \mu_w.
\]

where \( m_{\mu}(C_p) \) is given as input, instead of specifying \( \mu_m(C_p) \) directly. We also note a useful relation between the water and polymer flux, writing \( v_p = m(C_p)v_w \), where \( m(C_p) \) can be expressed as

\[
m(C_p) = \frac{\mu_{w,eff}}{\mu_{p,eff}} = \left[ \left( 1 - \frac{C_p}{C_{p,max}} \right) \left( \frac{\mu_p}{\mu_w} \right)^{1-\omega} + \frac{C_p}{C_{p,max}} \right]^{-1}.
\]

**Inaccessible pore volume.** Polymer molecules are large relative to the water and hydrocarbons in the reservoir and a portion of the pore volume may therefore be inaccessible to the polymer, as they are unable to enter the smallest pores. This may result in an accelerated polymer flow, and this effect is included in (1c) with the value \( S_{I.p} \), which is the fraction of the pore volume not accessible to polymer, commonly referred to as the inaccessible pore volume or dead pore space.

**Adsorption.** Adsorption is the effect of polymer molecules attaching to the rock surface. This will cause the polymer concentration in the solution to decrease, and consequently so will the viscosity. The adsorption function \( C_a(C_p) \) gives the relationship between the polymer concentration and the amount of adsorbed polymer. Herein, we assume the adsorption effect is instantaneous and reversible, such that polymer detaches from the rock if the concentration decreases.

**Permeability reduction.** The mobility of a water-polymer solution is reduced because of an increased viscosity, but some types of polymer also experience a reduction in the permeability \([23]\). This can be explained by the adsorption, as the large polymers attached to the rock reduces the effective pore size for the flowing polymer solution. The permeability reduction effect is modeled by the reduction factor \( R_k \), defined by

\[
R_k(C_p) = 1 + (R_{RF} - 1) \frac{C_a(C_p)}{C_{a,max}},
\]

where \( R_{RF} \geq 1 \) is called the residual resistance factor and \( C_{a,max} \) is the maximum adsorption.
Non-Newtonian behavior. A fluid is called Newtonian if there is a linear relationship between the shear stress and the strain rate (the change in shear strain). If this relationship is not linear, the fluid is called non-Newtonian. A pseudodilatant, or shear-thickening, fluid increases resistance with increasing applied stress, whereas a pseudoplastic, or shear-thinning, fluid decreases resistance with increasing stress. For a polymer solution at low shear rates, there is a linear relationship between the shear stress and the shear rate, and the solution is a Newtonian fluid. However, at higher shear rates, the viscosity decreases. This shear-thinning effect in polymer solutions is caused by the uncoiling and unsnagging of polymer chains when they are elongated in shear flow [23]. Throughout the reservoir, the flow-rates are usually low and so are the shear stresses. But near the injection well, the shear rates can be quite high and the effect of shear-thinning can consequently be large. The viscosity of a polymer solution is also highly sensitive to the salinity of the water, but this is not discussed herein.

Mathematical formulation of shear-thinning. The shear-thinning effect is included by modifying the effective viscosities by a shear-thinning multiplier $M_{sh}$ such that the resulting viscosities becomes

$$\mu_{w,sh} = \mu_{w,eff}M_{sh}, \quad \text{and} \quad \mu_{p,sh} = \mu_{p,eff}M_{sh}. $$

This multiplier is computed as follows [43]: Given the discrete water flux $v_w$, computed from the discretized form of equation (2b) without any shear effect, the water velocity is defined on each face of the discrete grid as

$$u_w = \frac{v_w}{\phi A},$$

where the porosity $\phi$ is taken as the average of the two neighboring cells, and $A$ is the face area. The product $\phi A$ is then the available area for the fluids to flow through each particular face. Using the water velocity, the shear-modified water velocity $u_{w,sh}$ is calculated using the implicit equation

$$u_{w,sh} = u_w \frac{\mu_{w,eff}}{\mu_{w,sh}}$$

where $\mu_{w,sh} = \mu_{w,sh}(u_w)$ is the shear-modified water viscosity defined by

$$\mu_{w,sh} = \mu_{w,eff}M_{sh} = \mu_{w,eff}\left(1 + \frac{(m\mu - 1)m_{sh}}{m_\mu}\right).$$

Here, $m_\mu = m_\mu(C_p)$ is the viscosity multiplier defined in (4), and $m_{sh} = m_{sh}(u_w)$ is a shear multiplier, which depends on the water velocity, and is given as input. Solving (6) gives the shear-viscosity multiplier $M_{sh}$, and so both modified viscosities can be computed, and consequently also the modified phase fluxes, which are then given by

$$v_{w,sh} = \frac{v_w}{M_{sh}}, \quad \text{and} \quad v_{p,sh} = \frac{v_p}{M_{sh}}.$$

3 Sequential Formulation

To apply the multiscale method, we need to formulate our equations (1) in a sequential form. In the following, we explain the splitting of the equations (1) without polymer shear-thinning, and then afterwards, we address how the shear-thinning calculation is included in the sequential solver. Wells are left out of the equations below to ease readability. For details on how wells are included in our implementation, we refer to [37].

3.1 Pressure equation

To find an equation for the pressure only, we start by discretizing the conservation equations (1) written in the backward (implicit) Euler method to obtain the following semi-discrete equations, which are left out of the equations below to ease readability. For details on how wells are included in our implementation, we refer to [37].
Here, superscript denotes the discrete time and \( \Delta t = t^{n+1} - t^n \) is the time step. To remove the dependence of the saturations and concentration at the next time step, we sum these equations as

\[
R_t = \beta_o R_o + \beta_w R_w + \beta_p R_p = 0,
\]

where we assume there exist \( \beta \)-factors such that \( R_o \) only depends on the pressure at time \( n + 1 \). We construct the pressure equation in the same way as is usual in the two-phase case, and so using factors

\[
\beta_o = \frac{1}{b_o^n + 1}, \quad \beta_w = \frac{1}{b_w^{n+1}}, \quad \text{and} \quad \beta_p = 0,
\]

we get

\[
R_t = \frac{\phi^{n+1}}{\Delta t} - \frac{\phi^n}{\Delta t} \left[ \frac{(b_o S_o)^n}{b_o^n + 1} + \frac{(b_w S_w)^n}{b_w^{n+1}} \right] + \nabla \cdot \left( \frac{(b_o^{n+1}) v_o^{n+1/2}}{b_o^n + 1} \right) + \nabla \cdot \left( \frac{(b_w^{n+1}) v_w^{n+1/2}}{b_w^{n+1}} \right) = 0,
\]

where we have used the relation \( S_o + S_w = 1 \) to eliminate the saturations at the next time step. This is the standard IMPES pressure equation where the effect of polymer comes from the water viscosity. The fluxes in \( R_t \) are given by

\[
v_o^{n+1/2} = -\frac{K r o (S_o^n)}{\mu_o (p^{n+1})} [\nabla p_o^{n+1} - \rho_o \nabla z], \quad (8a)
\]

\[
v_w^{n+1/2} = -\frac{K r w (S_w^n)}{\mu_w (p^{n+1}, C_p^n) R_k (C_p^n)} [\nabla p_o^{n+1} - \nabla p_{cow} (S_w^n) - \rho_w \nabla z]. \quad (8b)
\]

We denote the phase fluxes with the discrete time \( n + 1/2 \) to indicate that they are computed using pressures at time \( n + 1 \) and saturations and concentration at time \( n \), and thus can be thought of as intermediate fluxes. We have now obtained the equation

\[
R_t = R_t (p_o^{n+1}, p_w^n, S_w^n, C_p^n) = 0, \quad (9)
\]

which can be solved for \( p_o^{n+1} \) using the Newton–Raphson method. Here, we have written the equation in terms of the oil pressure, but we could equally well have solved for the water pressure.

The pressure equation (9) is discretized by a standard two-point flux-approximation scheme with upstream mobility weighting of all terms that depend on saturations and polymer concentrations; see e.g., [37] for more details. The resulting nonlinear system is linearized and solved by a Newton–Raphson method. Shear-thinning effects will generally induce strong nonlinearities in the discrete system since the effective water viscosity now depends on the water velocity \( v_w \) and the polymer concentration \( C_p \). In the linearized pressure equation of the sequential formulation, the dependence on \( C_p \) is treated explicitly while the dependence on \( u_w \) is treated implicitly. Moreover, to reduce the nonlinearity of the pressure system, the shear-thinning effect is accounted for in an inner iteration. That is, we do not account for the fact that the water viscosity depends on water velocity when linearizing the global pressure equation, but rather use another Newton loop to linearize and solve (6) to a prescribed tolerance for each (outer) pressure iteration. Once the overall system is solved, the pressure \( p_o^{n+1} \) at the next time step is known, as well as the fluxes (8) and all fluid viscosities, which all are considered to be constant during the subsequent transport step.

### 3.2 Transport equation

To evolve the saturations and the concentration, we first find expressions for the phase fluxes in terms of the total flux \( v_T = v_o + v_w \). From (2), we compute \( v_T \) and rearrange to find an expression for \( \nabla p_o \) in terms of \( v_T \). This is then inserted back into each phase flux (2) to obtain phase fluxes expressed in terms of the total flux. From the pressure solution, we are given the total flux \( v_T^{n+1} = v_o^{n+1/2} + v_w^{n+1/2} \). Each phase flux at the discrete time \( n + 1 \) is then given by

\[
v_o^{n+1} = f_o \left[ v_T^{n+1} + K \lambda_o (\rho_o g \nabla z - \rho_w g \nabla z) \right],
\]

\[
v_w^{n+1} = f_w \left[ v_T^{n+1} + K \lambda_o (\rho_w g \nabla z - \rho_o g \nabla z) \right],
\]

\[
v_p^{n+1} = f_p \left[ v_T^{n+1} + K \lambda_o (\rho_w g \nabla z - \rho_o g \nabla z) \right].
\]
where \( f_\alpha = \frac{\lambda_\alpha}{(\lambda_\alpha + \lambda_w)} \) is the fractional flow, and \( \lambda_\alpha \) are the mobilities given by

\[
\lambda_o = \frac{k_{rw}}{\mu_o}, \quad \lambda_w = \frac{k_{rw}}{\mu_w,_{\text{eff}}R_k}, \quad \lambda_p = \frac{k_{rw}}{\mu_p,_{\text{eff}}R_k}.
\]

The capillary pressure is left out of the above flux expressions just for readability. The resulting transport equation is discretized using a standard finite-volume method. To evaluate the mobilities \( \lambda_\alpha \) on the faces of each grid cell, we use potential ordering to determine the upstream weighting [7], which in most cases coincides with the upstream weighting used in the pressure solver.

We now choose two of the three conservation equations (7) to solve for, and then insert the above phase fluxes into these equations. Through numerical experiments, we have in some cases experienced convergence problems using conservation of water together with the polymer equation. We therefore choose to solve the oil and polymer equations, and then let the water fill up the remaining pore space. For the temporal discretization, we can either use the backward or forward Euler method, which result in a sequentially-implicit or a IMPES-type simulator, respectively. To allow for larger time-steps, we have chosen an implicit discretization. That is, we solve the system

\[
\mathcal{R}_s = \mathcal{R}_s(S_{w}^{n+1}, C_p^{n+1}, S_w^n, C_p^n, v_T) = [\mathcal{R}_o, \mathcal{R}_p]^T = 0,
\]

where \( \mathcal{R}_o \) is given in (7a), and \( \mathcal{R}_p \) in (7c), but now with the phase fluxes expressed in terms of the total flux. Herein, we use a standard Newton–Raphson method to solve (10), but remark that it is generally possible to use a topological ordering of grid cells according to total velocity to develop nonlinear solvers that are significantly more efficient for cases with small or moderate contrasts in water and oil densities [27,28].

3.3 Solving the nonlinear problem

For clarity, let us summarize our sequentially-implicit method. Assume the state \((p_o^n, S_w^n, C_p^n)\) is known at discrete time \( n \). One time-step is then performed using the following steps:

1) Solve the pressure equation (9) by the Newton–Raphson method. Let subscript \( i \) denote the iteration count in the Newton-loop. Set initial guess \( p_{i=0} = p_o^n \) (i.e., the oil pressure at time \( n \)), and then iterate the following steps:
   a) Construct \( \mathcal{R}_o(p_i) \) and its Jacobian \( J_p \). To this end, we have to perform a nested Newton-loop to compute the shear-thinning multipliers by solving (6).
   b) Find the pressure increment \( \Delta p \) by solving
   \[
   -J_p \Delta p = \mathcal{R}_s(p_i).
   \]
   c) Update pressure by \( p_{i+1} = p_i + \Delta p \).
   d) Check the following convergence criterion
   \[
   \|p_{i+1} - p_i\|_{\infty} \leq \varepsilon_p (\max p_i - \min p_i),
   \]
   for some prescribed tolerance \( \varepsilon_p \). If we have not converged, set \( i \leftarrow i + 1 \) and go to (a).
   
   At convergence, set the pressure at the next time-step, \( p_o^{n+1} = p_{i+1} \), and compute the total flux \( v_T \) by using phase-fluxes (8).

2) Given the total flux \( v_T \), construct and solve the transport equation (10) using Newton–Raphson iterations. If we denote \( s = (S_w, C_p)^T \), then set \( s_{i=0} = s^n \), and perform iterations
   \[
   -J_s \Delta s = \mathcal{R}_s(s_i), \quad s_{i+1} = s_i + \Delta s,
   \]
   where \( J_s \) is the Jacobian of \( \mathcal{R}_s(s_i) \). For the oil equation, the convergence is defined using both a maximum error and a total error,
   \[
   \|\Delta t \mathcal{R}_o\|_{\phi} \leq \varepsilon_{V, b_{\phi}^{\text{avg}}}, \quad \|\Delta t \mathcal{R}_o\|_{\phi} \leq \varepsilon_{M, b_{\phi}^{\text{avg}}},
   \]
   whereas for the polymer equation, we use an incremental tolerance,
   \[
   \|c_{i+1} - c_i\|_{\infty} \leq \varepsilon_c C_{p,\text{max}}.
   \]
   At convergence, the time-step is complete, and we proceed to the next time-step.
Note that in the above we are using an increment tolerance for the pressure. In the case that the linear solver used to solve the pressure equation is exact, the increment convergence criterion is equivalent to a residual criterion, as the pressure update is unconstrained and proportional to the residual magnitude in each cell (11). However, since we will also solve the problem using the approximate multiscale solver, we require a convergence criterion that will allow convergence with inexact pressure updates, i.e., \(-J_p \Delta p \approx R_c(p_i)\).

If required, the sequential implicit procedure outlined above can be extended with an outer iteration over the pressure and transport steps to ensure that the overall fine-scale residual is reduced to within a prescribed tolerance at the end of the time step. The result is what is sometimes referred to as an iterated fully-implicit solution procedure.

4 Multiscale Pressure Solution

Solving the linearized pressure system (11) is computationally expensive and becomes the largest contributor to the total runtime for large systems. Instead of solving (11) directly, we compute an approximate solution using an iterative multiscale solver. To this end, we will use the MsRSB method [36,38,37], which has proved to be a particularly versatile method that can robustly handle both the complexity in flow physics and geological description seen in real-life simulation models. The method incorporates concepts and insights from previous multiscale methods, including algebraic [46,30,45] and iterative [14,31,15,47,45] formulations, as well as flexibility with respect to fine grids and coarse partitions [1,2,26].

As explained briefly in the introduction, the basic form of the MsRSB method relies on three key ingredients: a coarse partition, a prolongation operator that maps unknowns from the fine grid to the coarse grid, and a restriction operator that maps the flow equations defined on the fine grid to a reduced system of flow equations defined over the coarse partition. Let the fine-scale grid consist of \( n \) cells \( \{C_i\} \). The coarse grid is then defined as a partition of the fine-scale grid into \( m \) coarse blocks \( \{B_j\} \), where each block \( B_j \) is a connected set of fine-scale cells defined so that each cell \( C_i \) only belongs to one block \( B_j \). Then, the restriction operator is represented as a matrix \( R \in \mathbb{R}^{m \times n} \), which is a map from the fine grid to the coarse grid. Here, we let \( R \) be the control-volume summation operator,

\[
R_{ji} = \begin{cases} 
1, & \text{if fine cell } i \text{ is part of coarse block } j, \\
0, & \text{otherwise.} 
\end{cases}
\]

Similarly, we define the prolongation operator as a matrix \( P \in \mathbb{R}^{n \times m} \), which maps from the coarse grid to the fine grid. The construction of the prolongation operator \( P \) will be described in detail below. First, however, we show how \( R \) and \( P \) can be used to construct a coarse system.

4.1 Coarse system

In each pressure Newton-iteration, we need to solve the linearized pressure system (11). To simplify the notation, we write this system as

\[
J \Delta p = r. 
\]

Instead of solving this linear system directly, we wish to solve a system on the coarse scale to create some approximate coarse solution \( \Delta p_c \). If we obtain such a solution, it can be mapped back to the fine grid using the prolongation operator to obtain \( \Delta p_f = P \Delta p_c \), and we then hope that \( \Delta p_f \) is a good approximation to the true solution \( \Delta p \).

Replacing \( \Delta p \) with the approximation \( \Delta p_f \) in (13), and then applying the restriction matrix \( R \) on both sides, we obtain the following coarse-scale equation simply as

\[
(R J P) \Delta p_c = Rr \quad \leftrightarrow \quad J_c \Delta p_c = r_c. 
\]

which is inexpensive to solve compared with the original fine-scale system provided the coarse partition induces a reasonable reduction in the number of unknowns.
4.2 Basis functions

We only give a very brief description of the construction of the basis functions and refer to [36,38,37] for details. The prolongation matrix $P$ is created from a set of localized basis functions, with one basis function associated with each block in the coarse partition, that together form a partition of unity of the domain. In the MsRSB method, the basis functions are computed numerically in an iterative process which is illustrated in Figure 1. Initially, the value of each basis function is set to unity inside the corresponding coarse block, and zero outside. Through repeated iteration of the form,

$$P_{j}^{ℓ+1} = (I - \omega D^{-1}A)P_{j}^{ℓ}$$

the basis function associated with block $B_j$ is gradually smoothed inside the support region surrounding the coarse block until a convergence criterion is met. Here $P_j$ is the $j$th column of the prolongation operator, $A$ is the fine-scale discretization matrix for a variable-coefficient Poisson operator $\nabla \cdot (K\lambda \nabla)$ that represents the elliptic part of our flow equation, $D$ is a diagonal matrix containing the diagonal entries of the weakly diagonally dominant matrix $A$, and $\omega$ is a relaxation factor. The purpose of the iteration is to make $p$ algebraically smooth, i.e., reduce $\|AP\|_1$ as much as possible. Because each iteration modifies cell values based on the topological neighbors, the support of the basis functions will gradually increase and eventually cover the entire domain. The purpose of the support region is to limit the expansion of the basis function outside of the coarse grid block. The basis functions are constructed based on the initial fluid distribution. As we will see in the next section, their main purpose is to reduce long-range error components as part of an iterative solver. In practice it is therefore not that important that they fully comply with dynamic changes in the effective permeabilities. If needed, however, the basis functions can later be modified to account for transient flow phenomena by continuing to iterate a few extra steps with updated dynamic mobility changes. These updates are inexpensive compared with other multiscale formulations, which need to solve full localized flow problem to account for mobility changes.

4.3 Iterative multiscale solution

Using the prolongation operator introduced above, or any other prolongation operator defined by piecing together localized flow solutions, to map a coarse scale solution to an approximate fine-scale solution, we will inevitably introduce local errors in the fine-scale approximation. These high-frequency errors are easy to reduce by using a smoother, which can be combined with the multiscale solve to formulate an iterative solver. That is, instead of approximating the solution of (13) by a single coarse solution of (14) prolongated onto the fine grid, we iteratively compute a multiscale increment followed by a few smoothing steps to limit the error of our approximation and hence systematically drive the fine-scale residual toward zero, thus following the ideas of [46,31,47,45]

Let the solution at each multiscale iteration $ℓ$ be denoted by $x^ℓ$, which is an approximation for the solution $\Delta p$. At each iteration, define the defect as

$$d^ℓ = r - Jx^ℓ.$$

Let also $y^ℓ = S(d^ℓ)$ denote a smoother applied to the defect. One iteration then consists of applying the smoothed update, together with a coarse correction,

$$x^{ℓ+1} = x^ℓ + P \left( J^{-1} R(d^ℓ - Jy^ℓ) \right) + y^ℓ.$$

(16)
4.4 Multiscale solution of the nonlinear problem

Let us go back to the nonlinear problem to get an overview of the solution strategy. To aid the explanation, we refer to the flow chart in Figure 2 showing one time step of the nonlinear problem. From the previous time step, we move into the pressure Newton-loop. When assembling the residual equation and the Jacobian matrix, we need to solve for the shear-thinning multiplier, which is done in a nested Newton solver. Once the linearized system (11) is assembled, we do not solve it directly, but instead approximate the pressure update $\Delta p$ through a number of multiscale cycles (16). When the multiscale cycles have converged to within the prescribed linear tolerance, we have an approximation for the pressure update, and we can continue the nonlinear Newton-loop for the pressure until the nonlinear increment tolerance (12) is met.

The approximate fine-grid pressures computed by the multiscale iteration do not necessarily give a conservative flux field from (8) that can be applied in the transport step. After the convergence of the nonlinear pressure step, we therefore need to solve a set of local Neumann problems to reconstruct a conservative flux field. In this reconstruction step, all pressure-dependent properties used are evaluated using the $J$ matrix from the last multiscale iteration, i.e., the second-last pressure approximation, as discussed in [37], thereby making the problem linear and simpler to solve. Once the flux field is reconstructed, we compute the total flux and move on to solve for saturations and polymer concentration in the transport step. Then the time step is complete, and we can move on to the next time step.

5 Numerical Experiments

The polymer model and sequential solvers outlined above have been implemented using the Matlab Reservoir Simulation Toolbox (MRST) [25,21,24,35], which is an open-source toolbox for rapid prototyping of new computational methods and reservoir engineering workflows. In the following, we
report a few numerical experiments we have run to verify the sequential solution procedure and our implementation of the polymer model, to validate the multiscale method, and to assess its computational efficiency against AGMG 3.0 [41], which is a relatively simple aggregation-based algebraic multigrid solver. Since AGMG does not come with many adjustable parameters, we decided to also use the multiscale method as a black-box solver and not make any attempts to tweak its parameters to optimize performance.

5.1 Verification of shear-thinning in multiscale solver

To verify computation of non-Newtonian rheology with the multiscale method, we consider a 1D, homogeneous, 1000 m long model, which initially is partially filled with water ($S_w = 0.7$). Water with polymer is injected at a constant rate over the west boundary. We use relative permeabilities (17) with $n = 1.5$, no adsorption, and a mixing parameter $\omega = 1$. (The remaining fluid parameters are presented in Section 5.2.) We run the fully-implicit solver in MRST and the MsRSB solver with 50 cells and 1000 cells. For MsRSB, the coarse grid has 10 blocks. Water saturation and polymer concentration at the end of the simulations are shown in Figure 3. The excellent match between MsRSB and the fully-implicit solver indicates that MsRSB reproduces the influence of shear-thinning to within the accuracy of the fine grid. The example also shows the importance of sufficient grid resolution for a case without self-sharpening waves. With only 50 cells in the fine-scale grid, both the fully-implicit and the multiscale simulator fail to capture the oil bank ahead of the displacement front.

5.2 Verification against a commercial fully-implicit simulator

We compare our sequential solver with fine-scale or MsRSB pressure solver to the commercial simulator ECLIPSE 100 [43]. To set up a challenging test case, we consider two horizontal subsamples of Model 2 from the 10th SPE Comparative Solution Project [9], see Figure 4. Layer 35 from the Tarbert formation follows a lognormal distribution that is resolved relatively well by most multiscale methods. Layer 85 of the Upper Ness formation is fluvial and has very long correlation lengths and permeabilities with orders-of-magnitude differences between neighboring cells, which enforces small time steps and makes this layer challenging both for the fully-implicit and the multiscale simulators.

Parameters are set so that all polymer effects described earlier are included and affect the simulation; the Todd–Longstaff mixing parameter $\omega = 0.5$ is neither 0 nor 1, the $R_{RF}$-factor $R_{RF} = 1.1$.
Fig. 4: Horizontal permeability field from Layer 35 (Tarbert) and Layer 85 (Upper Ness) of the SPE 10 benchmark. Each layer consists of 60 × 220 cells covering an area of 1200 × 2200 ft².

Fig. 5: Fluid polymer properties used in the SPE 10 simulations. From the left; adsorption isotherm $C_a$, viscosity multiplier $m_v$ from (4), and shear multiplier $m_{sh}$ from (2).

is larger than unity, and dead pore space $S_{pv} = 0.05$ is larger than zero. Relative permeabilities for oil and water are

$$k_{ro}(S_o) = (1 - S)^n, \quad \text{and} \quad k_{rw}(S_w) = S^n, \quad \text{where} \quad S = \frac{S_w - S_{wir}}{1 - S_{wir}} - S_{wir}, \quad (17)$$

with $n = 2, S_{or} = 0.2,$ and $S_{wir} = 0.2$. There is no capillary pressure. To produce a significant effect of the shear-thinning, we set $\mu_o = 3\,\text{cP}$ and $\mu_w = 0.3\,\text{cP}$. The compressibility-factors for oil and water are $c_o = 5 \cdot 10^{-5}/\text{bar}$ and $c_w = 1 \cdot 10^{-6}/\text{bar}$, respectively, and the rock compressibility is set to $c_r = 1 \cdot 10^{-5}/\text{bar}$. The remaining polymer properties plotted in Figure 5 are synthetic, but based on values found in literature. The nonlinearity of the shear-thinning curve has been somewhat exaggerated to ensure notable effect in the simulations.

Two wells are placed in opposite corners of the domain, a rate-controlled injector at logical indices $(56, 10)$, and a pressure-controlled producer at $(5, 211)$. The domain is initialized with $S_w = 0.2$ everywhere. We consider three scenarios. The first is pure water injection. In the other two, we first inject water and either a Newtonian or a shear-thinning polymer for 0.1 PV1 (pore volumes), and then continue to inject pure water until a total of 0.5 PV1 have been injected. For the MsRSB pressure solver, we choose a tolerance of 0.01 and a coarse $6 \times 11$ grid with square blocks. Both simulators us a time step that a gradually ramps-up towards a maximum time step of five days.

At water breakthrough, it is not possible to distinguish the sequential MsRSB solution from the fully-implicit reference solution, as shown in Figure 6. (The sequential solution with fine-scale pressure solver is indistinguishable and hence not included for brevity). Because of the strong heterogeneity, the discrete system is relatively stiff, and so both the commercial simulator and our sequential simulator need to take small time-steps initially to converge. As the time-steps are small, there is close agreement between the sequential and fully-implicit simulators. We have observed that the sequential solver is able to converge using larger time steps than the fully-implicit simulators after the initial ramp-up, but choose to use the same maximum time step of five days for better comparison.

Figure 7 shows well curves for all three flow scenarios. The bottom-hole pressure (BHP) in the injector is unrealistically high since we have injected a solution of very low mobility in a rate-controlled injector. In reality, the near-well region would fracture or the pressure would be regulated in other ways. However, the main point we want to make is our ability to reproduce the reference solution for a case with very strong shear effects. Both injector BHP and oil-production rate are nearly identical. Looking closely at the well-curves, we can observe a slight difference in the oil production when shear-thinning is included. As the other production curves are indistinguishable, this could indicate that there are some small differences in the implementation-details of shear-thinning in the wells. We also see distinct differences between polymer flooding with and without shear-thinning. As expected, the
Fig. 6: Water saturation and polymer concentration at the point of water breakthrough for the simulations with polymer and shear-thinning effect in the SPE 10 example. The breakthrough occurs after injecting 0.23 PVI in Layer 35, and after 0.20 PVI in Layer 85.

Injector BHP is lower when shear-thinning is included since this reduces the viscosity of the injected solution. Thus, lower pressure is needed to obtain the given injection rate.

5.3 Accuracy and efficiency on realistic reservoir model

To validate our multiscale method and assess its computational efficiency, we consider a realistic model representing a shallow-marine environment generated by the SAIGUP project [32]; this particular realization can be downloaded using MRST [35]. The model consists of 78,720 active cells on an underlying logical grid of size $40 \times 120 \times 20$. Five injection wells and five production wells are placed in the model as shown in Figure 8. The injectors are completed in the lowest 12 layers, while the producers are completed in the top 14 layers. For the fluid properties, we assume densities $\rho_o = 850 \text{ kg/m}^3$ and $\rho_w = 1050 \text{ kg/m}^3$, and viscosities $\mu_o = 5 \text{ cP}$ and $\mu_w = 0.55 \text{ cP}$. The formation volume factors are computed as $B_o = B_o^0 \exp(-c_o(p - p_r))$, where $B_o^0 = 1.06$, $B_w^0 = 1.01$. The compressibility factors are $c_o = 6.7 \times 10^{-5} \text{ bar}^{-1}$ and $c_w = 4.2 \times 10^{-5} \text{ bar}^{-1}$ and the reference pressure is $p_r = 300 \text{ bar}$. The relative permeability and the capillary pressure used are shown in Figure 9. We use the same polymer properties as in the SPE 10 example above, only adjusting the Todd–Longstaff mixing parameter, which is now set to $\omega = 0.8$.

The domain is initialized with the irreducible water saturation $S_w = 0.1$ everywhere. All the injectors are rate controlled with a constant injection rate of 600 m$^3$/day, and all the producers operate at a constant pressure of 300 bar. For simplicity, we also let all injectors use the same polymer injection schedule. First, pure water is injected for 1 year, then polymer (at maximum concentration $C_{p,\text{max}}$) is injected for 250 days, followed by continued pure water injection until reaching a total reservoir time of 15 years.
Fig. 7: Well solution curves for the SPE 10 example. The fully-implicit reference solutions are plotted in solid lines, while the sequential solutions using MsRSB are plotted in dashed lines on top using slightly stronger colors.

Fig. 8: The SAIGUP model. Left: the permeability field in $x$-direction in units of milli-Darcy on a log-scale, also showing the location of the five injectors and five producers. Right: the METIS partitioning used by the MsRSB method.

Fig. 9: Relative permeability (left) and capillary pressure (right) for the SAIGUP model.
As in the SPE 10 example above, we ran three variations of the simulation: polymer injection with all polymer properties, polymer injection without shear-thinning effect, and finally a pure water injection. We use sequential solutions computed using a direct fine-scale pressure solver as reference, and compare with approximate solutions computed with the MsRSB solver. Normally, we would set a tolerance for the multiscale iterative solver, but to highlight certain behavior of the solver we only run a fixed number of multiscale cycles. The coarse grid used by MsRSB has 250 blocks and is constructed using METIS [19] with transmissibilities for connectivity, see Figure 8.

Figure 10 shows the solution at the end of simulation in the top-most layer in which both the injectors and producers are completed. We see how the polymer slugs have spread out from the injectors as rings, while water has broken through in all producers except P5. Injecting polymer gives a higher total oil production, distributed between the wells as higher production in P1 to P4 and somewhat lower production in P5. The increase in production is higher if shear-thinning is not included, since this effect lowers the mobility of the polymer solution in high-flow regions and hence reduces the desired effect of a more favorable mobility ratio, which in turn reduces the sweep efficiency.

As a simple strategy to trade accuracy for efficiency, we ran the MsRSB solver with a fixed number of multiscale-cycles instead of a prescribed tolerance. Even with 1–3 cycles only, the well curves computed with the fine-scale pressure solver and the MsRSB solver are virtually identical. Oil rates and water-cut are reported in Figure 11 for P1; the four other producers have equally good match between reference solution and the multiscale solution. Also, performing more multiscale cycles gives even closer match with the reference solution. When using only a few multiscale cycles to solve the linearized pressure equation, each linearized solution will not be very accurate and we hence should expect to use a higher number of iterations to reduce the residual of the nonlinear pressure equation below the prescribed tolerance. In all the multiscale simulations above, we used the same value $\varepsilon_p = 10^{-3}$ for the nonlinear pressure convergence criterion (12). Figure 12 compares the number of nonlinear pressure iterations throughout the simulations for the different simulations. As expected, the number of nonlinear iterations increases as the number of multiscale cycles goes down. Likewise, the sequential simulation with exact fine-scale solver requires less iterations than the approximate
MsRSB solver. Especially, we note an increase in the number of iterations after the polymer injection has stopped.

Even though computational efficiency and code optimization is not the focus of this study, it is still of interest to consider the runtime for the different simulations. Of particular interest is the time spent in the linear pressure solver, as this is where we observe the speedup from using the MsRSB method. The timing for each simulation is divided into four categories: time spent in linear pressure solver, reconstruction of mass-conservative fluxes, linear transport solver, and computational overhead coming from other parts of the code. As explained in more detail in [37], the MRST implementation has a significant overhead in the evaluation of fluid properties and assembly of linear systems that most likely can be eliminated or significantly reduced if the sequential simulators were implemented in a computer environment optimized for performance rather than prototyping flexibility. Hence, only the first three categories are shown in Figure 13. As expected, the simulation times are higher for polymer flooding than for pure water injection. Likewise, the runtime for polymer simulations with shear-thinning is somewhat higher than simulations without as a result of the extra calculations performed to compute the shear-thinning in each pressure nonlinear iteration. For each simulation, we observe similar difference among the solvers; the AGMG [41] solver needs to reduce the fine-scale residual to machine precision and hence spends a considerably longer time solving the fine-scale pressure system than the time it takes the multiscale solver to iterate a few cycles on the coarse pressure system and reconstruct approximate fine-scale flux fields. Looking only at the time for the pressure solution, the multiscale solver is between 6 and 8 times faster than the fine-scale solver.

5.4 Accuracy and efficiency on unstructured polygonal grids

In this example, we wish to investigate MsRSB and ILU0 as a preconditioner for GMRES (see [38]), validate that this iterative solver converges to the fully-implicit solution for a more complex grid topology, and observe how the solution changes with different tolerance levels.

We consider a 2D rectangular test case from [38] having four sealing faults and petrophysical properties sampled from SPE 10, Layer 10. The mode is scaled to $73 \times 134$ ft$^2$ to ensure significant non-Newtonian rheology throughout the whole reservoir. The reservoir is gridded by a PEBI-mesh with approximately 3800 cells that adapt to the faults, has radial refinement at the injector and...
We assume incompressible fluids with viscosities $\mu_0 = 10$ cP and $\mu_w = 1$ cP. The relative permeabilities and the capillary pressure are shown in Figure 15, while the polymer property curves are shown in Figure 16. In addition, we assume a Todd–Longstaff mixing parameter $\omega = 0.9$, residual resistance factor $R_{RF} = 1.05$, dead pore space $S_{ipv} = 0.02$, and rock density $\rho_r = 2000$ kg/m$^3$. The producer is set to be pressure-controlled at 200 bar, while the injector is rate-controlled with constant rate. A total of one pore volume of water is injected over a simulation time of 5 years. During the first 0.05 pore volumes, polymers are added to the injection water, and then the remaining simulation has pure water injection.

We run the fully-implicit solver in MRST, the sequential solver with exact pressures, and the sequential solver with approximate pressures computed by MsRSB. As in previous examples, we
Fig. 17: Fully-implicit solution at time of water breakthrough for the PEBI model; water saturation shown to the left and polymer concentration to the right.

Fig. 18: Water cut for MsRSB solver relative to the sequential solver with exact pressure. Results shown for different nonlinear incremental pressure tolerance $\varepsilon_p$ as defined in (12), and different multiscale tolerances.

do not observe notable differences between the three solutions. Figure 17 therefore only shows the fully-implicit solution at water breakthrough.

There are two parameters that control the accuracy of the multiscale solver: the multiscale tolerance that controls how accurately MsRSB approximates each linearized equation, and the nonlinear pressure tolerance that controls how accurately we approximate the nonlinear pressure equation. Figure 18 shows the difference in water cut between the sequential solver with exact pressure updates and a very low nonlinear pressure tolerance, and the MsRSB solution with different multiscale tolerances and different nonlinear pressure tolerances. As the nonlinear pressure tolerance is lowered, MsRSB approaches the solution computed with exact pressure updates. However, the different multiscale tolerances give approximately the same accuracy, meaning that even at a low multiscale tolerance, the flux field is well approximated.

An important question is to what extent the accuracy of the multiscale approximation affects the convergence of the nonlinear iterations. Figure 19 reports the average number of nonlinear iterations for a test matrix of four different multiscale tolerances ($10^{-n}, n = 1, \ldots, 4$) and five different nonlinear tolerances ($10^{-m}, m = 0, \ldots, 4$). Note that the nonlinear tolerance is an incremental pressure tolerance, as defined in (12), which means that the pressure has converged when the approximate multiscale increments are sufficiently small. Two trends can be observed: At least for this particular example, the pressure converges very quickly, and no more than two pressure iterations are required on average for a incremental pressure tolerances of $10^{-3}$ or smaller, even when using a relatively coarse multiscale tolerance of $10^{-1}$. Secondly, for small nonlinear tolerances, it seems like a multiscale tolerance of $10^{-3}$ to $10^{-2}$ is sufficient to ensure convergence in less than 2.5 iterations on average. This is a quite promising result: If only a few iterations are needed in the multiscale method, there is a significant potential for reducing computational cost compared to a fine-scale solver.

As explained above, we may extend the sequential solver by using outer iterations on each time step to make it converge to the solution of a fully-implicit formulation. That is: after the transport equation has converged, we compute the residual of the pressure equation using the updated saturation and concentration. If the pressure residual is larger than the prescribed tolerance, the pressure and transport equations are solved again at that same time step. This is repeated until convergence. This way, we expect MsRSB to converge to the fully-implicit solution as the tolerance level is decreased. Figure 20 reports the water cut computed by the fully-implicit solver and by the sequential MsRSB solver without outer iterations and multiscale tolerance set to 0.01. Even without outer iterations, the solutions match closely. The right plot in Figure 20 how we can systematically reduce the discrepancy between the fully-implicit solution and MsRSB by adjusting the tolerance on the outer iterations. Here, “Seq, Fine-Scale” refers to the sequential simulator with a direct solver for the linearized pressure equations and no outer iterations, whereas the other solutions are multiscale solutions with different tolerances for the outer iterations. To be able to compare the results directly,
Fig. 19: Average number of nonlinear pressure iterations throughout a simulation as a function of the nonlinear incremental pressure tolerance $\varepsilon_p$ as defined in (12). The different colors represent different multiscale tolerances.

Fig. 20: Left: water-cut curves computed by the fully-implicit solver and the sequential MRST solver with no outer iterations and multiscale tolerance 0.01. Right: relative difference between the fully-implicit solution and different sequential solutions; using a direct fine-scale solver for the linearized pressure systems with no outer iterations, and also using MsRSB as an approximate solver for the linearized pressure systems with different tolerances for the outer iterations.

all simulations are run with the same time steps. With a high tolerance on the outer iterations, MsRSB is in close agreement with the fine-scale sequential solution. With finer tolerances for the outer iterations, the multiscale solver converges to the fully-implicit solution as expected.

6 Conclusion

We have developed a sequential solution procedure for simulating polymer flooding with shear effects and extended the formulation with a multiscale solver to accelerate the computation of the pressure part of the flow problem. The multiscale solver has the appealing feature that it can provide mass-conservative flux fields without having to converge the linearized pressure residuals to machine precision. In the resulting simulator, we can therefore trade accuracy for computational efficiency by prescribing coarse tolerances on the linear and/or nonlinear pressure updates. On the other hand, the simulator can be made to reproduce the results of a fully-implicit simulator by prescribing low tolerances for the linear and nonlinear pressure updates and/or possibly adding outer iterations on each time step to run the simulator in an iterated fully-implicit mode.

Our prototype implementation has been verified against the commercial ECLIPSE 100 simulator to demonstrate correctness of the sequential formulation, as well as the robustness of the MsRSB method. Likewise, we have illustrated that the multiscale method can efficiently resolve the highly nonlinear flow physics of a shear-thinning polymer flooding in a reservoir model with industry-standard geological complexity. Seen from the point of view of a multiscale method, this demonstrates the versatility of the MsRSB method, which has also been extended and applied to two-phase thermal and compositional flow. More details will be reported in forthcoming publications.

Acknowledgments

The research is partly funded by VISTA, which is a basic research programme funded by Statoil and conducted in close collaboration with The Norwegian Academy of Science and Letters. The authors
would also like to acknowledge funding by Schlumberger Information Solutions and by the Research Council of Norway under grant no. 226035.

References


43. Schlumberger: ECLIPSE 2013.2 Technical Description (2013)