LINEAR SOLUTION TECHNIQUES FOR RESERVOIR SIMULATION WITH FULLY COUPLED GEOMECHANICS

A REPORT
SUBMITTED TO THE DEPARTMENT OF ENERGY
RESOURCES ENGINEERING
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

Sergey Klevtsov

March 2017

I certify that I have read this thesis and that, in my opinion, it is fully adequate in scope and quality as partial fulfillment of the degree of Master of Science in Energy Resources Engineering.

(Hamdi Tchelepi) Principal Adviser

Abstract

Direct modeling of geomechanics in reservoir simulation has gained a massive amount of interest in the last decade, due both to the increase in the available computing power and advances in numerical modeling, and to the importance of accurate representation of mechanical effects in many unconventional reservoirs. Both fully implicit and sequential implicit modeling approaches have been developed, with the latter being more popular due to easier coupling of existing simulation codes. Fully implicit simulation of coupled multiphase flow and poromechanics presents a challenge to the linear solver, that must be capable of handling and efficiently solving a large sparse linear system characterized by block structure resulting from discretizing and linearizing the system of governing equations of mass and momentum conservation.

In this work a linear solver framework subject to the above requirements is developed, building on and extending the existing techniques, and implemented in Automatic Differentiation General Purpose Research Simulator (AD–GPRS). First, existing data structures for sparse Jacobian storage, previously specialized to flow and transport problems with wells, are extended to handle an unlimited number of physical problems, in particular, geomechanics. Second, a generalized block–partitioned preconditioning operator is implemented, to provide the baseline (though possibly inefficient) technique for handling coupled systems of equations of any kind. Finally, an efficient preconditioning operator for multiphase flow and geomechanics is presented based on a combination of Constrained Pressure Residual (CPR) approach and the Fixed–stress preconditioning operator recently developed for single–phase flow. The proposed methods are tested on several models and a clear advantage of the Fixed–stress CPR algorithm with respect to iteration count and CPU time is shown.

Acknowledgments

First of all, I am indebted to my adviser, Prof. Hamdi Tchelepi, for the opportunity to become a part of Stanford and the reservoir simulation research group. His understanding, encouragement and guidance were invaluable during my years here, and I'm happy to be continuing as a PhD student with his ongoing support. It is a true honor and a privilege to be learning from Prof. Tchelepi.

I am thankful to all current and former ERE staff, faculty and researchers I've had a chance to work with. Special gratitude goes to Denis Voskov for enabling my journey to Stanford, as well as his mentorship and positive attitude. My work would not have been possible without the help and advice from Nicola Castelletto, Timur Garipov, Pavel Tomin and Abdulrahman Manea, to whom I am sincerely grateful. I would also like to acknowledge Joshua White of Lawrence Livermore National Laboratory for his collaboration.

I would like to thank all my friends, colleagues and office mates, who made this journey a lot more fun and supported me both in and outside of study, in particular Ruslan Rin, Karine Levonyan, Christine Maier, Youssef Elkady, Jose Eduardo Ramirez, Jamal Cherry, Vinay Tripathi and many others. I appreciate all the laughs and meals we've shared over the years.

Financial support from the Department of Energy Resources Engineering and the SUPRI-B research group and industrial affiliates consortium is kindly acknowledged. Additional thanks are due to to TOTAL S.A. for both funding my research through STEMS project and providing me with invaluable internship opportunities.

Finally, I am deeply grateful to my family and my wife Yulia for their unconditional love and support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Geomechanics in Reservoir Simulation

Conventional reservoir simulation deals with solving partial differential equations governing multiphase multicomponent flow in a porous medium over the domain of interest. As time goes on and simulation techniques get more mature, more and more complex models and interacting physical phenomena are being introduced in simulators, which turns it into a coupled *multi-physics* problem. Examples include extending simulator capabilities to deal with heat flow, chemical reactions and geomechanics. The latter is of particular interest, because accurate modeling of subsurface mechanical effects plays a critical role in predicting reservoir performance and behavior and assessing the safety of oil field and CO_2 sequestration operations. Many physical phenomena, such as compaction, subsidence, wellbore failure and change in matrix and fracture permeability can be modeled and explained with geomechanics.

Traditionally, reservoir simulators have simplified the mechanical rock response by reducing it to a single uniaxial compressibility coefficient (usually assumed constant). However, in particular due to the increased interest in unconventional resources, such as shales, multiple efforts have been made to integrate proper geomechanical modeling in reservoir simulators, efforts required because of the tight coupling and complex dynamics taking place between the flow and mechanical problems. A number of solution strategies have been proposed, including [7]:

- *Loosely coupled*, where the coupling between the two problems is resolved only at certain time intervals. This approach is the least computationally expensive as it involves far fewer mechanics solves, but its accuracy cannot be reliably estimated a priori.

- *Iteratively coupled*, where one of the two problems (either flow and transport or mechanics) is solved first to obtain an intermediate solution estimate, and then the other problem is solved using this solution; the process is iterated until the desired convergence tolerance is achieved. Like the loosely coupled approach, this approach allows to use separate simulation codes for flow and mechanics, but this scheme is accurate, resulting in the same solution as the fully coupled one. Based on the manner in which the variables of each problem are fixed, a number of schemes have been devised, some of which are unconditionally stable. Still, a number of sequential iterations (between flow and mechanical parts) may be required.

- *Fully coupled*, where the equations governing flow, transport and geomechanics are discretized and solved simultaneously at every time step. This scheme is always accurate and unconditionally stable. However, it requires investing additional effort in developing a unified simulator capable of handling multiple physical problems, both on the nonlinear (discretization schemes, nonlinear loop and Newton-Raphson linearization) and linear (solving a large fully coupled linear system) levels.

For the reasons mentioned above, most activity in this area has been directed towards the iterative coupling schemes ([11]), and relatively few efforts have been invested in fully coupled simulators. Difficulties associated with generating fully coupled Jacobian matrices can be overcome with automatic differentiation (AD) techniques. Recently, an AD library, ADETL [26], has been developed by R. Younis targeting reservoir simulation applications. It became the basis for AD-GPRS [23], Stanford's in-house simulator, with capabilities for solving flow, transport, thermal and geomechanical problems using different coupling strategies (both sequential and fully implicit schemes).

Fully implicit simulation poses additional challenges for the linear solver, which must be capable to efficiently handle large linear systems with multiple sets of unknowns and a distinct block structure in the matrix arising from discretizing multiple types of governing equations. Therefore moving to fully coupled multiphysics simulation encourages development of specialized linear solution techniques.

1.2 Linear Solvers

The linear solver is one of the most crucial and performance-critical components of a reservoir simulator. In a typical simulation it can take anywhere between 40% and 80% of the total runtime. Due to the size of problems to be solved, use of direct solvers is computationally infeasible, and iterative solvers are preferred. Typically used algorithms include Krylov subspace methods, such as *Orthomin*, *GMRES* and *BiCGStab* [18], which are known to be the most efficient. However, they rely on *preconditioning* of the linear system to accelerate convergence, that is, improving the condition number of the matrix by pre– or post–multiplying the matrix by a preconditioning operator $\mathcal{P}$:

$$\mathcal{P}^{-1}Ax = \mathcal{P}^{-1}b \qquad\qquad A\mathcal{P}^{-1}\mathcal{P}x = b$$

Note that in the context of iterative linear solvers, the only operation required is application of $\mathcal{P}^{-1}$ to vectors. A preconditioner can therefore be viewed as a black box operator, acting on a vector v and producing another vector $w = \mathcal{P}^{-1}v$, which is the approximate solution of $Aw = v$, since the operator is chosen such that $\mathcal{P}^{-1}$ resembles A^{-1} in some sense, while being much cheaper to construct than the actual inverse. Any solver, including a Jacobi-type smoother, a nested Krylov iteration, a multigrid method, an incomplete factorization, or a block-partitioned scheme, can be used as a preconditioner.

1.3 Previous work

The linear solver framework in AD-GPRS is largely based on the work of Y. Jiang [10], who developed both the data structures and the solver code, which has been later extended by Y. Zhou [28]. Here the main features of the linear solver framework are described.

1.3.1 Data Structures

Efficient sparse data structures are required to store and manipulate the linear system during the solution. The *Multi Level Sparse Block* (MLSB) data structure is a hierarchical matrix storage format that highlights the separation of the original system into individual blocks (or subproblems) and facilitates the development of specialized preconditioning strategy. On the top level, and MLSB matrix is just a storage container for nested reservoir (J_{RR}) and facilities (J_{FF} matrices, together with coupling blocks (J_{RF} and J_{FR}). In this notation the first subscript index is used to denote the type of equation (reservoir of facilities) and the second index is the type of variables the derivative is taken with respect to. Individual submatrices are allowed to have any internal representation. For instance, the reservoir matrix J_{RR}

is stored in a customized block CSR (Compressed Sparse Row) format that emphasizes the block ordering of independent simulation variables and is particularly well suited for Adaptive Implicit Method (AIM) Jacobians. The facilities matrix J_{FF} in turn is another storage container for individual facility submatrices, whose storage format depends on the facility models used and can include more nested submatrix levels. The reservoir-facility coupling blocks J_{RF} and J_{FR} are also containers for individual facility coupling blocks, which are typically stored in coordinate (COO) format. The key feature of this approach is that all matrices and matrix blocks share a common interface, that includes operations like extraction from AD residuals, various forms of algebraic reduction (for example, from full system to primary and from primary to implicit system), matrix-vector operations required by iterative solvers and output in human– and machine–readable formats. These operations are defined recursively, with upper–level matrices calling submatrices to execute operations. At the same time, implementation details are completely hidden from the upper level matrices. Thus the MLSB format was designed to provide convenient abstraction and organization of systems with several coupled physical models. Existing implementation, however, has been limited to reservoir-well systems only, despite initially flexible high-level design. The fact that each physical model is stored in a separate container allows for easy extraction of individual blocks and implementing efficient specialized linear and nonlinear solution strategies. The high-level representation of the format is shown in Figure 1.1.

1.3.2 Algorithms

The second important part of the framework is a collection of solution algorithms. AD-GPRS linear solver framework provides interfaces to a number of external solver libraries (such as SuperLU [8, 15], PARDISO [20] and SAMG [21]), as well as it's own set of iterative solvers. The latter is of primary interest, since it allows to perform

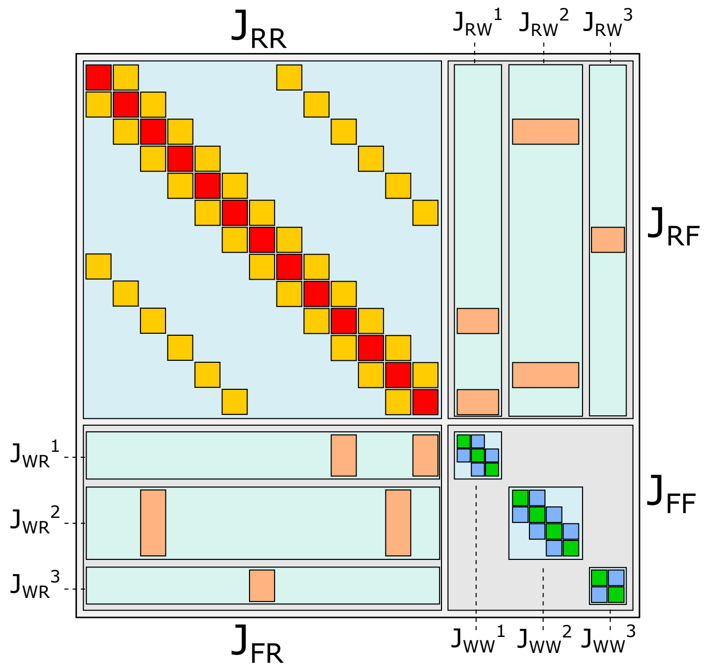


Figure 1.1: Multi Level Sparse Block matrix format [10] schematics.

research on efficient specialized solution strategies; however, external libraries are often used as building blocks for more complex solution algorithms. Despite the fact that matrix size is reduced after full to primary and primary to implicit linear system reductions, it still remains significant (often on the order of 10^6 and larger), especially in fully implicit simulations.

Krylov subspace iterative solvers are perhaps the most robust and widely used family of solvers. A nice common feature of them is independence of the details of matrix structure, that is, only the typical BLAS operations (matrix–vector product, axpy, dot products) need to be defined. This makes Krylov solvers perfectly suited to working with MLSB hierarchical matrices. In particular, GMRES [19] is known for its superior performance when used in conjunction with a good preconditioner. It guarantees monotone convergence of residual norm $||\mathbf{b} - \mathbf{Ax}||_2$ to zero within N (size of the system) iterations, however in practice the actual number of iterations required to converge to the desired tolerance is much smaller than N. In addition, the algorithm is typically restarted every m iterations (with m anywhere between 20 and 100) in order to keep memory requirements and numerical round-off error under control.

CPR Preconditioning

Convergence properties of Krylov subspace methods depend strongly on the availability of a high quality preconditioner. It is well-known that some generic families of preconditioners, such as, for instance, the Incomplete LU factorizations (ILU) [18], can be easily applied to any type of matrix, but their effectiveness varies from problem to problem and in general is suboptimal. On the other hand, specialized preconditioning strategies for reservoir simulation equations have been developed, among them the Constrained Pressure Residual [24] is perhaps the most efficient and well known. Here we briefly describe the idea and existing implementation of CPR.

Reservoir simulation equations are those of mass conservation (secondary thermodynamic and linear constraint equations are local and are taken care of during the first step of the algebraic reduction process mentioned previously). They implicitly include a near–elliptic diffusion part (through the pressure primary unknown) and near–hyperbolic advection part (through saturation, phase composition or other choices of primary unknowns), and as such, exhibit a mixed nature. The CPR preconditioning is a two-stage strategy based on the idea of isolating the near-elliptic part by constructing a pressure equation algebraically on the linear level and solving for an approximate pressure solution using an efficient elliptic solver (such as a few steps of a multigrid or multiscale method), which resolves global long-range error modes in the residual. The remaining degrees of freedom are treated in the second stage by a local smoother, such as ILU(k) (or its block counterpart, BILU(k)), which removes the high frequency local errors. The overall preconditioning scheme $\mathcal{P}_{cpr}$ can be represented as follows:

$$\mathcal{P}_{cpr}^{-1} = \mathcal{P}_{sm}^{-1}(I - J\mathcal{P}_p^{-1}) + \mathcal{P}_p^{-1} \tag{1.1}$$

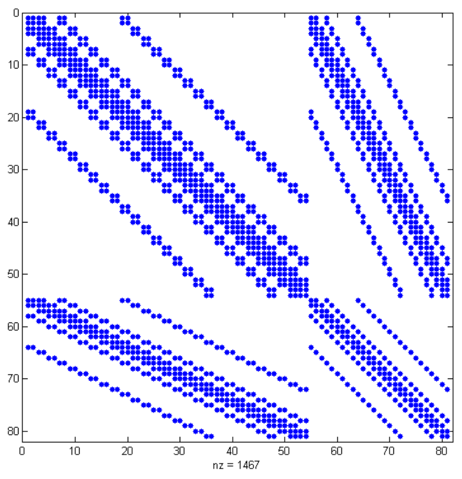
where $\mathcal{P}_p$ is the first–stage pressure preconditioner, $\mathcal{P}_{sm}$ is the second–stage smoother, J is the Jacobian matrix and I is an identity operator.

A detailed description of CPR implementation can be found in [28]. Here a high-level view of the preconditioning operator is presented. We begin with a fully implicit linear system that has a distinct block structure:

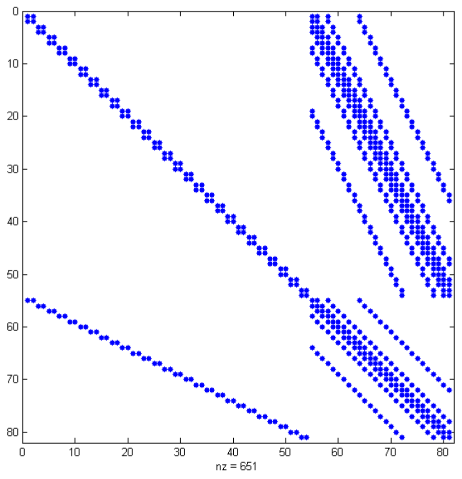
$$J\boldsymbol{\delta}\mathbf{x} = \mathbf{r} \qquad \text{or} \qquad \begin{bmatrix} J_{ss} & J_{sp} \\ J_{ps} & J_{pp} \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta}\mathbf{x}_s \\ \boldsymbol{\delta}\mathbf{x}_p \end{bmatrix} = \begin{bmatrix} \mathbf{r}_s \\ \mathbf{r}_p \end{bmatrix} \tag{1.2}$$

Here we use subscript p to denote pressure and s for all other implicit variables. Since there is no dedicated pressure equation in a FIM system, we choose the first equation in each block as the pressure equation (corresponding to the pressure unknown, which is also first in each block). Note that in the actual implementation the degrees of freedom are not split into global blocks, but rather are interleaved and arranged on a block–by–block basis. Here, for the simplicity of algebraic formulation of the method, we rearrange them equivalently to separate the pressure block. An example structure of such Jacobian on a 3D structured grid is shown in Figure 1.2a. All four blocks have the same block nonzero sparsity structure defined by the grid connectivity, with only the size of the blocks being different.

Construction of the pressure system is carried out through a process called *True-IMPES* reduction, which begins by applying simplifying assumptions to the Jacobian, namely, explicit treatment of all non–pressure unknowns in flux terms. This is achieved purely on the linear level by applying a column summation operator $\mathcal{C}$, which adds off-diagonal elements in each block column into the corresponding diagonal block



(a) Full reservoir Jacobian structure.



(b) Reservoir Jacobian structure after colsum.

Figure 1.2: Illustration of CPR reduction process.

element and then discards the off–diagonal entries, to the J_{ss} and J_{ps} blocks. Since every flux term adds contributions to the diagonal and off-diagonal of equal value but opposite sign, this operation effectively cancels out the derivatives of fluxes with respect to non–pressure variables, which roughly corresponds to IMPES nonlinear treatment, with the difference being that values are taken from previous nonlinear iteration rather than previous time level. After that, the approximated $\widetilde{J}_{ss} = \mathcal{C}(J_{ss})$ is diagonal and therefore easily invertible, and a sparse Schur complement can be constructed to obtain a decoupled pressure system:

$$\widetilde{J}_{pp} = J_{pp} - \widetilde{J}_{ps}\widetilde{J}_{ss}^{-1}J_{sp} = R^{cpr}\widetilde{J}P^{cpr} \qquad \widetilde{\mathbf{r}}_p = \mathbf{r}_p - \widetilde{J}_{ps}\widetilde{J}_{ss}^{-1}\mathbf{r}_s = R^{cpr}\mathbf{r} \tag{1.3}$$

where

$$R^{cpr} = \begin{bmatrix} -J_{ps}\widetilde{J}_{ss}^{-1} & I \end{bmatrix} \qquad P^{cpr} = \begin{bmatrix} 0 \\ I \end{bmatrix} \tag{1.4}$$

The resulting pressure matrix structure is that of the J_{pp} block; the Schur complement changes its entries but does not affect sparsity pattern. After the approximate pressure solution is obtained, it is prolongated to the full solution vector and the right–hand side is updated accordingly:

$$\mathbf{r}_2 = \mathbf{r} - J\boldsymbol{\delta}\mathbf{x}_1 = \mathbf{r} - J\mathcal{P}_p^{-1}\mathbf{r} = [I - J\mathcal{P}_p^{-1}]\mathbf{r} \tag{1.5}$$

Finally, applying a second-stage smoother to the full system with an adjusted residual yields an update to both pressure and non–pressure unknowns, which is combined with the first stage pressure solution for the final result:

$$\begin{aligned}\boldsymbol{\delta}\mathbf{x} = \boldsymbol{\delta}\mathbf{x}_1 + \boldsymbol{\delta}\mathbf{x}_2 = \mathcal{P}_p^{-1}\mathbf{r} + \mathcal{P}_{sm}^{-1}\mathbf{r}_2 = \mathcal{P}_p^{-1}\mathbf{r} + \mathcal{P}_{sm}^{-1}[I - J\mathcal{P}_p^{-1}]\mathbf{r} = \\ = [\mathcal{P}_p^{-1} + \mathcal{P}_{sm}^{-1}[I - J\mathcal{P}_p^{-1}]]\mathbf{r} = \mathcal{P}_{cpr}\mathbf{r}\end{aligned} \tag{1.6}$$

Wells are naturally incorporated within the CPR strategy. Standard well models have only a single unknown, the bottom hole pressure p^w, which is simply kept in the first-stage pressure system and solved for together with reservoir block pressures. Multisegment wells are first reduced to a single pressure unknown and then treated similarly [27], [28]. Note that construction of the restriction and prolongation operators, as well as setup of the pressure solver and smoother (all expensive operations) in CPR need to be performed only once per nonlinear iteration (when the matrix changes), while application these operators (relatively cheap) occurs on every linear iteration (since they need to be applied to each newly constructed vector of the Krylov subspace). Overall, CPR with highly optimized kernels has been shown to be a very robust and powerful preconditioner for reservoir simulation with standard and multisegment wells ([2], [10], [28], [3]).

Concluding Remarks

CPR is an example of a multi-stage preconditioning scheme. In general form, such schemes can be expressed as follows [3]:

$$\mathcal{P}^{-1} = \sum_{i=1}^{n} \mathcal{P}_i^{-1} \prod_{j=1}^{i-1} \left[I - J\mathcal{P}_j^{-1} \right] \tag{1.7}$$

where $\mathcal{P}_i$ is the preconditioner used at i–th stage, and for $i = 1$ the product term is defined to be equal to identity. This notation describes the consecutive application of stages, with and adjustment of the residual between stages. Note that each stage $\mathcal{P}_i$ may have a reduction and prolongation operators implicitly built-in.

Multistage/block–partitioned preconditioning is a powerful idea that can be exploited to implement strategies for any number of coupled problems, including but not limited to flow, mass transport, geomechanics and heat flow and transport. In the next chapter we develop the idea of a multistage preconditioner for coupled multiphase flow and transport with geomechanics.

Chapter 2

Linear Solver Developments

Advancements in geomechanical modeling and ongoing incorporation of mechanical models in reservoir simulators necessitates the development of efficient linear solution and preconditioning techniques for systems arising from fully implicit discretization. The size of these systems is much larger than those in conventional reservoir simulation. For example, on a structured 3D grid with a large number of grid blocks the total number of nodes in the limit is approximately equal to the number of blocks. With 3 additional mechanical unknowns per node (components of displacement vector $\mathbf{u}$, the size of the systems is twice as large as a regular simulation with 3 unknowns per cell (e.g. a Black–Oil model). In addition, the mechanical domain is typically taken much larger than that of fluid flow, in order to specify boundary conditions properly for the domain of interest, which could result in an even larger system. All of this indicates an efficient and scalable preconditioner is required to handle such systems.

In this chapter first the mathematical and numerical framework used for modeling is described to provide a foundation for discussion of linear systems arising during simulation. Then the necessary extensions to the linear solver framework of AD-GPRS are discussed, including both data structures and algorithms, and a new efficient preconditioning scheme is detailed.

2.1 Mathematical Framework for Coupled Flow and Mechanics

The principal equations governing multiphase multicomponent flow and mechanics in porous medium are those of mass and momentum conservation. The multiphase poromechanics problem is formulated after [6] in terms of displacement and fluid pressure unknowns and natural variables [2] (saturation/phase composition) for transport. We are considering an arbitrary porous domain $\Omega \in \mathbb{R}^3$ with boundary Γ with a normal vector $\mathbf{n}$ that is partitioned into non–overlapping regions as $\Gamma = \overline{\Gamma_{\mathbf{u}} \cup \Gamma_{\boldsymbol{\sigma}}} = \overline{\Gamma_p \cup \Gamma_{\mathbf{q}}}$ for the purpose of specifying boundary conditions.

Assuming isothermal conditions and infinitesimal displacements, and neglecting capillary pressure effects (thus, $p_p = p$ for each phase p) the strong form of the initial/boundary value problem can be stated as: find $\mathbf{u}$, p, S_p and x_{cp}, $p = 1..N_p, c = 1..N_c$, over the domain Ω and time interval $\mathcal{T} = [0, T]$ such that

$$\nabla \cdot (\boldsymbol{\sigma}' - bp\mathbf{1}) + \mathbf{f} = \mathbf{0} \qquad \text{on } \Omega \times \mathcal{T} \qquad \text{(linear momentum balance)}, \tag{2.1}$$

$$\dot{m}_c + \nabla \cdot \mathbf{w}_c - f_c = 0 \qquad \text{on } \Omega \times \mathcal{T} \qquad \text{(mass balance for component } c\text{)}, \tag{2.2}$$

subject to the following boundary conditions

$$\mathbf{u} = \bar{\mathbf{u}} \qquad \text{on } \Gamma_{\mathbf{u}} \times \mathcal{T} \qquad \text{(boundary displacements)}, \tag{2.3}$$

$$(\boldsymbol{\sigma}' - bp\mathbf{1}) \cdot \mathbf{n} = \bar{\mathbf{t}} \qquad \text{on } \Gamma_{\boldsymbol{\sigma}} \times \mathcal{T} \qquad \text{(boundary tractions)}, \tag{2.4}$$

$$p = \bar{p} \qquad \text{on } \Gamma_p \times \mathcal{T} \qquad \text{(boundary pore pressure)}, \tag{2.5}$$

$$-\mathbf{w}_c \cdot \mathbf{n} = \bar{q}_c \qquad \text{on } \Gamma_{\mathbf{q}} \times \mathcal{T} \qquad \text{(boundary mass flux for component } c\text{)}, \tag{2.6}$$

and initial conditions for $\xi \in \Omega$

$$\begin{aligned}
\mathbf{u}(\boldsymbol{\xi},0) &= \mathbf{u}^0(\boldsymbol{\xi}) && \text{(initial displacements)}, && (2.7)\\
p(\boldsymbol{\xi},0) &= p^0(\boldsymbol{\xi}) && \text{(initial pore pressure)}, && (2.8)\\
S_p(\boldsymbol{\xi},0) &= S_p^0(\boldsymbol{\xi}) && \text{(initial saturation for fluid phase } p\text{)}, && (2.9)\\
x_{cp}(\boldsymbol{\xi},0) &= x_{cp}^0(\boldsymbol{\xi}) && \text{(initial mole fraction of component } c \text{ in phase } p\text{)}. && (2.10)
\end{aligned}$$

Here, $\boldsymbol{\sigma}'$ is effective stress tensor; b is the Biot coefficient; $\mathbf{1}$ is the second-order unit tensor; $\mathbf{f}$ is the body force vector, which is assumed constant here; $m_c = \left(\sum_{p=1}^{N_p} \phi \rho_p S_p x_{cp} \right)$, $\mathbf{w}_c$ and f_c denote mass, mass flux and source/sink term for component $c = 1..N_c$, with ϕ and ρ_p porosity and fluid phase density, respectively; $\boldsymbol{\xi}$ is the position vector in $\mathbb{R}^3$; $\nabla\cdot$ and ∇ are the divergence and gradient operator, respectively. The superposed dot, $\dot{()}$, denotes a derivative with respect to time.

The formulation is completed by the addition of appropriate constitutive relationships. In particular, a fourth-order tensor of drained tangential moduli, C_{dr}, relates $\boldsymbol{\sigma}'$ to $\mathbf{u}$ via the the linearized second-order strain tensor, $\boldsymbol{\epsilon}$. In general incremental form such relationships reads:

$$\Delta\boldsymbol{\sigma}' = \mathsf{C}_{\mathrm{dr}} : \Delta\boldsymbol{\epsilon}, \tag{2.11}$$

$$\boldsymbol{\epsilon} = \nabla^s \mathbf{u} = \frac{1}{2}\left(\nabla\mathbf{u} + \nabla^T\mathbf{u}\right), \tag{2.12}$$

where ∇^s is the symmetric gradient operator, and (:) denotes a tensor product contracted on two indices. Following [6], porosity is modeled as:

$$\Delta\phi = b\Delta\epsilon_v + \frac{(b-\phi_0)}{K_s}\Delta p, \tag{2.13}$$

where $\epsilon_v = \mathrm{trace}(\boldsymbol{\epsilon})$ is the volumetric strain, and $K_s = K/(1-b)$ is the solid phase bulk modulus, with K the skeleton bulk modulus. Note that Biot's coefficient is restricted to the interval $\phi < b \leq 1$, with $b = 1$ representative of the incompressible solid phase limit case [9]. Mass flux for each component is given by $\mathbf{w}_c = \sum_{p=1}^{N_p} \rho_p \mathbf{v}_p x_{cp}$, which makes use of the classic multiphase flow generalization of Darcy's law [1]:

$$\mathbf{v}_p = -\frac{k_{rp}}{\mu_p} \boldsymbol{k} \cdot \left(\nabla p - \rho_p \mathbf{g} \right), \tag{2.14}$$

where $\boldsymbol{k}$ is the absolute permeability tensor; μ_p and k_{rp} are the viscosity and relative permeability factor for fluid phase p, respectively; and $\mathbf{g}$ is the gravity vector.

The source/sink terms $f_c = \sum_{p=1}^{N_p} q_p \rho_p x_{cp}$ in Equation (2.2) represent the effects of well production/injection and are typically line sources. These are governed by a suitable well model with certain assumptions—typically radial flow in the vicinity of the wellbore, that relates well control parameters, such as bottom hole pressure, to phase volumetric flow rates q_p through the wellbore.

The system of equations is closed by adding local thermodynamic equilibrium constraints in the form of fugacity equality relations:

$$\mathrm{f}_{cp_1}(P, \boldsymbol{x}_{p_1}) = \mathrm{f}_{cp_2}(P, \boldsymbol{x}_{p_2}), \quad \forall p_1, p_2 = 1..N_p, \quad \forall c = 1..N_c, \tag{2.15}$$

and linear saturation/composition constraints

$$\sum_{p=1}^{N_p} S_p = 1, \tag{2.16}$$

$$\sum_{c=1}^{N_c} x_{cp} = 1, \quad \forall p = 1..N_p. \tag{2.17}$$

The initial/boundary value problem (2.1)–(2.10) is discretized in space by a mixed finite–element (FE)/finite–volume (FV) approach. A single conforming computational grid that partitions Ω in n_{elem} non–overlapping elements Ω_e (control volumes in the finite-volume terminology) is used. The balance of linear momentum is solved based on the following variational formulation:

$$\mathcal{R}^{\text{mom.}} = -\int_{\Omega} \nabla^s \boldsymbol{\eta} : \boldsymbol{\sigma}' \, \mathrm{d}V + \int_{\Omega} bp \nabla \cdot \boldsymbol{\eta} \, \mathrm{d}V + \int_{\Omega} \boldsymbol{\eta} \cdot \mathbf{f} \, \mathrm{d}V + \int_{\Gamma_{\boldsymbol{\sigma}}} \boldsymbol{\eta} \cdot \mathbf{t}_{\sigma} \, \mathrm{d}A = 0 \quad \forall \boldsymbol{\eta}, \tag{2.18}$$

with $\boldsymbol{\eta}$ appropriate weighting functions. In particular, $\boldsymbol{\eta}$ varies over the spaces $(H^1(\Omega))^3$, and satisfies homogeneous boundary conditions on the essential boundaries $\Gamma_{\mathbf{u}}$. Here, H^1 is a Sobolev space of degree one. The derivation of the discrete form of the mass balance equations consists of writing Equation (2.2) for a generic Ω_e in integral form. Making use of divergence theorem leads to the following set of balance equations for each fluid component $c = 1..N_c$:

$$\mathcal{R}^{\text{mass}}_{c,e} = \int_{\Omega_e} \dot{m}_c \, \mathrm{d}V + \sum_{j \in \mathbb{J}_e} w_{c,ej} - \int_{\Omega_e} f_c \, \mathrm{d}V = 0 \qquad e = 1, \ldots, n_{\text{elem}} \tag{2.19}$$

where $\mathbb{J}_e$ denotes the set of indices j of elements Ω_j sharing a face with Ω_e, and $w_{c,ej}$ are the inter-element integral fluxes

$$w_{c,ej} = \int_{\Gamma_{ej}} \mathbf{w}_c \cdot \mathbf{n}_{ej} \, \mathrm{d}A, \tag{2.20}$$

with $\mathbf{n}_{ej}$ the unit normal vector to Γ_{ej} oriented from Ω_e to Ω_j.

The final discrete form is obtained based on a suitable time and space discretization of the set of residual equations (2.18)–(2.19). Time integration relies on the first–order accurate, unconditionally stable backward Euler scheme. The displacement field, $\mathbf{u}$, is approximated with trilinear ($\mathbb{Q}_1$) finite elements while a piecewise constant interpolation ($\mathbb{P}_0$) is used for both the pressure field p, and the fluid phase saturation, S_p, and composition, x_{cp}, fields. As far as the approximation of fluxes (2.20) is concerned, either a two–point flux (TPFA) or a multi–point flux (MPFA) approximation can be used, depending on the grid. Let $\mathbf{x}_n = [\mathbf{S}^n, \boldsymbol{x}^n, \mathbf{u}^n, \mathbf{p}^n]^T$ be the discrete solution at time t_n. The solution $\mathbf{x}_{n+1}$ at time $t_{n+1} = (t_n + \Delta t)$ is obtained by solving the following discrete system of equations

$$\mathbf{r}\left(\mathbf{x}^{n+1}, \mathbf{x}^n\right) = \begin{bmatrix} \mathbf{r}^{\text{mass}} \\ \mathbf{r}^{\text{mom.}} \end{bmatrix} = \mathbf{0}, \tag{2.21}$$

with the residual vector $\mathbf{r}$ being assembled as sum of element contributions. Specifically, the element–wise contributions to the discrete residual vector for the generic Ω_e read:

$$\begin{aligned}[\mathbf{r}^{\text{mom.}}]_i = &- \int_{\Omega_e} \nabla^s \boldsymbol{\eta}_i : \boldsymbol{\sigma}'^{,n+1} \,\mathrm{d}V + \int_{\Omega_e} b p^{n+1} \nabla \cdot \boldsymbol{\eta}_i \,\mathrm{d}V + \int_{\Omega_e} \boldsymbol{\eta}_i \cdot \mathbf{f}^{n+1} \,\mathrm{d}V \\ &+ \int_{\Gamma_{\boldsymbol{\sigma},e}} \boldsymbol{\eta}_i \cdot \bar{\mathbf{t}}^{n+1} \,\mathrm{d}V, \qquad i = 1, 2, \dots, n_{\mathbf{u}} \end{aligned} \tag{2.22}$$

$$[\mathbf{r}_c^{\text{mass}}]_e = \frac{1}{\Delta t} \int_{\Omega_e} \left(m_c^{n+1} - m_c^n\right) \,\mathrm{d}V - \sum_{j \in \mathbb{J}_e} T_{ej} \lambda_{p,ej}^{n+1} \left(\Phi_{p,j}^{n+1} - \Phi_{p,e}^{n+1}\right) - \int_{\Omega_e} f_c^{n+1} \,\mathrm{d}V \tag{2.23}$$

where vector functions $\boldsymbol{\eta}_i$ are the finite element bases for the displacement, with $n_{\mathbf{u}}$ the number of displacement degrees of freedom; T_{ej} is the geometric transmissibility of interface Γ_{ej}; $\Phi_{p,k} = (p_{p,k} - \rho_{p,k} \left|\mathbf{g}\right| d_k)$ is the p-th fluid phase flow potential evaluated at the k–th cell centroid depth, d_k; and $\lambda_{p,ej}$ is the p–th fluid phase mobility computed from upstream cells, namely

$$\lambda_{p,ej} = \begin{cases} \rho_{p,e}\dfrac{k_{rp,e}}{\mu_{p,e}}, & \text{if } \Phi_{p,e} > \Phi_{p,j} \\ \rho_{p,j}\dfrac{k_{rp,j}}{\mu_{p,j}}, & \text{otherwise} \end{cases} \tag{2.24}$$

Source/sink terms related to wells are discretized using a Peaceman type well model [17]:

$$\int_{\Omega_e} f_c^{n+1}\,\mathrm{d}V = T_e^w \lambda_{p,e}^{w,n+1}(\Phi_{p,e}^{n+1} - \Phi_{p,e}^{w,n+1}) \tag{2.25}$$

where T_e^w is the well index, and $\Phi_{p,e}^w$ is the wellbore phase potential at perforated cell e. If bottom hole pressure is specified, it is used directly to evaluate $\Phi_{p,e}^w$, otherwise it becomes an additional unknown and a separate well control equation is formulated to close the system (see [10] for more details). Here we restrict ourselves to the former case. Boundary conditions for flow and transport in principle are treated similarly: if a component flux is specified, it is added directly to the corresponding residual equation of the boundary cell; if pressure and saturations/compositions are specified, additional flux terms with a half–transmissibility appear in the boundary cell. In the current version, however, AD–GPRS lacks proper support for arbitrary flow boundary conditions (zero flux is always assumed), but they can be emulated in practice through an additional layer of boundary cells.

The residual vector (2.21) is nonlinear due to the following dependences: $\boldsymbol{\sigma}'(\mathbf{u})$, $m_c = m_c(\mathbf{S}, \boldsymbol{x}_c, \phi(\mathbf{u}), p)$, $\lambda_{p,ej} = \lambda_{p,ej}(S_p, p)$, $p = 1..N_p$. Here, we will restrict ourselves to isotropic linear elastic behavior for the solid skeleton, therefore $\mathbf{C}_{\mathrm{dr}}$ is constant and depends on two coefficients only, e.g. Young's modulus, E, and Poisson's ratio, ν.

Newton's method is used to drive $\mathbf{r}$ to zero:

$$J\left(\mathbf{x}^{n+1,k}\right)\delta\mathbf{x} = -\mathbf{r}\left(\mathbf{x}^{n+1,k},\mathbf{x}^{n}\right), \tag{2.26}$$

$$\mathbf{x}^{n+1,k+1} = \mathbf{x}^{n+1,k} + \delta\mathbf{x}, \tag{2.27}$$

with J the Jacobian matrix evaluated at Newton step k. In the rest of this work, for the sake of convenience, we label both saturation and composition unknowns with a lowercase s; additionally, even though all component mass balance equations are similar, we arbitrarily choose the first one in each cell and label it as 'pressure' (p) equation, to correspond to the pressure unknown, which is always first in each cell. Finally, although equations are written cell–by–cell (or node–by–node), we can rearrange all equations and unknowns based on the unknown type (saturation/composition, displacement or pressure), for convenience of algebraic notation. With this, the Jacobian matrix of the system can be written as:

$$J\left(\mathbf{x}^{n+1,k}\right) = \begin{bmatrix} \dfrac{\partial \mathbf{r}^{\text{mass}}_{2..N_c}}{\partial \mathbf{s}}\left(\mathbf{x}^{n+1,k}\right) & \dfrac{\partial \mathbf{r}^{\text{mass}}_{2..N_c}}{\partial \mathbf{u}}\left(\mathbf{x}^{n+1,k}\right) & \dfrac{\partial \mathbf{r}^{\text{mass}}_{2..N_c}}{\partial \mathbf{p}}\left(\mathbf{x}^{n+1,k}\right) \\ \dfrac{\partial \mathbf{r}^{\text{mom.}}}{\partial \mathbf{s}}\left(\mathbf{x}^{n+1,k}\right) & \dfrac{\partial \mathbf{r}^{\text{mom.}}}{\partial \mathbf{u}}\left(\mathbf{x}^{n+1,k}\right) & \dfrac{\partial \mathbf{r}^{\text{mom.}}}{\partial \mathbf{p}}\left(\mathbf{x}^{n+1,k}\right) \\ \dfrac{\partial \mathbf{r}^{\text{mass}}_{1}}{\partial \mathbf{S}}\left(\mathbf{x}^{n+1,k}\right) & \dfrac{\partial \mathbf{r}^{\text{mass}}_{1}}{\partial \mathbf{u}}\left(\mathbf{x}^{n+1,k}\right) & \dfrac{\partial \mathbf{r}^{\text{mass}}_{1}}{\partial \mathbf{p}}\left(\mathbf{x}^{n+1,k}\right) \end{bmatrix} = \begin{bmatrix} J_{ss} & J_{su} & J_{sp} \\ J_{us} & J_{uu} & J_{up} \\ J_{ps} & J_{pu} & J_{pp} \end{bmatrix} \tag{2.28}$$

An example structure of such Jacobian matrix is shown in Figure 2.1, except that flow/transport unknowns are not rearranged into separate blocks. Note that all local equations, including fugacity and linear constraints (2.15)–(2.17), are reduced during linearization process (either at AD level or algebraically) and do not appear in the final Jacobian. Additionally, as noted before, for simplicity we ignore well equations and unknowns in this discussions, restricting ourselves to bottom hole pressure controls, but they are naturally handled both by data structures and algorithms ([10, 28]).

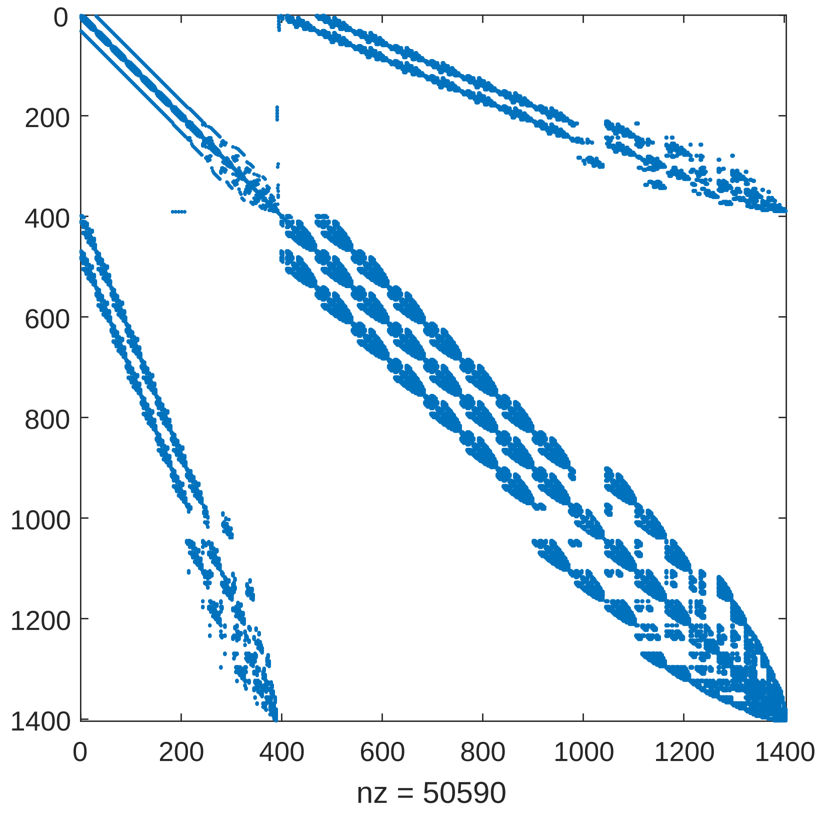


Figure 2.1: Example of Jacobian structure with reservoir, wells and geomechanics.

2.2 Data Structures Support for Mechanics

In order to store and manipulate systems with flow and geomechanical parts, the data structures used on the linear level must be made mechanics–aware. Luckily, the design of MLSB matrix format used in AD-GPRS is flexible enough to allow for such an extension. Previously, the system matrix has been implemented as a container with a fixed number of inner blocks (reservoir, facilities and couplings), templated on the type of those blocks so that implementation could be switched easily. Foreseeing future extensions to heat flow and other problems, instead of adding another inner

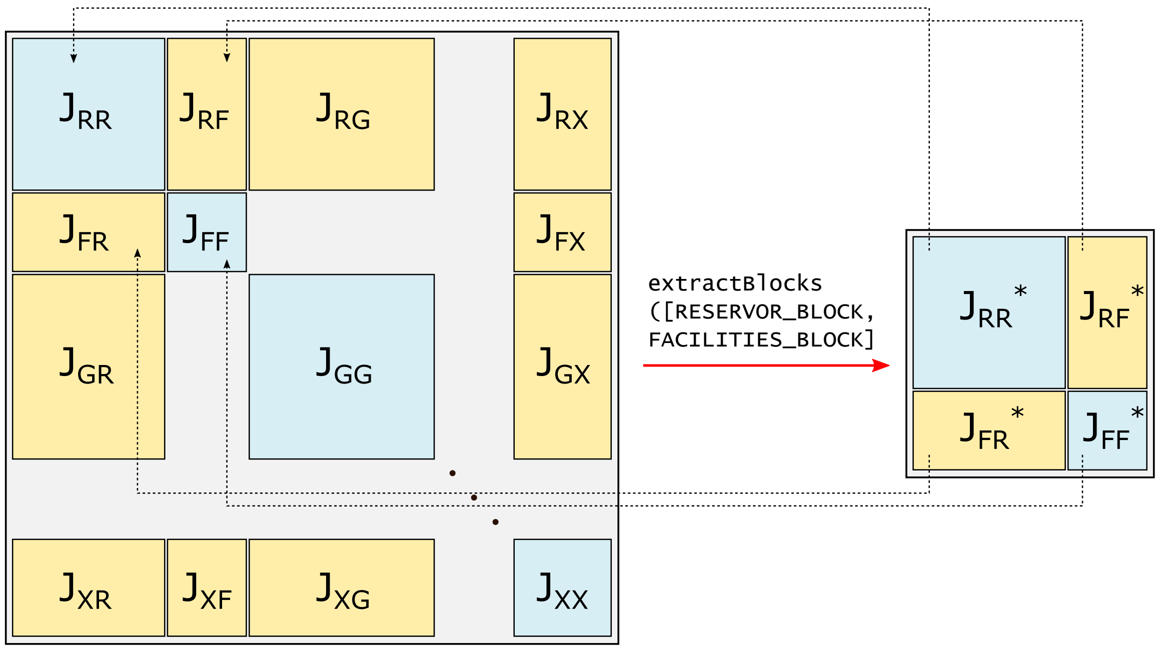


Figure 2.2: Extended Multi Level Sparse Block matrix format grid layout and submatrix extraction. **X** represents an arbitrary coupled subproblem.

block (together with relevant coupling blocks) into the fixed-size container, we reimplemented it as a dynamic container called *BlockMatrixWrapper* capable of holding a potentially unlimited (subject to memory considerations) number of blocks (see Figure 2.2 for a high-level representation of the extended MLSB matrix). Thus, a static polymorphism based implementation had to be switched with a dynamic one (with dynamic overhead being minimal, since these are high-level objects). It should be noted that not all coupling blocks may be present in the system, for example there may be no direct coupling between geomechanical problem and facilities (wells). The container, visually represented as a two-dimensional grid of submatrices, must therefore allow some of them to be missing (null).

In order to properly adhere to the requirements of MLSB matrix interface, the necessary operations have been implemented in BlockMatrixWrapper, such as, extraction from AD residual, calculation of size, sparse matrix–vector product, algebraic reductions, output and other. Most of them delegate work to submatrices and then accumulate the results, if necessary. Care has to be taken to call submatrices in appropriate order, for instance, the coupling blocks depend on their respective diagonal master blocks for size calculations. Moreover, while most operations are easily parallelizable across the submatrices (using OpenMP shared–memory parallelization directives), one must watch out for potential data races and dependencies, and load balancing between threads is likely to be far from perfect, since the amount of work to be done per submatrix varies significantly.

An additional operation has been added to the MLSB interface and implemented for all matrix types — extraction into CSR format, which is required for better interoperability with external solver libraries. The top level block matrix wrapper performs this by extracting each submatrix into CSR and then recursively merging them, first within each block row (horizontally) and then merging together block rows (vertically). It should be noted that, due to row orientation of CSR, while a vertical merge is relatively cheap (just a concatenation of the three storage arrays and trivial update of row pointers), a horizontal merge is quite a bit more expensive, requiring interleaving data from multiple matrices, and a more optimal implementation of this process may be sought.

In addition to standard MLSB interface, the block matrix wrapper features additional methods for access to individual blocks, as well as extraction groups of blocks into a new block matrix wrapper object. In order to identify blocks by their contents rather than their position in the grid (which can change if the clients of linear solver framework decide to change the relative order of subproblems in AD residual), each subproblem is assigned a meaningful tag during matrix initialization, such as

RESERVOIR_BLOCK, *GEOMECHANICS_BLOCK* and so on. The user then provides a tag or set of tags for blocks they would like to access or extract. During extraction, a new block wrapper is created that contains pointers to original matrix blocks, but the blocks themselves are not copied; the new object is thus a *shallow copy* and operates in reference mode. If desired, a deep copy may be created as well, but this is usually not required, since all algorithm implementations leave the matrices untouched and create their own copies of data whenever an in–place modification is required. The shallow copy matrix wrapper represents a convenient object for performing matrix-vector operations with and forms the basis for our implementation of block-partitioned preconditioning.

2.3 Block-Partitioned Preconditioning

The Constrained Pressure Residual preconditioner considered in section 1.3.2 is a highly specialized preconditioner tailored to a particular set of equations. However, the basic idea of multistage preconditioning utilized in CPR can in principle be generalized to any given problem. Stages can be used to separate (decouple on the linear level) and tackle different physical phenomena or different spatial subdomains. In particular, in domain decomposition methods a well-known family of preconditioners are additive and multiplicative Schwartz, which have a direct correspondence to block versions of Jacobi and Gauss–Seidel iterations [18]. Similar ideas can be applied to the case where virtual 'subdomains' correspond to different mathematical equations. Based on these observations, a generic sequential preconditioning scheme for coupled problems was implemented in AD-GPRS as a baseline approach, against which other, more advanced schemes can be compared. We refer to it as *block–partitioned* preconditioning.

The design objective was to create a flexible implementation, that can be easily configured for any set of coupled equations at hand, provided that solvers for individual equations are available. To that end, the implementation is based on a recursive tree structure (see Figure 2.3). Each leaf node corresponds to a particular subproblem associated with a solver, and each internal node represents a set of subproblems (its child nodes) that are solved together to desired criteria (residual norm or a given number of iterations) via an iterative scheme — either Jacobi–like or Gauss–Seidel–like. The difference between the two block-partitioned schemes is the same as in their pointwise counterparts: in Jacobi all blocks are solved independently and then the right-hand side is updated with solutions before proceeding to the next iteration; while in Gauss–Seidel the right–hand side is updated after each block solve. Due to less lag in the right–hand side, Gauss–Seidel typically converges faster, at the cost of not being able to parallelize simultaneous solution of multiple bocks. The latter is much less of an drawback in large models, where each block is already large enough so that parallelization of individual block solvers yields significant benefits.

Listing 2.1 shows a simplified example declaration of a preconditioning tree node. In addition to control parameters such as iteration type, tolerance and number of iterations, each nodes also contains a list of associated block tags (leaf nodes have only one, while intermediate have multiple), that are used to extract the submatrix from the master block matrix, and a shallow copy block matrix with block pointers (described in Section 2.2). Intermediate nodes also keep a set of right–hand side and solution vectors for each block, as those will change during the iteration, and leaf nodes maintain a pointer to the solver (preconditioner) to be used for current block. During both setup and solve phases, block–partitioned preconditioner performs a depth–first traverse of the tree. At setup, the blocks are extracted from the given matrix (again, this operation is cheap as it is nothing more than creating pointers to blocks) and leaf preconditioner setup is performed. During solve, child branches may be traversed multiple times until the intermediate node convergence criteria are met.

Listing 2.1: Example of a preconditioning tree node declaration.

```
struct PrecondNode
{
  enum IterType { GAUSSSEIDEL = 0, JACOBI = 1 };

  bool isLeaf;
  BlockMatrixWrapper*        matrix;
  std::vector<BlockType>     block_types;

  // for intermediate nodes
  std::vector<PrecondNode*> child_nodes;
  std::vector<Vector>        rhs_ectors;
  std::vector<Vector>        sol_vectors;
  IterType                   iter_type;
  int                        num_iter;
  double                     tolerance;

  // for leaf nodes
  SolverBase*                precond;
};
```

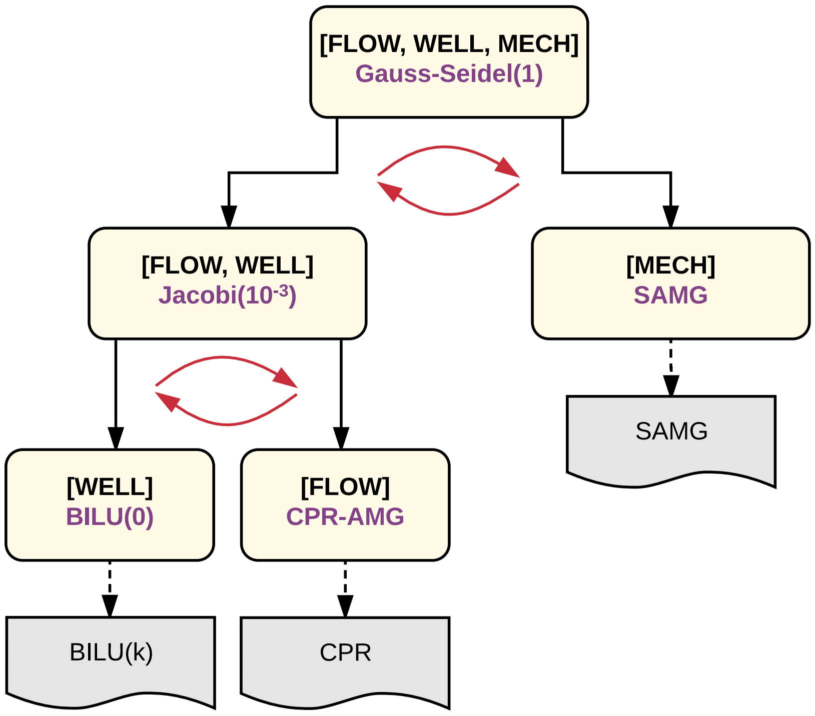


Figure 2.3: Example of a preconditioning tree structure. Top level is one Gauss-Seidel iteration between reservoir flow with wells and geomechanics, which amounts to just solving them in turn. Middle level includes a Jacobi iteration between reservoir and wells until a tolerance of $\varepsilon = 10^{-3}$, which means solving them independently, updating the right–hand side and iterating until convergence.

Thus, a solution strategy is defined recursively and may have multiple levels. For instance, we may wish to solve reservoir flow and facilities to a certain tolerance, and then combine with geomechanical solution in an outer iteration. As more and more subproblems are being added, more complicated strategies with many levels may be devised. Presently, however, the only problems considered are reservoir flow and transport with wells, and geomechanics. The design of CPR preconditioning scheme is monolithic, such that it handles the coupling between wells and reservoir internally (recall that well bottom hole pressure becomes a part of first–stage pressure system). Therefore, the scheme we use involves iterating between a CPR solution of reservoir with wells and a geomechanical solution (a multigrid solver is used). Gauss–Seidel iteration is used to better account for the strong coupling that exists between the flow and mechanical unknowns. Algebraically, this scheme can be represented as:

$$\mathcal{P}_{GS}^{-1} = \begin{bmatrix} \mathcal{P}_u^{-1} & B \\ & \mathcal{P}_{cpr}^{-1} \end{bmatrix} \qquad \text{where} \qquad B = \mathcal{P}_u^{-1} \begin{bmatrix} J_{up} & J_{us} \end{bmatrix} \mathcal{P}_{cpr}^{-1}$$

where $\mathcal{P}_u$ is a mechanics solver and $\mathcal{P}_{cpr}$ is the CPR preconditioner previously described.

The major disadvantage of the above schemes in their default version is that convergence may be far from optimal (and is not guaranteed in general case), unless special steps are taken to more properly account for the coupling between subproblems. The latter can be achieved through modifying the matrix blocks by introducing special kinds of relaxation matrices or approximate Schur complements. Next such an adjustment is derived, and its advantage over the Gauss–Seidel baseline approach is demonstrated in Chapter 3.

2.4 Fixed–stress CPR Preconditioning

In the previously described approach the geomechanical solution is completely isolated from the CPR strategy. This can be convenient because external black–box CPR and mechanics solvers can be used without any modification. However, in case of strong pressure-displacement coupling, this iterative procedure may take very long to converge due to oscillations in the residual introduced when one (or both) coupling blocks are ignored. In fact, it can be shown that Gauss-Seidel iteration on the linear level is analogous to the *fixed–strain* or *drained* sequential solution schemes, which are known to be only conditionally stable [11, 13, 12], meaning certain time step size restriction must be observed. A fully implicit solution is unconditionally stable, but linear convergence may be severely affected, as will be shown later in Chapter 3. On the other hand, the *fixed–stress* and *undrained* are unconditionally stable [11, 13, 12], which indicates they might also be good candidates for a linear solution strategy.

In [4, 25] a unified framework is introduced in the context of single-phase flow, in which sequential splitting schemes are re–interpreted as preconditioned Richardson iteration with particular choices of block–triangular preconditioning operators, and then compared with preconditioned Krylov iterations within the fully–implicit (*monolithic*) approach. Based on both theoretical argument and numerical experiments, the authors conclude that *fixed–stress* splitting is the best approach in terms of stability and efficiency, and propose a fixes–stress preconditioning scheme for fully or sequentially coupled single–phase flow and geomechanics. Extending their results, a flexible and robust preconditioner for fully coupled multiphase flow and mechanics has been proposed and implemented, building on a combination of CPR approach and single–phase fixed–stress scheme.

We begin by the observation that both pressure and displacement unknowns exhibit elliptic behavior with long-range coupling and produce corresponding low frequency error modes. Therefore it is advisable to isolate both of them from hyperbolic unknowns (advected saturations/compositions) through an algebraic reduction process similar to CPR True–IMPES. Essentially the same column summation operator $\mathcal{C}$ is applied to the J_{ss} and J_{ps} blocks yielding an approximated Jacobian:

$$\widetilde{J} = \begin{bmatrix} \widetilde{J}_{ss} & J_{su} & J_{sp} \\ J_{us} & J_{uu} & J_{up} \\ \widetilde{J}_{ps} & J_{pu} & J_{pp} \end{bmatrix} = \begin{bmatrix} \mathcal{C}(J_{ss}) & J_{su} & J_{sp} \\ J_{us} & J_{uu} & J_{up} \\ \mathcal{C}(J_{ps}) & J_{pu} & J_{pp} \end{bmatrix}, \tag{2.29}$$

after which two Schur complements may be formed with respect to the pressure and displacement variables, leading to a reduced first–stage Jacobian and right–hand side:

$$J^{(1)} = \begin{bmatrix} J_{uu} - J_{us}\widetilde{J}_{ss}^{-1}J_{su} & J_{up} - J_{us}\widetilde{J}_{ss}^{-1}J_{sp} \\ J_{pu} - \widetilde{J}_{ps}\widetilde{J}_{ss}^{-1}J_{su} & J_{pp} - \widetilde{J}_{ps}\widetilde{J}_{ss}^{-1}J_{sp} \end{bmatrix} = \begin{bmatrix} \widehat{J}_{uu} & \widehat{J}_{up} \\ \widehat{J}_{pu} & \widehat{J}_{pp} \end{bmatrix}, \tag{2.30}$$

$$\mathbf{r}^{(1)} = \begin{bmatrix} \mathbf{r}_u^{(1)} \\ \mathbf{r}_p^{(1)} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_u - J_{us}\widetilde{J}_{ss}^{-1}\mathbf{r}_s \\ \mathbf{r}_p - \widetilde{J}_{ps}\widetilde{J}_{ss}^{-1}\mathbf{r}_s \end{bmatrix} \tag{2.31}$$

This procedure can also be cast in terms of extended CPR restriction and prolongation operators:

$$R^{cpr} = \begin{bmatrix} -\widetilde{J}_{us}\widetilde{J}_{ss}^{-1} & I & 0 \\ -\widetilde{J}_{ps}\widetilde{J}_{ss}^{-1} & 0 & I \end{bmatrix}, \qquad P^{cpr} = \begin{bmatrix} 0 & 0 \\ I & 0 \\ 0 & I \end{bmatrix}, \tag{2.32}$$

$$J^{(1)} = R^{cpr}\widetilde{J}P^{cpr}, \qquad \mathbf{r}^{(1)} = R^{cpr}\mathbf{r}. \tag{2.33}$$

Additionally, one can use another simplifying assumption for preconditioning purposes, specifically assuming $J_{us} = 0$. By analyzing Equation (2.1), we can conclude that this holds true if capillarity effects are neglected and body force vector $\mathbf{f}$ is assumed constant (e.g. gravity only). It simplifies the reduction process above since now the displacement part remains effectively unchanged. This allows existing CPR codes to be reused without modification. The CPR implementation in AD-GPRS was adjusted in order to better isolate the restriction and prolongation operators and make them accessible to an outside caller (previously, they were embedded in a monolithic reduction process); but other than that, no modification was required to the code itself. It should be noted, that in cases with mild capillarity, the assumption $J_{us} = 0$ may still be appropriate for preconditioning purposes. With strong capillarity, the pressure p in Equation (2.1) should be treated as *effective* pressure (some form of average of phase pressures), and therefore a nonlinear saturation term would enter the equation, giving rise to a non–trivial J_{us}.

In order to further decouple the system, an efficient preconditioner described by [25] in the context of single–phase flow is employed, based on the fixed–stress concept. The LDU decomposition of the first stage Jacobian reads:

$$J^{(1)} = LDU = \begin{bmatrix} I & 0 \\ \widehat{J}_{pu}\widehat{J}_{uu}^{-1} & I \end{bmatrix} \begin{bmatrix} \widehat{J}_{uu} & 0 \\ 0 & S_{pp} \end{bmatrix} \begin{bmatrix} I & \widehat{J}_{uu}^{-1}\widehat{J}_{up} \\ 0 & I \end{bmatrix} \tag{2.34}$$

where $S_{pp} = \widehat{J}_{pp} - \widehat{J}_{pu}\widehat{J}_{uu}^{-1}\widehat{J}_{up}$ is a Schur complement with respect to $\widehat{J}_{uu}$.

A variety of preconditioning operators can be constructed based on this decomposition by preserving different factors. One particular choice is the upper triangular operator:

$$\mathcal{P}_{FS}^{-1} \approx U^{-1}D^{-1} = \begin{bmatrix} \widehat{J}_{uu}^{-1} & -\widehat{J}_{uu}^{-1}\widehat{J}_{up}S_{pp}^{-1} \\ 0 & S_{pp}^{-1} \end{bmatrix} \tag{2.35}$$

where $S_{pp}^{-1} \approx \mathcal{P}_p^{-1}$ and $\widehat{J}_{uu}^{-1} \approx \mathcal{P}_u^{-1}$ are approximated using two nested preconditioners for elliptic systems, such as multigrid methods. The algorithm for applying $\mathcal{P}_{FS}^{-1}$ to a residual vector takes advantage of its block–triangular nature by first computing the pressure components increment, $\delta\mathbf{x}_p^{(1)}$, followed by the displacement components increment, $\delta\mathbf{x}_u^{(1)}$, as follows

$$\delta\mathbf{x}_p^{(1)} = \mathcal{P}_p^{-1}\mathbf{r}_p^{(1)}, \qquad \delta\mathbf{x}_u^{(1)} = \mathcal{P}_u^{-1}(\mathbf{r}_u^{(1)} - \widehat{J}_{up}\delta\mathbf{x}_p^{(1)}). \tag{2.36}$$

After that, the solution vectors are combined to arrive at the first stage solution:

$$\delta \mathbf{x}^{(1)} = P^{cpr} \begin{bmatrix} \delta \mathbf{x}_u^{(1)} \\ \delta \mathbf{x}_p^{(1)} \end{bmatrix} = \begin{bmatrix} 0 \\ \delta \mathbf{x}_u^{(1)} \\ \delta \mathbf{x}_p^{(1)} \end{bmatrix}. \tag{2.37}$$

Finally, following the CPR approach, a local second stage preconditioner $\mathcal{P}_{sm}^{-1}$ is applied to the original system with an updated right-hand side, smoothing out the high-frequency local error modes from the residual:

$$\delta \mathbf{x}^{(2)} = \mathcal{P}_{sm}^{-1}(\mathbf{r} - J\delta \mathbf{x}^{(1)}), \qquad \delta \mathbf{x} = \delta \mathbf{x}^{(1)} + \delta \mathbf{x}^{(2)}. \tag{2.38}$$

Different smoothing techniques can be applied in this stage. One approach is to simply apply ILU or BILU family smoothers to the whole system, however, they are not likely to be effective due to the block structure and high bandwidth of the matrix. Based on the generalized block–partitioned preconditioner developed in the previous section, another approach would be to employ a block Gauss-Seidel iteration with separate BILU(0) or BILU(k) preconditioners on the flow/transport and mechanics parts. This, however, can be expensive, as matrix of the mechanical part can be quite large. One can note that the primary purpose of second stage is to smooth out the errors caused by saturation advection, and in cases without a direct dependence of displacement on saturation (such as when $J_{us} = 0$), second–stage updates to saturation will not cause the displacement solution to change drastically. It can therefore be beneficial to avoid the cost of constructing a factorization of geomechanical Jacobian and simply apply smoothing to pressure and saturation/composition, bearing the slight increase in iteration count.

Thus the overall scheme is a two–stage scheme with the first stage given by the fixed–stress block algorithm, and second–stage given by either block–partitioned

Gauss-Seidel smoother, or a simple ILU smoother on the flow and transport part only:

$$\mathcal{P}_{CPR-FS}^{-1} = \mathcal{P}_{sm}^{-1}(I - J\mathcal{P}_{FS}^{-1}) + \mathcal{P}_{FS}^{-1} \tag{2.39}$$

Finally, a feasible way of constructing the Schur complement in Equation (2.34) must be devised. Clearly, explicit construction of the full operator is infeasible due to the size of matrices involved and high order of fill generated. Therefore, a cheap and sparse approximation is sought. One possibility is implicit approximation by evaluating the effect of triple product $\widehat{J}_{pu}\widehat{J}_{uu}^{-1}\widehat{J}_{up}$ on vectors whenever required by the outer Krylov solver. Here $\widehat{J}_{uu}^{-1}$ can be approximated by the same preconditioning operator $\mathcal{P}_u^{-1}$ used in the solution stage, or a different one. This, however, can become prohibitively expensive with the cost of one extra invocation of $\mathcal{P}_u^{-1}$ per Krylov iteration.

We consider a sparse approximation $\widetilde{S}_{pp}$ that is constructed making use of the *fixed–stress* split concept [13]. Following [25], $\widetilde{S}_{pp}$ is explicitly computed from element–wise contributions as a pressure space mass matrix scaled by a weighting factor depending element–wise on the inverse of a suitable bulk modulus and Biot's coefficient. Note that this leads to a diagonal approximation $\tilde{S}_{pp}$ for a piecewise constant interpolation of the pressure field. As opposed to the original implementation [25], the construction of $\widetilde{S}_{pp}$ is here kept purely algebraic through the probing technique [18]. Denoting by $\mathbf{e}$ the probing vector, the diagonal approximation $\widetilde{S}_{pp}$ is computed as:

$$\widetilde{S}_{pp} = \widehat{J}_{pp} - \texttt{diag}(\widehat{J}_{pu}\mathcal{P}_u^{-1}\widehat{J}_{up} \cdot \mathbf{e}), \qquad \mathbf{e} = [1, 1, \dots, 1]^T \tag{2.40}$$

where $\texttt{diag}()$ is a diagonal matrix created from elements of the argument vector. This choice of approximation preserves row sums of the full operator by lumping elements within a row into the diagonal. Note that here the Schur complement assembly is only performed once per nonlinear iteration — or possibly per several, if the matrix does not change significantly. In addition, the constructed $\mathcal{P}_u^{-1}$ is reused later in the solution phase. Thus the extra cost of this approach, compared to, for instance, a Gauss-Seidel type scheme, is equal to one application of $\mathcal{P}_u^{-1}$ and two matrix-vector products. This cost is typically amortized across 10-30 Krylov iterations, and, as indicated by numerical results, the convergence benefits outweigh the extra cost.

This completes the description of the preconditioner developed in this work. It's worth noting that the implementation of the strategy, including generating the Schur approximation, is fairly straightforward in a framework with proper abstraction level, i.e. where matrix–vector operations are well-defined and all preconditioning operators have the same interface and can be treated as plug–in black–box solvers. The only time where low–level access to matrix storage is required is update of the pressure matrix diagonal elements while applying Schur complement.

Chapter 3

Numerical Examples

In order to assess the robustness and convergence of implemented preconditioners, a number of test cases have been set up, including completely synthetic as well as some derived from a well-known reservoir simulation benchmark SPE10 [5]. Here we discuss the results.

The general setup of the linear solver used in this chapter is as follows. The outer Krylov solver operating with a global coupled matrix is right–preconditioned GMRES [18]. While other iterations, including Richardson, BiCGStab and left–preconditioned GMRES, have been implemented, the focus here is on preconditioner efficiency; and Richardson iteration is too slow or fails to converge in a reasonable amount of time in some cases. The outer iteration runs with a convergence criteria of relative residual reduction of $\tau = 10^{-10}$ and true, rather than preconditioned, residual norm is used and reported. Additionally, GMRES is restarted every 100 iterations, and an overall limit of 1500 iterations is set.

First–stage pressure preconditioner $\mathcal{P}_p^{-1}$ is one V–cycle of algebraic multigrid [22], which is in practice enough to deal with long–range pressure error components for preconditioning purposes. First–stage mechanics solver is another flavor of algebraic multigrid, namely SAMG [21], well suited to non–scalar unknowns such as displacement vectors. Here we use multigrid cycling in conjunction with a built-in BiCGStab accelerator set to a loose tolerance of $\tau_u = 10^{-2}$, a value that provides a good balance between quality of approximation of J_{uu}^{-1} and performance — experiments show that using stricter tolerance does not provide enough overall convergence improvement to offset the higher computational cost. The second stage employs a block version of ILU(0) preconditioning, which is applied to the pressure–saturation system only. Note that in the actual implementation the pressure and saturation unknowns are not separated into $\delta\mathbf{x}_p$ and $\delta\mathbf{x}_s$, but interleaved and ordered cell-wise. Thus the flow and transport part of the Jacobian matrix is block-sparse with 2×2 dense blocks and is naturally suited to Block-ILU smoothing.

All tests were performed on a system with a single quad-core Intel Core i7–4790 CPU and 16 GB of RAM. A single–threaded version of AD–GPRS, 2016 release, was used for running simulations, linked with serial SAMG library.

3.1 Synthetic Flooding Problem

The first example is a poroelastic vertical slab of porous material initially saturated with oil (with water at residual saturation) shown in Figure 3.1. The slab is flooded with water from bottom to the top by subjecting its lower face to a fixed pressure and saturation boundary condition. The upper face is kept at the initial pressure and allowed to drain freely, while left and right faces are no-flow. Mechanically, the slab is assigned a linear elastic behavior, with right, left and bottom faces prescribed zero normal displacement, and top face allowed to deform. Both phases are slightly compressible fluids with contrasting properties (namely, viscosity). Gravity and capillary effects are neglected. Parameters of the system are summarized in Table 3.1.

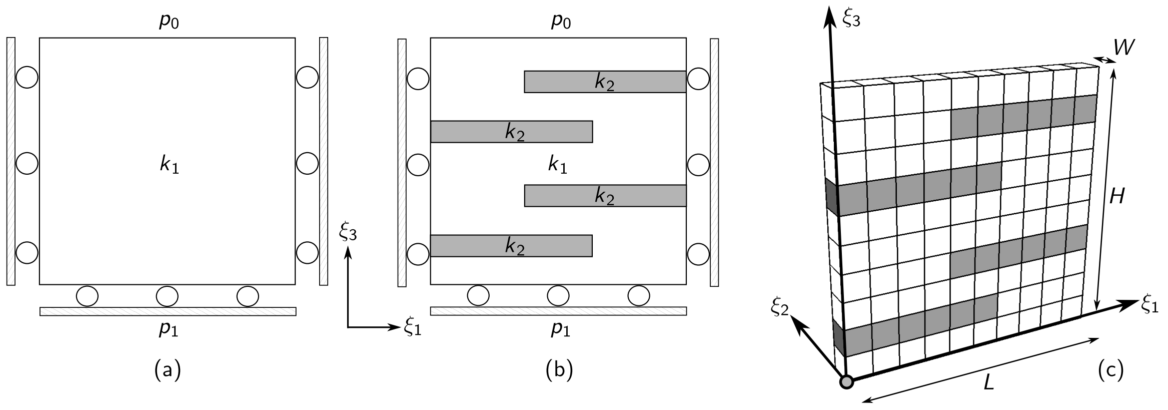


Figure 3.1: Synthetic flooding test, model setup: (a) homogeneous permeability, (b) with low–permeability barriers, and (c) simulation coarse grid ($h = 1$ m).

Table 3.1: Synthetic flooding test, model parameters.

Domain size	$L \times W \times H$	$10 \times 1 \times 10$	m^3
Young's Modulus	E	10^9	Pa
Poisson Ratio	ν	0.25	
Biot Coefficient	b	1.0	
Permeability	k_1	1	mD
Permeability	k_2	0.001	mD
Top Pressure	p_0	10^6	Pa
Bottom Pressure	p_1	3×10^6	Pa
Fluid Compressibility	c_f	2×10^{-10}	Pa^{-1}
Oil Viscosity	μ_o	3	cP
Water Viscosity	μ_w	0.3	cP

We consider two cases: one with homogeneous permeability, and one with embedded low-permeability barriers. Mechanical rock properties are kept homogeneous in all cases. Geometry of the models and boundary conditions are displayed in Figure 3.1. A Cartesian grid with characteristic mesh size h is used to discretize the domain. Table 3.2 reports problem space dimension for each of the grid sizes considered. Even though the problem is essentially 2–D (constant along y–axis), the grid is refined in all 3 dimensions. For both cases the convergence behavior is examined for time step sizes ranging from 10^{-5} days to 1 day and a variety of grid refinement levels. Convergence is compared between the combined Fixed–stress CPR preconditioner (denoted as $\mathcal{P}_{FS}$) developed in Section 2.4 and a similar, but simpler scheme, namely, a block Gauss-Seidel preconditioner (denoted by $\mathcal{P}_{GS}^{-1}$) described in Section 2.3.

Table 3.2: Synthetic flooding test, grid dimensions and problem size.

h [m]	$n_x \times n_y \times n_z$	# cells	# nodes	# DOFs
1.0	$10 \times 1 \times 10$	120	286	1,098
0.5	$20 \times 2 \times 20$	880	1,449	6,107
0.25	$40 \times 4 \times 40$	6,720	8,815	39,885
0.166	$60 \times 6 \times 60$	22,320	26,901	125,343
0.125	$80 \times 8 \times 80$	52,480	60,507	286,481
0.1	$100 \times 10 \times 100$	102,000	114,433	547,299

For each grid refinement level of both homogeneous and heterogeneous models, and each time step size, 5 time steps were run, and the number of linear iterations per Newton step was averaged across all nonlinear iterations of all time steps. Of primary interest here is in the convergence behavior as a function of time step size and grid block size. The expectation is that GMRES convergence with a good preconditioning scheme should not degrade significantly with mesh refinement. Results are summarized on Table 3.3 and presented visually in Figure 3.2.

These convergence results let us make the following observations. Using the Fixed–stress CPR preconditioner has a clear advantage over a simpler block Gauss–Seidel scheme, as it improves the iteration count by a factor of 1.5–3x over a wide range of parameters. Time step size is an important factor here, since it affects the relative scaling of flow/transport problem versus mechanics, and therefore the importance of the Schur complement. Specifically, we observe that iteration count peaks at time step sizes of 10^{-2}–10^{-3} days; note that this is a fully synthetically scaled problem, so the small time steps used in this test should not be dismissed as impractical. Grid

Table 3.3: Synthetic flooding test, average number of GMRES iterations per Newton step for the homogeneous permeability (a) and the low-permeability barriers case (b).

Δt [days]	h [m]	(a) $\mathcal{P}_{FS}^{-1}$	(a) $\mathcal{P}_{GS}^{-1}$	(b) $\mathcal{P}_{FS}^{-1}$	(b) $\mathcal{P}_{GS}^{-1}$
1×10^{-5}	1.0	28	32.17	51.17	58.5
	0.5	50	47	135.5	140.83
	0.25	54.17	50.33	154.57	158.43
	0.167	56	55.67	155.14	167.86
	0.125	57.17	58	153.5	177.5
	0.1	56	61.57	150.7	172.8
1×10^{-4}	1.0	28.6	33.6	54.17	61.5
	0.5	49.8	47.8	133.5	143.5
	0.25	50.5	51	144.75	152
	0.167	47.83	63	146.38	168.63
	0.125	46	63.33	147.29	189.57
	0.1	43.71	61.86	146.14	211.29
1×10^{-3}	1.0	32.67	37	56.71	60
	0.5	42.83	46.5	117.57	139.57
	0.25	37	48.75	136.83	160
	0.167	31.38	49.75	129.83	173.67
	0.125	27.29	50.86	118	190.56
	0.1	23.71	51.29	115.67	216.56
1×10^{-2}	1.0	33	35.14	46.67	48.5
	0.5	28.43	38.43	82.71	105.14
	0.25	18.67	38.67	99.4	132.1
	0.167	14.67	37.5	78.64	123
	0.125	12	37.67	66.64	119.55
	0.1	10.9	38.3	57.18	116.27
1×10^{-1}	1.0	20	22.14	35	38.29
	0.5	13	22.29	49.9	58.2
	0.25	9.4	23.2	41.09	55.82
	0.167	8.27	21.82	32.18	53.82
	0.125	7.82	22	26.64	53.18
	0.1	7.73	21.55	24.08	54.08
1×10^{0}	1.0	11	13.43	26.38	28.63
	0.5	7.38	13.38	24.13	26.63
	0.25	6.63	13	18.67	26.67
	0.167	7	12.89	16	25.33
	0.125	8.1	12.89	12.75	25.89
	0.1	8.6	12.44	12.92	26.63

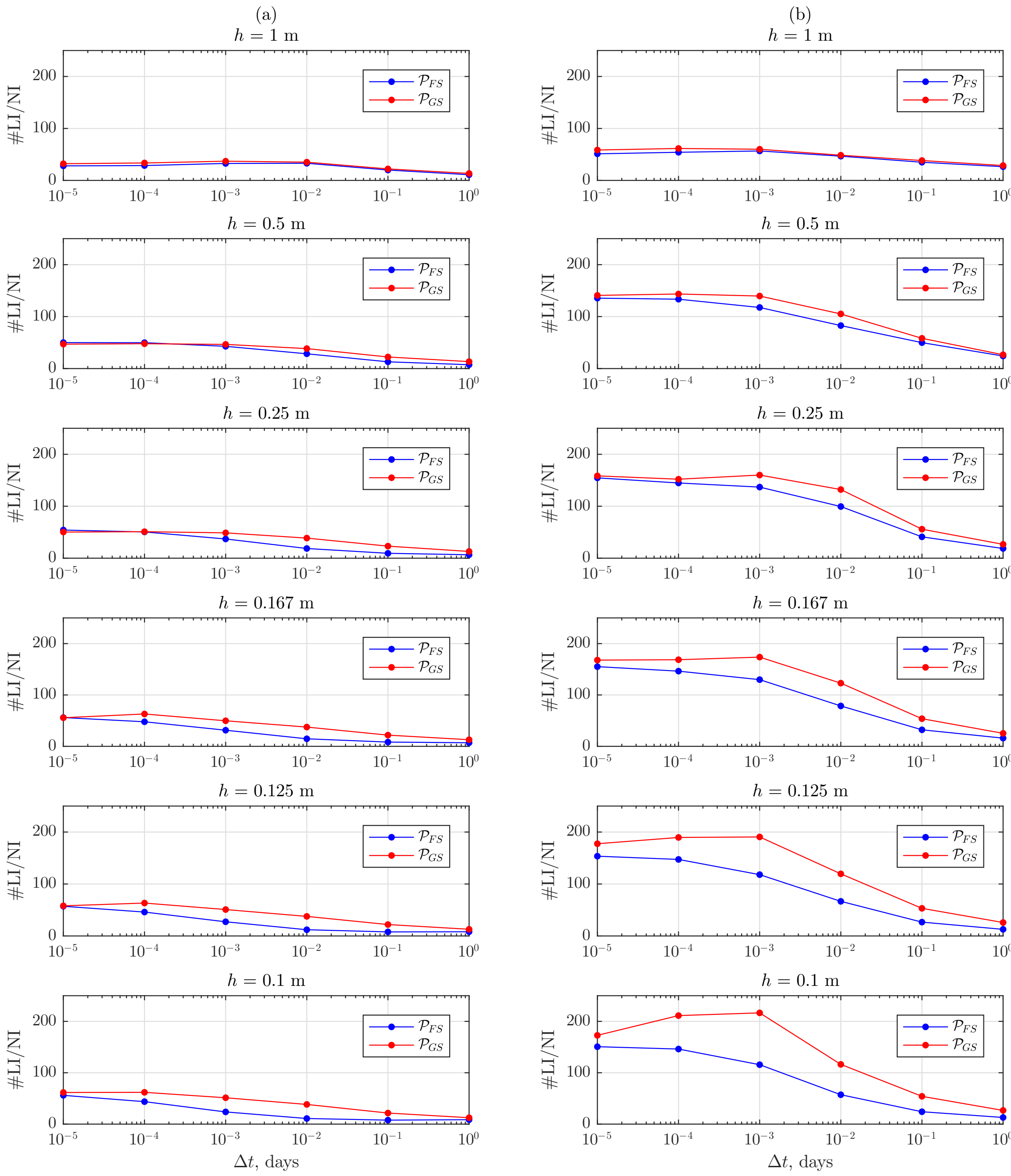


Figure 3.2: Synthetic flooding test, convergence results. Left column represents homogeneous runs, right – heterogeneous. FS abbreviates the Fixed–stress CPR preconditioner, while GS stands for Block Gauss–Seidel scheme.

refinement amplifies the effect, with the difference being barely noticeable on smaller grids and growing with the number of cells. Note that both solution algorithms suffer an initial increase in iteration count as the grid is refined 2 and 4 times; however, after that point the convergence of $\mathcal{P}_{FS}$ stabilizes, while for $\mathcal{P}_{GS}$ it continues to degrade all the way until the finest 100k cell grid (and is expected to continue to degrade with more refinement), making this approach infeasible on large grids.

In order to assess the performance of both schemes, the total linear solver CPU time is compared in Table 3.4 and the scaling with the number of cells is shown in Figure 3.3. In all cases, the Fixed–stress CPR method is superior in terms of runtime due to lower iteration count. It should be noted that both implementations used in comparison are not highly optimized for performance, however they are efficient enough that comparisons can be carried out. In terms of scaling, both schemes show a slightly higher than linear trend (a slope of ≈ 1.05 on the log–log plot), however lines for $\mathcal{P}_{GS}$ have a higher intercept indicating a larger coefficient in the big–O complexity. Thus, as mentioned in Section 2.4, the cost of extra operations in the Fixed–stress CPR scheme is largely outweighed by the improvements in convergence.

Table 3.4: Synthetic flooding test, linear solver time in seconds: (a) homogeneous permeability; (b) heterogeneous with low-permeability barriers.

h [m]	(a)		(b)	
	$\mathcal{P}_{FS}^{-1}$	$\mathcal{P}_{GS}^{-1}$	$\mathcal{P}_{FS}^{-1}$	$\mathcal{P}_{GS}^{-1}$
1.0	1.189	1.85	2.249	2.993
0.5	10.241	14.421	33.523	41.814
0.25	120.456	178.648	488.218	583.405
0.167	413.726	718.636	1640.732	2070.949
0.125	871.972	1689.172	3674.573	5716.947
0.1	1888.857	3880.485	7791.585	13512.506

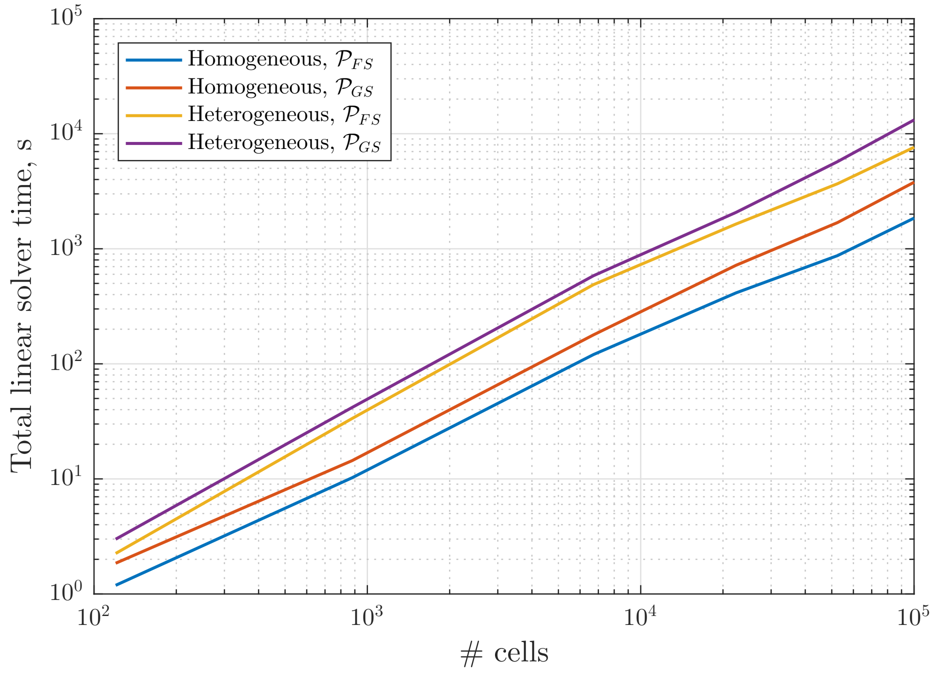


Figure 3.3: Synthetic flooding test, scaling results.

3.2 SPE10–Based Reservoir

Here we consider a more realistic problem setup and demonstrate that the proposed approach can be applied robustly to modeling multiphase reservoir flow with geomechanical effects. The test case is based on the SPE10 porosity and permeability fields, from which the top 32 layers have been extracted and additionally upscaled 2 and 4 times in each direction. Upscaling has been performed using the open–source MATLAB Reservoir Simulation Toolbox [16]. The sizes of each model are summarized in Table 3.5, displayed in Figure 3.4. The permeability field is characterized by high degrees of heterogeneity and anisotropy — although to a lesser degree in upscaled versions — and is generally considered to present a challenge for solvers.

The model is additionally equipped with poroelastic mechanical behavior with homogeneous properties. All boundaries are constrained to have zero normal displacement, except for the top one which is allowed to deform freely. The relevant

Table 3.5: SPE10–based test, grid dimensions and problem size.

$n_x \times n_y \times n_z$	# cells	# nodes	# DOFs
$15 \times 55 \times 8$	6,600	8,064	37,392
$30 \times 110 \times 16$	52,800	58,497	281,091
$60 \times 220 \times 32$	422,400	444,873	2,179,419

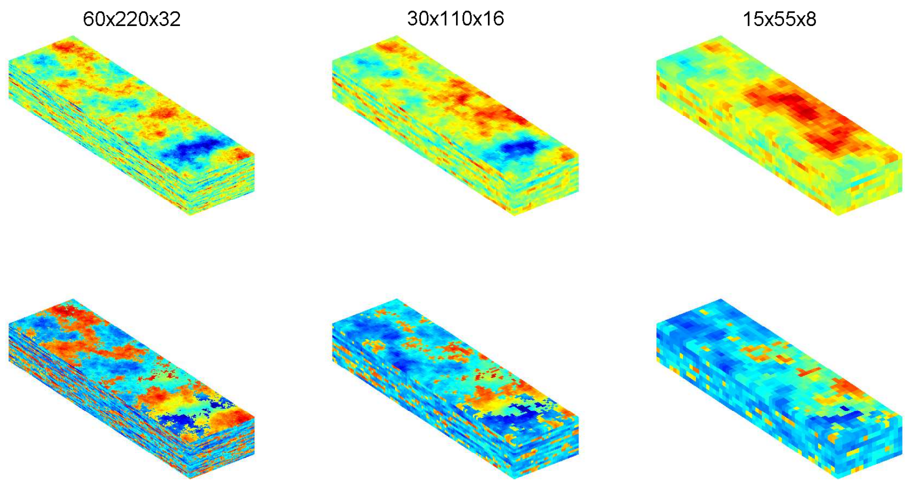


Figure 3.4: Permeability fields for SPE10-based test case: k_x, k_y (top row), k_z (bottom row).

physical parameters are summarized in Table 3.6. Multiphase dynamics is introduced by a pair of producing and water-injecting wells located in opposite corners of the domain (mimicking a quarter 5-spot) and penetrating all layers. Both wells are operating at constant bottom hole pressure, creating a pressure gradient across the reservoir that drives the flow and leads to formation compaction in some parts of the reservoir and formation expansion in others. This reservoir behavior cannot be correctly captured by using a simple uniaxial rock compressibility parameter due to boundary condition effects, however it can have a significant impact on production at early times. Thus having an efficient and robust coupled solver can be beneficial.

Table 3.6: SPE10-based reservoir: model parameters.

Domain size	$L \times W \times H$	$120 \times 220 \times 6.4$	m^3
Young's Modulus	E	3.1	GPa
Poisson Ratio	ν	0.2	
Biot Coefficient	b	0.8	
Injection Pressure	p_{inj}	50	MPa
Initial Pressure	p_{init}	30	MPa
Production Pressure	p_{prod}	10	MPa
Fluid Compressibility	c_f	4.35×10^{-10}	Pa^{-1}
Oil Viscosity	μ_o	3	cP
Water Viscosity	μ_w	0.3	cP

3.2.1 Convergence

Figure 3.5 shows convergence of FS–CPR preconditioned GMRES on the fine and upscaled grids for a variety of time step sizes. Convergence history was collected from the second Newton iteration of the first repetition of each time step size (the first Newton iteration is skipped as the matrix does not include an accumulation term and thus is not representative of most matrices the solver has to deal with during the course of the simulation). Generally it can be observed that for all except the largest (100 and 1000 days) time step sizes the linear solver reduces the residual by 10 orders of magnitude within 20-25 iterations; for large time step sizes this number is between 40 and 50 for large grids. It might seem like convergence deteriorates with increasing grid size, however, this is more likely to be the effect of upscaling, which tends to make the upscaled grids less heterogeneous and anisotropic, and thus less challenging for the solver. Thus we conclude that the proposed FS–CPR scheme can be reliably applied to more realistic reservoir modeling problems with a range of time step sizes. Note that even though pressure-displacement coupling strength is quite mild in this example and a simpler preconditioner might be sufficient, as mentioned previously, the FS–CPR scheme comes at only a small extra computational cost and therefore can be viewed as a universally applicable preconditioner for problems with coupled multiphase flow and mechanics.

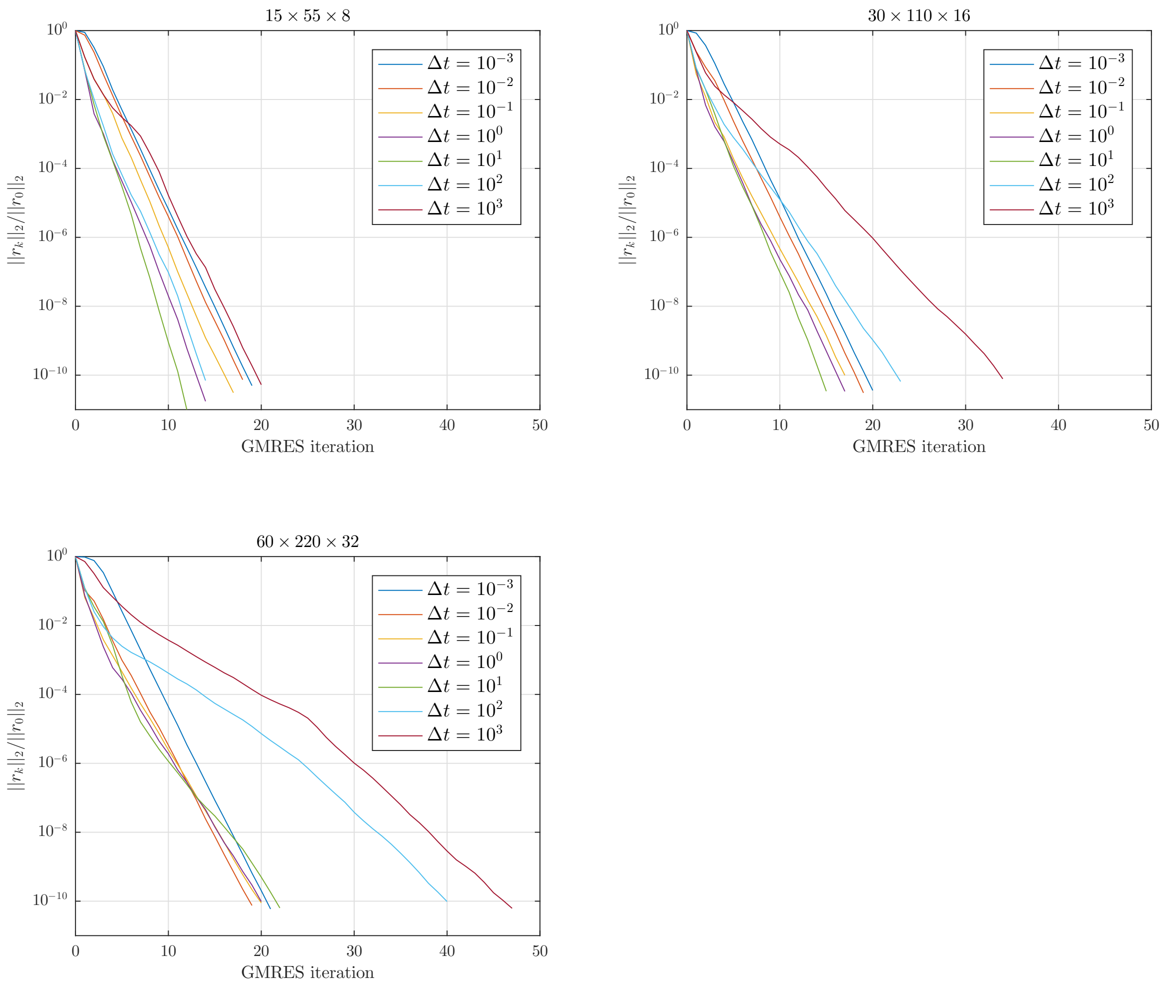


Figure 3.5: SPE10–based test, convergence on fine and upscaled grids. Time step sizes are in days.

3.2.2 Effect of capillarity

Additionally, we investigate the effect of capillary pressure on linear convergence. As was explained in Section 2.4, in the current implementation the block J_{us} is ignored, even though it can be present in presence of capillarity, since the effective pressure

p in Equation (2.1) is usually computed as volume (saturation) averaged fluid phase pressure:

$$p_{eff} = \sum_{p=1}^{N_p} S_p p_p \tag{3.1}$$

where

$$p_p = p_w + p_c(S_p) \tag{3.2}$$

with w being the wetting phase. Thus the derivatives of Equation (2.1) with respect to saturation unknowns, which make up J_{us}, are generally nonzero, and it is desirable to both better understand the limitations of current implementation and justify investing additional effort in the proper treatment of J_{us}. We use the smallest $15 \times 55 \times 8$ grid to investigate this, and the imbibition capillary pressure is modeled after [14] as:

$$p_c = p_c^{max}(1 - S_w^*)^m \tag{3.3}$$

where $m = 4$ is chosen and p_c^{max} is varied to derive models with different capillarity pressure curves. Figure 3.6 shows the GMRES convergence plots for different time step sizes and capillarity strength. It is immediately clear that for large enough time step sizes increasing capillary pressure severely affects convergence of the FS–CPR scheme. In order to assess whether this happens due to neglecting the J_{us} Jacobian contributions in the CPR–FS implementation or simply by the virtue of CPR stagnation with capillarity (due to the appearance of a diffusion–like saturation term that is not handled efficiently by the scheme), the same simulations were run with mechanics disabled, using the default CPR preconditioner. The results of these runs are shown in Figure 3.7, in the same format as Figure 3.6. The fact that convergence curves look almost exactly identical to the CPR–FS ones hints that the effect of capillarity on CPR completely dominates any convergence losses stemming from approximate Jacobian treatment in CPR–FS. In addition, Figure 3.8 shows the ratio of linear iteration count required by the two operators in their respective simulation runs. For smaller time step sizes CPR–FS takes a lot more iterations as it has to resolve the errors in mechanical solution (these time steps are taken in the beginning of simulation, when mechanical deformation is still ongoing), but for larger time steps the two operators very similar convergence. However, these results are not conclusive enough and a more in–depth investigation of this issue is in order.

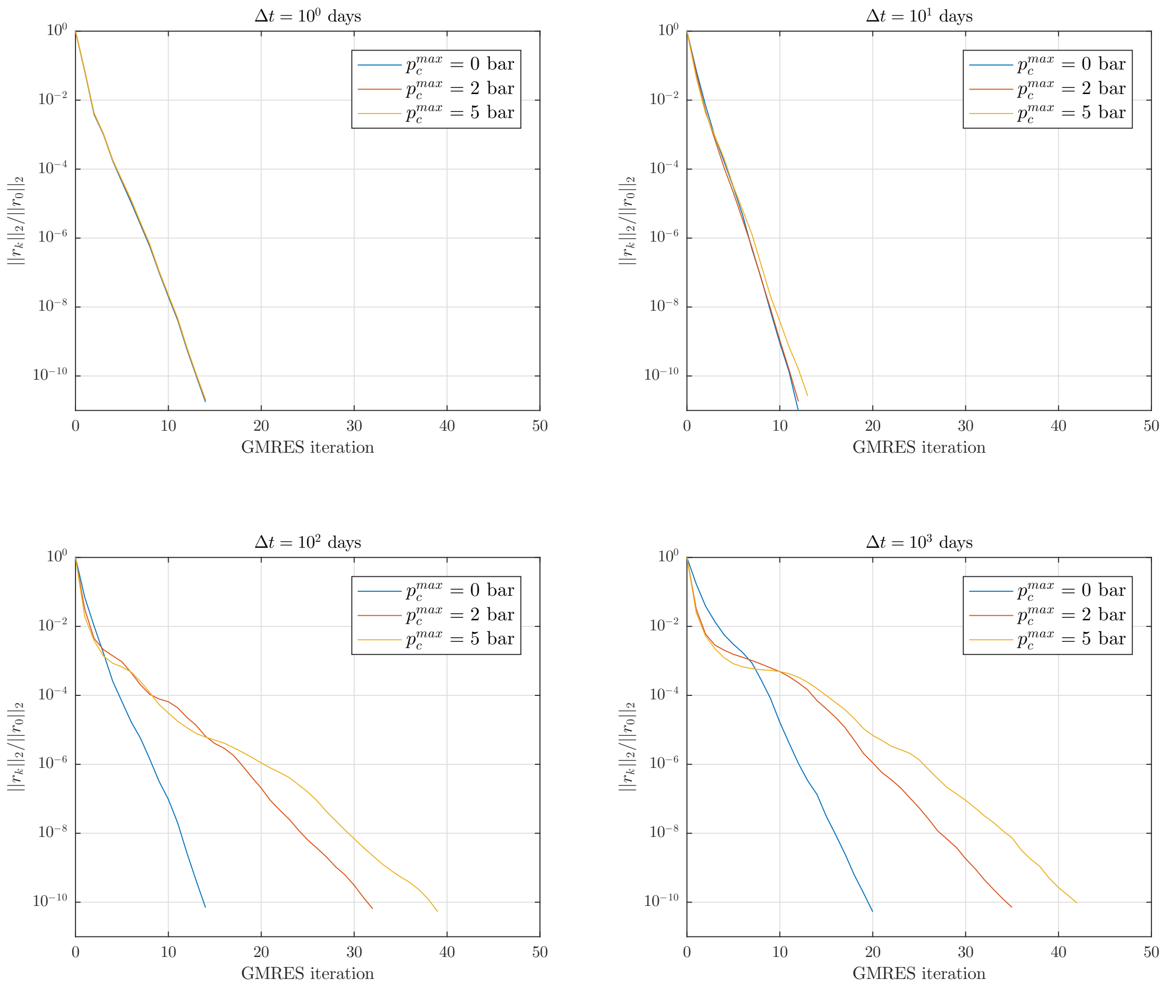


Figure 3.6: SPE10–based test, influence of capillarity on convergence. The lines overlap on top left plot ($\Delta t = 1$ day).

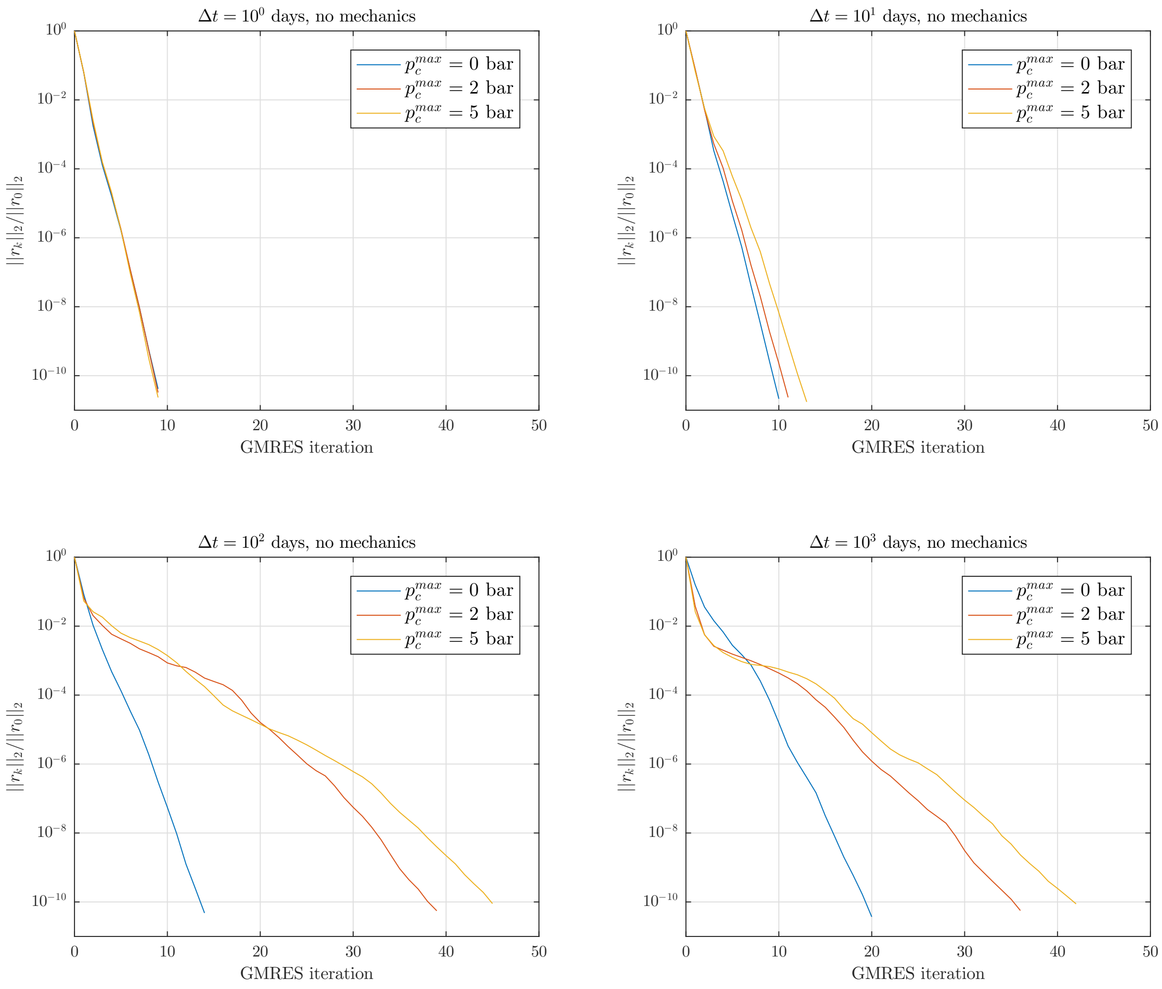


Figure 3.7: SPE10–based test without mechanics, influence of capillarity on convergence. The lines overlap on top left plot ($\Delta t = 1$ day).

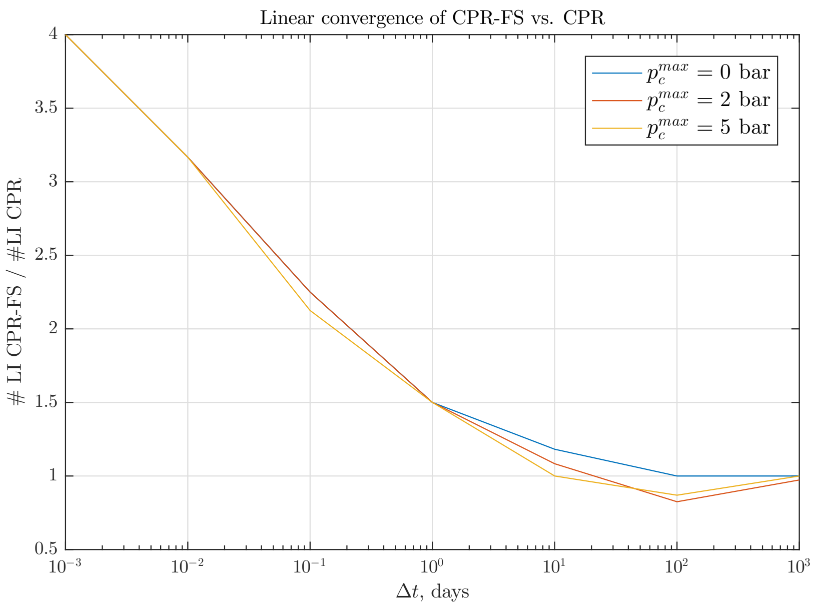


Figure 3.8: SPE10–based test, ratio of the number of linear iterations taken by CPR–FS vs. CPR scheme for a range of time steps and varying capillarity strength.

Chapter 4

Concluding Remarks

4.1 Conclusions

In present work the problem of achieving an efficient fully implicit solution of coupled multiphase flow and poromechanics has been tackled. A well implemented fully implicit simulator for a certain range of parameters may outperform sequential implicit schemes, since solution is achieved via an efficient Krylov subspace solver, rather than a fixed–point iteration that in the linear limit is equivalent to a Richardson scheme [25]. The linear systems arising during solution naturally possess a block structure and the major difficulty is the construction of a robust preconditioning operator, since simple preconditioners, such as ILU, are not sufficient. Such an operator has been constructed through an effective combination of the Constrained Pressure Residual preconditioner with the fixed–stress block-partitioned scheme. In order to deliver an efficient implementation within the research reservoir simulator AD–GPRS, existing data structures have been adapted and extended to handle fully implicit Jacobians with mechanics, while also laying the groundwork for a fully flexible linear solver platform in which specialized multiphysics preconditioners can be built and experimented with. In addition to the FS–CPR scheme for mechanics, a generic block–partitioned preconditioner that works via a Jacobi or Gauss-Seidel iteration has been implemented as a base building block for more complex schemes involving additional physics. The performance of both has been evaluated using a synthetically constructed and refined waterflooding problem, which revealed that the FS–CPR scheme outperforms simpler block–partitioned approaches and scales well by retaining constant iteration count as the number of grid elements is increased. Additionally, it has been applied to a more realistic waterflooding production scenario involving an SPE10–based permeability and porosity fields with added mechanical behavior. The preconditioner works reliably in absence of strong capillarity, however more effort is required to adapt it to capillarity by augmenting both the CPR and CPR–FS strategies with proper treatment of the capillary pressure terms and respective Jacobian blocks.

4.2 Future Work

While these initial results are very promising, a lot more needs to be done to achieve a fully robust linear solver. An obvious continuation of this work is to implement proper treatment of capillarity. Aside from that, it is quite important to investigate the relative impact of different kernels (pressure and mechanics solvers, second stage smoother) and tune their parameters (multigrid cycles, tolerances and other) to achieve maximum performance. More rigorous testing is required as well, including unstructured grids and heterogeneous mechanical properties, as well as performing 3–phase and/or compositional simulations to better assess the impact of multiphase physics on solver performance. Finally, building on the framework that is being developed for scalable multistage multiphysics preconditioning, advanced solution schemes for multiple fully coupled problems, such as thermo–hydro–mechanics, can be developed, incorporating ideas from sequential implicit operator splitting schemes. With additional effort invested in building specialized linear solvers, the fully implicit approach may become the method of choice for many multiphysics problems, especially ones that exhibit strong coupling.

Bibliography

[1] J. Bear. *Dynamics of Fluids in Porous Media.* Dover, 1972.

[2] H. Cao. *Development of Techniques for General Purpose Simulators.* PhD thesis, Stanford University, 2002.

[3] H. Cao, H.A. Tchelepi, J.R. Wallis, and H.E. Yardumian. Parallel scalable unstructured cpr-type linear solver for reservoir simulation. In *Proceedings - SPE Annual Technical Conference and Exhibition*, Dallas, TX, 2005.

[4] N. Castelletto, J.A. White, and Tchelepi H.A. A unified framework for fully-implicit and sequential-implicit schemes for coupled poroelasticity. In *Proceedings, ECMOR XIV - 14th European conference on the mathematics of oil recovery.* Catania, Italy, 2014.

[5] M. A. Christie and M. J. Blunt. Tenth spe comparative solution project: A comparison of upscaling techniques. *SPE Reservoir Evaluation and Engineering*, 4(04):308–317, 2001.

[6] O. Coussy. *Poromechanics.* John Wiley & Sons Ltd, Chichester, UK, 2004.

[7] R. H. Dean, X. Gai, C. M. Stone, and S. E. Minkoff. A Comparison of Techniques for Coupling Porous Flow and Geomechanics. *SPE Journal*, 11(01):132–140, 2006.

[8] James W. Demmel, Stanley C. Eisenstat, John R. Gilbert, Xiaoye S. Li, and Joseph W. H. Liu. A supernodal approach to sparse partial pivoting. *SIAM J. Matrix Analysis and Applications*, 20(3):720–755, 1999.

[9] E. Fjaer, R.M. Holt, P. Horsrud., A.M. Raaen, and Risnes R. *Petroleum Related Rock Mechanics.* Elsevier, Amsterdam and Oxford, 2nd edition, 2008.

[10] Y. Jiang. *Techniques for Modeling Complex Reservoirs and Advanced Wells.* PhD thesis, Stanford University, 2007.

[11] J. Kim. *Sequential Methods for Coupled Geomechanics and Multiphase Flow.* PhD thesis, Stanford University, 2010.

[12] Jihoon Kim, Hamdi A. Tchelepi, and Ruben Juanes. Stability, accuracy and efficiency of sequential methods for coupled flow and geomechanics. *SPE Journal*, 16(2):249–262, 2011.

[13] Jihoon Kim, Hamdi A. Tchelepi, and Ruben Juanes. Stability and convergence of sequential methods for coupled flow and geomechanics: Fixed-stress and fixed-strain splits. *Computer Methods in Applied Mechanics and Engineering*, 200(13-16):1591–1606, 2011.

[14] K. Li and R. N. Horne. An experimental and analytical study of steam/water capillary pressure. *SPE Reservoir Evaluation and Engineering*, 4(06):477–482, 2001.

[15] X.S. Li, J.W. Demmel, J.R. Gilbert, iL. Grigori, M. Shao, and I. Yamazaki. SuperLU Users' Guide. Technical Report LBNL–44289, Lawrence Berkeley National Laboratory, September 1999. `http://crd.lbl.gov/~xiaoye/SuperLU/`. Last update: August 2011.

[16] Knut-Andreas Lie, Stein Krogstad, Ingeborg Skjelkvåle Ligaarden, Jostein Roald Natvig, Halvor Møll Nilsen, and Bård Skaflestad. Open-source matlab implementation of consistent discretisations on complex grids. *Computational Geosciences*, 16(2):297–322, 2012.

[17] D. W. Peaceman. Interpretation of well–block pressures in numerical reservoir simulation. *SPE Journal*, 18(03):183–194, 1978.

[18] Y. Saad. *Iterative Methods for Sparse Linear Systems.* SIAM, Philadelphia, USA, 2003.

[19] Youcef Saad and Martin H Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7(3):856–869, July 1986.

[20] O. Schenk and K. Gärtner. Solving unsymmetric sparse systems of linear equations with paradiso. *Future Generation Computer Systems*, 20(3):475–487, apr 2004.

[21] K. Stüben. *SAMG User's Manual.* Fraunhofer SCAI, 2012.

[22] Klaus Stüben. Algebraic multigrid (amg): experiences and comparisons. *Applied mathematics and computation*, 13(3):419–451, 1983.

[23] D. Voskov and Y. Zhou. *Technical description of AD-GPRS.* Energy Resources Engineering, Stanford University, 2012.

[24] John Wallis. Incomplete gaussian elimination as a preconditioning for generalized conjugate gradient acceleration. In *Proceedings of the 7th SPE Reservoir Simulation Symposium*, San Francisco, CA, 1983.

[25] J. A. White, N. Castelletto, and H. A. Tchelepi. Block-partitioned solvers for coupled poromechanics: A unified framework. *Comput. Meth. Appl. Mech. Eng.*, 303:55–74, 2016.

[26] R.M. Younis. *Modern advances in software and solution algorithms for reservoir simulation.* PhD thesis, Stanford University, 2011.

[27] Y. Zhou. Multistage preconditioner for well groups and automatic differentiation for next generation gprs. Master's thesis, Stanford University, 2009.

[28] Y. Zhou. *Parallel General-Purpose Reservoir Simulation with Coupled Reservoir Models and Multisegment Wells.* PhD thesis, Stanford University, 2012.