TECHNIQUES FOR MODELING COMPLEX RESERVOIRS AND ADVANCED WELLS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ENERGY
RESOURCES ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Yuanlin Jiang

December 2007

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Dr. Hamdi Tchelepi Principal Advisor

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Dr. Khalid Aziz Advisor

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Dr. Roland Horne

Approved for the University Committee on Graduate Studies.

Abstract

The development of a general-purpose reservoir simulation framework for coupled systems of unstructured reservoir models and advanced wells is the subject of this dissertation. Stanford's General Purpose Research Simulator (GPRS) serves as the base for the new framework. In this work, we made significant contributions to GPRS, in terms of architectural design, extensibility, computational efficiency, and new advanced well modeling capabilities.

We designed and implemented a new architectural framework, in which the facilities (man-made) model is treated as a separate component and promoted to the same level as the reservoir (natural) component. The framework is quite general and extensible. It reduces the cost of incorporating new models into GPRS significantly. This is demonstrated by incorporating several new facility capabilities, namely, the Multi-Segment Well (MSWell) model and handling well-group constraints. Two new data structures have been built for GPRS to accommodate robust and computationally efficient simulation. They honor the separation of the reservoir and facilities components and give researchers freedom to develop their own discrete formulations and solution methods. The new data structures also simplify the construction and assembly of Jacobian matrices.

The MSWell was fully integrated into the new GPRS framework. This can be seen

as an example of a significant capability extension of a general-purpose simulator. Moreover, a new rigorous well-group treatment is proposed and implemented. This new model accounts for interactions between the wells in the group and allows for using one's favorite pipeline flow correlation. The well-group model benefits from the flexible data structures of GPRS. For example, one can combine any available facilities object such as standard and multi- segment wells and mixed well grouping of both kinds. With these new extensions, GPRS can simulate coupled systems of (unstructured) reservoir models and advanced wells.

A general discrete wellbore model (GenWell) that extends the MSWell treatment from the perforation all the way to the surface was developed. The GenWell model shares the advantages of the MSWell model, but it also accommodates complex pipeline topology, such as general branching, loops and multiple exits. In the GenWell model, wellbores and pipeline networks are discretized into various types of segments. Nodes and connections are defined based on the segments, and the discrete system is abstracted as a graph (nodes and connections) in a manner that is quite similar to the representation of unstructured reservoir models in GPRS.

As part of this work, advanced multi-stage linear solution strategies were developed. Based on the new block data structures, we developed and implemented block ILU preconditioners. These block preconditioners help further speed up the simulation. The two-stage CPR (Constrained Pressure Residual) preconditioning approach is extended to handle systems with the MSWell model. The data structures and linear solvers we developed are fully compatible with AIM (Adaptive Implicit Method). We demonstrate that the CPR preconditioned GMRES solver based on the new framework is the best linear solution strategy for large-scale AIM simulation of unstructured reservoir models and advanced wells.

Acknowledgement

I would like to express my sincere thanks to my advisers, Prof. Hamdi Tchelepi, and Professor Khalid Aziz for their support, encouragement and guidance during this work. Prof. Hamdi Tchelepi has always been deeply involved with the work and spent tremendous time on it. Prof. Khalid Aziz was my M.S. adviser and is my Ph.D. co-adviser. He has been a consistent source of knowledge, advice and support, since the first day I got to Stanford.

I would like to thank Prof. Roland Horne for reading my thesis and providing valuable suggestions and corrections. I also benefited from his contribution to the department as Chairman. Prof. Eric Darve kindly chaired my Ph.D. oral defense and is gratefully acknowledged. I would like to thank Prof. Louis Durlofsky for being my defense committee member. His valuable comments are highly appreciated.

I worked with Dr. Hua Shi and Jeremie Batias to integrate the MSWell model into GPRS. Their contributions are acknowledged. I also would like to thank Dr Huanquan Pan, who is the administrator of GPRS source code, for his support.

I would like to express my greatest gratitude to the Department of Energy Resources Engineering at Stanford University, all its faculty, staff and students. I feel lucky for being able to spend five years in this inspiring and friendly environment.

The friendships with them are the most important treasure of my life.

I would like to thank the industrial affiliates of the Stanford University Reservoir Simulation (SUPRI-B) and Advanced Wells (SUPRI-HW) research programs. Their financial supports made this work possible and are gratefully acknowledged.

Finally, I would like to thank my wife Weimin Sun for her unconditional support and for giving me the joy of life, our newborn son, Frank. I would also like to thank my mother-in-law for helping us take care of Frank. My parents and sister in China also deserve my gratitude for supporting and encouraging my education from the very beginning.

Contents

List of Tables

List of Figures

Chapter 1

Introduction and Objectives

1.1 Background

A modern oilfield is a very complicated system. The field may contain one or several reservoirs, a large number of wells, and complex surface production facilities. The system is dynamic and its various components may be tightly coupled. For example, during the life of a reservoir, recovery operations can change from simple depletion to waterflooding and perhaps advanced enhanced oil recovery (EOR) displacement processes. In practice, the number, type, and operating conditions of wells change as production operations and recovery strategies evolve. Pipeline networks may be reconfigured, or expanded, for optimal performance, and control devices and sensors may be installed throughout the system (surface and subsurface facilities) to monitor performance and manage field-scale production strategies dynamically. Numerical reservoir simulation serves as a primary tool for quantitative reservoir management, in which detailed modeling of such complex systems of reservoirs, wells, and surface facilities is required.

Reservoir simulation is the science of using computer programs to solve the equations the govern flow and transport in natural porous geologic formations and associated wells and production networks. The computer program - reservoir simulator - takes input information about the geometry and properties of the reservoir, and with appropriate initial and boundary conditions, the code simulates the petroleum recovery process being studied [2]. During the last 20 years, many new technologies have been developed in the oil industry, with which engineers can obtain much more information about reservoirs. Reservoir engineers integrate static and dynamic data from a variety of sources to build the reservoir models that are used for numerical simulation. These include structural, stratigraphic and property data obtained from seismic measurements, core analysis, well logs and transient well tests. Moreover, dynamic data including production and injection information, pressure and temperature measurement are collected in the field. All this static and dynamic information must be reflected in the long-term reservoir-management plan as well as the short-term field operations. The need to improve the quality of the predictions obtained using reservoir flow simulation has led to enormous growth in the reservoir characterization models, both in geometric complexity and level of detail, that are used. There has also been a comparable growth and development of well modeling technologies (e.g., horizontal, multilateral, and so-called smart wells), and the need for accurate numerical replantation of the flow behaviors in wellbores continues to grow. With these developments and the huge gains in computational power, there is strong demand for accurate and computationally efficient simulation of coupled large-scale reservoir models, advanced wells, and associated production facilities. Modern reservoir models may have millions of cells with highly complex structures and property distributions. Moreover, the physical models used to represent the dynamic recovery processes have grown in complexity; examples include compositional representations with large numbers of components, coupled thermal-compositional processes, geomechanical effects,

etc.

Researchers and engineers can use reservoir simulators to (1) design development plans for new fields, (2) improve the understanding about reservoirs with history matching, and (3) optimize the production under certain constraints.

While existing simulators can be used to perform some aspects of coupled real-field reservoir-facilities systems, there are several outstanding practical and computational research challenges that must be tackled. This dissertation is aimed at developing flexible and efficient computational research platform for numerical modeling of coupled systems of reservoirs, advanced wells, and production facilities. Such a platform serves to enhance the quality and efficiency of research efforts across a wide range of interest areas. Stanford's General Purpose Research Simulator (GPRS) [8] served as the starting point for this effort.

Next, we describe the overall general-purpose research simulation landscape, and we place this research effort in that broad context. Since this dissertation builds on, redesigns, and significantly extends the original GPRS simulator, the description given next is quite brief, with emphasis on the broad outlines of the previously missing capabilities in GPRS, as well as the new advances in general-purpose simulator design and solution methods that are developed as part of this work.

1.1.1 General-Purpose Reservoir Simulation

The literature related to the development of general-purpose reservoir simulation methods and tools is rich. This is particularly the case at Stanford, where it has been a long standing passion to develop research platforms for general-purpose simulation, and several researchers in the SUPRI-B/HW groups pursued various aspects of the

challenge. Efforts related to general-purpose numerical reservoir simulation can be traced back to the 1980s [3, 55]; in fact, these two references provide an excellent review of the various general-purpose thermal-compositional formulations that are still used by the reservoir simulation community, both in academia and industry. Since the mid-eighties, several simulation packages have been developed at Stanford, including FLEX [52], SPARTA [32], and Byer's simulator [6]. Based on this cumulative knowledge base, Cao developed a General Purpose Research Simulator (GPRS) in his Ph.D. work [8].

Cao's [8] primary objective was to allow for flexible nonlinear compositional formulations, so that researchers can compare existing methods and propose new and improved models. What distinguished GPRS form the previous work is that the computer code, which was written in C++, was designed and implemented with general-purpose reservoir simulation applications in mind. Since then, GPRS has grown significantly and has become a vital simulation research platform at Stanford and elsewhere. Since GPRS served as the starting point for the work reported in this dissertation, we provide a very brief summary of that simulator.

GPRS was developed based on object-oriented programming methods using the C++ language. A generalized compositional formulation was used in order to allow researchers to test and validate various choices for the variable sets used to describe compositional flows in porous media. GPRS employs a graph-based approach to represent the discrete equations and relations between reservoir cells (gridblocks), which was proposed by Lim [30], and Lim and Aziz [31]. Specifically, a connection list is used to describe the relation between the reservoir gridblocks. With this approach, one can handle generally unstructured reservoir models quite easily. For example, GPRS can be used to simulate models generated by discrete fracture modeling (DFM) approaches. Such models have extremely different cell sizes and complex

geometry [22]. Moreover, Cao [8] and Cao and Aziz [9] described the details of the IMPSAT (IMPlicit Pressure and SATurations, explicit compositions) formulation, and they derived the associated stability criteria based on a linear analysis. They demonstrated convincingly that for compositional problems, IMPSAT enjoys much better stability and overall computational performance compared to the IMPES (Implicit Pressure and Explicit Saturation) formulation. That development led to a multi-level AIM (Adaptive Implicit Method) approach, in which one can dynamically combine IMPES, IMPSAT, and FIM (Fully Implicit Method) in a single model to solve large-scale compositional problems efficiently. This powerful AIM framework is a significant contribution and is now considered a fundamental aspect of the GPRS simulation platform.

In recent years, GPRS has served as a research platform in the department as well as in some other institutes. Rather than coding a reduced version of a simulator from scratch, researchers can start from a well designed and actively maintained general-purpose, yet modular, code base. This dramatically reduces their workload and allows them to focus on their specific research interest. GPRS also acts as the integral container of the various research activities in the SUPRI-B/HW research groups. All of their research work may be preserved in GPRS, and it helps to push GPRS to a new level. When researchers leave, the newcomers may resume previous work easily. GPRS is one of the best vehicles to deliver the latest research work to the scientific community and the industry.

While GPRS had proved to be a flexible compositional simulator for unstructured reservoir models, it did not provide capabilities for modeling advanced wells (e.g., multi-segment wellbore) and coupling to surface facilities (e.g., well-group constraints and coupling to pipeline networks). In recent years, there have been several efforts aimed at modeling multiphase wellbore flows, and there has also been a major focus

on coupling strategies of reservoir models, advanced wells, and production facilities.

1.1.2 Discrete wellbore models

In the last decade, many ultra-deep, long horizontal, multilateral, wells have been drilled. These wells, which can be very expensive, can lead to complex wellbore flows. Accurate modeling of the flow behavior in wellbores, including frictional losses, multiphase flow effects, gas hydrates, is crucial for effective field operations. Meanwhile, intelligent wells, which have down-hole equipment, are developed and deployed in the field, and these add to the challenge of modeling the overall coupled system. The original Standard Well (StdWell) model cannot describe the flow behaviors in wells and production networks. Therefore, more complicated well models, in which the conservation laws for flow in wells are discretized and solved, have been developed.

The most common practice in reservoir simulation has been to treat wells as source/sink terms in the gridblocks they penetrate. In order to account for the disparity between the typical sizes of wells and computational gridblocks, a well model is usually employed. Nolen [33] describes the details of the Standard Well (StdWell) model, various extensions, and associated assumptions. In recent years, there has been strong interest in accurate and efficient modeling of the flow behaviors in advanced wells (e.g., horizontal, deviated, and multi-lateral). Holmes et al. presented a black-oil multisegment wellbore (MSWell) model [24]. In that discrete wellbore model, four variables, namely, mixture velocity, gas and water phase holdups, and pressure, are defined for each segment. The governing equations are the mass balance equations for oil, gas and water and one pressure relation (i.e., momentum balance) for each segment. Pressure loss due to gravity, friction, and acceleration are considered. Later, Stone et al. extended the multisegment well to compositional and

thermal simulation [48]. Shi et al. extended the drift-flux model for two-phase and three-phase flows, in which phase slip effects in the wellbore are accounted for [47, 46]. The multisegment model has been implemented in $Eclipse^{TM}$ [20], and it represents the state-of-the-art in terms of coupling reservoir models and advanced wells. However, the MSWell research developments of recent years [46] were implemented as an independent module; as a result, the MSWell model was not integrated into GPRS.

1.1.3 Coupled Reservoir-Facilities Simulation

In addition to new types of wells, new subsurface and surface equipment are being installed. For example, complex surface pipeline networks are installed to transport steam from the generator to the well sites and deliver oil and gas to tanks. All of these types of equipment have an important impact on the performance of the production system. We use the term 'facilities' to cover all of these objects, including wells, pipelines, sensors, separators, etc. Thus, the 'facilities' represent the aggregation of all man-made equipment installed in an oilfield. Several investigators, many at Stanford, had worked on coupling reservoir models and facilities.

For example, Schiozer and Aziz, and Schiozer [45, 44] presented a domain decomposition (DD) method to simulate reservoirs and surface facilities. They pointed out that the explicit treatment of the reservoir model and the facilities can lead to large errors. Small timesteps may help reduce error, but they make the overall performance prohibitively expensive. An iterative nonlinear solution strategy was proposed to solve the reservoir and facilities separately, followed by updating the boundary conditions until global convergence is reached. This can be a very expensive method. To improve the efficiency, a nonlinear preconditioner (different from the one used in iterative linear solvers) based on domain decomposition is applied at the beginning

of a timestep to predict the boundary conditions between the reservoir and the facilities. A fully implicit coupled reservoir-facilities simulation was investigated, and the conclusion was that the method is not very efficient, especially when the number of surface facility nodes is large. The iterative method with the nonlinear preconditioner based on DD was their preferred method for systems with complex facilities.

Later, Byer [6] and Byer et al. [7] extended Schiozer's work and presented a new solution strategy. Similar to Schoizer's work, they used a nonlinear preconditioner to predict the behavior of the reservoir and facilities models. However, the preconditioner was more sophisticated; it combined both DD and multigrid. An algebraically constructed coarse reservoir grid was, and the coarse-system solution was used to precondition the fully coupled problem. Byer also investigated an adaptive explicit coupling method. According to his results, a fully coupled system can be solved efficiently with that nonlinear preconditioner. The solution strategy developed by Coats et al. for coupled reservoir-facilities simulation is also based on domain decomposition [13]. In each Newton iteration, a subdomain including perforations and networks is decoupled. A smaller-size Jacobian matrix (compared with the global full Jacobian matrix) is formed and solved for this subdomain. The solution vector is used to update the unknowns of the perforations and the network. Then the global Jacobian (including both reservoir and network models) is constructed according to the latest results (updated perforations and network variables, and unchanged reservoir variables). The convergence of the Newton iterations guarantees that the final result is a fully implicit solution.

1.2 Objectives

One of the major objectives of this work has been to design and implement an efficient and extensible framework for compositional simulation of coupled unstructured reservoir models, advanced wells, and surface facilities. The reservoir and facilities models can have complex geometry and each may be discretized using very large numbers of computational elements. The new general-purpose framework should allow multiple researchers to work on GPRS at the same time, and it should also simplify the development work of each individual researcher. The impact of new models should be as local as possible, so that developers do not need to study the entire GPRS design and code base before they can contribute. This way, researchers can focus their efforts on their specific model developments.

In the original GPRS, the only available 'facilities' object was the standard well model. A major target of this research effort is the development and integration of new 'facilities' capabilities in GPRS, including discrete models for describing flow in wellbores and surface networks. The framework should allow for fully coupled treatment of reservoirs, advanced wells, and surface networks, and it should also be open to new models. A major first step is to integrate the multisegment well (MSWell) model [24, 47, 46] into the new GPRS platform. Another target is the development of rigorous and flexible treatment of well groups, and the integration of such models in the code base. We are also interested in the development of a general discrete wellbore/pipeline model that can handle general branching, loops, and counter-current flow efficiently.

As mentioned earlier, fully coupled treatment of reservoir-facilities systems is a difficult challenge. An important aspect of this dissertation is the development of

robust and computationally efficient linear-solution algorithms for coupled multisegment wells and complex unstructured reservoir models. This is quite different from previous work based on domain decomposition; here, we want to solve a consistent fully coupled system, where the global Jacobian matrix is always constructed using the latest iteration results.

1.3 Dissertation Outline

This dissertation proceeds as follows. Chapter 2 presents the architecture design we developed and implemented in the new GPRS. This is a new general reservoir-facilities framework, which separates the entire system into two parts: reservoir and facilities. General interfaces are designed for key components in the framework.

In Chapter 3, the basic data structures of the original version of GPRS are reviewed [8], and two new data structures are proposed and implemented in the new framework. These two data structures are built on top of the basic data structure, and they offer great flexibility and honor data encapsulation. Consistent with the framework design, the two data structures also separate the data for the reservoir model from the facilities components. Basic algorithms, e.g., matrix-vector multiplication and block ILU factorization, are implemented for these multilevel sparse block-based data structures.

In Chapter 4, the standard well model is reviewed. Based on the work of Holmes et al. and Shi et al. [24, 47, 46], a multisegment well (MSWell) model is implemented and integrated into the GPRS framework using the new multi-level data structures. This can be seen as an example of how to integrate new models into the redesigned GPRS framework. The MSWell model integration is investigated from several aspects,

including constraint equations, data structures, and initialization procedures.

In Chapters 5, a rigorous well-group model is developed and implemented in GPRS. The new approach accounts for the interactions between wells in the group and incorporates pipeline correlations. The model allows for fully coupled reservoir simulation with rigorous constraints on well groups. The new model is quite flexible and can easily accommodate new models to represent the subordinate well types.

Chapter 6 discusses our proposed general discrete wellbore model - GenWell, which can handle systems with complex branching and loops. This is an extended MSWell model. Objects and variables are defined differently compared to the MSWell model. GenWell can be used to simulate both wellbore and surface pipeline networks. The MSWell, well-group, pipeline models honor mass-balance strictly, and transient multiphase effects are accounted for throughout the system.

Chapter 7 covers the latest linear solution strategies in GPRS. The main focus of that chapter is on new robust and efficient multi-stage linear solvers for large-scale, compositional, unstructured reservoir models coupled with MSWells wells.

All these new developments, including the architectural framework, data structures, facility models, linear solvers and preconditioners, are fully compatible with the compositional AIM (Adaptive Implicit Method) formulation. The AIM simulation framework of the new GPRS is described and demonstrated in Chapter 8.

In Chapter 9, we make a summary of the work and draw conclusions. Several future work directions are recommended as well.

Chapter 2

Framework for Reservoir-Facilities Simulation

Oilfield production systems are made up of the *natural* underground reservoir(s), which contain the oil and gas resources, and the *man-made* facilities deployed for production operations, including wells, pipelines, pumps, separators, gathering systems, storage, etc. The physics of flow in natural porous media is different from that in wells and pipes, and the governing equations for flow in reservoirs and pipes are quite different. So, it is natural to decompose the oilfield system into two components: reservoirs and facilities. The two components are coupled through completions (open hole or perforations), which connect wells with the reservoir. The models used to represent the (natural) reservoir and the (man-made) facilities can be quite complex. Accurate representation of the flow behavior of coupled reservoir and facility models is an important aspect of state-of-the-art reservoir management. Therefore, a general simulation framework to incorporate various reservoir and facilities models is necessary. Such a framework should be computationally efficient in order to deal

with field-scale problems, and it should be readily extensible by researchers to new modeling ideas and solution strategies.

In this chapter, we propose a new general computational framework, based on an adaptive implicit formulation, that can flexibly handle various reservoir and facility models. This new framework has been implemented in Stanford's General Purpose Research Simulator (GPRS), and it has the following characteristics:

1. Flexibility: The framework can combine different models in one simulation case. For example, the user can have an unstructured reservoir model with different well types (e.g., standard and multi-segment wells) in one case. The framework should be able to incorporate new reservoir and facility models relatively easily.

2. Accuracy: The framework allows users to simulate multicomponent multiphase flow for highly detailed unstructured reservoir models and advanced wells accurately, including rigorous treatment of constraints on wells, well groups, and general production facilities.

3. Efficiency: The framework employs data structures and solution algorithms that lead to efficient and scalable computations.

4. Maintainability: The reservoir and facility models are implemented as separate components that communicate through a generalized interface. Moreover, within each component, the data and functions are encapsulated to the extent possible, and test modules are provided for components and sub-components. As a result, extensions and new developments will be largely localized. Moreover, the simulator is managed by a dedicated professional using sophisticated code-management software.

2.1 New GPRS Framework

In the original GPRS framework, the field (C++) class was responsible for constructing all the objects related to the reservoir model and the wells. In that design, the reservoir class was effectively at the highest level in the framework, and it was responsible for most numerical tasks, including time-step calculation, implicit level labeling, and Jacobian matrix assembly. Moreover, the reservoir class also owned the classes for the fluid system, rock properties, rock-fluid relations, computational grid, etc. In addition, the reservoir class also owned the objects and functionality of the wells and the linear solver. Figure 2.1 shows the original framework of GPRS

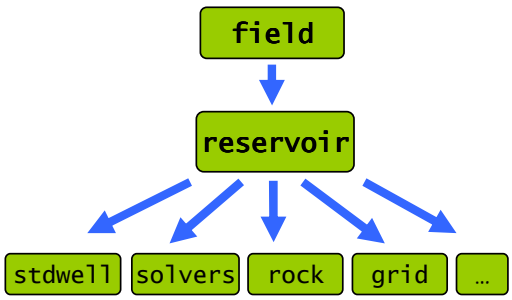


Figure 2.1: Original Framework of GPRS

The original framework reflected the traditional dominant position of the reservoir model in the simulation system. That design allowed for some level of extensibility. For example, new well models could be implemented into the framework as subordinates of the reservoir class. However, the new implemented well models may duplicate functionality that is already in place for the existing models and also lead to many inconsistencies in the interfaces, data structures, and solver related functions. These are not easy issues to resolve. The reservoir class of the previous design took on too

much responsibility, and that is counter to modular object oriented design principles. This extra functionality turned the reservoir class into a 'monster' class that is very difficult to maintain, or extend. The original framework did not reflect the important role of facilities in modern oilfield systems. The extensibility of the overall simulator was quite limited. Adding new models entailed significant changes in many components and sub-components.

We developed a more flexible, extensible, component-based framework to incorporate new research ideas for numerical modeling of coupled reservoirs and advanced facilities. A schematic of the new GPRS framework is shown in Figure 2.2. In the figure, blue arrows represent a subordinate relation and red arrows represent a 'derived-from' relation.

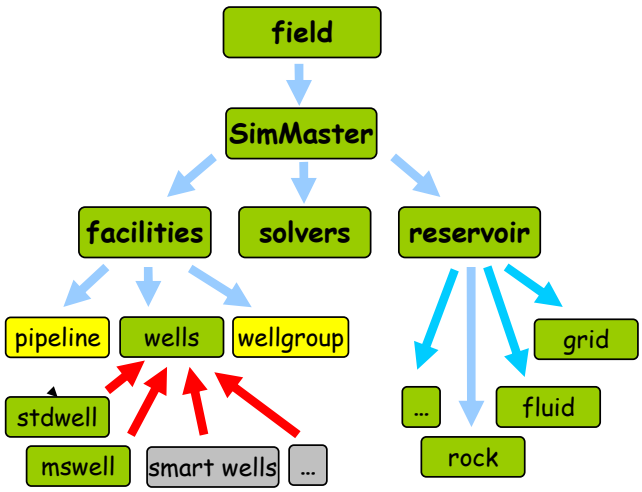


Figure 2.2: New Framework of GPRS

2.2 'SimMaster' and Reservoir Classes

In the new framework, we added a new class called 'SimMaster', which is the administrator of a system composed of directly related reservoirs and facilities. The system may contain one reservoir and a few wells, or multiple reservoirs linked by a common aquifer, wells, or a surface pipeline network. As the administrator, the SimMaster class takes over most of the simulation related numerical tasks from the original reservoir class, such as calculating timestep, allocating implicit levels and communicating with solvers. With respect to SimMaster, the solver class is a subordinate in the new design.

Since most of the numerical tasks and solver objects are taken over by SimMaster, the reservoir class has been greatly simplified. The new reservoir class focuses on describing the reservoir model, such as the properties of the grid, fluids, rock, etc. This is an improved object-oriented design. For the treatment of flow and transport in the reservoir, GPRS employs a generalized compositional-thermal formulation for both unstructured and structured grid with multi-level adaptive implicit treatment [8].

2.3 'Facilities' Class

The 'facilities' class represents the aggregation of all wells and other man-made equipment in the oilfield production system, and it is serves as a subordinate to the SimMaster class. In the new approach, the 'facilities' and 'reservoir' classes are at the same level. The two classes are equally important parts of the SimMaster class. This treatment reflects the importance of facilities in the overall system. The new facilities class has the well, well-group, and pipeline classes as subordinates. Note that the well class has no direct interface to either the reservoir or SimMaster classes. The well

class in this framework is an abstract class that defines a general interface to the facilities class and some common functionality for different subordinate well classes. The well class has the standard well (StdWell) and the multisegment well (MSWell) models as derivative classes. The detailed treatment of the StdWell and MSWELL models is given in Chapter 4. The new architecture is extensible. For example, a new model, e.g., well with intelligent completions and control devices, can be implemented in the framework as a new derivative of the well class. In the current framework, the new model automatically obtains the common functionality across well models. This facilitates code reuse and helps developers focus on new functionality requirements for the added model.

The 'well-group' class describes a group of wells connected to a junction point. The wells in the group share common constraints. A well group is composed of three parts: wells, pipeline connections, and a junction. The group may include a number of standard and multisegment wells. The junction, at which the pipelines join together, is the point where the constraint for the well-group is applied. The typical constraints are junction pressure and phase rate. In well-group modeling, the pipeline network connecting the wells is represented with a simple pressure correlation. The wells, however, are often discretized into segments, for which the mass and momentum balance are solved. Simulation of wells and well-groups is an effective tool for studying the injection and production rate allocation among interacting wells. Moreover, a multilevel nested well-group model can be used to simulate the entire production network. Details about modeling wells, well groups, and pipeline systems are presented in later chapters.

2.4 Interface of Key Classes

SimMaster, facilities, reservoir and solver are the four key classes in the new GPRS platform. Most data exchanges take place among them (Figure 2.3). Therefore, a general interface has been defined. For each timestep, SimMaster decides the timestep size and assigns it to both the reservoir and the facilities. The reservoir and facilities construct their own Jacobian matrices and residual equations using their own data and variables. They also exchange a small amount of information about the boundary to complete this step. For example, wells need information about perforated cells, and the reservoir model needs information about the wellbore pressure. All Jacobian related computations from the reservoir and facilities classes are delivered to the solvers, which are responsible for computing the numerical solution of the coupled system. The solution of the linear system is transferred back to the reservoir and facilities, so that both of them can perform a Newton update and calculate variables for the new iteration. Then, both the reservoir and facilities decide if they converged. Then, the convergence flags are delivered to SimMaster, which decides whether to proceed to the next timestep, or perform another iteration.

From this general interface (Figure 2.3), we can see that the facilities and reservoir classes play similar roles in the framework. They both share similar interfaces with other components. From a numerical simulation perspective, discrete representations of the governing equations, which describe the physics of interest, are used for both reservoir and facilities models. Typically, reservoir models are dominant, in terms of both physical size and the number of computational cells and variables used to describe the physics. However, there is growing interest in modeling coupled reservoir models and advanced facilities, such as multi-lateral and smart wells. In the new GPRS simulator, it is really up to the user how complex each of the reservoir

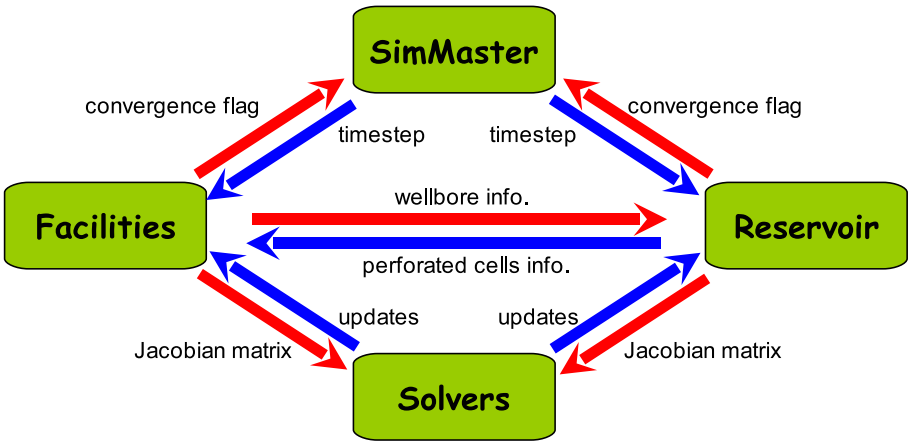


Figure 2.3: Interfaces between major components of GPRS

and facilities components are. For example, by using a very coarse reservoir model, but a very accurate multisegment well model, GPRS can be turned into a wellbore simulator. Well performance engineers can use it for their study interests, e.g., flow assurance. By using a very accurate discrete pipeline model, GPRS can be used as a multiphase transient pipeline simulator. Production engineers can use it to improve network design and optimize production operations. Of course, the main utility of GPRS continues to be efficient simulation of large-scale, heterogeneous reservoir models with unstructured grids. The new flexible framework enables us to have a combination of all of these components with an arbitrary level of complexity.

Chapter 3

Innovative Data Structures in GPRS

Reservoir models continue to grow in geometric complexity and resolution level. In recent years, there has been a strong focus on complex gridding techniques, and that is moving the simulation community toward unstructured reservoir models. The physical mechanisms that must be accounted for are also becoming more complex including coupled thermal-compositional processes involving large numbers of components. In addition, accurate modeling of geomechanical effects has become important. Facility models have also grown in complexity as more advanced wells (multilateral, horizontal), pipeline networks, and gathering systems are developed and integrated into reservoir management and production operations. Modeling the flow and transport in these facilities is a challenging problem and deserves a focused effort. In reservoir simulation, we are interested in integrated modeling of reservoirs and facilities in a flexible and computationally efficient manner.

In Chapter 2, we described the new GPRS framework where the reservoir and

the facilities models are treated as separate components. The physics, governing equations, and numerical treatment of flow and transport in reservoirs and facilities are quite different. Moreover, these systems are coupled and dynamic (i.e., their representation changes with time). In practice, this means that building a numerical simulation platform that is modular, efficient, and flexible, while allowing for tight coupling of reservoirs and facilities (i.e., all this information may go into a single global Jacobian matrix) is quite challenging. For that purpose, new flexible data structures were designed, and implemented in GPRS, with emphasis on compatibility with high-performance scalable computational algorithms.

The term 'data structure' refers to a scheme for organizing related pieces of information. A good data structure design has the following features:

- Accessibility: Users should be able to generate, access, and modify data easily.
- Encapsulation: Data should be hidden by the object that owns it, and other objects should have minimal dependency on the detailed format of the data.
- Extensibility: It should be easy to fit future developments into the data structure and expand it if necessary.
- Computational efficiency: The data structures should be compatible with state-of-the-art computational algorithms.

In this chapter, we first describe the fundamental data structures of GPRS. Then, the advantages and disadvantages of the original data structure will be discussed. After that, two newly developed data structures will be introduced. We will show how the new structures satisfy the four requirements listed above.

3.1 Original Data Structures in GPRS

Various pieces of information are stored in different formats in GPRS. Figure 3.1 shows the hierarchy of the data structures in GPRS, which will be introduced from top to bottom. At the topmost level, all the data structures in GPRS are split into two categories based on their storage formats: (1) raw data in arrays, and (2) matrices.

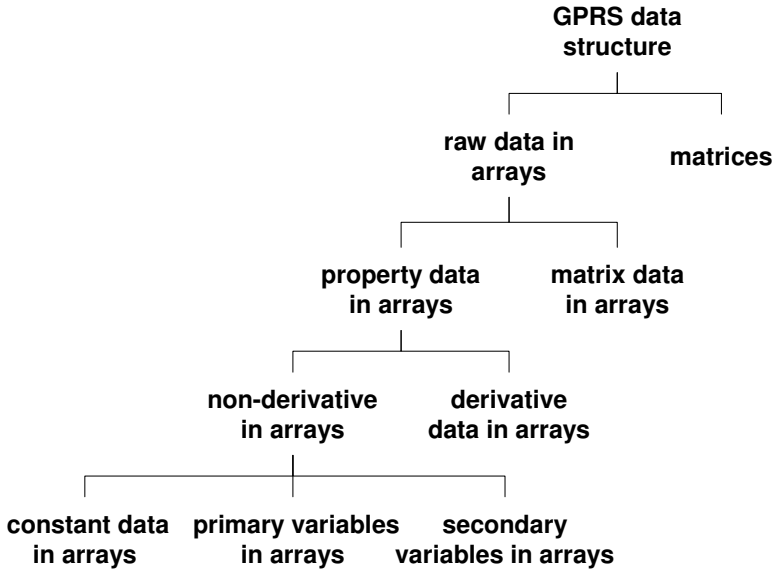


Figure 3.1: Hierarchy of the data structures in GPRS

3.1.1 Raw Data

The raw data is the data stored in arrays based on a certain order: cell-, connection-, or well-based, for example. The raw data is obtained from input files, or generated from simple property calculation functions. The raw data is stored contiguously in one dimensional arrays and can be easily accessed by indices. Direct operations on such data can achieve good cache utilization, if used wisely. The raw data in GPRS

includes property data and matrix data.

Property Data in Arrays

As suggested by their name, property-data arrays store the physical properties of the model. In order to optimize memory and cache utilization, the data is grouped according to property type, rather than cells, or connections. For example, there is one array that stores the pressure variables of all reservoir cells, by cell number. The length of the array is equal to the number of reservoir cells.

These property data have two subsets, and both of them have a very clear physical meaning. One is non-derivative data, such as saturation and porosity. The other one is derivative data, such as $\partial K_r/\partial S_w$, $\partial \phi/\partial P_o$. One of the primary uses of the property data is to generate Jacobian matrix information in the linearization process. For example, the derivative of the water-conservation residual equation with respect to the oil pressure of cell i, $\partial R_{w,i}/\partial p_{o,i}$, is an element in the Jacobian matrix. It will be calculated using property data according to the specific formulation of interest. The computed matrix data are also stored in arrays.

Matrix Data in Arrays

The derivatives of the governing discrete equations with respect to variables are stored in arrays. These derivatives are components of the Jacobian matrix, and they are stored using different data structures. In GPRS, three arrays are used to store the matrix information of the reservoir. They are the diagonal, upper off-diagonal, and lower off-diagonal arrays. In reservoir simulation, each cell may have several equations and variables, which leads to a natural block structure in the Jacobian matrix. The diagonal array stores the diagonal blocks, one by one in cell ordering. Within a small

block, the data is stored column-wise. The off-diagonal arrays are similar to the diagonal array, except that the small blocks are stored in the order of the connection list.

3.1.2 Compressed Row Sparse Matrix Format

GPRS supports various matrix formats, including dense, banded, and Compressed Row Storage (CRS) format. These data structures are used by the appropriate algorithms (e.g., linear solvers). The CRS format is the most frequently used format in GPRS, because it can represent general sparse matrices [37]. CRS is a widely used format in all kinds of simulation applications. Many algorithms have been developed for the CRS format, which is one of its biggest advantages. A new matrix format based on the CRS format is proposed and implemented in GPRS. We begin with a description of the CRS format here. A general sparse 5×5 matrix with 11 non-zero elements is used to illustrate the CRS format (in the following data structure, indices and labels are 0-based).

$$\begin{pmatrix} \mathbf{1.0} & \mathbf{2.0} & 0.0 & 0.0 & 0.0 \\ \mathbf{3.0} & \mathbf{4.0} & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & \mathbf{5.0} & \mathbf{6.0} & \mathbf{0.1} \\ 0.0 & 0.0 & \mathbf{7.0} & \mathbf{8.0} & 0.0 \\ \mathbf{0.2} & 0.0 & 0.0 & 0.0 & \mathbf{0.3} \end{pmatrix}$$

In the CRS format, there are three arrays, which are listed in Table 3.1 and 3.2. For a matrix with dimensions of $n_{row} \times n_{col}$ and nnz non-zero elements, the three

label	0	1	2	3	4	5
row_pointer	0	2	4	7	9	11

Table 3.1: Row pointer array of the CRS format

label	0	1	2	3	4	5	6	7	8	9	10
col_index	0	1	0	1	2	3	4	2	3	0	4
value	1.0	2.0	3.0	4.0	5.0	6.0	0.1	7.0	8.0	0.2	0.3

Table 3.2: Column index and value arrays of the CRS format

arrays are defined as follows:

- The 'value array' stores all the non-zero elements in the matrix row by row, whose length is always nnz.
- The row pointer (row_ptr) is a 0-based index array, which stores the position of the first non-zero 'value' in each row. Its length is $n_{row} + 1$ and always starts with 0 and ends with nnz.
- The column index (col_ind) is a 0-based index array, whose length is the same as the 'value array'. It stores the column index of each non-zero value.

A non-zero can be located using the 'row pointer array' and the 'column index array'. The 'value array' stores the non-zero element at that location.

Constructing a CRS matrix requires two steps. In the first step, the incidence matrix is built up based on the location of the non-zero elements. An incidence matrix function scans the equations one by one to count the non-zeros and builds the row-pointer and column-index arrays. In the second step, a filling function populates the matrix data (stored in the diagonal and off-diagonal arrays) into the value array of the CRS matrix. The elements are in different orderings, and complex indexing

operations are needed to fill-in the elements. We refer to these two steps as the Jacobian matrix assembly process.

In addition to complexity, the CRS matrix format lacks flexibility. It is almost impossible to add new elements into a constructed CRS matrix, unless you can foresee this request and prepare a stored 'zero' element at that location. For example, if the entry '1.0' needs to be inserted at (2,3) of the previous sample matrix, both the column index array and the value array have to be resized and most of their values need to be updated. The 'row pointer array' remains the same size, but needs to update its values. In such cases, the simulator has to completely destroy the matrix and build a new one. In reservoir simulation, the incidence matrix may change with time, due to changes in the treatment of unknowns in a cell from implicit to explicit, for example. Another common dynamic change is due to the addition or removal of wells, or a change in their operating constraints, or status. In each iteration, complicated indexing operations are needed to fill-in the new values (e.g., due to Newton updates) into the CRS format. Repeatedly calling these functions can be very expensive.

The CRS matrix format limits the extensibility of the simulator. It is very hard to incorporate new models into GPRS with CRS as the Jacobian matrix format. The matrix-skeleton building function and the filling function of the CRS matrix format need to know the exact location of every non-zero term. However, new models always introduce new equations and variables. Consequently there may be many new non-zero elements in the Jacobian matrix with very different fill-in patterns. With CRS, the matrix assembly functions need to be rewritten for every new model. If several different models are used together, the matrix structure can be very complicated. It is a nightmare to maintain this kind of data structure, and further extension can be extremely difficult.

In the following sections, we describe two newly developed data structures in

detail, and show how they can greatly facilitate the Jacobian matrix assembly process, retain flexibility, and achieve computational efficiency.

3.2 Multilevel Sparse Block Matrix

The Multi-Level Sparse Block (MLSB) matrix data structure is a hierarchical storage system, which corresponds to the component structures of the physical models. We developed this data structure for systems with many relatively independent components. For example, an oilfield production system, which includes reservoirs, wells, and surface facilities.

In this data structure, a higher level matrix is composed of a number of independent sub-matrices. The higher level matrix only contains some general information (e.g., the size of matrix) and the pointers to the sub-matrices. There are no requirements regarding the format of the sub-matrices, which means the overall system can be composed of several different types of matrices. The implementation details of the sub-matrices are completely hidden from the higher level matrix. This feature honors data encapsulation and achieves great flexibility and extensibility. New models can be integrated in GPRS without changing the high-level architecture. In the following subsections, we go through the matrix hierarchy from top to bottom.

3.2.1 First Level: Global Matrix

In general-purpose reservoir simulation, we often need to solve a coupled system that contains many complex objects such as reservoirs, wells, and surface facilities. Each of these objects has its own nonlinear equation and variable sets. Generally, the Newton-Raphson method is employed as the nonlinear solver. Therefore, in fully

coupled schemes, a global Jacobian matrix that incorporates all the derivatives of the different governing equations with respect to all the variables, is required. Figure 3.2 shows a Jacobian matrix generated for a three-dimensional structured reservoir model with several wells. The blue dots represent non-zero elements.

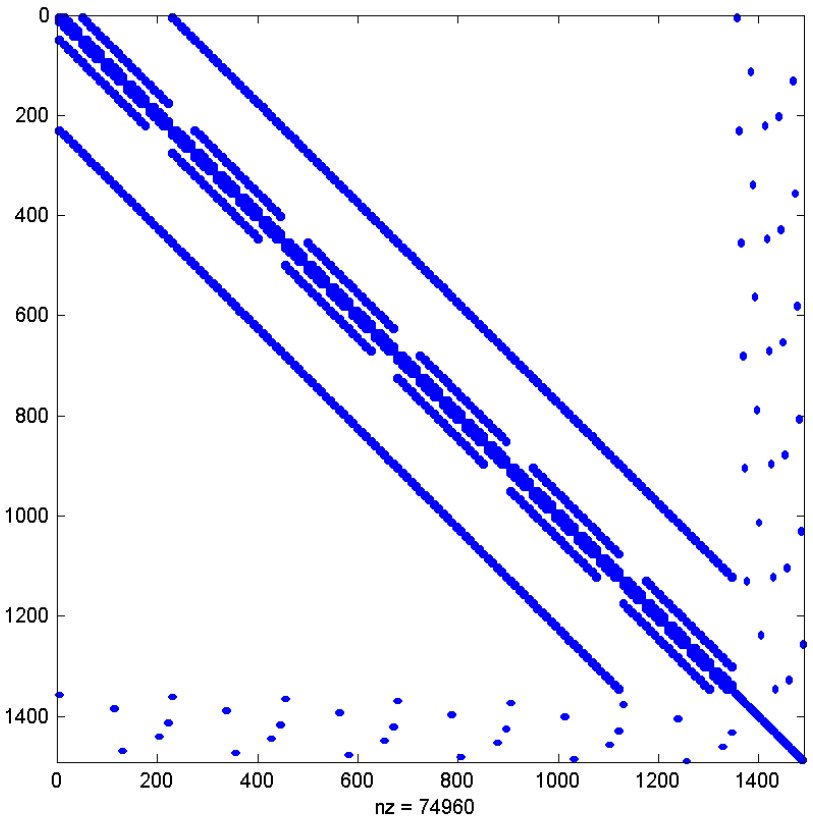


Figure 3.2: Typical matrix in reservoir simulation

The seven-diagonal banded structure in the matrix represents the reservoir equations and associated variables. The diagonal part in the bottom right corner represents the well equations and variables. The lower-left rows and upper-right columns represent the coupling terms between the reservoir and wells. In Chapter 2, we separated the facilities from the reservoir. Consequently, the global Jacobian matrix can be conceptually separated into four parts:

- $\boldsymbol{J}_{RR}$, derivatives of reservoir equations with respect to reservoir variables;
- $\boldsymbol{J}_{RF}$, derivatives of reservoir equations with respect to facilities variables;
- $\boldsymbol{J}_{FR}$, derivatives of facilities equations with respect to reservoir variables;
- $\boldsymbol{J}_{FF}$, derivatives of facilities equations with respect to facilities variables.

In the MLSB data structure, the first level is the global Jacobian matrix, which is defined as a wrapper. The matrix does not contain any substantial information other than handles to the four submatrices. The structure of the wrapper Jacobian matrix is shown in Figure 3.3. The wrapper has no requirement on the types of its submatrices. The prototype of the global Jacobian matrix is provided in the following pseudocode:

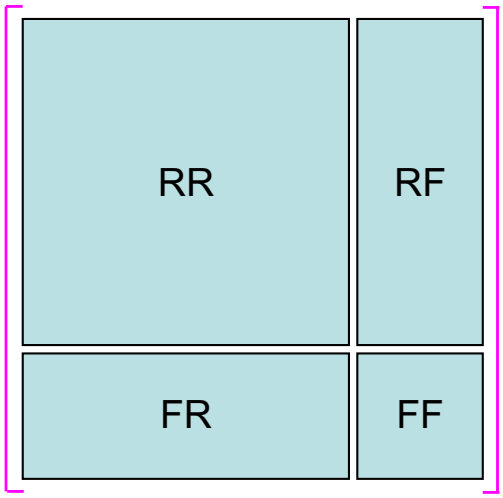


Figure 3.3: Submatrices in global matrix

```
template < class MatrixRRJ, class MatrixRFJ,
           class MatrixFRJ, class MatrixFFJ >
```

```
class SysMatWrapper {
    public:
        SysMatWrapper();        // Constructor
        operator*(Vector& v);   // Matrix-vector operator
    private:
        int nRows;              // No. of rows
        int nCols;              // No. of cols
        MatrixRRJ *mRRJ;        // RRJ pointer
        MatrixRFJ *mRFJ;        // RFJ pointer
        MatrixFRJ *mFRJ;        // FRJ pointer
        MatrixFFJ *mFFJ;        // FFJ pointer
};
```

The reservoir object in GPRS is in charge of building and managing $\boldsymbol{J}_{RR}$. The facilities object is responsible for the other three matrices. This design honors data encapsulation. Any update in the reservoir, or the facilities, is local and will not affect the global Jacobian matrix. An interface is provided, such that other objects may obtain access to the various submatrices. This structure give us great flexibility to design solution strategies. For example, we can get the decoupled facilities matrix $\boldsymbol{J}_{FF}$, without any cost since it is a stand-alone matrix. The same operation would be very expensive, if the global CRS format is used. Iterative solution strategies for the reservoir and facilities can be easily implemented. The details of the solution strategies will be covered in Chapter 7.

3.2.2 Second Level: Reservoir and Facilities Matrices

The four submatrices in the first level of the MLSB matrix have substructures of their own. The reservoir matrix contains information from the reservoir model. In order to improve data processing efficiency and avoid extensive memory usage, the basic data structure mentioned in Section 3.1 is reused. The major data in the $\boldsymbol{J}_{RR}$ matrix is the diagonal, upper off-diagonal, and lower off-diagonal data arrays. The reservoir matrix contains only handles to these basic arrays and some additional information, such as the implicit level of the gridblocks. Reusing the basic data structures significantly reduces the memory cost of the simulator. Any change in the basic arrays is equivalent to updating the matrix. Therefore, explicitly updating the matrix, which is an expensive operation, is avoided. The pseudocode for the reservoir matrix is presented as follows:

```
class SubRRBlk {
    public:
        SubRRBlk::SubRRBlk();   // Constructor
        operator*(Vector& v);   // Matrix-vector operator
    private:
        int nRows;              // No. of rows
        int nCols;              // No. of cols
        double *mDiag;          // Diagonal array
        doulbe *mOffD_A;        // Upper off-diagonal array
        double *mOffD_B;        // Lower off-diagonal array
        int *mImpLvl;          // Implicit level of cells;
};
```

The submatrices $\boldsymbol{J}_{RF}$, $\boldsymbol{J}_{FR}$, and $\boldsymbol{J}_{FF}$ are the responsibility of the facilities class, and each of them is a stand-alone matrix. As discussed in the previous chapter, the facilities class represents the aggregate of all the wells and surface facilities in the field. There may be hundreds of these objects in a field model. Each facility object, e.g., a multisegment well, owns a set of stand-alone matrices, named $\boldsymbol{J}_{RW}$, $\boldsymbol{J}_{WR}$, and $\boldsymbol{J}_{WW}$, respectively. $\boldsymbol{J}_{RW}$ is the section of the Jacobian matrix, which corresponds to the derivatives of the reservoir equations with respect to the well variables. $\boldsymbol{J}_{WR}$ and $\boldsymbol{J}_{WW}$ are defined in similar manner.

$\boldsymbol{J}_{FF}$ is a wrapper, and it contains the handles to the $\boldsymbol{J}_{WW}$ matrices from all subordinate facility objects. The structures of $\boldsymbol{J}_{FR}$ and $\boldsymbol{J}_{RF}$ are similar to the structure of $\boldsymbol{J}_{FF}$. Figure 3.4 shows a small reservoir model with two facility objects: one is a standard well, and the other is a well with four segments. The level-three matrix for the system is illustrated in Figure 3.5. The pseudo-code for the $\boldsymbol{J}_{FF}$ matrix is written as:

```
class FFJWrapper {
    public:
        FFJWrapper();           // Constructor
        operator* (Vector& v);  // Matrix-vector operator
    private:
        int nRows;              // No. of rows
        int nCols;              // No. of cols
        int nSubs;              // No. of submatrices
        std::vector<SubMat *> mMatList; // submatrix pointers
};
```

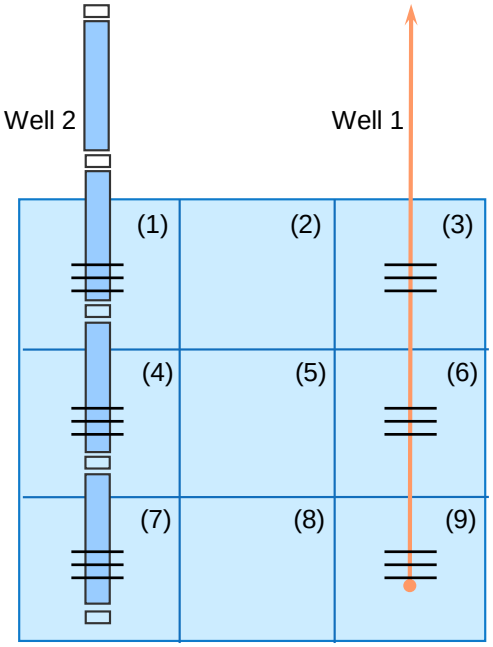


Figure 3.4: Sample reservoir with wells

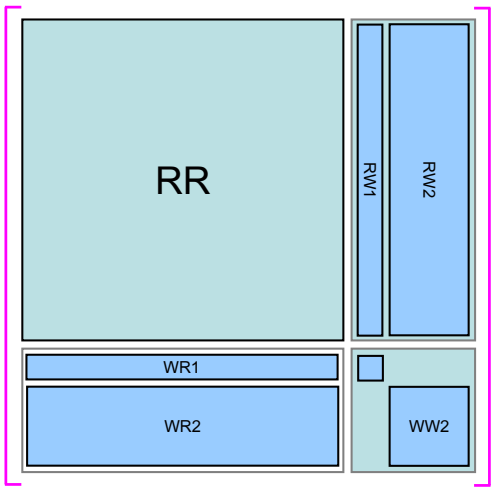


Figure 3.5: Submatrices in matFRJ, matRFJ and matFFJ

3.2.3 Third Level: Well Matrices

The matrices $\boldsymbol{J}_{RW}$, $\boldsymbol{J}_{WR}$, and $\boldsymbol{J}_{WW}$ are basic matrices, which may have substructures. As mentioned before, there is no format requirement for these basic matrices; they can even be full matrices. The implementation details are left for model developers. Here, we only discuss the data formats currently used in GPRS, which can serve as an example for future development.

Based on model types, the matrix set of $\boldsymbol{J}_{RW}$, $\boldsymbol{J}_{WR}$, and $\boldsymbol{J}_{WW}$ can be very different in both size and format. Figure 3.6 shows the structure of the $\boldsymbol{J}_{FR}$, $\boldsymbol{J}_{FF}$, and $\boldsymbol{J}_{RF}$ matrices. These illustrations correspond to the StdWell and the four-segment well shown in Figure 3.4. The blue blocks in the matrices represent non-zero elements. The layout of the multisegment well equations will be discussed in detail in Chapter 4.

In our implementation, this three-level MLSB matrix data structure forms the new foundation for GPRS. Within this structure, we can have arbitrary levels of matrices. This may be necessary when the system gets more complicated. For example, the global matrix may have four, or even more, levels, when a well-group model is incorporated into the system. This will be discussed in Chapter 5.

3.2.4 Matrix-Vector Multiplication

There are many linear solver options in GPRS. Among them, Krylov subspace solvers are used most frequently. These solvers can solve large general sparse matrices efficiently with the aid of good preconditioners [42]. Krylov solvers have no requirement for the matrix format and can solve any matrix for which a matrix-vector multiplication operation defined. The details of linear solvers and solution strategies are covered

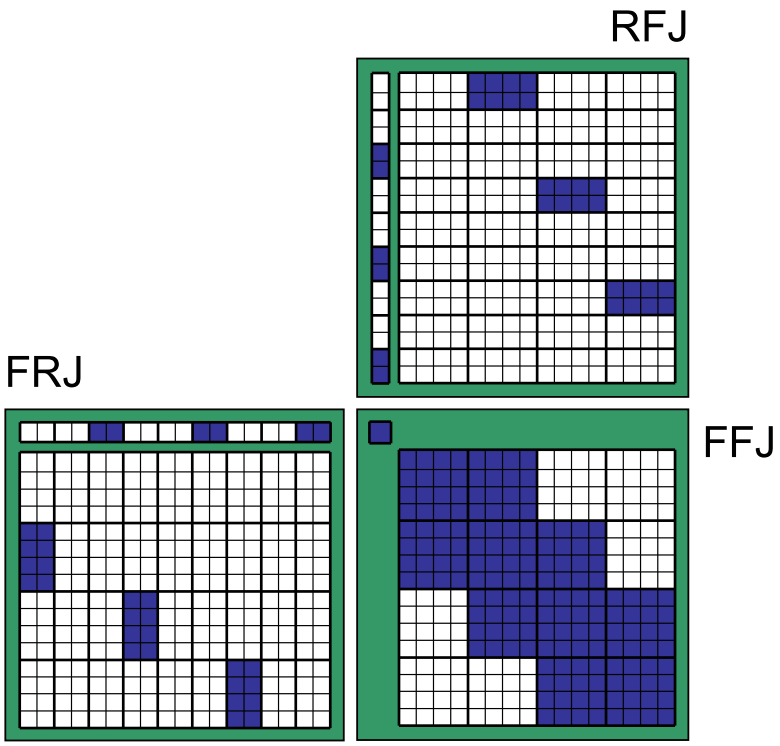


Figure 3.6: Sample FRJ, FFJ, and RFJ matrices

in Chapter 7. Here, we only focus on the matrix-vector operation of the MLSB matrix format.

As mentioned before, the global Jacobian matrix in the MLSB format has handles to the four sub-matrices in a 2×2 form. The details of the reservoir matrix, or the number and type of facility objects, are not visible to the global matrix. However, this does not prevent us from defining a generic matrix-vector operation. The operations of a high level matrix are defined based on operations on the sub-matrices. For example, the matrix-vector operation involving $\boldsymbol{A}_{global}$ can be defined as follows:

$$\boldsymbol{A}_{global} \cdot \boldsymbol{b} = \begin{pmatrix} \boldsymbol{J}_{RR} & \boldsymbol{J}_{RF} \\ \boldsymbol{J}_{FR} & \boldsymbol{J}_{FF} \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{b}_r \\ \boldsymbol{b}_f \end{pmatrix} = \begin{pmatrix} \boldsymbol{J}_{RR} \cdot \boldsymbol{b}_r + \boldsymbol{J}_{RF} \cdot \boldsymbol{b}_f \\ \boldsymbol{J}_{FR} \cdot \boldsymbol{b}_r + \boldsymbol{J}_{FF} \cdot \boldsymbol{b}_f \end{pmatrix}, \tag{3.1}$$

where $\boldsymbol{b}_r$, $\boldsymbol{b}_f$ are the reservoir and facilities parts of the vector, $\boldsymbol{b}$. $\boldsymbol{J}_{RF}$ has sub-matrices, so its matrix-vector operation is based on the matrix-vector operations of its sub-matrices, and it can be written in a similar manner to Equation 3.1. Hence, as long as the matrix-vector operations for the most basic matrices are defined, the matrix-vector operation for all levels of matrices can be easily constructed. The model developers need to propose, or adopt, matrix formats for their models, and they are also responsible for providing the matrix-vector operations for their matrices.

3.3 Block Compressed Row Sparse Matrix

In the previous sections, we presented a new matrix data structure, which has great flexibility, and that allows for reusing the basic data structures. The matrix-vector operation is defined for the overall matrix; therefore, it can be used by Krylov subspace solvers. However, it is very difficult to implement other algorithms, such as

LU factorization for systems composed of sub-matrices. Since the incomplete LU family of factorizations is a very important preconditioner class, we developed a new computationally efficient data structure that makes use of the basic data arrays.

As discussed before, the CRS matrix format is one of the most frequently used data structures in linear algebra. It has many compatible algorithms, including solvers and preconditioners. For example, SPARSKIT [42] provides many ILU solvers for CRS matrices. We define a new class named 'comprow_Mat_pointer', which is based on the CRS matrix format. The difference is that the elements in the matrix are not of common type, such as double, float, or integer. Instead, they are pointers. These entries point to the corresponding data blocks in the basic data arrays, as shown in Figure 3.8. This pointer matrix and the basic data arrays can be employed to represent a block matrix, as shown in Figure 3.7. The pointer matrix and the basic data arrays form our block compressed sparse row (BCRS) data structure. The pointer matrix has the following characteristics:

1. Easy to build and maintain;
2. Less memory cost than the original pointwise data structure;
3. Many algorithms available;
4. Robust and efficient;
5. Compatible with AIM (Adaptive Implicit Method).

Most algorithms that can make use of the standard CRS matrix format can be applied to the block CRS matrix format with little modification. All we need is to replace arithmetic operations with small matrix operations. Table 3.3 shows a few examples. The block version of the ILU factorization is the most important algorithm used with this matrix data structure in GPRS.

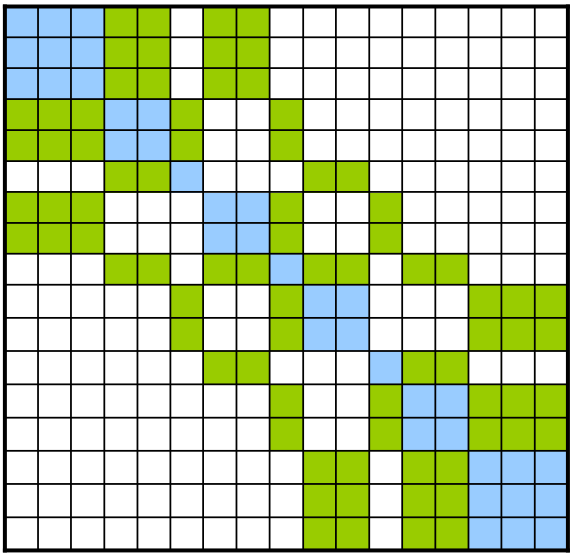


Figure 3.7: Mapped block compressed row matrix for a simple AIM Jacobian

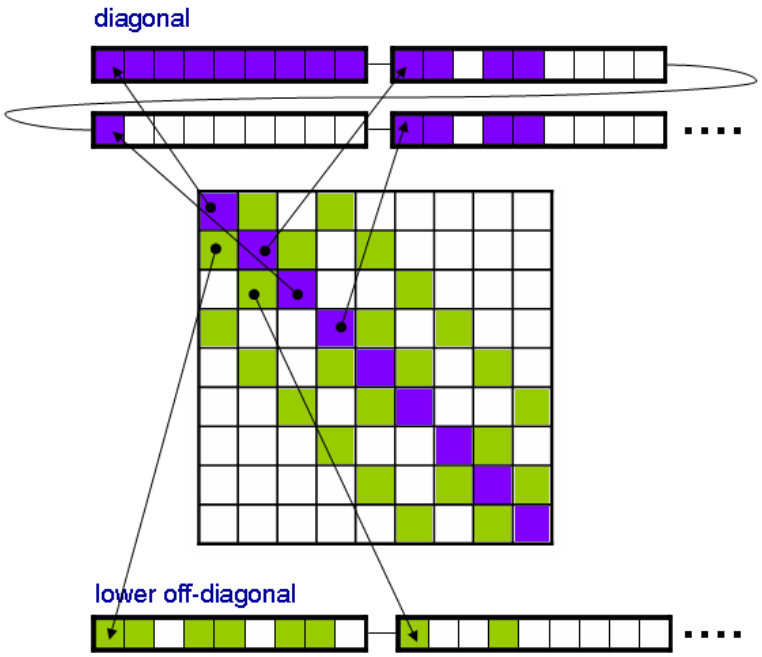


Figure 3.8: Block compressed row pointer matrix for AIM

pointwise	$a - b$	$1/a_{ii}$	$a_{ii} \times b_{ij}$
blockwise	$A - B$	$inv(A_{ii})$	$A_{ii} \cdot B_{ij}$

Table 3.3: Corresponding pointwise and blockwise matrices operations

Combined with block solvers, the block compressed row sparse matrix format has better numerical stability than the pointwise matrix format. Many algorithms, e.g., the ILU family, require the matrix to have non-zero (or not extremely small) diagonal element values to ensure the existence of the reciprocal of a diagonal. Extremely small, or zero, diagonal elements are not common, but they do occur in reservoir simulation. It is one of the major causes of solver related problems. This block matrix structure only requires that the diagonal blocks (see Table 3.3) be invertible. This helps to improve the stability of solver related computations. The details of the ILU algorithm based on the block compressed row sparse matrix format is discussed in Chapter 6.

The block compressed row matrix format is computationally efficient. The positions of the pointers in the pointer matrix are decided by the cell-based and connection-based numbering. These can be assumed static for most simulations. Therefore, the pointer matrix is usually constructed only once and remains unchanged in the course of a simulation. The new matrix format reuses the basic data structure; the 'pointers' point to different locations in these data arrays (Figure 3.8). This feature helps reduce memory cost, since the pointer matrix and the pointers reference locations are not changed. Updating the basic data arrays is equivalent to updating the block compressed row sparse matrix. This is a great advantage compared with a global pointwise representation as in the original CRS format.

Our block compressed row matrix format is fully compatible with AIM (Adaptive Implicit Method). An additional array stores the implicit level of each cell. Each

cell may be assigned an implicit level between one (for the pressure) and the number of primary variables. For example, the matrix in Figure 3.7 corresponds to a 3×3 two-dimensional problem with implicit levels of $\{3, 2, 1, 2, 1, 2, 1, 2, 3\}$.

3.4 Concluding Remarks

In this chapter, we introduced the basic data structures of GPRS. Two new matrix data structures were developed to represent the Jacobian matrix for complex reservoir-facilities simulation. The design of the data structures reflects the separation of the reservoir and facility models in the new framework. In the next few chapters, we describe how these data structures greatly facilitate the extension of GPRS to new modeling capabilities. The corresponding high-performance solvers and preconditioners based on these data structures are discussed in Chapter 7.

Chapter 4

Well Modeling

In Chapter 2, we described the new GPRS framework, in which the reservoir and facilities are treated as separate components. This was motivated by the growing complexity of well models and production facilities. We define facilities as the aggregation of all man-made objects in an oilfield. The most common facilities are wells and surface pipeline networks. In recent years, many new facilities, such as advanced wells, downhole separators, and complex offshore production systems have been developed and deployed in the field. Therefore, integrated modeling of reservoirs with these complicated facilities is necessary for reservoir management. In Chapters 4, 5 and 6, we discuss the modeling of different types of facilities.

In this chapter, we describe a new computational framework for numerical simulation of coupled reservoirs and standard well (StdWell) models, and we describe the detailed treatment of the multisegment well (MSWell) model in the new GPRS.

4.1 Standard Well Model

In reservoir simulation, the standard model treats a well as a source, or sink, term that is added to the gridblocks penetrated by the well [2, 35] [33]. At best, the well is treated as a boundary condition in a manner that ignores the details of fluid flow in the wellbore itself. The standard well (StdWell) model has been widely used in reservoir simulation. The standard model had been extended to handle a single lateral well, which can be vertical, tilted, or horizontal. In the StdWell model, density variations along the wellbore length are accounted for approximately. Some extended standard well models can deal with friction using simplified treatments of frictional losses [20]. We are interested in efficient and accurate representation of the flow in complex wells; however, detailed understanding of the standard well model is a necessary prerequisite for such models.

4.1.1 Variables and Equations

Consider a standard well model with n perforations. The pressure values at the sandface, p_i^w, are the unknowns. We can write n equations for the well, where the first $n-1$ equations describe the pressure relation between two neighboring perforations. Specifically, the expression can be written as [8]:

$$p_{i+1}^w - p_i^w = \frac{1}{2}(\rho_i^w + \rho_{i+1}^w)\, g\, \Delta h_{i,i+1} \quad (i = 1, 2,\ s, n-1), \tag{4.1}$$

where ρ_i^w stands for the density of the fluid mixture in the wellbore around perforation i, g denotes the gravity constant, and $\Delta h_{i,i+1}$ is the height difference between perforations i and $i+1$. Generally, it is difficult to determine the density of the fluid mixture in the wellbore. One of the most frequently used approximations is a volume

weighted phase density [33]:

$$\rho^w = \frac{\sum_{p=1}^{n_p}(\rho_p \, q_p^w)}{\sum_{p=1}^{n_p} q_p^w}, \tag{4.2}$$

where ρ_p is the phase density in the perforated reservoir cell, and q_p^{res} is the phase volumetric rate through a perforation, which is calculated by [35]:

$$q_p^w = WI \, \lambda_p \, (p_p - p^w). \tag{4.3}$$

In Equation 4.3, WI is the well index describing the transmissibility between the wellbore and the perforated cell, λ_p is the phase mobility (the ratio of the phase relative permeability and the viscosity), p_p is the phase pressure in the perforated gridblock.

Equation 4.2 assumes that the fluid property between two perforations in the wellbore of a production well is only affected by the fluid through the nearest upstream perforation. The assumption is that pressure differences between the wellbore and the perforated cell do not cause significant density changes. Equation 4.2 gives a rough estimate for the fluid density. The equation can be problematic when the throughput in the wellbore is very high, and the pressure differences between the wellbore and perforated cells are large.

In addition to the $n - 1$ pressure relation equations (Equation 4.1), there is one more equation, which is the well constraint equation. This constraint equation reflects the physical control strategy on a well. The most common constraint equations are phase rate control and wellbore pressure control. Here, we only describe the treatment of the oil-rate control case. The other rate constraints are very similar. The constraint

equation for oil-rate control using the standard well model can be written as:

$$R_{ctrl} = \sum_{i=1}^{n_{perf}} \left(\sum_{p=1}^{n_p} \rho_p \, \lambda_p \, x_{\bar{o},p} \, WI \, (p_p - p^w) \right)_i - \rho_{\bar{o}} \, q_{\bar{o}} = 0, \tag{4.4}$$

where n_{perf} is the number of perforations, n_p is the number of phases, ρ_p is the phase density, $x_{\bar{o},p}$ is the mass fraction of the oil component in phase p. Note that all these quantities are for a particular perforation i. $\rho_{\bar{o}}$ is oil density at standard conditions, and $q_{\bar{o}}$ is the specified oil production rate. The transmissibility term of each perforation in Equation 4.4 (i.e., the term multiplying the pressure difference) also appears in the reservoir equation of the perforated cells.

For a well with BHP (Bottom Hole Pressure) control, the constraint equation can be written as follows:

$$R_{ctrl} = p_{_{ref}} - p_{_{target}} = 0, \tag{4.5}$$

where $p_{_{ref}}$ is a reference pressure, which is typically the wellbore pressure at the first perforation, p_1^w, and $p_{_{target}}$ is the user-specified operating pressure.

4.1.2 Density Treatment

The density, ρ_i^w, in Equation 4.1 depends on the wellbore pressure and the phase fractions. However, in the standard well model, the phase density is lagged by one iteration when constructing the Jacobian matrix terms. In other words, the standard well model is not strictly fully implicit. Specifically, for each iteration, the derivatives of the wellbore density with respect to other variables are zero. This one-iteration lag treatment helps reduce the complexity of well modeling greatly. At convergence, one obtains the same solution as the fully implicit method. The drawback of this

treatment is that for different problems, it may take a large number of iterations to converge, and sometimes, the iteration scheme does not converge at all.

If density variations are not very important, then density can be treated explicitly and the pressure relation (Equation 4.1) is linear. That is:

$$p^w_{i+1} - p^w_i = C_i \quad (i = 1, 2, \cdots, n-1). \tag{4.6}$$

In this case, the pressure variables of all perforations in the reservoir conservation equations and the well constraint equation can be expressed in terms of a reference pressure. The oil-rate constraint equation for a standard well model can be written as follows:

$$R_{ctrl} = \sum_{i=1}^{n_{perf}} WI_i \left(\sum_{p=1}^{n_p} \rho_p \, \lambda_p \, x_{\bar{o},p} \left(p_p - p^{well}_{ref} - \Delta p^w \right) \right)_i - \rho_{\bar{o}} \, q_{\bar{o}} = 0, \tag{4.7}$$

where Δp^w_i is the pressure difference between perforation i and the reference point.

The standard well model is the simplest approach. A lot of assumptions and approximations are used to simplify the description of the physics in the wellbore. From the above discussion, we should be aware of the following shortcomings of the standard well model:

- The standard well model only considers the hydrostatic pressure change in the wellbore.
- The treatment of fluid density in the wellbore is approximate. At best, it is treated by a one-iteration lag.
- No transient effects in the wellbore are accounted for, and phase holdup is ignored.

In some circumstances, these assumptions and approximations may not be appropriate. Therefore, more accurate well models are needed to capture the fluid flow details in the wellbore.

4.2 Multisegment Well (MSWell) Model

In the last decade, long deviated and horizontal wells have became one of the most broadly used technologies in the petroleum industry. Many horizontal wells (especially offshore ones) may have huge volumetric rates. In such wells, friction and acceleration are not negligible [34]. Even for vertical wells, studies of multiphase flow show large phase holdup effects on the flow behavior [56, 47]. These effects question the viability of the standard well model when multiple fluid phases with strong density and viscosity contrasts are present in the wellbore.

The multisegment well model (MSWell) is commonly used to describe the important flow behaviors in the wellbore [24, 20]. The MSWell model accounts for the pressure drop due to friction and acceleration, in addition to the hydrostatic pressure drop. Friction and acceleration can be dominant factors in long horizontal and deviated wells with high flow rates. The MSWell model also accounts for differences in the phase velocities in the wellbore [56, 47].

4.2.1 Geometry and Variables

In the MSWell model, the wellbore is discretized into a number of segments (Figure 4.1). A segment is a section of the wellbore with two outlets and is represented as a control-volume in wellbore flow simulation. Typical parameters of a segment include its length, inner-diameter, friction coefficient, and inclination. The governing

equations and associated primary variables are defined on these segments. The relation between segments in a discrete wellbore model is similar to the relation between gridblocks in a reservoir model.

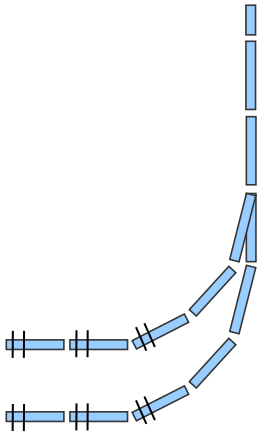


Figure 4.1: Discrete wellbore of the MSWell model

The multisegment well model is a prerequisite for modeling smart wells. In a smart well, there are some special devices, such as valves, sensors, and downhole separators. These objects can be modeled by special segments. The special segments may have different variables and governing equations compared to a regular segment. In this section, we focus on the multisegment well model (without special segments) for the black-oil formulation.

As in the StdWell model, an MSWell model may have many perforations along the wellbore. We assume that a segment can have at most one perforation. The segments are numbered from heel to toe. The toe-end of a perforated segment is always located in the center of the perforated reservoir cell. For a black-oil, isothermal MSWell model, four variables are associated with each segment, which are p^{seg}, α_g, α_w, V_m [20, 47]. The pressure of a segment, p^{seg}, is defined at the toe-end of the segment. Since the

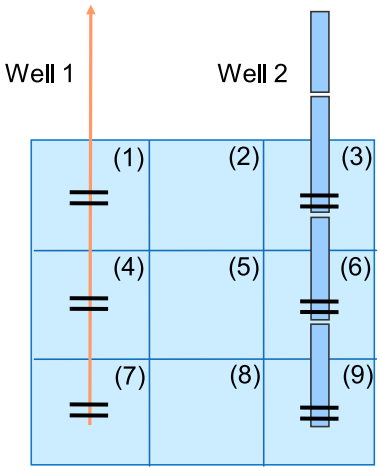


Figure 4.2: Sample MSWell model with a small reservoir model

toe-end of a perforated segment is aligned with the center of a reservoir cell, the flux between the reservoir and the well can be calculated from the pressures of the perforated segment and the reservoir cell directly. The gas and water phase fractions, α_g and α_w, are defined for the entire segment. The mixture velocity of a segment, V_m, is defined at the heel end of a segment. Figure 4.3 shows the locations of the variables for a single-lateral well with four segments.

The MSWell model can be used to simulate multilateral wells. A schematic of a multisegment model for a multilateral well is shown in Figure 4.4. In general, each segment may have one segment connected to its heel end, except for the top segment. Each segment may have one, or more, segments connected to its toe end, except a segment at the end of a lateral. A more general definition of variables and their locations for a general branching system is discussed in Chapter 6.

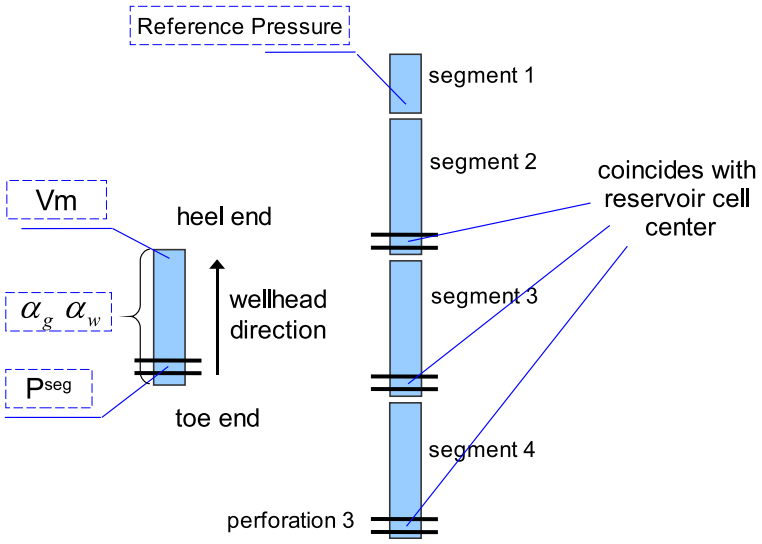


Figure 4.3: Schematic of a single-lateral MSWell model and its variables

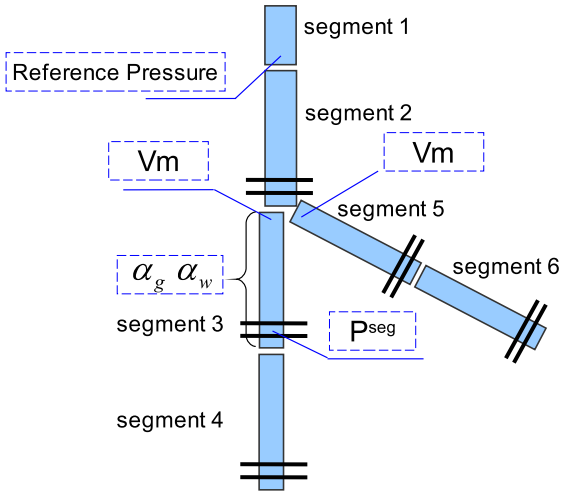


Figure 4.4: Schematic of a two-lateral MSWell model and its variables

4.2.2 Momentum and Mass Balance Equations

In the black-oil model, the governing equations for each segment are the momentum balance equation and three component mass balances. A multisegment model for a single-lateral well with n segments results in $4n$ equations. The momentum balance equation for a segment in the MSWell model can be written as follows [24]

$$R_{p,i}^{seg} = p_i^{seg} - p_{i-1}^{seg} - (\Delta p_{h,i} + \Delta p_{f,i} + \Delta p_{a,i}) = 0 \quad i = 2, 3, \cdots, n, \tag{4.8}$$

where p_i^{seg} is the pressure of segment i, $\Delta p_{h,i}$ is the hydrostatic pressure difference between segments $i-1$ and i, $\Delta p_{f,i}$ is the pressure difference between segments $i-1$ and i caused by friction, and $\Delta p_{a,i}$ is the pressure difference between segments $i-1$ and i due to fluid acceleration. Because the segment pressure is defined at the toe end of a segment, the pressure difference between segments $i-1$ and i is only determined by the properties of segment i. This momentum equation is also called the 'pressure relation' for segment i. Notice that we do not write a momentum balance equation for the top segment.

The hydrostatic pressure term in Equation 4.8 can be written as:

$$\Delta p_{h,i} = \rho_i^{seg}\, g\, \Delta h_i^{seg}, \tag{4.9}$$

where ρ_i^{seg} is the density of the fluid mixture in segment i, and Δh_i^{seg} is the vertical height of segment i. The friction pressure loss term can be written as [20]:

$$\Delta p_{f,i} = \left(\frac{2 f_{tp}\, \rho^{seg}\, V_m^2}{d} \right)_i \Delta x_i, \tag{4.10}$$

where f_{tp} is the friction factor, d is the diameter of the segment, and V_m is the velocity of the fluid mixture in the segment. The pressure loss due to acceleration

can be written as:

$$\Delta p_{a,i} = \left(\frac{2\, m_{in} V_m}{A} \right)_i , \tag{4.11}$$

where m_{in} is the mass flow rate of the mixture through a perforation, and A is the cross-sectional area. Equation 4.8 establishes the relation between the pressure of the segment and the pressure of the heel segment. Note, however, there is no pressure equation for the top segment, since it has no heel segment. The multisegment well may have a constraint equation, which is similar to Equation 4.4, or 4.5. The constraint equation specifies the operating condition of the well. Care must be taken in the treatment of the well constraint since the computational efficiency as well as the ability to model transient behavior in the wellbore depends on it.

In addition to the pressure (momentum balance) equation, each segment has three component mass conservation equations:

$$\begin{aligned} R^{seg}_{c,i} =& \frac{A_i L_i}{\Delta t} \sum_{p=1}^{np} \left[(\rho_p\, x_{c,p}\, \alpha_p)^{n+1}_i - (\rho_p\, x_{c,p}\, \alpha_p)^n_i \right] + \\ & \sum_{p=1}^{np} \left[(A\, \rho_p\, x_{c,p}\, V_{sp})_i - (A\, \rho_p\, x_{c,p}\, V_{sp})_{i+1} \right]^{n+1} - \\ & \sum_{p=1}^{np} \left[\rho_p\, x_{c,p}\, \lambda_p\, WI\, (p^{res} - p^{seg}) \right]^{n+1}_i = 0 \qquad c = g, o, w, \end{aligned} \tag{4.12}$$

where L_i is the length of segment i, α_p is the holdup (in-situ fraction) of phase p in the segment, and V_{sp} is the phase superficial velocity. These equations are similar to the component mass balance equations for flow in porous media, which we write for each cell in the reservoir model. The difference is in the flux term. In wellbore flow, phase velocities have no explicit relation with pressure gradient. The mixture velocity, V_m, becomes one of the primary variables. Therefore, the flux term is written based on

the phase superficial velocities rather than the pressure gradient.

Compared with the StdWell treatment, the MSWell model leads to more rigorous representation of the density of the fluid mixture in a segment. The fluid-mixture density can be written as

$$\rho_m = \alpha_g \rho_g + \alpha_o \rho_o + \alpha_w \rho_w, \tag{4.13}$$

where ρ_m is the density of the fluid mixture. In the equation, the phase holdups are primary variables. The dependence of density on segment pressure, $\partial \rho_m / \partial p$, is also considered when constructing the Jacobian matrix. Therefore, the fluid density in the well is fully coupled to the primary variables. This is much more rigorous compared with the approach used in the standard well model (Equation 4.2). Friction and acceleration are also accounted for in the MSWell model, which are treated in a fully implicit manner. The MSWell model provides information about the flow in the wellbore, including the distribution of pressure, velocity, and phase holdups. Moreover, since we write a momentum and mass-balance equations for each segment, transient effects in the wellbore can be represented.

These advantages come at a computational cost. Each segment has four equations and variables. A multisegment well may include hundreds of segments; this results in a large number of equations per well. Efficient solution of reservoir model that are coupled to MSWells is a challenging task. Our specially designed linear solution strategies for systems with MSWells are presented in Chapter 7.

4.2.3 Constraint Equations and Jacobian Matrix Structures

Wells in the field are subject to various control strategies. These controls are represented by constraint equations. Even for the same control strategy (e.g., constant oil-rate control), different discrete representations may be employed. For the MSWell model, the constraint equation takes the place of the 'pressure equation' for the top segment. Next, we discuss several constraint relations and their impact on the Jacobian matrix.

A simple illustration is shown in Figure 4.5, which has a small two-phase reservoir model with 3×2 gridblocks. The well has three segments and is perforated in the second and fifth reservoir gridblocks. For fully implicit oil-water simulation, there are

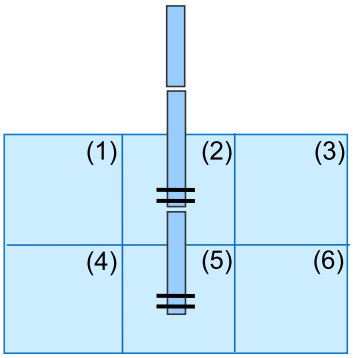


Figure 4.5: Sample system with the MSWell model

two equations per cell (oil and water conservation equations). Therefore, the total number of the reservoir equations of this case is twelve. Since each segment has four equations, the three-segment well results in twelve well equations. The total number of the equations in the system is 24. In the following discussion, sample Jacobian

matrices will be shown for this system. We represent a Newton iteration as follows:

$$\begin{pmatrix} \boldsymbol{J}_{RR} & \boldsymbol{J}_{RW} \\ \boldsymbol{J}_{WR} & \boldsymbol{J}_{WW} \end{pmatrix} \cdot \begin{pmatrix} \delta \boldsymbol{u}_r \\ \delta \boldsymbol{u}_w \end{pmatrix} = - \begin{pmatrix} \boldsymbol{R}_r \\ \boldsymbol{R}_w \end{pmatrix}, \tag{4.14}$$

where $\boldsymbol{J}_{RW}$ is the sector of the Jacobian corresponding to the derivatives of the reservoir equations with respect to the MSWell variables. $\boldsymbol{J}_{RR}$, $\boldsymbol{J}_{WR}$, and $\boldsymbol{J}_{WW}$ are defined in a similar manner. $\boldsymbol{R}_r$ and $\boldsymbol{R}_w$ are the residual vectors of the reservoir and well equations respectively. $\delta \boldsymbol{u}_r$ and $\delta \boldsymbol{u}_w$ are the corrections for the reservoir and well variables from the Newton iteration.

According to the multilevel sparse block data structure, the $\boldsymbol{J}_{RW}$, $\boldsymbol{J}_{WR}$, and $\boldsymbol{J}_{WW}$ should be within the facilities matrices ($\boldsymbol{J}_{RF}$, $\boldsymbol{J}_{FR}$, and $\boldsymbol{J}_{FF}$). Since the main focus of this chapter is on the MSWell model rather than the data structures, we do not consider the facilities level for simplicity.

Wellbore Pressure Control

Theoretically, one can specify the pressure of any segment to be a certain value. For simplicity, we always choose the toe-end of the top segment as the reference point. If the top segment is close to the first perforation in the well, the control is equivalent to bottom hole pressure (BHP) control. If the top segment is at the surface, the control is equivalent to wellhead pressure (WHP) control. The constraint equation for a well with pressure control can be written as:

$$R_1^{seg} = p_1^{seg} - p_{\,target} = 0. \tag{4.15}$$

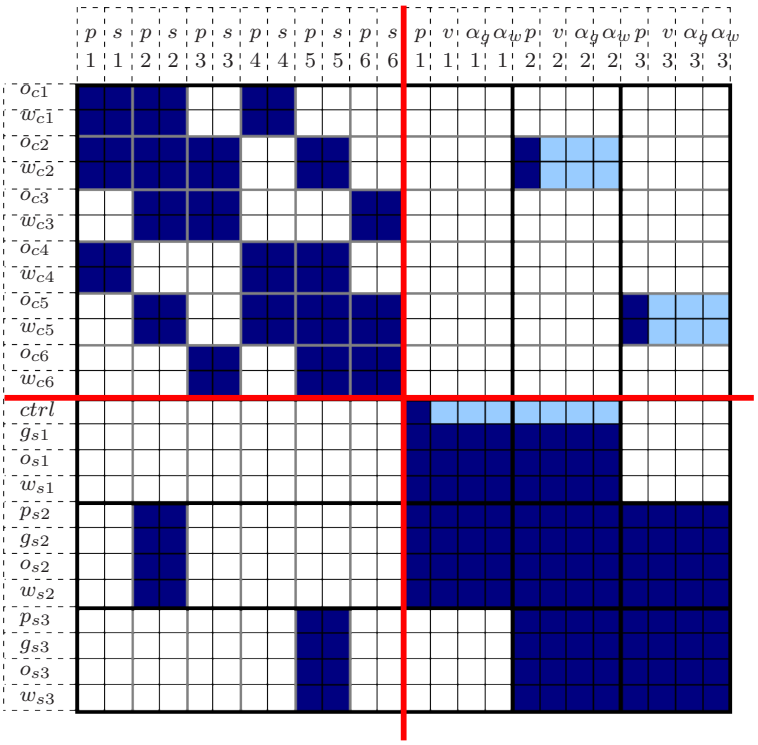


Figure 4.6: Jacobian matrix with pressure controlled well

Equation 4.15 has only one non-zero derivative:

$$\frac{\partial R_1^{seg}}{\partial p_1^{seg}} = 1. \tag{4.16}$$

The structure of the corresponding Jacobian matrix is shown in Figure 4.6. We labeled the equations and variables in the figure. The first twelve rows of the matrix are derived from the reservoir equations, and the remaining twelve rows are from the well equations. The constraint equation is the first well equation, which corresponds to the 13th row of the matrix and is labeled with 'ctrl'. The first twelve columns represent the reservoir variables and the remaining columns represent the well unknowns.

From the figure, the natural block structure of the reservoir equations and unknowns is quite clear; similarly, the multi-segment well equations and unknowns have their own block structure. The reservoir part (top-left 12×12), $\boldsymbol{J}_{RR}$, is made up of

6×6 blocks, each of which is a 2×2 block. Each off-diagonal (2×2) block corresponds to one connection between two reservoir gridblocks. Since the system in Figure 4.5 is two-dimensional with cell based ordering, the reservoir part is a block five-diagonal matrix. The well part $\boldsymbol{J}_{WW}$ (bottom-right 12×12) has 3×3 blocks, each of which is a 4×4 block, because each segment has four equations and four variables. The off-diagonal blocks in $\boldsymbol{J}_{WW}$ represent intersegment connections. A multisegment well with a single lateral leads to a tridiagonal block structure, which is similar to the discretization of a one-dimensional reservoir model. $\boldsymbol{J}_{RW}$ and $\boldsymbol{J}_{WR}$ represent the coupling between the reservoir model and the well model. $\boldsymbol{J}_{RW}$ contains two (4×2) blocks and $\boldsymbol{J}_{WR}$ contains two (2×4) blocks. The number of non-zero blocks in $\boldsymbol{J}_{RW}$ and $\boldsymbol{J}_{WR}$ is the same as the number of perforations.

The dark blue cells in Figure 4.6 represent non-zero elements. In order to store the data in a uniform block size, an entire block is kept even if there is only one non-zero element in it. These stored zero elements are marked with light blue. For a constant-pressure constraint, there is only one non-zero element, in the row corresponding to the constraint equation (the 13th row of the matrix shown in Figure 4.6).

Rate Control

One can specify a target for any phase, or for the total production. All of these constraints are very similar in implementation, so here we use oil-rate control as an example. For a well under oil-rate control, the constraint equation can be written in two different ways. These two forms are based on slightly different control strategies, and they lead to different matrix structures, numerical performance, and computational results.

The first form is very similar to the oil-rate constraint equation in the standard

well model (Equation 4.4). That is, the mass rate of the oil component from all the perforations is equal to the specified rate:

$$R_{ctrl} = \sum_{j=1}^{n_{perf}} \left(\sum_{p=1}^{n_p} \rho_p \, \lambda_p \, x_{\bar{o},p} \, WI \, (p_p^{res} - p^{seg}) \right)_j - \rho_{\bar{o}} \, q_{\bar{o}} = 0. \tag{4.17}$$

. In Equation 4.17, R_{ctrl} depends on both formation and segment pressures at the perforations. It is a function of the sandface fluid saturation as well. However, although the equation shows up as if it is one of the first segment's equations, it does not rely on any variable in the first segment (the first segment is not perforated in our example). The derivatives of Equation 4.17 with respect to the primary variables are:

$$\frac{\partial R_1^{seg}}{\partial p_j^{res}} = \sum_{p=1}^{n_p} \left(\frac{\partial (\rho_p x_{\bar{o},p} \lambda_p)}{\partial p^{res}} \, WI \, (p^{res} - p^{seg}) + \rho_p x_{\bar{o},p} \lambda_p \, WI \right)_j, \tag{4.18}$$

$$\frac{\partial R_1^{seg}}{\partial s_j^{res}} = \sum_{p=1}^{n_p} \left(\rho_p x_{\bar{o},p} \, WI \frac{\partial \lambda_p}{\partial s^{res}} \right)_j, \tag{4.19}$$

$$\frac{\partial R_1^{seg}}{\partial p_j^{seg}} = -\sum_{p=1}^{n_p} \left(\rho_p x_{\bar{o},p} \lambda_p \, WI \right)_j. \tag{4.20}$$

This results in the Jacobian matrix shown in Figure 4.7. The constraint row (row 13) has non-zero elements at the positions corresponding to the perforations (column 3, 4, 9, 10, 17, 21). Compared to the BHP-control case, the matrix has more non-zero elements, and this leads to additional storage. In our example, we have two additional 4×2 blocks, namely, block (13:16, 2:3) and block (13:16, 8:9), in the $\boldsymbol{J}_{WR}$ part and one additional 4×4 block, (13:16, 21:24), in the $\boldsymbol{J}_{WW}$ part.

In the matrix for the BHP case (Figure 4.6), the number of blocks in $\boldsymbol{J}_{WR}$ is the same as the number of perforations, and the number of upper, or lower, off-diagonal blocks in $\boldsymbol{J}_{WW}$ is the same as the number of intersegment connections. However, the

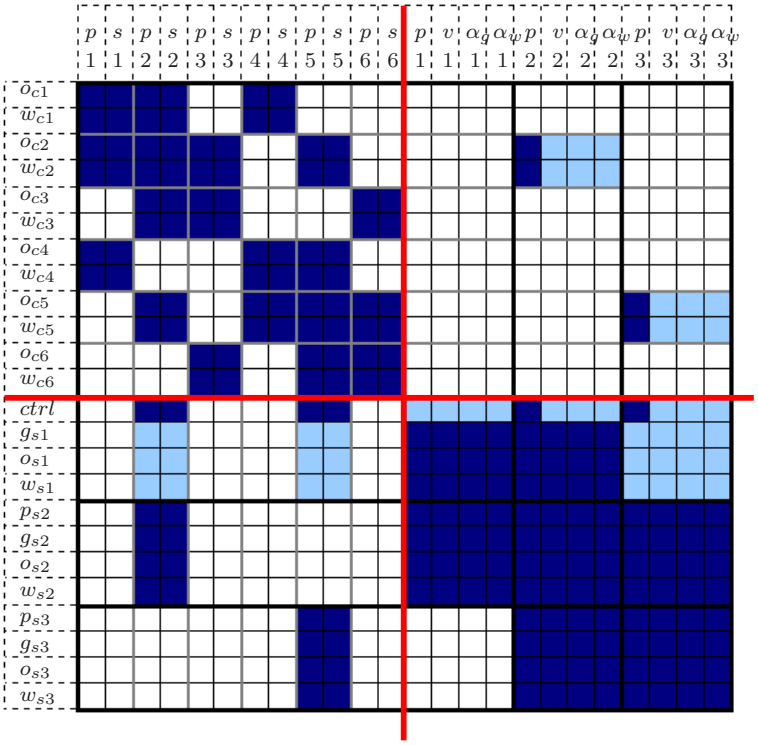


Figure 4.7: Jacobian matrix with oil rate constraint

additional blocks introduced by Equation 4.7 ruin this simple one-to-one relation and leads to many special treatments in implementation with complicated data structure manipulations.

The other form of the oil-rate constraint is to equate the mass-rate of the components out of the top segment (first segment) to the specified rate, which can be written as:

$$R_{ctrl} = A_1 \sum_{p=1}^{n_p} (\rho_p \, x_{\bar{o},p} \, V_{sp})_1 - \rho_{\bar{o}} \, q_{\bar{o}} = 0. \tag{4.21}$$

Equation 4.21 depends only on the variables of the first (top) segment. The derivatives

are as follows:

$$\frac{\partial R_1^{seg}}{\partial p_1^{seg}} = A_1 \sum_{p=1}^{n_p} \frac{(\partial \rho_p \, x_{\bar{o},p} \, V_{sp})_1}{\partial p_1^{seg}} , \tag{4.22}$$

$$\frac{\partial R_1^{seg}}{\partial V_1^{seg}} = A_1 \sum_{p=1}^{n_p} \rho_p \, x_{\bar{o},p} \, \frac{\partial V_{sp,1}}{\partial V_1^{seg}} , \tag{4.23}$$

$$\frac{\partial R_1^{seg}}{\partial \alpha_{g,1}^{seg}} = A_1 \sum_{p=1}^{n_p} \rho_p \, x_{\bar{o},p} \, \frac{\partial V_{sp,1}}{\partial \alpha_{g,1}^{seg}} , \tag{4.24}$$

$$\frac{\partial R_1^{seg}}{\partial \alpha_{w,1}^{seg}} = A_1 \sum_{p=1}^{n_p} \rho_p \, x_{\bar{o},p} \, \frac{\partial V_{sp,1}}{\partial \alpha_{w,1}^{seg}} . \tag{4.25}$$

Further expansion of these equations depends on the specific flow model employed. Here, we only focus on the structure of the Jacobian matrix, which is shown in Figure 4.8. The matrix structure is very similar to the one with BHP control (Figure 4.6). The number of stored data blocks is exactly the same.

Link between the Two Rate Constraint Equations

Recall the segment component mass balance equation, which is given by Equation 4.12; we rewrite the equation for the oil component as:

$$\begin{aligned} R_{\bar{o},i}^{seg} = {} & \frac{A_i \Delta x_i}{\Delta t} \sum_{i=1}^{np} \left[(\rho_p x_{\bar{o},p} \alpha_p)_i^{n+1} - (\rho_p x_{\bar{o},p} \alpha_p)_i^n \right] + \\ & \sum_{i=1}^{np} \left[(A \rho_p x_{\bar{o},p} V_{sp})_i^{n+1} - (A \rho_p x_{\bar{o},p} V_{sp})_{i+1}^{n+1} \right] - \\ & \sum_{i=1}^{np} \rho_{p,i} x_{\bar{o},p,i} \lambda_{p,i} W I_i (p^{res} - p^{seg})_i = 0. \end{aligned} \tag{4.26}$$

This segment conservation equation is composed of three terms, which are accumulation, flux between segments, and flux between the reservoir and a segment. If we

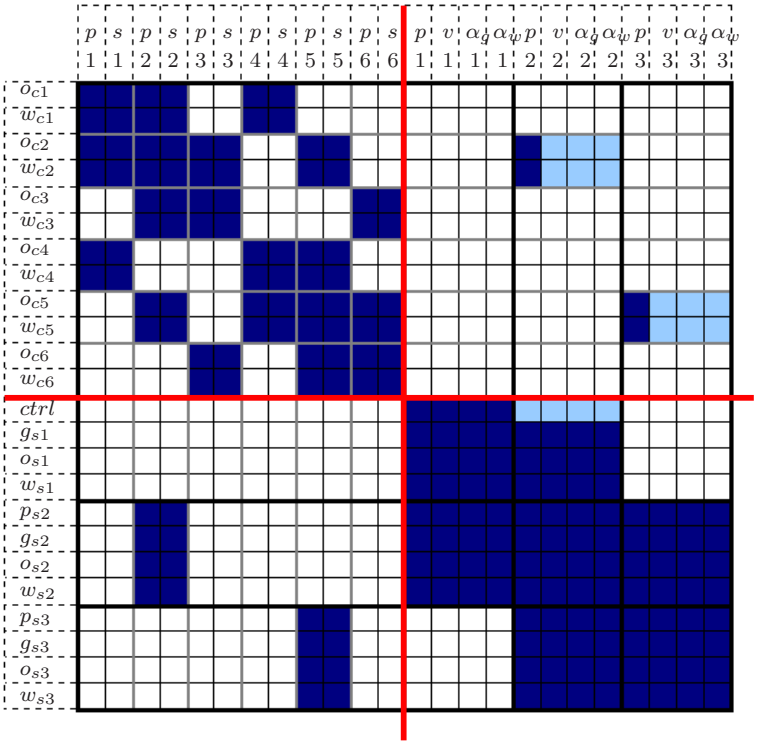


Figure 4.8: Jacobian matrix with oil rate constraint

sum the equations of all the segments, the flux terms between segments cancel out, and we get:

$$R_{\bar{o}} = \sum_{i=1}^{n_{seg}} \left(\frac{A\Delta x}{\Delta t} \sum_{i=1}^{np} \left[(\rho_p x_{\bar{o},p} \alpha_p)^{n+1} - (\rho_p x_{\bar{o},p} \alpha_p)^n \right] \right)_i + \sum_{p=1}^{np} (A \rho_p x_{\bar{o},p} V_{sp})_1^{n+1} - \sum_{i=1}^{n_{seg}} \left[\sum_{p=1}^{np} \rho_p x_{\bar{o},p} \lambda_p W I (p^{res} - p^{seg}) \right]_i = 0. \tag{4.27}$$

Only the outflow (for a production well) from the top segment remains. This term also shows up in the second rate constraint equation (Equation 4.21). Substitution

of Equation 4.21 in Equation 4.27 leads to:

$$R_{\bar{o}} = \sum_{i=1}^{n_{seg}} \left(\frac{A\Delta x}{\Delta t} \sum_{i=1}^{np} \left[(\rho_p x_{\bar{o},p} \alpha_p)^{n+1} - (\rho_p x_{\bar{o},p} \alpha_p)^n \right] \right)_i + \rho_{\bar{o}}\, q_{\bar{o}} - \sum_{i=1}^{n_{seg}} \left[\sum_{p=1}^{np} \rho_p x_{\bar{o},p} \lambda_p WI(p^{res} - p^{seg}) \right]_i = 0. \tag{4.28}$$

Equation 4.28 is the oil conservation equation for the entire well. If we ignore transient effects, i.e., accumulation in each segment is ignored, Equation 4.28 can be simplified to:

$$R_{\bar{o}} = \rho_{\bar{o}}\, q_{\bar{o}} - \sum_{i=1}^{n_{seg}} \left[\sum_{p=1}^{np} \rho_p x_{\bar{o},p} \lambda_p WI(p^{res} - p^{seg}) \right]_i = 0. \tag{4.29}$$

Since an unperforated segment has no inflow from the reservoir, we can use the n_{perf} to replace n_{seg} in Equation 4.29, and this leads to:

$$R_{\bar{o}} = \rho_{\bar{o}}\, q_{\bar{o}} - \sum_{j=1}^{n_{perf}} \left[\sum_{p=1}^{np} \rho_p x_{\bar{o},p} \lambda_p WI(p^{res} - p^{seg}) \right]_i = 0, \tag{4.30}$$

which is very similar to the traditional (standard model) oil-rate constraint equation (Equation 4.17).

Both Equation 4.17 and Equation 4.21 define a constant oil-rate control on the MSWell, but they have different forms. The two equations also lead to differences in the following aspects:

- Equation 4.17 may lead to a diagonal block (the 4 by 4 block in Figure 4.7) that is not invertible for the first segment. The first row in the diagonal block is all zeroes. This imposes serious difficulty to apply a block ILU preconditioner.
- Equation 4.17 does not consider transient effects. This may cause some errors

when the well status changes. Equation 4.21 accounts for transient effects, and naturally handles wellbore storage, which is important in simulating transient well tests.

- Equation 4.21 leads to a Jacobian matrix structure similar to that obtained for the pressure constraint. On the other hand, Equation 4.4 introduces additional blocks in both $\boldsymbol{J}_{WW}$ and $\boldsymbol{J}_{WR}$. This destroys the one-to-one relation between the number of intersegment connections and the number of off-diagonal blocks in $\boldsymbol{J}_{WW}$, as well as the one-to-one relation between the number of perforations and the number of blocks in $\boldsymbol{J}_{WR}$.

Based on this discussion, we can see that Equation 4.21 is a much better choice compared to Equation 4.17. We used Equation 4.21 as the rate constraint equation in GPRS. We also showed that Equation 4.21 and the component mass equations can be combined to form an equation similar to the standard well equation. This idea will be used in the preconditioning strategies used to solve linear systems with MSWells, which is investigated in Chapter 7.

4.2.4 MSWell Initialization

Before simulation begins, the system is assigned an initial state. However, significant changes in the variables usually take place during the first timestep. In order to ensure convergence of Newton's method, a small timestep is strongly recommended to begin the simulation.

A standard well with a rate constraint has only one variable, which is the reference pressure. The pressure may change significantly even if the pore volume of the perforated gridblock is reasonably large and the production rate is not too high.

Typically, the initial wellbore reference pressure (producer) is set slightly below the reservoir pressure in the first perforated gridblock to start the simulation. With a small initial timestep, the produced fluid volume is limited. Therefore, the pressure changes in the gridblocks and the wellbore are also limited. The Newton method usually can handle this setting reliably.

However, a multisegment well has more variables. As mentioned before, multisegment well may have more than one hundred segments. Each segment will have four variables for the black-oil case and many more for compositional models. The typical variables are pressure, mixture velocity, holdup of gas and water phases. Among these variables, velocity is the most problematic one. Usually, the fluids in the wellbore are stagnant before production starts. Once production begins, the velocity of the fluid can reach large values quickly. For example, a 500 bbl/day flow rate in a wellbore with a 0.32 feet radius yields a fluid velocity of about 8726 ft/day. Moreover, velocity is the dominant nonlinear term in the friction pressure loss relation (Equation 4.10). It is difficult for the Newton-Raphson method to converge when large changes in the velocity take place during the iteration process. Another interesting thing is that the big jump in velocity does not depend on time, which means that taking very small time steps does not cure the problem.

From the above discussion, we see that it is crucial to have a good estimate to the initial state of a multisegment well. This can be accomplished by the following steps:

1. Estimate holdups: Initial phase saturations in the perforated gridblocks are known. Using the relative permeability data, the phase inflow ratios (holdups) of all perforations is determined. Since we have no estimate of the pressure yet, we cannot compute estimates of the flow rates through perforations. So we assume a uniform flow rate distribution for all perforations. That is, all

segments have the same phase holdups, which is the average of phase holdups through all perforations.

2. Estimate density: Since we already have estimates for the holdup, we can compute the density distribution in the segments.

3. Estimate wellbore pressure: First, we need to estimate the pressure relations (momentum balance) of all segments. We consider gravity and friction effects. The gravity relation can be easily computed, since we already have the mixture density and the height of all segments. Friction is calculated using the current estimate of velocity; at simulation startup, the velocity is zero, which means we have no pressure drop due to friction yet. If the well is under pressure control, the reference pressure is known. Once we have the estimate for the pressure relations of all segments, the segment pressures can be computed. If the well is under rate control, given the known pressure relations between segments, the reference pressure can be calculated from Equation 4.7.

4. Estimate inflows from the reservoir: Once the wellbore pressure is estimated, Equation 4.3 is used to calculate the inflow through each perforation.

5. Estimate the velocity of the fluid mixture: The mixture velocity of each segment can be calculated from the rate of inflow from the reservoir and inflow from the toe segments

6. Loop (optional): Go back to step 3, use the mixture velocity to update the pressure loss.

4.2.5 Homogeneous and Drift-flux Models

Multiphase flow in pipes is very complicated. It is very difficult to solve the governing Navier-Stokes equations for multiphase flow. The computational cost is too high, and the results are not necessarily accurate. In petroleum engineering, three kinds of flow models are used to describe flow in wellbores and pipeline networks. These are (1) empirical correlations, (2) homogeneous models, and (3) mechanistic models [36]. The homogeneous model and its extension, the drift-flux model [56], are implemented in the new GPRS. Since the wellbore is discretized into segments, the main purpose of these flow models is to determine the flux term between segments in the mass conservation equations. Here we write Equation 4.12:

$$\begin{aligned} R_{c,i}^{seg} =& \frac{A_i L_i}{\Delta t} \sum_{p=1}^{np} \left((\rho_p \, x_{c,p} \, \alpha_p)_i^{n+1} - (\rho_p \, x_{c,p} \, \alpha_p)_i^n \right) + \\ & \sum_{p=1}^{np} \left((A \, \rho_p \, x_{c,p} \, V_{sp})_i^{n+1} - (A \, \rho_p \, x_{c,p} \, V_{sp})_{i+1}^{n+1} \right) - \\ & \sum_{p=1}^{np} \rho_{p,i} \, x_{c,p,i} \, \lambda_{p,i} \, WI_i \, (p^{res} - p^{seg})_i = 0 \qquad c = g, o, w. \end{aligned} \tag{4.31}$$

In the flux terms, the phase superficial velocity, V_{sp}, is the key variable. In the following discussion, we will see how the homogeneous and drift-flux models link V_{sp} to the primary variable V_m.

Homogeneous Model

The basic homogeneous model assumes the fluid mixture flows as if it was a single-phase fluid [20]. In the homogenous wellbore model, all phases share the same in-situ

flow velocity. The slip between phases is ignored. The velocity relation is

$$V_p = V_m, \tag{4.32}$$

where V_p is the in-situ phase velocity in the segment. By definition, V_p is related to the superficial phase velocity by:

$$V_{sp} = V_p \, \alpha_p, \tag{4.33}$$

where α_p is the phase holdup (volume fraction) in a segment. Thus, in the homogenous model, the superficial phase velocity has a simple relation with the mixture velocity:

$$V_{sp} = V_m \, \alpha_p. \tag{4.34}$$

The homogenous model is the simplest discrete wellbore flow model. The equation is differentiable, so it is suitable for the Newton-Raphson nonlinear solver. However, the homogeneous model does not consider slip between phases. The model may be quite off from the true physics in the wellbore. For example, the homogenous model cannot be used to model gravity segregation of different phases [47].

Drift-flux Model

The drift-flux model is an extension of the homogeneous model. The model was first published by Zuber and Findlay [56]. Unlike the basic homogeneous model, the drift-flux model considers the relative velocity between different fluid phases. The different phases may have different velocities, and in some cases, even different flow directions. Similar to the homogenous model, the drift-flux model is continuous and differentiable. Therefore, it is a suitable option to implement in reservoir simulators [47, 20].

The general form of the drift-flux model can be written as [56]:

$$V_g = C_0 V_m + V_d, \tag{4.35}$$

where V_g is the gas-phase velocity, C_0 is the profile parameter, which reflects the effect of velocity and concentration profiles, V_d is the drift velocity, which reflects the buoyancy effect of the gas phase (Figure 4.9) [11]. The superficial gas-phase velocity can be written as:

$$V_{sg} = \alpha_g \, V_g, \tag{4.36}$$

and therefore

$$V_{sg} = \alpha_g \, C_0 V_m + \alpha_g \, V_d, \tag{4.37}$$

C_0 and V_d are the key parameters in the drift-flux model. In the discrete representation of the drift-flux model, these two parameters and the phase holdup determine the relation between the phase superficial velocity and the fluid mixture velocity, which are taken to be at the interface between two neighboring segments. The values of these two parameters are calculated from the phase properties of the upstream and downstream segments [47].

The homogeneous and drift-flux options have been implemented as part of the MSWell model in GPRS. In the following discussion, a few cases with different flow models will be tested.

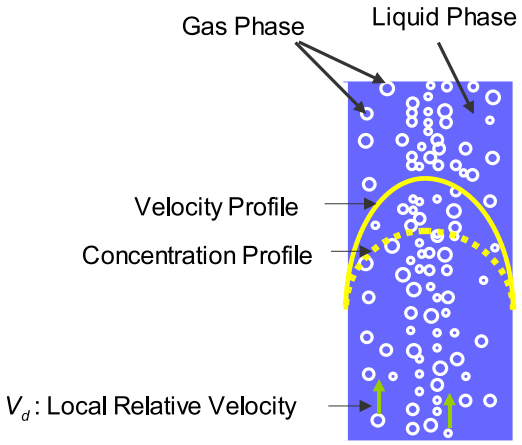


Figure 4.9: Schematic of velocity and concentration profiles [11]

4.2.6 Test Cases

Case 1: 1800-segment Well with the Drift-flux Model

The multisegment well model in GPRS is robust and computationally efficient. The fully coupled reservoir-facilities framework enables users to decide on the level of complexity for both the reservoir and well models. GPRS can be turned into a wellbore simulator, by using a simple reservoir model as a boundary condition and a complicated wellbore model. In order to test the numerical performance with large numbers of segments and complex wellbore geometry, a test case was set up.

In this test case, a live-oil reservoir has a uniform initial pressure of 6400 psi at a depth of 6000 ft. The bubble-point of the reservoir is 5014 psi. A $5 \times 5 \times 2$ grid is used to represent the reservoir model. The reservoir model is not the focus of this case. The reservoir model mainly acts as a boundary condition. One bilateral well is perforated in the top layer of the reservoir. Each lateral intersects five cells in a row. The well contains three sections according to its shape: vertical, tilted, and

horizontal. A diagram for the system is shown in Figure 4.10. The 5000 ft long vertical zone is uniformly discretized into 1000 segments. At the end of the vertical zone, the wellbore is split into two tilted branches. Each tilted branch is discretized into 200 segments (7.07 ft/segment). The tilted branches have a slope of 45 degrees, which means they deepen the well by 1000 ft. Each horizontal zone is 1000 ft long and uniformly discretized into 200 segments. Overall, the entire wellbore is discretized into 1800 segments, and each segment is about $5 - 7$ ft long. This is a very fine grid for a wellbore. The well produces 1000 bbl/day under oil rate control. A drift-flux model is used to simulate multiphase flow in the wellbore. We consider pressure losses caused by gravity, friction and acceleration.

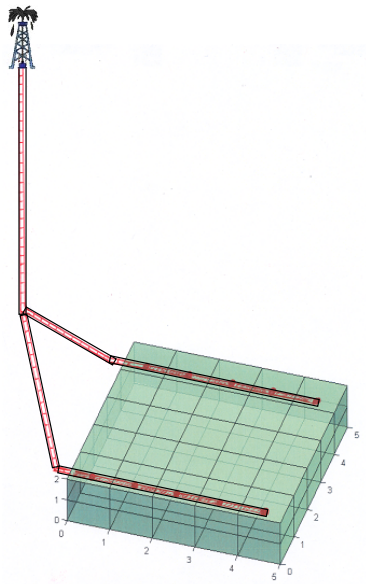


Figure 4.10: Layout of a bilateral MSWell and a small reservoir

With very small initial timesteps (on the order of a minute), we capture the wellbore storage effect, in which the downhole rate is not equal to the wellhead rate. GPRS is able to output the solution details of each segment. Profiles of segment pressures and the gas-phase holdup along the wellbore are shown in Figure 4.11. The

Y axes in the two subplots represent wellbore length (not depth) starting from the wellhead. The curves for 1, 100 and 200 days are shown in the figure. The reservoir bubble-point pressure (5014 psi) is marked with an orange vertical dash line in the pressure plot.

The bubble-point line intersects the pressure curve of the first day at about 800 ft. The segment pressure above this point is below the bubble point pressure. The gas holdup curve also shows that gas comes out from the oil phase at the same point in the gas-phase holdup plot. The bubble-point line intersects the pressure curve of the 100^{th} day at about 3600 ft. We also observe gas coming out of solution at the same point. The wellbore pressure keeps decreasing as we deplete the reservoir. Hence the two-phase zone extends downward gradually, until the entire wellbore becomes two-phase zone, which is observed at 200 days.

The gas holdup curve at 200 days is worth a detailed look. The well geometry changes abruptly from vertical to tilted to horizontal. Since the tilted angle has an important impact on the flow regime in the wellbore, abrupt transitions can lead to discontinuities in the gas-phase holdup profile. The gas holdup in the tilted zone is smaller than the holdups in the other two zones. This is because the gas phase has much larger slip effect over the oil phase when the wellbore is tilted at 45 degrees [23, 47]. In other words, the gas phase in the tilted zone has relatively larger speeds than those in the other two zones. Therefore, smaller gas holdup in the titled zone is needed to obtain the gas-phase rate.

On the gas holdup curve at 200 days, we see a small kink at a depth of about 2000 ft. The slope changes at the segment with the pressure of 4014 psi. This value is a data point in the gas PVT table. In GPRS, linear interpolation is used to calculate phase-formation-volume factors from the PVT table. Since we only have a few data points for the gas phase in the PVT table, the gas-phase compressibility obtained

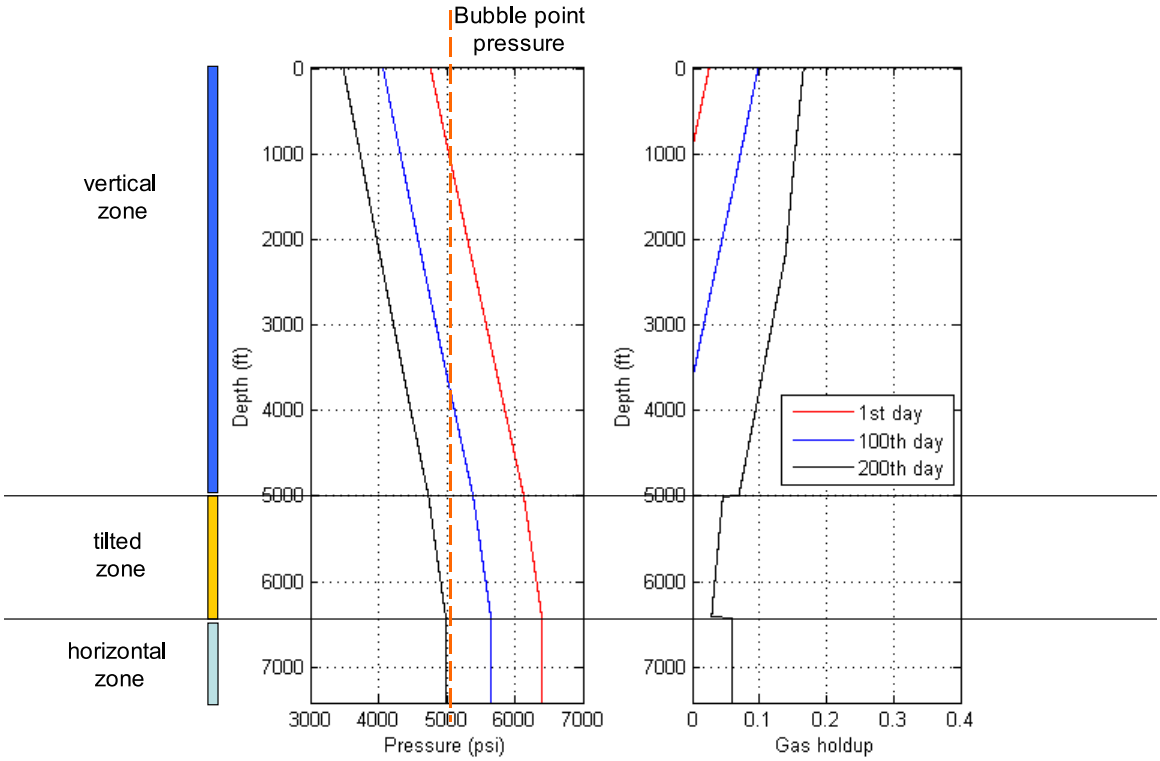


Figure 4.11: Pressure and gas-phase holdup profiles along the wellbore using a drift-flux model with 1800 segments (Case 1)

from the table is not continuous, which causes the kink. This reminds us that the gas-phase holdup profile is very sensitive to the PVT data. Thus, the PVT properties should be fine enough to capture the dynamic behaviors of interest.

In the plot, the gas holdup of the first segment is much higher than that of any other segment. For example, at 200 days, the top segment has a gas holdup of 0.23, while the gas holdup of the second segment is about 0.16. The discontinuity comes from switching the segment types. Even for a wellbore using the drift-flux model, the first segment is still a homogeneous segment. This is because the phase velocity in a drift-flux segment relies on properties of the current segment and its heel segment. The top segment has no heel segment. Therefore, the top segment has to be treated as a homogeneous segment. In practice, the top segment is very small in size. So the usage of the homogeneous model does not affect the results of the drift-flux model. In a homogeneous segment, the gas phase has the same speed as the oil phase. On the other hand, in the drift-flux segment, the gas-phase has a higher speed than the oil-phase. When a two-phase fluid mixture flows from a drift-flux segment into a homogenous segment, a higher value of the gas-phase holdup is needed to balance the reduction in gas-phase speed.

The pressure profiles in the horizontal zone are almost vertical. In fact, there is a pressure gradient in this part due to friction. However, the throughput (500 bbl oil/day) is small and the wellbore roughness coefficient is normal. As a result, the pressure loss is too small to be observed in the plot.

This case is also designed to test the solver capability of GPRS. We deliberately discretized the wellbore into a large number of segments, which may be enough for most wellbore simulations. The case is run on a personal computer with one 2.2 Ghz CPU. The timesteps are one day, except a few smaller ones in the beginning. GPRS can finish 200 days of simulation in 116 seconds. More performance information is

listed in Table 4.1. We can see that GPRS can be used as a highly efficient multiphase flow wellbore simulator.

Timestep	208
Newton Iteration	1009
Solver Iteration	5967
Solver Time (sec)	45.0
Total Time (sec)	116.0

Table 4.1: GPRS timing performance for Case 1

Case 2: 180-segment Well with the Drift-flux Model

The physical setting of the second case is the same as Case 1. However, the number of segments is reduced to one-tenth of the first case. The entire wellbore is discretized into 180 segments. The pressure and gas holdup profiles along the wellbore are plotted in Figure 4.12.

The plot shows similar results to those of the 1800-segment well in Case 1. We can see that the pressure and holdup properties change relatively smoothly along the wellbore. In general, just as in reservoir modeling, grid refinement studies should be performed to analyze the accuracy of the computed solutions.

Case 3: 180-segment Well with the Homogeneous Model

The third case is exactly the same as Case 2, except all the segments use the homogeneous model. The pressure and gas holdup profiles along the wellbore are plotted in Figure 4.13.

By comparing Figure 4.12 and Figure 4.13, we find that the pressure profiles are similar in both test cases, but the gas-phase holdup profiles are very different. In

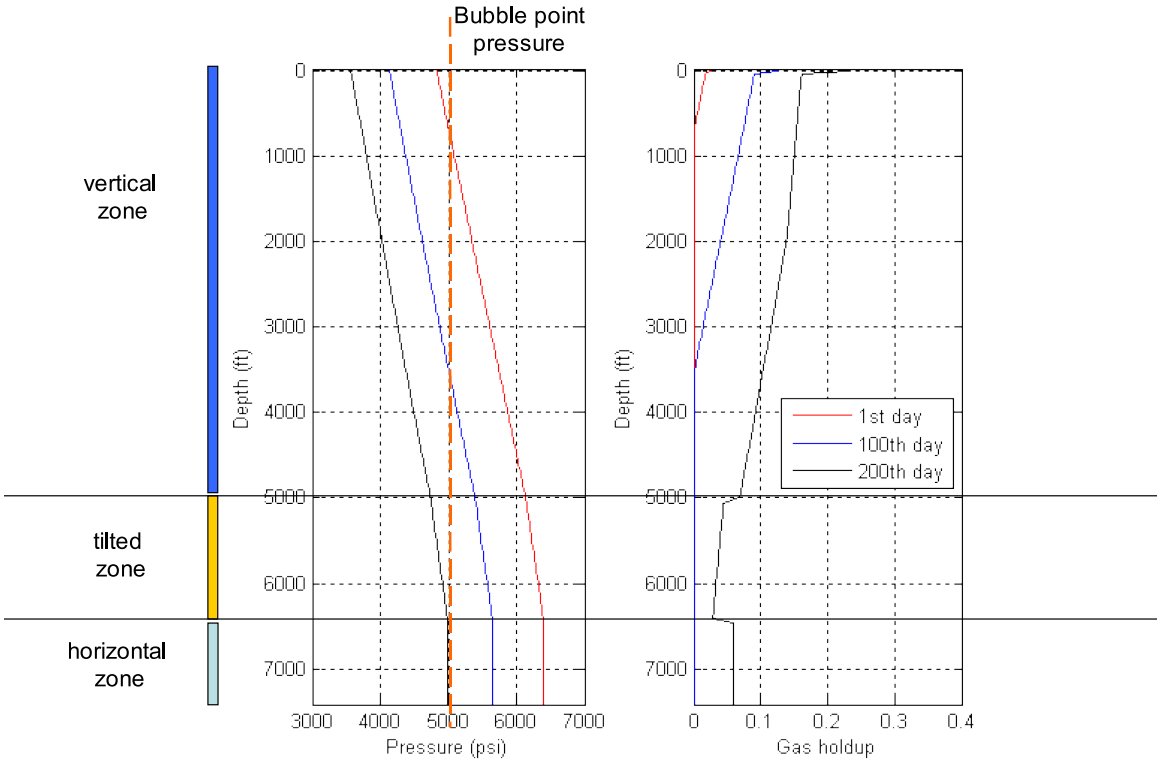


Figure 4.12: Pressure and gas-phase holdup profiles along the wellbore using a drift-flux model with 180 segments (Case 2)

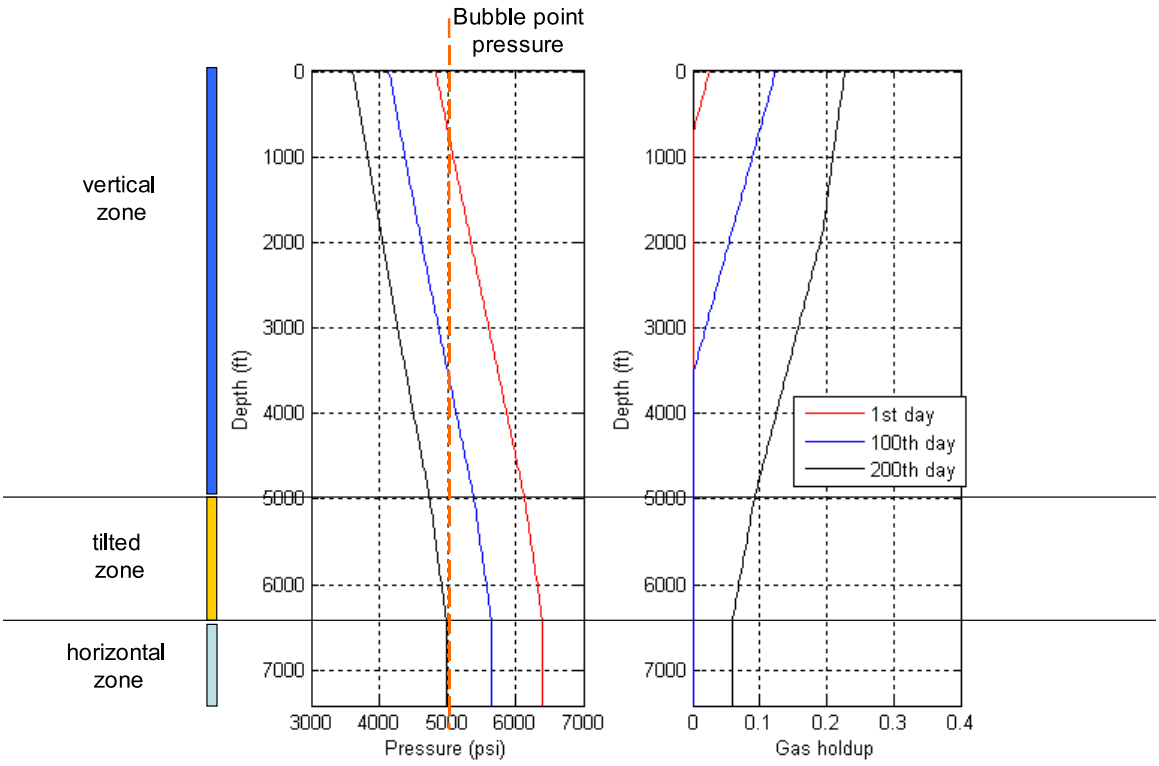


Figure 4.13: Pressure and gas-phase holdup profiles along the wellbore using a homogeneous model with 1800 segments (Case 3)

order to be precise, both pressure and gas-phase holdup profiles from the drift-flux and homogeneous models at 200 days are plotted together in Figure 4.14.

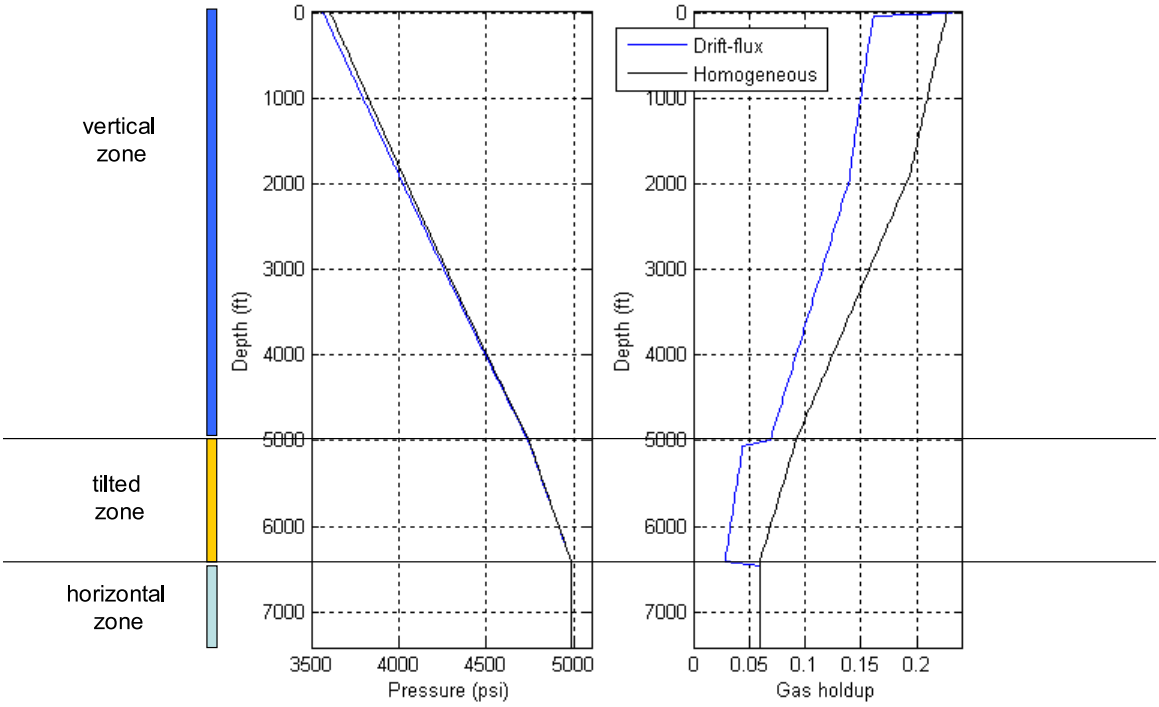


Figure 4.14: Pressure and gas holdup profiles along the wellbore with both homogeneous and drift-flux models at the 200 days

There is no phase slip in a horizontal segment, which means the gas-phase holdup of a drift-flux segment is identical to that of a homogeneous segment. The well is under oil-rate control of 500 bbl/day. Therefore, the results for the horizontal part (pressure, phase holdups, velocities) from both models are exactly the same. There is some pressure drop along the horizontal part due to friction. However, compared to the hydrostatic pressure loss, it is too small to be observed in the figure. In the tilted and vertical zones, the two pressure profiles deviate from each other. This deviation

comes mainly from the difference in the mixture density. The drift-flux model leads to higher fluid mixture density, because the gas phase holdup is much smaller. Recall that the pressure drop profiles in the horizontal part are the same. Therefore, heavier fluid mixtures result in smaller pressures in the wellbore. The largest deviation is about 41 psi, which takes place at the top segment.

The gas phase holdup profile from the homogeneous model is smooth and monotonically increasing from the end of the tilted zone to the surface. The velocity of the gas phase is always the same, by definition, to the velocity of the oil phase. So gas holdup is independent of the segment shape (tilted or not). Pressure is the dominant factor in gas holdup. Smaller pressure leads to larger gas holdup. Both curves have a kink at the location corresponding to 4000 psi, which is due to the gas phase PVT table. The gas phase velocity in the homogenous model is much lower than that from the drift-flux. This results in much higher gas holdup in the homogenous model for most segments.

The MSWell model was integrated into GPRS as an independent module. The model can work with the various kinds of reservoir models and formulations that GPRS supports. A case with MSWells and large-scale highly heterogeneous reservoir model is tested in the chapter on Linear Solution Strategies. The plot of that system is shown in Figure 7.22.

4.3 Concluding Remarks

The multisegment well model is based on discretizing the wellbore into segments, or control volumes, and writing the governing equation. This is a very general framework for wellbore simulation. There are many extensions that can be made to the MSWell

model we designed and implemented in GPRS. One is modeling various devices with special segments, such as valves, downhole separators, etc. These extensions can be used to model smart wells, or other special well models.

Another direction is to incorporate new flow models. The homogenous and drift-flux models have been implemented in GPRS as two subclasses, 'HomoSegment' and 'DFSegment'. They share a common base class - 'Segment'. The only difference between these two segment types is the way they handle the flux term between segments. New flow models can be incorporated without changing any higher level functionality, or component, in the architecture. These extensions can make the MSWell framework a more sophisticated and accurate wellbore flow modeling tool.

It is also important to handle highly complex wellbore geometries with segments. When we define segments and variables in MSWell, the flow direction is implicitly assumed. For example, in Figure 4.4, segments 3 and 5 are the toe segments of segment 2. This setting implies that segments 3 and 5 are similar to each other but segment 2 is different. For a production well, the flow direction is assumed to be from segments 3 and 5 to segment 2. However, from a more general viewpoint, the flow direction of the phase may be quite complex. The flow direction should not be presumed. In order to handle general branching systems with loops. We need to change the way we define segments and variables. Such a general model can be used to simulate surface pipeline systems as well as wellbores. This extension will be discussed in Chapter 6.

Chapter 5

Well Group Modeling

In the last chapter, we discussed well modeling in reservoir simulation. Both the standard well (StdWell) and multisegment (MSWell) were covered. Our focus was on the MSWell model due to its generality. In practice, an oilfield may contain thousands of wells, and most of them do not have independent controls. For example, pipelines may be used to link a number of wells to a junction and form a well group. Control is then applied on the junction to tune the pressure, or flow rate, of the system.

Here, we define the well-group as a single facility object. The definition covers both the underground part (well) and the surface part (pipeline). Traditionally, well-group modeling does not consider the pipeline part and uses a guide-rate to simplify the problem. This is an easy method, but not rigorous. In this chapter, we propose a new well-group modeling approach, which can avoid the shortcomings of the traditional method. We will also show how the well-group model can be integrated into the new reservoir-facilities simulation framework as a facility object.

5.1 Objects and Concepts

A well group is two, or more, wells that are connected to a common junction by pipelines. These wells share a common outlet/inlet at the junction. The junction is also the point where constraints for the entire well group are applied. The wells in a group can be production or injection wells. The wells interact with each other through the reservoir, pipelines, and the junction. The wellhead is the terminal point of a wellbore at the surface. The wellhead may have valves, or pumps, installed to adjust the exit pressure or fluid flow rate. Figure 5.1 shows a diagram of a well group.

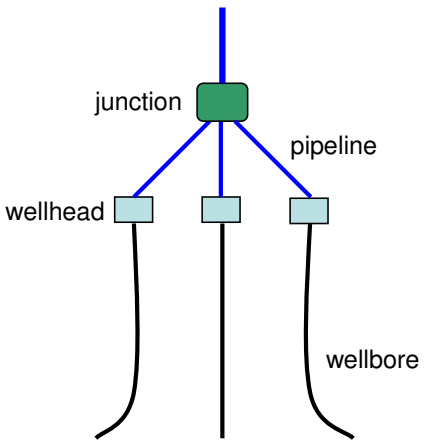


Figure 5.1: Single-level well group with three subordinate wells

The wells in an oilfield may be distributed over a very large area. Nearby wells are usually grouped together, and a number of well groups may be connected by pipelines to form a higher well-group level. Figure 5.2 shows a two-level well group, in which the level-two group has three well groups. Even higher well-group levels can be constructed in a nested manner.

Figure 5.3 shows a diagram of a more complex layout of a multilevel well group in

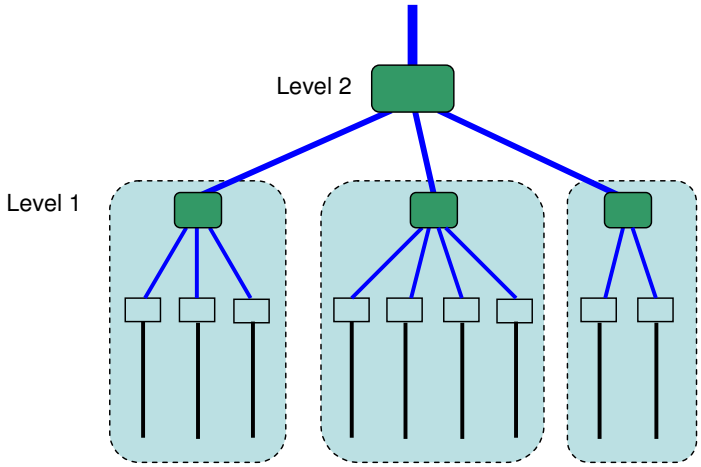


Figure 5.2: Two-level well group with three subordinate well groups

a modern offshore oilfield. A number of wells in two reservoirs are grouped into three groups, and the three groups are connected by pipelines to form a level-two well group. Some pipelines connecting the level-one well group junctions to the level-two junction can be several kilometers long. The exit of the level-two well group is connected to floating production, storage, and offloading facilities (FPSO). We are interested in simulating systems like this with arbitrary implicit levels, from the reservoir to the storage.

Some commercial simulators use a guide-rate to handle well groups. The guide rate is specified by the user, or calculated according to well production potential from the last timestep [20]. The target rate of a well-group is allocated to the individual wells based on the guide rate. In other words, during a timestep in a simulation, there is no group constraint in force. Each well operates at a constraint according to its share of the total rate. This is an easy method to handle well groups. Although the target rate is strictly honored, the well operating rates may not be consistent with the true behavior. The error can be significant when the timestep size is large, or

when there is a special event, e.g., water breakthrough.

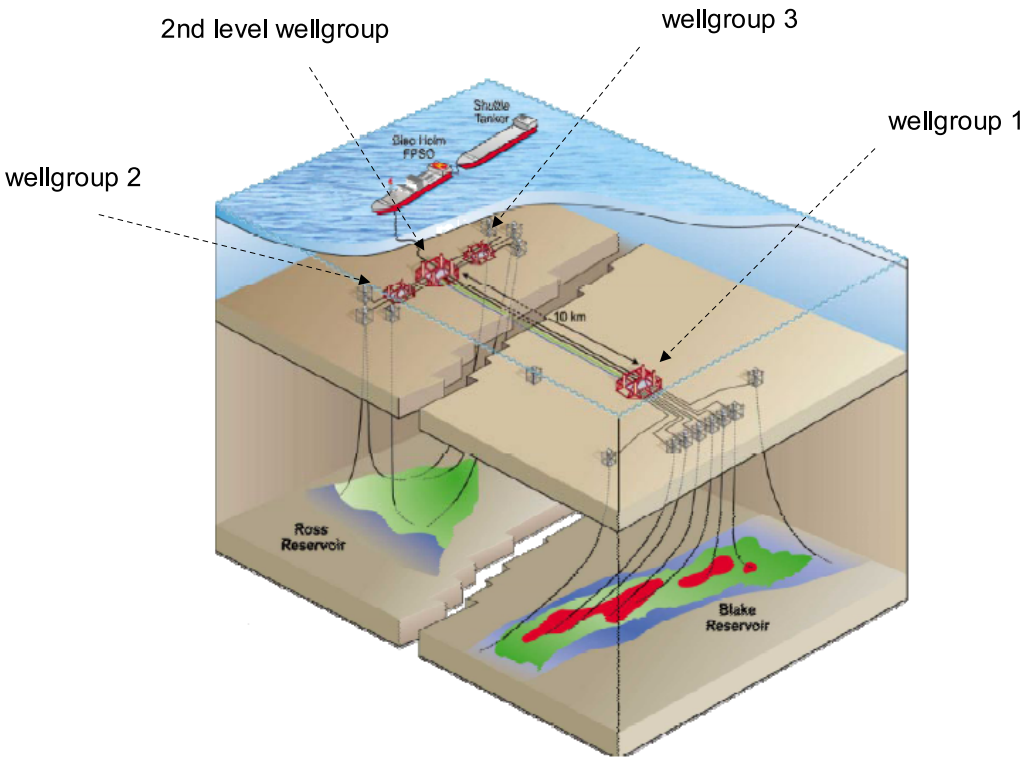


Figure 5.3: Production system of a modern offshore field [18]

In this chapter, we propose and implement a rigorous fully implicit treatment of coupled well-groups and reservoir models. The well-group model accounts for interactions between the wells in the group and incorporates pipeline correlations when necessary. This framework is extensible to other well models. For instance, the user can include MSWells, StdWells, or both types in one group. Minimal code changes are needed for incorporating newly developed well models in the future.

In a well group system, we associate one constraint with the highest well-group level. The lower well-group levels, or individual wells, may have their own economic and physical limits, such as minimal wellbore pressure, maximum water cut, etc.

However, during the simulation, these lower-level limits are not in force. The simulator only checks whether the limits are violated at the end of each timestep. The limits act as triggers leading to some events, such as switching group constraints, closing or reopening of wells.

5.2 Jacobian Matrix Structure

The new general framework of GPRS offers great advantages in handling systems with well groups. With some extension, the multilevel data structure can effectively represent and process the well-group information. We describe the process using the simple synthetic case shown in Figure 5.4. In this case, the system is composed of a small reservoir model and four wells. Three wells, two MSWells and one StdWell, are grouped together; there is also one individual MSWell in the system. As always, the system can be expressed using a 2×2 block Jacobian matrix (upper right corner of Figure 5.4), which is composed of four submatrices, $\boldsymbol{J}_{RR}$, $\boldsymbol{J}_{RF}$, $\boldsymbol{J}_{FR}$, $\boldsymbol{J}_{FF}$.

Although there are four wells in the system, the 'facilities' class only sees two objects: a well-group and one individual well. This is because there are only two constraints in the system. The individual wells in the group do not have their own constraints; hence, they do not appear as facility objects. As a result, the facilities matrices have two sets of submatrices, which are the contributions from the well-group and the individual well (lower right corner of Figure 5.4). At this level, the matrices from the individual well and the well-group are equivalent, but it is quite obvious that the details of the contribution from the well-group will be much more complicated than the contribution from the single MSWell. (A diagram of MSWell matrices is shown in Figure 3.6).

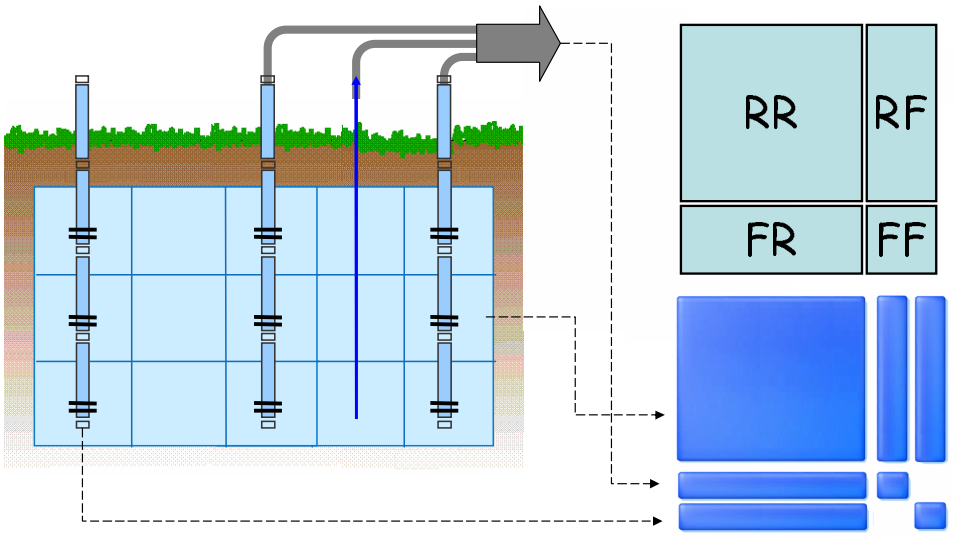


Figure 5.4: Well group in reservoir and corresponding matrices

The matrices from a well group are $\boldsymbol{J}_{GR}$, $\boldsymbol{J}_{RG}$ and $\boldsymbol{J}_{GG}$. Here, $\boldsymbol{J}_{RG}$ represents the part of the Jacobian matrix corresponding to reservoir equations with respect to well-group variables. The other two are named in a similar manner. The individual well submits its matrix contribution to the facilities object, while the wells in the group submit their matrices to the well group. The well-group matrices are wrappers, which contain matrices from the wells in the group and three matrices from the junction. The detailed structures of the well-group matrices $\boldsymbol{J}_{GR}$ and $\boldsymbol{J}_{GG}$ are shown in Figure 5.5. $\boldsymbol{J}_{GR}$ contains $\boldsymbol{J}_{WR}$ matrices from the wells. $\boldsymbol{J}_{GG}$ contains a number of $\boldsymbol{J}_{WW}$ matrices (in blue) from the wells and three matrices from the junction (in green). $\boldsymbol{J}_{RG}$ is in a form similar to transposed form of $\boldsymbol{J}_{GR}$. $\boldsymbol{J}_{RG}$ is composed of $\boldsymbol{J}_{RW}$ matrices from the wells in the group.

Overall, the matrices of the well-group model are very similar to those of the facilities class, except that $\boldsymbol{J}_{GG}$ is enlarged by matrices from the junction, and the dimensions of $\boldsymbol{J}_{GR}$ and $\boldsymbol{J}_{RG}$ are changed correspondingly. (See Figure 3.5 for the

structure of the facilities' matrices). In the next section, we will discuss the equations and variables that make up these matrices.

5.3 Governing Equations and Variables

The variables of a well-group model are composed of the variables from the wells and the ones from the junction. The variables from the wells depend only on the well type. The StdWell model has only one primary variable - bottom hole pressure, while the MSWell model has a large variable set. No new variable is defined for the wells in the group, but the junction introduces a few new variables. For the case shown in Figure 5.4, the well variables in the well-group have three subsets, coming from the three wells. The variables from the junction depend on the type of constraint on the well-group. If the well-group is under constant oil-rate control, the junction variables are the pressure at the junction, p_J, and n_w (number of wells in group) oil component rates from the wells. In this case, they are $q_{\bar{o}1}$, $q_{\bar{o}2}$, $q_{\bar{o}3}$.

For the first well in the well group shown in Figure 5.4 (MSWell), its governing equations are very similar to the ones if the well was an individual one. The equations for each segment are exactly the same, which are the pressure relation equation and mass balance equations (Equation 4.8 and 4.12). However, the constraint equation is slightly different. We use a constant oil-rate constraint as an example, and its expression is written as:

$$R = F(p^w, ...) - q_{\bar{o}1} = 0. \tag{5.1}$$

In this equation $F(p^w, ...)$ is the wellbore flux, which is a function of the wellbore variables. The formulation of the flux depends on the type of well model. For the

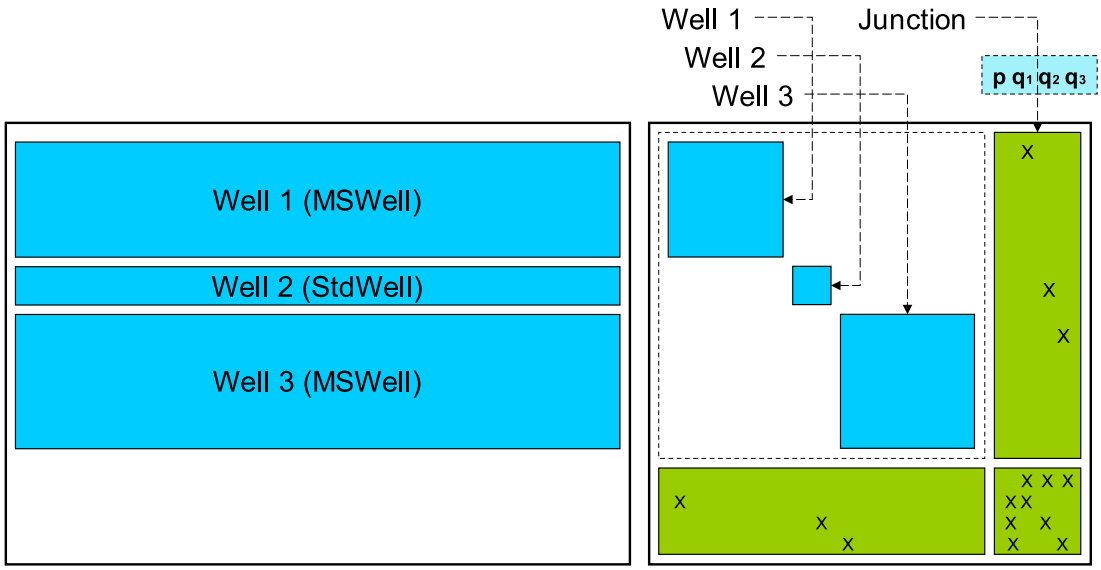


Figure 5.5: $\boldsymbol{J}_{GR}$ and $\boldsymbol{J}_{GG}$ matrices of well group

MSWell model, the formulation is Equation 4.4. For the StdWell model, the formulation is Equation 4.21. To be general, we do not write down the specific expression of the flux function.

For an individual well, $q_{\bar{o},1}$ is a user-specified constant. In the well group case, $q_{\bar{o},1}$ is a variable that is associated with the junction object. In both scenarios, the derivatives of Equation 5.1 with respect to the well variables are identical. Therefore, the Jacobian matrices of the well in the well group are exactly the same as if the well was alone. In other words, no special treatment is needed for the Jacobian matrix of the well in a well-group. Since $q_{\bar{o},1}$ is a junction variable, there is one additional derivative, $\partial R/\partial q_{\bar{o},1}$. According to Equation 5.1, it is a constant of -1. The derivative belongs to one of the junction matrices.

In each Newton update, once the junction variables, $q_{\bar{o},i}$, are updated, they will be sent to the corresponding well. Hence, when the well model linearizes its equations

for the next Newton iteration, it will use the proper rate. Consequently, no special treatment is needed for the right-hand-side vector.

The junction object introduces $n_w + 1$ new variables. Therefore, the same number of governing equations is needed for the junction. The first is the constraint equation for the entire well group, which is written in Equation 5.2:

$$R = \sum_{i=1}^{n_w} q_{\bar{o},i} - q_{\bar{o},t} = 0, \tag{5.2}$$

where $q_{\bar{o},t}$ is the target constant-rate for the entire well group. If the well is under constant pressure control, the constraint equation is written as:

$$R = p_J - p_t = 0. \tag{5.3}$$

The remaining n_w governing equations are the pressure relations between the well reference pressure and the junction pressure:

$$R = p_i^w - p_J - \Delta p(q_{\bar{o},i}, ...) = 0 \ \ (i = 1, 2, 3, ...), \tag{5.4}$$

where $\Delta p(q_{o,i}, ...)$ is the pressure correlation for the pipelines between the wellheads and the junction. Here, we consider the friction effect along the pipeline, which is:

$$\Delta p_f = \left(\frac{2 f_{tp} \rho V_m^2}{d} \right) \ L, \tag{5.5}$$

where L is the length of the pipeline. This equation can be replaced by other pipeline correlations as needed.

For each iteration, the Jacobians of the well objects in the group are updated with the last iteration results. Then, the wells submit their matrices to the well-group

object. The well-group object prepares the matrices for the junction and combines the matrices from both the junction and the wells to form the well-group matrices. Then, the well-group matrices are submitted to the facilities object. The process of constructing the right-hand-side vector for the well-group is similar to the above process.

5.4 Example

A simple case was designed to show preliminary results of well-group simulation. The reservoir model is homogenous with 20×20 gridblocks. There are four wells in the system. Three MSWell producers are grouped into one well group. The constraint for the well group is 1500 bbl/day of oil production. One StdWell is an individual water injector under constant-pressure control. Producer #1 has the shortest distance to the injector and Producer #3 is the farthest well from the injector. The system is shown in Figure 5.6.

Initially, the reservoir contains oil and immobile water. Hence only oil is produced in the early period. The oil and water production rates from the three subordinate wells and the entire group are plotted in Figure 5.7 and Figure 5.8, respectively. Among the three producers in the group, Producer #1 has the largest production rate since it has the shortest distance to the injector. Producer #2 has a smaller rate, and the rate from Producer #3 is the smallest one. The production rates of the three wells are quite stable during the early stage. After about 170 days, water breaks through in Producer #1, and its oil production drops significantly thereafter. The well group has to lower the junction pressure to boost oil production from Producers #2 and #3. After about 220 days, water breaks through in Producer #2, and the oil rate of Producer #2 begins to drop. Producer #3 becomes the workhorse of the

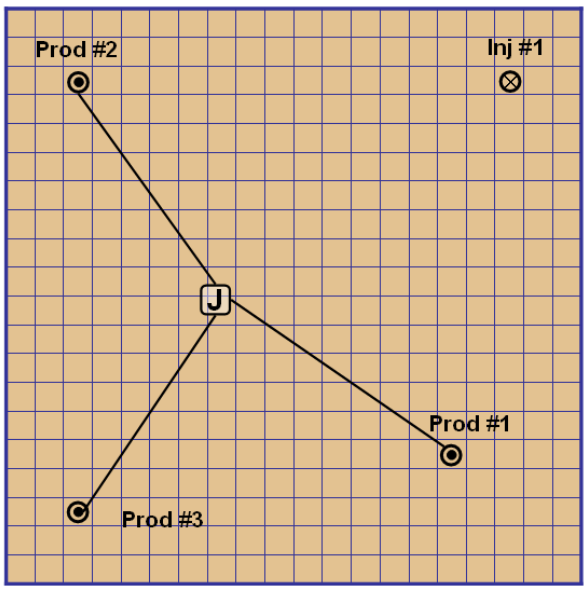


Figure 5.6: Reservoir with well group

system providing most of the oil production.

In this case, the simulation runs in fully implicit mode. The well group model is fully coupled with the reservoir model. There is no assumption on the rates, or pressures, for the wells in the group or for the junction. The wells in the group compete with each other to produce at the junction constraint.

5.5 Concluding Remarks

In this chapter, we proposed a new approach to model well groups in reservoir simulation. The treatment honors the well-group constraint and accounts for the coupling between the wells and the reservoir. The model was integrated into the new GPRS framework. The well-group model in GPRS takes advantage of the flexible data

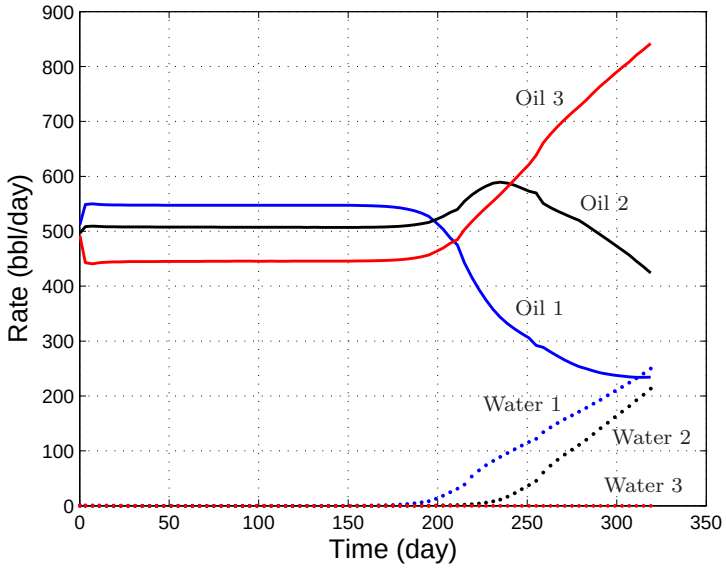


Figure 5.7: Oil and water rates of subordinate wells in well group

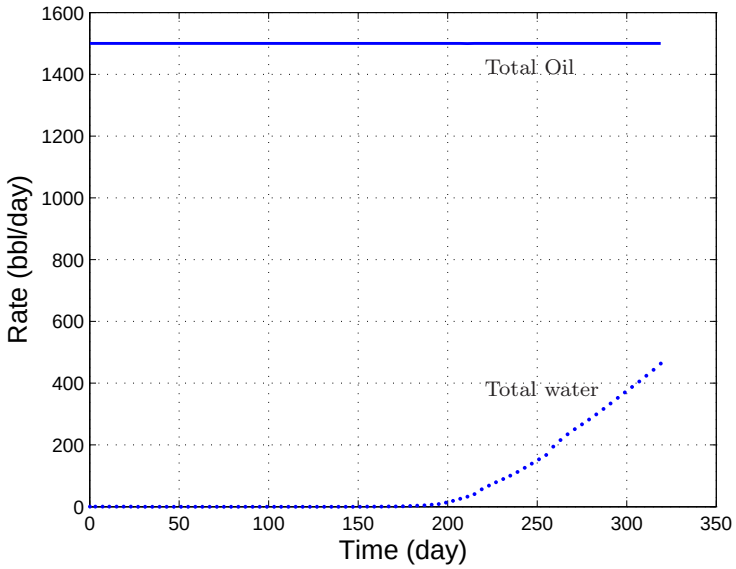


Figure 5.8: Total oil and water rates of well group

structures, and it is built using existing well models as a base. This reduces the development and maintenance cost significantly. The new option enables GPRS to perform fully coupled reservoir and well group simulation. The development of the well-group model also demonstrates the flexibility of the new GPRS framework.

There are some suggested future extensions for well-group modeling:

- Implement other pressure correlations and flow models.
- Implement nested well groups, whose matrix is shown in Figure 5.9.
- Develop more effective linear solution strategies for systems with well groups.

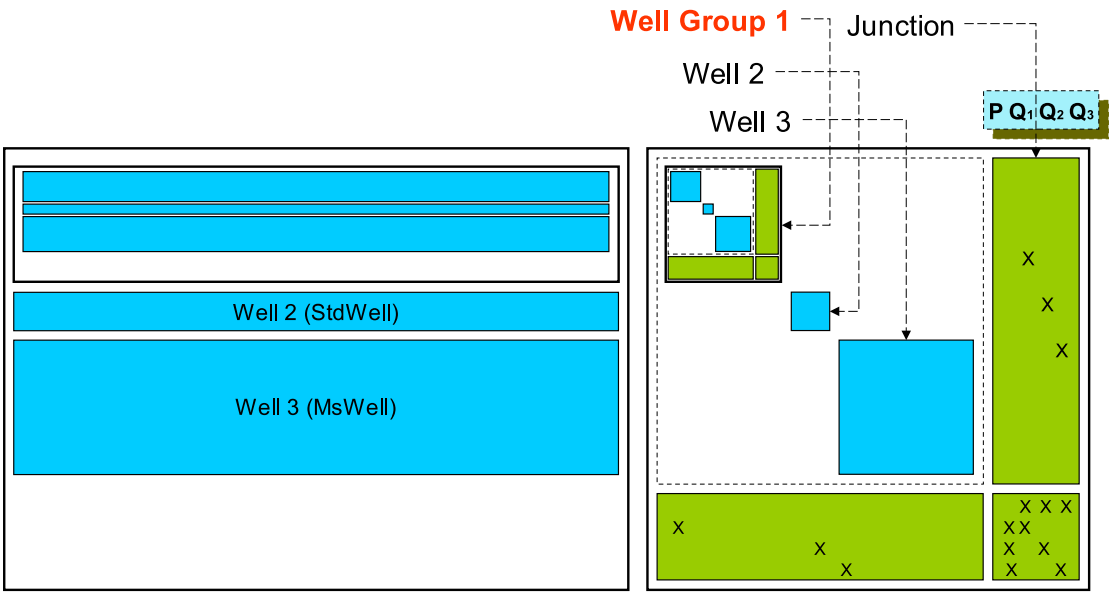


Figure 5.9: $\boldsymbol{J}_{WR}$ and $\boldsymbol{J}_{WW}$ matrices of a nested well group

Chapter 6

General Pipeline and Wellbore Modeling

The pipeline network plays a very important role in oilfield production operations. Most of the time, the constraints in a production system are not on wellheads, but on gathering systems and storage facilities. The pipeline network links the wellheads to these facilities. Therefore, it is crucial to have fast and accurate pipeline simulation tools.

The multisegment well model (MSWell) can handle transient effects very well. In order to simulate the flow from the perforation to the production gathering system seamlessly, we extend the MSWell treatment to cover pipeline networks. However, pipeline networks have much more complex topology, including general branching, loops, and multiple exits. The flow direction in pipeline networks may not be preassumed. The existing MSWell model is unable to handle such cases. In this chapter, we propose a new consistent model that works for both wellbore and general piping systems.

Most of the discussion in this chapter is based on modeling pipeline systems, but, the approach is directly applicable to discrete wellbore models. This new general discrete wellbore model is a stand-alone simulation package, but we describe how it can be integrated into the GPRS simulator.

6.1 Segments, Nodes and Connections

Segments and their variables were defined in Chapter 4 for the MSWell model (Figure 4.3 and 4.4). However, those specifications may not be effective for pipeline networks. The pipeline system may contain loops, crossovers and multiple exits. Therefore, we developed a general model, referred to as "GenWell", to handle a general branching system like a pipeline network or a complex well.

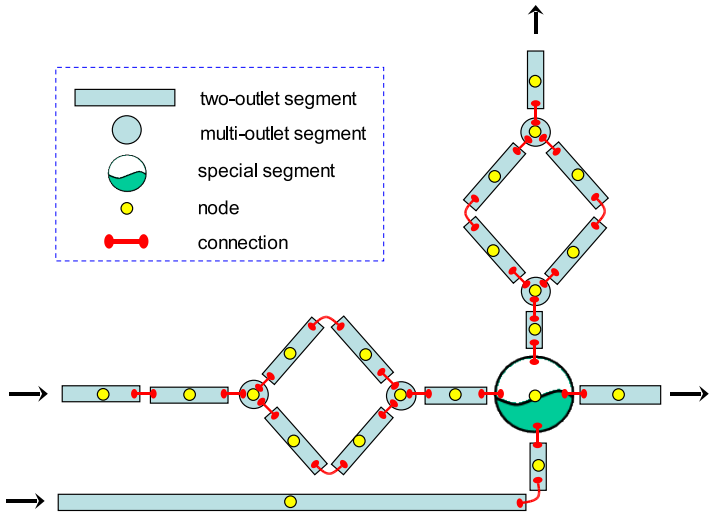
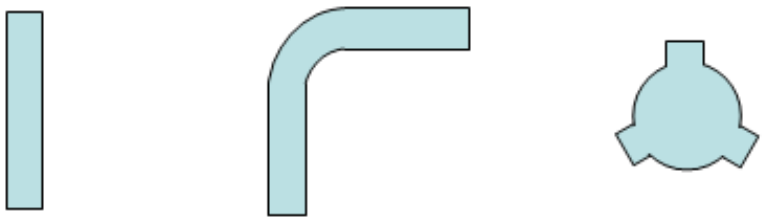


Figure 6.1: Discretized pipeline

The pipeline system is discretized into cells, called segments. An example of discrete pipeline networks is shown in Figure 6.1. In the GenWell model, there are three types of segments: two-outlet, multi-outlet, and special. The two-outlet segment is the most common segment object. It is a section of pipeline with outlets at either end. A two-outlet segment can always be split into an arbitrary number of smaller two-outlet segments. A two-outlet segment is not necessarily straight, but may contain bends and corners. However, in order to obtain optimal accuracy, we suggest splitting the segments until all of them are nearly straight. The two-outlet segments are used to represent the main part of the pipeline system. A multi-outlet segment, as suggested by its name, has more than two outlets. These segments are used to represent joints in a pipeline system. At a joint, there may be three or more branches. Each branch is connected to one outlet of a multioutlet segment. Figure 6.2 shows a straight two-outlet segment, a bent two-outlet segment, and a multioutlet (three) segment. Two-outlet and multioutlet segments have the same set of variables and equations. A shape factor is the only difference between them. With these two types of segments, we are able to describe arbitrary shapes in pipeline networks.



(a) straight two-outlet segment (b) bent two-outlet segment (c) multioutlet segment

Figure 6.2: Pipeline segments

It is common to have some special equipment in a pipeline network, such as separators, valves etc. They may have very different functionality than just being a

"pipe". We use special segments to model such equipment. They may have different variables and governing equations compared to two-outlet and multioutlet segments. Special segments are on the list for future development. We provide a framework to handle these special segments, but they are not the focus of this discussion.

In the GenWell Model, each segment has one and only one node associated with it. The node is located at the center of the segment. The variables associated with a segment are defined on the node. An abstract object "connection" is defined to represent the relation between two connected segments. Connections have one-to-one relation with outlets, i.e., a two-outlet segment has two connections associated with it, and a multioutlet segment has more than two. The fluid-mixture velocity is defined on the connection between two segments. Exits in the pipeline system are treated as special connections, which connect exit segments to the external environment.

Discretizing a general branching system into segments is a gridding process. For a given pipeline network, joints in the network are first identified and labeled as multi-outlet segments. These multi-outlet segments split the entire pipeline network into a number of two-outlet piping sections. In the next step, these large two-outlet sections can be split into many smaller two-outlet segments according to accuracy requirement. Once the segments are set up, nodes are assigned to the centers of the segments. All the segments are labeled and form a segment list. Connections are then built up according to the structure of the network. Each connection represents a connection relation between two segments. Generally, the number of connections is not the same as the number of segments. All of these connections form a connection list, which is very similar to the one used in reservoir simulation.

The volume of a two-outlet segment is decided by its length and the cross-sectional area. In the GenWell model, the fluid is assumed to be well mixed at junction points. Therefore, the multi-outlet segments are junction points with reasonably

small volumes. The suggested value is on the order of D_{in}^3, where D_{in} is the inner diameter of the pipe at the joint point. The small size of the joint segments reflects reality and ensures the fluid is well mixed at a joint. In some circumstance, different branches may not split the phases evenly. This can be modeled by using different governing equations for the multi-outlet segment. However, we do not consider this scenario in this work.

After the discretization process, the pipeline system is abstracted as a graph. There are a number of nodes, on which primary variables are defined. A segment can be seen as a certain volume attached to a node. Component mass balance equations will be written based on this volume. The connection list describes the topological structure of the nodes, and the relative height of nodes is needed as well. This is very similar to the method used for reservoir models. For some specific pipeline flow model (e.g., drift-flux model), the inclination can have an impact on the flow. So this information is preserved in the segment. Figure 6.1 shows a discretized general branching system, where all the segments, nodes, and connections are labeled.

6.2 Governing Equations and Variables

By introducing connections and multioutlet segments, the GenWell model can easily discretize arbitrary pipeline networks and wellbores. The variables of GenWell are very similar to MSWell, but the places they are defined are different. Velocity is not a segment variable any more; instead, it is defined as a connection property. In the MSWell model, the momentum equation of one segment reflects the relation between itself and its heel segment. In the GenWell model, the momentum equation is written for a connection, which naturally reflects the pressure relation of two connected segments.

For the black-oil GenWell model, the variables of a segment are very similar to the ones in a reservoir cell. If the fluid is single-phase/single-component, pressure p^{seg} is the only variable. If the fluid is three-phase/three-component, two phase holdups $\alpha_{g,i}$, $\alpha_{w,i}$ are primary variables. If the fluid is two-phase/two-component, we pick one of the holdups as a primary variable. These variables are defined on nodes, which are at the center of segments. The variable defined on a connection is the fluid mixture velocity, V_m. In a general branching system, the number of connections is not the same as the number of segments. Typically, when a loop is introduced into a network, it adds more connections than segments.

For each connection (except an exit connection), the pressure relation between two segments can be written as (ignoring acceleration)

$$R_p = p^{seg}_{up} - p^{seg}_{down} + \rho\, g\, (h_{up} - h_{down}) - \frac{2 f_{tp} \rho V_m^2}{D}\, \Delta x = 0, \tag{6.1}$$

where h is the node height of each segment. The reference point for h is the ground surface, and h is negative for subsurfaces pipes (i.e., wells). Subscripts *up* and *down* are used to avoid pre-assumed flow directions.

For each segment, mass conservation equations of components are the governing equations. Their discretized form is:

$$\begin{aligned} R_{m,c} =& \frac{\boldsymbol{V}}{\Delta t} \sum_{p=1}^{np} \left[(\rho_p\, x_{c,p}\, \alpha_p)^{n+1} - (\rho_p\, x_{c,p}\, \alpha_p)^n \right] + \\ & \sum_{i=1}^{n_{conn}} \sum_{p=1}^{np} (A\, \rho_p\, x_{c,p}\, V_{sp})_i^{n+1} \qquad (c = g, o, w), \end{aligned} \tag{6.2}$$

where $\boldsymbol{V}$ is the segment volume, and n_{conn} is the number of connections associated with the segment.

Besides the momentum and mass balance equations, the pipeline system may have constraint equations on exits. Similar to other well models, constraints in the GenWell model can be constant pressure, or constant rate. In the GenWell model, the pipeline network, or well object, may have more than one exit, and each of them has an independent constraint. The constraint types and parameters can be different. For example, a GenWell may have three exits. One may be operating under constant pressure of 100 psi; the second may have a constant exit pressure of 120 psi; and the third one could be operating under a constant oil-rate of 200 bbl/day.

The pressure constraint is applied to an exit segment, because pressure is a segment (node) variable. The expression for a constant pressure constraint is written as

$$R_{ctrl,p} = p^{seg} - p_{\,target} = 0. \tag{6.3}$$

The rate constraint is applied to an exit connection, because the velocity is defined on connections. In this case, we have

$$R_{ctrl,q} = A \sum_{p=1}^{n_p} (\, \rho_p \, x_{\bar{o},p} \, V_{sp} \,) - \rho_{\bar{o}} \, q_{\bar{o}} = 0. \tag{6.4}$$

V_{sp} is the phase superficial velocity, which is the same as the one in the MSWell model. In the GenWell model, we start to use a homogeneous flow model (Equation 4.34), in which all phases share the same velocity. Other flow models, e.g., drift-flux model, can be implemented as future extensions.

We use a fully implicit scheme with our GenWell model, which has unconditional stability and allows large timesteps. For a system with n_e exits, n_c connections, and n_s segments, we get n_e constraint equations, $n_c - n_e$ momentum equations (exits are counted as connections) and $n_{comp} \, n_s$ mass balance equations, where n_{comp} is the number of components. In total, we have $n_{comp} n_s + n_c$ governing equations, and this is

also the number of rows in the Jacobian matrix. The ordering starts with the velocity at exit connections, followed by other velocities in the order of the connection list. After that, the pressure and phase holdups of all segments are listed in the order of the segment list. For an oil-water case, the equation and variable sets are given by

$\{R_{p\,1}\ ,\ R_{p\,2}\ ,\ ...\ ,\ R_{p\,n_{conn}}\ ,\ R_{m,o\,1}\ ,\ R_{m,w\,1}\ ,\ ...\ ,\ R_{m,o\,n_{seg}}\ ,\ R_{m,w\,n_{seg}}\}^T$,

$\{V_1\ ,\ V_2\ ,\ ...\ ,\ V_{nConn}\ ,\ p_{o\,1}\ ,\ \alpha_{w\,1}\ ,\ ...\ ,\ p_{o\,n_{seg}}\ ,\ \alpha_{w\,n_{seg}}\}^T$. Therefore, the Jacobian matrix for the system can be written in 2×2 block form as

$$\begin{pmatrix} \frac{\partial R_p}{\partial X_c} & \frac{\partial R_p}{\partial X_s} \\ \frac{\partial R_m}{\partial X_c} & \frac{\partial R_m}{\partial X_s} \end{pmatrix},$$

where R_p represents the pressure equations in residual form, R_m the mass balance residual equations, X_c the vector of connection variables, and X_s the vector of segment variables.

Figure 6.3 shows a Jacobian matrix from a pipeline simulation with the GenWell model. The pipeline structure is shown in Figure 6.4, which has two components, 26 segments, and 28 connections. Therefore, the dimensions of the Jacobian matrix is 80×80. It is very easy to see different non-zero patterns in the four parts of the Jacobian matrix.

6.3 Example

A pipeline simulation case with complex geometry and strong transient effects is designed to test the GenWell model. The structure of the pipeline network is shown in Figure 6.4. The pipeline is discretized into 26 segments and 28 connections. Both

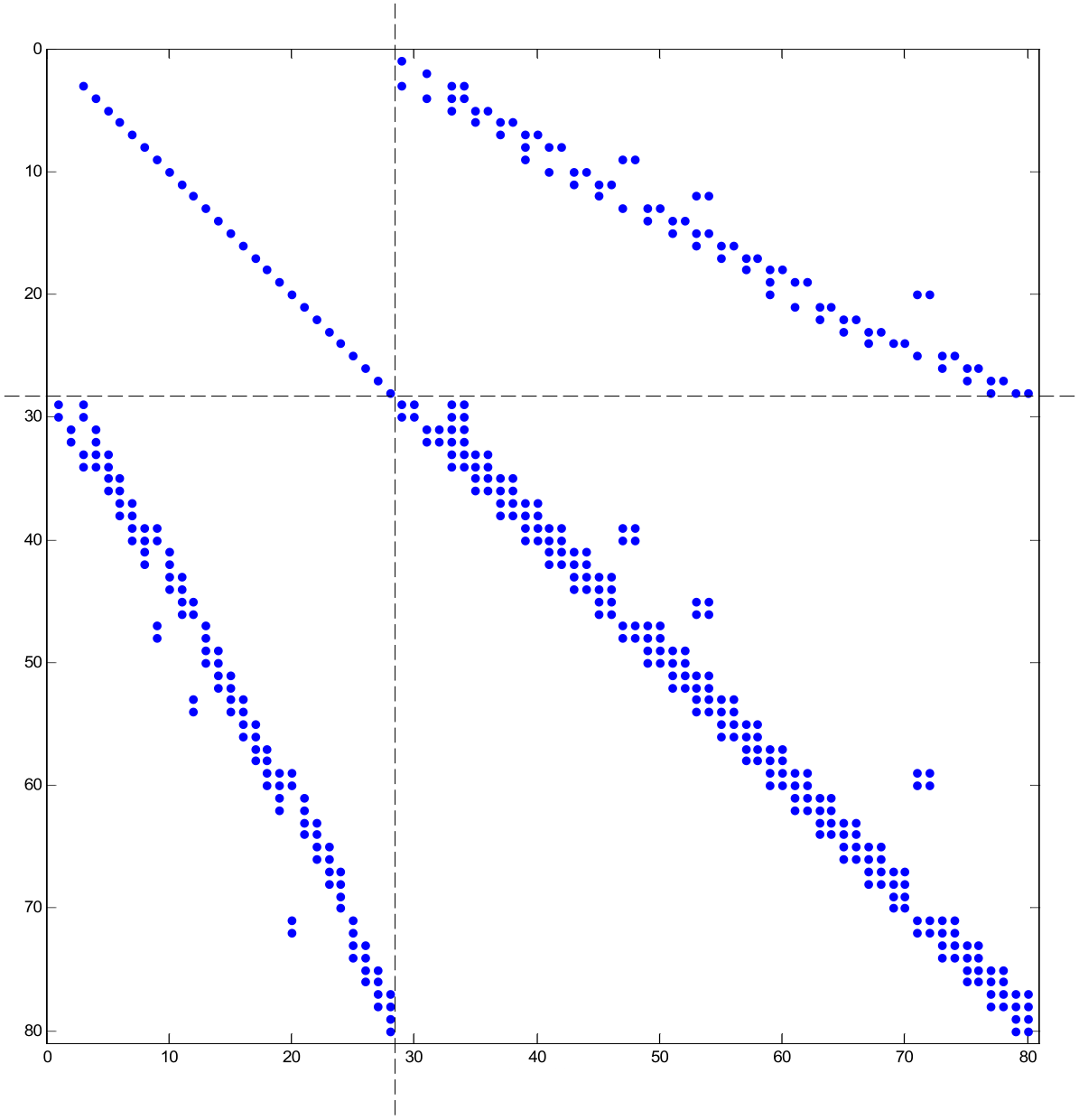


Figure 6.3: Jacobian matrix for a pipeline simulation

segment numbers (labels) and connection labels are shown in the figure. In the discretized pipeline network, segments 3, 6, 13, 16 are multioutlet segments. Segments 21 and 26 have very large volumes and high initial pressure. They can be seen as reservoirs, or tanks, and act as boundary conditions. Other segments are two-outlet ones and have the same geometry. Among them, segments 1 and 2 are the exits of the pipeline system. There is one loop in the system (between segments 6 and 13). The frictional resistance of the right branch (segments 10, 11, 12) is four times larger than the left branch (segments 7, 8, 9). The system contains both oil and water. Initially, Tank #1 (segment 21) has 10% water and Tank #2 (segment 26) has 20% water. The oil phase has a larger compressibility than the water phase. The system runs at constant pressure applied on both segments 1 and 2. The driving force is liquid expansion. In order to limit the depletion rate, the two tank connections (connections 24 and 28) are assigned very high friction coefficients. Therefore, most of the pressure decline happens at these two locations. The energy of the system depletes within 180 days.

The pressure histories of selected segments are shown in Figure 6.5. The left plot shows the pressure of the two tanks (segments 21 and 26). Both of them start from the same initial pressure (1000 psi), but Tank #1 maintains at a higher pressure most of the time. This is because Tank #1 has a higher oil phase percentage (90%) than Tank #2 (80%). The larger overall compressibility of Tank #1 helps it maintain a higher pressure. The right plot of Figure 6.5 shows the pressure history for segments 8, 11 and 16. From Figure 6.4, we can see that segments 8 and 11 are in the corresponding positions of two parallel branches. Although the resistance of the right branch is four times that of the left one, the pressure of segments 8, 11 are always the same. Segment 16 has a slightly higher pressure, since it is the upstream segment.

The water-phase holdup history for several selected segments is shown in Figure

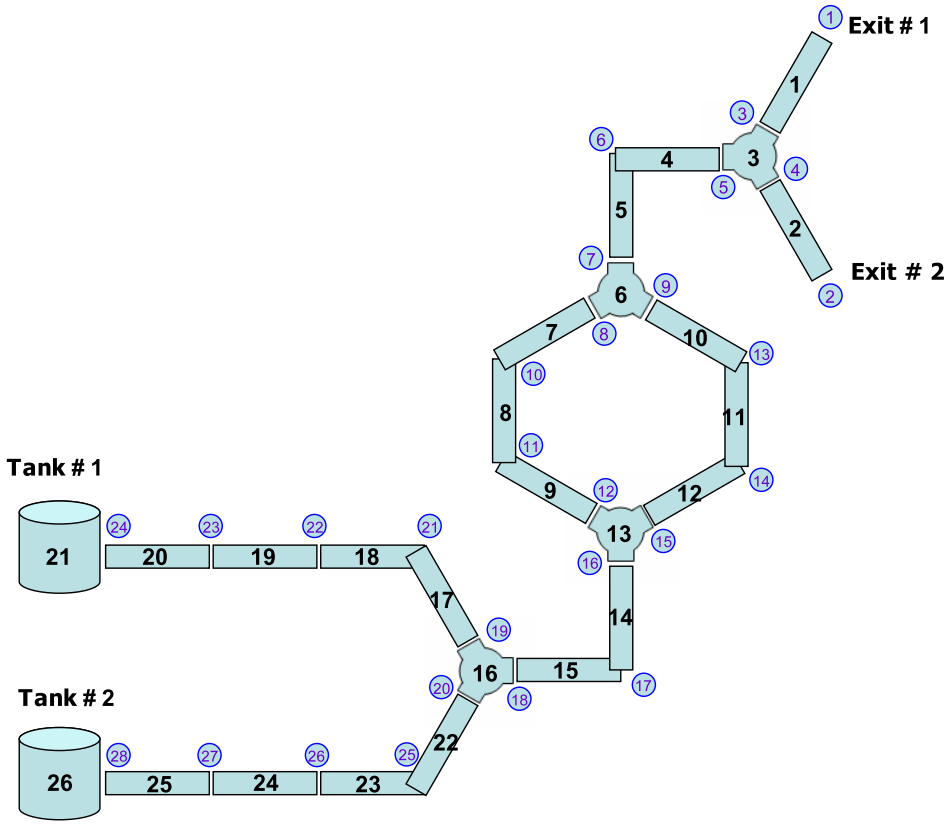


Figure 6.4: Discretized pipeline of the example

6.6. The left plot shows the water phase percentage of the two tanks, which are basically at their initial values (0.1 and 0.2). With time, the water holdups of the tanks drop slightly, because the oil-phase expands more with pressure decline. The right plot of Figure 6.6 shows the water phase holdup histories of segments 3, 16, 17 and 22. Segments 17 and 22 belong to the two branches directly connected to the two tanks, respectively. Therefore, their phase holdup histories are almost identical to the tanks. The water holdups of segments 3 and 16 start from almost the same value, which is a little bit smaller than 0.15. A water holdup of 0.15 is the average value of the two tanks. At early time, almost the same amount of fluid comes out of the two tanks and mixes at the junctions. Since the pressures of segments 3 and 16 are much smaller than the tank pressure, their water holdups are slightly smaller than 0.15 due to different phase compressibilities. Later in time, the water-phase holdups of the two segments drop significantly and that of segment 16 reaches 0.1. Tank #2 has more water and less overall compressibility. Tank #2 loses its energy faster, and the system is gradually dominated by Tank #1, which has 10% water.

The mixture velocity histories of selected connections are shown in Figure 6.7. The labels of the connections are shown in Figure 6.4 with blue circled numbers. The left plot of Figure 6.7 shows the velocities of the two tank connections (connections 24 and 28). Again, we see that Tank #2 depletes faster than Tank #1. The right plot shows the velocity histories of four segments. At any given time, the velocity ratio of connections 16, 11, 1 and 14 is about 6:4:3:2. All the fluids from both tanks need to go through connection 16. Therefore, the connection has the largest velocity in the pipeline. The two exits are under the same constant pressure control, so they share the total-rate evenly. The two branches of the loop have different resistances. The resistance of the right branch is four times that of the left one. Hence the total-rate is split according to the resistance. From Equation 5.5, it is easy to see that the velocity

ratio between the two branches should be 2:1. This analysis ignores volume change due to compressibility (we consider this in the simulation), because the system is slightly compressible and the pressure differences between these segments are small (refer to the right plot of Figure 6.5).

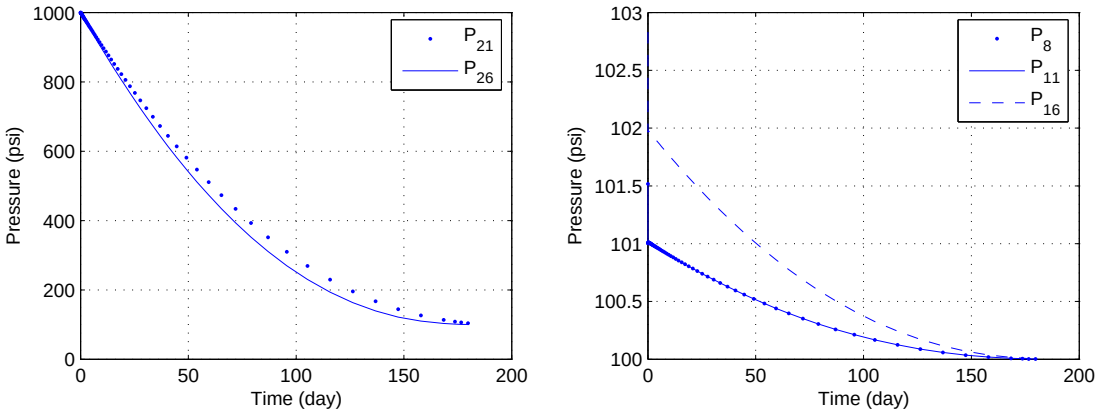


Figure 6.5: Pressure of selected segments

6.4 Comments on Future Extension

We suggest that the GenWell model be implemented in GPRS to completely replace the current MSWell model. This is because the traditional MSWell model is a subset of the GenWell described here. The GenWell model can handle general branching and loops with arbitrary flow directions, and multiple exits with different constraints. An important step in that direction is to incorporate more flow models, e.g., the drift-flux model, into GenWell. Other pressure correlations can be implemented as additional options.

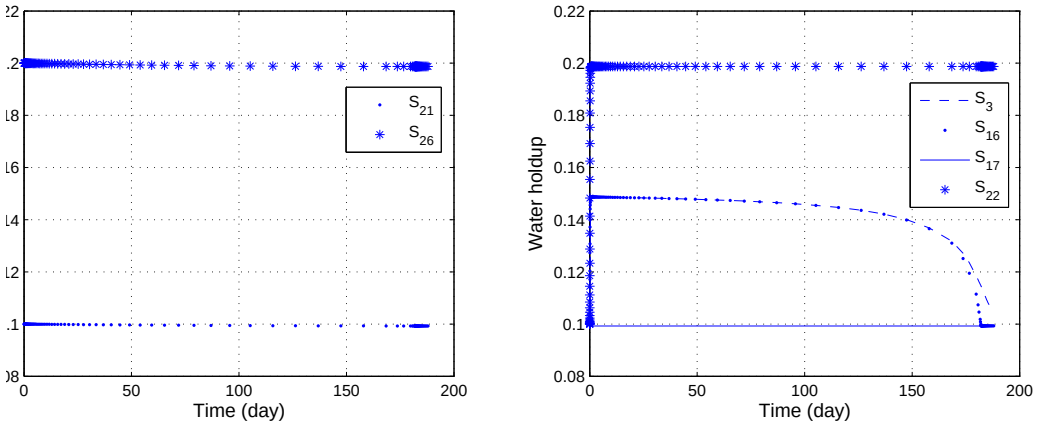


Figure 6.6: Water holdup of selected segments

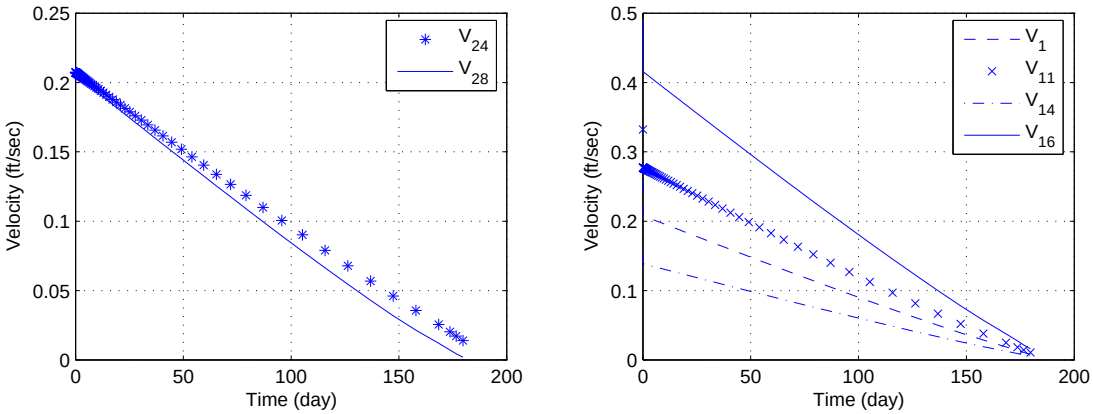


Figure 6.7: Velocity of selected connections

The GenWell model overlaps somewhat with the well-group model. The well-group model (discussed in Chapter 5) contains independent subordinate objects, i.e., the wells in the group. Therefore, the well-group matrix has substructure (Figure 5.5). The advantage of the well-group model is that it can combine different well models (e.g., StdWells and MSWells) in one group easily. With the implementation of new well models, the capability of the well-group model is naturally expanded. However, in the well-group model, the treatment for the pressure relation between the wellhead and the junction point is relatively simple (Equation 5.4). The well-group model has pre-assumed flow directions and cannot handle complex topology (e.g., loops). The GenWell model is much more general. GenWell discretizes the entire subsurface and surface piping system into segments and is able to provide much more detailed information about the flow behaviors in complex wellbores and pipe networks.

Chapter 7

Linear Solution Strategies

In Chapters 2-6, we described the new design of the GPRS computational framework, in which the reservoir and facility models are treated as separate numerical components. New data structures consistent with this separation were designed and implemented. The information is encapsulated within different objects and organized in an efficient manner, and close attention has been paid to efficient computation using these new multi-level data structures. The multi-segment well (MSWell) model and rigorous treatment of well-group constraints have been integrated into the new GPRS simulator. The computational platform is quite flexible and can be further expanded in a variety of ways.

As described in detail in Chapters 4, the MSWell model introduces a large number of equations and unknowns for each well. The Jacobian matrices for reservoir models with MSWells are much more complex than those from reservoir models with StdWells. One needs to tailor the solution strategy to accommodate the Jacobian matrices obtained when large numbers of facilities equations and unknowns are present. To efficiently solve large-scale, highly heterogeneous, unstructured reservoir models

with advanced well models, we developed a new linear solver strategy for the new GPRS framework. In this chapter, we discuss these new solvers and preconditioners.

Specifically, we designed and implemented a scalable multistage preconditioner for large-scale reservoir models coupled with multilateral, multisegment wells. The overall strategy is based on the two-stage, CPR approach [53]. In the first stage, the full Jacobian matrix goes through two restrictions. Each MSWell is algebraically reduced to a StdWell-like equation first. Then IMPES reduction is applied to the reduced matrix to generate the corresponding pressure matrix. A multilevel, block ILU(k) preconditioner is used for the second stage. Our multistage linear solver allows for the simultaneous presence of standard and multisegment wells (with a wide range of well controls) in the model. Several cases are used to demonstrate the performance of the CPR-based preconditioner for reservoir models with multisegment wells.

7.1 Linear Solvers and Preconditioners in GPRS

The nonlinear coupled algebraic equations in reservoir simulation can be written as:

$$\boldsymbol{R}^{n+1} \equiv \boldsymbol{R}(\boldsymbol{u}^{n+1}) = 0, \tag{7.1}$$

where $\boldsymbol{R}$ is the residual function, $\boldsymbol{u}$ is the unknown vector, n is the timestep level. In reservoir simulation, the Newton-Raphson method is usually used to solve the nonlinear system shown in Equation 7.1, which is linearized as follows:

$$\boldsymbol{R}^{v+1} = \boldsymbol{R}^{v} + \left(\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{u}}\right)^{v} \delta \boldsymbol{u}^{v+1} = 0, \tag{7.2}$$

where v is the Newton-Raphson iteration level. Equation 7.2 can be written as follows:

$$\boldsymbol{J}^v \delta \boldsymbol{u}^{v+1} = -\boldsymbol{R}^v, \tag{7.3}$$

where the Jacobian matrix is defined as

$$\boldsymbol{J}^v = \left(\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{u}} \right)^v. \tag{7.4}$$

Every Newton-Raphson iteration requires solution of the linear system (Equation 7.3). The system of linear equations has the following characteristics:

1. Large - the model can have $O(10^6)$ cells with more than 10 components per cell;
2. Discontinuous coefficients;
3. Anisotropic coefficients;
4. Unstructured;
5. Non-symmetric;
6. Varying degrees of freedom;
7. Extended stencils (e.g., MPFA).

Therefore, linear solvers have a critical impact on the performance of a reservoir simulator. We are interested in robust and efficient general-purpose simulation. In this regard, GPRS is designed to serve as a platform for developing and testing various solvers and preconditioners. These current solver options include:

1. Full matrix direct solver;

2. Banded matrix direct solver;
3. BlitzPaK solver, which is a package of efficient solvers and preconditioners designed specifically for reservoir simulation [15];
4. Pointwise GMRES solver;
5. Blockwise GMRES solver;
6. BiCGstab solver [17].

Preconditioners are algorithms to speed up the solution of systems of linear equations. For large general linear systems in reservoir simulation, an iterative Krylov method (e.g., GMRES, BiCGstab) without a preconditioner is not a feasible method [42]. In reservoir simulation, most of the research effort on linear solvers focuses on the preconditioner rather than the Krylov projection method. GPRS provides various preconditioner options, which are listed here:

1. Diagonal preconditioner;
2. Block diagonal preconditioner;
3. ILU(0) preconditioner [4, 37];
4. True-IMPES CPR preconditioner [53] (AMG [49] serves in the first stage);
5. Quasi-IMPES CPR preconditioner [53] (AMG serves in the first stage);
6. AMG preconditioner;
7. True-IMPES CPR preconditioner (SAMG [51] serves in the first stage);
8. Quasi-IMPES CPR preconditioner (SAMG serves in the first stage);

9. SAMG preconditioner;
10. Block ILU(k) preconditioner;
11. Reservoir-facilities multilevel block ILU(k) preconditioner;
12. CPR preconditioner for advanced well models;

In this long list, the diagonal preconditioner is the simplest method. Each row and corresponding RHS element are scaled with the diagonal element. The block diagonal preconditioner scales the block rows and RHS elements with the diagonal blocks. Generally, these two preconditioners are not effective, and they will not be discussed further. The other preconditioners will be discussed in the following sections.

Although we have several solvers and a large number of preconditioners, not all combinations are possible. For example, direct solvers do not work with preconditioners. Some combinations are possible, but are not provided as standard options. For instance, the block GMRES solver (solver option 5) can work with the quasi-IMPES CPR preconditioner (preconditioner option 5). However, we have not encountered many cases for which the quasi-IMPES CPR preconditioner gives better performance than true-IMPES with a pointwise GMRES solver. All compatible solver and preconditioner combinations are listed in Table 7.1. In the table, P0 – P12 denote different preconditioner options (P0 means no preconditioner used), and S1–S6 represent different solver options.

7.2 Introduction to GMRES

The generalized minimum residual (GMRES) solver is widely used in our community. GMRES is also the primary solver option in GPRS. A brief introduction to GMRES

	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12
S1	x												
S2	x												
S3	x												
S4	x	x	x	x	x	x	x	x	x	x			
S5	x				x			x			x	x	x
S6	x	x	x	x	x	x		x	x				

Table 7.1: Compatible solvers and preconditioners in GPRS

is given here.

GMRES is an iterative algorithm used for solving linear system of equations in the form of $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ [43]. Here, $\boldsymbol{A}$ denotes a sparse invertible matrix, $\boldsymbol{b}$ is a right-hand-side (RHS) vector, and $\boldsymbol{x}$ is an unknown vector. For an $m \times m$ matrix, GMRES guarantees convergence to the exact solution within m iterations. But in most realistic problems, m is a very large number. For some cases in reservoir simulation, m can be more than ten million. In practice, GMRES converges after a small number of iterations when it is used in conjuction with a good preconditioner. The GMRES algorithm is listed in Algorithm 7.1:

Algorithm 7.1 Algorithm of GMRES [42]

1: *Compute* $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$, $\beta := \|\boldsymbol{r}_0\|_2$, and $\boldsymbol{v_1} := \boldsymbol{r}_0/\beta$
2: *Define the* $(m+1) \times m$ *matrix* $\bar{\boldsymbol{H}}_m = \{h_{ij}\}_{1 \le i \le m+1, 1 \le j \le m}$. *Set* $\bar{\boldsymbol{H}}_m = 0$.
3: *For* $j = 1, 2, ..., m$ *Do*
4: *Compute* $\boldsymbol{w}_j = \boldsymbol{A}\boldsymbol{v}_j$
5: *For* $i = 1, ..., j$ *Do*
6: $h_{ij} := (\boldsymbol{w}_j, \boldsymbol{v}_i)$
7: $w_j := \boldsymbol{w}_j - h_{ij}\boldsymbol{v}_i$
8: **end** *For*
9: $h_{j+1,j} = \|\boldsymbol{w}_j\|_2$. *If* $h_{j+1,j} = 0$ *set* $m := j$ *and go to* 12
10: $v_{j+1} = \boldsymbol{w}_j / h_{j+1,j}$
11: **end** *For*
12: *Compute* $\boldsymbol{y}_m$ *the minimizer of* $\|\beta \boldsymbol{e}_1 - \bar{\boldsymbol{H}}_m \boldsymbol{y}\|_2$ and $\boldsymbol{x}_m = \boldsymbol{x}_0 + V_m \boldsymbol{y}_m$

From the algorithm, we notice that GMRES is a general solver, which has no specific requirements for the type of matrix or its data structure. It only requires that a matrix-vector multiplication function be available (step 1 and step 4 in the algorithm). The independence from the matrix data structure, or format, gives us freedom to design and implement data structures that are specialized for reservoir simulation. Two new matrix data structures specially designed for reservoir simulation were presented in Chapter 3.

GPRS provides both pointwise and blockwise GMRES solvers (solver options 4 & 5). These two options use exactly the same implementation of GMRES. The difference is the type of matrices and their matrix-vector operations. The pointwise GMRES solves matrices in the CRS (compressed row storage) format (see Sparselib++ [37]), which is a pointwise matrix format that can be used for structured and unstructured systems. We employ our MLSB (multilevel sparse block) matrix when using blockwise GMRES. The definition and matrix-vector operation for the MLSB matrix format are given in Chapter 3.

Generally, the blockwise GMRES solver has better computational performance, because the matrix-vector operation for block matrices benefits from better cache utilization. Besides this, the matrix updating cost is much lower, because the new matrix data structure avoids the complex indexing operations in the matrix filling function. Some performance comparisons between the pointwise and blockwise GMRES are provided later.

GPRS also provides a BiCGstab solver (solver option 6). Later, a benchmark case is simulated with both GMRES and BiCGstab solvers. Based on this, and several other cases, we have found that the performance of GMRES is usually better than that of BiCGstab. Generally, the GMRES solver has about a 10-20% speedup over the BiCGstab solver. Orthogonal minimum residual (ORTHOMIN) used to be the

most prevalent linear solver in reservoir simulation, and is still the primary choice in the $Eclipse^{TM}$ simulator. A recent study showed that GMRES can be a better option than ORTHOMIN for black-oil reservoir simulation [29].

7.3 Single-stage Preconditioners

The convergence rate of iterative linear solvers depends highly on the condition number of the matrix to be solved as well as the detailed character of the eigenvalue spectrum. Preconditioners are used to reduce the matrix condition number and speed up the convergence of iterative solvers. Some stand-alone algorithms, e.g., Incomplete Lower Upper (ILU) factorization, can be directly applied as preconditioners. These preconditioners are so-called single-stage preconditioner. Sometimes, several algorithms are combined together to form a multistage preconditioner. Multistage preconditioners have some additional overhead, but they may achieve much better numerical performance. In this section, we will introduce the basic concept of a preconditioner (left and right) and then discuss the single-stage preconditioner options in GPRS.

7.3.1 Introduction

For a given matrix $\boldsymbol{A}$, a preconditioner $\boldsymbol{M}$ is a matrix satisfying the following two criteria [16]:

- $\boldsymbol{M}^{-1}\boldsymbol{A}$ should have a much smaller condition number than that of $\boldsymbol{A}$. In other words, $\boldsymbol{M}$ should be similar to $\boldsymbol{A}$ in some way. Of course $\boldsymbol{M} = \boldsymbol{A}$ is a perfect preconditioner, since the condition number of $\boldsymbol{M}^{-1}\boldsymbol{A}$ is unity. However, this

'perfect' preconditioner violates the second criteria.

- $\boldsymbol{M}^{-1}$ should be cheap to compute. Typically, instead of getting the inverse of $\boldsymbol{M}$, we requires that the linear system of $\boldsymbol{M}\,\boldsymbol{x} = \boldsymbol{b}$ can be solved cheaply, where $\boldsymbol{b}$ is a given RHS vector. In other words, we need to provide an efficient solver for our preconditioning matrix $\boldsymbol{M}$.

The preconditioner defined above is a left preconditioner, because $\boldsymbol{M}$ is applied to the left side of matrix $\boldsymbol{A}$. For a general sparse matrix system:

$$\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}, \tag{7.5}$$

it would be more efficient to solve an equivalent linear system with a much smaller condition number [42]:

$$\boldsymbol{M}^{-1}\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{M}^{-1}\boldsymbol{b}. \tag{7.6}$$

As discussed before, Krylov subspace solvers (e.g., GMRES, BiCGstab) are based on matrix-vector multiplication. Hence we do not need to compute $\boldsymbol{M}^{-1}\boldsymbol{A}$ explicitly. We can simply provide an operation for $\boldsymbol{M}^{-1}\boldsymbol{A}$:

$$\boldsymbol{w} \leftarrow \boldsymbol{M}^{-1}\boldsymbol{A}\,\boldsymbol{v}, \tag{7.7}$$

where $\boldsymbol{v}$ is a given vector and $\boldsymbol{w}$ is the result of the operation. Equation 7.7 with a left preconditioned Krylov subspace methods is performed in two steps. The first step is a basic matrix-vector multiplication:

$$\boldsymbol{v}^* \leftarrow \boldsymbol{A}\,\boldsymbol{v}, \tag{7.8}$$

where $\boldsymbol{v}^*$ is an intermediate vector. Then, Equation 7.7 can be rewritten as:

$$\boldsymbol{w} \leftarrow \boldsymbol{M}^{-1}\boldsymbol{v}^*. \tag{7.9}$$

The second step (Equation 7.9) is a preconditioner call, in which we solve the linear system:

$$\boldsymbol{M}\,\boldsymbol{w} = \boldsymbol{v}^*. \tag{7.10}$$

Then the vector $\boldsymbol{w}$ of Equation 7.7 is obtained. Thus, each iteration in the preconditioned GMRES calls the preconditioner once.

A right preconditioner is performed in a similar manner, but the preconditioning matrix $\boldsymbol{M}$ is applied in the right side of matrix $\boldsymbol{A}$. Instead of solving Equation 7.5, we solve an equivalent equation:

$$\boldsymbol{A}\boldsymbol{M}^{-1}(\boldsymbol{M}\,\boldsymbol{x}) = \boldsymbol{b}. \tag{7.11}$$

The above equation can be separated into two steps. In the first step, the iterative solver solves a linear system shown in Equation 7.12:

$$\boldsymbol{A}\boldsymbol{M}^{-1}\,\boldsymbol{y} = \boldsymbol{b}, \tag{7.12}$$

where

$$\boldsymbol{y} = \boldsymbol{M}\,\boldsymbol{x}. \tag{7.13}$$

For this linear system with a small condition number, $\boldsymbol{A}\boldsymbol{M}^{-1}$, Equation 7.12 will be solved until convergence. Each matrix-vector operation involves one preconditioner

call and one normal matrix-vector (MV) operation involving $\boldsymbol{A}$, which is in the reverse order compared to left preconditioning. As a result of the first step, the interim vector $\boldsymbol{y}$ is obtained. In the second step, one more preconditioner call solves for the original unknown vector $\boldsymbol{x}$

$$\boldsymbol{M}\,\boldsymbol{x} = \boldsymbol{y}. \tag{7.14}$$

In the following sections, we will discuss several important single-stage preconditioners used in GPRS.

7.3.2 Incomplete LU Factorization (ILU)

In linear algebra, LU factorization is the process of representing a matrix $\boldsymbol{A}$ as a product of a lower triangular matrix, $\boldsymbol{L}$, and an upper triangular matrix, $\boldsymbol{U}$,

$$\boldsymbol{A} = \boldsymbol{L}\,\boldsymbol{U}. \tag{7.15}$$

For an $n \times n$ matrix, the LU factorization requires approximately $\mathcal{O}(n^2)$ operations with proper ordering, which is not competitive with iterative solvers [21]. Moreover, the LU factorization of a sparse matrix does not necessarily result in sparse matrices. So, LU factorization is not a practical solver option for large sparse matrices. However, incomplete LU (ILU) factorization is used effectively as a preconditioner in many iterative solvers [42]. In this section, we discuss several ILU preconditioners used in GPRS.

Pointwise ILU

Incomplete LU (ILU) factorization is one of the most popular preconditioner families. Some non-zero elements in the L and U factors are ignored to reduce the cost and the number of fill-ins. ILU has many varieties based on the level of fill-in [42]. Among them, no fill-in ILU, ILU(0), is the simplest one. In the ILU(0) factorization, the lower and upper triangular matrices only keep non-zero elements, whose positions have non-zero elements in the original matrix. Therefore, if the lower and upper triangular matrices of ILU(0) are overlapped together, we get a matrix with the same non-zero structure as the original matrix, except that both lower and upper matrices have diagonal elements. The diagonal elements in the lower triangular matrix are all '1's. Figure 7.1 shows a matrix and its ILU(0) factorization results. The ILU(0) algorithm can be performed using the storage of the original matrix; no significant additional memory is required. The two resulting L and U matrices are stored in place, and the diagonal elements of the lower triangular matrix are not stored.

ILU(0) is a very simple and fast factorization. As a special case, ILU(0) of a tridiagonal matrix is a complete factorization. Tridiagonal matrices often arise from simple discretization of one-dimensional problems. However, generally the ILU(0) approximation to the original matrix can be very poor. In order to improve the accuracy, some ILU factorization algorithms with fill-in are developed, e.g., ILU with threshold (ILUT) [41] and ILU with fill-in level k (ILUK) [42]. The more fill-in, the more time the factorization takes. It is a trade-off between accuracy and speed.

There are several open-source high-performance pointwise ILU preconditioners provided by software packages. The most popular ones are from SparseLib++ [37] and SPARSKIT [40]. SparseLib++ provides an ILU(0) preconditioner, which is the version used in GPRS (preconditioner option 3). SPARSKIT provides a large set of

ILU family preconditioners, including ILU(0) and many other fill-in ILU preconditioners. All of these ILU preconditioner packages require matrices to be stored in a specific pointwise compressed matrix format.

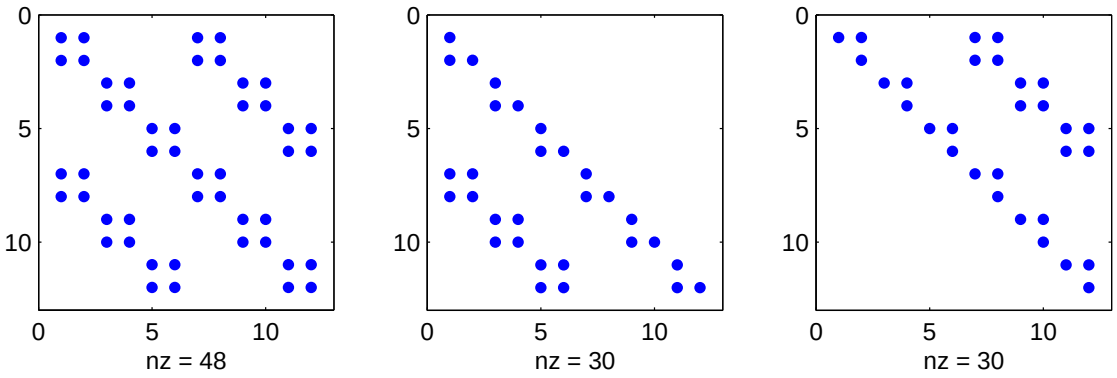


Figure 7.1: Distribution of non-zero elements in ILU(0) factorization

Blockwise ILU

In Chapter 3, we discussed the block compressed row sparse (BCRS) matrix format. BCRS includes a compressed row sparse matrix of pointers and the Jacobian entries in a special array form (Figure 3.7 and 3.8). Based on this data structure, we implemented a blockwise incomplete LU factorization preconditioner with no fill-in, BILU(0). The BILU(0) algorithm is the same as that of ILU(0), but all algebraic operations in ILU(0) are mapped into small matrix operations for BILU(0). Our BILU(0) preconditioner is fully compatible with the Jacobian matrices associated with the Adaptive Implicit Method (AIM). BILU(0) can handle matrices with various block sizes in the diagonal and off-diagonals.

The first matrix shown in Figure 7.1 is composed of a number of 2×2 blocks. This kind of matrix is very common in two-component black-oil reservoir simulation.

The other two matrices in the same figure are the results of ILU(0) factorization. Again, by definition, the diagonal elements in the lower-triangle matrix (the second matrix in Figure 7.1) are all '1's. The same matrix and its BILU(0) factorization results are shown in Figure 7.2. The second and third matrices are lower-triangle and upper-triangle matrices, respectively, from a block-based viewpoint. The diagonal blocks of the lower-triangle matrix are all 2×2 identity matrices.

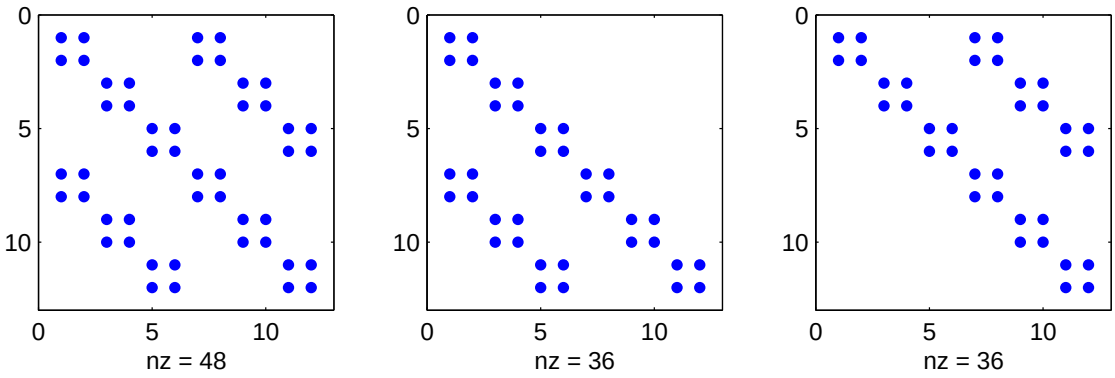


Figure 7.2: Distribution of non-zero elements in BILU(0) factorization

If we compare Figure 7.1 and Figure 7.2, it appears that BILU(0) has more non-zeros elements than ILU(0). However, this is not the case, because the diagonal blocks of the lower-triangle matrix are all 2×2 identity matrices. They have only '1's on the diagonal.

We define the number of blocks on the diagonal of the original matrix as n_b ($n_b = 6$ for Figure 7.1). The upper-triangle matrix of BILU(0) has n_b more non-zeros than that of ILU(0). These non-zero elements are located at the first super diagonal of the upper-triangle block matrix. The lower-triangle block of BILU(0) has n_b fewer non-zeros compared to that of ILU(0), since all the elements on the subdiagonal are zeros. Therefore, the overall non-zeros of BILU(0) and ILU(0) are the same. If the block size is 3×3 or more, the non-zeros are also the same in both ILU(0) and BILU(0).

Based on the same data structure, blockwise ILU with fill-in level k, BILU(k), is developed and integrated into GPRS [28]. The BILU(k) preconditioner has fill-in blocks and offers better accuracy. BILU(k) is also more expensive compared to BILU(0). Comparisons of BILU(k) and BILU(0) and the application of BILU(k) in GPRS will be investigated later.

Comparison between Pointwise ILU & Blockwise ILU

The criteria to evaluate a solver or preconditioner are its robustness, accuracy, efficiency and range of applicability. The performance of ILU and BILU are investigated with respect to these four aspects. Two no-fill ILU preconditioners, BILU(0) and ILU(0) are used as reference. Some analysis is covered in the following discussion and some examples are provided in later sections.

The stability of the BILU(0) preconditioner is better than that of ILU(0). ILU(0) without pivoting (pivoting could be an expensive operation for compressed sparse matrices) requires all the diagonal elements to be non-zero. Zero, or nearly-zero diagonal elements are common in reservoir simulation matrices. BILU(0) only requires that all diagonal blocks are invertible, which is a less stringent requirement.

Matrices from reservoir simulation can have large blocks, because the reservoir cells usually have multiple primary variables and governing equations. These blocks can be seen as small dense matrices. This is a property of the Jacobian matrix, no matter which format the matrix is stored in. The typical storage options are a pointwise method (e.g., the CRS format) or a blockwise method (e.g., the MLSB format). Due to the upwinding scheme in simulation, some of the elements in those small blocks (generally, the off-diagonal blocks) are zeros, but these zeros are still stored. The positions and the total number of those stored zero elements are dynamic. It is not

a feasible option to identify the zero elements in the blocks and squeeze them out of the matrix for every Newton iteration in a real simulation. It simply increases the computational cost dramatically with trivial gain (slightly reduced memory cost).

When the standard pointwise ILU(0) algorithm is applied to block matrices (although stored in pointwise matrix format), mathematically it is equivalent to the BILU(0) algorithm as long as both of them exist. If both:

$$\mathbf{ILU0}(\boldsymbol{A}) = \boldsymbol{L}_p \boldsymbol{U}_p \tag{7.16}$$

$$\mathbf{BILU0}(\boldsymbol{A}) = \boldsymbol{L}_b \boldsymbol{U}_b, \tag{7.17}$$

exist, we can always find an invertible matrix $\boldsymbol{P}$, which can make:

$$\boldsymbol{L}_p = \boldsymbol{L}_b \boldsymbol{P} \tag{7.18}$$

$$\boldsymbol{U}_p = \boldsymbol{P}^{-1} \boldsymbol{U}_b. \tag{7.19}$$

Again, this is only true when matrix $\boldsymbol{A}$ is a block matrix. In this circumstance, there is no accuracy difference between ILU(0) and BILU(0), except truncation errors in computation.

BILU(0) may be faster compared to ILU(0), because it benefits from better cache utilization. Cache is the memory component embedded in a CPU, which lies between the normal memory and the processor. Any data requested by a processor has to be fetched from memory into cache and then passed to the processor. A single fetch loads a series of contiguous data starting with the requested data item. The data transfer speed from main memory to cache is much slower than the data transfer speed between cache and the processor [27].

The operations in a block solver are based on block data. For example, one of the

steps in BILU(0) is to calculate the inverse of all diagonal blocks and store them in the original space. In our block matrix format, these related elements, e.g., elements in diagonal blocks, are continuous in memory. When a block solver fetches data from memory to cache, a single fetch loads in a series of contiguous related elements. This significantly reduces the number of fetches (slow operation) and speeds up the overall computation. Typically, the advantage of BILU(0) over ILU(0) is more significant for compositional models with large numbers of components, because of the larger size of the data blocks.

The ILU family (both pointwise and blockwise) of preconditioners is quite general. They can be applied to many types of matrices without knowing the details of the problem that the matrix represents. However, the performance of ILU preconditioners does depend on the properties of the linear system. The preconditioners converge faster for more diagonally dominant matrices, which are associated with smaller timestep in flow simulation. ILU preconditioners also work better for matrices that are obtained from near-hyperbolic equations. For tightly coupled systems, like the pressure equations in reservoir simulation, the ILU family of preconditioners may not be the best option. The nature of the system has a strong impact on the numerical performance of a preconditioner. In the next section, we explore some two-stage preconditioners, which combine different single-stage preconditioners. Generally, two-stage preconditioners require knowledge about the physics of the system; however, when designed properly, they can be much more efficient than single-stage approaches.

7.3.3 Algebraic Multigrid (AMG)

For matrices from near-elliptic problems, e.g., the pressure system in reservoir simulation, the ILU preconditioned Krylov subspace solvers may stagnate after a few iterations. The errors in problems of this type usually spans a large frequency spectrum. The local error (high frequency error) can be successfully removed by the solver during the first few iterations, but it is very difficult for ILU preconditioned Krylov subspace solvers to remove the low frequency error [10].

Multigrid methods are ideal linear solvers for elliptic problems [5]. In multigrid methods, a hierarchy of coarse grids is generated on top of the original grid. Multigrid can effectively remove the low frequency errors in near-elliptic problems by employing both simple relaxation schemes (e.g., Gauss-Seidel) and coarse-grid correction. Algebraic Multigrid (AMG) is one kind of multigrid methods [49]. AMG does not rely on geometry information, but obtains the information directly from the matrix. This makes it an ideal "plug-in" solver, which greatly facilitates its application. In most simulation fields, including reservoir simulation, the gridding module and the solver module are independent. It is difficult to exchange information between these two modules. Beside that, several characteristics of the reservoir simulation models, such as discontinuity and anisotropy in the coefficients, and generally unstructured grid make it even harder to generate coarse-grid operators based on geometric information.

In reservoir simulation, IMPES (IMplicit Pressure Explicit saturation) is one of the most important formulations [2]. In the IMPES scheme, pressure is the only implicit variable in each gridblock. The other primary variables (saturations and component mole fractions) are treated explicitly [2]. This results in a near-elliptic pressure system. AMG is the most suitable solver for this linear system. In GPRS, AMG is used as a "plug-in" solver for the pressure matrix. GPRS incorporates two

AMG packages. One is the open-source version AMG developed in 1991 [38], and the other one is a commercial package, SAMG, from SCAI [51].

AMG has strict requirements on the characteristics of the pressure matrix. The most preferable matrix is an M matrix, which has nonpositive off-diagonal elements, and all eigenvalues with a nonnegative real part [50]. Only the matrices from elliptic or near-elliptic systems can be solved effectively by AMG. Otherwise, it may run into problems. Because of this, AMG is usually used as a preconditioner for iterative solvers rather than as a solver by itself. In GPRS, AMG preconditioned GMRES is the primary linear solver option for IMPES simulation.

7.4 CPR for Reservoir Models with StdWells

The equations and unknowns in reservoir simulation have mixed properties. They have both elliptic (pressure equations) and hyperbolic (advection equations) parts. In FIM simulation, all primary variables are treated implicitly, and Newton-Raphson method is employed as the nonlinear solver. Hence, we must solve an FIM Jacobian matrix every iteration. When we solve the Jacobian with linear solvers, the error contains both low frequency components (mainly through pressure) and high frequency components (mainly through saturations and other variables). AMG cannot solve the matrix, because the matrix arises from a mixed system, rather than an elliptic system. The ILU family of preconditioners can be used as iterative-solver accelerators to solve the matrix; however, the performance is usually poor. In the first few iterations, the high frequency errors are usually reduced successfully by ILU, but the low frequency errors persist as the iterations continue. [10].

In order to handle the Jacobian matrices in reservoir simulation, which have a

mixed character, a two-stage preconditioner, the constrained pressure residual (CPR) method, was proposed by Wallis et al. [53, 54]. In the first stage, the pressure system is obtained from the fully coupled Jacobian using an efficient algebraic reduction scheme, which mimics the steps associated with constructing the pressure equation of the IMPES formulation. This pressure system preserves the coupling of the well-reservoir system, and is solved with AMG. The low frequency errors are resolved in this stage. In the second stage, an ILU preconditioner is usually applied to the full Jacobian matrix, which removes the high frequency (local) errors.

The general form of the two-stage preconditioner can be written as [10]:

$$M_{1,2}^{-1} = M_2^{-1}[I - AM_1^{-1}] + M_1^{-1}, \tag{7.20}$$

where $M_{1,2}$ denotes the two-stage preconditioner, M_2 is the second stage preconditioner, I is an identity matrix, A is the matrix to be solved, M_1 is the first stage preconditioner. In the following sections, we discuss the implementation details of the two stages. A few large-scale, highly heterogeneous, unstructured models are used to demonstrate the performance of the CPR preconditioner. Later, CPR is extended to handle reservoir models with advanced well models. Therefore, it is important to have a detailed investigation of CPR here.

7.4.1 First Stage: Pressure System

In the first stage of CPR, we need to construct a pressure matrix and solve it efficiently. There are two approaches to get the pressure matrix from a fully implicit system in reservoir simulation, which are plotted in Figure 7.3. Both of them start from a fully coupled nonlinear equation set and end up with a pressure matrix. We will use a reservoir with n grids and oil-water fluids as an example to describe the procedure.

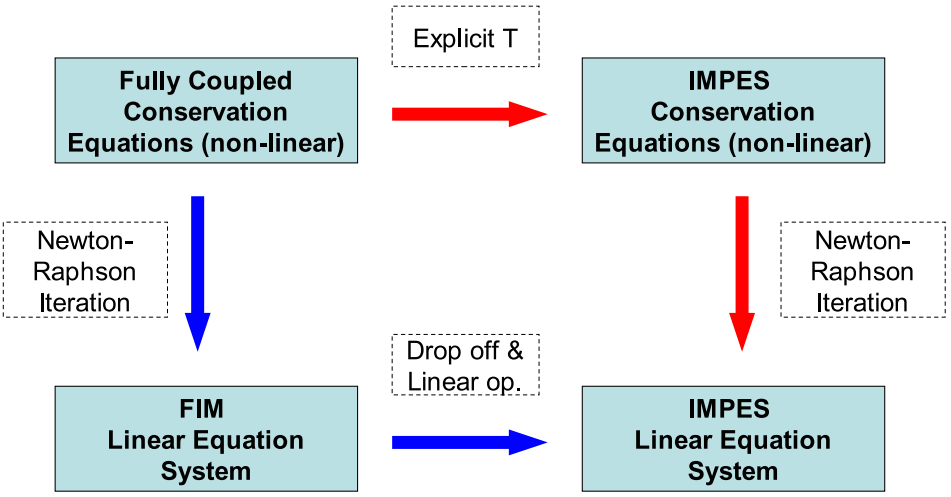


Figure 7.3: Relation of FIM and IMPES formulations

The first approach is the standard IMPES reduction (the red path in the plot). In the first step of IMPES reduction, the transmissibility terms are treated explicitly, i.e., the flux term does not depend on s^{n+1}. Therefore, s^{n+1} only exists in the accumulation term of the conservation equations. By combining the oil and water conservation equations of each cell (this is a linear operation), s^{n+1} can be completely removed. Eventually, we get a total of n IMPES nonlinear conservation equations, in which p_i^{n+1} are the unknowns. In the second step of IMPES reduction, Newton-Raphson method is used to linearize the n nonlinear equations and an $n \times n$ pressure matrix is generated for each iteration [1].

The blue path in Figure 7.3 shows another two-step approach to generate a pressure matrix. In the first step, the $2n$ conservation equations are linearized by the Newton-Raphson method. One $2n \times 2n$ matrix is generated for each iteration. The matrix is the FIM Jacobian. In the second step, some saturation derivatives in the FIM matrix are ignored and some derivatives are eliminated by linear operations. Then we get the desired pressure matrix. The process of the second step will be

discussed in detail later.

The two approaches lead to pressure systems that are quite similar. Both of them contain important information about the fully coupled system. Typically, CPR serves as the preconditioner for solving FIM matrices. Since the FIM matrix has been built for the linear solver, the "blue path" is the better approach to decouple the pressure matrix. In the second step of the 'blue path', two options, namely, true-IMPES and quasi-IMPES, can be used to perform the reduction of the FIM Jacobian into an IMPES-like matrix.

True IMPES Reduction

We select a block row in the FIM Jacobian matrix, which corresponds to the oil and water conservation equations in one cell. These two rows may contain one 2×2 diagonal block and several off-diagonal blocks. We plot the diagonal block of cell i and an off-diagonal block representing the flux between cell i and cell j as follows:

$$\begin{pmatrix} A_{op} + F_{op} & A_{os} + F_{os} \\ A_{wp} + F_{wp} & A_{ws} + F_{ws} \end{pmatrix}_{i,i} \qquad \begin{pmatrix} F^*_{op} & F^*_{os} \\ F^*_{wp} & F^*_{ws} \end{pmatrix}_{i,j},$$

where subscripts o, w, p, s represent oil, water, pressure, and saturation, respectively. A_{op} is the derivative of the accumulation term of the oil equation with respect to pressure, F_{ws} is the derivative of the water flux term between cell i and cell j with respect to the saturation of cell i. F^*_{ws} is the derivative of the same water flux term with respect to the saturation of neighboring cell j. We see that the diagonal block contains contributions from both accumulation and flux terms. The off-diagonal entries depend on the unknowns in the specific off-diagonal gridblock only. In the 'true-IMPES' reduction approach of an FIM Jacobian, we treat the saturation in the

flux terms explicitly, but we use the the saturations of the last iteration rather than the ones from the last timestep. This step is performed algebraically to remove all the derivatives of the flux terms with respect to saturation. Then the above diagram can be simplified as follows:

$$\begin{pmatrix} A_{op}+F_{op} & A_{os} \\ A_{wp}+F_{wp} & A_{ws} \end{pmatrix}_{i,i} \qquad \begin{pmatrix} F^*_{op} & 0 \\ F^*_{wp} & 0 \end{pmatrix}_{i,j} .$$

The dropped derivatives allow us to decouple the pressure system. Linear operations can be used to eliminate the saturation derivative in the diagonal block. The result is written as follows:

$$\begin{pmatrix} J_{op}-A_{os}A_{ws}^{-1}J_{wp} & 0 \\ J_{wp} & A_{ws} \end{pmatrix}_{i,i} \qquad \begin{pmatrix} F^*_{op}-A_{os}A_{ws}^{-1}F^*_{wp} & 0 \\ F^*_{wp} & 0 \end{pmatrix}_{i,j} ,$$

where J denotes $A+F$. To be consistent, the same linear operation is applied on the RHS vector as well. We can see that the first row corresponds to a pressure equation, since it contains no saturation derivatives. The same linear operation is applied to the cells (block rows) one by one, and we obtain a pressure matrix from the FIM Jacobian. The procedure is valid for an arbitrary number of conservation equations per cell.

The pressure matrix constructed with this algebraic reduction is very similar to the one in the IMPES formulation. The only difference is related to the saturation. In the algebraic construction of the 'true-IMPES' pressure matrix, the saturations and related terms are treated with a one-iteration lag, because the last iteration values are used to build the FIM matrix. However, for the matrix from the nonlinear IMPES formulation, the saturations are fixed at the previous timestep.

Quasi-IMPES Reduction

The quasi-IMPES algebraic reduction starts from the FIM matrix:

$$\begin{pmatrix} A_{op}+F_{op} & A_{os}+F_{os} \\ A_{wp}+F_{wp} & A_{ws}+F_{ws} \end{pmatrix}_{i,i} \qquad \begin{pmatrix} F^*_{op} & F^*_{os} \\ F^*_{wp} & F^*_{ws} \end{pmatrix}_{i,j}.$$

In quasi-IMPES, we eliminate the saturation derivatives in the diagonal block by a linear operation, which can be written as

$$\begin{pmatrix} J_{op}-J_{os}J_{ws}^{-1}J_{wp} & 0 \\ J_{wp} & J_{ws} \end{pmatrix}_{i,i} \qquad \begin{pmatrix} \boldsymbol{F_1} & \boldsymbol{F_2} \\ F^*_{wp} & F^*_{ws} \end{pmatrix}_{i,j},$$

where:

$$\boldsymbol{F_1} = F^*_{op} - J_{os}J_{ws}^{-1}F^*_{wp}, \tag{7.21}$$

$$\boldsymbol{F_2} = F^*_{os} - J_{os}J_{ws}^{-1}F^*_{ws}. \tag{7.22}$$

After this step, $\boldsymbol{F_2}$ is ignored, and we get a decoupled system as follows:

$$\begin{pmatrix} J_{op}-J_{os}J_{ws}^{-1}J_{wp} & 0 \\ J_{wp} & J_{ws} \end{pmatrix}_{i,i} \qquad \begin{pmatrix} \boldsymbol{F_1} & 0 \\ F^*_{wp} & F^*_{ws} \end{pmatrix}_{i,j}.$$

The first row of above block row is the pressure equation for cell i. Since $\boldsymbol{F_2}$ is dropped, we actually do not need to calculate Equation 7.22.

Remarks on the Two Reduction Schemes

The true-IMPES and the quasi-IMPES reductions take similar steps but in different orders. The true-IMPES reduction ignores some saturation derivatives first and then conducts linear operations to remove the remaining saturation derivatives. The quasi-IMPES conducts linear operations first and then ignore saturation derivatives afterward. The dropped terms in the true-IMPES reduction can be explained as explicit treatment of saturation in the flux terms. But the physical meaning of the dropped terms in the quasi-IMPES is not clear. Based on a large number of simulations, we have found that the true-IMPES reduction has consistently turned out to be the better choice to decouple the pressure system. This is related to how the resulting pressure system is solved as well, which is discussed later.

Pressure System and AMG

The above reduction process can be represented by the following expression:

$$\boldsymbol{A}_p = \boldsymbol{R}\,\boldsymbol{A}_{FIM}\,\boldsymbol{P}, \tag{7.23}$$

where $\boldsymbol{A}_{FIM}$ is the fully implicit matrix, $\boldsymbol{R}$ is a row operation matrix, and $\boldsymbol{P}$ is a column operation matrix, and $\boldsymbol{A}_p$ is an IMPES-like pressure matrix.

AMG is the best method for this kind of near-elliptic system. There are two options to solve the matrix. One is to use AMG as a solver directly working on the pressure matrix. An alternative is to use GMRES as the solver and AMG as the preconditioner. Many cases have been tested with these two options. The results show that using AMG directly is faster, while GMRES with AMG has slightly better stability in very special cases. The investigation also shows that the pressure system

in the first stage of CPR need not to be solved very accurately. One "V-cycle" of AMG gives the best overall performance [25]. This conclusion is drawn based on the typical linear accuracy requirement of reservoir simulation, which is to reduce the residual by three to six orders of magnitude.

The solution vector from AMG, $\boldsymbol{x}_p$, is in the vector space of the IMPES system. A prolongation operator is needed to project the vector into the vector space of the FIM system:

$$\boldsymbol{x}_1 = \boldsymbol{P}\, \boldsymbol{x}_p, \tag{7.24}$$

where $\boldsymbol{x}_1$ is the solution vector of the first stage of CPR. Matrix $\boldsymbol{P}$ in Equation 7.24 is same as the one in Equation 7.23. For example, for a three-phase black-oil FIM model (i.e., three unknowns per cell) with cell-based ordering, $\boldsymbol{x}_p$ and $\boldsymbol{x}_1$ may look like:

$$\boldsymbol{x}_p = \{\delta p_1,\ \delta p_2,\ \delta p_3,\ ...,\ \delta p_n,\ \delta p^w\}^T, \tag{7.25}$$

$$\boldsymbol{x}_1 = \{\delta p_1, 0, 0,\ \ \delta p_2, 0, 0,\ \ \delta p_3, 0, 0,\ ...,\ \delta p_n, 0, 0,\ \ \delta p^w\}^T, \tag{7.26}$$

where δp_i is the Newton update of p_i (pressure of cell i), δp^w is the Newton update of well pressure. The prolongation process can also be done for the AIM scheme, in which the '0's will be filled in according to the implicit level of the cells.

7.4.2 Second Stage: Simple Preconditioning of the Full System

In the second stage of CPR, the ILU family of preconditioners is commonly used. The solution vector from the first stage is used to update the right-hand-side of the

second stage, $\boldsymbol{x}_1$. The expression can be written as:

$$\boldsymbol{M}_2 \, \boldsymbol{x}_2 = \boldsymbol{b} - \boldsymbol{A} \, \boldsymbol{x}_1, \tag{7.27}$$

where $\boldsymbol{M}_2$ stands for ILU preconditioner, $\boldsymbol{x}_2$ is the solution of the second stage of CPR. The final solution of CPR preconditioner, $\boldsymbol{x}_{_{CPR}}$, is the combination of results from both stages:

$$\boldsymbol{x}_{_{CPR}} = \boldsymbol{x}_1 + \boldsymbol{x}_2. \tag{7.28}$$

GPRS provides ILU(0), BILU(0), BILU(k) [28], and multilevel BILU(k) as stand-alone preconditioners. All of them can serve as the second stage preconditioner in CPR. More detailed investigation about the performance of the ILU family of preconditioners is given later.

7.4.3 Test Cases

In order to investigate the solver capabilities of GPRS, especially the block ILU and CPR preconditioners, we present some test examples. A large black-oil case was selected to validate GPRS and demonstrate its capabilities. A large compositional case is used to show the different performance of several preconditioning options in GPRS. The third case is for a fractured reservoir with highly unstructured grid and a compositional fluid.

Case 1: SPE10 - Comparative Solution Project

The 10^{th} SPE Comparative Solution Project is the latest comparative solution project organized by the Society of Petroleum Engineers (SPE) [12]. The case is designed for comparing upscaling approaches, and is also widely used as a benchmark case for

large-scale highly heterogeneous reservoir simulation.

The dimensions of the SPE10 model are $1200 \times 2200 \times 170$ ft. It has $60 \times 220 \times 85$ (1,122,000) gridblocks. "The top 70 ft (35 layers) represent the Tarbert formation, and the bottom 100 ft (50 layers) represent Upper Ness." [12] The reservoir model is shown in Figure 7.4. The porosity and permeability maps of the 5th layer (Tarbert formation) and the 40th layer (Upper Ness) are shown in Figure 7.5 and 7.6. The simulated period is 2000 days, with 0.74 pore volume injected (PVI). The total oil production and water cut of the first producer from GPRS are plotted with results from other simulators. From Figure 7.7 and 7.8, we can see that the GPRS results are in good agreement with industrial simulators for this large-scale case.

By using our CPR preconditioned GMRES solver, GPRS can finish the SPE10 case using a single 2.2 GHz CPU in about 7 hours. Detailed timing performance is listed in Table 7.2. The CPR preconditioned BiCGstab solver is used for the same case. Its performance is listed in Table 7.3. By comparing the two tables, we see that BiCGstab needs about 10% more time to finish the simulation. We also listed the timestep, Newton and linear solver iteration information of both GMRES and BiCGstab in Table 7.4. The results show small differences in the numbers of timesteps and Newton iterations. However, BiCGstab needs 30% more preconditioner calls to satisfy the same accuracy requirement. The preconditioner settings are identical in the two runs. This case is consistent with our observation that GMRES, is on average, a better solver than BiCGstab.

Overall, we see that the linear solver is the most expensive part for large-scale black-oil reservoir simulation. In the test case, the linear solver takes 86.5% of the total simulation time. Within the solver time, the CPR preconditioner accounts for 73.5%. The remainder is spent on the GMRES solver itself and some overhead. Within the preconditioning part, solving the pressure matrix is the most expensive

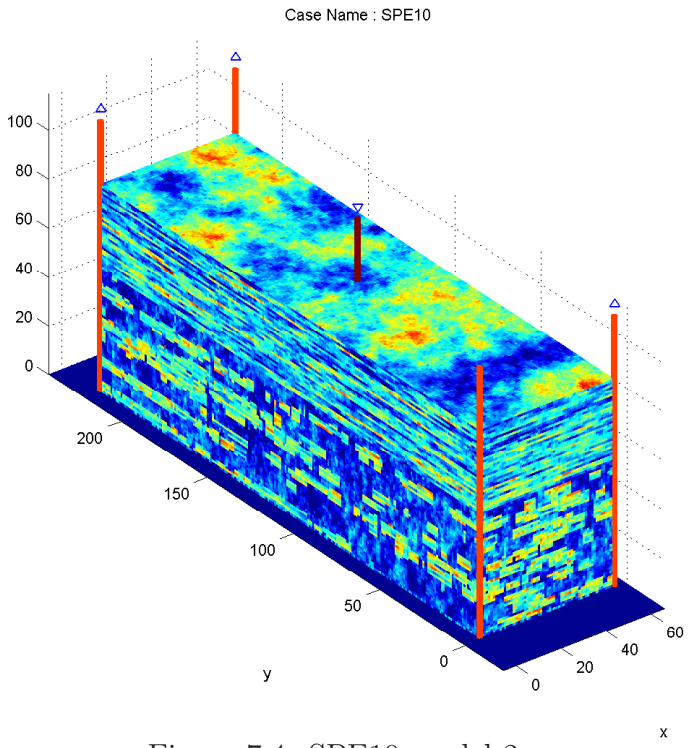


Figure 7.4: SPE10 model 2 case

operation, which takes 55.8% of the solver time.

Case 2: 9-component compositional simulation

A 9-component compositional case was designed to further test the performance of solvers and preconditioners. The reservoir model has $100 \times 100 \times 5$ gridblocks, and the scheme is fully implicit. So, we have 50,000 gridblocks and 9 equations and unknowns per block, which leads to 450,000 equations and unknowns for the reservoir model. One production well is completed in the reservoir. The system is undergoing primary depletion. Many solver and preconditioner combinations are tested to compare the performance. The results are listed in Table 7.5.

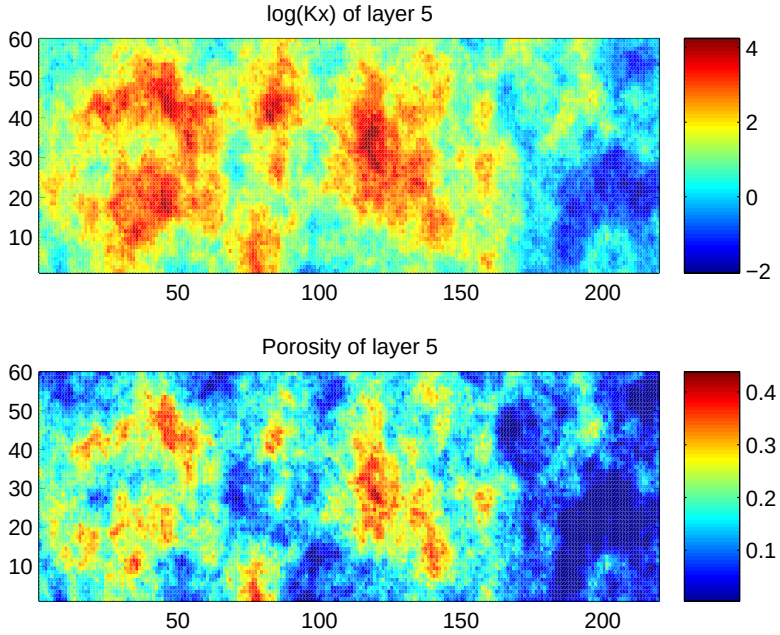


Figure 7.5: Permeability and porosity map of the 5th layer of SPE10 case

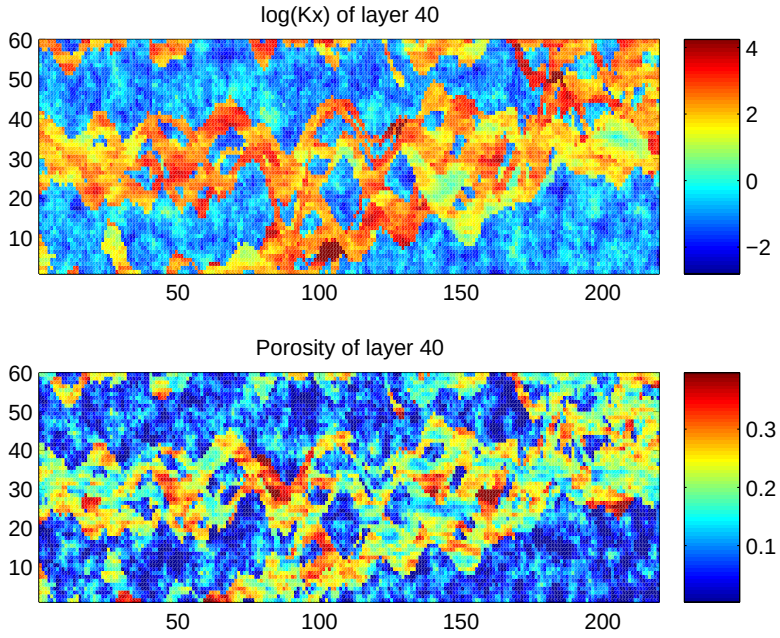


Figure 7.6: Permeability and porosity map of the 40th layer of SPE10 case

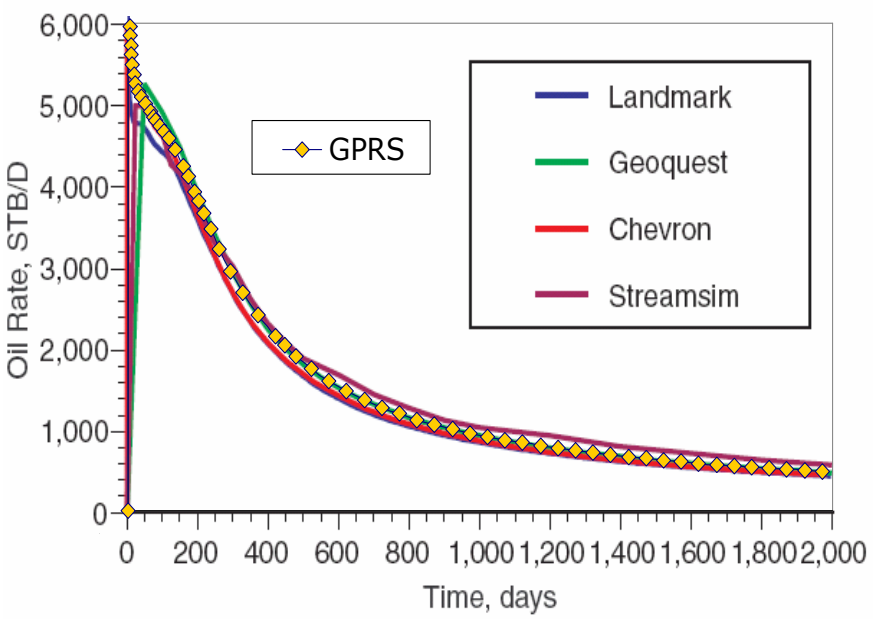


Figure 7.7: Oil rate of the four producers in SPE 10th case

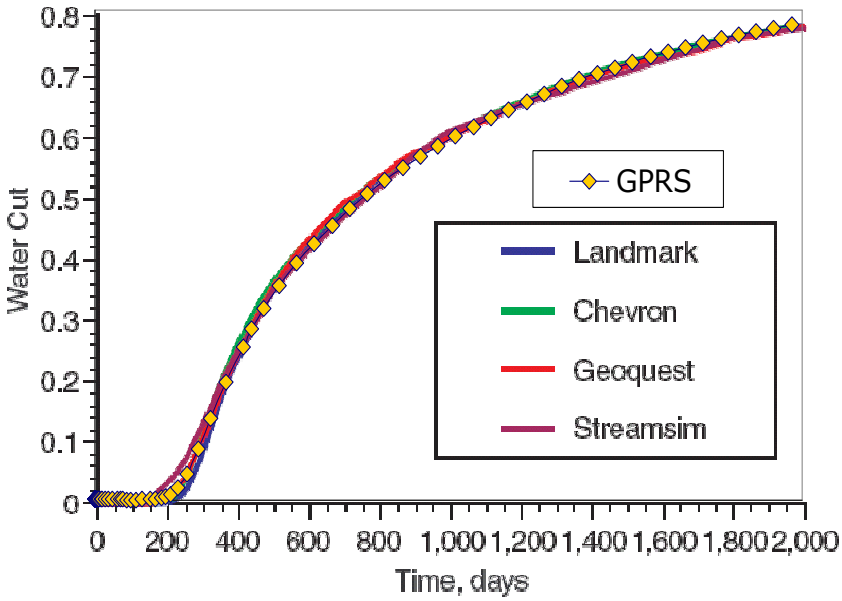


Figure 7.8: Water cut of the first producer in SPE 10th case

	Time (sec)	% of total time	% of solver time
Total running time	25406	100.0	N/A
Property calc. time	492	1.9	N/A
Linearization time	1059	4.2	N/A
Solver time	21973	86.5	100.0
GMRES/MV	5823	22.9	26.6
ILU factorization	2324	9.1	10.6
ILU solution	1057	4.2	4.8
Pressure matrix decoupling	501	2.0	2.3
Pressure solution	12268	48.3	55.8

Table 7.2: CPR preconditioned GMRES performance for SPE10 case

	Time (sec)	% of total time	% of solver time
Total running time	26973	100.0	N/A
Property calc. time	493	1.8	N/A
Linearization time	1053	3.9	N/A
Solver time	24050	89.2	100.0
GMRES/MV	5876	21.7	24.4
ILU factorization	2341	8.8	9.7
ILU solution	1320	4.9	5.5
Pressure matrix decoupling	513	1.9	2.1
Pressure solution	14009	51.9	58.2

Table 7.3: CPR preconditioned BiCGstab performance for SPE10 case

B-GMRES stands for blockwise GMRES and P-GMRES stands for pointwise GMRES. CPR-BILU(0) means BILU(0) serves as the second stage solver in the CPR preconditioner. The 'solver iteration' in the table refers to the total number of linear-solver iterations in the entire simulation. The GMRES solver calls the preconditioner once per iteration. Therefore, the number of the solver iterations is the same as the number of times that the preconditioner is called. This is not always the case, e.g., the BiCGstab scheme calls the preconditioner twice per iteration. The more accurate a preconditioner is, the less number of times it needs to be called. However, a

Solver	GMRES	BiCGstab
Preconditioner	CPR	CPR
Timesteps	82	82
Newton iterations	351	352
Preconditioner calls	3307	4346

Table 7.4: Comparison of GMRES and BiCGstab solvers for SPE10 case

No.	Solver	Preconditioner	Solver iteration	Solution time	Total time
1	P-GMRES	ILU(0)	2748	2676.4	3117.7
2	B-GMRES	BILU(0)	2748	2484.3	2926.6
3	B-GMRES	BILU(1)	1328	1696.4	2137.9
4	B-GMRES	BILU(2)	998	2182.8	2626.6
5	P-GMRES	CPR-ILU(0)	68	321.4	762.1
6	B-GMRES	CPR-BILU(0)	68	217.1	658.6
7	B-GMRES	CPR-BILU(1)	66	469.1	909.4

Table 7.5: GPRS timing performance of a 9-component case (*time in second)

more accurate preconditioner is not necessarily preferred. Typically, the accuracy of a preconditioner is correlated with computational cost. We need to carefully balance convergence rate with computational cost.

Option 1 in Table 7.5 shows the performance of pointwise GMRES with pointwise ILU(0) as a preconditioner. The options numbered 2, 3 and 4 use blockwise GMRES with different levels of blockwise ILU as a preconditioner. We can see that BILU(1) has the best performance among the three. BILU(2) provides more accurate factorization, hence it needs much less solver iterations. However, due to the high cost of level two factorization, its overall timing performance cannot beat that of BILU(1) [28]. Item 5 uses pointwise GMRES with the CPR preconditioner, in which pointwise ILU(0) serves as the second stage. Items 6 and 7 both use blockwise GMRES with the CPR preconditioner; note that different Blockwise ILU preconditioners are used for the second stage. From the performance, we can see that BILU(0) is

the best choice for the second stage of CPR. Only two solver iterations (or 3%) is reduced by increasing the level of BILU in the second stage from 0 to 1. We also observe that ILU(0) and BILU(0) are exactly the same in iteration numbers, and so are CPR-ILU(0) and CPR-BILU(0). We explained in the previous section that ILU(0) and BILU(0) have the same accuracy.

The most basic ILU(0) preconditioner is chosen as a base case in our comparison. The solver time speedup factors of the other preconditioners are shown in Figure 7.9. There are two key observations from the plot. The first one is that CPR clearly outperforms the ILU family of preconditioners. The other one is that BILU preconditioners give much better performance than their pointwise counterparts. The CPR preconditioner can achieve even better performance by using BILU in the second stage. Note that the combination of block solvers and CPR can give more than twelve fold speedup.

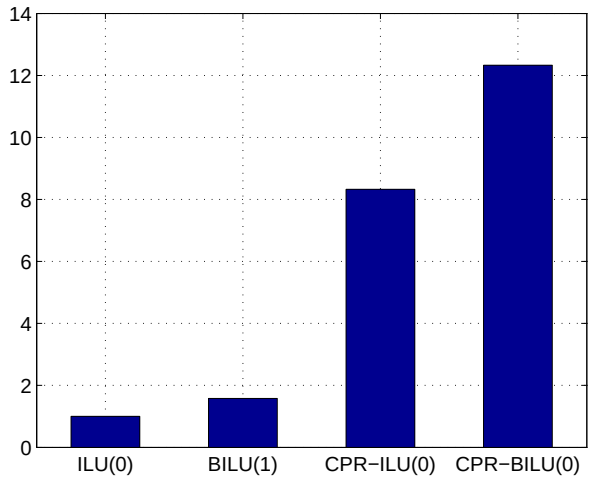


Figure 7.9: Speedup factors of preconditioners

In Table 7.6, the detailed solver timing performance of CPR-ILU(0) and CPR-BILU(0) are given. We can see that most of the gain in speed (86.3 of 104.3 sec)

Solver	P-GMRES	B-GMRES	-
Preconditioner	CPR-ILU(0)	CPR-BILU(0)	(Difference)
Solver time	321.4	217.1	+104.3
GMRES/MV	92.6	73.8	+18.8
(B)ILU factorization	175.1	88.8	+86.3
(B)ILU solution	15.7	17.6	-1.9
Pressure matrix decoupling	19.8	20.1	-0.3
Pressure solution	18.2	16.8	+1.4

Table 7.6: Comparison of pointwise and blockwise solvers (time in sec.)

comes from the ILU factorization. BILU(0) has much faster factorization speed than ILU(0). The BILU(0) algorithm we developed is based on our BCRS data structure discussed in Chapter 3. It benefits from better cache utilization. The rest of the gain comes from the GMRES solver itself. The matrix-vector operation based on the multilevel block matrix (discussed in Chapter 3) also has better cache utilization compared to traditional compressed matrix formats.

Case 3: 6-component compositional DFM (Discrete Fracture Model) simulation

A synthetic reservoir model was generated based on real data from a fractured carbonate reservoir [22]. The model contains 35 fractures and is discretized into 131,817 cells using a discrete fracture model (DFM) [26]. The model properties are highly heterogeneous with a matrix permeability of 0.1 md and a fracture permeability of 1,000,000 md. The matrix porosity is 0.25 and the fracture porosity is 1.0. The largest pore-volume of a gridblock is 1.24e+3 ft^3, and the smallest pore volume is 7.22e-9 ft^3. The reservoir fluid is described using 6 hydrocarbon components. An injector and a producer are drilled in two disconnected fractures located at opposite

corners of the reservoir model. The system is shown in Figure 7.10.

This case was chosen to show the performance of GPRS for a large-scale, highly heterogeneous compositional model with unstructured grid. Particularly, we are interested in the performance of the block solver and preconditioner. The case is run with both pointwise and blockwise GMRES solvers and CPR preconditioners. The performance information is listed in Table 7.7.

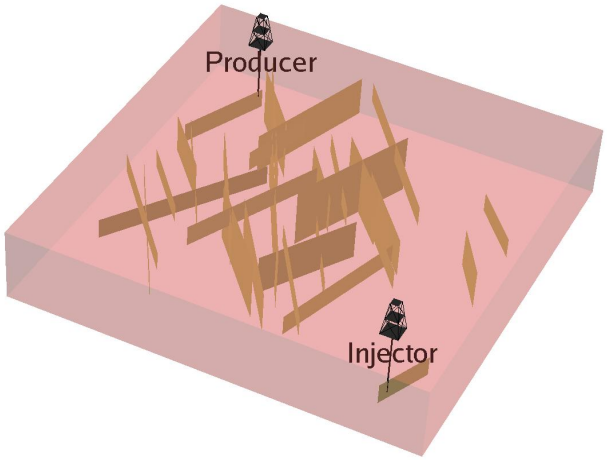


Figure 7.10: Reservoir model with 30 vertical fractures [22]

Solver	P-GMRES	B-GMRES	-
Preconditioner	CPR-ILU(0)	CPR-BILU(0)	(Difference)
Solver time (sec)	10442	6599	+3843
GMRES/MV time (sec)	2494	2213	+281
(B)ILU factorization (sec)	4566	676	+3890
(B)ILU solution (sec)	635	980	-345
Pressure matrix decoupling (sec)	432	422	+10
Pressure solution (sec)	2315	2308	+7
Memory cost (mega)	1396	1071	+325

Table 7.7: Comparison of pointwise and blockwise solvers for a DFM model with unstructured grid.

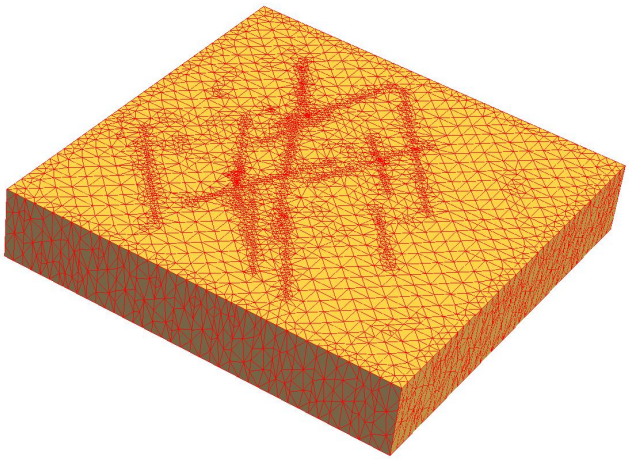


Figure 7.11: Unstructured grid conforming to fractures in Figure 7.10 [22]

For this case, the block solver and preconditioner were 37% faster than the pointwise one. The major gain comes from the BILU factorization, which has 6.8 times speedup. In addition, the matrix-vector multiplication also contributes to the overall speedup. The same as in Case 2, the solution time of pointwise ILU is slightly faster than the blockwise counterpart. Since the number of timesteps and Newton iterations are almost the same in both case, the time spent on pressure decoupling and solution are quite similar for both options. We note that this test case cannot be solved in a reasonable time if CPR is not used.

Summary

The CPR method has been proven to be an excellent preconditioner for FIM simulation of complex reservoir models and the standard well model. The entire system is closely coupled through the pressure. CPR requires some additional cost to construct the pressure equation from the full system, which is then solved with an appropriate preconditioner, such as AMG. After the residual is updated using the pressure

solution, which is reflected in the RHS, the system is nearly decoupled. The ILU preconditioner is quite effective in dealing with the remaining errors in the global solution. Block second-stage preconditioners further improve the performance of CPR.

7.5 CPR for Reservoir Models with Multisegment Wells

In Chapter 4, we discussed the multisegment well (MSWell) model, which has been integrated in GPRS. A large number of segments may be used to represent multilateral wells; for example, for a black-oil MSWell model, there are four equations (Equations 4.8 and 4.12), and correspondingly four unknown variables, per segment. This can easily lead to hundreds of equations and variables per well, and for systems with large numbers of MSWells, this can become a significant part of the construction and solution of the coupled system of reservoir models and wells.

Figure 7.12 shows a Jacobian matrix for a system with an MSWell. In fact, both the reservoir and well parts of the Jacobian can be large and complex. Recall that the reservoir and MSWell models are governed by different equations and different variable sets. This increases the complexity of the Jacobian. Note that the Jacobian matrix arising from a system with MSWells is stored using the multilevel sparse block (MLSB) data structure. In systems with either MSWells or StdWells, the pressure is always the major coupling mechanism. This implies that a CPR-like approach can be a promising method to precondition the matrix for a system with MSWells. The basic idea of CPR remains the same for a system with MSWells. In the first stage, an approximate pressure matrix is constructed and solved. In the second stage, a preconditioner, usually from the ILU family, is applied to the full matrix. There are

two specific challenges we need to overcome before the CPR preconditioner can be applied to coupled reservoir models with MSWells:

1. How to generate an approximate pressure matrix for a system with MSWells?
2. How to apply the ILU preconditioner using the multilevel sparse block (MLSB) matrix in the second stage?

7.5.1 First Stage: Pressure Equation Construction and Solution

The MSWell model dramatically increases the number of equations per well. In Chapter 4, for example, we showed a model with 1800 segments (Figure 4.10), where we solve for four unknowns per segment. The resulting Jacobian is of mixed (elliptic-hyperbolic) character, for both the reservoir and the wellbore. For the first stage of CPR, we need to construct a pressure system of equations from the full Jacobian in an algebraic manner; the constructed equation must capture the pressure coupling in the reservoir-facilities model, and it should display near-elliptic character so that it can be solved efficiently using multigrid. The algebraic construction, or decoupling process, is accomplished in two steps. First, the MSWell equations in the Jacobian matrix are algebraically reduced to a form similar to a StdWell equation. In the second step, an IMPES reduction is applied to the modified matrix (the reservoir model and the reduced well model) to generate the pressure matrix. The second step is identical to the process described for the first stage of CPR for systems with StdWells.

For the system shown in Figure 4.5, the Jacobian matrix with an oil-rate constraint is shown in Figure 7.12. The two red lines separate the reservoir and the facilities (only one well) parts, and these four parts are stored in separate data structures. In

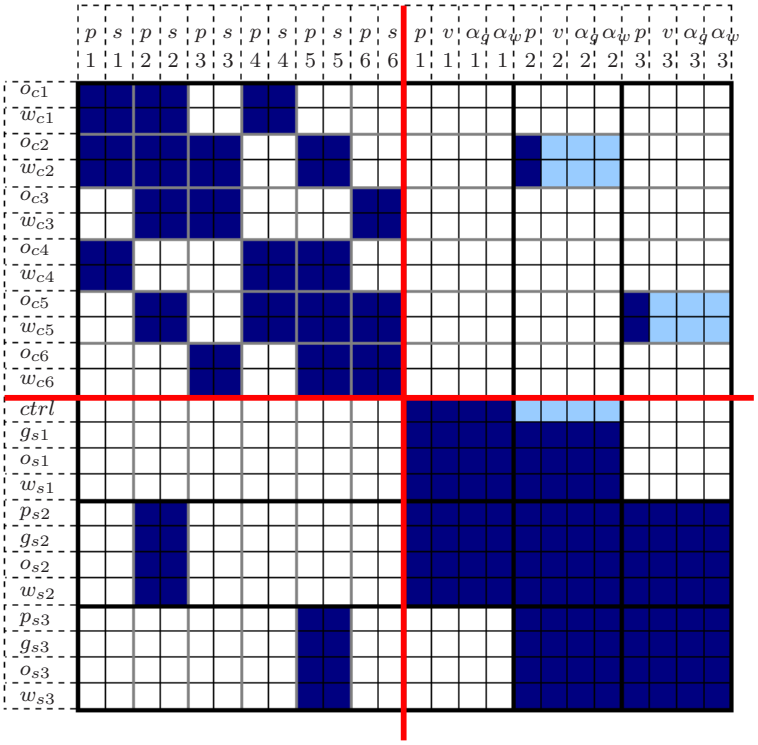


Figure 7.12: Jacobian matrix with oil rate constraint (same as Figure 4.8)

Chapter 4, we discussed that the oil-rate constraint on the top segment is equivalent to a StdWell-like constraint, given the oil conservation equations for all segments (Equation 4.27 - 4.30). Summation of the oil-component conservation equations of the segments, we get (same as Equation 4.28):

$$
\begin{aligned}
R_{\bar{o}} \;=\; & \sum_{i=1}^{nsegs} \left(\frac{A\Delta x}{\Delta t} \sum_{i=1}^{np} \left[(\rho_p x_{\bar{o},p} \alpha_p)^{n+1} - (\rho_p x_{\bar{o},p} \alpha_p)^{n} \right] \right)_i + \\
& \rho_{\bar{o}} q_{\bar{o}} - \sum_{i=1}^{nsegs} \left[\sum_{p=1}^{np} \rho_p x_{\bar{o},p} \lambda_p WI (p^{res} - p^{seg}) \right]_i = 0.
\end{aligned} \tag{7.29}
$$

Equation 7.29 is a summation of (non)linear algebraic equations. Since both differentiation and summation operations can be exchanged, it does not matter if we sum the

equations first and then generate derivatives, or do these operations in reverse order. In our problem, the derivatives in the Jacobian matrix have already been generated, and it is easy to work on the matrix directly. A matrix row operation can be applied to the Jacobian to sum up the related equations. The dimension of the row operation matrix is $(n_{re} + n_w) \times (n_{re} + n_{we})$, where n_{re} is the number of reservoir equations, n_w is the number of wells and n_{we} is the number of total well equations. The expression for the reduction operation is:

$$\boldsymbol{A}^*_{MS} = \boldsymbol{R}_{MS}\, \boldsymbol{A}_{MS}, \tag{7.30}$$

where $\boldsymbol{A}_{MS}$ is the Jacobian matrix for systems with MSWells, $\boldsymbol{R}_{MS}$ is the row operation matrix, $\boldsymbol{A}^*_{MS}$ is an intermediate Jacobian matrix.

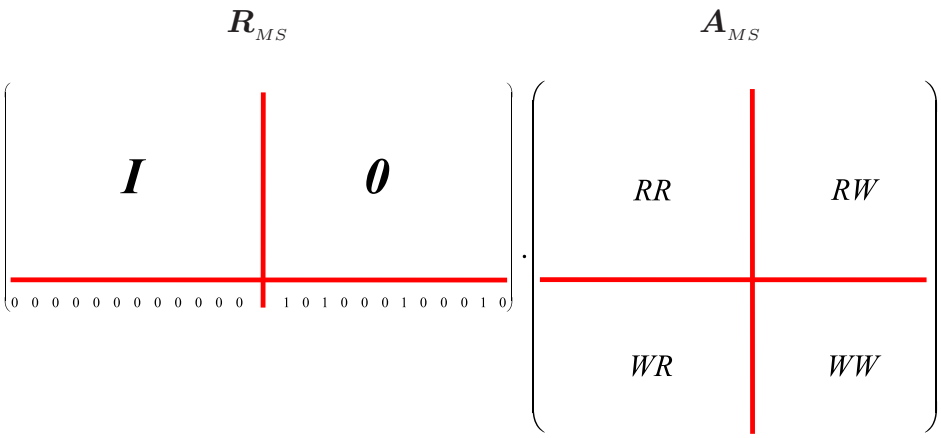


Figure 7.13: MSWell matrix reduction in equations (row operation)

This process is shown in Figure 7.13 ($n_w = 1$, $n_{we} = 12$). In the plot, $\boldsymbol{I}$ represents an identity matrix and $\boldsymbol{0}$ stands for a zero matrix. The row operation matrix leaves the reservoir part of the matrix unchanged and replaces the constraint equation with the summation. The other well equations are ignored. The last row of the matrix acts

as a row selector. The positions of the non-zeros may be different for other constant rate controls. The resulting matrix is shown in Figure 7.14. The same row operations should be applied to the right-hand-side of the system.

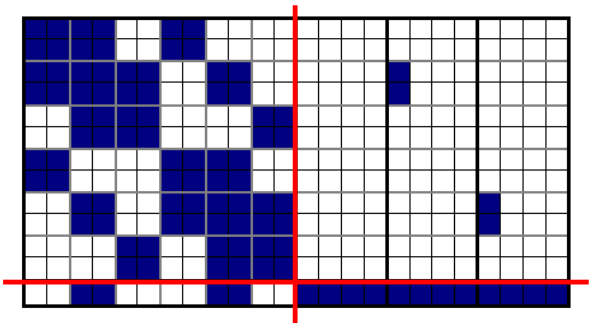


Figure 7.14: Result of MSWell matrix reduction in equations (row operation)

The matrix in Figure 7.14 is not square, but has more unknowns than equations. If the values of the segment variables (except pressure) are fixed at the last iteration, the new constraint equation (Equation 7.29) has no derivatives with respect to them. The matrix can be simplified as shown in Figure 7.15. The right-hand-side vector remains unchanged in this reduction. If the pressure relation between segments (not pressure) is fixed at the last iteration, then the pressure of the reference point (top segment) is the only variable. Pressure derivatives of other segments will be lumped into the reference pressure derivative (Equation 7.31):

$$\frac{\partial R}{\partial p_i^{seg}} = \frac{\partial R}{\partial (p_{ref}^{seg} + \Delta p_i)} = \frac{\partial R}{\partial p_{ref}^{seg}}, \tag{7.31}$$

where Δp_i is the pressure difference between segment i and the top segment. Since the pressure relation is fixed at the last iteration, Δp_i is a constant value. In this step, no update is needed for the right-hand-side vector.

The one-iteration lag treatment for the segment pressure relations and the other segment variables makes the reference pressure the only variable for the MSWell. The

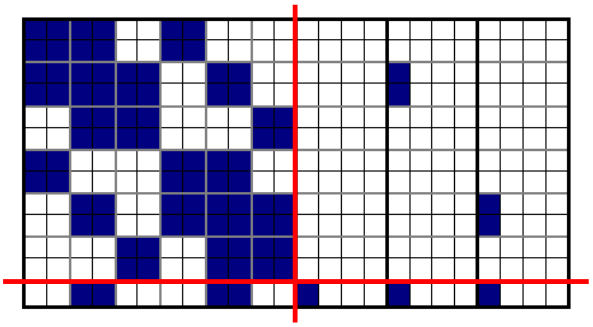


Figure 7.15: Reduced matrix with explicit treatment of velocity and holdups

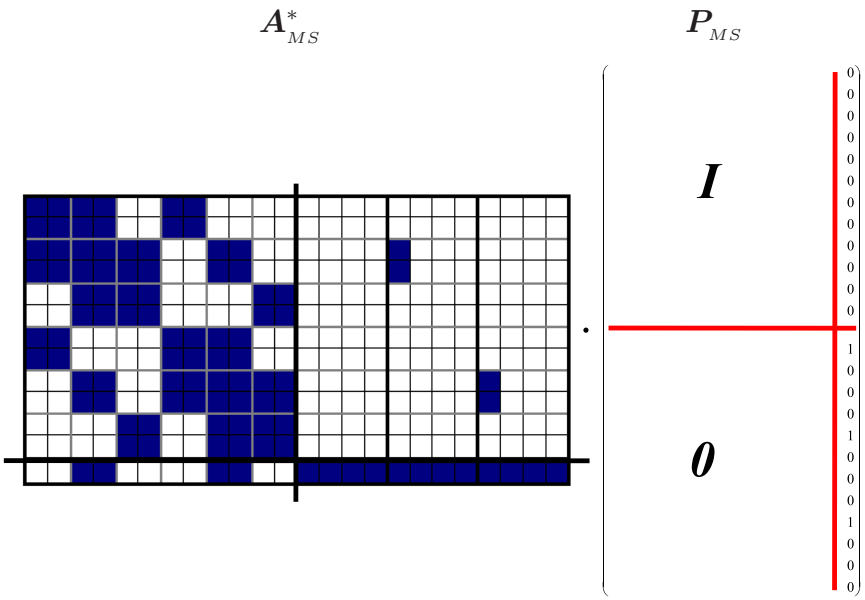


Figure 7.16: MSWell matrix reduction in variables (column operation)

process can be represented by matrix column operations shown in Figure 7.16. The final result of the algebraic reduction of the MSWell equations is shown in Figure 7.17, which has an identical structure to the matrix of a system with StdWells. The equation is written as:

$$\boldsymbol{A}_{Std} = \boldsymbol{R}_{MS}\, \boldsymbol{A}_{MS}\, \boldsymbol{P}_{MS}, \tag{7.32}$$

where $\boldsymbol{A}_{Std}$ is the reduced matrix, which has the same form as the one arising from a system with StdWells; $\boldsymbol{P}_{MS}$ is a column operator, which is also the prolongation matrix for the solution vector.

After simplification of the MSWell representation in the algebraic system of equations, the IMPES reduction (true-IMPES or quasi-IMPES) is used to obtain the pressure matrix. This step is exactly the same as that used in the system with the StdWell model. Therefore, combining Equation 7.32 and Equation 7.23, the relation between the Jacobian matrix (with MSWell) and the pressure matrix can be written as:

$$\boldsymbol{A}_p = \boldsymbol{R}\, \boldsymbol{R}_{MS}\, \boldsymbol{A}_{MS}\, \boldsymbol{P}_{MS} \boldsymbol{P}. \tag{7.33}$$

The pressure matrix, $\boldsymbol{A}_p$, is solved with AMG, and the pressure correction, $\boldsymbol{x}_p$ needs to go through two prolongation operations. The first one is for the reservoir part (Equation 7.24), which is exactly the same as the procedure for a system with StdWells. After the prolongation, the solution vector $\boldsymbol{x}_1$ is mapped back to the vector space for an FIM system with StdWells. We need one more prolongation to expand the well part:

$$\boldsymbol{x}_{1,MS} = \boldsymbol{P}_{MS}\, \boldsymbol{x}_1, \tag{7.34}$$

where $\boldsymbol{x}_{1,MS}$ is the final solution of the first stage of CPR, and $\boldsymbol{P}_{MS}$ is the matrix

shown in Figure 7.16. This operation assigns the pressure change of the reference point to the pressure variables of all the segments and leaves the other variables as zeros. The overall prolongation operation can be written as:

$$\boldsymbol{x}_{1,MS} = \boldsymbol{P}_{MS}\, \boldsymbol{P}\, \boldsymbol{x}_p. \tag{7.35}$$

If the MSWell model is operated under constant-pressure control, the pressure at the reference point is set at a certain value. Again, for preconditioning purposes, the pressure relation of segments will be fixed at the last iteration. Therefore, the pressures of all the segments are known, and the reservoir is naturally decoupled from the wells.

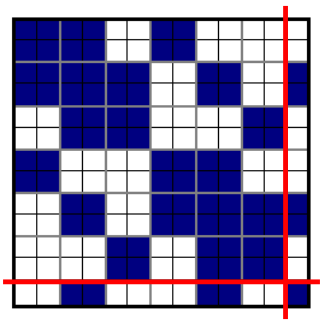


Figure 7.17: Final result of MSWell matrix reduction

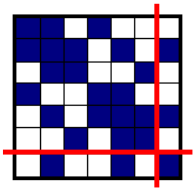


Figure 7.18: Result of IMPES matrix reduction

7.5.2 Second Stage: Reservoir-Facilities Block ILU Preconditioner

The global Jacobian matrix of the system with MSWells is stored in the multilevel sparse block (MLSB) data structure. At the top level, the matrix can be seen as a 2×2 block matrix. Current ILU preconditioners cannot factorize this matrix, because the information is encapsulated within different submatrices.

The coupling between the reservoir and facilities is mainly through the pressure relation. Since the pressure system including wells has been solved in the first stage and removed from the right-hand-side vector, we can drop off $\boldsymbol{J}_{RF}$ and $\boldsymbol{J}_{FR}$ matrices in the second stage for preconditioning purposes (Figure 7.19). Then the reservoir and facilities parts are completely decoupled. ILU factorization can be applied to each of them. It is worthwhile to restate that neglecting the two coupling matrices is only as a preconditioner in the second stage of CPR. The linear solver always deals with the fully coupled Jacobian with the 2×2 substructure.

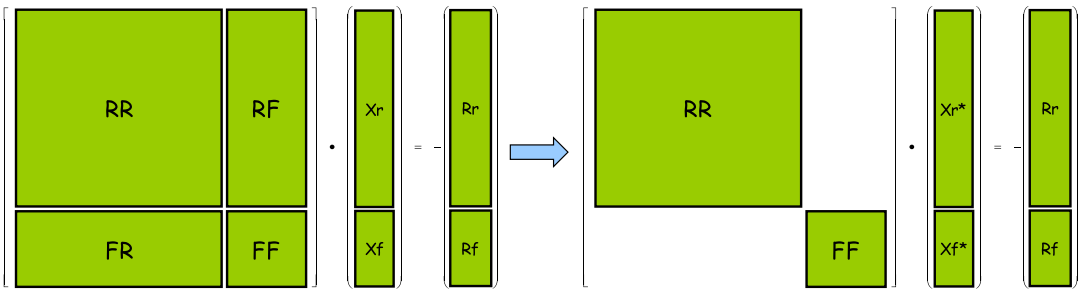


Figure 7.19: Decouple reservoir and facilities in the second stage of CPR

When we investigated the preconditioner options for systems with StdWells in the previous section, we showed that BILU(0) is the best choice (Table 7.5) for the second stage in CPR. A similar comparison indicates that for systems with MSWells,

BILU(0) is still the best option for the reservoir part in the second stage of CPR. The $\boldsymbol{J}_{FF}$ matrix is composed of a number of submatrices from wells or well groups, which lie on the diagonal of the $\boldsymbol{J}_{FF}$ (Figure 3.5). These $\boldsymbol{J}_{WW}$ matrices are not coupled with each other, so they can be factored one by one.

For a single lateral well, the segments form a one-dimensional grid system. Hence, the $\boldsymbol{J}_{WW}$ matrix of the well is a tridiagonal block matrix. Figure 7.20 shows a 9-segment well and its $\boldsymbol{J}_{WW}$ matrix. Under this circumstance, the BILU(0) is an exact LU factorization. If a well has multiple laterals, the $\boldsymbol{J}_{WW}$ matrix is not tridiagonal any more. Figure 7.21 shows the $\boldsymbol{J}_{WW}$ matrix for a two-lateral, 9-segment well. For this case, BILU(0) gives a poor approximation to the original matrix. Most of the time, it fails to converge. For multilateral wells, BILU(1) is used for preconditioning the $\boldsymbol{J}_{WW}$ matrices in the second stage of CPR, and that gives excellent overall performance.

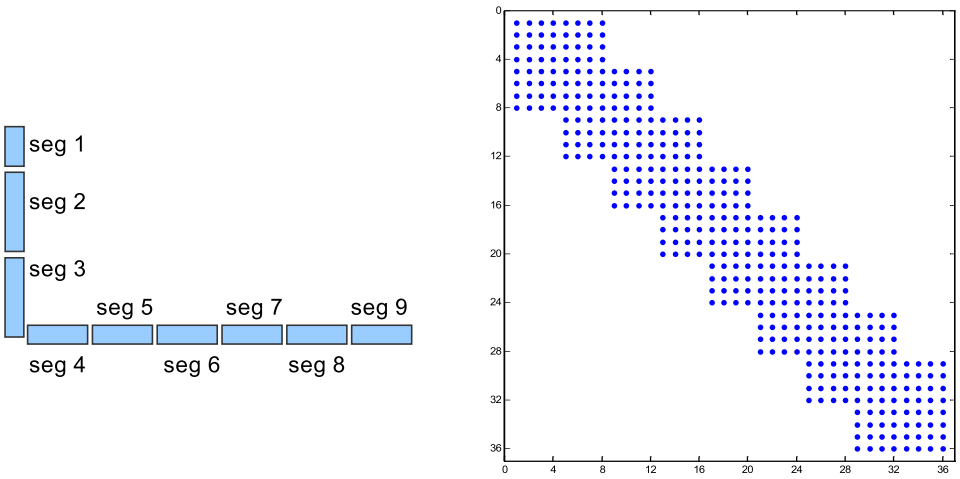


Figure 7.20: Single-lateral MSWell and its $\boldsymbol{J}_{WW}$ matrix

In summary, the multilevel BILU preconditioner in the second stage of CPR for coupled reservoir-facilities systems is composed of one BILU(0) preconditioner for the reservoir matrix and a number of BILU(1) preconditioners for the well matrices

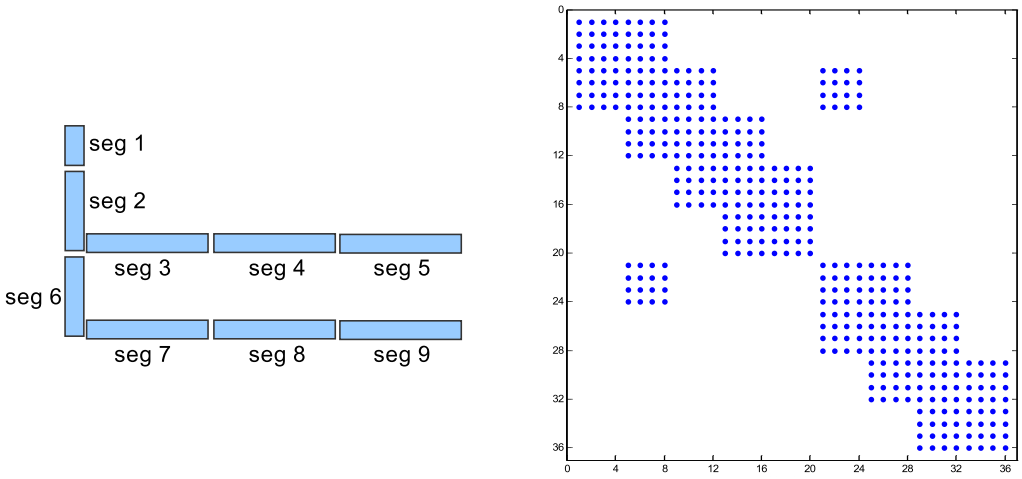


Figure 7.21: Two-lateral MSWell and its $\boldsymbol{J}_{WW}$ matrix

in the facilities part. This multilevel BILU strategy can be used as an independent preconditioner (preconditioner option 11) for systems with MSWells. In this scenario, without considering the pressure coupling, neglecting the coupling terms ($\boldsymbol{J}_{FR}$ and $\boldsymbol{J}_{RF}$) may make the multilevel BILU a weaker preconditioner compared with the CPR strategy used for StdWells.

7.5.3 Test Case

Case 4

A test case was set up to compare the performance of the MSWell and StdWell models. This case also highlights the performance of the CPR and ILU preconditioners with both well models. In this case, the reservoir model is an upscaled version of the top part of SPE10, which has 110×30×16 gridblocks. The model is a highly heterogeneous and has oil and water. Five bilateral horizontal producers and two vertical water

injectors are drilled in the reservoir. The producers have three perforations in each horizontal branch. Each injector has two perforations at its bottom. The entire system is shown in Figure 7.22. In order to have a detailed comparison, the case is run with four different settings:

1. All seven wells use the StdWell model. The preconditioner option is CPR for a system with StdWells.
2. All seven wells use the StdWell model. The preconditioner option is ILU.
3. The five bilateral producers use the MSWell model, and the two injectors use the StdWell model. The preconditioner option is CPR for a system with MSWells.
4. The five bilateral producers use the MSWell model, and the two injectors use the StdWell model. The preconditioner option is ILU, same as the second stage of CPR for a system with MSWells.

The performance information is listed in Table 7.8. The solver time, which is the time spent on solving the matrix, is the benchmark target. There are three key observations from the table.

1. By comparing options 1 and 2, we see that CPR has much better performance than ILU for a system with StdWells. For this case, the speedup factor is about 4.3. Similar performance is observed for the other examples presented in this chapter.
2. By comparing options 3 and 4, we find that the CPR preconditioner gives much better performance for system with MSWells compared to the ILU preconditioner. The speedup factor is about six. This clearly shows the importance of developing a CPR method for systems with MSWells.

3. The system with MSWells is more costly to solve than the one with only StdWells. For this case, the additional cost is about 40% (compare options 1 and 3). This is not surprising, since the MSWell model provides information about the flow behavior inside the wellbore by solving the relevant conservation equations.

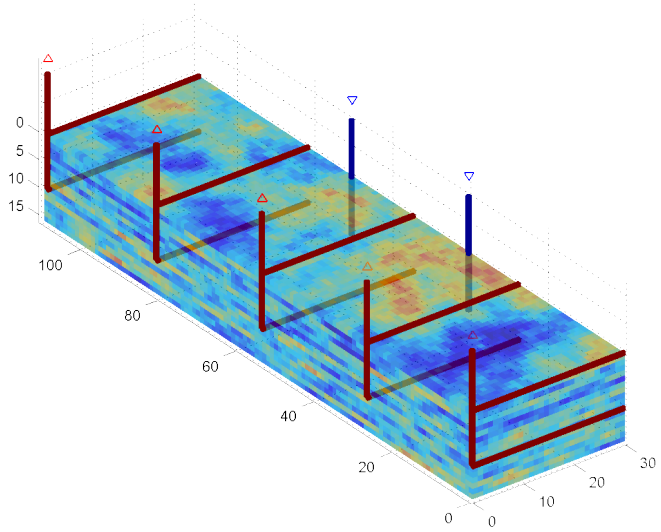


Figure 7.22: Upscaled top formation of SPE10 reservoir model with MSWells

Well Type	Stdwell	Stdwell	MS+Std	MS+Std
Solver	PGMRES	PGMRES	BGMRES	BGMRES
Preconditioner	CPR	ILU	CPR	ILU
Timestep	19	19	19	19
Newton iteration	43	43	42	42
Solver iteration	239	4419	358	6339
Pressure iteration	239	0	358	0
Solver time	110.1	470.5	156.0	927.2
Total time	124.2	485.0	170.8	941.0

Table 7.8: GPRS timing performance for multisegment well case.

7.6 Concluding Remarks

In this chapter, we discussed the solver and preconditioner options in GPRS. Preconditioners are the focus of this chapter. We analyzed the implementation details of the CPR preconditioner and demonstrated its excellent performance for various large-scale simulation cases. As a two-stage preconditioner, each stage of CPR can be further improved. The SAMG package is a better choice compared to the open-source AMG preconditioner for the first stage. Block ILU preconditioners are developed to further enhance the second stage performance. Optimal parameters are found for various preconditioning options.

The two-stage CPR preconditioning approach is extended to handle systems with MSWells. In the first stage, an algebraic two-step reduction procedure is developed to construct a pressure matrix for the system. In the second stage, the coupling terms between the reservoir and facilities are ignored, and a multilevel block ILU preconditioner is developed to handle the decoupled objects. This approach can be extended to a system composed of complex reservoir models and various facility models. In the first stage, the coupled pressure system is constructed to cover all the reservoir and facility objects. In the second stage, the objects are decoupled, and preconditioners from the ILU family can be used to precondition the system.

Chapter 8

Adaptive Implicit Method

8.1 Introduction

In Chapter 7, we discussed preconditioning options for the linear solution of the Jacobian matrices in GPRS. Among the various options, the block-based two-stage CPR preconditioned GMRES solver, with AMG in the first stage and BILU in the second, is the most robust and most computationally efficient approach. The CPR-based linear solver framework speeds up FIM (Fully Implicit Method) simulations significantly in the presence of standard and/or multi-segment wells. In this chapter, we discuss another approach to speed up compositional simulation, namely, the Adaptive Implicit Method (AIM) [19], in which a primary variable in a gridblock may be treated implicitly, or explicitly, in a dynamic manner.

In FIM simulation, all the primary variables in a gridblock are treated implicitly (i.e., at the current time level), and this leads to an unconditionally stable time discretization scheme. Therefore, in FIM simulation, one can take larger timesteps

compared with explicit methods. However, for compositional models with large numbers of components, the FIM approach leads to very large Jacobian matrices, and that increases the computational cost of constructing and solving the Jacobian matrices significantly. On the other hand, in the mixed-implicit IMPES (IMplicit Pressure and Explicit Saturation) formulation, only one variable - the pressure - is treated implicitly in every gridblock, and the size of the Jacobian matrix is much smaller than the corresponding FIM Jacobian; the price is that IMPES is conditionally stable with possibly severe limits on the time step size, especially for highly heterogeneous domains. However, for a given stable timestep size, IMPES has less discretization error than FIM [2]. In most reservoir simulation problems, the IMPES timestep is controlled by a few gridblocks with significant saturation changes, or high component throughput. Even for highly detailed, strongly heterogeneous oppositional models, the vast majority of gridblocks in a simulation model at a particular time do not require such a small timestep size. Note that the location of the 'problem gridblocks' may change with time.

The Adaptive implicit method (AIM) combines both the fully implicit and explicit methods in one scheme to take advantage of both schemes, while avoiding the shortcomings of each [19, 39]. In AIM, gridblocks can have different implicit levels. Based on stability criteria, the 'problematic' gridblocks are identified and their primary variables are assigned fully implicit treatment. This helps ensure stability, while allowing for large timestep sizes. Subject to the stability criteria, the primary variable in other gridblocks are treated explicitly. This AIM scheme reduces the size of the Jacobian matrix and lowers the computational solution cost per Newton iteration. Explicit treatment of the majority of the gridblocks also helps to reduce the discretization error. The ultimate goal of compositional AIM simulation is to achieve timestep sizes that are comparable, on average, to those employed in corresponding

FIM simulation. When this can be achieved, AIM allows us to construct and solve smaller Jacobians per Newton iteration; moreover, the results are more accurate than FIM computations (smaller numerical dispersion effects).

The traditional AIM approach uses a combination of IMPES and FIM schemes [19]. A more general AIM scheme was proposed by Cao that combines IMPES, IMPSAT (IMplicit Pressure and SATuration, explicit compositions) and FIM [8]. Carefully tuned percentages of the basic schemes can make AIM a very efficient method. Our focus in this chapter is on the data structures, linear solvers and preconditioners for AIM simulation. In particular, AIM with fixed percentages of FIM and IMPES gridblocks is used in the following discussion.

Generally, the implicit level of a gridblock in AIM can be different for different timesteps. (Note that theoretically, one can label individual primary variables as opposed to gridblocks). This leads to dynamic changes in the Jacobian matrix, where both the overall size and the sparsity pattern of the Jacobian matrix can change with time. Due to limitations associated with the pointwise data structures in the previous version of GPRS, the Jacobian matrix had to be destroyed and rebuilt every timestep [8]. The rebuilding process is very expensive and adds to difficulties in maintaining and extending the simulator. This partly explains the limited availability of good preconditioning methods for AIM. A pointwise ILU preconditioner was the only option for AIM in the previous version of GPRS [8].

In this chapter, flexible and efficient data structures for the AIM formulation are presented. With these data structures, the block based CPR preconditioned GMRES solver is fully compatible with AIM, which is our proposed linear solution strategy for compositional AIM models. A few large-scale, highly heterogeneous, compositional cases are used to demonstrate the performance of AIM with our data structures and CPR-based linear solver framework. The impact of AIM on the timestep size is also

investigated.

8.2 Timestep and CFL Number

Although FIM is unconditionally stable, if the timestep size is too large, the Newton iterations may not converge. In GPRS, the size of the FIM timestep is chosen as follows [2]:

$$\Delta t^{n+1} = \Delta t^n \left[\frac{(1+\omega)\eta_i}{\delta_i + \omega\eta_i} \right], \tag{8.1}$$

where Δt^{n+1} is the size of timestep $n+1$, Δt^n is the size of timestep n (the previous timestep), η_i is the maximum allowed change in a primary variable. The values of η_i for different variables should be chosen carefully, so that timesteps converge within a reasonable number of Newton iterations. In GPRS, the default values are 200 psi for pressure, 0.2 for saturations, and 0.02 for component mole fractions. δ_i is the actual change in a primary variable during the previous timestep, and ω is a damping factor between 0 and 1. Smaller ω means adjusting the timestep size more aggressively. Equation 8.1 is computed for every primary variable for all the gridblocks in the reservoir model. The minimal value of Δt^{n+1} is used as the size of the next timestep [8].

Equation 8.1 constrains the timestep based on changes in the primary variables (pressure, saturation(s), component mole fractions) only, and this may not be enough in practical cases. For example, for some high-permeability gridblocks around a BHP (Bottom Hole Pressure) controlled injector, the primary variables do not change significantly after the displacement front passes through the well region. However, these gridblocks may have extremely high throughput and cause stability issues. If

these gridblocks are not given proper consideration when calculating the timestep size, the simulator may overestimate the size of the timestep significantly, and that can lead to convergence problems that are quite difficult to diagnose in practice.

In the AIM scheme, a CFL number is calculated based on the flow field of the last timestep. The expression of the CFL limit for compositional two-phase flow is given by [8, 14]:

$$CFL_{IMPES} = \frac{\Delta t}{V\phi}\, MAX\left(\frac{\frac{\lambda_o}{\lambda_g}\frac{d\lambda_g}{dS_g}q_g - \frac{\lambda_g}{\lambda_o}\frac{d\lambda_o}{dS_g}q_o}{\lambda_o+\lambda_g}, \frac{\rho_o q_o x_c + \rho_g q_g y_c}{\rho_o S_o x_c + \rho_g S_g y_c}\right), \tag{8.2}$$

CFL_{IMPES} is the IMPES-based CFL for a gridblock, V and ϕ denote the bulk-volume and porosity of the gridblock, Δt is the timestep size, q_p $(p = o, g)$ denotes the phase rates, x_c and y_c are the mass fractions of component c in the oil and gas phases, respectively. Note that the CFL number is a local quantity [39]; for each gridblock, the MAX operator selects the larger of the two terms: namely, saturation change, and component-throughput. If only one phase (gas or oil) is present in a gridblock, Equation 8.2 can be reduced to :

$$CFL_{IMPES} = \frac{\Delta t\, q}{V\, \phi}, \tag{8.3}$$

where q is the flow rate for a gridblock. For a cell to be treated with IMPES, the following condition must be satisfied.

$$CFL_{IMPES} \leq 1. \tag{8.4}$$

Recall that in an IMPES simulation (i.e., only pressure is treated implicitly in all gridblocks), the global timestep is controlled by the gridblock with the largest CFL_{IMPES}

number. In AIM, there are two approaches for assigning the implicit level to the primary variables in a gridblock [8]. One is to fix the percentages of the FIM and IMPES gridblocks. In this approach, we compute the $CFL_{IMPES}/\Delta t$ number for each gridblock using Equation 8.2, or 8.3. Then the gridblocks in the model are sorted according to these values. Then, the gridblocks that belong to the fraction that exceeds the CFL limit are assigned FIM treatment; all the other gridblocks are assigned IMPES treatment (i.e., only pressure is treated implicitly). So, the IMPES gridblock with the largest $CFL_{IMPES}/\Delta t$ value dictates the timestep size such that its CFL number is not larger than unity.

Figure 8.1 shows the sorted $CFL_{IMPES}/\Delta t$ for a fractured reservoir model at 10 days. The model has highly heterogeneous properties and unstructured grid (see Case 3 in Section 5). About 16% of the gridblocks have extremely large $CFL_{IMPES}/\Delta t$ numbers (up to 3.1e+8, which does not fit in the figure). As shown in the figure, if the percentage of gridblocks with FIM treatment is fixed at 30%, we obtain

$$\frac{CFL_{IMPES}}{\Delta t} = 2.8. \tag{8.5}$$

Since CFL_{IMPES} should be smaller, or equal, to unity (Equation 8.4) in order to guarantee a stable numerical solution, the timestep size is limited to $2.8\Delta t \leq 1$, or 0.35 days. On the other hand, if the FIM fraction is fixed at 50%, the timestep size can be up to one day. Note that if the FIM fraction is fixed at say 10%, then the stable timestep size is limited to 7×10^{-5} days, which is too small to be of any practical value.

From this discussion, we see that user-specified percentages of FIM and IMPES gridblocks have a critical impact on the timestep size and consequently the overall simulation performance.

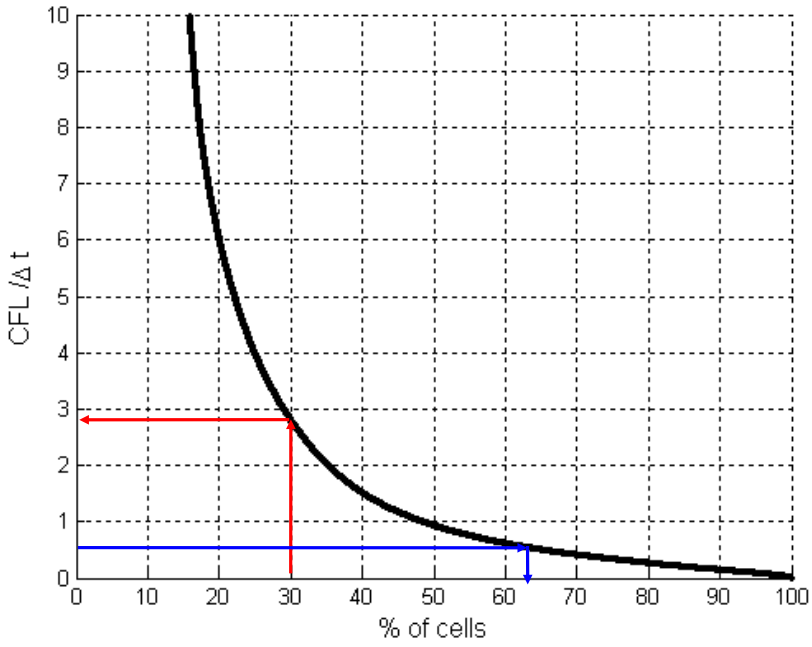


Figure 8.1: CFL number distribtion for a reservoir model with highly unstructured grid (Case 3)

In AIM simulation, the timestep size should be chosen based on the CFL constraint (Equation 8.2); however, in order to obtain well-behaved and relatively smooth profiles of Newton iterations per timestep, the CFL-based stability criteria are usually supplemented with 'reasonable' limits on the maximum allowable changes in the primary variables. In order to determine the percentage of the basic schemes, a plot like Figure 8.1 can be quite helpful. One can also analyze the statistics of the (single-phase) CFL numbers, for example, to help choose the FIM and IMPES percentages. In practice, the percentages of the basic schemes are fixed at the beginning of a run, and Δt is chosen accordingly in the course of the simulation. In this approach, the size of the Jacobian matrices is the same for all the timesteps. This approach is used due to programming, memory management, and computational efficiency considerations, especially when dealing with field-scale AIM compositional models. If the CFL based Δt is larger than a user-specified maximum length, the user-specified limit is used. It is important to point out that even though the overall size of the Jacobian matrix is the same, the sparsity pattern of the computed Jacobian may still change with time, due to different distributions of the basic schemes.

An alternative approach is to vary the percentages of the basic schemes to achieve a desired timestep size. The purpose of this approach is to achieve timestep sizes that are close to those that would be obtained in an FIM simulation. In this case, the CFL numbers for all the gridblocks are calculated with the desired timestep size (Equation 8.2). The gridblocks with a CFL number larger than unity are treated using FIM, and the others are treated with IMPES. The percentage of the basic schemes can also be determined from a CFL-number plot like Figure 8.1. For a given timestep, we can look up the point on the curve whose y coordinate is the reciprocal of the timestep. Then, the x coordinate of the point is the percentage of the FIM scheme. For example, in Figure 8.1, a two-day timestep leads to an AIM scheme with

about 64% FIM gridblocks (blue arrows in Figure 8.1). Generally, the percentages of the different schemes change every timestep using this approach. Consequently, the overall size of the Jacobian matrices may change in the course of a simulation.

8.3 Data Structures for AIM

In Chapter 3, we proposed and implemented two new data structures to deal with matrices in GPRS. In Chapter 7, we showed that our new multi-stage linear solvers and preconditioners, which take full advantage of these new data structures, are robust and efficient for FIM simulation. In fact, the new data structures were especially designed with AIM (Adaptive Implicit Method) in mind. Here, we show how our data structures and new linear-solver framework can be used to solve AIM problems efficiently. In the previous chapters, we showed that for the FIM scheme, the new data structures can simplify the matrix assembly process, reduce memory cost, and speed up the simulation. For AIM simulation, the new data structures retain all of these advantages. Additionally, the new data structures avoid the high cost for destroying and rebuilding the Jacobian every timestep, which used to be the case for AIM simulation with the original data structures in GPRS.

In an FIM simulation, all the cells have the highest implicit level. Hence, the sparsity structure of Jacobian matrices remains the same. However, in the AIM scheme, the overall size of the Jacobian matrices may change if the approach with varying percentages of basic schemes is used. The Jacobian matrices using the approach with fixed percentages of the basis schemes are of the same overall size, but the structure of the Jacobian matrices may still change, due to the changing spatial distributions of the basic schemes. Therefore, the AIM scheme always leads to dynamic Jacobian matrices.

The compressed row storage (CRS) data structure does not accommodate dynamic changes in the sparsity pattern of the Jacobian matrix. With the CRS format, if the sparsity structure of the Jacobian matrix changes in the course of a simulation run, the matrix has to be destroyed and rebuilt from scratch. On the other hand, the new data structures can handle dynamic matrix structures in a very efficient manner. We use the Jacobian matrix for an AIM reservoir model as an example. This is also the matrix format used as a sub-component in the multilevel sparse block (MLSB) matrix structure. The pseudocode of the matrix representation is as follows:

```
class SubRRBlk {
    public:
        SubRRBlk::SubRRBlk();   // Constructor
        operator*(Vector& v);   // Matrix-vector operator
    private:
        int nRows;              // No. of rows
        int nCols;              // No. of cols
        double *mDiag;          // Diagonal array
        double *mOffD_A;        // Upper off-diagonal array
        double *mOffD_B;        // Lower off-diagonal array
        int *mImpLvl;           // Implicit level of cells;
}.
```

The matrix contains three data arrays: diagonal, upper and lower off-diagonal arrays. Logically, these arrays represent the blocks (small dense matrices) on the diagonal and the off-diagonals of the Jacobian matrix. The size of the diagonal array is $n_{cell} \times n_c \times n_c$, where n_{cell} is the number of cells in the reservoir model and n_c is the number of components. The size of each off-diagonal array is $n_{conn} \times n_c \times n_c$, where n_{conn} is the number of the connections between cells. The sizes of these arrays

remain the same regardless of the specific spatial distribution of the basic schemes. For the FIM scheme, all the elements in the arrays are used. For the AIM scheme, only some of the elements in the blocks that make up these arrays are used. There is an additional array 'mImpLvl' (whose size is equal to the number of gridblocks), in which the implicit level of each gridblock is stored. The positions of the useful elements in the blocks within the diagonal, upper, and lower off-diagonal arrays, can be obtained from the 'mImpLvl' array.

Figure 8.2 shows a synthetic reservoir model with 4×2 gridblocks. The model has three components. The gridblocks are labeled with the numbers in their upper-left corners. A producer is located in cell 1 and an injector is in cell 8. In this case, the maximum implicit level is three. Hence the diagonal and off-diagonal arrays are made up of 3×3 blocks (although they are stored in one-dimensional arrays in physical memory). The corresponding logical layout of the diagonal and off-diagonal arrays is shown in Figure 8.3. The blue (both light and dark blue) blocks represent the diagonal array, and the green (both light and dark green) blocks represent the two off-diagonal arrays.

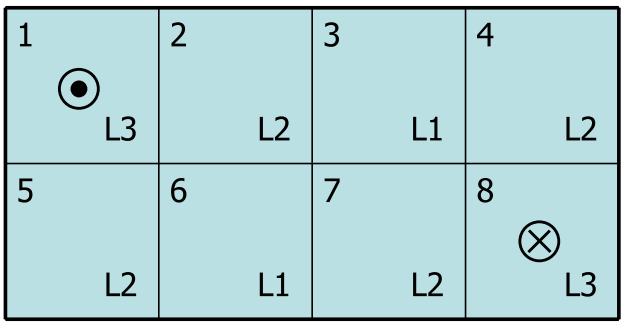


Figure 8.2: A synthetic reservoir model with AIM scheme

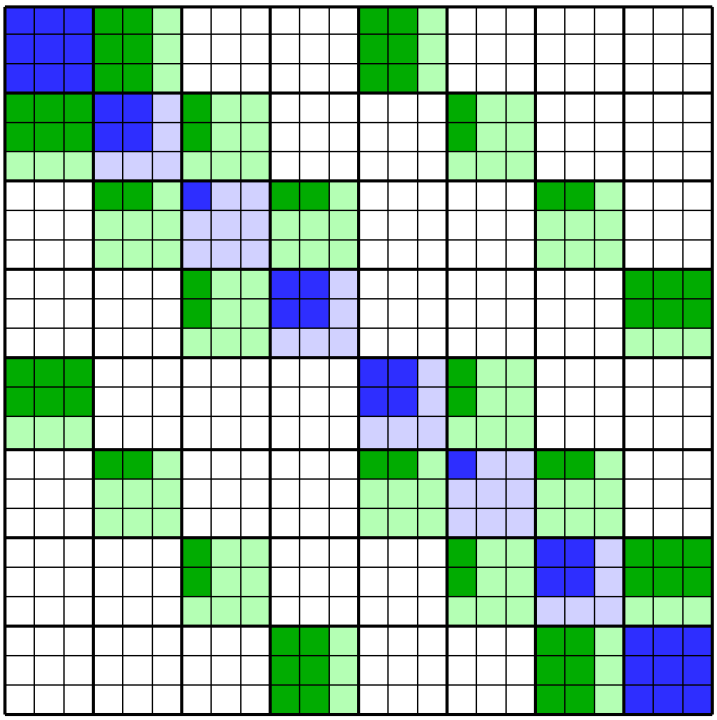


Figure 8.3: Layout of the Jacobian matrix of reservoir model for AIM scheme

For a certain timestep, the implicit level of each gridblock is shown in the bottom-right corner. This information is stored in the "mImpLvl" array as $\{3, 2, 1, 2, 2, 1, 2, 3\}$. In this scenario, only some of the elements in the diagonal and off-diagonal arrays store valid information, which are marked with darker colors in Figure 8.3. The distribution of the useful entries can be inferred from the "mImpLvl" array easily. For example, a block (i, j) has a useful subblock of $l_i \times l_j$, where l_i is the implicit level of cell i and l_j is the implicit level of cell j. In each Newton iteration, the linearization function of the AIM scheme is responsible for updating the three basic arrays. This is equivalent to updating the reservoir Jacobian matrix. If the implicit level of some gridblocks changes, the sparsity pattern of the Jacobian matrix is easily updated.

The Jacobian matrix of the reservoir model can be solved with Krylov solvers, which require a matrix-vector operation. The reservoir matrix is composed of the three data arrays, which can be written in matrix form as:

$$\boldsymbol{A}_{res} = \boldsymbol{D} + \boldsymbol{L} + \boldsymbol{U}, \tag{8.6}$$

where $\boldsymbol{A}_{res}$ is the reservoir Jacobian matrix, $\boldsymbol{D}$, $\boldsymbol{L}$ and $\boldsymbol{U}$ are the diagonal, lower off-diagonal, and upper off-diagonal matrices. The matrix-vector operation is implemented as follows:

$$\boldsymbol{A}_{res}\, \boldsymbol{b} = \boldsymbol{D}\, \boldsymbol{b} + \boldsymbol{L}\, \boldsymbol{b} + \boldsymbol{U}\, \boldsymbol{b}, \tag{8.7}$$

where $\boldsymbol{b}$ is a given vector. The matrix-vector multiplication is performed in three separate steps, $\boldsymbol{D}\, \boldsymbol{b}$, $\boldsymbol{L}\, \boldsymbol{b}$, and $\boldsymbol{U}\, \boldsymbol{b}$. Each step can be decomposed into many small block-vector multiplications. In AIM simulation, the size of the small blocks (dark-color blocks in Figure 8.3) is dictated by the implicit level of the reservoir gridblocks.

The new matrix data structure can efficiently handle dynamic Jacobians during

an AIM simulation. The size, sparsity pattern, entries of the Jacobian matrix can be updated with minimal cost. In the new framework, the FIM Jacobian matrix can be thought of as a special AIM case, where the implicit level of all gridblocks is equal to the number of primary variables and is constant everywhere throughout the simulation. In the new framework, the three arrays, namely, $\boldsymbol{D}$, $\boldsymbol{L}$, and $\boldsymbol{U}$ reuse the three basic data arrays (introduced in Chapter 3 Section 1); this helps reduce the memory cost of the simulation.

The Block Compressed Row Storage (BCRS) format is fully compatible with the AIM scheme. In this data structure, a pointer matrix in compressed row storage format is used. These pointers locate the blocks in the basic data arrays: diagonal, upper, and lower off-diagonal arrays. The data structure also has the "mImpLvl" array, which describes the dimension of the useful subblocks in the arrays. The data structure offers wide flexibility in preconditioning strategies. For example, efficient blockwise ILU factorizations can be developed based on this data structure. An example of that is shown in Figure 3.8 and 3.7. Please refer Section 3 of Chapter 3 for details.

8.4 Preconditioners for AIM

The AIM scheme has some extra cost for calculating the CFL numbers, allocating the implicit level, and reducing the implicit level in the gridblocks. For black-oil isothermal simulation, the number of primary variables is up to three. It is not always worthwhile to reduce the implicit level of gridblocks from three to one. The AIM scheme is used more often in compositional simulation with large numbers of components. The AIM scheme may reduce the size of the Jacobian significantly for such cases. Recall that the size of the blocks is $n_c \times n_c$. As n_c increases, the block

solvers and preconditioners can take more advantage of better memory and cache utilization.

In Chapter 7, we showed that CPR is the most effective preconditioner for FIM simulation. In the AIM scheme, some of the gridblocks use the FIM scheme, and the others have lower implicit levels (e.g., IMPES). With the flexible and efficient data structures we designed and implemented, we propose to use CPR for AIM simulation.

In the first stage, the pressure matrix is reduced from the AIM matrix as follows:

$$\boldsymbol{A}_p = \boldsymbol{R}_{AIM}\, \boldsymbol{A}_{AIM}\, \boldsymbol{P}_{AIM}, \tag{8.8}$$

where $\boldsymbol{A}_p$ is the reduced pressure matrix, $\boldsymbol{A}_{AIM}$ is the AIM Jacobian matrix, $\boldsymbol{R}_{AIM}$ and $\boldsymbol{P}_{AIM}$ are the restriction and prolongation matrices for AIM. The procedure is similar to the one that generates the pressure matrix from the FIM matrix (Equation 7.23), which is discussed in Chapter 7 Section 4. The difference is that the restriction and prolongation operations are affected by the implicit level of the gridblocks. The same row restriction operation is applied to the right-hand-side vector:

$$\boldsymbol{b}_p = \boldsymbol{R}_{AIM}\, \boldsymbol{b}_{AIM}\ , \tag{8.9}$$

where $\boldsymbol{b}_p$ is the RHS vector of the pressure system, and $\boldsymbol{b}_{AIM}$ is the RHS of the original AIM system. The pressure solution can be obtained by solving:

$$\boldsymbol{A}_p\ \boldsymbol{x}_p = \boldsymbol{b}_p\ . \tag{8.10}$$

AMG is used as the solver for this stage. The resulting $\boldsymbol{x}_p$ is in the vector space of the IMPES system. A prolongation operation is needed to transfer the pressure solution

back to the vector space of the AIM system, and this is written as:

$$\boldsymbol{x}_{1,AIM} = \boldsymbol{P}_{AIM}\, \boldsymbol{x}_p \,, \tag{8.11}$$

where $\boldsymbol{x}_{1,AIM}$ is the solution of the first stage of CPR for the AIM scheme.

In the second stage, an ILU preconditioner is used to solve the original AIM matrix, and the RHS vector is updated with the first stage solution. That is:

$$\boldsymbol{M}_2\, \boldsymbol{x}_{2,AIM} = \boldsymbol{b}_{AIM} - \boldsymbol{A}_{AIM}\, \boldsymbol{x}_{1,AIM}, \tag{8.12}$$

where $\boldsymbol{M}_2$ represents the ILU preconditioner, and $\boldsymbol{x}_{2,AIM}$ is the solution of the second stage. The correction for the CPR preconditioner is the sum of the solutions from both stages:

$$\boldsymbol{x}_{CPR,\,AIM} = \boldsymbol{x}_{1,\,AIM} + \boldsymbol{x}_{2,\,AIM}. \tag{8.13}$$

In the following section, three cases are presented to demonstrate the performance of the two-stage CPR-based linear solver for AIM systems.

8.5 Examples

Case 1

This is a basic case used to compare the performance of both FIM and AIM as well as the performance of different preconditioners. The reservoir model is composed of $100 \times 100 \times 5$ gridblocks, and the fluid has nine components. There is one production well located in one corner of the reservoir. The system is in primary depletion. The AIM scheme uses a combination of 80% IMPES and 20% FIM. We consider AIM

simulation with three different preconditioners: blockwise CPR, pointwise CPR, and ILU. An FIM run with pointwise CPR is provided as reference.

The simulation period is 20 days, and the timestep size is quite small. We attempted to keep the numbers of timesteps and Newton iterations the same in both AIM and FIM runs, so we can focus on the performance of the linear solver. More complicated cases and analysis regarding timesteps will be provided later. The iteration statistics and timing performance of the four runs are listed in Table 8.1.

Runs	A	B	C	D
IMPES/FIM	0.8/0.2	0.8/0.2	0.8/0.2	0.0/1.0
Solver	BGMRES	PGMRES	PGMRES	PGMRES
Preconditioner	CPR-BILU	CPR-ILU	ILU	CPR-ILU
Implicit variables	130000	130000	130000	450000
Timesteps	6	6	6	6
Newton iterations	19	19	19	19
Solver iterations	84	86	2503	75
Pressure iterations	84	86	N/A	75
Solver time	52.1	133.7	484.0	329.8
GMRES/MV	21.0	23.4	237.2	51.8
(B)ILU factorization	9.3	88.3	87.8	239.0
(B)ILU solution	6.3	6.4	159.0	17.2
Pressure decoupling	5.1	5.2	N/A	12.3
Pressure solution	10.4	10.4	N/A	9.5

Table 8.1: Performance of AIM and FIM schemes for Case 1

The upper part of the table lists the basic information of the simulation runs, including the numbers of the implicit variables, timesteps, and the numbers of Newton, linear solver and pressure iterations. The lower part of the table lists the timing performance of the linear solvers and their subcomponents. For example, the "GMRES/MV" row lists the time spent on the GMRES solver and the matrix-vector multiplication operation (excluding preconditioning time). The reported time is in

second.

If we compare Runs B and D, it is very clear that AIM is more computationally efficient than FIM when the same pointwise CPR preconditioner is used. The speedup factor in the solver is about 2.5. The AIM Jacobian matrix is much smaller than that of FIM. Therefore, GMRES, ILU factorization, and the ILU solution time of Run B are much less than those of Run D. Since 80% of the cells are IMPES, the workload of the pressure decoupling process in AIM is also much smaller than that of FIM. The pressure solution time is quite similar, since both schemes solve pressure matrices of the same size and of very similar characteristics.

A comparison of Runs B and C shows that the CPR preconditioner in AIM has much better numerical performance than ILU. The speedup in the solver time is about 3.6 times. The major gain comes from GMRES and ILU solution times. The ILU factorization time is almost the same, since the number of factored matrices (i.e., number of Newton iterations) is the same in both cases. If we compare Runs A and B, we see that the use of blocking in the solver and preconditioner speeds up the computation. Most of the gain comes from the BILU factorization, and a small portion comes from the blockwise GMRES solver. These gains are related to better cache utilization.

Case 2

This is a highly heterogeneous case. The reservoir model is the top layer of SPE10, which has 60×220 gridblocks. The porosity and permeability maps of the model are shown in Figure 8.4. The reservoir fluid has 6 components. The five standard wells in the reservoir are in a 5-spot pattern. The wells are labeled in the porosity map of Figure 8.4. The initial reservoir pressure is above the babble point. The injector

(in the center of the reservoir) pumps light components (87% C1, 10% C3, 3% C6) into the reservoir at a constant pressure. Both FIM and AIM schemes are used to simulate the case for 4000 days. The AIM scheme uses 80% IMPES cells and 20% FIM cells. All settings of the two runs are identical except the implicit level.

In Figure 8.5, we plot the distribution of the logarithm of the $CFL_{IMPES}/\Delta t$ number throughout the domain at 1818 days. The figure shows that a relatively small number of cells have a very high $CFL_{IMPES}/\Delta t$ value (in red), and that many cells have a relatively high value (in yellow and green). These cells form the flow paths between the injector and Producers 1, 2, 3. The rest of reservoir cells have very low $CFL_{IMPES}/\Delta t$ values (in blue). According to the CFL constraint, the more FIM cells in the model, the larger the timestep that one can take. The relation between the percentage of FIM cells and the timestep size at 1818 days is plotted in Figure 8.6. From the plot, we can see that the CFL constraint for AIM, with 20% of the cell treated with the FIM scheme, leads to a timestep size of approximately 50 days. Since 50 days is larger than the user specified maximum timestep size (30 days), a 30-day period is selected as the size of the next step. Figure 8.7 shows the distribution of FIM and IMPES in the reservoir model, with 20% of the cells treated with FIM (red), and the IMPES cells (blue) make up 80% of the total cells.

The pressure and gas-phase distribution of the FIM scheme at 4000 days are shown in Figure 8.8; the same quantities for the AIM scheme are shown in Figure 8.9. The oil production histories of Producer 2 (upper-left corner in Figure 8.4) for both schemes are plotted in Figure 8.10. As expected, the results of both schemes are very similar.

The numerical performance of both schemes is listed in Table 8.2. The most effective solver and preconditioner options, namely, blockwise GMRES and blockwise CPR, are used in the simulations. In the AIM case, we deal with a much smaller

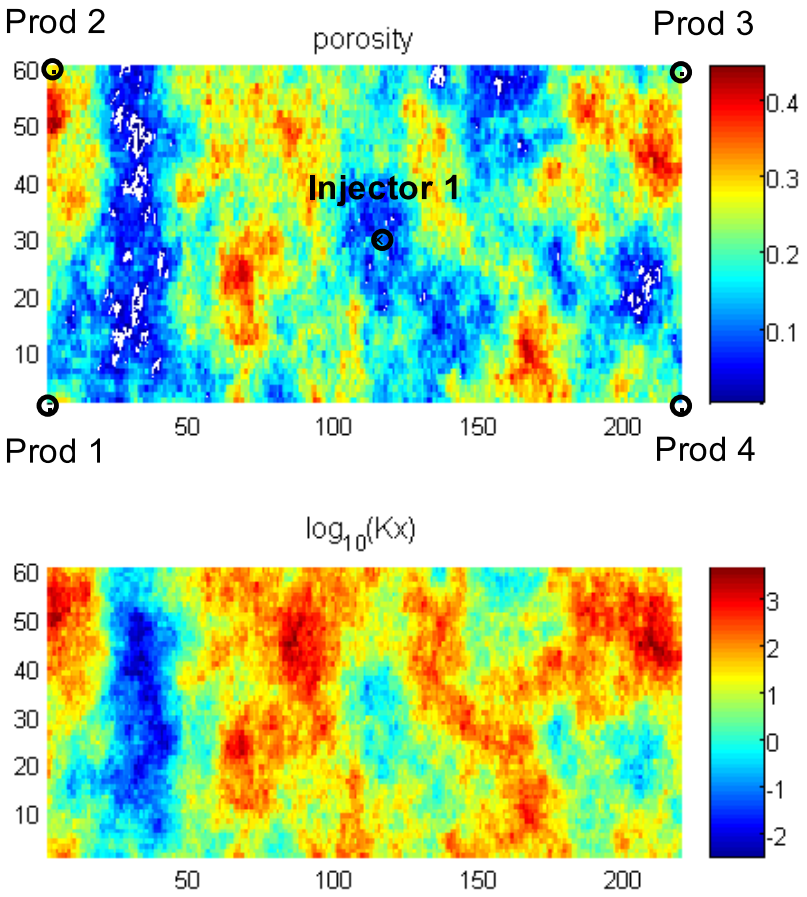


Figure 8.4: Porosity and Permeability maps of the first layer of SPE10

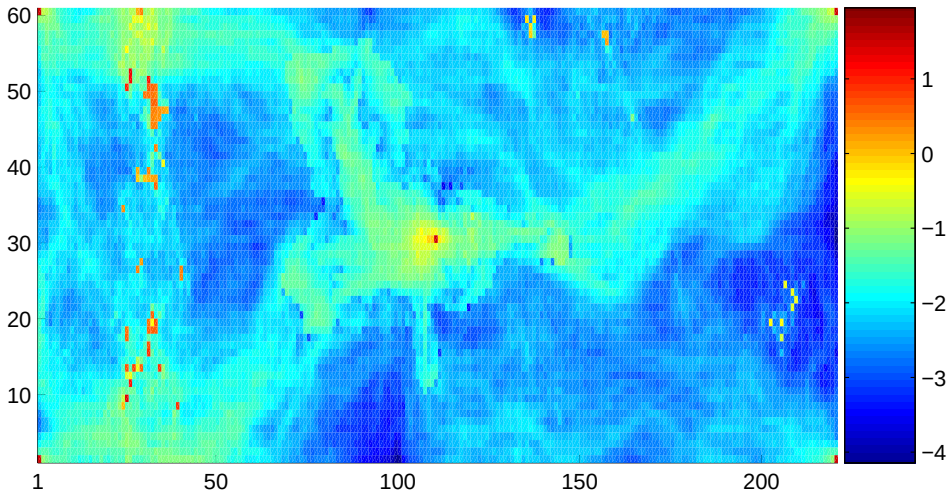


Figure 8.5: Distribution of $\log_{10}(\frac{CFL_{IMPES}}{\Delta t})$ at 1818 days

number of implicit variables and equations, which means we solve smaller Jacobian matrices. For this case, the number of equations of the AIM scheme is one-third of the FIM scheme. The size of the pressure matrices in the CPR preconditioner is always the same.

In this test case, the maximum timestep allowed in the simulation is set at 30 days. The sizes of the timestep for the simulations using both schemes are plotted in Figure 8.11. The simulation runs start with very small timestep size, which are then increased relatively quickly. AIM only considers the constraint by the CFL number. Therefore, in the early stage, the AIM run selects larger timesteps than the FIM run. Several timestep cuts are observed in the early stages of the AIM simulation. Later, the timestep size of both runs reaches the maximum user-specified value. If both the CFL constraint (Equation 8.2) and variable-change limits (Equation 8.1) are considered to decide the AIM timestep size, one may be able to avoid the timestep cuts in the early stages.

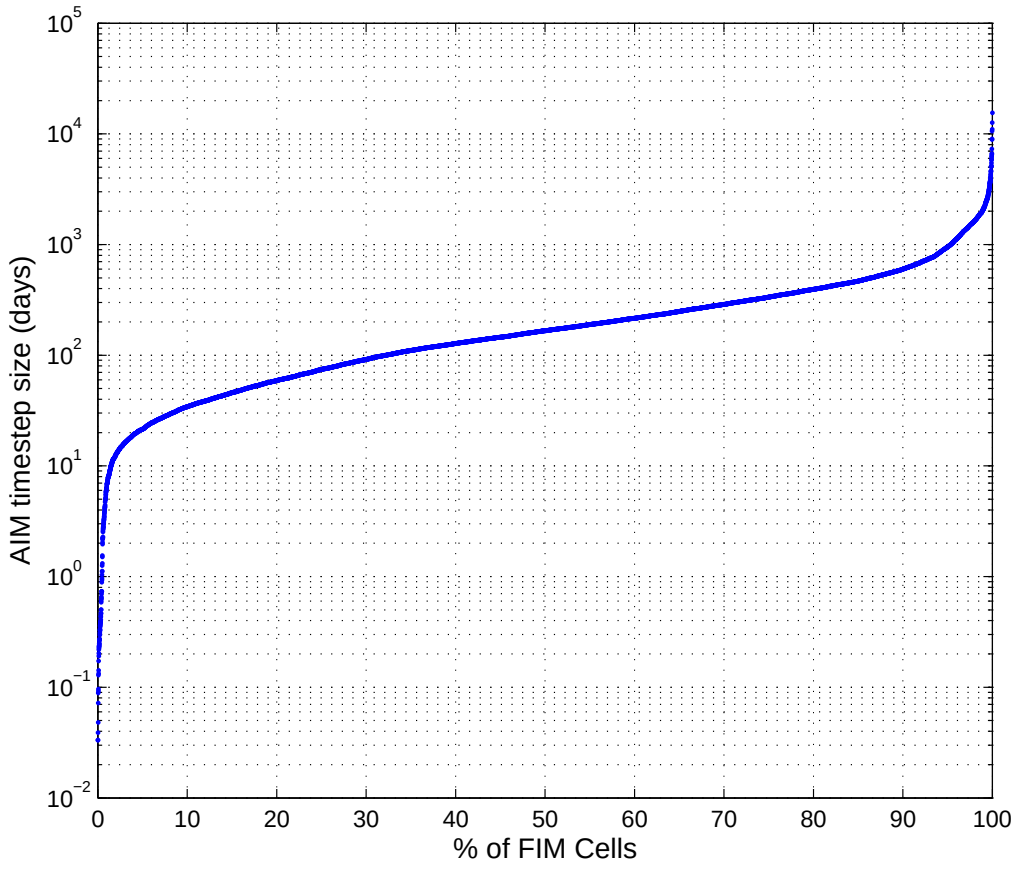


Figure 8.6: Relation between AIM timestep and percentage of FIM cells

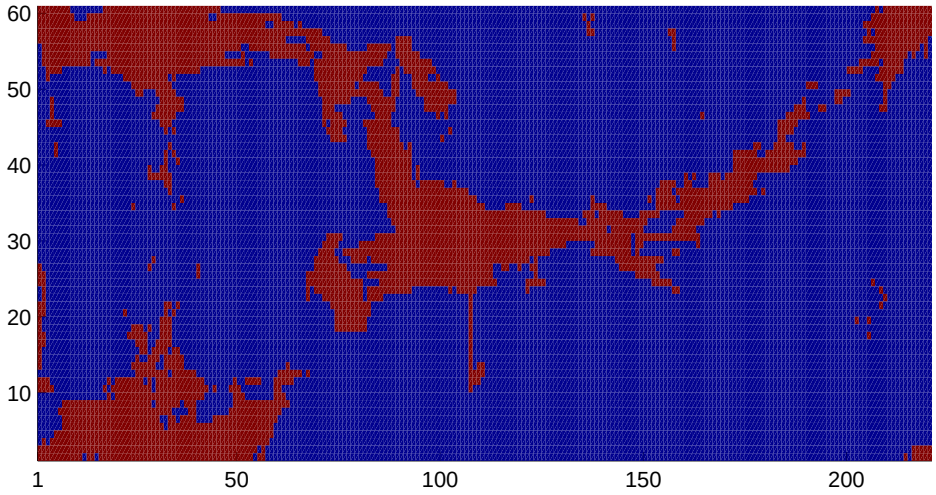


Figure 8.7: Distribution of FIM (red) and IMPES (blue) schemes at 1818 days

Case 3

A compositional case for a fractured reservoir model was selected to further test the performance of the AIM scheme, especially the impact of different percentages of IMPES and FIM on timesteps. The reservoir model and the layout of the fractures and the wells are shown in Figure 7.10, and the grid is shown in Figure 7.11. The reservoir model is discretized using a discrete fracture method (DFM) [26]. The matrix and fractures are discretized into gridblocks according to their shape. About 12% of the reservoir cells are fracture cells, and 24% are matrix cells with connections to the fractures. The fracture cells have 1,000 darcy permeability and unit porosity; the matrix cells have 0.1 md permeability and a porosity of 0.25. The two wells intersect two fractures that are not directly connected by other fractures. The wells have very high throughput due to intersecting the fractures.

GPRS was run with three different options: AIM with 30% FIM and 70% IMPES (referred to as AIM30), AIM with 50% FIM and 50% IMPES (referred to as AIM50),

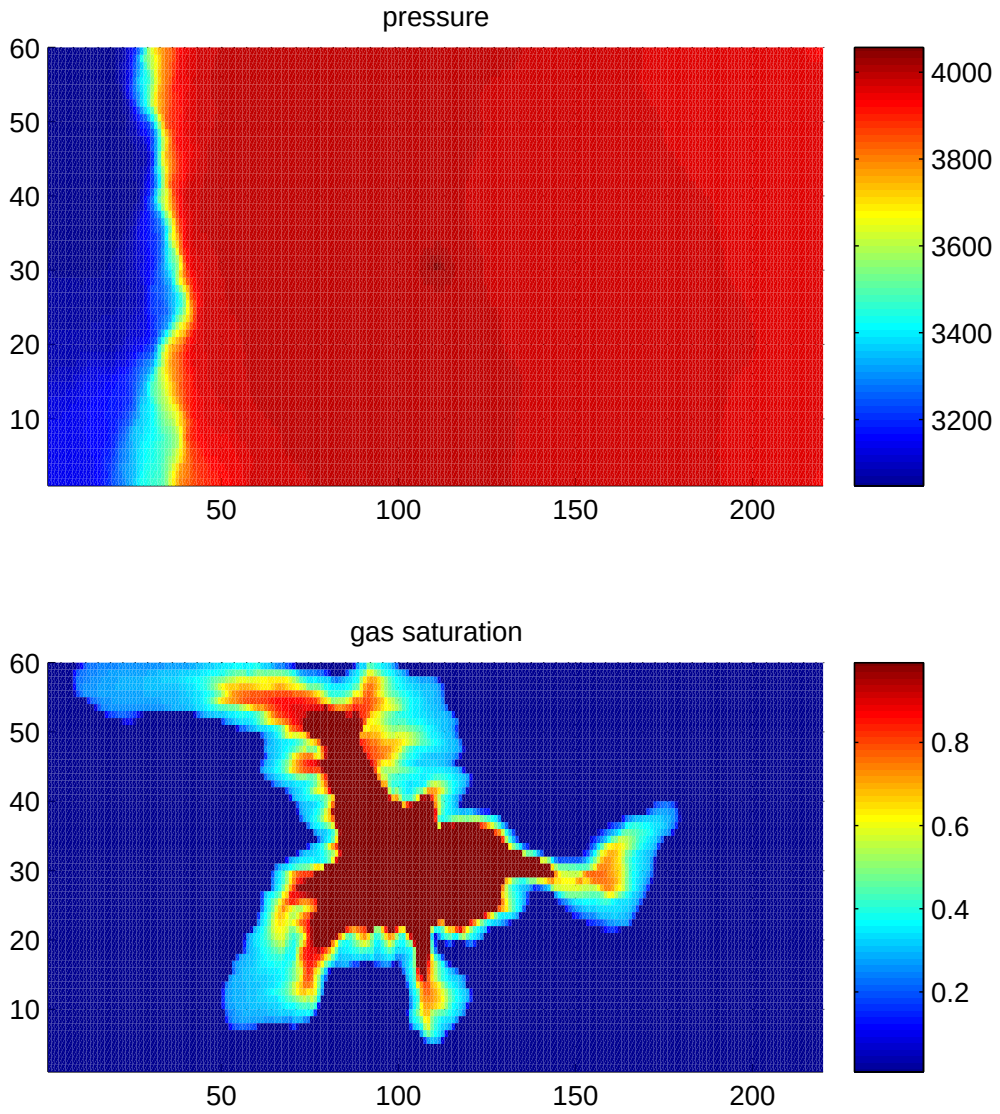


Figure 8.8: Pressure and gas saturation profile maps of FIM scheme at 4000 days

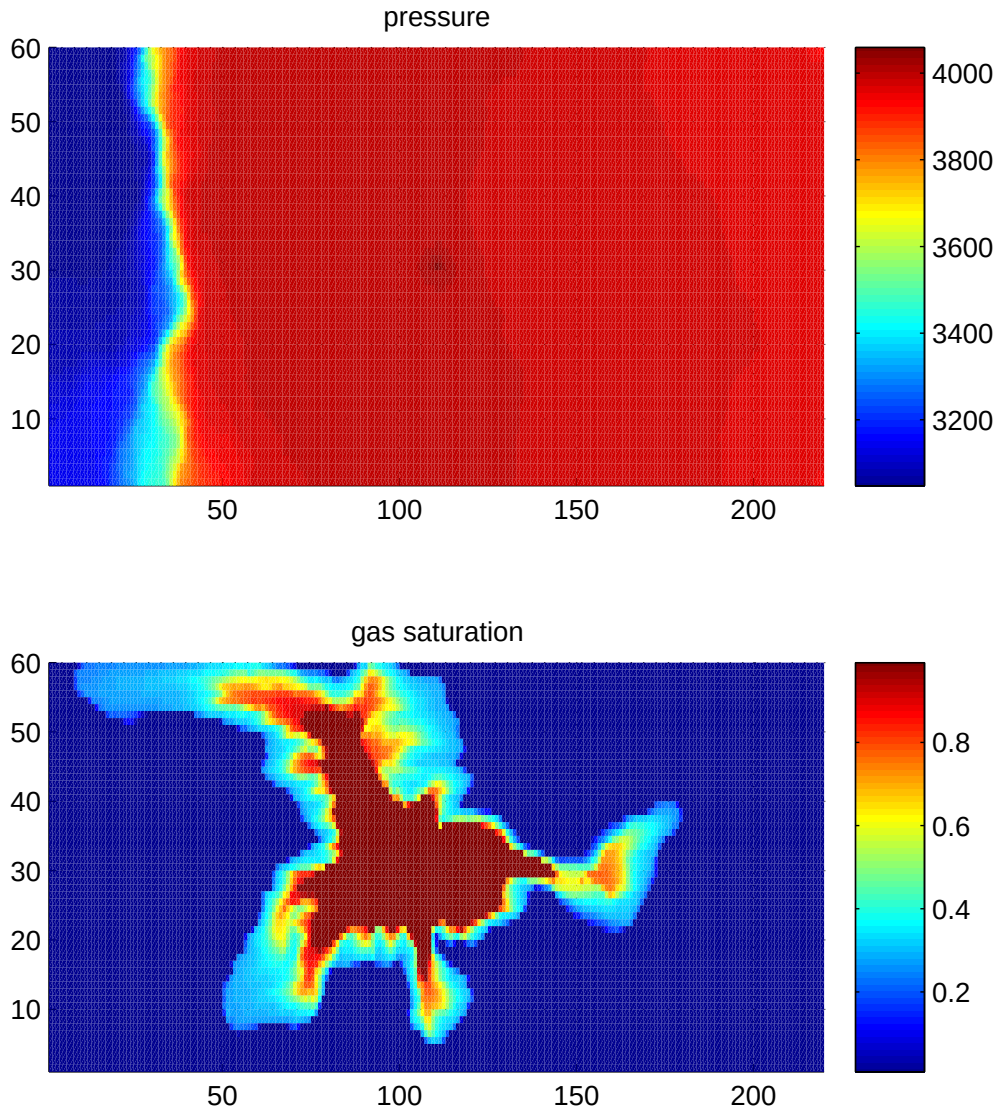


Figure 8.9: Pressure and gas saturation profile maps of AIM scheme at 4000 days

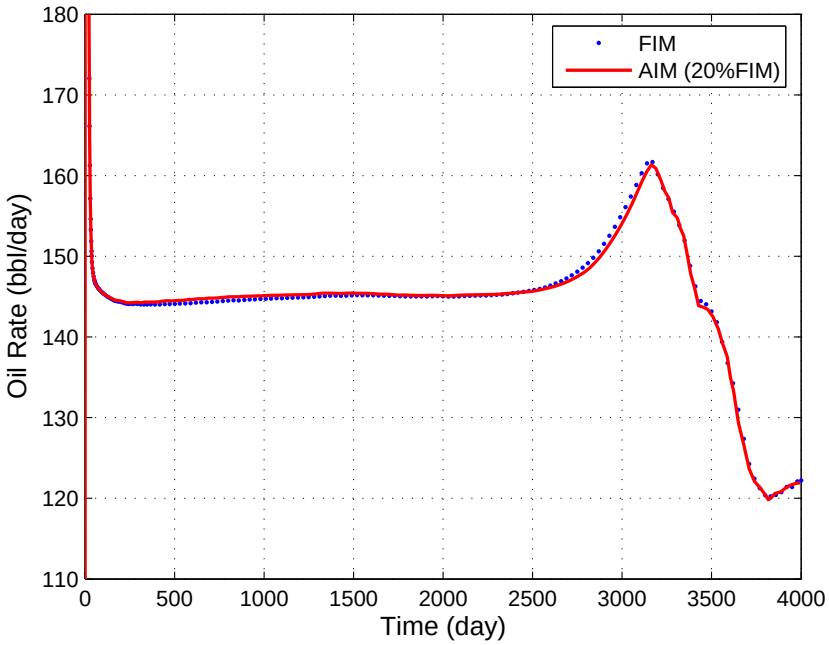


Figure 8.10: Oil rate history of Producer 2 with both schemes

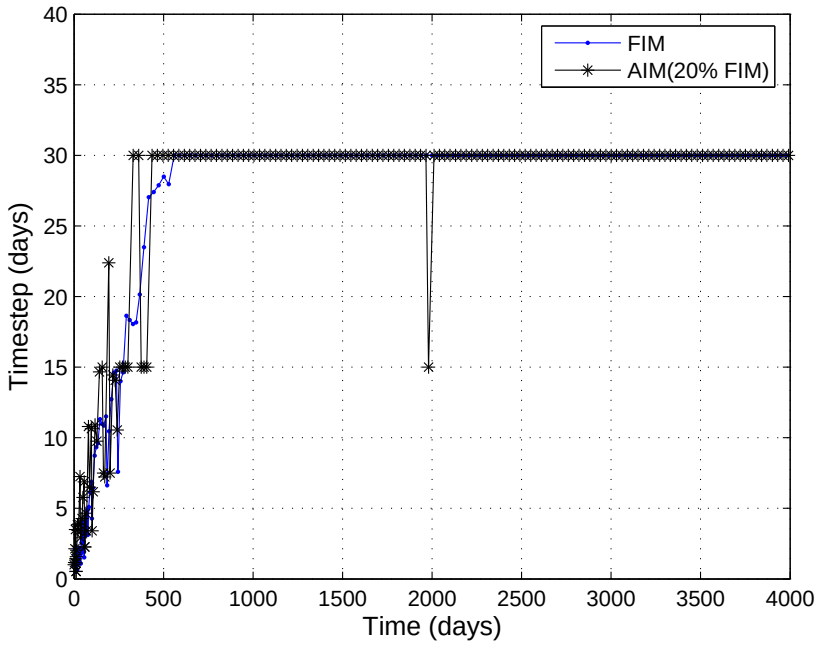


Figure 8.11: Timestep of AIM and FIM schemes for Case 2

IMPES/FIM	0.8/0.2	0.0/1.0
Solver	BGMRES	BGMRES
Preconditioner	CPR-BILU	CPR-BILU
Implicit variables	26400	79200
Timesteps	169	187
Newton Iterations	707	722
Solver Iterations	2298	2327
Pressure Iterations	2298	2327
Solver time	195.9	367.9
GMRES/MV	80.1	153.4
BILU factorization	23.4	63.6
BILU solution	23.3	50.4
Pressure decoupling	15.4	48.5
Pressure solution	53.7	52.0

Table 8.2: Performance of AIM and FIM schemes for Case 2

and 100% FIM. Recall that the model has about 12% fracture cells and about 24% matrix cells connected to fractures. These 36% of cells are potentially problematic. AIM30 has enough FIM gridblocks to cover all the fracture cells, but not enough for the matrix cells that are connected to fractures. AIM50 can cover both types of cells.

The simulation period is 30 days. The maximum allowed timestep size is 5 days. The timestep sizes of the different schemes are plotted in Figure 8.12. From the figure, we can see that the three curves overlap for early time. This is because very small timesteps are used to ensure that the Newton method converges. Later, AIM30 hits its CFL limit, and its timesteps are significantly lower than the other two schemes. The maximum timestep of AIM30 is about 0.44 days. AIM50 has larger timesteps, which are up to 1.3 days. In both AIM schemes, the timestep sizes are quite stable and build up gradually.

The FIM scheme has identical timesteps with AIM50 for the first four days. Later

on, the FIM scheme tries to use more aggressive timesteps, but most of these attempts fail to converge within a given number of Newton iterations. The simulator has to cut the timestep by half and restart the simulation from the last timestep. Because of this, the timestep curve of FIM has many oscillations in the later stages. Cutting a timestep means all previous Newton iterations are wasted. Hence it is a very costly event in a simulation run.

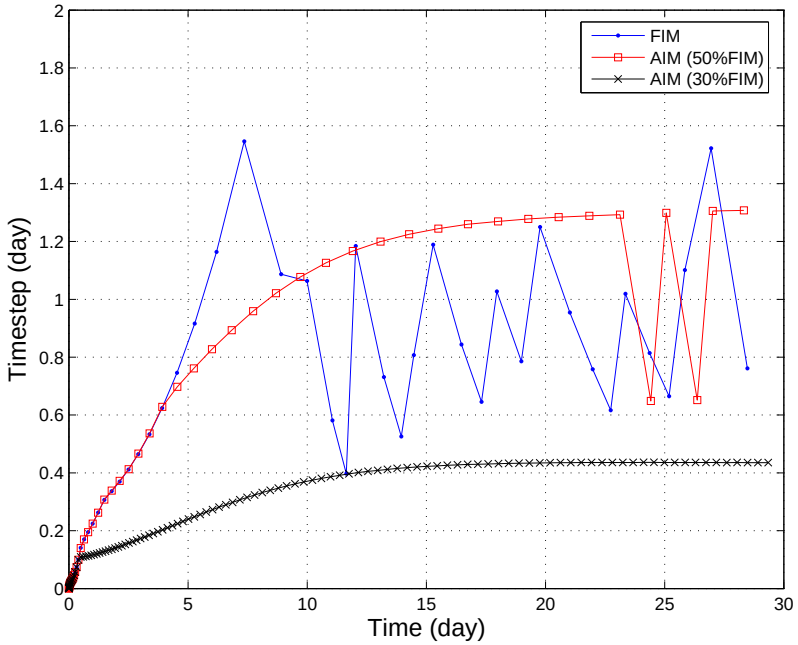


Figure 8.12: Timestep size of FIM and AIM schemes for Case 3

The iteration statistics and timing performance of the three schemes are shown in Table 8.3. From the upper part of the table, we can see that AIM50 has the least number of timesteps. Due to the timestep cuts, the FIM scheme uses three more timesteps compared to AIM50. AIM30 has much smaller timesteps, and it consequently needs many more timesteps to complete the simulation. Beside the number of timesteps, AIM50 also wins in terms of the number of Newton iterations, solver iterations, etc. In this table, the numbers in brackets are the wasted operations

due to timestep cuts. The AIM30 scheme has no timestep cuts; AIM50 has two and FIM has 34 timestep cuts. From the iteration statistics, it is not a surprise that AIM50 has the best timing performance. In fact, the lower part of the table shows that AIM50 has the best performance across the board. AIM50 is 3.2 times faster than FIM in solver time, and 1.7 times faster than AIM30. The oil production rates from the three runs are shown in Figure 8.13. The three curves are very close to each other.

IMPES/FIM	0.7/0.3	0.5/0.5	0.0/1.0
Solver	BGMRES	BGMRES	BGMRES
Preconditioner	CPR-BILU	CPR-BILU	CPR-BILU
Implicit variables	26400	35636	79200
Timestep	138 (0)	81 (2)	85 (34)
Newton iterations	1056 (0)	564 (30)	651 (510)
Solver iterations	5510 (0)	2776 (308)	3666 (4585)
Pressure iterations	5510 (0)	2776 (308)	3666 (4585)
Solver time	8167.7	4910.1	15779.3
GMRES/MV	2336.6	1526.2	3661.7
BILU factorization	601.1	418.6	1286.3
BILU solution	932.8	591.0	2572.6
Pressure decoupling	284.7	221.4	786.1
Pressure solution	4010.6	2153.0	4867.6

Table 8.3: Performance of AIM and FIM schemes for Case 3

8.6 Concluding Remarks

The data structures we developed and implemented in GPRS are flexible and efficient compared to the standard ones. The new data structures offer significant advantages in processing AIM Jacobian matrices. The speedup factors between the different schemes and preconditioner options are case sensitive. However, we can draw a few

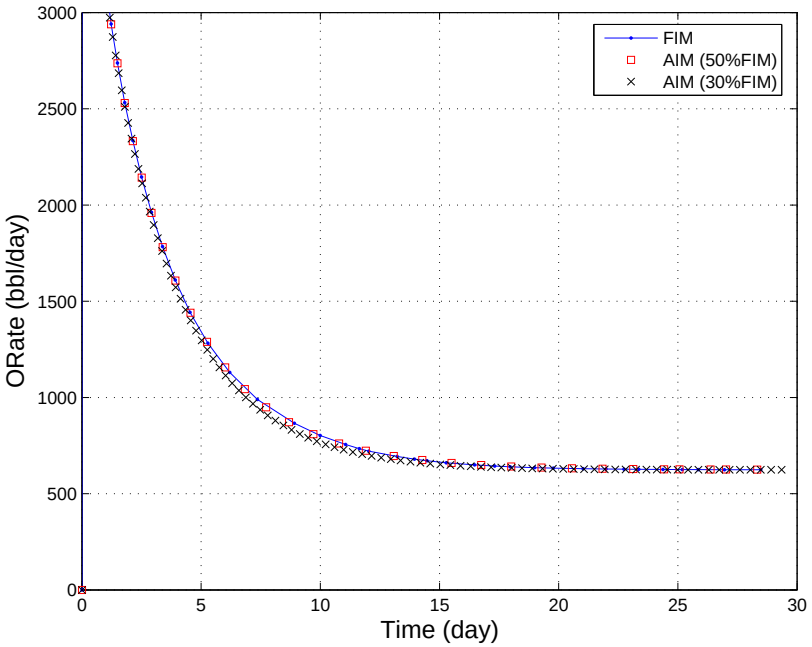


Figure 8.13: Production history of the producer in Case 3

general conclusions about AIM simulation:

- AIM is an effective approach for compositional simulation, which can reduce the matrix size significantly by using explicit treatment for most gridblocks, while taking timestep sizes comparable to FIM.

- With the new flexible and efficient multi-level block-based data structures, CPR is a much better preconditioner compared to ILU for AIM simulation. The block solvers and preconditioners can further speed up CPR for AIM simulation.

- For AIM simulation with large numbers of components and a high percentage of FIM, the pressure matrix of the first stage in CPR become less important as n_c increases. In this case, the second stage preconditioner (BILU or ILU) plays a more important role. Both GMRES and BILU benefit more from cache

utilization due to larger block sizes in the Jacobian.

Chapter 9

Conclusions and Future Work

9.1 Conclusions

The development of a general-purpose reservoir simulation framework for coupled systems of unstructured reservoir models and advanced wells is the subject of this dissertation. Stanford's General Purpose Research Simulator (GPRS) serves as the base for the new framework. GPRS is a research platform for reservoir simulation and related research (SUPRI-B/HW groups) at Stanford and elsewhere, and it has been extended and enhanced in the last few years by a number of researchers. In this work, we made significant contributions to GPRS, in terms of architectural design, extensibility, computational efficiency, and new advanced well modeling capabilities. Moreover, new robust and efficient linear solution algorithms for coupled reservoir-facilities systems were developed and implemented.

As a first step, we designed and implemented a new architectural framework, in which the facilities (man-made) are grouped and treated as a separate component,

which is at the same level as the reservoir (natural) component. This reflects the growing importance and complexity of facilities modeling, including accurate representation of multiphase flow in wellbores and pipe networks. A new 'solvers' component now sits at the same level as the reservoir and facilities models. Well-defined component interfaces were designed and implemented. As a result, the new framework is quite general and extensible, which reduces the cost of incorporating new models into GPRS significantly. Adding a new model into a simulator is generally a tough job, especially when a model relates to core components, e.g., changing the structure of the Jacobian matrices. With the new flexible reservoir simulation platform, the necessary effort (both in human resources and extent of code modification) for incorporating a new model is expected to be much smaller than that with the previous GPRS design. Instead of having to worry about the entire simulator and get overwhelmed by the vast details of the GPRS code base, researchers can work on specific components, or subcomponents, that allows them to quickly implement and test their research ideas. The extensibility of the new GPRS framework is demonstrated by incorporating several new facility models, namely, the Multisegment Well (MSWell) and capabilities for handling constraints on well groups.

While flexibility and extensibility of the simulator framework are the primary objectives, making sure that robustness and computational efficiency are at the leading edge has also been a major objective for this thesis. For that purpose, new data structures have been built specially for GPRS. These new data structures honor the separation of the reservoir and facilities components. Furthermore, these data structures were designed with the computational algorithms in mind. Specifically, we designed and implemented a multilevel sparse block (MLSB) data structure, which is consistent with the hierarchial structure in the new framework. The MLSB data structure is flexible and allows the use of other data structures as subcomponents.

This gives researchers freedom to develop their own discrete formulations and solution methods. This new data structure also simplifies the construction and assembly of Jacobian matrices. The pointer-based block matrix format takes advantage of both the standard Compressed Row Storage (CRS) data structure (which is widely used for sparse matrix computations) and the basic array-based data structure in GPRS, in which the data representing the discrete form of the conservation equations of the gridblocks is stored in diagonal and off-diagonal arrays.

The new data structures honor data encapsulation, and they are built with careful attention to vectorization and block-based data representation. This leads to efficient processing of the most important computational kernels (e.g. matrix-vector multiplication, dynamic updating of the Jacobian). We emphasize that all these data structures are fully compatible with the adaptive implicit method (AIM), which is a fundamental and highly desirable feature of GPRS.

The MSWell model, which has been extended significantly over the last few years, was previously developed as a separate code base for research into accurate modeling of multiphase flow in wellbores. As part of this work, the MSWell model was fully integrated into the new GPRS framework, and it may be seen as an example of a significant capability extension of a general-purpose simulator. There is a huge difference between a piece of separate code that implements a small component, and a stable option in a software package aimed at general-purpose reservoir simulation. So, as expected, a great deal of effort has been spent in order for the flow modeling capability of complex reservoirs and MSWells to be robust and computationally efficient. The MSWell capability in GPRS offers homogeneous and drift-flux flow models. The MSWell treatment can account for transient and wellbore storage effects. We also provide stable initialization, several well control methods, and highly effective linear solvers and preconditioners for the MSWell model.

GPRS has been extended to simulate coupled systems of (unstructured) reservoir models and advanced wells. As part of this effort, a new rigorous well-group treatment was proposed and implemented. This new model accounts for interactions between the wells in the group and allows for using one's favorite pipeline flow correlation. The well-group model benefits from the new flexible data structures of GPRS. For example, one can combine any available facilities object, such as standard and multisegment wells and mixed well groups that include both kinds of wells.

Advanced linear solution strategies were also developed for GPRS. We demonstrated the outstanding performance of the CPR preconditioner, with a number of large-scale, highly heterogeneous, unstructured cases. Based on the new block data structures, we developed and implemented block ILU preconditioners. These block preconditioners help further speed up the simulation. The two-stage CPR preconditioning approach is extended to handle systems with the MSWell model. In the first stage, an algebraic two-step reduction procedure is developed to construct a pressure matrix for the system with MSWells. In the second stage, the coupling terms in Jacobian between the reservoir and facilities are ignored, and a multilevel block ILU preconditioner is developed and used to handle the decoupled objects.

The data structures and linear solvers we developed are fully compatible with AIM. In fact, they were designed to be especially efficient in handling the dynamic Jacobian matrices associated with AIM simulation. We demonstrated that the CPR preconditioned GMRES solver based on the new framework is the best linear solution strategy for large-scale AIM simulation of unstructured reservoir models and advanced wells. We also showed that the block data structures and solvers have significant advantage over the original pointwise ones regarding efficiency and flexibility. With the new efficient data structures and linear solvers, AIM simulation outperforms FIM simulation for compositional simulation with large numbers of components.

Another significant contribution of this work is the development of a general discrete wellbore model, which we call GenWell. (This model is a stand-alone module). GenWell extends the MSWell treatment from perforations all the way to the surface. The GenWell model shares the advantages of the MSWell model, but it also allows us to model flow in surface pipeline networks. Unlike the standard MSWell model, GenWell accommodates complex pipeline topology, such as general branching, loops and multiple exits. In the GenWell model, wellbores and pipeline networks are discretized into various types of segments. Nodes and connections are defined based on the segments, and the discrete system is abstracted as a graph. Using small, yet representative, examples, we showed that the GenWell model can handle multiphase transient flow in wellbore and pipeline systems with loops, junctions, multiple exits, and arbitrary flow directions.

9.2 Future Work

GPRS is now an efficient AIM compositional simulator for unstructured reservoir models with advanced wells. With our work on facilities modeling and suggested future extensions, GPRS can perform fully coupled reservoir-facilities simulation efficiently. Here, we list few high priority research directions related to GPRS:

- Implement the drift-flux flow model in the GenWell framework.
- Develop and implement special segments to model surface equipment in pipeline networks.
- Integrate the GenWell model into GPRS.
- Implement nested well-groups.

- Develop and implement other well models in the GPRS framework.
- Develop efficient linear solution strategies for systems with well groups and the GenWell model.
- Develop parallel solvers and preconditioners based on the CPR method.
- Implement other multiphase flow correlations for well and surface networks.

With the new design of GPRS, research in the suggested areas can be conducted in a much more efficient manner than with the previous version of GPRS.

Nomenclature

α_p Phase holdup in segment, page 52

$\boldsymbol{A}$ A square matrix, page 112

$\boldsymbol{A}_p$ IMPES-like pressure matrix, page 131

$\boldsymbol{A}_{FIM}$ Fully implicit matrix, page 131

$\boldsymbol{A}_{AIM}$ Adaptive implicit matrix, page 171

$\boldsymbol{A}_{global}$ Global reservoir-facilities Jacobian matrix, page 36

$\boldsymbol{b}$ A right-hand-side vector, page 112

$\boldsymbol{b}$ A vector, page 36

$\boldsymbol{b}_f$ Facility subvector, page 36

$\boldsymbol{b}_r$ Reservoir subvector, page 36

$\boldsymbol{J}_{FF}$ Facility equations with respect to facility variables, page 29

$\boldsymbol{J}_{FR}$ Facility equations with respect to reservoir variables, page 29

$\boldsymbol{J}_{RF}$ Reservoir equations with respect to facility variables, page 29

$\boldsymbol{J}_{RR}$ Reservoir equations with respect to reservoir variables, page 29

$\boldsymbol{L}$ Lower triangle matrix, page 117

$\boldsymbol{M}$ A preconditioning matrix, page 115

$\boldsymbol{P}$ Column operation matrix for IMPES-like reduction, page 131

$\boldsymbol{P}_{AIM}$ Prolongation operation matrix for IMPES-like reduction from AIM matrix, page 171

$\boldsymbol{R}$ Row operation matrix for IMPES-like reduction, page 131

$\boldsymbol{R}_{AIM}$ Restriction operation matrix for IMPES-like reduction from AIM matrix, page 171

$\boldsymbol{U}$ Upper triangle matrix, page 117

$\boldsymbol{v}$ A vector, page 115

$\boldsymbol{v}^*$ A intermediate vector, page 116

$\boldsymbol{w}$ A vector, page 115

$\boldsymbol{x}$ A solution vector, page 112

$\boldsymbol{y}$ A unknown vector, page 116

Δh Height difference between neighbor perforations, page 43

Δh_i^{seg} Height of segment i, page 51

Δp_a Pressure difference caused by acceleration, page 51

Δp_f Pressure difference caused by friction, page 51

Δp_h Hydraulic pressure difference, page 51

λ_p Phase mobility in perforated cell, page 45

ρ_i^w Wellbore fluid density at ith perforation, page 43

ρ_i^{seg} Density of fluid mixture in segment i, page 51

ρ_p Phase density in perforated cell, page 45

ρ_p Phase density in the perforated reservoir cell, page 44

$\rho_{\overline{o}}$ Oil density in standard condition, page 45

$\boldsymbol{0}$ Zero matrix, page 147

$\boldsymbol{I}$ Identity matrix, page 147

A_i Area of cross section of segment i, page 52

A_{op} Derivative of accumulation term in oil equation with respect to pressure, page 128

d Diameter of segment, page 51

f_{tp} Friction factor, page 51

F_{ws} Derivative of flux term in water equation with respect to saturation, page 128

g Gravity constant, page 43

L_i Length of segment i, page 52

m_{in} The mass flow rate of the mixture through perforation, page 52

n_p Number of phases in reservoir condition, page 45

n_{perf} Number of perforations, page 45

p_i^w Wellbore pressure at ith perforation, page 43

p_i^{seg} Pressure of segment i, page 51

p_p Phase pressure at perforated cell, page 45

p_{ref} Well reference pressure, page 45

p_{target} User specified well operating pressure., page 45

q_p^w Phase volumetric rate through a perforation, page 44

$q_{\bar{o}}$ User specified oil production rate., page 45

V_m The velocity of fluid mixture in segment, page 51

V_p in-situ phase flow velocity in segment, page 67

V_{sp} Phase superficial velocity, page 52

WI_i Well index of perforation i, page 45

$x_{\bar{o},p}$ is mass fraction of oil conponent in phase p , page 45

Bibliography

[1] K. Aziz, L.J. Durlofsky, and H. Tchelepi. *Notes for Petroleum Reservoir Simulation.* Stanford University, 2006.

[2] K. Aziz and A. Settari. *Petroleum Reservoir Simulation.* Applied Science Publishers, 1979.

[3] K. Aziz and T.W. Wong. Considerations in the development of multipurpose reservoir simulation models. *Proceedings of the 1st International Forum on Reservoir Simulation*, September 1988.

[4] G.A. Behie. Practical considerations for incomplete factorization methods in reservoir simulation. *SPE 12263, Proceedings of the 7th SPE Symposium on Reservoir Simulation, San Francisco, CA*, November 1983.

[5] W.L. Briggs, V. Henson, and S.F. McCormick. *A Multigrid Tutorial, 2nd Edition.* SIAM, 2000.

[6] T.J. Byer. *Preconditioned Newton Methods for Simulation of Reservoir with Surface Facilities.* PhD thesis, Stanford University, 2000.

[7] T.J. Byer, M.G. Edwards, and K. Aziz. Preconditioned Newton methods for fully coupled reservoir and surface facility models. *SPE 12263, presented at*

the SPE Annual Technical Conference and Exhibition, New Orleans, Louisiana, September 1998.

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

[9] H. Cao and K. Aziz. Performance of IMPSAT and IMPSAT-AIM models in compositional simulation. *SPE 77720, presented at the SPE Annual Technical Conference and Exhibition, San Antonio, Texas*, 29 September 29-October 2 2002.

[10] H. Cao, H.A. Tchelepi, J.R. Wallis, and H. Yardumian. Parallel scalable CPR-type linear solver for reservoir simulation. *SPE 96809, presented at the SPE Annual Technical Conference and Exhibition, Dallas, Texas*, October 2005.

[11] Y. Chen. Modeling gas-liquid flow in pipes: Flow pattern transitions and drift-flux modeling. Master's thesis, Stanford University, 2001.

[12] M.A. Christie and M.J. Blunt. Tenth SPE comparative solution project: A comparison of upscaling techniques. *SPE 72469, SPE Reservoir Evaluation and Engineering*, 4(4):308–317, August 2001.

[13] B.K. Coats, G.C. Fleming, J.W. Watts, Rame M., and G.S. Shiralkar. A generalized wellbore and surface facility model, fully coupled to a reservoir simulator. *SPE 87913, SPE Reservoir Evaluation and Engineering*, 7(2):132–142, April 2004.

[14] K.H. Coats. IMPES stability: Selection of stable timesteps. *SPE 84924, SPE Journal*, 8(2):181–187, June 2003.

[15] Landmark Graphic Corporation. *BlitzPak User's Guide*, 1998.

[16] J.W. Demmel. *Applied Numerical Linear Algebra*. SIAM, 1997.

[17] J Dongarra, A. Lumsdaine, R. Pozo, and K.A. Remington. *IML++ V.1.2, Iterative Methods Library Reference Guide*. National Institute of Standards and Technology, 1996.

[18] K.E. Du, S. Pai, J. Brown, R.M. Moore, and M. Simmons. Optimising the development of blake field under tough economic and environmental conditions. *SPE 64714, presented at International Oil and Gas Conference and Exhibition in China, Beijing, China*, November 2000.

[19] P.A. Forsyth and P.H. Sammon. Practical considerations for adaptive implicit methods in reservoir simulation. *J. Comp. Phys.*, 62:265–281, 1986.

[20] Schlumberger GeoQuest. *Eclipse Technical Description, Multi-Segment Wells*, 2005.

[21] A George and J.W. Liu. *Computer Solution of Large Sparse Positive Definite*. Prentice Hall Professional Technical Reference, 1981.

[22] B. Gong. *Effective Models of Fractured Systems*. PhD thesis, Stanford University, 2007.

[23] A.R. Hasan and C.S. Kabir. *Fluid Flow and Heat Transfer in Wellbores*. SPE, 2002.

[24] J.A. Holmes, T. Barkve, and O. Lund. Application of a multisegment well model to simulate flow in advanced wells. *SPE 50646, presented at the SPE European Petroleum Conference, The Hague, Netherlands*, 1998.

[25] Y. Jiang. Tracer flow modeling and efficient solvers for GPRS. Master's thesis, Stanford University, 2004.

[26] M. Karimi-Fard, L. J. Durlofsky, and K. Aziz. An efficient discrete fracture model applicable for general-purpose reservoir simulators. *SPE 79699, SPE Journal*, 9(2):227–236, 2004.

[27] K. Kaspersky. *Code Optimization: Effective Memory Usage*. A-List Publishing, 2003.

[28] F. Kwok. A block ILU(k) preconditioner for GPRS. Technical report, Stanford University, December 2004.

[29] W. Li, Z. Chen, R.E. Ewing, Huan G., and Li B. Comparison of the GMRES and ORTHOMIN for the black oil model in porous media. *International Journal for Numerical Methods in Fluids*, 48(5):501–519, 2005.

[30] K. Lim. *Simulation of Fractured Reservoir with Sorption and Applications to Geothermal Reservoir*. PhD thesis, Stanford University, 1995.

[31] K.T. Lim and K. Aziz. A new approach for residual and Jacobian array construction in reservoir simulators. *SPE 84228, SPE Computer Applications*, August 1995.

[32] C. Nogaret. Implementation of a network-based approach in an object oriented reservoir simulator. Master's thesis, Stanford University, 1996.

[33] J.S. Nolen. Treatment of wells in reservoir simulation. Technical report, July 1990.

[34] L. Ouyang. *Single Phase and Multiphase Fluid Flow in Horizontal Wells*. PhD thesis, Stanford University, 1998.

[35] D.W. Peaceman. Interpretation of well-block pressures in numerical reservoir simulation. *SPE Journal*, 18(3):183–194, June 1978.

[36] N. Petalas and K. Aziz. A mechanistic model for multiphase flow in pipes. *Journal of Canadian Petroleum Technology*, 39(6):43–55, 2000.

[37] R. Pozo, K.A. Remington, and A. Lumsdaine. *SparseLib++ v. 1.5 Sparse Matrix Class Library Reference Guild*, 1986.

[38] H. Ritzdorf. *Readme to AMG Package*. GMD Software, 1991.

[39] T.F. Russell. Stability analysis and switching criteria for adaptive implicit methods based on the CFL condition. *SPE 18416, proceedings of the 10th SPE Symposium on Reservoir Simulation, Houston, TX*, February 1989.

[40] Y. Saad. Sparskit: a basic tool kit for sparse matrix computations. Technical report, Research Insititute for Advanced Computer Science, May 1990.

[41] Y. Saad. ILUT: a dual threshold incomplete *LU* factorization. *Numerical linear algebra with applications*, 1(4):387–402, 1994.

[42] Y. Saad. *Iterative Methods for Sparse Linear Systems (1st edition)*. PWS Publishing, 1996.

[43] Y. Saad and M.H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7(3):856–869, 1986.

[44] D.J. Schiozer. *Simultaneous Simulation of Reservoir and Surface Facilities*. PhD thesis, Stanford University, 1994.

[45] D.J. Schiozer and K. Aziz. Use of domain decomposition of simultaneous simulation of reservoir an surface facilities. *SPE 27876, presented at the SPE Western Regional Meeting, Long Beach, CA*, March 1994.

[46] H. Shi, J.A. Holmes, L.R. Diaz, L. J. Durlofsky, and K. Aziz. Drift-flux parameters for three-phase steady-state flow in wellbores. *SPE 89836, SPE Journal*, 10(2):130–137, 2005.

[47] H. Shi, J.A. Holmes, L. J. Durlofsky, K. Aziz, L.R. Diaz, B. Alkaya, and G. Oddie. Drift-flux modeling of two-phase flow in wellbores. *SPE 84228, SPE Journal*, 10(1):24–33, 2005.

[48] T.W. Stone, J. Bennett, D.H.S. Law, and Holmes J.A. Thermal simulation with multisegment wells. *SPE 78131, SPE Reservoir Evaluation and Engineering*, 5(3):206–218, 2002.

[49] K. Stueben. Algebraic multigrid (AMG): Experiences and comparisons. *proceedings of the International Multigrid Conference*, 1983.

[50] K. Stueben. An introduction to algebraic multigrid. *Appendix in book 'Multigrid'*, pages 413–532, 2001.

[51] K. Stueben and T. Clees. *SAMG User's Manual*. Fraunhofer Institute SCAI, 2003.

[52] S. Verma. *Flexible Grids for Reservoir Simulation*. PhD thesis, Stanford University, 1996.

[53] J.R. Wallis. Incomplete Gaussian elimination as a preconditioning for generalized conjugate gradient acceleration. *SPE 12265, presented at the 7th SPE Symposium on Reservoir Simulation, San Francisco, CA*, 1983.

[54] J.R. Wallis, R.P. Kendall, T.E. Little, and J.S. Nolen. Constrained residual acceleration of conjugate residual methods. *SPE 13536, presented at the 8th SPE Symposium on Reservoir Simulation, Dallas, TX*, 1985.

[55] T.W. Wong. Compositional and black oil modeling with emphasis on the volume balance method. Master's thesis, Stanford University, 1988.

[56] N. Zuber and J. A Findlay. Average volumetric concentration in two-phase flow systems. *Journal of Heat Transfer*, 87:453–468, November 1965.