10

Numerical Modeling of Fractured Unconventional Oil and Gas Reservoirs

OLUFEMI OLORODE, BIN WANG, AND HARUN UR RASHID

Abstract

The significant increase in the contribution of unconventional oil and gas reservoirs to the world's total petroleum production has led to a corresponding interest in the study of these resources over the last decade. Various researchers have focused on the study of storage and transport mechanisms that are unique to these naturally fractured unconventional resources. In this chapter, we show how to extend the MATLAB Reservoir Simulation Toolbox (MRST) to model these physical mechanisms using a `shale` module we have developed. Some of the features of this module include the modeling of sorption, molecular diffusion, stress-sensitive permeability, and realistic fractures in any orientation. This chapter starts with a discussion of the design of the `shale` module. We then present the governing equations for compositional simulation and show how to use the `hfm` and `compositional` modules in MRST to perform a compositional simulation in a fractured reservoir. To demonstrate the practicality of the `shale` module, we model an Eagle Ford shale oil reservoir with hundreds of natural fractures. We conclude this chapter with a discussion of how to implement certain storage and transport mechanisms that are unique to shale oil/gas reservoirs.

10.1 Introduction

Unlike conventional petroleum reservoirs, unconventional oil and gas (UOG) reservoirs have very low matrix permeability and porosity. Although the terms shale gas/oil reservoirs and UOG reservoirs are typically used interchangeably, they include organic-rich source rocks with considerable amounts of mudstone, siltstone, or carbonate. These source rocks are known to be made up of mostly inorganic matter (such as quartz, clay, pyrites), in addition to the organic matter (called kerogen). The total organic carbon contents of these source rocks range from approximately 1% to 12% by mass.

https://doi.org/10.1017/9781009019781.016 Published online by Cambridge University Press

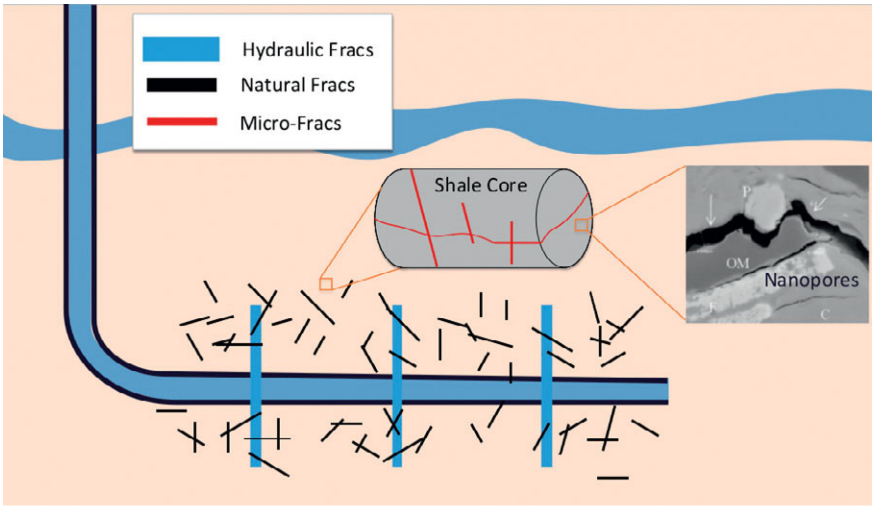

Figure 10.1 This sketch illustrates the presence of fractures at multiple scales in organic-rich source rocks. The man-made hydraulic fractures are on the order of millimeters and tens/hundreds of meters in aperture and length, respectively, whereas the microcracks are on the order of micrometers in length. The natural fractures are at an intermediate scale between these two and could be on the order of meters in length.

Pore-size distributions from UOG reservoirs show that the pores in these source rocks could be less than 2 nm in the shale matrix and could be on the order of micrometers in the largest pores and natural fractures [16]. UOG reservoirs are also known to contain fractures across multiple scales, as illustrated in Figure 10.1. Microfractures (or microcracks) and larger-scale natural fractures are both naturally occurring cracks or openings in the organic-rich source rocks, whereas hydraulic fractures are man-made. Microfractures are typically developed from internal processes (such as crack nucleation due to the maturation of kerogen or dehydration of clay) or external processes such as tectonic loading [28].

Natural fractures are much larger cracks that occur within a rock and are typically characterized by a lack of displacement across the crack surface. Even though the terms "fractures" and "faults" are sometimes used interchangeably, a fault differs from a fracture in that there is displacement along the crack surface in faults but there is no rock displacement in fractures. The orientation of each fracture in a fractured rock is controlled by the prevailing stress states when the fracture was created. These stress states typically evolve over many years, leading to different natural fracture orientations [31]. Some of these natural fractures get sealed by the accumulation of fine-grained particles (called cementing materials) within the fractures and are referred to as "sealing fractures." The other natural fractures with little or no sealing materials are more conductive to the flow of reservoir fluids and are referred to as "conductive fractures."

The reservoir fluids (oil, gas, and water) we produce from UOG reservoirs are stored in void spaces in the rock matrix, microcracks, and natural fractures. Although fluids tend to flow through the path of least resistance (the largest pores and hundreds/thousands of conductive natural fractures), a considerable amount of the reservoir fluids is stored in the nanoscale pores. This, coupled with the low connectivity between larger pores and fractures, results in the low matrix permeability and consequently low recovery from UOG reservoirs. To produce commercially from these reservoirs, we typically employ the combination of horizontal wells with multistage hydraulic fracturing.

Hydraulic fractures are large-scale fractures that are artificially induced by pumping in a mixture of water and chemical additives (known as slickwater) at high pressures. They are typically created by plugging and perforating the horizontal wells in stages, starting from the toe of the well and progressing toward the heel of the well. After these fractures are initiated, we usually pump in proppants, which are solid particles dispersed in the fracturing fluid. These help to prevent the fractures from closing during production, when the pressure near the fracture surfaces drops and additional compressive stresses are induced. The solid proppants occupy a significant proportion of the void space in the hydraulic fractures, leading to a fracture porosity that is typically much less than 100%. These proppants also make the permeability of propped hydraulic fractures orders of magnitude lower than the values that would be obtained using the cubic law, which is applicable for cracks without proppants.

Hydraulic fractures facilitate the commercial production from UOG reservoirs by connecting the conductive natural fracture networks to the horizontal production wells. Figure 10.1 illustrates the relative length scales of the microcracks and natural and hydraulic fractures, which are on the scale of micrometers, meters, and tens or hundreds of meters, respectively. Their apertures are much smaller, with propped hydraulic fractures having apertures on the order of a millimeter. The apertures of the microcracks and natural fractures are less known but are expected to be much smaller than those of the hydraulic fractures. This contributes to the wide variation in the pore-size distribution of these UOG reservoirs, which have multiscale fractures and nanoporous source rocks.

The wide spread in the pore-size distribution of UOG reservoirs could lead to differences in the storage and transport mechanisms at different length scales. To simulate flow in these reservoirs, we typically modify the mass-balance equations in conventional petroleum reservoirs to account for the specific storage and transport mechanisms expected in these organic-rich source rocks. This, coupled with the multiphase and compositional nature of fractured UOG reservoirs, makes the modified governing equations very nonlinear and difficult to solve using analytical or semi-analytical methods [11]. Therefore, we focus only on the solution of these complex nonlinear equations using the numerical methods available in the MATLAB Reservoir Simulation Toolbox (MRST). For more details on the state-of-the-art analytical and semi-analytical methods for modeling UOG reservoirs, the interested reader is referred to chapter 10 of the Dynamic Data Analysis report [12].

In shale-gas reservoirs, we could have gas stored in the compressed form as free gas within the large pores (which is usually the only storage mechanism considered in conventional gas reservoirs) and in the sorbed state. The storage of gas molecules in the sorbed state refers to the adsorption of the gas molecules on the internal surfaces of the organic-rich source rock, as well as the dissolution of some gas molecules into the solid matrix. We usually consider sorption in the gas phase but not in the liquid phase because in gases the density of the sorbed fluid near the pore walls is much higher than the fluid density in the middle of the pores [6]. However, this is typically not the case in the liquid state, where oil molecules are much more closely packed even in the middle of the pores. Some of the models that have been proposed for sorption in shale-gas reservoirs include the Langmuir [18] and Brunauer–Emmett–Teller [3] isotherms, among others. The Langmuir isotherm assumes that only one layer of fluid is adsorbed on the pore walls, whereas the Brunauer–Emmett–Teller isotherm assumes that multiple layers of the fluid are adsorbed on the pore walls. The Langmuir is the most common sorption model because of the simplicity in its use of only two parameters.

Transport mechanisms in UOG reservoirs are also more complex than in conventional reservoirs. For instance, in conventional reservoirs, molecular diffusion (driven by the concentration gradient) is usually neglected because the high matrix permeability causes the advective mass transfer (driven by the pressure gradient) to dominate the mass transport mechanism. However, in unconventional reservoirs with very low matrix permeability, the contribution of the diffusive flux of each hydrocarbon component to the total mass flux could be significant. This chapter shows how to model the diffusion of multicomponent hydrocarbon mixtures using Fick's law.

In UOG reservoirs, we also need to account for the tendency of hydraulic and natural fractures to close because of the compressive stresses induced when pressure drops during production. As in Olorode et al. [26], the model presented in Guo and Liu [10] can be implemented in a coupled flow and geomechanics simulator and used to model the closing of the propped hydraulic fractures. On the other hand, the change in the permeability of a fractured rock can be modeled using the Gangi [7] model. This model uses a conceptual bed of nails to represent a naturally fractured rock, with the bed of nails representing the surface roughness or asperity of the natural fracture surfaces. It relates the permeability of a matrix and natural fracture system to the pore pressure and confining stresses acting to close the natural fractures. This chapter shows how to implement the Gangi model in MRST and provides an example that illustrates the effect of this permeability reduction on production.

This chapter starts with a presentation of the design of the `shale` module, which leverages the rapid prototyping features of MRST to facilitate the simulation of UOG reservoirs. Considering that most of the standard MRST modules were designed for conventional petroleum reservoirs, we discuss the functions, classes, and scripts we have added or modified to enable the simulation of UOG reservoirs. For instance, we provide additional codes that augment the `hfm` module from Chapter 9 to allow us use the projection-based embedded discrete fracture model (pEDFM) instead of the embedded discrete fracture model (EDFM), which has been shown to be inaccurate at low fracture conductivities [32]. We also provide additional codes that augment the `compositional` module to model shale mechanisms such as sorption, diffusion, and geomechanics.

In Section 10.3, we present the governing equations for compositional flow in conventional petroleum reservoirs. In a later section (Section 10.7), we will show how these equations are modified to include each of the shale mechanisms discussed in this chapter. Although compositional reservoir simulation is more computationally challenging than the black-oil simulation approach, we focus on compositional simulation so that the `shale` module can be used to model enhanced/improved oil recovery (EOR/IOR) by miscible CO_2 or lean-gas injection. EOR/IOR in unconventional oil reservoirs is an active research area because these reservoirs have very low recovery factors that are typically less than 10%. It is worth mentioning that the modeling of EOR/IOR processes in UOG reservoirs is complicated by the presence of multiscale natural fracture networks in these organic-rich source rocks.

To provide a broad perspective on the modeling of fractured reservoirs, we discuss the different groups of modeling approaches that have been used for fractured reservoirs. We then focus on three specific modeling approaches that will be demonstrated in this chapter. These include the full-dimensional modeling of the fractures using several three-dimensional (3D) cells to represent the fractures in a 3D reservoir model, the use of the EDFM disscussed in Chapter 9, and the use of the 3D pEDFM model proposed by Olorode et al. [27]. Considering the significance of natural and hydraulic fractures in these ultra-low-permeability reservoirs, a great part of this chapter focuses on the modeling of hydraulic and natural fractures using EDFM and pEDFM. We show that pEDFM is more accurate, using a UOG reservoir with hundreds of sealing natural fractures in addition to the conductive natural fractures in the reservoir.

In Section 10.4, we systematically show how to use the hfm and compositional modules to model the production from a fractured compositional reservoir. For simplicity and computational speed, the case presented in this section only models a two-component mixture (10% CO_2 and 90% n-decane) with three fractures using EDFM. To model more realistic fractured reservoirs with hundreds or thousands of fractures in arbitrary orientations, we use the Alghalandis Discrete Fracture Network Engineering (ADFNE) code [1], an open-source MATLAB code to generate a stochastic fracture network distribution, in Section 10.5. Considering that there is currently no technology available to find the location, orientation, and dimension of all of the natural fractures in the subsurface, several realizations of the fracture network can be generated and simulated to evaluate the effect of the uncertainties in these fracture properties.

Section 10.6 shows how to model a representative Eagle Ford shale oil reservoir using the 3D pEDFM implemented in the shale module. We provide the eagleFord.m and eagleFordEDFM.m scripts (in the examples folder of the shale module) to regenerate the results presented in this section. The eagleFord.m script uses the 3D pEDFM functionality in the shale module, whereas eagleFordEDFM.m uses EDFM in the hfm module and is provided only for comparison. In both cases, we use the representative Eagle Ford compositional fluid given in Yu et al. [37]. In the final section of this chapter, we discuss how to implement three physical mechanisms that are unique to shale-gas reservoirs, namely, sorption, diffusion, and geomechanics. For each of these mechanisms, we present the corresponding mathematical model and modified governing mass-balance equation. In a tutorial style, we discuss the steps required to implement each of these models in MRST. Each subsection on these mechanisms ends with simulation results that illustrate the effect of the mechanism on production. We have also provided sorption.m, diffusion.m, and gangi.m scripts to facilitate the reproduction of the results presented in these subsections. They can be found in the examples folder of the shale module, which is discussed in the next section.

10.2 Shale Module

This section discusses the features and design of the shale module. Figure 10.2 shows that all of the codes in this module are contained within six folders. The design of the shale module is such that it depends on the compositional and hfm modules, which are discussed in Chapters 8 and 9, respectively. The module extends a few functions and classes from the original compositional module (which are stored in the compositionalFns folder) to facilitate the modeling of physical mechanisms unique to UOG reservoirs. The NatVarsShaleModel class extends

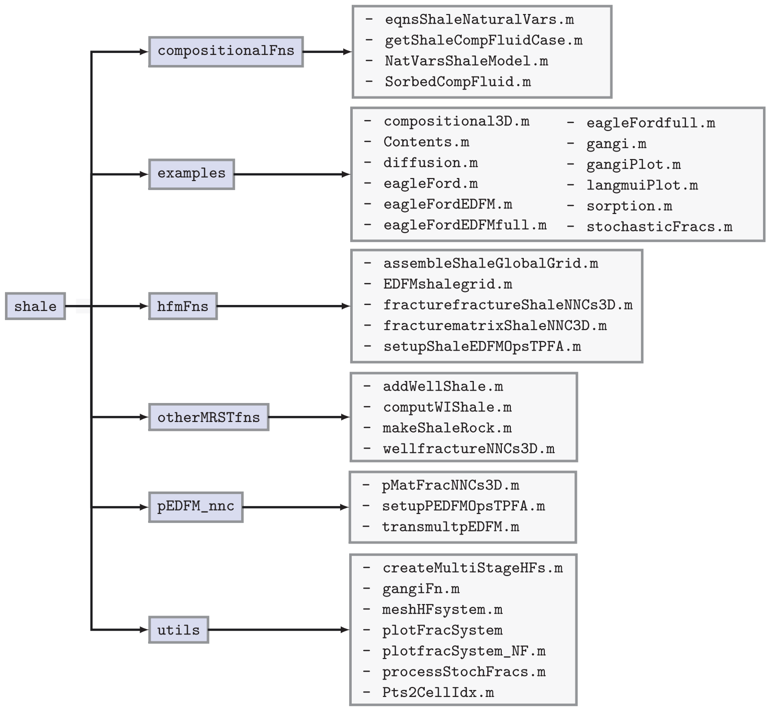

Figure 10.2 This data diagram shows the design of the `shale` module, as well as the functions and classes modified from the `hfm` and `compositional` modules.

the functionalities provided in the `NaturalVariablesCompositionalModel` class in the `compositional` module to model the shale mechanisms outlined in the previous section. The `NatVarsShaleModel` class uses its `eqnsShaleNaturalVars` method to model the equations that govern flow in UOG reservoirs. Although the examples presented in this chapter use the natural variables approach, the overall composition approach can also be used to model these shale mechanisms by extending the `OverallCompositionCompositionalModel` class instead of the natural variables class.

The `shale` module also depends on the module for EDFM and provides modifications of some `hfm` functions in the `hfmFns` folder. The functions that are unique to our 3D projection algorithm discussed in Subsection 10.3.2 are stored in the `pEDFM_nnc` folder. Considering that most of the shale mechanisms involve some modification of the `setupEDFMOperatorsTPFA` function in the `hfm` module, we created `setupShaleEDFMOpsTPFA` and `setupShalePEDFMOpsTPFA` to facilitate the integration of these additional physical mechanisms with the EDFM and pEDFM fracture modeling approaches, respectively. These two functions call `setupEDFMOperatorsTPFA`, before making the additional changes required to implement the shale mechanisms in EDFM and pEDFM. In the next listing, we show how we modify the `s` field returned from `setupEDFMOperatorsTPFA` to make the discrete operators compatible with the 2019b version of MRST:

```
s = setupEDFMOperatorsTPFA(G, G.rock, tol);
n = size(s.N,1);
C = (s.C)';  % transpose it only once for speed
if n == 0
    s.AccDiv = @(acc, flux) acc;
    s.Div = @(x) zeros(G.cells.num, 1);
else
    s.AccDiv = @(acc, flux) acc + C*flux;
    s.Div = @(x) C*x;
end
```

The `examples` folder contains the MATLAB scripts that can be used to regenerate all of the results presented in this chapter. The `sorption.m`, `diffusion.m`, and `gangi.m` scripts demonstrate how we model these three shale mechanisms. As shown in Section 10.7, they all use the `shaleMechanisms` field to specify the shale mechanism to be modeled. The `otherMRSTfns` folder contains modifications to functions taken from other parts of MRST, and `utils` contains utility functions that are used in the example scripts. To run these examples smoothly, the user is advised to first add the `shale` module and its subfolders to the MATLAB path.

The `shale` module also depends on the use of ADFNE [1], an open-source toolkit for stochastic fracture network generation. ADFNE requires the `Statistics & Machine Learning Toolbox` in MATLAB, so we provide the `EagleFordEDFM.mat` and `StochasticFracs.mat` files generated with ADFNE to allow users reproduce our simulation results without installing any toolboxes. The reader is also encouraged to explore the use of the discrete fracture network (DFN) generator in the `hfm` module, as discussed in Chapter 9. We discuss stochastic fracture modeling and its application in a fractured reservoir simulation in Sections 10.5 and 10.6. We also provide Table 10.1 to help readers identify the sources of the different functions discussed throughout this chapter.

To lay a foundation for the discussion of the physical mechanisms implemented in these functions, the next section discusses the governing equations for compositional flow in conventional petroleum reservoirs. Subsequent sections will show how these equations are modified to account for the storage and transport mechanisms expected in UOG reservoirs.

Table 10.1 *Location of all functions discussed.*

hfm	shale	ADFNE
EDFMgrid	processStochFracs	DFN
fracturematrixNNC3D	plotfracSystemNF	Field
fracturefractureNNCs3D	createMultiStageHFs	Orientation
setupEDFMOpsTPFA	setupPEDFMOpsTPFA	Stereonet
	wellfractureNNCs3D	
	pMatFracNNCs3D	
	makeShaleRock	

10.3 Compositional Flow and Modeling of Fractured Reservoirs

This section starts with a presentation of the governing equations for compositional flow in conventional petroleum reservoirs. We summarize how these equations are discretized and solved in MRST and provide appropriate references for further details on the numerical solution procedure employed in MRST. We also discuss how the discretized form of the governing equations are modified to model fractured reservoirs with EDFM and pEDFM.

10.3.1 Governing Equations for Compositional Flow in Conventional Reservoirs

In this chapter, we consider a petroleum reservoir where the hydrocarborn fluids can exist only in either the liquid or the gas phase. The equation for the conservation of each hydrocarbon species i in the oil (liquid) or gas (vapor) phase is given as

$$\begin{aligned}\partial_t\left[\phi(\rho_l S_l X_l^i + \rho_v S_v X_v^i)\right] + \nabla\cdot(\rho_l X_l^i \vec{v}_l + \rho_v X_v^i \vec{v}_v)\\ - \left(\rho_l X_l^i q_l + \rho_v X_v^i q_v\right)/V = 0.\end{aligned} \tag{10.1}$$

If an aqueous phase (water) is present, its conservation equation can be written as

$$\partial_t(\phi\rho_w S_w) + \nabla\cdot(\rho_w \vec{v}_w) - \rho_w q_w / V = 0. \tag{10.2}$$

In these conservation equations, ϕ represents porosity, ρ_α and S_α respectively represent the mass density and saturation of phase α, and X_l^i and X_v^i are the mass fractions of each component i in the liquid and vapor phases, respectively. The division of the volumetric flow rates (q_α) by volume (V) ensures that all terms in the equation are dimensionally consistent. Subscripts w, l, and v, refer to the water, liquid, and vapor hydrocarbon phases. Thus, $\vec{v}_w$, $\vec{v}_l$, and $\vec{v}_v$ represent the Darcy velocity for the water, liquid, and vapor hydrocarbon phases, which is defined as

$$\vec{v}_\alpha = -K\frac{k_\alpha(s)}{\mu_\alpha}(\nabla p_\alpha - \rho_\alpha g \nabla z) = -K\lambda_\alpha(\nabla p_\alpha - \rho_\alpha g \nabla z). \tag{10.3}$$

Using the subscript α for the liquid and vapor phases, (10.1) can be rewritten in terms of X_α^i (which is the mass fraction of a component i in phase α) as follows:

$$\partial_t \left[\phi \sum_{\alpha=l,v} \rho_\alpha S_\alpha X_\alpha^i\right] + \sum_{\alpha=l,v} \nabla \cdot \left(\rho_\alpha X_\alpha^i \vec{v}_\alpha\right) - \sum_{\alpha=l,v} \rho_\alpha X_\alpha^i q_\alpha / V = 0. \tag{10.4}$$

Here, the three terms on the left-hand side of the equation (which are the accumulation, flux, and source/sink terms, respectively) are expressed as a sum over each phase, α. The capillary pressure definitions, as well as saturation and composition constraints (that is, saturations and compositions sum up to one) are also applied to the governing equations. The primary variables selected depend on the mathematical formulation used for the simulation. Although we can use either the natural variables formulation [4] or the overall composition variables formulation [5] in MRST as discussed in Chapter 8, we focus on the use of the natural variables in this chapter. This approach involves the use of pressure, phase saturation(s), and phase composition of each component as the primary variables. Voskov and Tchelepi [33] provides further details and other alternative formulations for compositional simulation, and Møyner and Tchelepi [23] gives more details on the governing equations, their discretization, and numerical solution procedure.

10.3.2 Modeling of Fractured Reservoirs in MRST

Depending on whether each individual fracture in a fractured reservoir is modeled or not, we can classify the modeling approach into three broad groups (see also discussion in Chapter 11):

1. **Effective medium models:** These fracture models represent a fractured system with an effective medium. They include the dual-porosity, dual-permeability, and multicontinuum models [30, 34]. They are computationally faster than the discrete models but are based on the assumption that the reservoir is densely fractured and has homogeneous fracture properties. This limits their application in UOG reservoirs, which could have a heterogeneous fracture distribution and fractures with very different lengths, apertures, permeabilities, orientations, etc.
2. **Discrete models:** These models account for each individual fracture in the reservoir. They include the full-dimensional model, discrete fracture model (DFM) [14, 15], EDFM [19], pEDFM [32], etc. The added flexibility of modeling each fracture explicitly makes these models more computationally expensive.

Considering the importance of modeling the multiscale fractures in UOG reservoirs accurately, we focus on these DFMs in this chapter. To clarify, even though we use the term "discrete fracture models" in this chapter, we also model the reservoir matrix.

3. **Hybrid models:** These fracture models combine some form of effective medium modeling with a DFM. Examples include the multiple subregion model [8], multilevel DFMs [9, 20], etc.

Full-Dimensional Modeling

This is the most direct approach to model fractures. It involves a full volumetric representation that subdivides each fracture into cells in the same way the matrix is subdivided into cells. Cells that make up the fractures are referred to as fracture cells, whereas those that make up the matrix are referred to as matrix cells. The properties (such as porosity and permeability) of all of the fracture cells are set to the corresponding property for that fracture. For a 3D reservoir simulation domain, the fracture cells are also three-dimensional. Therefore, for a vertical fracture that is parallel to the (y, z)-plane, the width of each fracture cell in the x-direction (Δx) is specified as the fracture aperture. The full-dimensional model is the most computationally expensive of all of the fracture models because it requires the largest number of n-dimensional cells (where n is the number of dimensions of the reservoir domain). In most cases, the large number of cells, the small fracture apertures, and their typically high permeabilities make such models computationally intractable.

Embedded Discrete Fracture Modeling

EDFM involves gridding the matrix and fractures independently of each other. For an n-dimensional model, the matrix cells will be n-dimensional, whereas the fracture cells are of dimension $n-1$. The lower dimensionality of the fracture cells and its independent gridding give the flexibility to embed one or more fracture cells within a matrix cell. To account for the flow of fluids between the matrix and fracture cells, the governing equations presented in Subsection 10.3.1 are modified to include an additional source/sink term, which is implemented like the non-neighboring source/sink terms in commercial reservoir simulators. Chapter 9 provides a detailed discussion on the EDFM and how it is implemented in the `hfm` module.

Although EDFM is more computationally efficient than the full-dimensional and discrete fracture models, it is limited by its inability to accurately capture the effects of low conductivity or sealing faults/fractures on flow. Although UOG reservoirs have low matrix permeabilities, it could be misleading to treat all natural fractures as having high conductivity given that many of these fractures are no longer active in the current stress state of the rock. By the critically stressed fault hypothesis, faults that are mechanically alive or active act as conduits for fluid flow, whereas inactive faults act as barriers to flow [38]. To address the inaccuracy of the EDFM in modeling low-conductivity fractures, Ţene et al. [32] developed the pEDFM, which is discussed in the next subsection.

Projection-Based Embedded Discrete Fracture Modeling

The pEDFM basically modifies EDFM by adding two more types of non-neighboring connections (NNCs). It involves projecting the fracture cells into some of its neighboring matrix cells and computing the additional transmissibilities introduced as a result of this projection. Jiang and Younis [13] presented an algorithm to find the matrix cells into which the fracture cells will be projected. Their algorithm only applies to 2D or extruded 2D (2.5D) systems, where all fractures have to be perfectly vertical. In Olorode et al. [27], we presented a 3D pEDFM algorithm that provides the capability of modeling fractures with any arbitrary orientation in 3D space. This algorithm was developed for structured hexahedral meshes for the matrix but could be extended for use with corner-point grids. The general idea in pEDFM is to project a fracture in a cell into three of the six neighbors of the matrix cell in which the fracture is located. The matrix cell that hosts the fracture is referred to as the "host matrix cell," whereas the neighboring cells into which the fracture is projected are referred to as the "projection matrix cells." Fracture cells within a host matrix cell are referred to as "interior fracture cells." In the rare cases where fractures lie at the interface between two matrix cells, pEDFM simplifies into the DFM [32].

Figure 10.3 shows the flowchart for the 3D pEDFM algorithm, as presented in Olorode et al. [27]. It starts with a loop that estimates the distances between all interior fracture centroids and the six faces of their host matrix cells. Because these are hexahedral grids, the algorithm compares each pair of distances in the same spatial direction. For instance, we chose one projection face out of either the left or right face (in the x-direction). Depending on the computed values of the six distances, we could have a total of four possible scenarios. In all four cases, whenever a fracture cell is closer to one of a pair of faces in more than one spatial direction, the closer of the pair of faces is selected. We then compute the area that the fracture projects on each of the selected projection faces (called projection area) and its corresponding transmissibility (called projection transmissibility). We summarize each of these four cases here but refer to Olorode et al. [27] for more details and graphical examples of these cases.

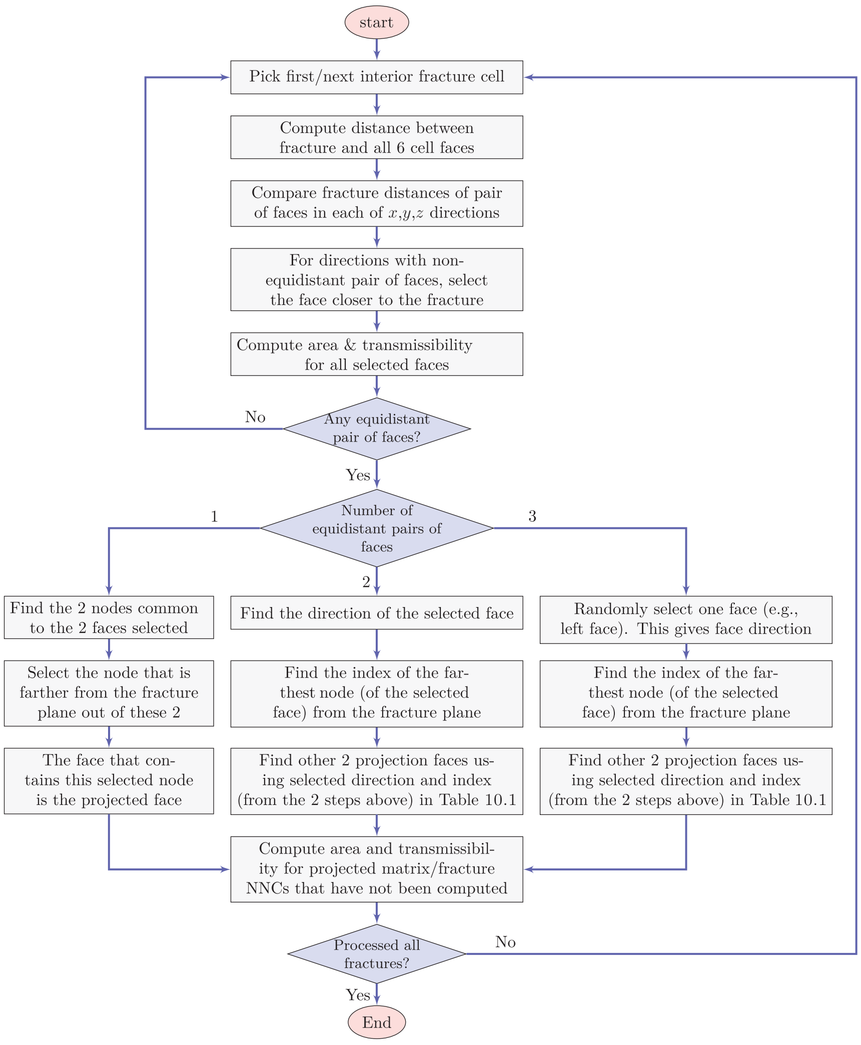
start
Pick first/next interior fracture cell
Compute distance between fracture and all 6 cell faces
Compare fracture distances of pair of faces in each of x,y,z directions
For directions with non-equidistant pair of faces, select the face closer to the fracture
Compute area & transmissibility for all selected faces
Any equidistant pair of faces?
No
Yes
Number of equidistant pairs of faces
1
2
3
Find the 2 nodes common to the 2 faces selected
Select the node that is farther from the fracture plane out of these 2
The face that contains this selected node is the projected face
Find the direction of the selected face
Find the index of the farthest node (of the selected face) from the fracture plane
Find other 2 projection faces using selected direction and index (from the 2 steps above) in Table 10.1
Randomly select one face (e.g., left face). This gives face direction
Find the index of the farthest node (of the selected face) from the fracture plane
Find other 2 projection faces using selected direction and index (from the 2 steps above) in Table 10.1
Compute area and transmissibility for projected matrix/fracture NNCs that have not been computed
Processed all fractures?
No
Yes
End

Figure 10.3 The proposed algorithm automatically determines the three neighboring cells, whose transmissibilities and connection areas need to be modified in 3D pEDFM.

– **No pair of equidistant faces in any spatial direction:** When a fracture cell is closer to one of a pair of faces in all three directions, we select the three projection cells (based on proximity to the fracture centroid). We then compute the projection areas and transmissibilities for all three selected projection faces using the equations presented in the next section. The algorithm then proceeds to the next fracture cell.

Table 10.2 *Selection of remaining two faces based on a combination of the node index and the direction of the only selected face.*

Node index	Face 1	Face 2	Face 1	Face 2	Face 1	Face 2
1	Front	Bottom	Left	Bottom	Left	Front
2	Back	Bottom	Left	Top	Right	Front
3	Back	Top	Right	Top	Right	Back
4	Front	Top	Right	Bottom	Left	Back

– **One equidistant pair of faces in one of the three spatial directions:** Because there is only one equidistant pair of faces, we readily determine two projection faces based on the proximity to the fracture centroid and compute their corresponding projection areas and transmissibilities. We then find which of the two nodes common to the selected projection faces is farther from the fracture plane and select the third projection face as the one that contains this shared node. The last step in this case is to compute the area and transmissibility for the third projection face and continue with the next interior fracture cell.
– **Two equidistant pairs of faces:** We determine one projection face based on the proximity to the fracture centroid and compute its corresponding projection area and transmissibility. We then find which of the four nodes of this selected projection face is farthest from the fracture plane and combine this with the direction of the selected face to select the other two projection faces. We use Table 10.2 to facilitate the selection of the projection faces and ensure that they do not meet along the path of the fracture plane; see Olorode et al. [27]. For each cell, the faces are ordered from face 1 to face 6, representing the left, right, front, back, top, and bottom faces, respectively. The algorithm continues with the computation of the area and transmissibility for these two newly selected (second and third) faces before proceeding to the next interior fracture cell.
– **Three equidistant pairs of faces:** We randomly select one face as the projection face and compute its projection area and transmissibility. The rest of the algorithm in this case then becomes identical to that of the previous case with two equidistant pairs of faces.

The algorithm discussed in this section has been implemented as a function named `pMatFracNNCs3D` within the `shale` module. This module leverages the EDFM functionality that is already built into the `hfm` module. The additional code needed to implement the 3D pEDFM involves the computation of the transmissibilities for two distinct groups of NNCs that are absent in EDFM. These two pEDFM NNCs are discussed in the next subsection.

10.3.3 The pEDFM Transmissibilities

In this subsection, we discuss the two additional NNC transmissibilities that are required in the pEDFM model. These are the transmissibilities between projection matrix cells and fracture cells (referred to as the projection matrix/fracture transmissibilities) and those between projection matrix cells and host matrix cells (referred to as the projection matrix/matrix transmissibilities).

Projection Matrix/Fracture Transmissibility The projection matrix/fracture transmissibility (T_{pMF}) is expressed as [32]

$$T_{\mathrm{pMF}} = \frac{A_{\perp x} K_{\mathrm{pMF}}}{d_{\mathrm{pMF}}}, \tag{10.5}$$

where

$$K_{\mathrm{pMF}} = \frac{K_{\mathrm{pM}} K_f}{K_{\mathrm{pM}} + K_f}. \tag{10.6}$$

Here, K_{pMF} represents the harmonic average of the projection matrix and fracture cell permeabilities, d_{pMF} represents the distance between the centroid of the fracture and that of the projection cell, and $A_{\perp x}$ is the area of the fracture projection along each dimension. The projection matrix/fracture transmissibility is computed in the `computeNNCprojAreanTrans` function, which is provided in `pMatFracNNCs3D`.

Projection Matrix/Matrix Transmissibility The projection matrix/matrix transmissibility (T_{pMM}) is given as

$$T_{\mathrm{pMM}} = K\frac{A - A_{\perp x}}{\Delta \vec{x}_e}. \tag{10.7}$$

In (10.7), $\Delta \vec{x}_e$ refers to the cell sizes in all three spatial directions, A refers to the area of the face between the projection and the host matrix cells, and $A_{\perp x}$ refers to the projection area. When the interior fracture cell fully intersects the host matrix cell and is parallel to the interface between the projection and host matrix cells, the projection area obtains its maximum value and is equal to A. It reduces to zero as the orientation of the interior fracture cell becomes perpendicular to the interface between the projection and host matrix cells.

To simplify the computation of T_{pMM} in MRST, we take advantage of the standard matrix–matrix transmissibilities (T_{MM}) already computed as

$$T_{\mathrm{MM}} = K\frac{A}{\Delta \vec{x}_e}. \tag{10.8}$$

Combining this with (10.7) yields

$$T_{\mathrm{pMM}} = T_{\mathrm{MM}} \frac{A - A_{\perp x}}{A}. \tag{10.9}$$

The fractional term in this equation is implemented as a transmissibility multiplier, as shown in the last couple of lines in the `transmultpEDFM.m` script:

```
oldArea = newarealist(globalFaceIdx);  % denominator in eq (9)
newArea = oldArea - projectedArea;     % numerator in eq (9)
newArea(newArea<tol) = 0;              % zeros out very small areas
newarealist(globalFaceIdx) = newArea;  % broadcasting numerator to T array
transmultp = newarealist./oldarealist; % fraction in eq (9)
```

The transmissibility multiplier computed in `transmultpEDFM` is then multiplied with the EDFM transmissibility multiplier that is computed in the `transmultEDFM` function of the `hfm` module. This is implemented in `setupPEDFMOpsTPFA` as shown:

```
T = getFaceTransmissibility(G, rock, opt.deck);
transmultMFM = transmultEDFM(G,tol);   % EDFM NNCs already implemented
transmultpMM = transmultpEDFM(G,tol);  % pEDFM update: compute T_pM-M mults
T = T.*transmultMFM.*transmultpMM;     % pEDFM update: using the multipliers
s.T_all = T; % T_all will not contain nnc transmissibilities
T = [T; G.nnc.T]; % modified line
T = T(intInx);
```

The next section shows how to perform a compositional simulation of fractured reservoirs using a combination of both the `compositional` and `hfm` modules.

10.4 EDFM and Compositional Simulation in MRST

The objective of this section is to provide a tutorial on how to perform a simple compositional simulation of a reservoir with only three fractures. The idea is to provide the reader with the fundamentals required to simulate more complex and realistic compositional fractured reservoirs. Figure 10.4 presents the simplistic scenario to be simulated in this section. It shows the location of a water injection and a production well at two diagonally opposite edges of the simulation domain. The injection well injects 100% water at a constant rate, whereas the production well produces the reservoir fluid at a constant flowing bottom-hole pressure. The hydrocarbon fluid simulated is a two-component mixture of 90% n-decane and 10% CO_2. To facilitate further compositional studies of recovery processes

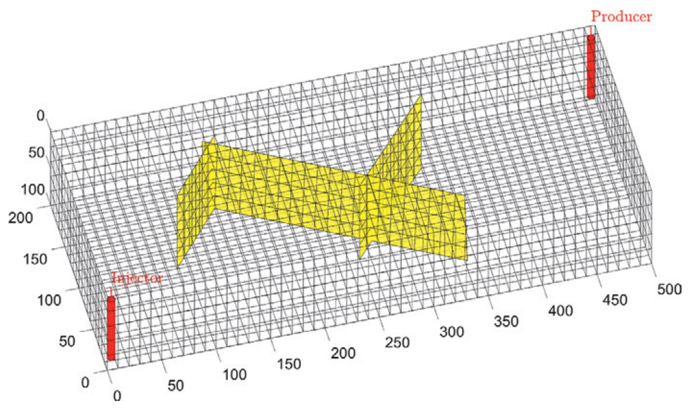

Figure 10.4 The three yellow planes represent the fractures in the 3D structured mesh of the simulation domain. The injection and production wells are shown as the vertical red lines at two corners of the simulation domain.

(like EOR/IOR) in UOG reservoirs, the complete code for this simplistic case is included as `compositional3D.m` in the `examples` folder. For instance, an interested reader could replace the water in the injection well with a lean gas or replace the simplistic fractures with the stochastic fractures discussed in the next section.

The simple case discussed here starts by loading the required modules:

```
mrstModule add compositional hfm;
```

Select fluid composition and compute surface densities: The first line of Listing 10.1 shows how to select the multicomponent fluid for compositional simulation. The `compositional` module provides a `getBenchmarkMixture` function that facilitates the specification of the composition, chemical properties, and initial conditions of the multicomponent hydrocarbon mixture to be simulated. The function is generic enough to allow the simulation of single hydrocarbon components,

Listing 10.1 *Create single cell model with selected composition.*

```
casename = 'verysimple'; % Simple 2-component fluid
%% Create a single-cell model with a given initial composition
[fluid, info] = getBenchmarkMixture(casename);
eosname = 'pr';                % Select the Peng-Robinson equation of state
G1cell = cartGrid([1 1],[1 1]); % Create a single-cell grid
G1cell = computeGeometry(G1cell);
EOSModel = EquationOfStateModel(G1cell, fluid, eosname);
```

as well as fluid mixtures with tens of hydrocarbon components. To speed up the simulation runs in this section, we simulate the flow of a simple fluid that is made up of 10% CO_2 and 90% n-decane. The fluid is specified via a `casename` called `'verysimple'`, which is defined in the `getBenchmarkMixture` function. We use the Peng–Robinson equation of state [29] to compute the fluid properties such as phase compressibility factors (Z) and densities.

To convert the reservoir fluid volumes to volumes at surface conditions, we flash the selected fluid composition to surface conditions in a single-cell model and obtain the surface densities of the oil and gas phases (as shown in Listing 10.2). The reader is reminded that formation volume factor can be expressed as the ratio of the surface densities to the corresponding densities at reservoir conditions.

Listing 10.2 *Flash fluid in single-cell model to surface conditions.*

```
%Surface Conditions
p_sc = 101325;  % atmospheric pressure
T_sc = 288.706; % 60 Farenheit
[~, ~, ~, ~, ~, rhoO_S, rhoG_S] = ...
          standaloneFlash(p_sc, T_sc, info.initial, EOSModel);
flowfluid = initSimpleADIFluid('phases', 'WOG', 'n', [2, 2, 2], ...
                               'rho', [1000, rhoO_S, rhoG_S]);
```

Select compositional formulation to be used: Listing 10.3 shows how to use the natural variables approach for compositional simulation. To prevent the use of the standard two-point approach to calculate the transmissibilities, we do not provide a second argument to `setupShaleEDFMOpsTPFA`. Note that we use our own function instead of `setupEDFMOperatorsTPFA` from the `hfm` module because our modification makes the function compatible with the 2019b version of MRST, as discussed in Section 10.2.

Listing 10.3 *Use the natural variables formulation.*

```
model = NaturalVariablesCompositionalModel(G, [], flowfluid, ...
          fluid, 'water', true);
% Setup TPFA operators that incorporate all the EDFM NNCs
model.operators = setupShaleEDFMOpsTPFA(G, G.rock, tol);
```

Specify the initial composition of the injection/production wells: We provide the initial composition of the injected and produced fluids after the definition of the injector and producer. The actual composition of the fluid produced will be determined based on the thermodynamics and flow processes implemented in the reservoir simulator. The last line of the code listing shows how to initialize a compositional reservoir model using the `initCompositionalState` function:

```
W(1).components = info.injection; % obtained from getBencharkMixture
W(2).components = info.initial;
%% Set up initial state and schedule
s0 = [0, 1, 0];
state = initCompositionalState(G, pRef, info.temp, s0, info.initial, ...
          model.EOSModel);
```

Simulation results: Figure 10.5 presents the pressure and ternary (water, gas, and oil) saturation profiles after simulating production, and water injection for three and a half years. These profiles show a marked change in the pressure and saturations near the surfaces of the fractures modeled. This is because the fractures act as the paths of least resistance to flow, and they accelerate the rate of flow of the injected water toward the production well. Although we modeled only three fractures in this section, real fractured reservoirs can have hundreds or thousands of natural fractures with any orientation in a 3D space. The next section discusses how to generate such realistic fractures using a stochastic approach.

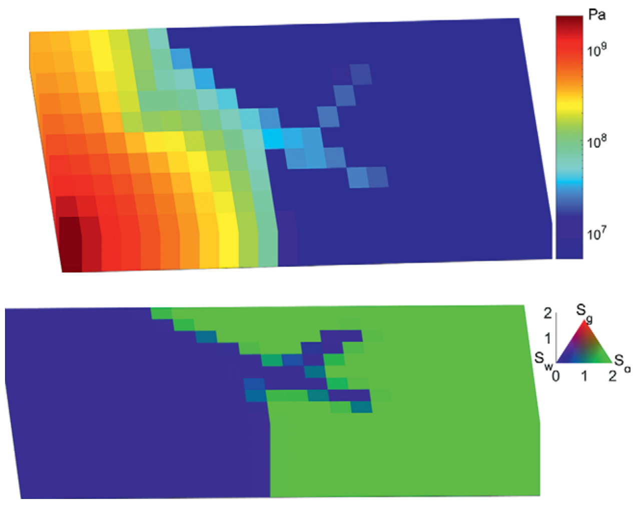

Figure 10.5 The pressure (top) and overall saturation (bottom) profiles after three and a half years of production show the effect of the fractures on the flow of fluids.

10.5 Stochastic Generation of Fractures with Arbitrary Orientations in 3D

Considering that there is no technology available to determine the exact location of all natural fractures in the subsurface, we could evaluate the effect of the uncertainties in the fracture location and orientation by generating several realizations of the fracture network. This section shows how to generate such realistic stochastic fracture networks in 3D using an open-source MATLAB code called ADFNE. Our pEDFM example codes assume that ADFNE has been added to the MATLAB path. To minimize MATLAB crashes on the Windows platform, the user is also advised to delete the `R2015a` folder within the ADFNE folder. Alghalandis [1] provides a comprehensive tutorial on the stochastic generation of fractures using ADFNE. The `Demo.m` file in the ADFNE folder illustrates virtually all of the functionality available in this package. Our goal in this section is to show the modifications required to use the stochastic fractures from ADFNE in the `hfm` and `shale` modules.

10.5.1 Generation of Fracture Sets Using ADFNE

We start by demonstrating how to generate two sets of natural fractures with 350 fractures each. The complete code is available as `stochasticFracs.m` in the `examples` folder. The first fracture set contains fractures with a dip angle and a dip direction of 45°, and the second contains fractures with a dip angle of 45° and a dip direction of 315° (or −45°). As illustrated in Figure 10.6, the dip angle is the acute angle that a rock surface makes with a horizontal plane, whereas the dip direction is the azimuth of the direction of the dip, as projected to the horizontal.

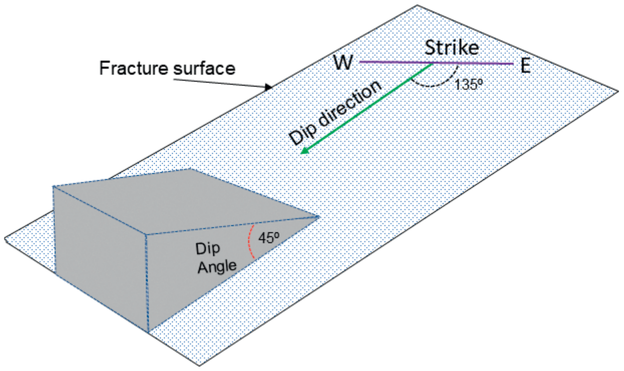

Figure 10.6 Illustration of the dip (angle) and dip direction of a fracture or fault surface.

In the code excerpt, the DFN function generates random DFNs when no arguments are provided and customizes the fracture networks when the function arguments are given. The Field function takes two arguments: a DFN and a field that is set to either line (for 2D models) or poly (for 3D models). In the DFN function arguments, dim is the number of dimensions modeled, n is the number of fractures, dip is the dip angle, and dir is the dip direction. The minimum, mean, and maximum lengths (used in the exponential distribution for the lengths) of each of these fractures in any direction are set to 4, 10, and 30 m, respectively ('minl',4,'mu',10,'maxl',30). All fractures are constrained to lie within the reservoir domain, which is specified with a bounding box (bbx). Considering that the pEDFM algorithm requires each matrix cell that hosts a fracture (called “host matrix cell”) to have six neighbors, we prevent the fractures from touching the reservoir boundary by starting the minimum spatial coordinate from a very small value (tol=0.01) instead of the origin. This tolerance value is also subtracted from the maximum values of the spatial coordinate of the box:

```
tol = 0.01;
physdim = [3300 1300 250]*ft;
set1 = Field(DFN('dim',3,'n',350,'dir',45,'ddir',-100,'minl',4,'mu',10, ...
          'maxl',30,'bbx',[tol,tol,tol,physdim(1)-tol, ...
          physdim(2)-tol,physdim(3)-tol],'dip',45,'ddip',-100, ...
          'shape','l','q',4),'Poly');
set2 = Field(DFN('dim',3,'n',350,'dir',-45,'ddir',-100,'minl',4,'mu',10, ...
          'maxl',30,'bbx',[tol,tol,tol,physdim(1)-tol, ...
          physdim(2)-tol,physdim(3)-tol],'dip',45,'ddip',-100, ...
          'shape','l','q',4),'Poly');
[set1_,nonPlanarSets1,fracArea1] = processStochFracs(set1);
```

The processStochFracs function used in the last line of the code excerpt provides a robust test for coplanarity and removes unrealistic fractures that either have negligible areas or appear as lines (instead of planes). This is needed because the EDFM and pEDFM codes presented herein require the vertices of each fracture to be coplanar. To allow the orientation of fractures to vary, we provide ddip and ddir as function arguments that control the degree of the variation in the dip and dip directions, respectively. The degree of variation in the fracture orientation increases as the magnitudes of these two negative numbers increase. We set both variables to –100 and use plotfracSystemNF to plot the resulting fracture network and wellbore trajectories in Figure 10.7. The plotfracSystemNF function extends the plotting functionality in MRST to allow us plot the wellbore trajectories and hydraulic and natural fractures with custom colors as shown in Figures 10.7 and 10.10. The yellow-colored fractures in

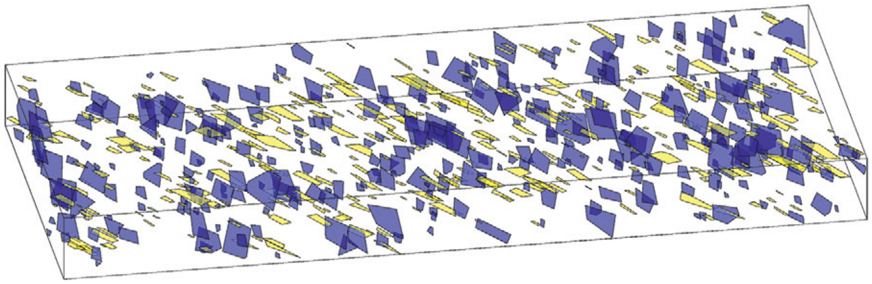

Figure 10.7 This figure illustrates the two fracture sets created. The 350 fractures in the first fracture set are colored yellow, and the remaining 350 fractures in the second fracture set are colored blue.

Figure 10.7 belong to the first fracture set, whereas the blue-colored fractures belong to the second. We compute the dip and dip direction of the two fracture sets using the `Orientation` function and plot these using the `Stereonet` function from `ADFNE`:

```
plotfracSystemNF(G, fracplanes, numHFplanes, wells, 'label', false)
o = Orientation([set1;set2]); % extract and store dip angle and direction
figure,
Stereonet([o.Dip],[o.Dir],'density',true,'marker','*','ndip',6,...
          'ndir',24,'cmap',@jet,'color','y');
```

This yields the stereonet in Figure 10.8, which is similar to a scatterplot in the sense that it plots the dip and dip direction of each fracture as a point (with a yellow asterisk). The red and blue colors in this stereonet indicate the relative number of fractures in each segment of the stereonet. The red (or hotter) colors indicate the segments with the largest number of fractures, whereas the blue (or cooler) colors indicate the segments with fewer fractures. As seen in the figure, the white segments are the regions of the stereonet with no fractures. In this stereonet, a positive dip direction is read starting from the 0° point and moving in a counter-clockwise direction, whereas the dip angle starts out from the circumference of the outermost circle and increases to a maximum of 90° in the center of the circle. The stereonet allows us to summarize the orientation of all of the fractures in the system in one simple plot. The reader is encouraged to modify the dip, dip direction, and their degrees of variation and observe the change in the orientation of the fractures in both Figures 10.7 and 10.8. The next section shows how we model an Eagle Ford shale oil reservoir using natural fracture networks generated as described in this section.

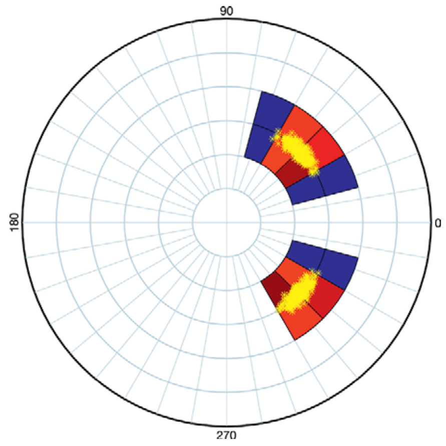

Figure 10.8 This stereonet shows the limited variation in the orientation of the fractures in the two fracture sets modeled. Each yellow asterisk represents a fracture, whereas the red and blue segments indicate the regions with the most and fewest number of fractures, respectively.

10.6 Applications of 3D pEDFM to Model UOG Reservoirs

This section discusses how to model unconventional reservoirs (with hundreds of natural fractures in any orientation) using the 3D pEDFM algorithm recently presented by Olorode et al. [27]. Although we only model a realization from a stochastic fracture network, our goal is to show how the `shale` module can be used to predict the production corresponding to any realization from a stochastic fracture model. This could then be used to study and evaluate the sensitivity of model forecasts to the uncertainty in the amount, distribution, geometry, and orientation of these natural fractures.

10.6.1 Basic Model Parameters Representative of the Eagle Ford Shale

Most of the model parameters presented in this section were taken from Yu et al. [37]. Table 10.3 summarizes the well, matrix, and fracture properties, and Tables 10.4 and 10.5 present the compositional data and binary interaction constants, respectively. These parameters are used in the Peng–Robinson equation of state [29], which is implemented in the `compositional` module. To obtain a high-resolution discretization of the simulation domain, we take advantage of the symmetry observed in bi-wing planar hydraulic fractures. The idea is that we can represent the multiply fractured horizontal well system with a smaller repeatable fraction of the entire domain, called a "stencil." Figure 10.9 illustrates the stencil

Table 10.3 *Representative Eagle Ford shale parameters.*

Parameters	SI units	Field units
Stencil model dimension	150 × 300 × 38.1 m	492.1 × 984.3 × 125.0 ft
Initial reservoir pressure, p_i	$5.6 \cdot 10^7$ Pa	8 125 psia
Reservoir temperature, T	405.4 K	270°F
Reservoir thickness, h	76.2 m	250.0 ft
Matrix permeability, K_m	$8.88 \cdot 10^{-19}$ m^2	$9 \cdot 10^{-7}$ D
Matrix porosity, ϕ	0.12	0.12
Hydraulic fracture porosity, ϕ_{frac}	0.5	0.5
Hydraulic fracture half-length, x_f	180 m	590.6 ft
Hydraulic fracture aperture, w_f	0.003 m	0.01 ft
Hydraulic fracture height, h_f	45.7 m	150 ft
Hydraulic fracture permeability	$9.87 \cdot 10^{-13}$ m^2	1.0 D
Natural fracture aperture	$1 \cdot 10^{-7}$ m	$3.28 \cdot 10^{-7}$ ft
Conductive natural fracture permeability	$9.87 \cdot 10^{-17}$ m^2	$1 \cdot 10^{-4}$ D
Sealing natural fracture permeability	$9.87 \cdot 10^{-22}$ m^2	$1 \cdot 10^{-9}$ D
Number of fracture stages	7	7
Number of fractures per stage	3	3
Cluster spacing	10 m	32.8 ft
Fracture spacing	150 m	492.13 ft
Well radius, r_w	0.1 m	0.33 ft
Initial water saturation, S_w	0.17	0.17
Flowing bottom-hole pressure, p_{wf}	$6.895 \cdot 10^6$ Pa	1 000 psia
Number of natural fractures	150	150
Well length	920 m	3 018.4 ft

Table 10.4 *Compositional data for Eagle Ford shale oil.*

Components	Mole fraction	Critical pressure (atm)	Critical temperature (K)	Critical volume (L/mol)	Molar weight (g/g mol)	Acentric factor	Parachor
CO_2	0.01183	72.80	304.20	0.0940	44.01	0.2250	78.0
N_2	0.00161	33.50	126.20	0.0895	28.01	0.0400	41.0
C_1	0.11541	45.40	190.60	0.0990	16.04	0.0080	77.00
C_2-C_5	0.26438	36.50	274.74	0.2293	52.02	0.1723	171.07
C_6-C_{10}	0.38089	25.08	438.68	0.3943	103.01	0.2839	297.42
C_{11+}	0.22588	17.55	740.29	0.8870	267.15	0.6716	661.45

Table 10.5 *Binary interaction constants for Eagle Ford shale oil.*

	CO_2	N_2	C_1	C_2–C_5	C_6–C_{10}	C_{11+}
CO_2	0	0.0200	0.1030	0.1299	0.1500	0.1500
N_2	0.0200	0	0.0310	0.0820	0.1200	0.1200
C_1	0.1030	0.0031	0	0.0174	0.0462	0.1110
C_2-C_5	0.1299	0.0820	0.0174	0	0.0073	0.0444
C_6-C_{10}	0.1500	0.1200	0.0462	0.0073	0	0.0162
C_{11+}	0.1500	0.1200	0.1110	0.0444	0.0162	0

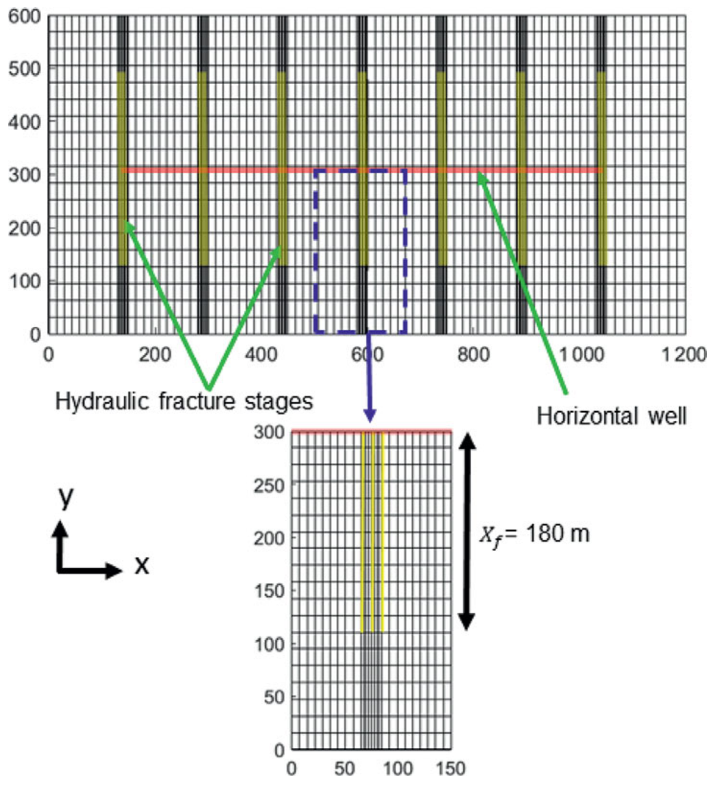

Figure 10.9 Plan view of grid shows a multiply fractured horizontal well with seven fracture stages (top) and the “minimum repetitive element” or “stencil” (bottom).

in relation to a multiply fractured horizontal well system. The red horizontal line in the middle of the figure shows the well location. The stencil in the bottom figure is a model of half of the simulation domain around one fracture stage and half of this domain in the z-direction. Thus, it is essentially a quarter of the domain around one fracture stage. The production rates obtained for the stencil can then be multiplied by the number of times the stencil is repeated in the full simulation domain. This means that for a system with seven fracture stages, we will multiply the simulation results for the stencil by 28. Olorode [24] showed that the use of the stencil is able to replicate the behavior of the full system of multiple fractures for the expected production period of a typical shale oil/gas well.

10.6.2 Implementation Steps

This section discusses the steps required to perform a compositional simulation of fractured UOG reservoirs using MRST. We use an Eagle Ford shale oil reservoir as a case study and provide the complete code (`eagleFord.m`) in the `examples` folder of the `shale` module. We also provide an `eagleFordEDFM.m` script, which only differs from `eagleFord.m` in that it uses EDFM instead of pEDFM. The next three subsections show and explain excerpts from `eagleFord.m`.

Create the hydraulic fractures: We provide a `createMultiStageHFs` function to facilitate the modeling of multiple stages of hydraulic fractures with one or more fractures per stage. Using this function, we can model either the stencil discussed in the previous subsection or a multiply fractured horizontal well system with any number of hydraulic fractures. Here, we show how we model a stencil with the three fractures that make up one of the stages of the multiply fractured horizontal well. All other examples in the `shale` module involve modeling the complete system with multiple fracture stages.

We use `clusterSpacing` to specify that the three fractures in a fracture stage are 10 m apart, whereas `fracSpacing` is used to specify that the fracture stages are 150 m apart in a multistage fracture system. In this stencil case, the fracture spacing is inferred from the size of the domain in the x-direction (which is 150 m, as in Table 10.3). We also show how to specify the input values for the aperture, porosity, and permeability of each hydraulic fracture:

```
[fracplanes,frac_centroid_s] = createMultiStageHFs('numStages',NumStages,...
            'fracSpacing',fractureSpacing,'numFracsPerStage',fracPerStage,...
            'fracHalfLength',fracHalfLength,'fracHeight',fracHeight,...
            'clusterSpacing', clusterSpacing,'fracCentroid1',fracCentroid1,...
            'isStencil',1);
for i=1:numel(fracplanes)
     fracplanes(i).aperture = fracAperture;
     fracplanes(i).poro = fracPoro;
     fracplanes(i).perm = fracPerm;
end
```

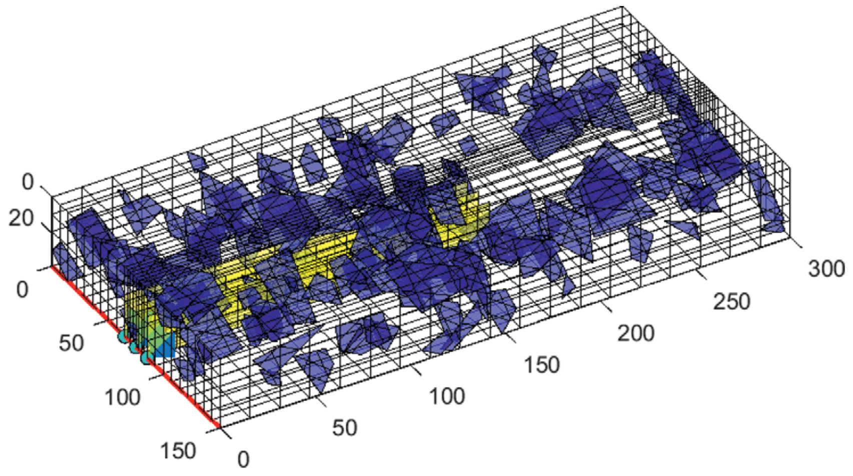

Figure 10.10 This figure shows the hydraulic (in yellow) and natural (in blue) fractures in the stencil used in our simulation of the Eagle Ford test case. This stencil is essentially a quarter of the simulation domain around one fracture stage.

Create the natural fractures: Next, we use ADFNE (as shown in Subsection 10.5.1) to generate the natural fractures. To demonstrate the advantage of pEDFM over EDFM, we randomly set half of the natural fractures as highly conductive and the other half as sealing (or flow barriers). The reader is reminded that the main advantage of pEDFM over EDFM is that pEDFM models both low- and high-conductivity fractures accurately, whereas EDFM is unable to model low-conductivity fractures accurately. The following code excerpt shows how to specify the properties of the natural fractures, such as aperture, porosity, and permeability:

```
for i=1:numNFplanes
    idxGlobal = numHFplanes + i;
    fracplanes(idxGlobal).points = fracSet{i}(1:end-1,:);
    fracplanes(idxGlobal).aperture = 100*nano*meter;
    fracplanes(idxGlobal).poro=0.5; % high because HF is not propped everywhere
    if(mod(i,2)==false) %even NF index set as conductive fractures
       fracplanes(idxGlobal).perm=100*micro*darcy;
    else %odd NF index set as sealing fractures
       fracplanes(idxGlobal).perm=1*nano*darcy;
    end
end
```

Figure 10.10 shows the hydraulic and natural fractures obtained. The three hydraulic fracture planes that make up a hydraulic fracture stage are shown in yellow, whereas the natural fractures are shown in blue.

Listing 10.4 *Required NNCs for pEDFM.*

```
 [G,fracplanes]=EDFMgrid(G,fracplanes,...
       'Tolerance',tol,'plotgrid',false,'fracturelist',1:numel(fracplanes));
G = fracturematrixNNC3D(G,tol);                              % Frac-Matrix NNCs
 [G,fracplanes]=fracturefractureNNCs3D(G,fracplanes,tol); % Frac-Frac NNCs
 [G,wells] = wellfractureNNCs3D(G,fracplanes,wells,tol);  % Well-Fracs NNCs
G = pMatFracNNCs3D(G,tol);                                   % pEDFM NNCs
TPFAoperators = setupPEDFMOpsTPFA(G, G.rock, tol); % Setup pEDFM operators
```

Compute all pEDFM NNCs: The last three function calls in Listing 10.4 show how we model the well/fracture, projection matrix/matrix, and projection matrix/fracture NNCs in the `shale` module. The other parts of this code listing are identical to those used in embedded fracture modeling with the `hfm` module.

To model fractures that are connected to a production well, we use the `well fractureNNCs3D` function to find all of the fracture cells that intersect the well. We then compute the effective wellbore index (WI) for each of these fracture–well intersections using a form of the Peaceman model presented in Moinfar [21]:

$$WI_f = \frac{2\pi K_f w_f}{\ln\left(r_e/r_w\right)}, \tag{10.10}$$

where

$$r_e = 0.14\sqrt{l_f^2 + h_f^2}. \tag{10.11}$$

Here, K_f is fracture permeability, w_f is fracture aperture, r_e is effective or representative wellbore radius, and r_w is the actual wellbore radius given in Table 10.3. Additionally, l_f and h_f represent the length and height of a fracture segment that is bounded within the fracture cell. The `pMatFracNNCs3D` function implements our pEDFM algorithm to find the projection cells, as well as the non-neighboring transmissibilities between the projection and host matrix cells. Similarly, the `setupPEDFMOpsTPFA` function computes the non-neighboring transmissibilities between the projection cells and fracture cells, in addition to the other standard and EDFM transmissibilities, as discussed in Subsection 10.3.3.

10.6.3 Eagle Ford Shale Reservoir Simulation Results

We ran the Eagle Ford shale oil simulation case for 15 years and obtained the results presented in Figures 10.11–10.16. From the pressure profile given in Figure 10.11, we observe a sharp drop in the pressure near the hydraulic fracture surfaces. An inspection of Darcy's law in the context of a fractured ultra-low matrix permeability reservoir confirms that the pressure gradient in the matrix has to be large to sustain

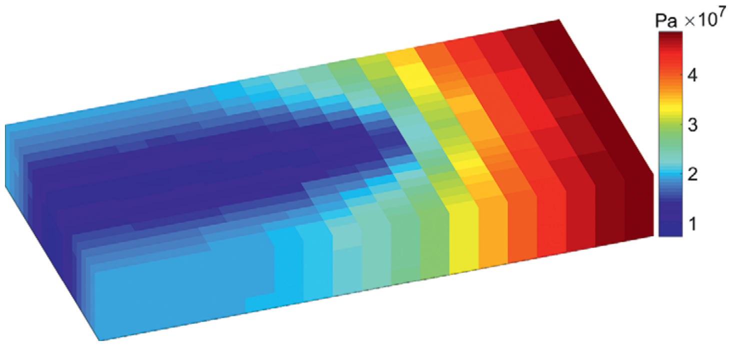

Figure 10.11 Pressure profile after 15 years of production shows the typical sharp drop in pressure near the fracture surfaces. The pressure right by the fracture surface is approximately equal to the flowing bottom-hole pressure of 1 000 psia.

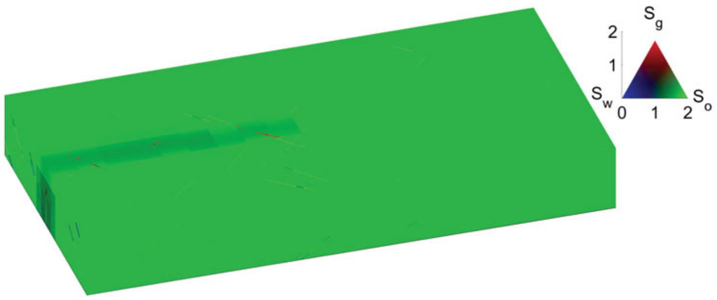

Figure 10.12 The profiles of water, oil, and gas saturation after 15 years of production show that oil vaporizes in the vicinity of the fracture surface due to the sharp pressure drop in this region.

the much higher flow rates expected in the fractures. In highly conductive fractures, the fluid pressure is also expected to drop rapidly toward the value of the flowing bottom-hole pressure. It is worth noting that the results shown in this section are for the stencil and have not been multiplied by the number of times the stencil is repeated in the actual multistage fracture system of interest.

Figure 10.12 presents a ternary plot of the three-phase saturation profile obtained after 15 years of simulated production. An inspection of the three saturations reveals that the gas saturation near the fracture surface is higher than elsewhere in the reservoir. This is because at initial conditions, the reservoir fluid exists only in two phases (oil and water). However, as the pressure near the fracture surfaces drops below the bubble-point pressure, gas comes out of the solution. This results in the localized increase in gas saturation and corresponding decrease in oil saturation in Figure 10.12, whereas the change in water saturation is negligible. Although we

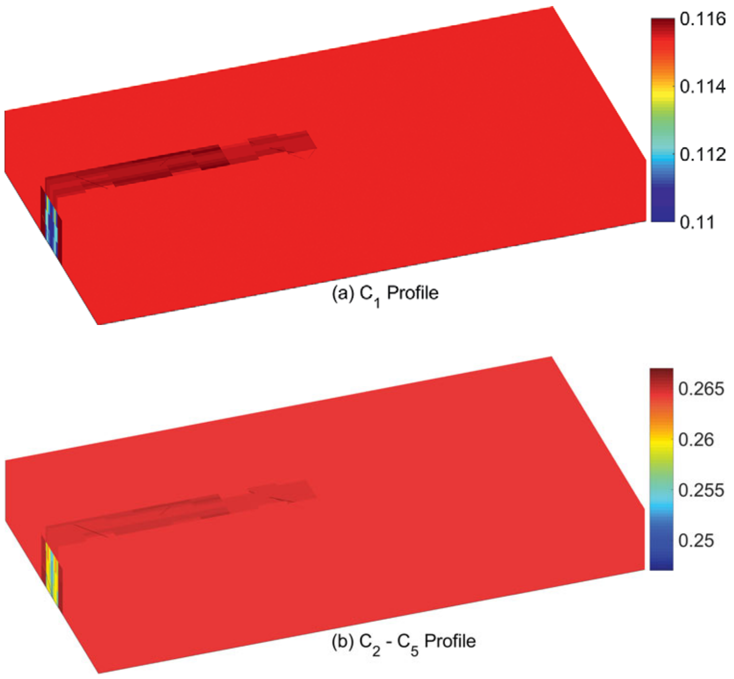

Figure 10.13 The profiles of the mole fractions of C_1 and C_2–C_5 show the change in composition due to the vaporization of the oil near the fracture surfaces. The profiles of the other four fluid components are not presented for brevity.

modeled a six-component hydrocarbon mixture, we show only the mole fractions of the first two hydrocarbon components in Figure 10.13. The marked change in composition near the fracture surface is because the mole fractions of the hydrocarbon components in the oil and gas phases deviate from the original single-phase oil composition when the pressure drops below the bubble point near these fractures.

Figure 10.14 compares the oil and gas rates from the pEDFM implemented in the `shale` module to the corresponding rates from the EDFM implemented in the `hfm` module. The oil-rate plot shows the typical half-slope that is indicative of linear flow after about 100 days of production. The gas-rate plot shows that gas production begins after approximately 15 days of production and rises sharply to a peak gas rate of about 5 Mscf/day. The gas rate then declines because the total reservoir fluid withdrawal declines, as shown in Figure 10.16. As expected, the use of a log–log plot in Figure 10.14 (left) masks the overestimation of oil production when EDFM is used. Therefore, we provide the corresponding cumulative production plots in

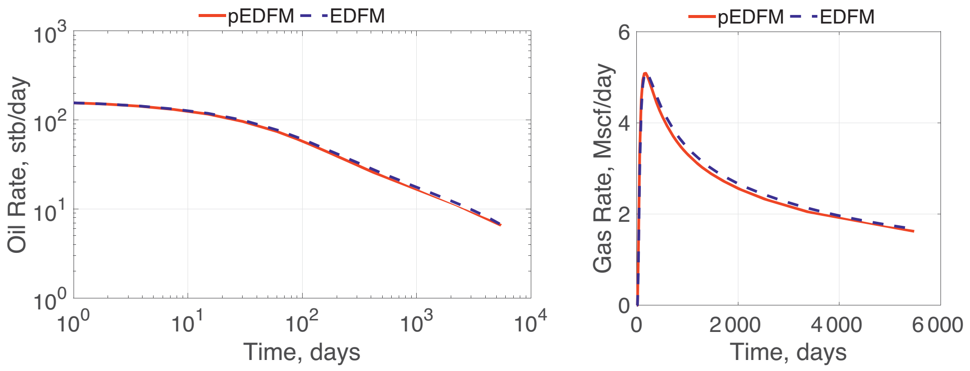

Figure 10.14 The simulation results show that both oil and gas production rates from EDFM are higher than those from pEDFM. Note that the oil rates are presented on a log–log plot, whereas the gas rates are shown on a Cartesian plot.

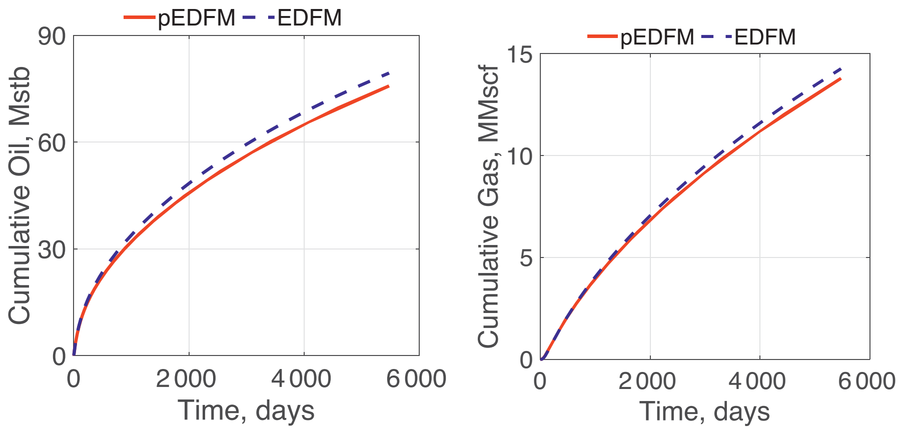

Figure 10.15 The results show that EDFM overestimates the oil and gas expected ultimate recovery by 4.78% and 3.44%, respectively (in comparison to pEDFM).

Figure 10.15. It shows that EDFM overestimates the expected ultimate recovery of oil and gas by 4.78% and 3.44%, respectively (when compared to the results from pEDFM). Figure 10.16 shows that the total reservoir fluid withdrawal from EDFM is 4.32% more than that from pEDFM. This is expected because half of the natural fractures are essentially flow barriers and EDFM is unable to account for sealing fractures accurately. Additionally, the modeling of the hydraulic fractures with pEDFM is more accurate than with EDFM when the fracture lies on the interface between two matrix cells. In this case, pEDFM simplifies to DFM whereas EDFM does not [32], and this adds to its overestimation of production.

Table 10.6 *Some transport and storage mechanisms expected in shale reservoirs.*

Mechanism	Models	Continuum	Type
Adsorption	Langmuir	Matrix	Storage
Diffusion	Fick's law	Matrix and fracture	Storage
Geomechanics	Gangi	Fracture	Transport

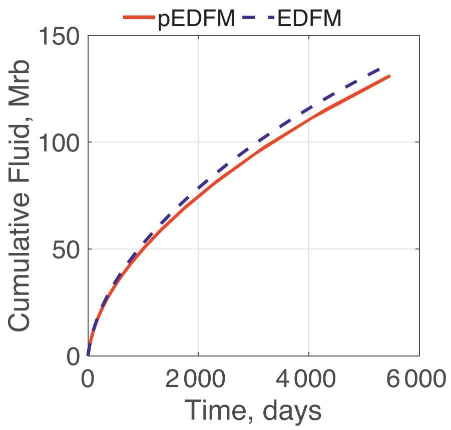

Figure 10.16 The results show that EDFM overestimates the total reservoir fluid withdrawal volume by 4.32% (in comparison to pEDFM).

10.7 Modeling Transport and Storage Mechanisms in Organic-Rich Source Rocks

The study of the physical mechanisms in unconventional oil and gas reservoirs has been an active research area over the last decade. Most of these studies have focused on the understanding of the storage, flow, and transport mechanisms in the shale plays. Consequently, several models have been developed to describe the complex nature of adsorption, diffusion, and fracture closure in UOG reservoirs. This section shows how to implement some of these physical mechanisms in MRST and provides tutorial cases that illustrate the application of these mechanisms in shale-gas reservoir simulation. Our goal with these examples is to show how to easily incorporate additional physical mechanisms as needed. Table 10.6 summarizes the physical mechanisms and models discussed in this section. The remaining subsections show how the governing equations are modified and implemented to account for these physical mechanisms.

To illustrate the significance of all of the shale mechanisms discussed in this section, we model a Barnett shale-gas reservoir with the parameters summarized in Table 10.7. These parameters were taken from Ambrose [2], Olorode et al. [26],

Table 10.7 *Representative Barnett shale-gas model parameters.*

Parameters	SI units	Field units
Fracture half-length, x_f	91.4 m	300 ft
Fracture width, w_f	$3 \cdot 10^{-3}$ m	$9.84 \cdot 10^{-3}$ ft
Reservoir thickness, h	100.6 m	330 ft
Matrix permeability, K_m	$1.0 \cdot 10^{-19}$ m^2	$1.0 \cdot 10^{-4}$ md
Fracture permeability, K_f	$5.0 \cdot 10^{-11}$ m^2	$5.0 \cdot 10^{4}$ md
Matrix porosity, ϕ	0.04	0.04
Fracture porosity, ϕ_{frac}	0.33	0.33
Temperature, T	366.5 K	200°F
Well radius, r_w	0.1 m	0.32 ft
Initial reservoir pressure, p_i	$3.45 \cdot 10^{7}$ Pa	5 000 psia
Initial mole fractions, z_i	[0.85, 0.1, 0.05]	[0.85, 0.1, 0.05]
Flowing bottom-hole pressure, p_{wf}	$1.03 \cdot 10^{7}$ Pa	1 500 psia
Tortuosity, τ	2–10	2–10
Biot's constant, α	0.5	0.5
Confining pressure, P_c	$1.03 \cdot 10^{8}$ Pa	15 000 psia
Effective stress, P_1	$1.8 \cdot 10^{8}$ Pa	26 000 psia
ρ_{sL} of C_1, C_2, and C_3	[3.0, 4.9, 9.6] kg/m^3	[56, 91, 179] scf/ton
Langmuir pressure, p_L for C_1, C_2, C_3	$[10.8, 5.6, 5.8] \cdot 10^{6}$ Pa	[1 562, 811, 844] psia
Bulk density, ρ_b	2 500 kg/m^3	$1.56 \cdot 10^{5}$ lb/ft$_3$

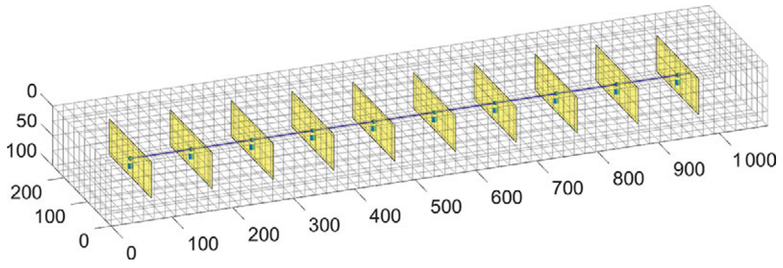

Figure 10.17 Simple Barnett shale-gas simulation domain with 10 fractures. This grid is used in all of the remaining simulation results presented in this chapter.

and Xiong et al. [36]. Figure 10.17 shows the multiply-fractured horizontal well and dimensions of the reservoir modeled in this section. The reservoir domain is meshed with a structured Cartesian grid, and we simulate 10 planar and orthogonal hydraulic fractures connected to a horizontal well.

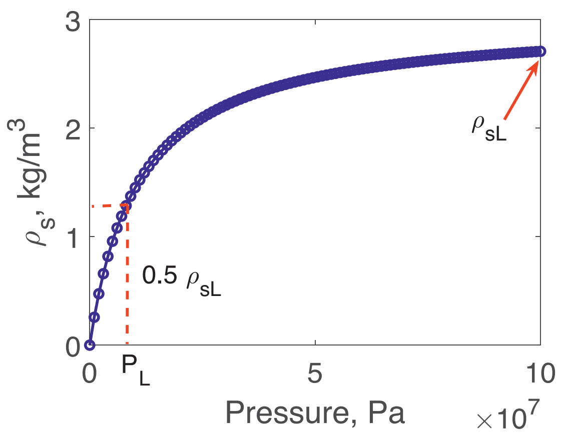

Figure 10.18 The Langmuir isotherm illustrates that the change in the amount of gas adsorbed is much less at high pressures.

10.7.1 Sorption

Shale-gas reservoirs have been observed to have a wide pore size distribution that ranges from as low as 2 nm to hundreds of nanometers or more. As pore sizes get smaller than approximately 10 nm, gas molecules begin to interact a lot more with the pore walls, leading to a deviation from the classical bulk behavior of fluids. In conventional gas reservoirs, gas is stored in the compressed (or free) state, but in unconventional gas reservoirs, gas could also be stored in the sorbed (adsorbed plus dissolved) state, in addition to free gas storage. Next, we present the Langmuir isotherm and show how it is modeled in the `shale` module.

Langmuir Isotherm

For a single-component gas, the Langmuir isotherm can be written as

$$\rho_s = \rho_{sL}\frac{p}{p + p_L}. \tag{10.12}$$

Here, ρ_s refers to the mass density of gas sorbed, and ρ_{sL} refers to the maximum mass density of gas that can be sorbed in the reservoir rock. The SI unit for both variables is kilograms per cubic meter. The variables p and p_L refer to the reservoir and Langmuir pressures, respectively. As shown in Figure 10.18, the Langmuir pressure is the pressure at which ρ_s is equal to half of ρ_{sL}.

For multicomponent gas mixtures, it is common to use the extended Langmuir isotherm, which is given as

$$\rho_s^i = \rho_{sL}^i \frac{y_i \frac{p}{p_L^i}}{1 + \sum_{j=1}^n y_j \frac{p}{p_L^j}}, \tag{10.13}$$

where the superscripts and subscripts i and j refer to each hydrocarbon component. Thus, y_i refers to the mole fraction of component i in the gas phase. The other parameters remain as defined in the single-component case. To obtain the total sorbed-gas density (ρ_s) at any pressure, we take the sum of the sorbed-gas density of each component (ρ_s^i) in the multicomponent fluid. To account for the sorption of each hydrocarbon gas component i on the nanoporous shale pore walls, we modify the storage (first) term in the governing equation (10.1) as in Moridis et al. [22]:

$$\partial_t \left[\phi \sum_{\alpha=l,v} \rho_\alpha S_\alpha X_\alpha^i + \delta_s (1-\phi) \rho_s^i \right] + \sum_{\alpha=l,v} \nabla \cdot \left(\rho_\alpha X_\alpha^i \vec{v}_\alpha \right) - \sum_{\alpha=l,v} \rho_\alpha X_\alpha^i q_\alpha / V = 0, \tag{10.14}$$

where ρ_s is sorbed-gas density in kilograms per cubic meter of shale. The multiplication of the sorbed-gas density of component i by $1-\phi$ accounts for the fact that the sorbed-gas amount is considered per mass unit of rock. Therefore, each term in this equation is in units of component mass per unit bulk volume and per unit time (kg m^{-3} s^{-1}). As in Moridis et al. [22], δ_s is a logical parameter that is set to one in the shale matrix and zero elsewhere. This allows us to model sorption only in the shale matrix.

Implementation of the Extended Langmuir Isotherm

We provide a `sorption.m` script (in the `examples` folder) to illustrate our implementation of sorption in MRST. We use `makeShaleRock` to add an `isMatrix` field to the `G.rock` data structure. It is set to one for every matrix cell and zero otherwise. This allows us to turn on certain physical mechanisms (such as sorption) only in the matrix if needed. We then activate sorption by setting the `sorption` subfield of the `shaleMechanisms` field to one, as shown:

```
G_matrix.rock = makeShaleRock(G_matrix, matrix_perm, matrix_poro);
G.rock.shaleMechanisms.sorption = 1;  % set to 0 to turn off sorption
```

In `setupShaleEDFMOpsTPFA` and `setupShalePEDFMOpsTPFA`, we compute and store the grain volume as shown:

```
if isfield(G.rock,'shaleMechanisms') && isfield(G.rock.shaleMechanisms,'sorption')
   s.isSorbed = rock.isMatrix;
   s.gv = s.isSorbed.*(1-rock.poro) .* G.cells.volumes;
end
```

Given that the parameters of the extended Langmuir isotherm are specified for each pure component in the fluid mixture, we provide these sorption parameters and other standard compositional fluid parameters in `getShaleCompFluidCase`. We also provide a `SorbedCompositionalFluid` class that extends the `TableCompositionalFluid` class to include these sorption parameters.

To demonstrate our implementation of sorption, we provide a representative Barnett shale-gas fluid case named `barnett3comps`. In `getShaleCompFluidCase`, we store the multicomponent ρ_{sL} and p_L values from Ambrose [2] in the first and second columns of the `isotherm` field as shown:

```
case 'barnett_3comps'
    names = {'Methane', 'Ethane', 'n-Propane'};
    % Pressure in Pa in first column, rho_sL in kg/m^3 in 2nd column
    isotherm = [10769611, 2.99;  5591648, 4.86; 5819175, 9.56];
    fluid = SorbedCompositionalFluid(names,isotherm');
```

Our `eqnsShaleNaturalVars` function shows how we compute the sum in the denominator of (10.13) at the current and previous timesteps:

```
if isfield(model.G.rock, 'shaleMechanisms') && ...
   isfield(model.G.rock.shaleMechanisms,'sorption')
    sumIso = y{1}./model.EOSModel.fluid.isotherm(1,1);
    sumIso0 = y0*(1./model.EOSModel.fluid.isotherm(1,:)');
    for ii=2:numel(y)
        sumIso = sumIso + y{ii}./model.EOSModel.fluid.isotherm(1,ii);
    end
end
```

We then modify the governing equations in `eqnsShaleNaturalVars` to include sorption when necessary, using the `if` block in Listing 10.5. The reader is encouraged to compare this code to the modified governing equations (10.12) and (10.14). The next subsection shows our simulation results with and without sorption.

Simulation Results With and Without Sorption

Figure 10.19 reports a comparison of the gas production obtained with and without the effect of sorption. These results show that sorption contributes an additional 7% of gas production over 15 years. The contribution of sorption is usually limited because the shape of the Langmuir isotherm (as in Figure 10.18) is such that it flattens out at high pressures. The average reservoir pressures need to be very low to see a more significant contribution from sorption.

Listing 10.5 *Option to choose to model sorption or not.*

```
if isfield(model.G.rock, 'shaleMechanisms') && ...
   isfield(model.G.rock.shaleMechanisms,'sorption')
    eqs{i} = (1/dt).*( ...
        rhoO.*pv.*sO.*xM{i} - rhoO0.*pv0.*sO0.*xM0{i} + ...
        rhoG.*pv.*sG.*yM{i} - rhoG0.*pv0.*sG0.*yM0{i} +...
        (s.gv.*model.EOSModel.fluid.isotherm(2,i) ...
        ./model.EOSModel.fluid.isotherm(1,i)).*...
        ((y{i}.*p)./(1+p.*sumIso)- (y0(:,i).*p0)./(1+p0.*sumIso0) )  );
else
    eqs{i} = (1/dt).*( ...
        pv.*rhoO.*sO.*xM{i} - pv0.*rhoO0.*sO0.*xM0{i} + ...
        pv.*rhoG.*sG.*yM{i} - pv0.*rhoG0.*sG0.*yM0{i});
end
```

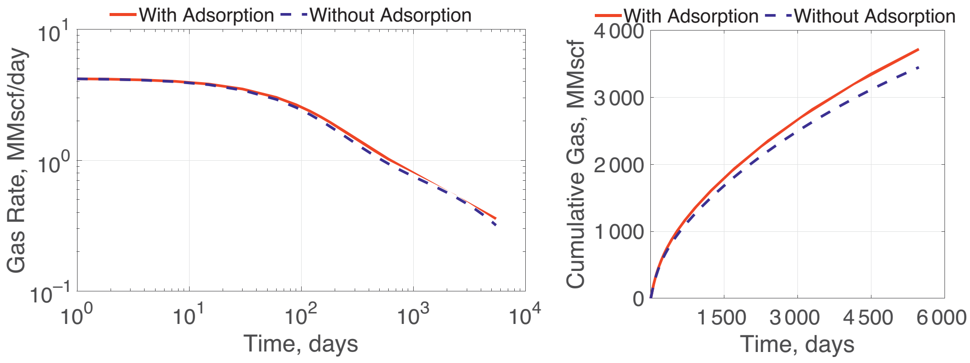

Figure 10.19 Comparison of (left) gas production rate and (right) cumulative gas production of a fractured reservoir with and without sorption. The incorporation of sorption in the model results in an increase of 7% in the cumulative gas production.

10.7.2 Molecular Diffusion

The two primary mass transport mechanisms considered in petroleum reservoirs are the advective (also referred to as convective) and diffusive mass transport mechanisms. Advective transport is driven by a pressure gradient and modeled using Darcy's law, whereas molecular diffusion is driven by a concentration gradient and is part of a more general phenomenon referred to as "hydrodynamic dispersion." Hydrodynamic dispersion basically incorporates both molecular diffusion and mechanical dispersion but tends to be dominated by molecular diffusion at the very low flow velocities expected in a shale matrix [17].

In conventional petroleum reservoirs under primary recovery, molecular diffusion is usually neglected because the high matrix permeability causes the advective mass transfer to dominate the mass transport mechanism. In such reservoirs, molecular diffusion is expected to be on the same order as the numerical dispersion associated with the typical block sizes used in field-scale reservoir simulation. However, in unconventional reservoirs with very low matrix permeability, the advective fluxes are much lower. Therefore, the contribution of the diffusive flux to the total mass flux could be significant and needs to be modeled. The next subsection presents Fick's law, which is used to model hydrodynamic dispersion or molecular diffusion.

Fick's Law

Fick's law is one of the most common models used to describe the diffusive transport of multicomponent mixtures due to a gradient in concentration. For the diffusion of a component i in a box filled with a single-phase fluid, it is written as

$$J_\alpha^i = -D_\alpha^i \nabla \left(\rho_\alpha X_\alpha^i\right), \tag{10.15}$$

where the product $\rho_\alpha X_\alpha^i$ refers to the mass concentration of component i in phase α. To account for the presence of multiple phases, a tortuous path, and a solid matrix in porous systems, the diffusion coefficient D_α^i (in m^2/s) of component i in phase α is typically multiplied by the porosity ϕ and saturation S and divided by tortuosity τ, which is the ratio of the actual length of the flow path in the porous medium to the thickness of the medium in the direction of the flow. The modified form of the Fickian diffusion for a porous medium is given as

$$J_\alpha^i = -\frac{\phi S_\alpha}{\tau_\alpha} D_\alpha^i \nabla \left(\rho_\alpha X_\alpha^i\right). \tag{10.16}$$

As in Lake et al. [17], the governing equation can be modified to include molecular diffusion by adding J_α^i (in kg m^{-2} s^{-1}) to (10.1) as follows:

$$\partial_t \left[\phi \sum_{\alpha=l,v} \rho_\alpha S_\alpha X_\alpha^i\right] + \sum_{\alpha=l,v} \nabla \cdot \left(\rho_\alpha X_\alpha^i \vec{v}_\alpha + J_\alpha^i\right) - \sum_{\alpha=l,v} \rho_\alpha X_\alpha^i q_\alpha / V = 0. \tag{10.17}$$

Implementation of Fickian Diffusion

The complete code that illustrates our implementation of Fickian diffusion is found as `diffusion.m` in the `examples` folder. As in the implementation of sorption, we turn diffusion on by setting the `diffusion` subfield of `G.rock.shaleMechanisms` to one. We also provide the two additional parameters needed to implement Fickian diffusion (D_α^i and τ). Considering that gases diffuse much faster than liquids, our focus in this section is on the implementation of diffusion in the gas phase only. We specify the effective diffusion coefficient using the values given in Xiong et al. [36], as shown:

```
G.rock.shaleMechanisms.diffusion = 1; % Turns diffusion on
G.rock.Di=[2.8,2.5,1.9]*10^-7;        % in m^2/s
G.rock.tau = 2;                       % Specify tortuosity
```

Finally, we implement Fickian diffusion by adding the diffusive flux equation (10.17) to the governing equations in `eqnsShaleNaturalVars`. This is done only when the diffusion transport mechanism is turned on, as shown:

```
if isfield(model.G.rock, 'shaleMechanisms') && ...
   isfield(model.G.rock.shaleMechanisms,'diffusion')
    Sg_rhoG_poro = s.faceUpstr(upcg,sG.*model.G.rock.poro);
    eqs{i} = eqs{i} - s.Div(model.G.rock.Di(i)./model.G.rock.tau ...
        .*Sg_rhoG_poro.*s.Grad(yM{i}.*rhoG));
end
```

Simulation Results with and without Diffusion

To illustrate the effect of molecular diffusion, we model the same Barnett shale-gas reservoir discussed in Subsection 10.7.1 but with a matrix permeability of 1 nD (instead of 100 nD, as in the table). The production profiles obtained with and without diffusion (given in Figure 10.20) show that molecular diffusion contributes up to 14% additional production when the matrix permeability is set to 1 nD. This extremely low permeability value results in the rather flat decline observed in the log–log rate plot after approximately 5 days. At the higher permeability values in the other shale mechanisms studied, the effect of molecular diffusion is negligible. The reader is encouraged to increase this matrix permeability in multiples of 10 and observe that this contribution becomes much less significant at higher permeability values. These results indicate that it is important to account for the diffusive transport mechanism in shale-gas reservoirs, which typically have very low matrix permeability.

10.7.3 Geomechanics Effect

As illustrated in Figure 10.1, UOG reservoirs contain multiscale fractures in addition to the hydraulic fractures we create. As reservoir fluids are produced from these reservoirs, the pore pressure reduces, leading to an increase in effective stress. These induced effective stresses are compressive and tend to close the fractures. The next subsection discusses one of the common models that have been applied to estimate the closure of natural fractures in UOG reservoirs.

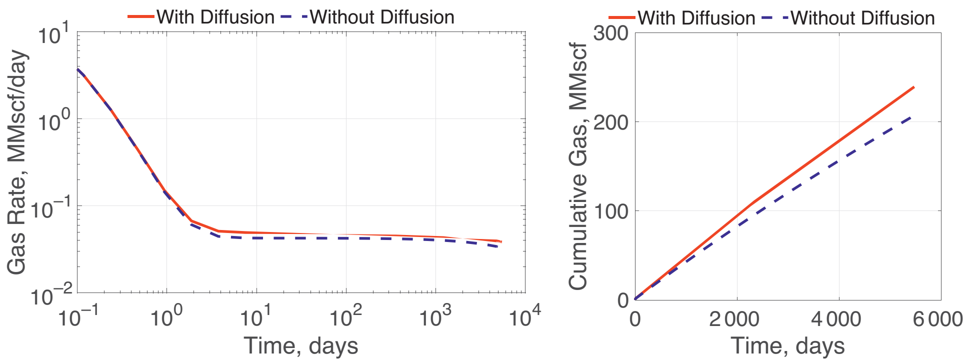

Figure 10.20 Comparison of (left) gas production rate and (right) cumulative gas production with and without diffusion. In this fractured reservoir with a matrix permeability of 1 nD, diffusion increases the cumulative production by 14%.

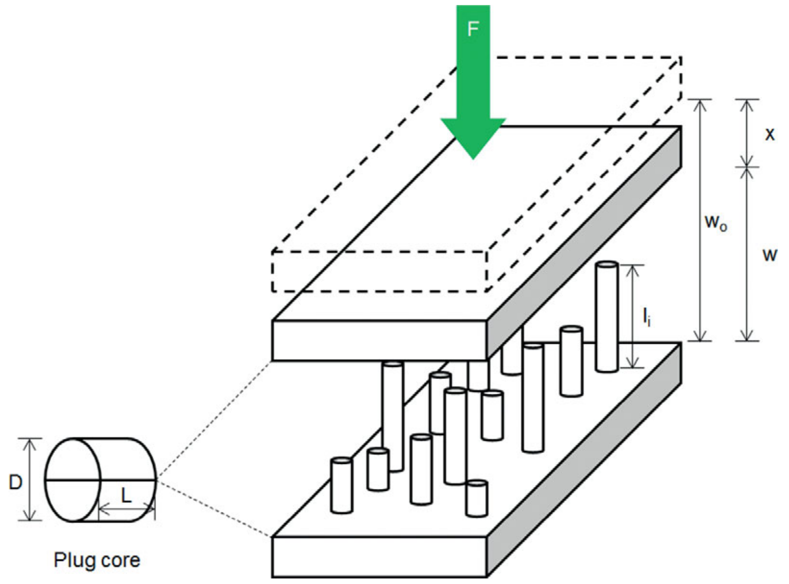

Figure 10.21 Modifed sketch of Gangi's "bed of nails" model from Wasaki [35], who used it to model changes in the effective permeability of fractured organic-rich source rocks.

Gangi Model

Gangi [7] developed a model to account for the change in the permeability of a fracture (or fractured rock) with pressure and confining stress. This model uses a conceptual bed of nails (Figure 10.21) to represent the expected fracture surface roughness or asperity. Considering that UOG reservoirs contain propped hydraulic

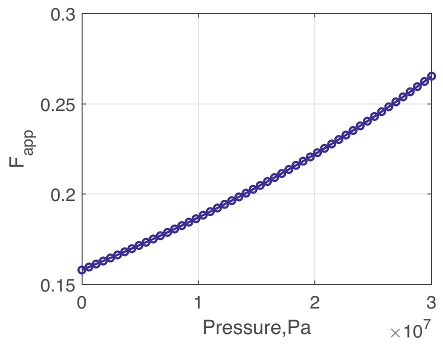

Figure 10.22 Permeability correction factor F_k versus pore pressure with $m = 0.5$, $p_L = 180$ MPa, $\sigma_c = 38$ MPa, and $\alpha = 0.5$. The reduction in F_k results in a decrease in the permeability (of a naturally fractured rock) as the pore pressure drops.

fractures in addition to the multiscale natural fractures, Olorode [25] modeled the change in the hydraulic fracture permeability using the analytical model from Guo and Liu [10] in a fully coupled geomechanics simulator, whereas the Gangi model was used to account for the "effective" permeability of organic-rich source rocks with multiscale fractures. The Gangi model [7] is given as

$$K = K_0 \left[1 - \left(\frac{\sigma_c - \alpha_B p}{\sigma_1} \right)^m \right]^3 , \tag{10.18}$$

where α_B is Biot's constant, σ_c is the confining stress, σ_1 is the maximum effective stress that closes the fracture completely, K_0 is the permeability at zero confining pressure, and m is a constant related to the surface roughness of the fracture. To simplify the implementation of the Gangi model, all of the terms on the right-hand side of the equation (with the exception of K_0) can be grouped together and referred to as the apparent Gangi permeability correction factor, F_k. Figure 10.22 presents a plot of this permeability correction factor against pressure. It shows that F_k (and consequently permeability) decreases as the pore pressure decreases.

Implementation of the Gangi Model

The complete code that demonstrates the implementation of the Gangi model is found in `gangi.m`. In Listing 10.6, we specify the Gangi model parameters and implement the equation in the `gangiFn` function. The actual implementation of the Gangi model in `gangiFn` is shown as the last line commented in Listing 10.6. The reader is encouraged to implement other types of pressure-dependent permeability functions in a similar manner.

It is important to note that K_0 is the matrix permeability at zero confining stress and is not the same as the matrix permeability at initial reservoir conditions. We divide K_0 by the initial matrix permeability, so that the result of this function can be multiplied by the standard flow velocity obtained from the Darcy equation.

Listing 10.6 *Gangi model implementation.*

```
Pc = 10000*psia;
alpha = 1.0;
Pmax = 14000*psia;
m = 0.4;
k0 = 2090.2259*nano*darcy;
flowfluid.KGangiFn = @(p) gangiFn(p, Pc, alpha, Pmax, m, k0, matrix_perm);
%gangiFn = (k0./matrix_perm).*power((1-power(((Pc - alpha.*p)./Pmax),m)),3);
```

In the modified `eqnsShaleNaturalVars`, we compute the apparent permeability correction factor and use the Boolean `isMatrix` variable to ensure that the Gangi model is only applied in the matrix cells. Because the number of items in `F_k` is equal to the number of cells in the domain but the number of items in the Darcy oil velocity, `vO`, is equal to the number of faces, we use the `splitFaceCellValue` function to map the cell `F_k` values to the corresponding faces. The standard Darcy flow velocity is then scaled to account for the pressure-dependent matrix permeability. Although the following code excerpt only shows the scaling of the oil velocity (`vO`), the same correction factor is applied to the gas and water velocities:

```
if isfield(model.G.rock, 'shaleMechanisms') && ...
   isfield(model.G.rock.shaleMechanisms, 'Gangi')
    F_k = model.fluid.KGangiFn(p);
    F_k(~s.isMatrix) = 1;
    [KGangif, ~] = s.splitFaceCellValue(s, upco, F_k);
    vO = (KGangif.*vO);
end
```

Simulation Results with and without the Pressure-Dependent Permeability

To illustrate the effect of a pressure-dependent matrix permeability, we model the representative Barnett shale-gas reservoir described at the beginning of this section. Figure 10.23 shows that the decrease in matrix permeability (as a result of fluid withdrawal) can lead to a 24% reduction in the cumulative gas production over 15 years. This result indicates that it is important to account for the pressure dependence of the matrix permeability in UOG reservoirs.

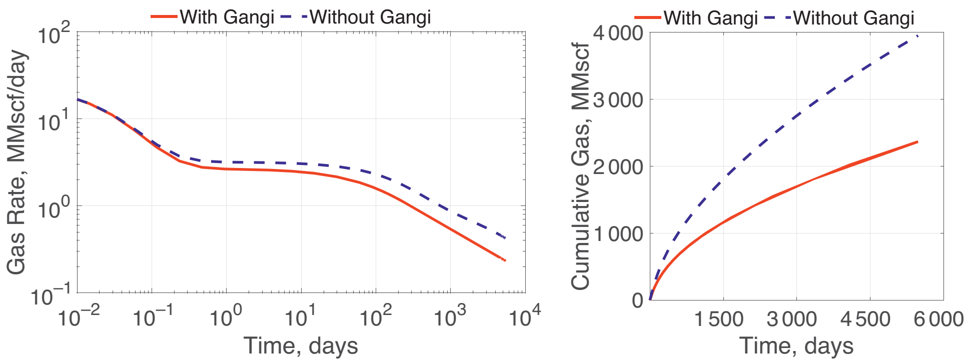

Figure 10.23 Comparison of (left) gas production rate and (right) cumulative gas production from a Barnett shale-gas reservoir with and without a pressure-dependent matrix permeability. The Gangi correction decreases the cumulative gas production by 24%.

Acknowledgement. The authors recognize the support of the Craft and Hawkins Department of Petroleum Engineering at Louisiana State University (LSU). This work was facilitated by the use of the computational resources and software packages available in the department and at the Center for Computation and Technology (CCT).

References

[1] Y. F. Alghalandis. *DFNE Practices with ADFNE*. Alghalandis Computing, Toronto, 2018.

[2] R. J. Ambrose. Micro-structure of gas shales and its effects on gas storage and production performance. PhD thesis, The University of Oklahoma, 2011. URL hdl.handle.net/11244/318875.

[3] S. Brunauer, P. H. Emmett, and E. Teller. Adsorption of gases in multimolecular layers. *Journal of the American Chemical Society*, 60(2):309–319, 1938. doi: 10.1021/ja01269a023.

[4] K. H. Coats. An equation of state compositional model. *SPE Journal*, 20(5):363–376, 1980. doi: 10.2118/8284-PA.

[5] D. A. Collins, L. X. Nghiem, Y. K. Li, and J. E. Grabonstotter. An efficient approach to adaptive-implicit compositional simulation with an equation of state. *SPE Reservoir Engineering*, 7(2):259–264, 1992. doi: 10.2118/15133-PA.

[6] B. R. Didar and I. Y. Akkutlu. Pore-size dependence of fluid phase behavior and properties in organic-rich shale reservoirs. In *SPE International Symposium on Oilfield Chemistry, 8–10 April, The Woodlands, Texas, USA*. Society of Petroleum Engineers, 2013. doi: 10.2118/164099-MS.

[7] A. F. Gangi. Variation of whole and fractured porous rock permeability with confining pressure. *International Journal of Rock Mechanics and Mining Sciences & Geomechanics Abstracts*, 15(5):249–257, 1978. doi: 10.1016/0148-9062(78)90957-9.

[8] B. Gong. Effective models of fractured systems. PhD thesis, Stanford University, 2007. URL pangea.stanford.edu/ERE/pdf/pereports/PhD/Gong07.pdf.

[9] B. Gong, G. Thakur, G. Qin, X. Peng, and W. Xu. Application of multi-level and high-resolution fracture modeling in field-scale reservoir simulation study. In *SPE Reservoir Characterisation and Simulation Conference and Exhibition, 8–10 May, Abu Dhabi, UAE*. Society of Petroleum Engineers, 2017. doi: 10.2118/186068-MS.

[10] J. Guo and Y. Liu. Modeling of proppant embedment: elastic deformation and creep deformation. In *SPE International Production and Operations Conference & Exhibition, 14–16 May, Doha, Qatar*. Society of Petroleum Engineers, 2012. doi: 10.2118/157449-MS.

[11] O. Houze, E. Tauzin, V. Artus, and L. Larsen. *The Analysis of Dynamic Data in Shale Gas Reservoirs – part 1 – version 2*. KAPPA Engineering, Houston, TX, 2010.

[12] O. Houze, D. Viturat, and O. S. Fjaere. *Dynamic Data Analysis v.5.20.02*. KAPPA Engineering, Houston, TX, 2019. URL https://www.kappaeng.com/papers.

[13] J. Jiang and R. M. Younis. An improved projection-based embedded discrete fracture model (pEDFM) for multiphase flow in fractured reservoirs. *Advances in Water Resources*, 109:267–289, 2017. doi: 10.1016/j.advwatres.2017.09.017.

[14] M. Karimi-Fard and A. Firoozabadi. Numerical simulation of water injection in 2d fractured media using discrete-fracture model. In *SPE Annual Technical Conference and Exhibition, September 30–October 3, New Orleans, Louisiana*. Society of Petroleum Engineers, 2001. doi: 10.2118/71615-MS.

[15] J.-G. Kim and M. D. Deo. Finite element, discrete-fracture model for multiphase flow in porous media. *AIChE Journal*, 46(6):1120–1130, 2000. doi: 10.1002/aic.690460604.

[16] U. Kuila and M. Prasad. Understanding pore-structure and permeability in shales. In *SPE Annual Technical Conference and Exhibition*, Denver, Colorado, USA, 2011. Society of Petroleum Engineers. doi: 10.2118/146869-MS.

[17] L. W. Lake, R. Johns, W. R. Rossen, and G. A. Pope. *Fundamentals of Enhanced Oil Recovery*. Society of Petroleum Engineers, Richardson, TX, 2014.

[18] I. Langmuir. The constitution and fundamental properties of solids and liquids. Part I. Solids. *Journal of the American Chemical Society*, 38(11):2221–2295, 1916. doi: 10.1021/ja02268a002.

[19] L. Li and S. H. Lee. Efficient field-scale simulation of black oil in a naturally fractured reservoir through discrete fracture networks and homogenized media. *SPE Reservoir Evaluation & Engineering*, 11(4):750–758, 2008. doi: 10.2118/103901-PA.

[20] L. Li and D. Voskov. Multi-level discrete fracture model for carbonate reservoirs. In *ECMOR XVI – 16th European Conference on the Mathematics of Oil Recovery, Barcelona, Catalonia, Spain, September 3–6*. European Association of Geoscientists & Engineers, 2018. doi: 10.3997/2214-4609.201802164.

[21] A. Moinfar. Development of an efficient embedded discrete fracture model for 3D compositional reservoir simulation in fractured reservoirs. PhD thesis, The University of Texas at Austin, 2013. URL hdl.handle.net/2152/21393.

[22] G. Moridis, T. Blasingame, and C. Freeman. Analysis of mechanisms of flow in fractured tight-gas and shale-gas reservoirs. *SPE Latin American and Caribbean Petroleum Engineering Conference Proceedings, 1–3 December, Lima, Peru*, 2010. doi: 10.2118/139250-MS.

[23] O. Møyner and H. A. Tchelepi. A mass-conservative sequential implicit multiscale method for isothermal equation-of-state compositional problems. *SPE Journal*, 23(6):2–376, 2018. doi: 10.2118/182679-PA.

[24] O. M. Olorode. Numerical modeling of fractured shale-gas and tight-gas reservoirs using unstructured grids. Master's thesis, Texas A&M Univerisity, College Station, 2012. URL hdl.handle.net/1969.1/ETD-TAMU-2011-12-10286.

[25] O. M. Olorode. A new multicontinuum model for compositional gas transport in a deformable shale formation. PhD thesis, Texas A&M Univerisity, College Station, 2017. URL hdl.handle.net/1969.1/165828.

[26] O. M. Olorode, I. Y. Akkutlu, and Y. Efendiev. Compositional reservoir-flow simulation for organic-rich gas shale. *SPE Journal*, 22(6):1963–1983, 2017. doi: 10.2118/182667-PA.

[27] O. M. Olorode, B. Wang, and H. U. Rashid. Three-dimensional projection-based embedded discrete fracture model for compositional simulation of fractured reservoirs. *SPE Journal*, 25(4):2143–2161, 2020. doi: 10.2118/201243-PA.

[28] A. Ougier-Simonin, F. Renard, C. Boehm, and S. Vidal-Gilbert. Microfracturing and microporosity in shales. *Earth-Science Reviews*, 162:198–226, 2016. doi: 10.1016/j.earscirev.2016.09.006.

[29] D.-Y. Peng and D. B. Robinson. A new two-constant equation of state. *Industrial & Engineering Chemistry Fundamentals*, 15(1):59–64, 1976. doi: 10.1021/i160057a011.

[30] K. Pruess and T. N. Narasimhan. A practical method for modeling fluid and heat flow in fractured porous media. *SPE Journal*, 25(1):14–26, 1982. doi: 10.2118/10509-PA.

[31] A. Shafiei, M. B. Dusseault, E. Kosari, and M. N. Taleghani. Natural fractures characterization and in situ stresses inference in a carbonate reservoir – an integrated approach. *Energies*, 11(2):312, 2018. doi: 10.3390/en11020312.

[32] M. Ţene, S. B. M. Bosma, M. S. A. Kobaisi, and H. Hajibeygi. Projection-based embedded discrete fracture model (pEDFM). *Advances in Water Resources*, 105: 205–216, 2017. doi: 10.1016/j.advwatres.2017.05.009.

[33] D. V. Voskov and H. A. Tchelepi. Comparison of nonlinear formulations for two-phase multi-component EoS based simulation. *Journal of Petroleum Science and Engineering*, 82-83:101–111, 2 2012. doi: 10.1016/j.petrol.2011.10.012.

[34] J. E. Warren and P. J. Root. The behavior of naturally fractured reservoirs. *SPE Journal*, 3(3):245–255, 1963. doi: 10.2118/426-PA.

[35] A. Wasaki. Dynamics of matrix-fracture coupling during shale gas production. Master's thesis, Texas A&M University, 2015. URL hdl.handle.net/1969.1/155369.

[36] Y. Xiong, P. Winterfeld, C. Wang, Z. Huang, and Y.-S. Wu. Effect of large capillary pressure on fluid flow and transport in stress-sensitive tight oil reservoirs. In *SPE Annual Technical Conference and Exhibition, 28–30 September, Houston, Texas, USA*. Society of Petroleum Engineers, 2015. doi: 10.2118/175074-MS.

[37] W. Yu, Y. Zhang, A. Varavei, K. Sepehrnoori, T. Zhang, K. Wu, and J. Miao. Compositional simulation of CO_2 huff-n-puff in Eagle Ford tight oil reservoirs with CO_2 molecular diffusion, nanopore confinement, and complex natural fractures. *SPE Reservoir Evaluation & Engineering*, 22(2):492–508, 2019. doi: 10.2118/190325-PA.

[38] M. D. Zoback. *Reservoir Geomechanics*. Cambridge University Press, Cambridge, UK, 2010. doi:10.1017/CBO9780511586477.