11

A Unified Framework for Flow Simulation in Fractured Reservoirs

RAFAEL MARCH, CHRISTINE MAIER, FLORIAN DOSTER, AND SEBASTIAN GEIGER

Abstract

Simulation of multiphase flow in fractured reservoirs still poses a challenge due to the different timescales of fluid flow in fractures and matrix. Common approaches to modeling fractures in reservoir simulators include the discrete fracture and matrix (DFM) method, where the fractures are explicitly represented as lower-dimensional elements in the computational mesh, and multicontinuum approaches (e.g., dual-porosity and dual-permeability models) where the behavior of the fractures and matrix are integrated and treated as distinct continua. The latter requires models (bespoke "transfer functions") that upscale the multiphase transfer between fracture and matrix. There are several formulations for transfer functions available in the literature, and they are often application dependent.

Here, we propose a unified framework for simulation of flow in fractured media. The framework makes no distinction between dual-continuum and DFM methods, treating fractures and one or more matrix domains as flowing domains and virtual domains. Transfer functions are reinterpreted as fluxes between cells of different domains. This enables us to create an abstraction that encompasses both methods and makes it easy to build hybridized models including different regions with different matrix/fracture interaction concepts. We present a series of cases to illustrate the main differences between both modeling approaches and the benefit of a flexible implementation that enables the development of a fit-for-purpose simulator for fractured reservoirs based on automatic differentiation.

11.1 Introduction

Modeling of fluid flow in fractured rocks is relevant for a variety of applications that involve the use of the subsurface, such as recovery of oil and gas from hydrocarbon reservoirs, enhanced geothermal systems, and geological storage of CO_2.

https://doi.org/10.1017/9781009019781.017 Published online by Cambridge University Press

Enhanced oil recovery (EOR) techniques are typically used to economically extract hydrocarbons from fractured reservoirs. These techniques often rely on injection cycles of aqueous and gaseous fluids to exploit capillary and gravitational forces to sweep oil and gas toward the production wells [7, 10, 37] (see Chapter 7 for EOR simulations using the MATLAB Reservoir Simulation Toolbox [MRST]). The presence of fractures can lead to a series of undesired operational problems, such as the early breakthrough of injected fluids, that can severely impact hydrocarbon production and the economics of the project [45].

Geothermal energy is a clean and sustainable resource that is based on extracting heat from the subsurface of the Earth. Conventional natural geothermal systems are typically restricted to volcanic areas, which hinders the contribution of this form of energy to the global energy portfolio. The quest for improving the contribution of this resource has led to the development of enhanced geothermal systems (EGS) [6]. In EGS, hydraulic stimulation is used to engineer fracture-hosted permeability for extracting heat from low-permeability rocks outside volcanic areas. The performance of EGS and the control of circulation efficiency relies heavily on the modeling of fluid and heat flow in the fractured host rock [17].

Geological storage of CO_2 consists of injecting CO_2 in subsurface formations and is regarded as a key technology to decrease the concentration of greenhouse gases in the atmosphere and reduce its impact on the global climate. Fractured reservoirs may offer significant potential for combining CO_2 storage with enhanced oil recovery [2] or for storage in fractured saline aquifers [34]. In these applications, it is important to know how fast CO_2 will travel in the fracture system, how fast CO_2 will enter the rock matrix, and how much CO_2 will be trapped in the matrix [34, 35].

All the abovementioned applications have in common the need for understanding and quantifying fluid flow and transport in fractured formations. Although laboratory experiments are important to obtain insights about the main physical mechanisms, they fail to reproduce the space and timescales seen in natural systems. Hence, numerical simulations arise as key tools to not only understand the behavior of fluid flow in fractured rocks but also provide quantitative estimates of fluid storage and extraction potential.

In this chapter, we review the main techniques for simulating multiphase flow in fractured reservoirs and introduce `fractures`: an MRST module to enable flexible implementation and evaluation of techniques for simulation of fractured reservoirs. The `fractures` module leverages the automatic differentiation framework of MRST to provide a platform for quick development of fit-for-purpose simulators for fractured reservoirs based on automatic differentiation (AD).

This chapter is organized as follows. In Section 11.2 we describe the challenges found when modeling fluid flow in fractured reservoirs and explain the main

https://doi.org/10.1017/9781009019781.017 Published online by Cambridge University Press

modeling techniques. In Section 11.3 we describe the implementation of the `fractures` module, focusing on the utilization of MRST's AD framework to achieve a flexible and extensible implementation. In Section 11.4 we present some simple applications that outline how the package may be used to obtain insight about fluid flow in fractured formations for different geo-energy-related applications. Finally, in Section 11.5 we present the summary and conclusions of this chapter.

11.2 Modeling and Simulation Techniques for Fractured Reservoirs

The geological, physical, and mathematical modeling of flow in fractured reservoirs is a challenging task. Figure 11.1 shows pictures of two outcrops featuring fracture networks. In these illustrations we see that fractures are narrow zones of material discontinuity in the underlying rock. Fractures are often thought of as surfaces but exhibit a thickness at some spatial scale (here referred to as *fracture aperture*). Their distribution can be sparse, with just a few fractures spread over a domain of interest, or dense, with thousands of fractures located in a typical simulation grid-block. Fractures can occur isolated or form large connected networks. In any case, fractures typically dominate the flow behavior.

Fractures are usually invisible to seismic surveys and other geophysical tools but are often too large to be studied in cores obtained from wells [13]. It is hence impossible to obtain a mapping of a real 3D fracture network. Geological models of fracture systems therefore rely on statistical representations, which in turn require the simulation of many realizations, making computationally effective tools paramount.

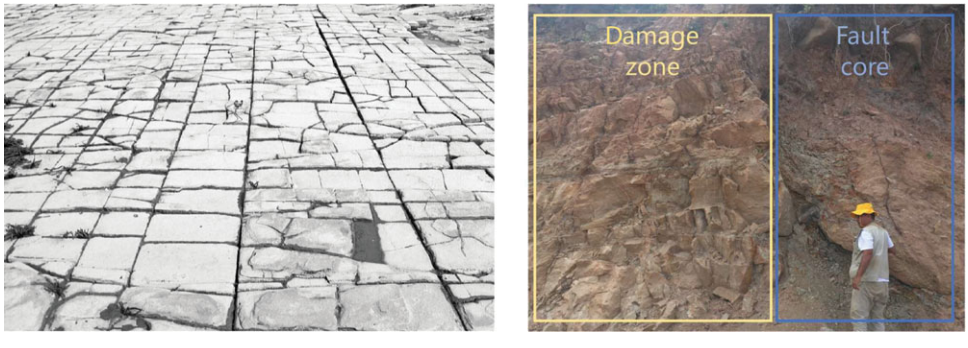

Figure 11.1 Two examples of outcrops displaying fracture networks. (left) Fracture network from the Bristol channel basin, UK, and (right) a fault outcrop in the Araripe basin, northeast Brazil.

https://doi.org/10.1017/9781009019781.017 Published online by Cambridge University Press

Often fracture distributions follow scale-invariant power-law distributions which preclude the definition of a characteristic length scale. Hence, effective upscaled representations have to be applied cautiously. Fractures can be void and enhance flow or be filled with sealing material preventing flow. They can be the only available pore space (e.g., in granite geothermal reservoirs) or act as main conduits, whereas the matrix contains most of the volume (e.g., shale-gas reservoirs). The density of fractures can vary substantially within a domain of interest as shown in the picture of a fault damage zone to the right in Figure 11.1, where the fracture density increases next to the fault core. Given these complexities, it is no surprise that there are many different techniques available to simulate flow in fractured porous rocks. None of them is a panacea. The ideal technique to model fluid flow in a particular fractured reservoir should take into account the characterization of the fracture system and the individual fractures (e.g., fracture connectivity and fracture density), the amount of information available, and the goal of the simulation model (a simplified model might suit if the goal is to obtain an order of magnitude estimate of a certain engineering variable).

Techniques to simulate flow in fractured porous rocks are typically separated into two large families of methods [5]. The first family includes methodologies that explicitly represent the fractures as geometrical features. Pertaining to this family are the discrete fracture and matrix (DFM) methods [12, 15, 18, 36] and the embedded discrete fracture model (EDFM) [25]. The second family of methods considers the fractures as a second continuum. Of particular importance are the dual-porosity [4, 49] and dual-porosity dual-permeability [24] methods, together with their extensions like the multiple interacting continua (MINC) [38], multirate dual-porosity [16, 32], and multiple subregions [19, 22] methods.

Perhaps the most used variation of the DFM method is the one introduced in [21]. In this work, the authors propose representing each fracture explicitly as virtual cells; that is, cells that are present in the final linear system but not explicitly present in the geometrical mesh. This is done by considering fractures as lower-dimension elements that match the faces (or edges, for 2D meshes) of the mesh elements. Each face of the mesh that is marked as a fracture will hold additional degrees of freedom to represent fields that are stored in the virtual fracture cell. Intersections between multiple fractures are treated via the "star-delta" transformation that computes the transmissibility between the different fracture cells without the need of explicitly accounting for a small element in the intersection. Though the DFM method is useful for applications (e.g., upscaling of fracture properties), conforming the computational mesh to a complex system of fractures can be challenging, time-consuming or even computationally unfeasible.

The EDFM method solves the meshing issue by embedding fractures within the simulation grid blocks. The method is implemented in the MRST module `hfm` and

https://doi.org/10.1017/9781009019781.017 Published online by Cambridge University Press

is discussed in Chapter 9. In the EDFM approach, the fractures do not need to match edges/faces of the computational grid and the meshing of the fractures is completely independent from the matrix. This enables, for instance, the utilization of Cartesian or corner-point grids with arbitrary systems of fractures. In EDFM, the conservation equations are solved for fracture and matrix separately, and matrix–fracture transfer is modeled as source terms in this equation. The transmissibility between fracture and matrix cells is calculated by considering the potential difference between the centroids of these cells and the cross-sectional area of the fracture cell within the grid block. The EDFM has been applied successfully to complex fracture patterns. A currently known limitation of this approach is its inability to simulate rocks with anisotropic matrix permeability. Moreover, the preprocessing step that computes fracture–matrix and fracture–fracture intersections is computationally intensive.

The second family of methods, the multicontinuum methods, are popular in the reservoir simulation community due to their computational efficiency. They are based on representing the fracture system as a separate continuum in addition to rock matrix instead of representing each individual fracture. The communication between fracture system and rock matrix system is modeled by mass transfer functions that aim to capture the dynamics of the fluid exchange between fractures and the matrix. Representing the transfer between fractures and the matrix through simple differential equations based on continuum quantities is a very challenging task: though the first transfer functions were developed in the 1960s [49], the development of transfer functions that correctly capture the physics of flux exchange between different continua is still an active field of research.

The first transfer functions were based on scaling the pressure potential difference between fracture and matrix, which makes them simple to implement in any conventional reservoir simulator. However, a known problem with this formulation is that it fails to capture the early-time transfer behavior during imbibition into the matrix. To solve this problem, several authors have attempted to develop transfer functions that carry more physical knowledge about the fracture–matrix system [29, 33, 34, 42, 47, 48].

Despite these challenges, multicontinuum methods address the main limitation of methods that represent fractures explicitly: They do not require the representation of all fractures at the reservoir scale and hence increase the computational efficiency. Most important, the exact location of the fractures in the subsurface is generally not known: there are no effective methods to obtain such information.

In summary, both families of techniques have their realm in the modeling of fluid flow in fractured reservoirs. Often, we find that some combination of these techniques forms a powerful tool to evaluate fluid flow in fractured reservoirs

[30, 31, 50]. For instance, a typical workflow involves generating stochastic realizations of fracture networks, running fluid flow simulations with these fractures explicitly on a smaller spatial scale, upscaling the effective properties of the fracture system (such as permeability of the fracture network and the fracture–matrix transfer), and finally using these effective models as precomputed proxies for simulation of flow at the field scale with a multicontinuum method.

In the next subsections we will take a deeper look into the mathematical and computational details of multicontinuum and explicit fracture models. We will focus our attention on dual-porosity models and DFM models, because these are the most widely used models for flow simulation in fractured reservoirs. However, we emphasize that the computational package described in this chapter allows for quick implementation of other important techniques such as the MINC and EDFM models.

11.2.1 Governing Equations

Before we start introducing the different methods for simulation of flow in fractured reservoirs, it is useful to specify the physical model we consider in this chapter. For a comprehensive presentation of the conservation equations that govern flow in porous media, we refer to [27]. We consider immiscible flow of three phases, aqueous, oleic, and gaseous, represented by the subscripts w, o, and g, respectively. The mass conservation equations for this system are given by

$$\frac{\partial \phi \rho_\alpha S_\alpha}{\partial t} + \nabla \cdot (\rho_\alpha \vec{v}_\alpha) = \rho_\alpha q_\alpha, \tag{11.1}$$

where $\alpha \in \{w, o, g\}$, ϕ is the rock porosity, ρ_α is the specific mass of phase α, S_α is the saturation of phase α, q_α are volumetric sources/sinks of phase α due to wells and/or boundary conditions, and $\vec{v}_\alpha$ is the Darcy velocity of phase α. We will often assume that the fluids are compressible, and we may normalize the density of phase α by its density at standard conditions: $\rho_\alpha = \rho_\alpha^{std} b_\alpha (p_\alpha)$, where b_α is the *shrinkage factor*. The Darcy velocity of phase α is given by

$$\vec{v}_\alpha = -\mathbf{K} \lambda_\alpha (\nabla p_\alpha - \rho_\alpha \vec{g}), \tag{11.2}$$

where $\mathbf{K}$ is the permeability tensor, λ_α and p_α are the mobility and pressure of phase α, and $\vec{g}$ is the gravity acceleration vector. Mobility is defined as the ratio between the relative permeability to phase α and the viscosity of phase α: $\lambda_\alpha = k_{r\alpha}/\mu_\alpha$.

After inserting Darcy's law (11.2) into the mass conservation equations (11.1), the model consists of three equations for six unknowns (S_w, S_o, S_g, p_w, p_o, p_g). The remaining three equations to close the system are the restriction that saturations should sum up to one ($S_w + S_o + S_g = 1$) and two capillary pressure models.

https://doi.org/10.1017/9781009019781.017 Published online by Cambridge University Press

Capillary pressure is usually assumed to be a saturation-dependent function that establishes a relationship between the phase pressures. We consider in this work that p_w and p_o are related via a capillary pressure function that only depends on water saturation: $p_o = p_w + p_{cow}(S_w)$. Further, the pressures p_o and p_g relate through a capillary pressure function that only depends on gas saturation: $p_g = p_o + p_{cog}(S_g)$. The underlying concept assumes that there is a clear hierarchy in wetting behavior, with water being the most and gas the least wetting fluid, and that the interface structure between oil and water is as if the gas was oil. Similarly, it assumes that the interface structure between oil and gas is identical to a situation where all water was oil. Power-law models like the Brooks–Corey model are usually assumed for $k_{r\alpha}$ and p_c; see the MRST textbook [27, section 8.1].

11.2.2 Multicontinuum Models

In the multicontinuum approach, the fracture network is represented as a continuum that is superposed to the rock matrix. The continua interact by a transfer term τ that models the mass exchange between them. We focus here on a widely used formulation of multicontinuum models: the dual-porosity model. This formulation assumes the existence of two continua, one for fractures and one for matrix, usually called the "flowing" and "stagnant" domains. Hence, (11.1) is extended by a set of conservation equations for the rock matrix:

$$\frac{\partial}{\partial t}(\phi_m \rho_{\alpha m} S_{\alpha m}) = \tau_\alpha. \tag{11.3}$$

In the remainder of this chapter, we use subscripts m and f to denote matrix and fracture continua, respectively. The transfer-rate term τ_α models the rate of mass exchange of phase α between fracture and matrix per unit bulk volume. Note that, to ensure conservation of mass, we need to add an equivalent sink term $-\tau_\alpha$ to the right-hand side of the flowing domain equations (11.1).

Several formulations for τ_α have been suggested in the past decades. The first model was introduced in [49] for a single compressible phase. The multiphase extension presented in [24] consists of scaling the fracture–matrix pressure potential by a transmissibility between the continua:

$$\tau_\alpha = \sigma \rho_\alpha k_m \lambda_\alpha \left(p_{\alpha f} - p_{\alpha m}\right), \tag{11.4}$$

where k_m is a representative value of the matrix permeability and σ is the *shape factor*, which has units of $[1/L^2]$ and represents the matrix-block geometry. The shape factor encompasses the matrix-block area open for fluid exchange and the distance between the point in the matrix block where the matrix pressure is represented. There are several formulations for the shape factor; some of them are presented in Table 11.1.

https://doi.org/10.1017/9781009019781.017 Published online by Cambridge University Press

Table 11.1 *Summary of shape factors implemented in the* `fractures` *module. Shape factor classes are placed under the* `transfer-functions` *directory. For further details on the module folder structure, see Subsection 11.3.3.*

Reference	Shape factor (σ)	Class name
Warren and Root (1963) [49]	$60/L^2$, $L = L_x = L_y = L_z$	`WarrenRootShapeFactor`
Kazemi et al. (1976) [24]	$4\left(\frac{1}{L_x^2} + \frac{1}{L_y^2} + \frac{1}{L_z^2}\right)$	`KazemiShapeFactor`
Coats (1989) [8]	$24/L^2$, $L = L_x = L_y = L_z$	`CoatsShapeFactor`
Lim and Aziz (1995) [28]	$\frac{\pi^2}{\bar{k}_m}\left(\frac{k_{mx}}{L_x^2} + \frac{k_{my}}{L_y^2} + \frac{k_{mz}}{L_z^2}\right)$, $\bar{k}_m = (k_{mx}k_{my}k_{mz})^{1/3}$	`LimAzizShapeFactor`

The transfer function given by (11.4), with some minor modifications, is used in the majority of the academic and commercial simulators that solve dual-porosity systems. One of the modifications of this transfer function was suggested in [43] to capture gravity-drainage effects. The modified transfer function takes the following form:

$$\begin{aligned}\tau_o &= \sigma\rho_o k_m \lambda_o \left[\left(p_{of} - p_{om}\right) - \tfrac{1}{2}\left(\rho_o - \rho_g\right) g\left(h_{gf}^e - h_{gm}^e\right)L_z\right],\\ \tau_g &= \sigma\rho_g k_m \lambda_g \left[\left(p_{gf} - p_{gm}\right) + \tfrac{1}{2}\left(\rho_o - \rho_g\right) g\left(h_{gf}^e - h_{gm}^e\right)L_z\right],\end{aligned} \tag{11.5}$$

with

$$h_{gf}^e = \frac{S_{gf} - S_{gr}}{1 - S_{or} - S_{gr}}, \qquad h_{gm}^e = \frac{S_{gm} - S_{gr}}{1 - S_{or} - S_{gr}}, \tag{11.6}$$

where L_z is the matrix-block height. With this correction, the gravity potential is split between wetting and nonwetting phases. This particular formulation of the transfer function with the gravity-transfer correction is implemented in the widely used commercial reservoir simulators Eclipse [41] and IMEX [9].

The family of transfer functions defined by (11.4) and (11.5) assumes that the matrix–fracture transfer is the same across all of the matrix-block faces. To obtain a more accurate representation of the transfer physics, it was suggested in [39] that the transfer across each face is treated independently, and the total transfer is given by the sum over all matrix-block faces:

$$\tau_\alpha = \frac{1}{L_x L_y L_z}\sum_i \tau_\alpha^i, \tag{11.7}$$

where L_x, L_y, and L_z are the block dimensions and τ_α^i stands for the transfer of phase α across face i. The face transfer is written as

$$\tau_\alpha^i = -\sigma^i \rho_\alpha k_m^i \lambda_\alpha^i \left(\Phi_{\alpha f}^i - \Phi_{\alpha m}^s\right), \tag{11.8}$$

https://doi.org/10.1017/9781009019781.017 Published online by Cambridge University Press

where $\Phi^i_{\alpha f}$ stands for the potential of phase α at the fracture side of face i, $\Phi^s_{\alpha m}$ is the potential at the center of the matrix block, and σ^i is the shape factor associated with face i. In theory, the separation of the transfer across each face enables the treatment of block anisotropy and the representation of gravity drainage. However, in a later study, the authors acknowledge that the computation of the transfer across each face of the matrix block does not fully solve the transfer-function modeling problem [26]. They recommend that simulations of fully-resolved matrix blocks are carried out to calibrate the transfer model.

A fundamentally different approach is based on the "divide and conquer" concept: Each transfer mechanism is modeled separately and the total transfer between the continua is obtained by the sum of the transfers of the individual mechanisms. Modeling each transfer mechanism separately has been done since the work of [39], but the different models to capture the different physical transfer mechanisms were combined in the same transfer for the first time in [29]. The exponential model of [3] has been regularly used to model the transfer due to spontaneous imbibition and gravity drainage separately [11, 23]. It leads to a transfer function that is linear with respect to the saturation in the matrix:

$$\tau_\alpha = \beta \phi_m \left(S^*_{\alpha m} - S_{\alpha m} \right), \tag{11.9}$$

where β is the transfer-rate coefficient that models the speed of the transfer process and $S^*_{\alpha m}$ stands for the maximum saturation of phase α that the matrix block can hold. For gravity drainage, $S^*_{\alpha m}$ is determined by the capillarity–gravity equilibrium, whereas $S^*_{\alpha m} = 1 - S_{or}$ for spontaneous imbibition (or $S^*_{\alpha m} = 1 - S_{gr}$, depending on the phases present in the system).

There are several expressions to obtain β for imbibition and drainage processes; unfortunately, most of them have limitations. For spontaneous imbibition, the most accurate expression suggested in [42] is based on the analytical solution for spontaneous imbibition in a matrix block and captures reasonably well the general trend of experimental data available in the literature. However, it underestimates the early-time imbibition behavior that scales with the square root of time. In [33], the authors suggested a way to combine the analytical solution for imbibition having the appropriate $\sqrt{t}$ scaling in early time with the late-time solution of Schmid and Geiger [42]. The drawback of this solution is that it includes an explicit dependency on time, which makes it challenging to implement in simulators. For gravity drainage, the formulation suggested in [11] is focused on oil and gas systems and relies on fitting parameters based on high-resolution numerical simulations. This approach was improved in [34], where the authors revisited the methodology for estimating β based on the timescales of fluid flow and developed a model that is more accurate and does not require fitting to fully-resolved block simulations.

https://doi.org/10.1017/9781009019781.017 Published online by Cambridge University Press

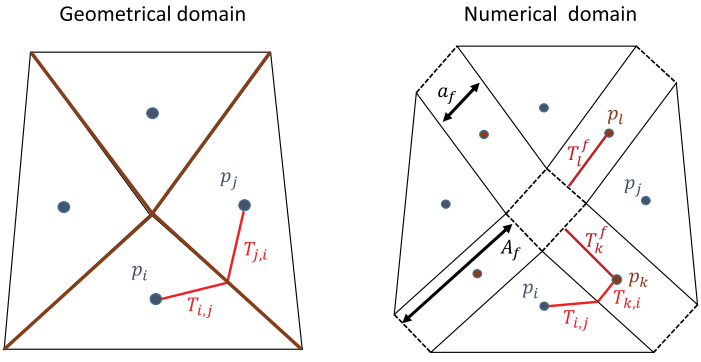

Figure 11.2 Schematic: geometrical and numerical domains for a 2D fracture intersection.

In summary, there are numerous ways to model the matrix–fracture transfer in dual-porosity simulators. The presented list is not exhaustive and we refer to [1, 40] for comprehensive reviews. All formulations have their weaknesses and strong points, and they are usually developed to suit a particular application and physical setting. This is one of the main motivations behind the development of this MRST-based framework: providing the modeler with a toolbox that enables the development of a fit-for-purpose simulator for fractured reservoirs.

11.2.3 Discrete Fracture and Matrix Model

The DFM model is the most fundamental approach to model fluid flow in a fractured porous media, because it represents the discontinuities explicitly. The representation of fractures as volumetric entities in the geometrical mesh would require prohibitively fine meshes. Hence, to reduce computational time, the use of lower-dimensional entities for fractures (1D lines in a 2D model and 2D surfaces in a 3D model) was introduced by [21]. We focus on 2D fracture geometries in this section.

In DFM, the fractures need to be matched to the edges (or faces) of the computational grid. Although the fractures are not volumetric elements in the grid, they still retain their actual aperture a_f through a property assigned to the edges associated with the fracture system (see Figure 11.2). The degrees of freedom associated with the fractures are added to the final system of equations as virtual cells with a pore volume of $V_f = a_f A_f$, where A_f is the surface area of the fracture. The area A_f reduces to the length of the fracture edge in 2D.

The two-point flux approximation involves approximating the flux across each face using fields that live in the two cells connected by the face. The flux of phase α across the face is written as

https://doi.org/10.1017/9781009019781.017 Published online by Cambridge University Press

$$v_{\alpha ij} = \left(T_{i,j}^{-1} + T_{j,i}^{-1}\right)^{-1} \lambda_\alpha \left(p_{\alpha i} - p_{\alpha j}\right) \equiv T_{ij} \lambda_\alpha \left(p_{\alpha i} - p_{\alpha j}\right), \tag{11.10}$$

where $T_{i,j}$ is the one-side transmissibility of cell i associated to the face that connects cells i and j and $T_{j,i}$ is the one-side transmissibility of cell j associated to the face that connects cells j and i. The face transmissibility between cells i and j is denoted by T_{ij} and is calculated as the harmonic average between the one-side transmissibilities. (For more details on the derivation of the two-point flux approximation, please refer to the MRST textbook [27].)

Now, to reflect the additional degrees of freedom that live in the virtual cells representing the fractures, the list of cell connections and the associated transmissibilities need to be updated. Each existing connection between matrix cells i and j that are separated by a fracture edge of index k needs to be replaced by two new matrix–fracture connections T_{ik} and T_{jk}. The transmissibilities between fracture and matrix are calculated as

$$T_{ik} = \left(T_{i,j}^{-1} + T_{k,i}^{-1}\right)^{-1}. \tag{11.11}$$

The one-side transmissibility between fracture and matrix is computed as

$$T_{k,i} = \frac{2A_k k_k}{a_k}, \tag{11.12}$$

where k_k is the fracture edge permeability.

Equation (11.11) defines the transmissibility between the fracture edges and the matrix cells separated by them. To complete the construction of the DFM formulation, we need to define the transmissibility between the fracture edges. The transmissibility between two fracture edges is computed based on the number of fracture edges that are connected to the node that separates these edges. We write the transmissibility between fracture segment k and l as

$$T_{kl}^f = \frac{T_k^f T_l^f}{\sum_{i=1}^n T_i^f}, \qquad T_i^f = \frac{2a_i k_i}{A_i}, \tag{11.13}$$

where n is the number of fracture edges connected to the intersection node and T_i^f is the fracture–fracture one-side transmissibility.

This formulation avoids creating virtual cells with small volume at the intersection nodes. It is called star-delta transformation and was first introduced in [21]. Note that in the case of $n = 2$ segments the transmissibility collapses to a standard fracture–fracture cell connection. Therefore, we can treat all fracture–fracture connections in the same way without any special treatment to intersection nodes.

https://doi.org/10.1017/9781009019781.017 Published online by Cambridge University Press

In 3D, intersections between fracture planes are lines, and there might be flow along these lines. This requires treatment of lower-dimensional flowing domains within the computational grid. We refer to [44] for a formal treatment of the interaction between domains of different dimensions.

11.3 Implementation in MRST

The different techniques to simulate flow in fractured reservoirs presented in the previous section appear very different and a general formulation unifying all of them may seem out of reach. Multicontinuum techniques are usually formulated as different continua that interact with the flowing continuum via source terms. On the other hand, DFM methods introduce virtual cells that live in the edges and interact with the geometrical mesh via transmissibilities that quantify the matrix–fracture fluid exchange.

However, the methods actually have more in common than meets the eye. Both families of methods involve a series of common steps:

1. The definition of one or more *regions* (sets of edges and/or cells in the physical grid) with virtual cells that represent additional continua;
2. The definition of connections between the grid cells and the virtual cells;
3. The definition of rock and fluid properties for the virtual cells (e.g., relative permeabilities, porosity, permeability, capillary pressure).
4. The definition of a *flux model* that defines how the flux of each phase between the cells is calculated.

In the following subsections, we show that these steps can all be naturally accommodated considering the finite-volume method with the two-point flux approximation in the general framework of MRST. We will present the computational background behind the `fractures` module, which provides a framework for quick implementation of models for flow simulation in fractured reservoirs.

11.3.1 Multicontinuum and Discrete Fracture and Matrix Models

The finite-volume method involves the integration of the mass-conservation equations (11.1) in a grid volume:

$$\frac{\partial}{\partial t}\int_{\Omega_e}\phi\rho_\alpha S_\alpha \,\mathrm{d}V + \oint_{\partial\Omega_e}\rho_\alpha\vec{q}_\alpha\cdot\vec{d}S = \int_{\Omega_e}\rho_\alpha I_\alpha \,\mathrm{d}V, \tag{11.14}$$

where Ω_e is the domain of a cell element. The mass-conservation equation is obviously valid for any cell element in the domain, geometrical or virtual (we call a

https://doi.org/10.1017/9781009019781.017 Published online by Cambridge University Press

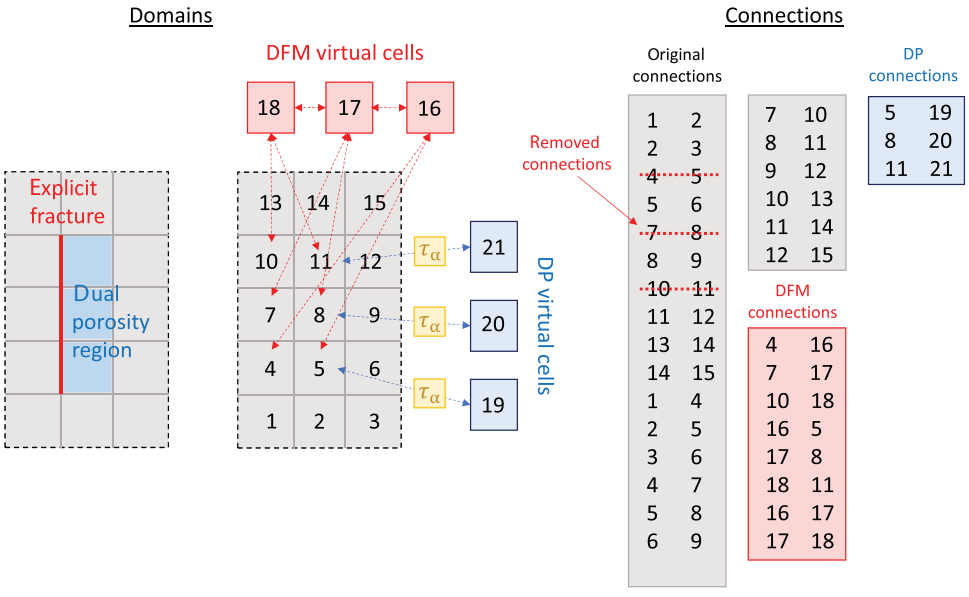

Figure 11.3 Schematic of flowing domain with one explicit fracture and a dual-porosity region. Left figure shows the geometrical domain and the new virtual cells with the new virtual connections. Right figure shows the final connection list, including the original connections, the DFM connections, and the dual-porosity connections. Note that some of the original connections need to be removed due to the explicit fracture that "breaks" the connection between the cells at both sides of the fracture.

geometrical cell a cell that belongs to the geometrical grid). The first term is the mass accumulation term and depends on the pore volume of Ω_e. For geometrical cells and for dual-porosity virtual cells, the pore volume is simply the porosity of the continuum at that point in space times the cell volume. For a DFM virtual cell, we need to use the aperture of the fracture segment to calculate the pore volume (see definition for fracture segment pore volume in Subsection 11.2.3).

The second term of (11.14) accounts for the mass flow rate of phase α across the faces of element Ω_e. Note that these faces can be faces in the geometrical grid (between geometrical cells), virtual faces that connect geometrical and virtual cells, and virtual faces that connect virtual cells. We refer to all of them as connections. Figure 11.3 shows a schematic of the connections for a simple Cartesian grid with a dual-porosity region and one explicit fracture. Again we need to treat the fluxes differently depending on the nature of the connection. For connections between geometrical cells, the flux is the normal flux across the corresponding cell interfaces in the grid. For connection that involve DFM virtual cells, the flux is calculated based on the pressure difference and the fracture–matrix and fracture–fracture

https://doi.org/10.1017/9781009019781.017 Published online by Cambridge University Press

transmissibilities as shown in Subsection 11.2.3. For connections between dual-porosity virtual cells and geometrical cells, the flux is defined by the transfer functions τ_α. Note that transfer functions have units of kilograms per second, and though they are commonly formulated as source terms in equations for fracture and matrix continua, they can be alternatively seen as fluxes across virtual faces that connect geometrical and virtual cells.

The discrete divergence operator in MRST, discussed in subsection 4.4.2 of the MRST textbook [27], is a linear mapping from faces to cells that ensures that mass is conserved across all faces. That is, this operator ensures that the mass we take out from one cell (geometrical or virtual) enters the neighbor cell (geometrical or virtual). Note that this is equivalent to adding the source terms τ_α and $-\tau_\alpha$ to the matrix and fractures in the dual-porosity model.

In short, the implementation of a unified framework that encompasses different methods for fracture representation in the same model requires updating the geometrical grid with new cells (each cell group having different models to compute pore volume depending on the selected fracture representation) and the grid connection list (represented by `G.cells.neighbors` in the grid structure in MRST) with new connections, where each connection has a different model for the phase fluxes. Luckily, MRST offers a code structure that enables a very flexible implementation of the main functions and variables required to implement the conservation equations for flow in fractured porous media.

The important point to note is that the current structure of the AD models of MRST is based on *state functions* and enables us to redefine the phase fluxes and pore volumes for the virtual cells and connections. A state function is an abstraction of a function of one or more state variables that can be evaluated in the domain or part of it. The importance of state functions is that they enable us to separate the logic on how these functions are computed from the logic of the governing equations. State functions have dependencies, which can be state variables such as pressure or saturation or other state functions. The state of the system is updated via a call to the member function `evaluateOnDomain` of the `StateFunction` class. When this function is called for a particular state function, the whole chain of dependencies of this state function is recursively updated via calls of this member function. For instance, the phase pressures are defined via a state function that depends on the state variable pressure and the state function capillary pressure. The latter depends on the state variable saturation. Note that, when the phase pressures are computed, we do not need to know exactly *how* the capillary pressure was computed. This logic belongs to the capillary pressure state-function class.

https://doi.org/10.1017/9781009019781.017 Published online by Cambridge University Press

By detaching the logic of the different state functions, MRST creates a powerful framework for flexible model concept implementation. For a comprehensive presentation of the state functions concept, we refer to Chapter 5.

The following code snippet helps to understand how this concept is applied in our unified framework. It was extracted from the equations file for immiscible two-phase flow in MRST (`equationsOilWater.m`). Building a physical model in MRST involves evaluating state functions on the domain:

```
[b, pv]   = model.getProps(state,  'ShrinkageFactors', 'PoreVolume');
[b0, pv0] = model.getProps(state0, 'ShrinkageFactors', 'PoreVolume');
[phaseFlux, flags] = model.getProps(state, 'PhaseFlux',  'PhaseUpwindFlag');
[pressures, mob, rho] = model.getProps(state, 'PhasePressures', 'Mobility',
                                               'Density');
```

and using these to construct the residual of each conservation equation:

```
[bW, bO]     = deal(b{:});
[bW0, bO0]   = deal(b0{:});
[vW, vO]     = deal(phaseFlux{:});
[upcw, upco] = deal(flags{:});

% Accumulation term for water and oil phase
water = (1/dt).*( pv.*bW.*sW - pv0.*bW0.*sW0 );
oil   = (1/dt).*( pv.*bO.*sO - pv0.*bO0.*sO0 );
eqs   = {water, oil};

% Fluxes across faces (connections)
eqs{1} = eqs{1} + s.Div(s.faceUpstr(upcw, bW).*vW);
eqs{2} = eqs{2} + s.Div(s.faceUpstr(upco, bO).*vO);
```

We can see that the mass conservation equations defined by (11.1) are implemented and the AD residuals are stored in the `water` and `oil` variables. Note that there is no information on *how* the phase fluxes, pore volume, and other state functions are calculated inside the equations file. We simply need to make sure that we redefine these state functions in an external class. Note also that the new `vW` and `vO` variables will now have as many entries as connections in our augmented domain. Therefore, operators like `s.Div` need also to be augmented by the new connections that are introduced.

In order to implement the concepts just described in MRST, we develop a class called `FracturedDomainManager` that acts as a wrapper around the standard AD models of MRST. The class has no attributes and provides a method called `addFracturedDomain(model,...)` that includes the information about the fractured domains in the physical model. Note that we send in a `model` to this method and receive back the same model with extra information. With this

https://doi.org/10.1017/9781009019781.017 Published online by Cambridge University Press

structure, we can easily "stack" several virtual domains, all with different models for the phase flux. For instance, to define a model like the one shown in Figure 11.3, we write:

```
%% Adding DFM domain
model = manager.addFracturedDomain(model,'dfm',edges_dfm,rock_dfm,fluid_dfm);

%% Adding dual-porosity domain
transfer_model = KazemiMultiphaseTransferFunction(...
                        WarrenRootShapeFactor([10,10,10]));
model = manager.addFracturedDomain(model, 'multi_continuum', region_dp,...
                        rock_dp, fluid_dp, 'transfer_model', transfer_model);
```

where `edges_dfm` and `region_dp` are lists of edges and cells of the geometrical grid. Note that we need to inform the method which concept of virtual domain we are adding, because this defines the way we create the connections and the phase fluxes. If the domain is of type `multi_continuum`, we need to provide a transfer model that calculates fluxes between virtual and geometrical cells for each phase. We can have a glimpse at how such transfer models look. The following code snippet is adapted from the method `transfer` of the class `KazemiMultiphaseTransferFunction`:

```
function v = transfer(obj, model, state, domain_id)
    [p, s, flag, mob, rho] = ...  % Get state variables
        model.getProps(state, 'PhasePressures', ...
                    'Saturation', 'PhaseUpwindFlag', 'Mobility', 'Density');
    % Get domain object
    dom = model.G.FracturedDomains.domains{domain_id};
    % Calculate shape factor
    sigma = obj.shape_factor_model.calculate_shape_factor(dom.rock);
    vb = model.G.cells.volumes(dom.region);
    % Compute fluxes (transfer)
    pf = {}; pm = {};        % fracture and matrix pressures
    for i = 1:length(p)
        % Pressures in fracture and matrix
        pf{i} = p{i}(dom.connections(:,1));
        pm{i} = p{i}(dom.connections(:,2));
        % Fluxes
        fmob = model.operators.faceUpstr(flag{i}, mob{i});
        fmob = fmob(dom.global_connection_ids);
        v{i} = vb.*fmob.*sigma.*(pf{i}-pm{i});
    end
end
```

Note how state functions help us to easily compute the pressure gradient and the upwind properties to define the transfer function. The method `getProps` is defined for any subclass of the `PhysicalModel` class. This method enables us to get state

https://doi.org/10.1017/9781009019781.017 Published online by Cambridge University Press

variables and their corresponding AD-computed derivatives, given the string that defines the state variable name. We then get the corresponding domain structure given by the `domain_id` identifier. This step is required, because the framework allows the user to define simulation models having an arbitrary number of different domains with different transfer functions. For this particular transfer function, a shape factor computation is required. The calculation of the shape factor based on the matrix-block geometry is carried out in the `calculate_shape_factor` member function of the `ShapeFactor` class. Note that no particular definition of shape factor model is assumed here, which enables the user to explore different shape factor concepts with any transfer function. Finally, we iterate through the pressure cell arrays and compute the phase fluxes based on the pressure gradients of each fluid phase. Note that the pressures are restricted to the cells that define the virtual connections, as previously defined in the call to the `addFracturedDomain` method.

In order to understand how we compute the phase fluxes for the different types of connections, we can have a look at the `evaluateOnDomain` method of the state function `FracturedDomainPhaseFlux`:

```
function v = evaluateOnDomain(prop, model, state)
    [mob, kgrad] = prop.getEvaluatedDependencies(state, 'FaceMobility',
                                                 'PermeabilityPotentialGradient');
    % Standard flux evaluation
    v = cellfun(@(x,y)-x.*y, mob, kgrad, 'UniformOutput', false);

    % Compute fluxes between multi-continuum domains
    for j = 1:length(model.G.FracturedDomains.domains)
        dom = model.G.FracturedDomains.domains{j};
        if(strcmp(dom.type,'multi_continuum'))
            ids = dom.global_connection_ids;
            vf = dom.transfer_model.transfer(model, state, j);
            for i = 1:numel(mob)
                v{i}(ids) = vf{i};
            end
        end
    end
```

Like any standard state function, `FracturedDomainPhaseFlux` provides a method called `evaluateOnDomain` that implements the logic of evaluation of the state function on the domain given a certain system state. This class is a specialization of the `PhaseFlux` state function that treats the flux between cells differently depending on the cell type. Geometrical connections (i.e., those defined by standard transmissibilities) are calculated as `v = -mob.*kgrad`. This happens in a call to the MATLAB function `cellfun`, which evaluates a function handle on each element of a cell array. Here, the function `@(x,y)-x.*y` is evaluated for all

https://doi.org/10.1017/9781009019781.017 Published online by Cambridge University Press

of the entries of the `mob` and `kgrad` cell arrays, which have as many entries as fluid phases. Hence, after this call, the cell array `v` contains the phase fluxes for the fluid phases. An additional loop is required to compute the fluxes of connections between multicontinuum domains. In this case the phase flux is computed via an arbitrary transfer model defined in run time by the user. The `transfer` method of this transfer model is called to compute the flux for these connections. A similar treatment is made in `FracturedDomainPoreVolume`, in which the cell pore volume is computed differently depending on the cell type. Hence, the definition of these state functions is at the core of the `fractures` module, because it enables us to treat the flow and pore volume associated to any type of connection or cell by changing the appropriate state functions without modifying the functions that define the conservation equations.

This concludes our description of the framework for simulation of flow in fractured reservoirs with different model concepts. At this stage we note that though this chapter focuses on dual-porosity and DFM models, any other model for simulation in fractured reservoirs, such as EDFM or MINC, fits in this framework. We briefly discuss some of these models in the next subsection.

11.3.2 A Brief Note on Other Methods

The framework described in the previous section is general and can be extended to accommodate other models for simulation of fractured reservoirs. In the multicontinuum family, a natural extension of the dual-porosity model is dual-porosity dual-permeability models. These models are useful when the permeability contrast between fractures and matrix is lower and there is significant viscous flow across the matrix continuum. The implementation of a dual-porosity dual-permeability model in the framework would require extending the connection list to include connections between multicontinuum virtual cells and updating the `FracturedDomainPhaseFlux` class (or the transmissibility in `model.operators`) to treat the flux between the matrix cells.

Another subfamily of methods in the multicontinuum approach involves discretizing the matrix blocks to better represent the fluid transfer between fracture and matrix. One popular formulation of such methods is the MINC model, which consists of discretizing the cubic matrix blocks in shells to better describe the imbibition front inside the matrix during waterflooding in fractured reservoirs. With some minor modifications, this method could be easily implemented in the framework: One needs to stack several multicontinuum domains and set their cell lists as the virtual cells instead of the geometrical cells. Then, the shape factors and the pore volume of the virtual domains have to be changed to represent the shell structure.

https://doi.org/10.1017/9781009019781.017 Published online by Cambridge University Press

By default, any new domain considers connection between its cells and the physical region in which it is defined (geometrical cells). It is possible, however, to define connection between virtual domains using the optional argument `cell_connection_list` of `FracturedDomainManager`. This is useful to implement models that consider discretization of matrix blocks, such as MINC and subdomain methods.

The implementation of other methods that represent the fractures explicitly is also straightforward in the framework presented in this chapter. Generally speaking, the implementation of EDFM requires the computation of intersections between fracture cells and between fracture and matrix cells. These intersections define the connection lists that are implemented in the class `FracturedDomainManager`. Additionally, EDFM approximates the transmissibility between these cells, just like we define transmissibilities between fracture and matrix connections in DFM (see Figure 11.2). Hence, the implementation of EDFM in the `fractures` module would follow very closely the structure of the DFM code.

11.3.3 Description of the `fractures` *Module*

The `fractures` module is a standard MRST module with the following directory structure:

```
fractures
├── examples
│   ├── dfm-and-dual-porosity
│   ├── multirate-transfer
│   ├── pressure-buildup
│   └── validation-dfm
├── manager
│   ├── state-functions
│   └── FracturedDomainManager.m
├── transfer-functions
│   ├── shape-factors
│   ├── KazemiMultiphaseTransferFunction.m
│   └── SaturationDifferenceTransferFunction.m
└── utils
```

The `examples` folder contains the scripts to run the simulations that will be described in Section 11.4. The `utils` folder contains helper functions that are used

https://doi.org/10.1017/9781009019781.017 Published online by Cambridge University Press

throughout the module's code. The `manager` folder has a subfolder containing the state functions that are defined for the fractured domains and the script `FracturedDomainManager.m`, which contains a class of the same name and is considered the "brain" of the `fractures` module. In this class we implement the logic of domain superposition that is described in Section 11.3 of this chapter. This class is able to take a general physical model as an input and superpose multiple domains to the geometrical grid by defining the method `addFracturedDomain`.

The `transfer-functions` folder contains a subfolder with the shape factors from Table 11.1 and two classes implementing dual-porosity transfer functions that cover most of the models presented in Section 11.2: the ones that are based on a pressure potential with a gravity correction term (11.5) and the ones that are based on a saturation difference scaled by a transfer-rate coefficient (11.9).

At the present stage, the code only implements the DFM method and dual-porosity models with the aforementioned transfer functions and shape factors. However, as explained in Subsection 11.3.2, the implementation of other approaches for flow simulation in fractured reservoirs can be incorporated with little effort in the `fractures` module.

11.4 Applications

In this section, we present example cases that outline the main features of the `fractures` module. The examples aim to show the reader how to setup simulation models to simulate fluid flow in fractured reservoirs. Each of the following subsection corresponds to one folder inside the `examples` folder in the accompanying code. Table 11.2 shows a brief description of each example case and the reference to the corresponding folders.

Table 11.2 *Description of the examples available in the* `fractures` *module. Folders are placed in the* `fractures/examples` *folder.*

Subsection	Folder name	Description
11.4.1	`validation-dfm`	Validation of the DFM implementation against [21]
11.4.2	`pressure-buildup`	Analysis of pressure buildup during fluid injection in a fractured aquifer
11.4.3	`dfm-and-dual-porosity`	Simulation model with combined DFM and dual-porosity regions
11.4.4	`multirate-transfer`	A multirate-transfer dual-porosity simulation example

https://doi.org/10.1017/9781009019781.017 Published online by Cambridge University Press

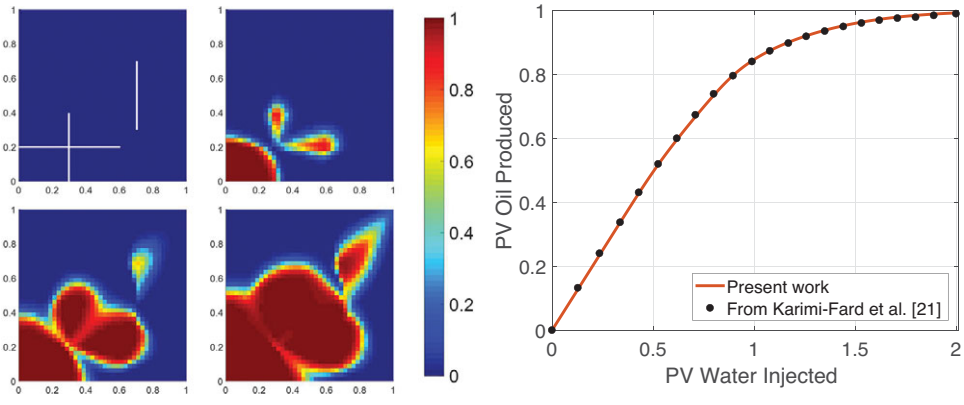

Figure 11.4 Left: Comparison of water saturation profiles after 0.0, 0.1, 0.3, and 0.5 pore volumes injected obtained with the present implementation in MRST. Right: Cumulative oil produced against pore volume water injected yield with present work compared to data extracted from [21].

11.4.1 Validation of the DFM Implementation

We validate the DFM implementation using a 2D model introduced in [21]. This model consists of a 1 m $\times$ 1 m domain with three fractures and one intersection (see Figure 11.4). The permeability and the porosity of the matrix are $k_m = 1$ md and $\phi_m = 0.2$, respectively. The fracture aperture is set to $a_f = 0.1$ mm, yielding a fracture permeability of $k_f = a_f^2/12 = 844.37$ D. The matrix and fractures are initially fully saturated with oil. The viscosities of water and oil are $\mu_w = 1$ cP and $\mu_o = 0.45$ cP. Water is injected in the lower-left corner with an injection rate of $q = 0.01$ pore volume per day. The producer well is located in the opposite, upper-right corner and is operated at a fixed pressure of 0 Pa. Linear relative permeabilities and no capillary pressure are assumed for both the fractures and the matrix. To set up this model, we first identify the fracture edges and then apply the `addFracturedDomain` method with domain type `'dfm'`:

```
%% Defining fractures
[xf, yf]    = deal( G.faces.centroids(:,1),G.faces.centroids(:,2));
frac1       = find(abs(yf-0.2) <= 1e-5 & xf <= 0.6);
frac2       = find(abs(xf-0.3) <= 1e-5 & yf <= 0.4);
frac3       = find(abs(xf-0.7) <= 1e-5 & yf <= 0.7 & yf >= 0.3);
frac_edges = [frac1; frac2; frac3];

%% Fractured domain manager
manager = FracturedDomainManager();
model = manager.addFracturedDomain(model, 'dfm', frac_edges,...
                                   rock_fracture, fluid_fracture);
```

https://doi.org/10.1017/9781009019781.017 Published online by Cambridge University Press

We discretized the model using a Cartesian grid with dimensions of 40×40 grid cells. This approximately corresponds to the spatial resolution of the fine grid used in [21]. We ran the simulation for a total time of 200 days using a timestep of $dt = 1$ day. Figure 11.4 shows the water saturation profiles after 0.0, 0.1, 0.3, and 0.5 pore volumes injected. By analyzing the cumulative oil production, shown in Figure 11.4, we see that we obtain a nearly identical result compared to the reference solution taken from [21]. This validation benchmark gives us confidence that our implementation in MRST is consistent and can be applied for simulation of flow through explicitly represented fractures using the DFM method.

11.4.2 Pressure Buildup in Fractured Aquifers during CO_2 Storage Operations

One of the key points of attention during CO_2 storage operations is the control of the pressure buildup during the lifetime of the project. As CO_2 is injected in an underground formation, the imbalance between mass increase and pressure diffusion might lead to overpressurization of the formation, which could in turn lead to a series of serious operational problems, such as fault reactivation and fracturing of the caprock. During the design of storage operations, the injection rate must be designed in such a way that the maximum pressure in the caprock never exceeds the caprock's fracture pressure. Numerical simulations (and analytical solutions) may be used to achieve this goal.

Whereas fractures are deemed to pose a challenge to storage operations due to the fast-traveling CO_2 plume, the presence of a fracture system in an aquifer might boost the injection rates due to the increased permeability provided by the fracture system. In this exercise, we use MRST and the dual-porosity model to evaluate the gain in injectivity provided by fractures and obtain a better insight on the key controls on CO_2 injection rates in fractured and unfractured formations. This example is inspired by the work of [46], in which the authors use similar numerical and analytical models to evaluate the storage potential of large saline aquifers.

Model Setup. We consider an aquifer with size $10\,\text{km} \times 10\,\text{km} \times 100$ m, at a depth of 2 km, with a pressure of $P_{top} = 20$ MPa at the top of the formation. We assume that rock and fluid properties are constant and uniform throughout the model domain. We discretize the domain using $50 \times 50 \times 10$ grid blocks. For simplicity, we consider a single-phase model with a well at the center of the aquifer that operates at a constant injection rate for a maximum injection time of 30 years. The aquifer is initially saturated with water having constant density

https://doi.org/10.1017/9781009019781.017 Published online by Cambridge University Press

of $\rho = 10^3$ kg/m^3, constant viscosity of $\mu = 1$ cP, and compressibility of $c = 0.1$ GPa^{-1}. The pressure is initially hydrostatic and we assume the aquifer is confined (no-flow boundary conditions).

Note that in this example we disregard two-phase flow effects that might be important for the pressure evolution in the aquifer. However, considering a single-phase model for the pressure buildup typically leads to a conservative estimate, because CO_2 has a compressibility that is orders of magnitude higher than that of brine. Similarly, whereas most of the target formations for storage projects are not confined aquifers, assuming a confined formation leads to higher and faster pressure buildup and hence to a conservative design.

We compare the pressure buildup in two cases: one fractured and one unfractured version of the aquifer. The unfractured aquifer is modeled as a conventional single-porosity model, whereas its fractured version is modeled as a dual-porosity model using the `multi_continuum` domain type of the `FracturedDomain Manager`. Both models have the same total pore volume and the same matrix permeability ($k_m = 10$ md). For the unfractured model, we assume a matrix porosity of $\phi_m = 0.2$, and for the fractured model we assume a matrix porosity of $\phi_m = 0.19$ and a fracture porosity of $\phi_f = 0.01$. We consider three permeability contrasts between fracture and matrix for the fractured model: $k_f = 10k_m$, $k_f = 10^2 k_m$, and $k_f = 10^3 k_m$. We assume the standard Kazemi et al. [24] transfer function (11.4) and shape factor (see Table 11.1) for the matrix–fracture transfer, with a constant fracture spacing of $L_x = L_y = 100$ m and $L_z = 10$ m. We consider a fracture pressure of $P_{frac} = 30$ MPa for the caprock for all cases. The setup of the fractured continuum in this case reads:

```
%% Fractured domain manager
manager = FracturedDomainManager();
t_model = KazemiMultiphaseTransferFunction( KazemiShapeFactor([100,100,10]));
region  = 1:G.cells.num;
model   = manager.addFracturedDomain(model, 'multi_continuum', region,...
                                     rock_matrix, fluid_matrix, t_model);
```

Results. We first analyze the pressure buildup for the fractured and unfractured models for a constant injection rate of $q_i = 1.6 \times 10^{-2}$ MtCO$_2$/year. Figure 11.5 shows the pressure buildup in the aquifer for the unfractured model and the fractured model with $k_f = 10k_m$ at $t = 10$ years and $t = 30$ years. We note that this injection rate leads to a pressure buildup in the well regions that reaches the fracture pressure of the caprock in the unfractured model. For the fractured model, we see that the pressure diffuses throughout the domain more quickly and this alleviates the pressure increase in the well region. Note that at the end of the

https://doi.org/10.1017/9781009019781.017 Published online by Cambridge University Press

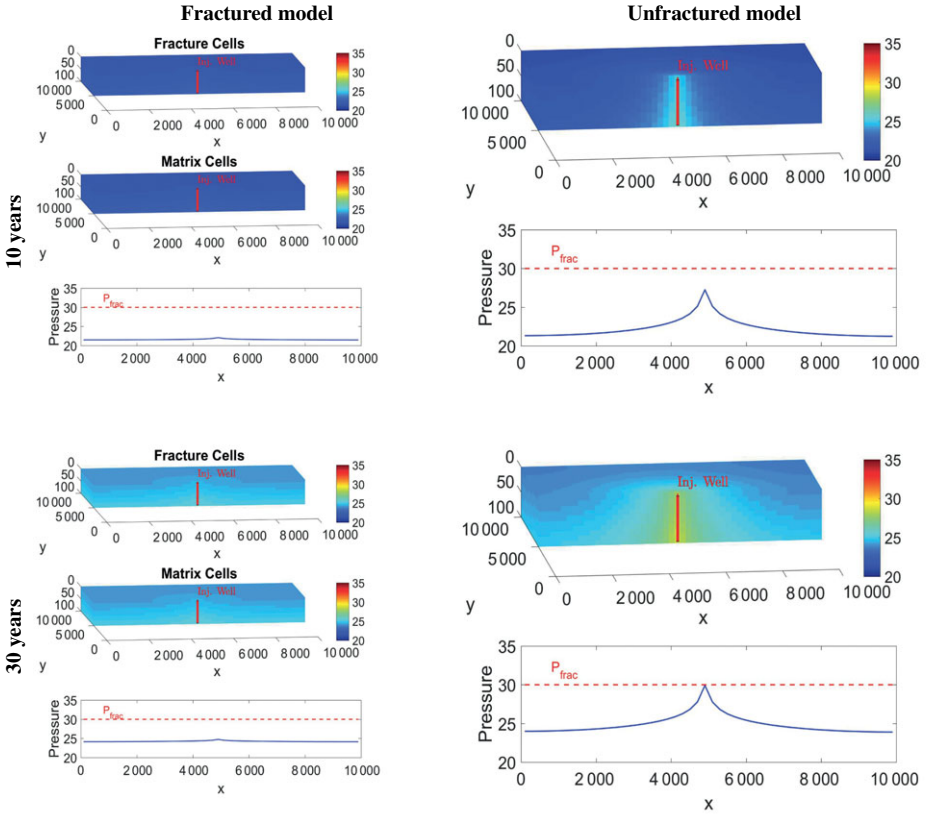

Figure 11.5 Pressure buildup in fractured model (left) and unfractured model (right) after 10 years and after 30 years of fluid injection. The fluid pressure increases uniformly in both matrix and fracture systems in the fractured model but stays below the fracture pressure of the caprock. In the unfractured model, the fluid pressure builds up at the well region such that fracture pressure of the caprock is reached after 30 years of injection.

injection period, after 30 years, the pressure in the fractured model is still much lower than the fracture pressure in the caprock. We also notice that the pressure fields in fracture and matrix look quite similar in the fractured model. Indeed, due to the low compressibility of the resident fluid, the pressure fields in the fracture system and matrix equilibrate at a relatively fast timescale. For formations with lower permeability, we can expect a larger transient pressure diffusion time. This setting usually requires more complex transfer functions, because it is well known that the standard Kazemi et al. [24] transfer functions underestimate the early-time transfer, an effect that might lead to a high discrepancy when the transient diffusion period is long.

https://doi.org/10.1017/9781009019781.017 Published online by Cambridge University Press

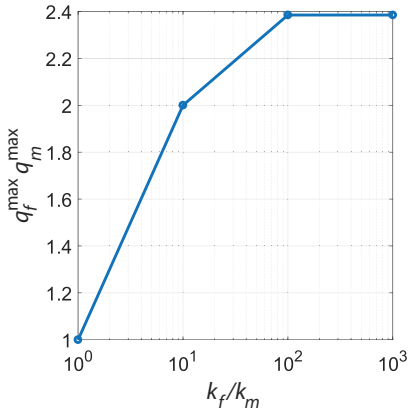

Figure 11.6 Scaling the increase in maximum injection rate with increase in permeability contrast between fracture an matrix. First point (leftmost) is the unfractured simulation case. The other three data points correspond to the cases with $k_f/k_m = 10$, $k_f/k_m = 100$, and $k_f/k_m = 1\,000$.

Figure 11.6 shows the maximum injection rates for the unfractured and fractured cases with $k_f = 10k_m$, $k_f = 10^2k_m$, $k_f = 10^3k_m$. We observe a factor two increase when comparing the maximum injection rates in the unfractured and fractured models with $k_f = 10k_m$. In principle, one could think that a further increase in the fracture permeability would lead to a significant further increase in the maximum injection rate. This does not happen for this model as evidenced by Figure 11.6.

In fact, when we increase the permeability contrast to a factor of $k_f/k_m = 10^2$, we observe an increase in the maximum injection rate of only factor $\approx$2.4, and the cases with $k_f = 10^2k_m$ and $k_f = 10^3k_m$ show essentially the same maximum injection rate. This is not surprising, as seen in the pressure fields in Figure 11.5, because the pressure profiles in the fractured model with $k_f = 10^2k_m$ are already uniform throughout the domain. In other words, there is no further room for improvement in pressure diffusivity for this particular choice of rock and fluid properties. This application example – though perhaps simplistic – shows that fractures often provide an injectivity boost and allow a pressure relief in injection operations, which may be advantageous to prevent the fracturing of the caprock.

11.4.3 A Model with Explicit Fractures and Dual Porosity

In this section, we present an example that shows how to combine DFM and dual-porosity concepts in the same model. The model we create is inspired by the fault model described by [20] and considered in Karimi-Fard et al. [21] for flow simulation using DFM. A fault typically consists of a low-permeability fault core and a system of fractures surrounding the fault core, called the fault damage zone.

https://doi.org/10.1017/9781009019781.017 Published online by Cambridge University Press

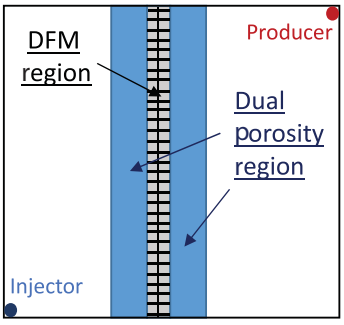

Figure 11.7 Schematic of the virtual domains for the present simulation model. Blue rectangles show the dual-porosity region, which are separated by a region with explicitly defined fractures using the DFM model.

The fault core is often an extensively fractured region, and a fault may be sealing or not sealing depending on the connectivity of this fracture network. A variety of possible scenarios arise due to the uncertainty about the connectivity of fracture systems in the subsurface. The evaluation of different possible scenarios is of utmost importance to understand fluid flow in a reservoir that has the presence of a large-scale fault.

Model Setup. In our illustrative model, we consider a square 2D domain of size 1 000 m × 1 000 m, with a fault crossing the entire domain in the middle of the x axis (see Figure 11.7). We assume that the fault core ($x = 500$ m to $x = 560$ m) has no matrix permeability ($k_m = 0$) but has a system of conductive fractures that establish a hydraulic connection between the left and right parts of the domain. We consider all grid edges located in the fault core region as fracture edges with an aperture $a_f = 0.1$ mm, which corresponds to a permeability of $k_f \approx 844$ D. The domains are initially saturated with the nonwetting phase. We inject the wetting phase via a well that operates at constant injection rate of $q_i = 10^{-4}$ m^3/s and is placed at the bottom-left corner of the flowing domain. Fluids are produced via a well that operates at constant bottom-hole pressure of 500 psia and is placed at the top-right corner of the flowing domain.

In fractured reservoir simulation, we often assume that the fracture permeability is related to its aperture via the so-called cubic law: $k_f = a_f^2/12$. Although this is a useful concept to initialize the permeability of a single fracture, it assumes that the fractures are flat surfaces, a somewhat unrealistic assumption. However, we can interpret this aperture as a hydraulic aperture, equivalent to an aperture of a flat-surface fracture that matches the permeability of a fracture with a rough surface.

https://doi.org/10.1017/9781009019781.017 Published online by Cambridge University Press

Around the fault core, we consider two regions with a system of natural fractures and nonzero permeability in the matrix that will be modeled with the dual-continua approach. This is the fault damage zone. The left dual-porosity region is located between $x = 400$ and $x = 500$ and the right region is located between $x = 560$ and $x = 650$. For these regions, we assume an upscaled fracture permeability of $k_f = 100$ md. The matrix of the dual-porosity regions has the same properties of the flowing domain outside the fault zone, $k_m = 1$ md.

Fluid transfer in the dual-porosity region is driven by the capillary contrast between fracture and matrix. We assume a simple Corey-type model for the capillary pressure of the matrix in this zone: $p_{cow}(S_w) = p_o - p_w = P_e S_w^{-0.5}$, with an entry pressure $P_e = 100$ kPa. We consider the Kazemi et al. [24] transfer function with a constant and uniform fracture spacing of $L_x = L_y = L_z = 10$ m. The setup of the configuration is as follows. We first define the dual-porosity region cells and the edges that define the DFM fractures:

```
%% Dual-porosity cells, fault core cells and DFM edges
xc = G.cells.centroids(:,1);
xf = G.faces.centroids(:,1);
yf = G.faces.centroids(:,2));
dual_porosity_cells = find(((xc >= 400) & (xc <= 500)) | ...
                           ((xc >= 560) & (xc <= 650)));
fault_core_cells = find(xc>=500 & xc<=550);
dfm_edges = find((xf >= 490) & (xf <= 570) & (yf < y_size) & (yf > 0));
```

We then instantiate the `FracturedDomainManager` class and add the two fractured domains to the model:

```
%% Fractured domain manager and transfer model
manager = FracturedDomainManager();
t_model = KazemiMultiphaseTransferFunction(KazemiShapeFactor([10, 10, 10]));

% Dual-porosity domain
model = manager.addFracturedDomain(model, 'multi_continuum',...
                            dual_porosity_cells, dual_porosity_matrix_rock,...
                            dual_porosity_matrix_fluid,...
                            'transfer_model', t_model);

% DFM domain
model = manager.addFracturedDomain(model, 'DFM', dfm_edges,...
                                   dfm_fracture_rock, dfm_fracture_fluid);
```

where `dual_porosity_matrix_rock` and `dual_porosity_matrix_fluid` are the rock and fluid models for the dual-porosity region and `dfm_fracture_rock` and `dfm_fracture_fluid` are the rock and fluid models for the DFM region.

https://doi.org/10.1017/9781009019781.017 Published online by Cambridge University Press

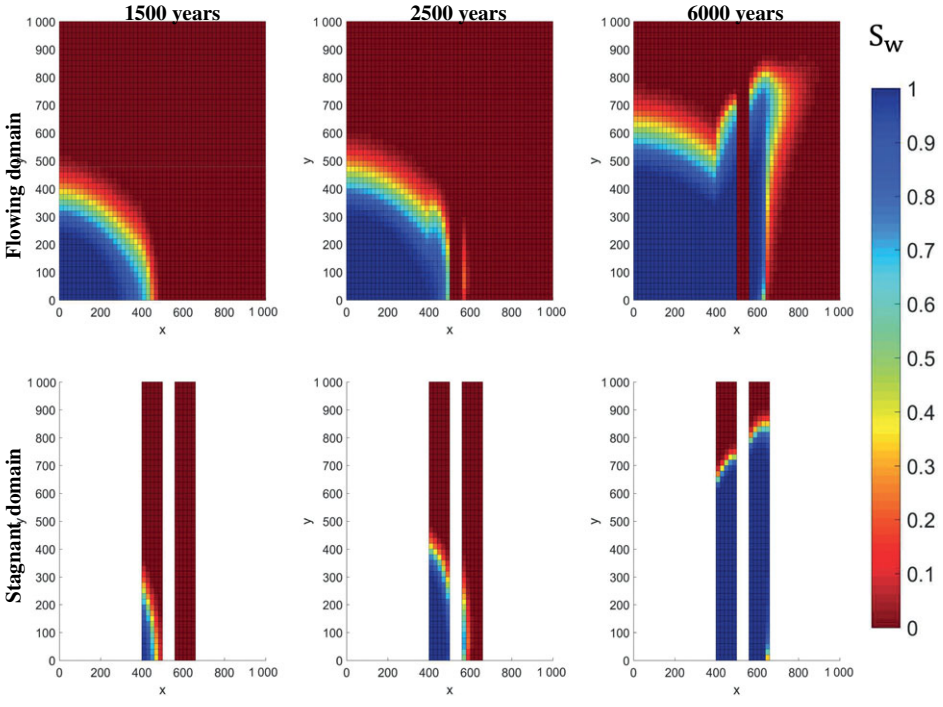

Figure 11.8 Water saturation profiles after 1 500 days, 2 500 days, and 6 000 days of simulation. Top row shows the water saturation at the flowing domain, which consists of the matrix of the single-porosity region, the fractures of the dual-porosity region, and the DFM fractures. Bottom row shows the water saturation at the stagnant domain, which consists of the matrix of the dual-porosity region.

Results. Figure 11.8 shows the saturation fields after 1 500, 2 500, and 6 000 days of injection. We observe the water saturation spreading radially from the injection well after 1 500 days. Note that at this stage the water front has just reached the dual-porosity zone, and we observe some water in the matrix of the left part of the fault damage zone. After 2 500 days, the water has crossed the fault core via the high-conductivity fractures and has reached the right half of the domain. In this region, we also observe transfer to the matrix. Finally, after 6 000 days, we see the water starting to break through in the producer well. The high permeability of the damage zone creates an elongated water saturation profile in the dual-porosity region. We note that, whereas it is not possible to see any water saturating the fault core region due to the zero permeability of the matrix, the high-conductivity fractures located on the edges of the geometrical mesh in this part of the domain successfully establish hydraulic connection between the left and the right part of the model.

https://doi.org/10.1017/9781009019781.017 Published online by Cambridge University Press

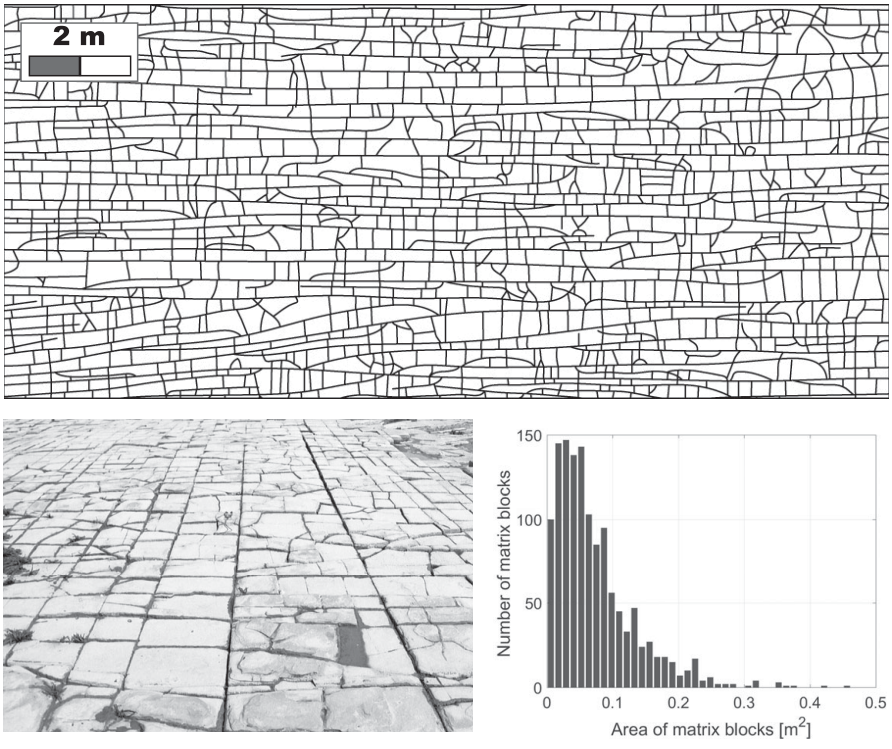

Figure 11.9 Fracture network mapped in Bristol Channel and the corresponding matrix-block size distribution. The histogram shows the heterogeneity of the fracture network in terms of matrix-block sizes. The transfer from the bigger matrix-blocks will occur at a slower rate compared to the fluid transfer from smaller matrix blocks.

11.4.4 Multirate Transfer in Multicontinuum Model

The fundamental assumption of dual-porosity models is a uniform, well-connected fracture network with a single set of matrix properties per grid cell. This was named the *sugar cube model* in the paper that first introduced dual-porosity models [49]. This assumption leads to one transfer-rate value for each grid cell. However, geological outcrops show that in reality the properties of a fracture network and its host rock are far from being uniform (see Figure 11.9). At the scale of a simulation grid cell, one can find heterogeneity in several properties, like the matrix-block sizes, matrix permeability, and matrix porosity. To capture this heterogeneity, one can use the multirate-transfer model that makes use of a distribution of transfer rates arising from sub-grid-scale heterogeneity [14, 32]. In this example, we show how we can use our unified framework to set up multirate-transfer models.

https://doi.org/10.1017/9781009019781.017 Published online by Cambridge University Press

Model Setup. We consider a 2D model with dimensions of 1 km × 1 km discretised with 400 grid cells (each cell with size of 50 m × 50 m). For the flowing domain we set an effective permeability of $k_f = 1$D and an effective porosity of $\phi_f = 0.01$. The permeability and porosity for the stagnant domain (the matrix domain) are $k_m = 0.1$ md and $\phi_m = 0.1$, respectively. Taking the *Klive* fracture network from the Bristol Channel as inspiration, we assume that on the subgrid scale we can distinguish between five subdomains Ω_i with five distinct transfer rates β_i (see Figure 11.10). In this illustrative example, the majority of the stagnant domain volume will exchange fluid with the flowing domain at rates higher than 10^{-9} s^{-1}, whereas the remaining resident fluid will be recovered at lower rates. Note that here we select our transfer rates and the subdomain volume fractions in an arbitrary way. In reality these two properties will be informed by statistical data of the fracture network – e.g., matrix-block size distribution (see Figure 11.9) – and other rock matrix properties like permeability, porosity, and capillary pressure.

We can use a cell array to store the transfer model objects with different transfer rates in order to set up the multirate-transfer model. We add five fractured domains using the `FracturedDomainManager` class as follows:

```
%% Setting pore volume multiplier to account for subdomain volume fractions
sub_domains_fraction = [0.2; 0.35; 0.3; 0.1; 0.05];
for i=1:length(sub_domains_fraction)
    mrtr_matrix_fluid{i}.pvMultR = @(p) p*0 + sub_domains_fraction(i);
end

%% Fractured domain manager
manager = FracturedDomainManager();

%% Transfer models
t_rates = [1e-8, 1e-9, 1e-10, 1e-11, 1e-12];
t_model = cell(length(sub_domains_fraction),1);
for i = 1:length(t_rates)
  t_model{i} = SaturationDifferenceTransferFunction(t_rates(i));
end

for i=1:length(sub_domains_fraction)
    model = manager.addFracturedDomain(model, 'multi_continuum', ...
                    mrtr_cells, mrtr_matrix_rock, mrtr_matrix_fluid{i}, ...
                    'transfer_model', t_model{i});
end
```

Note that to account for the correct pore volume of each subdomain, we are using the `pvMultR` function handle of the fluid structure. This automatically scales the pore volume of each subdomain to its correct size. Because we are setting the pore volume multipliers to constant values, we need to define the function handle as `p*0+K`, where K is the subdomain fraction. This is necessary because the `pvMultR`

https://doi.org/10.1017/9781009019781.017 Published online by Cambridge University Press

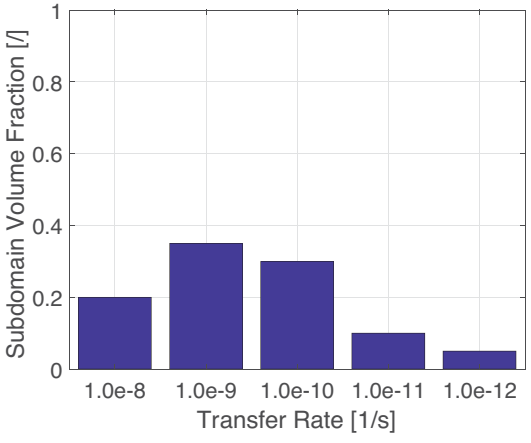

Figure 11.10 Partition of the stagnant domain Ω_m into five subdomains Ω_i with five different rates β_{Ω_i} associated with them.

field was originally designed to be a pressure-dependent pore volume multiplier. We use a "saturation difference" transfer function [11, 29] that computes the flux for the virtual connection depending on the difference between the saturation value in the matrix and the maximum possible water saturation in the matrix (see (11.9)).

We model a two-phase system with water and oil. The viscosities of both phases are $\mu_w = 1$cP and $\mu_o = 1.5$cP, respectively. We assume linear relative permeabilities for all domains. We place an injector in the lower-left corner and a producer in the upper-right corner connected to the flowing domain. The system is initially 100% saturated by oil and we inject water at a fixed rate of $q_i = 10^{-4}$ m^3/s. The production well operates at a constant bottom-hole pressure of 10^3 psia. No-flow boundary conditions are assumed at the boundaries of the domain.

Results. We run the simulation with a timestep of $dt = 0.5$ year for a total time of 50 years. For comparison, we also run five other simulations, each with the standard dual-porosity model using a constant transfer rate. The pore volume of each individual dual-porosity simulation is the same as the sum of the pore volumes of the matrix domains in the multirate-transfer simulation. Figure 11.11 shows the results of the simulations in terms of oil recovery factor and the water cut. Comparing the single-rate dual-porosity models, we clearly see that lower transfer rates result in a slower and thus lower total recovery of oil from the rock matrix. Because less water is imbibing into the rock matrix and more water remains in the fractured continuum, we see an earlier water breakthrough at the producer for these models. Accordingly, the cases with higher transfer rates show later water breakthrough times and higher oil recoveries. Using the multirate-transfer model

https://doi.org/10.1017/9781009019781.017 Published online by Cambridge University Press

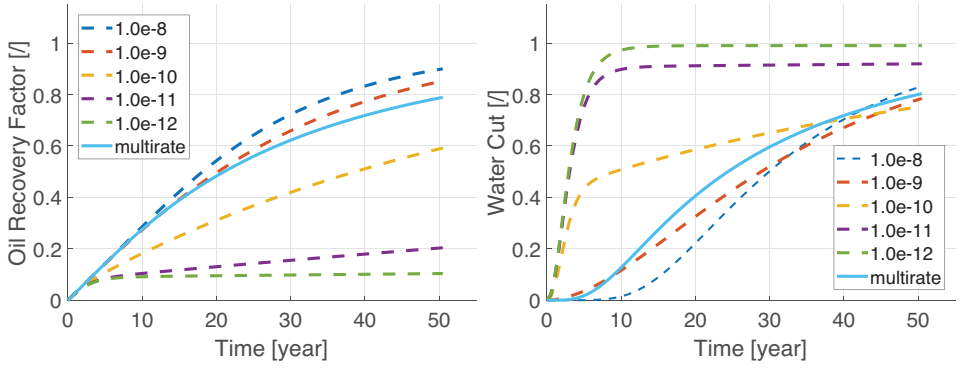

Figure 11.11 Oil recovery factor and water cut for different rates compared with the multirate case.

that accounts for all transfer rates proportional to the matrix volume, we see that the oil recovery curve follows the curves of the higher rates at early time but deviates from it at late times. At the time of deviation, the recovery of oil from the matrix volume associated with higher rates is nearly completed, whereas the subdomains with lower rates are still mainly saturated by oil. The overall recovery of oil is therefore delayed and the water breakthrough expedited.

11.5 Summary and Conclusion

Simulation of multiphase fluid flow in fractured porous media is relevant to a variety of subsurface applications ranging from geo-energy to water resources. Fractures are ubiquitous in the subsurface and it is often important to represent them in simulation models, because they may significantly affect fluid flow. However, simulation of flow in fractured reservoirs still poses a challenge due to the different spatial and temporal scales of fluid flow introduced by fractures.

There are different techniques for simulating flow in fractured reservoirs. These techniques are typically separated into two families of methods: those that explicitly represent the fractures and those that consider fractures as a different continuum. Pertaining to the first family is the discrete fracture and matrix model and to the second family is the dual-porosity model. The first model is typically used when: (i) there is a need to represent fractures explicitly in the computational model and (ii) we can afford to conform the computational grid to the fracture network. The second model is typically used for larger reservoir models, where the explicit representation of fractures would be unfeasible or uninteresting because the exact location of each fracture is not known.

https://doi.org/10.1017/9781009019781.017 Published online by Cambridge University Press

Particularly for dual-porosity models, there is a wide range of models that upscale the mass transfer between the fracture and matrix continua. Typical models used in commercial reservoir simulators sometimes fail to describe the physics of fluid transfer at certain timescales, and many alternative formulations have been suggested in the literature to overcome such downsides. These formulations can be significantly different, and it is important to develop a computational simulator that is not necessarily tailored to one particular transfer model.

In this chapter, we presented a unified framework for simulation of fluid flow in fractured reservoirs using MRST. The framework makes no distinction between the different conceptual models for simulation of flow in fractured reservoirs. It treats different regions of the geometrical grid (e.g., edges or cells) as locations for virtual domains, which could be dual-continua or explicit fractures. The different domains (the geometrical plus one or more virtual) interact with each other via the definition of virtual connections and models that define the phase fluxes. We leverage the newly introduced framework of state functions in MRST to provide a flexible computational toolbox that enables users to implement their own models.

Through a series of application examples we demonstrated how you can use the presented code to solve a variety of problems relevant to geo-energy applications. Though most of the examples consider DFM and dual-porosity models, the presented framework could be easily extended to include most fractured reservoir simulation concepts introduced in the past decades (e.g., MINC and EDFM models).

Acknowledgement. Rafael March and Florian Doster thank the European Commission, the Research Council of Norway, the Rijksdienst voor Ondernemend Nederland, the Bundesministerium für Wirtschaft und Energie, and the Department for Business, Energy & Industrial Strategy, UK, for funding through the ERANET Cofund ACT (Project No. 271497). Rafael March, Florian Doster, and Sebastian Geiger thank the U.S. Department of Energy (DOE) National Energy Technology Laboratory (NETL), which funded Rafael's PhD work under grant FE0023323. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the U.S. Government or any agency thereof. The views and opinions of the authors expressed herein do not necessarily state or reflect those of the U.S. Government or any agency thereof. Christine Maier and Sebastian Geiger thank Energi Simulation for supporting Christine's Research Associate position and Sebastian's Chair for Carbonate Reservoir Simulation at Heriot–Watt University.

References

[1] A. S. Abushaikha and O. R. Gosselin. Matrix-fracture transfer function in dual-media flow simulation: Review, comparison and validation. In *Europec/EAGE Conference and Exhibition*. Society of Petroleum Engineers, 2008. doi: 10.2118/113890-MS.

[2] S. Agada, S. Geiger, and F. Doster. Wettability, hysteresis and fracture–matrix interaction during CO_2 EOR and storage in fractured carbonate reservoirs. *International Journal of Greenhouse Gas Control*, 46:57–75, 2016. doi: 10.1016/j.ijggc.2015.12.035.

[3] J. S. Aronofsky, L. Masse, and S. G. Natanson. A model for the mechanism of oil recovery from the porous matrix due to water invasion in fractured reservoirs. *Petroleum Transactions, AIME*, 213(2):17–19, 1958. doi: 10.2118/932-G.

[4] G. I. Barenblatt, I. P. Zheltov, and I. N. Kochina. Basic concepts in the theory of seepage of homogeneous liquids in fissured rocks. *Journal of Applied Mathematics and Mechanics*, 24:1286–1303, 1960.

[5] I. Berre, F. Doster, and E. Keilegavlen. Flow in fractured porous media: a review of conceptual models and discretization approaches. *Transport in Porous Media*, 130:215–236, 2019. doi: 10.1007/s11242-018-1171-6.

[6] K. Breede, K. Dzebisashvili, X. Liu, and G. Falcone. A systematic review of enhanced (or engineered) geothermal systems: past, present and future. *Geothermal Energy*, 1(1):1–27, 2013. doi: 10.1186/2195-9706-1-4.

[7] J. R. Christensen, E. H. Stenby, and A. Skauge. Review of WAG field experience. In *International Petroleum Conference and Exhibition of Mexico*. Society of Petroleum Engineers, 1998. doi: 10.2118/39883-MS.

[8] K. H. Coats. Implicit compositional simulation of single-porosity and dual-porosity reservoirs. In *SPE Symposium on Reservoir Simulation*. Society of Petroleum Engineers, 1989. doi: 10.2118/18427-MS.

[9] Computer Modelling Group. *IMEX User Guide*. 2016.

[10] L. Denoyelle, C. Bardon, and E. Couve de Murville. Interpretation of a CO_2/N_2 injection field test in a moderately fractured carbonate reservoir. *SPE Reservoir Engineering*, 3(1):220–226, 1988. doi: 10.2118/14942-PA.

[11] G. Di Donato, H. Lu, Z. Tavassoli, and M. J. Blunt. Multirate-transfer dual-porosity modeling of gravity drainage and imbibition. *SPE Journal*, 12(1):77–88, 2007. doi: 10.2118/93144-PA.

[12] L. J. Durlofsky. A triangle based mixed finite element–finite volume technique for modeling two phase flow through porous media. *Journal of Computational Physics*, 105(2):252–266, 1993. doi: 10.1006/jcph.1993.1072.

[13] H. Fossen. *Structural Geology*, 2nd ed. Cambridge University Press, Cambridge, UK, 2016.

[14] S. Geiger, M. Dentz, and I. Neuweiler. A novel multi-rate dual-porosity model for improved simulation of fractured and multiporosity reservoirs. *SPE Journal*, 18(4):670–684, 2013. doi: 10.2118/148130-PA.

[15] S. Geiger, S. Matthäi, J. Niessner, and R. Helmig. Black-oil simulations for three-component, three-phase flow in fractured porous media. *SPE Journal*, 14(2), 338–354, 2009. doi: 10.2118/107485-PA.

[16] R. Haggerty and S. M. Gorelick. Multiple-rate mass transfer for modeling diffusion and surface reactions in media with pore-scale heterogeneity. *Water Resources Research*, 31(10):2383–2400, 1995. doi: 10.1029/95WR10583.

[17] S. Held, A. Genter, T. Kohl, T. Kölbel, J. Sausse, and M. Schoenball. Economic evaluation of geothermal reservoir performance through modeling the complexity of the operating EGS in Soultz-sous-Forêts. *Geothermics*, 51:270–280, 2014. doi: 10.1016/j.geothermics.2014.01.016.

[18] R. Huber and R. Helmig. Multiphase flow in heterogeneous porous media: a classical finite element method versus an implicit pressure–explicit saturation-based mixed finite element–finite volume approach. *International Journal for Numerical Methods in Fluids*, 29(8):899–920, 1999. doi: 10.1002/(SICI)1097-0363(19990430)29:8<899::AID-FLD715>3.0.CO;2-W.

[19] M.-H. R. Hui, M. Karimi-Fard, B. Mallison, and L. J. Durlofsky. A general modeling framework for simulating complex recovery processes in fractured reservoirs at different resolutions. *SPE Journal*, 23(2):598–613, 2018. doi: 10.2118/182621-PA.

[20] H. Jourde, E. A. Flodin, A. Aydin, L. J. Durlofsky, and X.-H. Wen. Computing permeability of fault zones in Eolian sandstone from outcrop measurements. *AAPG Bulletin*, 86(7):1187–1200, 2002. doi: 10.1306/61EEDC4C-173E-11D7-8645000102C1865D.

[21] M. Karimi-Fard, L. J. Durlofsky, and K. Aziz. An efficient discrete-fracture model applicable for general-purpose reservoir simulators. *SPE Journal*, 9(2):227–236, 2004. doi: 10.2118/88812-PA.

[22] M. Karimi-Fard, B. Gong, and L. J. Durlofsky. Generation of coarse-scale continuum flow models from detailed fracture characterizations. *Water Resources Research*, 42(10), 2006. doi: 10.1029/2006WR005015.

[23] H. Kazemi, J. Gilman, and A. Elsharkawy. Analytical and numerical solution of oil recovery from fractured reservoirs with empirical transfer functions (includes associated papers 25528 and 25818). *SPE Reservoir Engineering*, 7(2):219–227, 1992. doi: 10.2118/19849-PA.

[24] H. Kazemi, L. S. Merrill Jr., K. L. Porterfield, and P. R. Zeman. Numerical simulation of water-oil flow in naturally fractured reservoirs. *Society of Petroleum Engineers Journal*, 16(6):317–326, 1976. doi: 10.2118/5719-PA.

[25] S. H. Lee, M. F. Lough, and C. L. Jensen. Hierarchical modeling of flow in naturally fractured formations with multiple length scales. *Water Resources Research*, 37(3):443–455, 2001. doi: 10.1029/2000WR900340.

[26] P. Lemonnier and B. Bourbiaux. Simulation of naturally fractured reservoirs. State of the art: Part 2 – matrix-fracture transfers and typical features of numerical studies. *Oil & Gas Science and Technology-Revue de l'Institut Français du Pétrole*, 65(2):263–286, 2010. doi: 10.2516/ogst/2009067.

[27] K.-A. Lie. *An Introduction to Reservoir Simulation Using MATLAB/GNU Octave: User Guide for the MATLAB Reservoir Simulation Toolbox (MRST)*. Cambridge University Press, Cambridge, UK, 2019.

[28] K. Lim and K. Aziz. Matrix–fracture transfer shape factors for dual-porosity simulators. *Journal of Petroleum Science and Engineering*, 13(3–4):169–178, 1995. doi: 10.1016/0920-4105(95)00010-F.

[29] H. Lu, G. Di Donato, and M. J. Blunt. General transfer functions for multiphase flow in fractured reservoirs. *SPE Journal*, 13(3):289–297, 2008. doi: 10.2118/102542-PA.

[30] C. Maier and S. Geiger. Combining unstructured grids, discrete fracture representation and dual-porosity models for improved simulation of naturally fractured reservoirs. In *SPE Reservoir Characterization and Simulation Conference and Exhibition*. Society of Petroleum Engineers, 2013. doi: 10.2118/166049-MS.

[31] C. Maier, M. Karimi-Fard, A. Lapene, and L. Durlofsky. An MPFA-based dual continuum–discrete feature model for simulation of flow in fractured reservoirs. In *ECMOR XV-15th European Conference on the Mathematics of Oil Recovery*. European Association of Geoscientists & Engineers, 2016. doi: 10.3997/2214-4609.201601845.

[32] C. Maier, K. S. Schmid, M. A. Elfeel, and S. Geiger. Multi-rate mass-transfer dual-porosity modelling using the exact analytical solution for spontaneous imbibition. In *75th EAGE Conference and Exhibition 2013*, pp. 3386–3399. EAGE Publishing BV, 2013. doi: 10.3997/2214-4609.20130869.

[33] R. March, F. Doster, and S. Geiger. Accurate early-time and late-time modeling of countercurrent spontaneous imbibition. *Water Resources Research*, 52(8):6263–6276, 2016. doi: 10.1002/2015WR018456.

[34] R. March, F. Doster, and S. Geiger. Assessment of CO_2 storage potential in naturally fractured reservoirs with dual–porosity models. *Water Resources Research*, 54(3):1650–1668, 2018. doi: 10.1002/2017WR022159.

[35] R. March, H. Elder, F. Doster, and S. Geiger. Accurate dual-porosity modeling of CO_2 storage in fractured reservoirs. In *SPE Reservoir Simulation Conference February 20–22, Montgomery, TX*. 2017. doi: 10.2118/182646-MS.

[36] J. E. Monteagudo and A. Firoozabadi. Control-volume model for simulation of water injection in fractured media: incorporating matrix heterogeneity and reservoir wettability effects. *SPE Journal*, 12(3):355–366, 2007. doi: 10.2118/98108-PA.

[37] M. Panda, J. G. Ambrose, G. Beuhler, and P. L. McGguire. Optimized EOR design for the Eileen West End area, Greater Prudhoe Bay. *SPE Reservoir Evaluation & Engineering*, 12(1):25–32, 2009. doi: 10.2118/123030-PA.

[38] K. Pruess. A practical method for modeling fluid and heat flow in fractured porous media. *SPE Journal*, 25(1):14–26, apr 1985. doi: 10.2118/10509-PA.

[39] P. Quandalle and J. Sabathier. Typical features of a multipurpose reservoir simulator. *SPE Reservoir Engineering*, 4(4):475–480, 1989. doi: 10.2118/16007-PA.

[40] B. Ramirez, H. Kazemi, and M. Al-Kobaisi. A critical review for proper use of water–oil–gas transfer functions in dual-porosity naturally fractured reservoirs – part I. In *SPE Annual Technical Conference and Exhibition*. Society of Petroleum Engineers, 2007. doi: 10.2118/109821-MS.

[41] Schlumberger. *Eclipse Technical Description*. Schlumberger, 2014.

[42] K. Schmid and S. Geiger. Universal scaling of spontaneous imbibition for arbitrary petrophysical properties: water-wet and mixed-wet states and Handy's conjecture. *Journal of Petroleum Science and Engineering*, 101:44–61, 2013. doi: 10.1016/j.petrol.2012.11.015.

[43] F. Sonier, P. Souillard, and F. T. Blaskovich. Numerical simulation of naturally fractured reservoirs. *SPE Reservoir Engineering*, 3(4):1–114, 1988. doi: 10.2118/15627-PA.

[44] I. Stefansson, I. Berre, and E. Keilegavlen. Finite-volume discretisations for flow in fractured porous media. *Transport in Porous Media*, 124(2):439–462, 2018. doi: 10.1007/s11242-018-1077-3.

[45] R. D. Sydansk and L. Romero-Zeron. *Reservoir Conformance Improvement*. Society of Petroleum Engineers, Richardson, TX, 2011.

[46] M. L. Szulczewski, C. W. MacMinn, H. J. Herzog, and R. Juanes. Lifetime of carbon capture and storage as a climate-change mitigation technology. *Proceedings*

of the National Academy of Sciences, 109(14):5185–5189, 2012. doi: 10.1073/pnas.1115347109.

[47] J. Tecklenburg, I. Neuweiler, J. Carrera, and M. Dentz. Multi-rate mass transfer modeling of two-phase flow in highly heterogeneous fractured and porous media. *Advances in Water Resources*, 91:63–77, 2016. doi: 10.1016/j.advwatres.2016.02.010.

[48] J. Tecklenburg, I. Neuweiler, M. Dentz, J. Carrera, S. Geiger, C. Abramowski, and O. Silva. A non-local two-phase flow model for immiscible displacement in highly heterogeneous porous media and its parametrization. *Advances in Water Resources*, 62:475–487, 2013. doi: 10.1016/j.advwatres.2013.05.012.

[49] 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.

[50] D. Wong, F. Doster, S. Geiger, and A. Kamp. Partitioning thresholds in hybrid implicit–explicit representations of naturally fractured reservoirs. *Water Resources Research*, 56(2), 2020. doi: 10.1029/2019WR025774.