Master's Thesis

Adaptive Deflated Multiscale Solvers

Dmitrii Boitcov

to obtain the degree of Master of Science
at the Delft University of Technology

Thesis committee:	Prof. dr. ir. C. Vuik,	TU Delft, supervisor
	Dr. A. Lukyanov,	TU Delft, daily supervisor
	Dr. H. M. Schuttelaars	TU Delft, Mathematical Physics

Contents

1 Problem Formulation **1**
- 1.1 Reservoir Simulation . . . 1
- 1.2 Rock and Fluid properties . . . 1
- 1.3 Governing equations . . . 3
  - 1.3.1 Darcy's law . . . 3
  - 1.3.2 Mass-Balance equation . . . 4
  - 1.3.3 Single-phase flow equations . . . 4
  - 1.3.4 Multiphase flow equations . . . 5
  - 1.3.5 Boundary and initial conditions . . . 6
- 1.4 Black-Oil fluid model . . . 7
- 1.5 Discretization methods . . . 8
  - 1.5.1 Two-Point Flux Approximation (TPFA) . . . 8
  - 1.5.2 Multi-Point Flux Approximation (MPFA) . . . 12
- 1.6 MATLAB Reservoir Simulation Toolbox (MRST) . . . 13

2 Linear Solvers **15**
- 2.1 Introduction . . . 15
- 2.2 Newton-Raphson method . . . 15
- 2.3 Direct methods . . . 16
- 2.4 Iterative methods . . . 17
  - 2.4.1 Projection methods . . . 19
- 2.5 Preconditioning . . . 22

3 Deflation Method **25**
- 3.1 Introduction . . . 25
  - 3.1.1 Deflation preconditioning . . . 25
- 3.2 Preconditioner and GMRES . . . 26
  - 3.2.1 Restarted GMRES . . . 26
  - 3.2.2 Preconditioned GMRES . . . 28
- 3.3 Construction of deflation vectors . . . 29
  - 3.3.1 Ritz deflation . . . 29
  - 3.3.2 Harmonic Ritz deflation . . . 31
  - 3.3.3 Physics-based deflation . . . 31

4 Multiscale Methods **33**
- 4.1 Introduction . . . 33
  - 4.1.1 Upscaling Methods . . . 33
  - 4.1.2 Dual-Grid Methods . . . 34
  - 4.1.3 Multiscale Finite Element Method (MsFEM) . . . 34
  - 4.1.4 Multiscale Volume Element Method (MsFVM) . . . 35

5 Multiscale Restriction-Smoothed Basis Method (MsRSB) 37
5.1 Introduction . . . 37
5.2 Multiscale formulation . . . 38
5.2.1 Construction of basis functions . . . 38
5.3 Iterative multiscale formulation . . . 39

6 Adaptive Deflated Multiscale Solvers (ADMS) 41
6.1 Motivation . . . 41
6.2 Various forms of ADMS . . . 41
6.2.1 Fully ADMS . . . 41
6.2.2 Decoupled ADMS . . . 42
6.2.3 Mixed ADMS . . . 42

7 Results 43
7.1 ADMS parameters . . . 43
7.2 Test cases . . . 44
7.2.1 ”Islands” model problem . . . 45
7.2.2 Fractured reservoir . . . 50
7.2.3 SPE10 layer . . . 55

References 59

1

Problem Formulation

1.1. Reservoir Simulation

Reservoir simulation is used to implement and verify a mathematical model of fluid behaviour in a hydrocarbon reservoir over time depending on its petrophysical properties. The main purpose purpose of this modelling is to collect the geological information, process it and provide oil companies with the guidelines how to optimize and maximize oil and gas recovery.

In order to mathematically describe flow processes inside reservoir, two kinds of models are required. The first one is represented as a set of partial differential equations (PDEs) depicting how fluids actually flow in a porous medium. These relations are typically based on the principle of mass conservation of fluid phases, complemented with dependencies between various physical quantities. In addition, one needs a mathematical interpretation of the porous rock formation under consideration reflecting its geological characteristics. The second model has to be constructed in the form of a grid and used as input to the first (flow) model, thereby forming a complete reservoir simulation model.

This chapter is devoted to the physics and mechanics behind reservoir simulation. We start by defining the properties of the porous rock and fluids, and proceed with the mathematical formulations of the physical laws and relationships governing the flow.

1.2. Rock and Fluid properties

In a petroleum reservoir the size of the underlying rock bodies varies from ten to hundred meters in the vertical direction and can be an order of magnitude more (up to a few kilometres) in the lateral direction. That is why on this modelling scale it is almost impossible to take into account the storage and transport in individual pores and channels. For an accurate modelling of the reservoir geology, in that section we introduce macroscopic petrophysical quantities (we use them as input to flow simulators) that are based on a continuum hypothesis and volume averaging over

1

a sufficiently large Representative Elementary Volume (REV) in order to reproduce the authentic rock heterogeneities.

Continuum hypothesis implies that the amount of particles (molecules) in each control volume is large enough to express all flow variable as statistical averages in each balance volume of the entire computational domain [1]. *Representative Elementary Volume (REV)* indicate the smallest volumes, over which measurements can be taken so that they would be representative of the whole [2].

Pressure p denotes the force distributed over a surface.

Porosity ϕ is the relative volume of pores in the rock. To determine a rock partition, we need to know a ratio between the space occupied by impermeable rock and the space populated by pores in which a fluid can pass through. A porosity is equal to 0, when there are no pores at all, which means no fluid can flow through the rock. A porosity of 1 will on the other hand mean that the volume is void. Porosity value can be obtained using the following formula of rock compressibility c_r:

$$c_r = \frac{1}{\phi}\frac{\mathrm{d}\phi}{\mathrm{d}p} = \frac{\mathrm{d}\ln(\phi)}{\mathrm{d}p}.$$

Absolute Permeability is the rock property which characterizes the ability to transmit a fluid in different directions (one direction depends on other directions), or in other words the resistance that the rock offers to flow at certain (reservoir) conditions. The rock is called *permeable*, when it has large (well-connected) pores, and we call it *impermeable* otherwise, i.e. while pores are smaller and, as a consequence, less interconnected. The permeability is expressed in the form of a full tensor, ranging over many orders of magnitude throughout the reservoir. Having a full tensor, it often possible to diagonalize it to avoid certain difficulties. The permeability tensor is denoted as K, and is in 3D on the general form

$$\mathrm{K} = \begin{pmatrix} \mathrm{K}_{xx} & \mathrm{K}_{xy} & \mathrm{K}_{xz} \\ \mathrm{K}_{yx} & \mathrm{K}_{yy} & \mathrm{K}_{yz} \\ \mathrm{K}_{zx} & \mathrm{K}_{zy} & \mathrm{K}_{zz} \end{pmatrix}.$$

If the permeability can be represented by a scalar function $\mathrm{K}(\vec{x})$, we say that the permeability is *isotropic* as opposed to the *anisotropic* case where we need a full tensor $\mathrm{K}(\vec{x})$.

Relative permeability is a non-linear function of the saturation, which measures the effective permeability of a phase. Generally it is determined via experiments and denoted by $k_{r\alpha}$ for a phase α.

Saturation indicates proportions of the entire volume filled up with existing phases. A reservoir is completely saturated when there is no free space in the void in pores. Basically, a phase can be liquid or gas, and we distinguish three phases: water (ω), oil (o) and gas (g). We denote the saturation of each phase as S_α, $\alpha = \omega, o, g$, and for a fully saturated volume, we can write

$$\sum_{\alpha=\omega,o,g} S_\alpha = 1.$$

Velocity v is the rate that a phase travel through a medium. Sometimes we refer to the velocity of a single phase α as v_α, and the total velocity of all phases is

$$v = \sum_\alpha v_\alpha.$$

Density of a phase α denoted by ρ_α is the mass per unit of volume.

Viscosity μ_α for fluids is a measure of "thickness". A liquid with high viscosity is more resistant to flow than a liquid with low viscosity; the more viscous a fluid is, the slower it flows with the same pressure difference. Water has a lower viscosity than light oil, which again has lower viscosity than heavy oil. Heavy oil is also referred to as viscous oil.

Mass Fraction of a Phase Component c_α^l is the mass fraction that a component l occupies in a phase α. For each phase the mass fractions should add up to unity,

$$\sum_{l=1}^{M} c_\alpha^l = 1. \tag{1.1}$$

1.3. Governing equations

For the flow model, pressure, velocity and saturation are the primary unknowns. In this section we will derive the governing modelling equations for flow in porous media. An extensive introduction to the physics and dynamics behind fluid flow thorough porous media can be found in [3, 4].

1.3.1. Darcy's law

To model the fluid flow through the porous rock, the constitutive relation called Darcy's law is used. This empirical relation is derived from the Navier-Stokes equations via homogenization and only valid for slow, viscous flow [5]; flow in subsurface reservoirs falls into this category. Darcy's law states that the velocity of the fluid is proportional to a combination of the gradient of the fluid pressure and the effects due to gravity:

$$\vec{v} = -\frac{\mathrm{K}}{\mu}\left(\nabla p - g\rho\nabla z\right), \tag{1.2}$$

where $\vec{v}$ is the fluid velocity, which represents the volume of fluid per total area per time, K is the permeability tensor of the porous medium, μ is the fluid viscosity, p is the fluid pressure, g is the gravitational acceleration, ρ is the density of the fluid and z is the vertical coordinate.

Assuming that we suppress gravitational forces, i.e., $g = 0$, we can make the following two observations: no flow will occur if there is no pressure gradient; if there is a pressure gradient, the fluid will flow from high pressure towards low pressure.

1

1.3.2. Mass-Balance equation

The law of mass conservation is fundamental for the mass-balance equation, which gives the basic equation for the flow model. The law states that mass can neither be created nor destroyed; mass can only change its form. For a closed system where mass is neither flowing in nor out, the mass inside the system is constant, regardless of the processes acting inside.

In most practical applications of flow simulation, we do not have closed systems. For a reservoir model, we might have flow coming in and out through boundaries, or wells that inject or produce mass. To account for material entering or leaving the system, we use the principle of mass-balance. The mass-balance principle states that mass entering the system must either leave the system, or be accumulated within it. This is a direct extension of the law of mass conservation. The mass-balance equation for an arbitrary domain Ω with boundary $\partial\Omega$ and normal vector $\vec{n}$ can be written in integral form as,

$$\underbrace{\frac{\partial}{\partial t}\int_{\Omega}\phi\rho\,\mathrm{d}\vec{x}}_{\text{accumulation}}+\underbrace{\int_{\partial\Omega}\rho\vec{v}\cdot\vec{n}\,\mathrm{d}s}_{\text{outflow}}=\underbrace{\int_{\Omega}\rho q\,\mathrm{d}\vec{x}}_{\text{inflow}}. \tag{1.3}$$

The change of a fluid within Ω is determined by the flux $\vec{v}$ through $\partial\Omega$, and the amount of the matter created within Ω, denoted by q. A positive q implies that we have a source, and a negative value implies a sink.

The Gauss' (divergence) theorem applied to (1.3) gives

$$\int_{\Omega}\left[\frac{\partial}{\partial t}\phi\rho+\nabla\cdot(\rho\vec{v})\right]\mathrm{d}\vec{x}=\int_{\Omega}\rho q\,\mathrm{d}\vec{x}. \tag{1.4}$$

Since (1.4) holds for an arbitrary domain Ω, and in particular for infinitesimally small, the integrands have to be equal, which means integral signs may be dropped, i.e. it follows that the macroscopic behaviour of the single-phase fluid must satisfy the continuity equation:

$$\frac{\partial(\phi\rho)}{\partial t}+\nabla\cdot(\rho\vec{v})=\rho q. \tag{1.5}$$

1.3.3. Single-phase flow equations

Equation (1.5) contains more unknowns than equations and to derive a closed mathematical model, we need to introduce what is commonly referred to as constitutive equations that give the relationship between different states of the system (pressure, volume, temperature, etc.) at given physical conditions. While Darcy's law provides a relationship between the fluid velocity $\vec{v}$ and pressure p, the rock compressibility c_r definition describes the relationship between the pressure p and porosity ϕ. In a similar way, we can introduce the fluid compressibility to relate the fluid pressure p to the fluid density ρ:

$$c_f=\frac{1}{\rho}\frac{\mathrm{d}\rho}{\mathrm{d}p}=\frac{\mathrm{d}\ln(\rho)}{\mathrm{d}p},$$

where c_f denotes the *isothermal compressibility*, which we henceforth will refer to as the fluid compressibility, is non-negative and will generally depend on both pressure and temperature, i.e., $c_f = c_f(p,T)$.

Inserting Darcy's law (1.2) and compressibility notations c_r and c_f into (1.5), the following *parabolic* equation for the fluid pressure is derived:

$$c_t \phi \rho \frac{\partial p}{\partial t} - \nabla \cdot \left[\frac{\rho \mathrm{K}}{\mu} (\nabla p - g\rho \nabla z) \right] = \rho q, \tag{1.6}$$

where $c_t = c_r + c_f$ denotes the total compressibility. It is important to that this equation is generally *nonlinear* since both ρ and c_t may depend on p. However, there are several scenarios, where the governing single-phase flow equation becomes linear for the initially unknown variables. More extensive discussions on derivation of the governing equations and some specific cases can be found in standard textbooks [6, 7].

Incompressible flow. In the special case of an incompressible rock and fluid (that is, ρ and ϕ are independent of p so that $c_t = 0$), (1.6) simplifies to an *elliptic* equation with variable coefficients,

$$-\nabla \cdot \left[\frac{\rho \mathrm{K}}{\mu} (\nabla p - g\rho \nabla z) \right] = q. \tag{1.7}$$

Introducing the fluid potential Φ, which is defined as $\Phi = p - g\rho z$, (1.7) is identified as the generalized Poisson's type *pressure equation*

$$-\nabla \cdot \mathrm{K} \nabla \Phi = q \tag{1.8}$$

or as the Laplace equation

$$\nabla \cdot \mathrm{K} \nabla \Phi = 0,$$

if there are no volumetric fluid sources or sinks.

1.3.4. Multiphase flow equations

Similarly to the derivation of single-phase flow equations we can state generic equations which describe multiphase flow of immiscible fluids. For the detailed overview of multiphase flow fundamentals we refer to [8] and [9].

Single-component phases. For a system of N immiscible fluid phases that each consists of a single component, we write one mass conservation equation for every phase α,

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

Here, each phase may contain multiple chemical species, which make up a single component since the composition of each phase remains constant in time and there is no transfer between phases.

To obtain a closed model we can extend Darcy's law (1.2) by applying relative permeabilities concept [10],

$$\vec{v}_\alpha = -\frac{\mathrm{K} k_{r\alpha}}{\mu_\alpha} (\nabla p_\alpha - g\rho_\alpha \nabla z).$$

1

Multicomponent phases. There are a lot of real-world examples of reservoirs, where phases may comprise a few chemical species which are mixed at the molecular level and keep the same properties like temperature and velocity. Different from immiscible fluids, such a scenario is influenced by Brownian motion and dispersion, causing the components redistribution in case of macroscale gradients in the mass fractions. To simulate this situation we need to apply a linear Fickian diffusion,

$$\vec{J}_\alpha^l = -\rho_\alpha S_\alpha \mathrm{D}_\alpha^l \nabla c_\alpha^l,$$

where l refers to the component, α denotes the phase and D_α^l is the diffusion tensor.

For a system of N fluid phases and M chemical species, the mass conservation balance equation for components $l = 1, \dots, M$ reads,

$$\frac{\partial}{\partial t}\left(\phi \sum_\alpha c_\alpha^l \rho_\alpha S_\alpha\right) + \nabla \cdot \left(\sum_\alpha c_\alpha^l \rho_\alpha \vec{v}_\alpha + \vec{J}_\alpha^l\right) = \sum_\alpha c_\alpha^l \rho_\alpha q_\alpha,$$

where $\vec{v}_\alpha$ is the superficial phase velocity and q is the source term.

The system is closed in the same way as for single-component phases, except that we now also have to use that the mass fractions sum to one (1.1).

1.3.5. Boundary and initial conditions

To guarantee that the solution to any of the governing flow equations is well-posed (i.e. there exists a unique solution, which is continuously dependent on the initial and boundary conditions) inside a finite computational domain, we need to specify boundary conditions that determine the behaviour on the external boundary. In a parabolic case one also needs to impose an initial condition that determines the initial state of the fluid system. In the following paragraphs, we will give a more detailed overview of these auxiliary conditions.

In reservoir simulation one of the most realistic scenarios is closed systems where no flow occurs across its outer boundaries, since we actually study a lot of full reservoirs that have trapped and contained petroleum fluids for million of years. Formally, such boundary conditions are modelled in terms of homogeneous Neumann conditions,

$$\vec{v} \cdot \vec{n} = 0 \quad \text{for } \vec{x} \in \partial\Omega.$$

Alternatively, the reservoir can be connected to a larger object which provides additional pressure support. This situation is described by Dirichlet type boundary conditions of the form

$$p(\vec{x}) = p_a(\vec{x}, t) \quad \text{for } \vec{x} \in \Gamma_a \subset \partial\Omega,$$

where the function p_a may, for example, be given as a hydrostatic condition.

It is also common that there is a prescribed influx at the boundary, which is described by supplying inhomogeneous Neumann conditions,

$$\vec{v} \cdot \vec{n} = u_a(\vec{x}, t) \quad \text{for } \vec{x} \in \Gamma_a \subset \partial\Omega.$$

Combinations of the various boundary conditions are used to model and analyse parts of a reservoir.

1.4. Black-Oil fluid model

Black-oil modelling is the most common technique in the petroleum industry to simulate oil and gas recovery for a multicomponent, multiphase flow, when there is no diffusion among components [11]. This model allows to predict compressibility and mass transfer effects between phases, which are crucial parts of primary (pressure depletion) and secondary (water injection) recoveries.

The black-oil flow equations exploit several laws and relationships:

- equation of state;
- thermodynamic equilibrium;
- Darcy's law for the volumetric flow rates;
- mass conservation equation for each component.

Equation of state (EOS) provides constitutive relationships between mass, pressures, temperature, and volumes at thermodynamic equilibrium. There are a lot of different ways to write this relation, but here we present a popular one used in reservoir simulation, specifically in a cubic form,

$$Z^3 + f_2(A,B)Z^2 + f_1(A,B)Z + f_0(A,B) = 0,$$

with

$$Z = \frac{pV}{RT}, \quad A = \frac{a\beta p}{(RT)^2}, \quad B = \frac{bp}{RT},$$

where a and b are functions of the temperature and the molar volume at the critical point; β depends on the acentric factor of the species, that measures molecules centricity in the fluid, temperature, and temperature at the critical point; and R is the universal gas constant.

By convention, the black-oil equations are formulated as conservation of gas, oil, and water volumes at standard (surface) conditions rather than conservation of the corresponding component masses. To formulate equations in the final form let us introduce a few parameters used in standard PVT (Pressure-Volume-Temperature) models that employ some pressure-dependent functions to establish a relation between fluid volumes at reservoir and surface conditions. More precisely, we need the inverse of formation-volume factors B_l defined as

$$B_l = \frac{V_l}{V_l^s} \;\Rightarrow\; b_l := B_l^{-1} = \frac{V_l^s}{V_l},$$

where V_l and V_l^s are volumes occupied by a bulk of component l at reservoir and surface conditions, respectively. Also we use solution oil-gas ratio denoted by

$$r_{so} = \frac{V_g^s}{V_0^s}.$$

1

Using all these notations, the black-oil equations for a live-oil (gas is dissolved in oil) system reads,

$$\frac{\partial}{\partial t}(\phi b_o S_o) + \nabla \cdot (b_o \vec{v}_o) - b_o q_o = 0,$$
$$\frac{\partial}{\partial t}(\phi b_w S_w) + \nabla \cdot (b_w \vec{v}_w) - b_w q_w = 0,$$
$$\frac{\partial}{\partial t}\left[\phi\left(b_g S_g + b_o r_{so} S_o\right)\right] + \nabla \cdot \left(b_g \vec{v}_g + b_o r_{so} \vec{v}_o\right) - \left(b_g q_g + b_o r_{so} q_o\right) = 0.$$

Note that, in order to account for any specific phenomenon in the reservoir, the model can be extended by adding necessary quantities and relations.

1.5. Discretization methods

To solve the model equations numerically, the problem needs to be discretized. There are a large amount of various finite-difference and finite-volume schemes, as well as finite-element techniques based on standard (or mixed discontinuous) Galerkin approximations. In classical finite-difference approach the partial derivatives are approximated by linear combinations of function values at the discrete set of grid points in the domain. On the other hand, finite-volume methods rely on a more physical aspect and consist of two steps. Firstly, the initial computational domain is divided into a number of control volumes, where the primary unknowns are (usually) located at the centroid of the control volume. Secondly, it is necessary to integrate the differential form of the governing equations over each control volume that results in so-called discretized (or discretization) equation. Thus, the discretization equation represents the conservation principle for the variable inside the control volume. The most crucial feature of finite-volume methods is that the obtained solution satisfies the conservation law of physical quantities like mass, energy and momentum for any control volume as well as for the entire domain.

In this section, we introduce several recently designed methods, that are specially suited for problems with highly discontinuous coefficients, which are typically seen in realistic reservoir simulation models. In particular, two discretization schemes are presented: two-point flux approximations (TPFA), which uses only two cells to approximate the flux, and multi-point flux approximations (MPFA), which consider more than two cells.

1.5.1. Two-Point Flux Approximation (TPFA)

The two-point flux approximation (TPFA) scheme is used extensively throughout industry as the simplest example of a finite-volume discretization, which is basically used for elliptic flow problem (pressure equation) [3]. In this method, the flux across the certain face of the control volume is approximated by the pressure values in corresponding cells on each side of the face. The calculated discretization coefficient is called $transmissibility$, which in fact is the harmonic mean of the adjacent transmissibilities weighted by the distance to its cell centres.

To avoid any technical difficulties related to model complexity, we will without loss of generality consider the simplified single-phase flow equation. We actually

want to solve the equation for pressure p given by the conservation law

$$-\nabla \cdot \lambda \nabla p = q, \quad \vec{x} \in \Omega,$$

assuming that there is no-flow on the boundary, i.e.,

$$\vec{v} \cdot \vec{n} = 0, \quad \vec{x} \in \partial\Omega,$$

where velocity is given by Darcy's law:

$$\vec{v} = -\lambda \nabla p, \quad \vec{x} \in \Omega.$$

Let us divide the physical domain Ω into a set of control volumes E_i:

$$\Omega = \bigcup_{i=1}^{N_E} E_i, \quad E_i \cap E_j = \emptyset \text{ for } i \neq j$$

where quantities inside E_i represent the average of the physical quantities inside this control volume.

A set of mass-balance equations is obtained by integrating the pressure equation over an arbitrary control volume E_i:

$$\int_{E_i} -\nabla \cdot \lambda \nabla p \, \mathrm{d}\vec{x} = \int_{E_i} q \, \mathrm{d}\vec{x} \tag{1.9}$$

We assume, that $\vec{v} = \lambda \nabla p$ is sufficiently smooth for (1.9) to hold. Let γ_{ij} be the interface between E_i and a neighbouring control volume E_j with area A_{ij} and normal $\vec{n}_{ij}$,

$$\gamma_{ij} = \partial E_i \cap \partial E_j.$$

Invoking the divergence theorem transforms (1.9) into

$$\sum_j \int_{\gamma_{ij}} -\lambda \nabla p \cdot \vec{n}_{ij} \, \mathrm{d}s = \int_{E_i} q \, \mathrm{d}\vec{x}$$

The two-point flux approximation uses two points to calculate the flux across an interface. The flux between E_i and E_j across γ_{ij} can simply be expressed as

$$\vec{v}_{ij} = -\int_{\gamma_{ij}} \lambda \nabla p \cdot \vec{n}_{ij} \, \mathrm{d}s, \tag{1.10}$$

where the gradient of pressure is calculated using a central finite difference stencil:

$$\nabla p \approx \frac{2(p_j - p_i)}{d_i + d_j} \quad \text{on} \quad \gamma_{ij}. \tag{1.11}$$

Here p_i and p_j indicate the averaged pressure in E_i and E_j volumes, respectively, whereas d_i and d_j indicate the distance between γ_{ij} and the corresponding cell centres.

1

Using the approximation of ∇p in (1.11), (1.10) becomes

$$\vec{v}_{ij} = -\frac{2(p_j - p_i)}{d_i + d_j} \int_{\gamma_{ij}} \lambda \cdot \vec{n}_{ij} \, \mathrm{d}s. \tag{1.12}$$

Let us denote the directional cell permeabilities as $\lambda_{i,ij} = \vec{n}_{ij} \cdot \lambda_i \vec{n}_{ij}$ and $\lambda_{j,ij} = \vec{n}_{ij} \cdot \lambda_j \vec{n}_{ij}$. A distance-weighted harmonic mean for λ across the interface γ_{ij}

$$\lambda_{ij} = (d_i + d_j) \left(\frac{d_i}{\lambda_{i,ij}} + \frac{d_j}{\lambda_{j,ij}} \right)^{-1}$$

is applied to estimate the integral in (1.12) and gives:

$$\begin{aligned} \vec{v}_{ij} &= -\frac{2(p_j - p_i)}{d_i + d_j} \int_{\gamma_{ij}} \lambda \cdot \vec{n}_{ij} \, \mathrm{d}s \\ &= -\frac{2(p_j - p_i)}{d_i + d_j} \int_{\gamma_{ij}} (d_i + d_j) \left(\frac{d_i}{\lambda_{i,ij}} + \frac{d_j}{\lambda_{j,ij}} \right)^{-1} \cdot \vec{n}_{ij} \, \mathrm{d}s \\ &= 2|\gamma_{ij}|(p_i - p_j) \left(\frac{d_i}{\lambda_{i,ij}} + \frac{d_j}{\lambda_{j,ij}} \right)^{-1}. \end{aligned} \tag{1.13}$$

We now define the transmissibilities as

$$[\mathrm{T}]_{ij} = 2|\gamma_{ij}| \left(\frac{d_i}{\lambda_{i,ij}} + \frac{d_j}{\lambda_{j,ij}} \right)^{-1}$$

and we can write

$$\sum_j [\mathrm{T}]_{ij}(p_i - p_j) = \int_{E_i} q \, \mathrm{d}\vec{x}. \tag{1.14}$$

Let $\bar{q} = \{\bar{q}_i\}$ be a vector defined by

$$\bar{q} = \int_{E_i} q \, \mathrm{d}\vec{x}.$$

Then (1.14) is a symmetric linear system

$$\mathrm{A}p = \bar{q}, \tag{1.15}$$

since

$$\begin{aligned} \sum_j [\mathrm{T}]_{ij}(p_j - p_i) &= [\mathrm{T}]_{i1}(p_i - p_1) + \ldots + [\mathrm{T}]_{ii}(p_i - p_i) + \ldots + [\mathrm{T}]_{iN_E}(p_i - p_{N_E}) \\ &= \bar{q}_i. \end{aligned}$$

Consequently, in a short form

$$[\mathrm{A}]_{ij} = \begin{cases} \sum_k [\mathrm{T}]_{ik} & \text{if } j = i \\ -[\mathrm{T}]_{ij} & \text{if } E_i \cap E_j \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

Forcing $p_1 = 0$ by adding a positive constant to the first diagonal element of A makes the system positive definite. Moreover, symmetry is also preserved by specifying the pressure in a single point. What is essential for computational reasons, the discretization leads to the M-matrix A has a sparse banded structure for structured grids (tridiagonal for 1D grids and penta- and heptadiagonal for Cartesian grids in 2D and 3D, respectively).

After solving the linear system for p, the fluxes across each interface are calculated via (1.10).

In order to identify drawbacks of the TPFA, we will briefly discuss orthogonality of grids. Let us consider an arbitrary grid-cell E_i with cell-centre x_i connected to the neighbouring grid-cell E_j with cell-centre x_j by the line l_{ij}. In addition, we denote by γ_{ij} the interfaces of E_i for $j = 1, \dots, N_\gamma^i$, where N_γ^i signifies the total number of E_i neighbours.

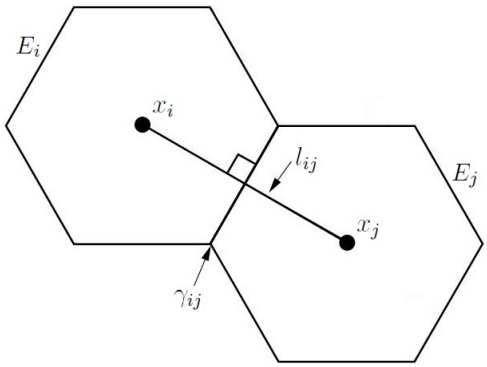


Figure 1.1: Orthogonality between grid-cells E_i and E_j

The grid is called $orthogonal$ (Fig. 1.1), if

$$l_{ij} \cdot \gamma_{ij} = 0$$

holds for all unequal $i = 1, \dots, N_E$ and $j = 1, \dots, N_\gamma^i$.

Similarly, the grid is called $\mathrm{K} - orthogonal$ (orthogonality with respect to the permeability tensor K), if

$$l_{ij} \cdot \mathrm{K}\gamma_{ij} = 0.$$

The TPFA scheme is monotone and robust as well as relatively cheap and easy to implement. However, the TPFA scheme is consistent and convergent only for K-orthogonal grids [12].

For non-K-orthogonal grids, TPFA gives an error in the solution which does not vanish as the grids are refined. Unfortunately, in case of the reservoir with special features like faults and channels K-orthogonality is often lost. Therefore, in order to be able to solve the flow equations on general grids, the multi-point flux approximation (MPFA) is introduced.

1

1.5.2. Multi-Point Flux Approximation (MPFA)

The main idea of multi-point flux approximation (MPFA) schemes is to increase the number of points in the stencil in the expression of the flux in order to overcome the TPFA approach disadvantages and improve the flexibility of grids to be used in reservoir simulation [13]. MPFA methods use a generalization of the harmonic mean of permeabilities (as mentioned above, a distance-weighted harmonic mean is used in standard TPFA scheme) to handle general anisotropy, heterogeneity, and grid irregularity. Convergence and stability of the MPFA methods are discussed in [14]. Obviously, the coefficient matrix for MPFA is less sparse than TPFA matrix. In particular, the sparsity patterns for both discretization types are presented in the following table:

dimension (grid)	TPFA	MPFA
1D	tridiagonal	-
2D (quadrilateral)	pentadiagonal	9-diagonal
3D (hexahedral)	heptadiagonal	27-diagonal

O-method. The most popular MPFA scheme is the O-method, which for quadrilateral grids was introduced in [15]. For numerical experiments and convergence proofs we refer to [16]. Note that the O-method is not symmetric for general grids, and consequently is only conditionally convergent (see [14] for a convergence criterion on quadrilateral grids and [17] on rough grids). In this method we firstly have to define an interaction region around each corner-point in the grid, restricted by faces that connect cell centroids and face centroids (see Fig. 1.2 [18]). Next, a set of linear (pressure) functions is defined for each interaction region, where it is required that they are continuous at the cell centroids and flux-continuous across the face patches. A globally coupled system is constructed using the gradients of the linear functions to express the corresponding flux across the face patches inside the interaction region, in terms of the unknown cell pressures p_i. It is important to remember that one needs to solve a local system of equations at this step. During this process a list of subface transmissibility coefficients is generated and combined with neighbouring interaction regions in order to generate faces contributions. Eventually, the cell pressures are calculated by requiring mass conservation and summing the fluxes across all face patches. It was shown by many authors [12, 15, 19] that O-method outperforms TPFA discretization and produce lesser errors.

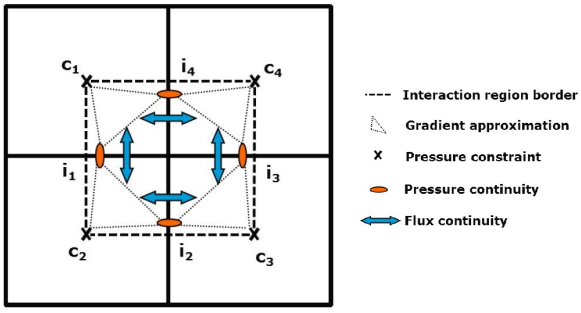
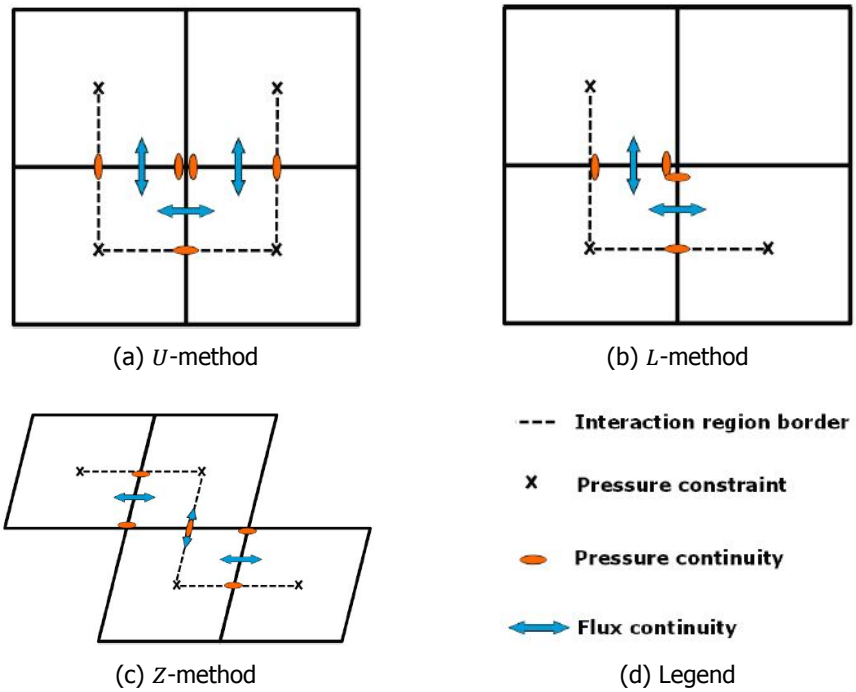


Figure 1.2: Two dimensional O-method

In addition to the classic O-method, there is another family of multi-point schemes, namely *compact* methods, which basically have smaller stencils than cell's direct neighbourhood compared to full schemes. These methods are constructed in a similar way to the O-method, since the same dual grid and interaction regions are used to enforce flux and pressure continuity on the subfaces. The only difference between various compact methods is due to distinction in the enforcement of these continuity constraints. In the figure below (Fig. 1.3 [18]) the most widely used compact techniques such as U-method [20], L-method [21] and Z-method [22] are presented:



(a) U-method (b) L-method

(c) Z-method (d) Legend

Figure 1.3: Two dimensional examples of compact schemes

1.6. MATLAB Reservoir Simulation Toolbox (MRST)

All simulations and numerical experiments of that thesis are conducted in the **MATLAB Reservoir Simulation Toolbox (MRST)**, which is developed by the Computational Geosciences group in the Department of Applied Mathematics at *SINTEF ICT* as a free open-source software for reservoir modelling and simulation. It includes many state-of-the-art numerical methods to process a wide range of public data sets: real-world reservoir examples such that the Norne field in the Norwegian Sea operated by Statoil and the Johansen formation on the south-west coast of Norway as well as realistic geological scenarios like SPE 1,3,9 and 10 benchmarks. Basically it is not a simulator itself, but allows to create your own custom model and analyse fluid behaviour. Also the functionality offers a large set of add-

on modules including discretizations (both TPFA and MPFA-O methods) and solvers, simulators for incompressible and compressible flow, multiscale methods and visualization of simulation output, and so on.

Public releases of the package can be downloaded from the webpage:

http://www.sintef.no/MRST/

A comprehensive introduction to the reservoir simulation and its implementation in the MRST is available in the user guide [23].

2

Linear Solvers

2.1. Introduction

Numerical simulation of the flow in porous media consists of the solving the discretized linear system arising from the non-linear governing equations for each iteration in the Newton-Raphson method. Unfortunately, in general case there can be a large number of Newton iterations to be done required for an accurate reservoir simulation. Therefore, the corresponding number of linear equations need to be solved, and, as a consequence, linear solver execution time becomes the dominant part of the total computational effort.

2.2. Newton-Raphson method

Newton-Raphson method, also simply known as Newton's iteration, is a famous commonly used *root-finding algorithm*, where the initial guess x_0 is already pretty close to the exact solution x^*.

Let us consider the non-linear system

$$f(x) = 0, \tag{2.1}$$

which represents governing equations in a generalized form, where x includes all unknown variables such as pressure, saturation and mass fraction variables. A Taylor expansion of (2.1) in the neighbourhood of x, involving first two terms and writing higher order terms as a remainder using big-$\mathcal{O}$ notation, yields,

$$f(x + \delta x) = f(x) + \nabla f(x) \cdot \delta x + \mathcal{O}(\delta x^2).$$

Then, we construct a convergent sequence of approximate solutions

$$x_{k+1} = x_k + \delta x, \ k = 1, 2, 3, \ldots$$

by means of the iterative process,

$$\nabla f(x_k) \cdot (x_{k+1} - x_k) = -f(x_k),$$

which can be rewritten in a matrix equation form as

$$A\delta x = b,$$

where A is the *Jacobian* of the function f at x_k and $b = -f(x_k)$.

In the next two sections we consider (sparse) linear algebraic systems $Ax = b$ with a nonsingular matrix $A \in \mathcal{R}^{n\times n}$, so that $x = A^{-1}b$ is well defined, and introduce two families (*direct* and *iterative*) of techniques for solving these equations.

2.3. Direct methods

Direct methods for solving $Ax = b$ are implicitly based on a decomposition of A into easily solvable factors and subsequent solution of the systems involving these factors. Different from iterative methods, which will be described in the next section, no intermediate approximations are calculated, and the final approximate solution is available only at the end of the process.

Cholesky decomposition. For any symmetric positive definite matrix $A \in \mathcal{R}^{n\times n}$, i.e. $x^T Ax > 0$ for all $x \in \mathcal{R}^n \setminus \{0\}$, there exists the Cholesky decomposition of the form

$$A = LL^T,$$

where $L = [l_{ij}] \in \mathcal{R}^{n\times n}$ with $l_{ii} > 0$ is a uniquely determined lower triangular matrix. The total cost of this procedure is $\frac{1}{3}n^3$ flops for large n. Once this factorization is done, we obtain

$$x = A^{-1}b = (LL^T)^{-1}b = L^{-T}(L^{-1}b),$$

so that x can be computed by solving two triangular systems,

1. *forward substitution*: solve for y from $Ly = b \;\Rightarrow\; y = L^{-1}b$,
2. *backward substitution*: solve for x from $L^T x = y \;\Rightarrow\; x = L^{-T}y = A^{-1}b$.

A lower (upper) triangular system can be solved using forward (backward) substitution. For $Ly = b$ we have

$$y_j = \frac{1}{l_{jj}}\left(b_j - \sum_{k=1}^{j-1} l_{jk}y_k\right),\ j = 1, 2, \ldots, n.$$

Computing y_j costs j multiplications and $j-1$ subtractions, and hence the total cost is $\sum_{j=1}^{n}(2j-1) = n^2$ flops.

LU decomposition. Gaussian elimination. Let us now consider a general nonsingular matrix $A \in \mathcal{R}^{n\times n}$. If the matrices $A(1:k, 1:k) \in \mathcal{R}^{k\times k}$ for all $k = 1, \ldots, n$ are nonsingular, then there exists an LU decomposition,

$$A = LU,$$

where L is a unit lower triangular and U is an upper triangular matrix. For this factorization,

$$x = A^{-1}b = U^{-1}(L^{-1}b),$$

so that again x can be computed via forward and backward substitutions. The LU decomposition can be calculated using *Gaussian elimination*: at step $j = 1, \dots, n-1$, multiples of the j-th row are subtracted from rows $j+1, \dots, n$ in order to put zeros in the column j below the entry in cell (j, j). As a result, we obtain the upper triangular matrix U. Each step j in the process can be considered as one left-multiplication of A by a suitable lower triangular matrix L_j. After $n-1$ steps we get

$$L_{n-1} \cdots L_1 A = U \;\Rightarrow\; A = (L_1^{-1} \cdots L_{n-1}^{-1})U =: LU.$$

Note that, generally cost is (approximately) twice as expensive as Cholesky decomposition, i.e. around $\frac{2}{3}n^3$ flops for large n.

Pivoting strategy should be applied to avoid numerical instabilities, if A has fairly small elements on the diagonal.

2.4. Iterative methods

As we mentioned in the previous section, a significant difference between iterative and direct solution methods for solving linear algebraic systems is that the former generate intermediate approximations (called *iterates*), while the latter yield an approximation of the exact solution only at the very end of the computation. Using these iterates it is possible to estimate the error (or *residual*) norm, which allows to stop the iteration when desired accuracy is achieved. This can be an essential advantage in practical applications, where we usually do not require a highly accurate approximation of the exact solution.

In this section we consider a linear algebraic system $Ax = b$ with nonsingular $A \in \mathcal{R}^{n,n}$. Most "classical" iterative methods rely on a splitting $A = M - N$, where M should be easily invertible (e.g. diagonal or triangular). Then $Ax = b$ can be rewritten as $(M - N)x = b$, or

$$x = M^{-1}Nx + M^{-1}b,$$

which leads to the iterative method

$$x_{k+1} = M^{-1}Nx_k + M^{-1}b, \quad k = 0, 1, 2, \dots, \tag{2.2}$$

where x_0 is a given initial approximation.

The k-th error is defined by $e_k := x - x_k$ and satisfies

$$e_k = x - x_k = x - (M^{-1}Nx_{k-1} + M^{-1}(M - N)x) = M^{-1}Ne_{k-1}.$$

Therefore by induction we get

$$e_k = (M^{-1}N)^k e_0.$$

The matrix $M^{-1}N$ is called the *iteration matrix* of the method (2.2). Applying the Jordan decomposition to the iteration matrix we obtain

$$M^{-1}N = PJP^{-1} \;\Rightarrow\; (M^{-1}N)^k = PJ^kP^{-1}.$$

Thus, if the *spectral radius* (i.e., the largest absolute value of its eigenvalues: $\rho(A) = \max\{|\lambda_1|, \dots, |\lambda_n|\}$) of the iteration matrix satisfies $\rho(M^{-1}N) < 1$, then $J^k \to 0$ and hence $(M^{-1}N)^k \to 0$ for $k \to \infty$, and for each x_0 the method (2.2) converges with $e_k \to 0$ for $k \to \infty$.

If we split A as

$$A = L + D + U,$$

where L and U are strictly lower and upper triangular matrices, while D is a diagonal part, then the following classical methods are derived:

- Jacobi method

$$\begin{aligned} M &= D, \\ N &= -(L + U), \\ M^{-1}N &= -D^{-1}(L + U) =: R_J. \end{aligned}$$

- Gauss-Seidel method

$$\begin{aligned} M &= L + D, \\ N &= -U, \\ M^{-1}N &= -(L + D)^{-1}U =: R_{GS}. \end{aligned}$$

Both methods converge when A is diagonally dominant, i.e. $|a_{ii}| \geq \sum_{i \neq j} |a_{ij}|$ for all i. However, when $\rho(M^{-1}N)$ is close to 1, the convergence may be very slow.

In order to improve the convergence rate a *relaxation parameter* $\omega > 0$ is introduced and instead of $Ax = b$ the modified equation $\omega Ax = \omega b$ is considered. Now splitting looks like

$$\omega A = \omega(L + D + U) = (D + \omega L) + (\omega U + (\omega - 1)D) =: M - N$$

which results in the method

$$x_{k+1} = R_{SOR}(\omega)x_k + \omega M^{-1}b,$$

where

$$R_{SOR}(\omega) = -(D + \omega L)^{-1}(\omega U + (\omega - 1)D).$$

For $0 < \omega < 1$ the method is called *under-relaxation method*, in case of $\omega = 1$ it is the Gauss-Seidel method and for $\omega > 1$ the method is called *Successive Over Relaxation (SOR) method*.

More generally, we can consider $A = M - N$ and apply relaxation parameter $\omega > 0$ to this splitting as follows,

$$\begin{aligned} \omega Ax = \omega b &\iff \omega(M-N)x = \omega b \\ &\iff x = (\omega M^{-1}N + (1-\omega)I)x + \omega M^{-1}b, \end{aligned}$$

which results in the iterative sequence,

$$x_{k+1} = R(\omega)x_k + \omega M^{-1}b,$$

where

$$R(\omega) = (1-\omega)I + \omega M^{-1}N.$$

The convergence criterion is now determined by $\rho(R(\omega))$ or $\|R(\omega)\|$, since $e_k = R(\omega)^k e_0$, giving

$$\frac{\|e_k\|}{\|e_0\|} \leq \|R(\omega)\|^k, \quad k = 0, 1, 2, \ldots$$

In all such iterations the convergence is asymptotically (for large k) linear, with the average reduction factor per step given by $\|R(\omega)\|$.

2.4.1. Projection methods

Projection methods are based, as follows from its title, on projections onto subspaces. Suppose that x_0 is a given initial guess of $x = A^{-1}b$. In step $k = 1, 2, 3, \ldots$ we construct approximations of the form

$$x_k \in x_0 + \mathcal{S}_k, \tag{2.3}$$

where $\mathcal{S}_k$ is a k-dimensional ($k \leq n$) subspace of $\mathbb{C}^n$ called *search space*. Since there are k degrees of freedom used to construct x_k, k constraints are needed to determine x_k. Imposing these conditions on the residual $r_k = b - Ax_k$, we require that

$$r_k \perp \mathcal{C}_k, \tag{2.4}$$

where $\mathcal{C}_k$ is a k-dimensional subspace of $\mathbb{C}^n$ called the *constraint space*. Suppose that the columns of $S_k, C_k \in \mathbb{C}^{n\times k}$ form bases of $\mathcal{S}_k$ and $\mathcal{C}_k$, respectively, then (2.3) and (2.4) can be rewritten as,

$$x_k = x_0 + S_k t_k \quad \text{for some} \quad t_k \in \mathbb{C}^k,$$

and

$$0 = C_k^H r_k = C_k^H(b - Ax_0 - S_k t_k) \iff C_k^H A S_k t_k = C_k^H r_0.$$

Let the projection method to be well-defined at step k, i.e. t_k is uniquely determined (it depends only on $A, \mathcal{S}_k, \mathcal{C}_k$, but not on the choice of its bases). Then

$$t_k = (C_k^H A S_k)^{-1} C_k^H r_0,$$

hence

$$x_k = x_0 + S_k t_k = x_0 + S_k (C_k^H A S_k)^{-1} C_k^H r_0,$$

and

$$r_k = b - Ax_k = (I - P_k)r_0, \tag{2.5}$$

where

$$P_k = AS_k(C_k^H AS_k)^{-1}C_k^H.$$

2

We call operator P_k a projection, since $P_k^2 = P_k$. For all $y \in \mathbb{C}^n$ we have

$$P_k y \in AS_k, \text{ and } (I - P_k)v \in C_k^\perp$$

and hence P_k projects onto AS_k orthogonally to C_k. Consequently, (2.5) can be written as

$$r_0 = \underbrace{P_k r_0}_{\in AS_k} + \underbrace{r_k}_{\in C_k^\perp}.$$

If $AS_k = C_k$, then the decomposition is orthogonal and the method is called an *orthogonal projection method*, otherwise ($AS_k \neq C_k$) we call it an *oblique projection method*.

Now we would like to study when the method terminates (in exact arithmetic) with $r_k = 0$. In order to establish termination conditions we consider search spaces S_k with

$$S_1 = \text{span}\{r_0\}, \text{ and } S_1 \subset S_2 \subset S_3 \subset \dots$$

such that these spaces automatically satisfy

$$AS_k = S_k \text{ for some } k.$$

These properties are guaranteed when

$$S_k = \mathcal{K}_k(A, r_0) := \text{span}\{r_0, Ar_0, \dots, A^{k-1}r_0\}, \ k \geq 1,$$

where $\mathcal{K}_k(A, r_0)$ is called the k-th *Krylov subspace* generated by A and r_0.

For any nonsingular $A \in \mathbb{C}^{n\times n}$ and $r_0 \in \mathbb{C}^n \backslash \{0\}$ we have

- There exists a uniquely determined $d = d(A, r_0)$ with $1 \leq d \leq n$ and $\mathcal{K}_1(A, r_0) \subset \dots \subset \mathcal{K}_d(A, r_0) = \mathcal{K}_{d+j}(A, r_0)$ for all $j \geq 1$. This d is called the *grade of* r_0 *with respect to* A. In particular, $\dim(\mathcal{K}_k(A, r_0)) = k$ for $k = 1, \dots, d$.
- If r_0 is of grade d with respect to A, then $A\mathcal{K}_d(A, r_0) = \mathcal{K}_d(A, r_0)$.
- If r_0 is of grade d with respect to A and the method is well-defined at step d with $S_k = \mathcal{K}_k(A, r_0)$, then $r_d = 0$.

Now we have all required setup to introduce the two most widely used Krylov subspace methods:

1. **Conjugate Gradient (CG) method**

   If A is an Hermitian positive definite matrix, $S_k = C_k = \mathcal{K}_k(A, r_0), k = 1, 2, \dots$, then the projection method is well-defined at every step k until it terminates with $r_d = 0$ at step d. It is characterized by the orthogonal property

$$r_k \perp \mathcal{K}_k(A, r_0) \quad \text{OR} \quad x - x_k \perp_A \mathcal{K}_k(A, r_0),$$

and the equivalent optimality property

$$\|x - x_k\|_A = \min_{z \in x_0 + \mathcal{K}_k(A, r_0)} \|x - z\|_A$$

2. **Generalized Minimal Residual (GMRES) method**

   If A is nonsingular, $\mathcal{S}_k = \mathcal{K}_k(A, r_0)$, $\mathcal{C}_k = A\mathcal{S}_k = A\mathcal{K}_k(A, r_0), k = 1, 2, \ldots$, then the projection method is well-defined at every step k until it terminates with $r_d = 0$ at step d. It is characterized by the orthogonal property

$$r_k \perp A\mathcal{K}_k(A, r_0) \quad \text{OR} \quad x - x_k \perp_{A^H A} \mathcal{K}_k(A, r_0),$$

   and the equivalent optimality property

$$\|r_k\|_2 = \min_{z \in x_0 + \mathcal{K}_k(A, r_0)} \|b - Az\|_2$$

In order to implement the above described methods we need to construct bases of $\mathcal{S}_k$ and $\mathcal{C}_k$. The "canonical" basis $r_0, Ar_0, \ldots, A^{k-1}r_0$ of $\mathcal{K}_k(A, r_0)$ should not be used in practical applications, since the corresponding matrix usually is ill-conditioned. Therefore, for numerical reasons, a well-conditioned, at best orthonormal basis of $\mathcal{K}_k(A, r_0)$ is preferred. Such a basis may be generated be the *Arnoldi process*:

Algorithm 1 Arnoldi process

1: $v_1 = r_0 / \|r_0\|$
2: **for** $j = 1, \ldots, k-1$ **do**
3: $\quad \tilde{v}_{j+1} = Av_j - \sum_{l=1}^{j} h_{l,j} v_l$, where $h_{l,j} = (Av_l, v_j)$
4: $\quad h_{j+1,j} = \|\tilde{v}_{j+1}\|$
5: $\quad v_{j+1} = \tilde{v}_{j+1} \, \|h_{j+1,j}\|$
6: **end for**

Note that in line 3 of algorithm 1 we perform a "classical" Gram-Schmidt orthogonalization of Av_k with respect to the previous orthonormal basis vectors $v_1, \ldots, v_k$. There also exists a "modified" version of Gram-Schmidt algorithm. In addition, what is important for practical computations is that the Arnoldi algorithm does not explicitly require operations on matrix A, but only a matrix-vector multiplication needs to be implemented. It is an essential benefit, when A is sparse and, consequently, the product $A \cdot v$ can be computed inexpensively. However, storage and memory requirements grow linearly as the number of iterations k increases, since the Arnoldi algorithm employs a full recurrence for calculating the orthonormal basis of $\mathcal{K}_k(A, r_0)$. Moreover, even for sparse matrices the Arnoldi basis vectors usually are not sparse, which may result in storage problems for large applications with extremely small desired tolerance and, as a consequence, a huge amount of required GMRES iterations to converge.

The convergence will be fast when the condition number of A is close to 1 and eigenvalues of A are in a single "cluster" that is as far as possible from zero, which motivates to apply *preconditioning* and *deflation* techniques to the system $Ax = b$.

2.5. Preconditioning

Preconditioning is the basic technique to improve the robustness and efficiency of iterative solvers for large linear systems of equations by reducing the condition number of A [24]. In general preconditioning means a transformation of the system into an equivalent one which has the same solution, but this solution is much faster computed using iterative methods. Basically, preconditioning consists of a matrix M which must satisfy following requirements:

- M should be nonsingular easily (inexpensively) invertible approximation to A;
- the mapping $x \mapsto M^{-1}x$ should be easy and cheap to perform;
- the eigenvalues of $M^{-1}A$ (or AM^{-1}) should belong to only one region (at best around 1 in the complex plane).

There are three ways to use such a matrix M:

1. Left preconditioner $M^{-1}Ax = M^{-1}b$ is more common and does not require any additional step to calculate x;
2. Right preconditioner $AM^{-1}y = b$, where $x = M^{-1}y$, does not change the residual norm;
3. Split preconditioner $M = LU$ such that $L^{-1}AU^{-1}y = b$ with $x = U^{-1}y$ keeps symmetry property.

It should be noted that all types of preconditioners mentioned above give the same spectrum for the preconditioner operator which is much better clustered compared to the spectrum of A. Also the preconditioned matrix usually has a smaller condition number. Preconditioned matrix $M^{-1}A$ (AM^{-1} or $L^{-1}AU^{-1}$) is not formed explicitly, since it is too expensive and sparsity is lost. Instead, consecutive matrix-vector products are performed with a solution of the transformed linear system. The choice of the preconditioner highly depends on the problem and the computer architecture used for the implementation.

The most popular preconditioners are

- Jacobi (or diagonal scaling) takes $M = diag(A)$ as preconditioner which has low storage requirements and could be used as a first step in combination with other preconditioners;
- Gauss-Seidel is defined by $M = L + D$, where L is strictly lower-triangular part of A and $D = diag(A)$ and gives faster error reduction than Jacobi. By introducing a relaxation parameter ω we get successive over-relaxation (SOR) preconditioner $M = \omega L + D$, which even more improve the convergence rate;
- ILU preconditioner is based on the idea of computing incomplete (sparse approximation of) LU decomposition. The exact $A = LU$ has fill-in, and zero entries in A become non-zero in L and U, but $M = L_{approx}U_{approx}$ (*incomplete* factorization, i.e. $A \approx L_{approx}U_{approx}$) can keep only the fill-in entries above a fixed tolerance.

- Algebraic Multigrid (AMG) preconditioner [25] is extremely robust in terms of algorithmic efficiency preconditioner. Modern commercial reservoir simulators for subsurface fluid flow in porous media unconditionally use AMG preconditioner for the solution pressure equation. However, there are still challenges to address in implementing strongly scalable AMG solver [26]. Klie et al. [27] considered deflation AMG solvers for highly ill-conditioned reservoir simulation problems. However, the manual construction of the deflation vectors is a time consuming process and will most likely lead to suboptimal results [26]. Therefore, the deflation strategy will be considered in the following chapter. The deflation strategy has a lower algebraic complexity and is inexpensive to set up. This strategy in combination with the multiscale may lead to a robust alternative of AMG in general.

3

Deflation Method

3.1. Introduction

Usually eigenvalues of small magnitude and corresponding eigenvectors slow down the convergence or even cause divergence of the iterative method used to solve matrix equation $Ax = b$. For this reason, the *deflation* technique removing the influence of extremal eigenvalues of small magnitude while leaving the remainder eigenvalues unchanged is introduced. There are, basically, two approaches of how harmful eigenvalues may be deflated:

1. the proper projector P is applied as a special (singular) preconditioner;
2. extend utilized Krylov subspace by adding eigenvectors corresponding to some extreme eigenvalues, whereupon the convergence rate is improved due to modified spectrum. This process is called *augmentation*.

But in this work we will only use and concentrate on the first variant, since in the second case desired eigenvectors generally are not easily computed, that forces to use other deflation subspaces. Firstly, the concept of *deflation preconditioning* will be described, and then some methods to construct the deflation vectors will be presented.

3.1.1. Deflation preconditioning

Deflation preconditioning was developed to construct a preconditioner in such a way that it eliminates problematic subspaces influence by removing them from preconditioned operator. For the detailed overview of possible types of deflation preconditioning we refer to [28–31]. But here we consider the specific deflation preconditioner in the form of the projector P introduced in [32].

Let us consider general case with non-symmetric linear system matrix $A \in \mathbb{R}^{n\times n}$ and suppose that the deflation matrix $Z \in \mathbb{R}^{n\times d}$ with d deflation vectors spanning a subspace $\mathcal{Z}$ is given. Then define the matrix $E \in \mathbb{R}^{d\times d}$ as

$$E = Z^T AZ,$$

assuming the existence of E^{-1}, which is computed relatively cheap, since, in general, $d \ll n$. The matrix equation $Ax = b$ is preconditioned by the deflation operator (projector)

$$P_1 = I - AZE^{-1}Z^T,$$

i.e., the initial problem is transformed into the *deflated* system,

$$Ax = b \iff P_1 Ax = P_1 b.$$

One should observe that if the columns of Z (the deflation vectors) form an invariant subspace of A, then $P_1 A$ has d zero eigenvalues, which makes $P_1 A$ singular. In particular, when Z contains d exact eigenvectors of A, then applying P_1 to A *deflates* these d corresponding eigenvalues to zero.

Then second deflation operator (projector)

$$P_2 = I - ZE^{-1}Z^T A$$

is used to obtain the solution x as follows. Since

$$x = (I - P_2)x + P_2 x$$

and because

$$(I - P_2)x = ZE^{-1}Z^T b$$

may be easily and directly calculated, the only computation of $P_2 x$ is required, which can be done by solving the deflated system using the identity $AP_2 = P_1 A$ and pre-multiplying by P_2:

$$AP_2\tilde{x} = P_1 A\tilde{x} = P_1 b. \tag{3.1}$$

Since $P_1 A$ is singular, $\tilde{x}$ contains arbitrary components in its null space, and, hence, the solution of the deflated system differs from the solution of the original system. However, due to the fact that $P_2\tilde{x} = P_2 x$ and $P_2 x$ has no components in the null space of $P_1 A$, the projected solution $P_2\tilde{x}$ is unique.

As a result, we have that reconstructed solution of the original system looks like

$$x = ZE^{-1}Z^T b + P_2\tilde{x},$$

where the right part of (3.1) needs to be solved to find $\tilde{x}$.

3.2. Preconditioner and GMRES

In this section we will present the basic GMRES algorithm (thorough mathematical formulation is explained in the previous chapter) and describe its preconditioned version.

3.2.1. Restarted GMRES

As we know, the full GMRES always terminates after at most n iterations and the underlying Krylov subspace dimension grows up to n as maximum [33]. On the other hand, for a quite large n it is worthwhile to restrict the Krylov subspace

dimension to a fixed value m ($m \ll n$), after reaching this value the method restarts the Arnoldi process using the last approximation x_m as a new initial guess. Restarted algorithm 2 is usually denoted by GMRES(m):

Algorithm 2 GMRES(m)

choose ε as the tolerance for the residual norm;
choose initial guess x_0 and dimension m of the Krylov subspace;
ok := FALSE;
$e_1 := [1, 0, \dots, 0]^T$;
while *ok* is FALSE **do**
    $r_0 = b - Ax_0$;
    $\beta = \|r_0\|$;
    $v_1 := r_0/\beta$;
    for $j = 1, \dots, m$ **do**
        $w := Av_j$;
        for $i = 1, \dots, j$ **do**
            $h_{ij} := v_i^T p$;
            $w = w - h_{ij} v_i$;
        end for
        $h_{j+1,j} := \|p\|$;
        $v_{j+1} := p/h_{j+1,j}$;
        $s := \|b - Ax_j\|$;
        if $s < \varepsilon$ **then**
            solve $\min_{y_j \in \mathbb{R}^j} \|\beta e_1 - H_{j+1,j} y_j\|$;
            $x_j := x_0 + V_j x_j$;
            ok := TRUE;
            break;
        end if
    end for
    solve $\min_{y_m \in \mathbb{R}^m} \|\beta e_1 - H_{m+1,m} y_m\|$;
    $x_m := x_0 + V_m x_m$;
    if $\|b - Ax_m\| < \varepsilon$ **then**
        ok := TRUE;
    else
        $x_0 = x_m$;
    end if
end while

As we will explain in the next section, $H_{k+1,k}$ is an upper Hessenberg matrix for $k = 1, \dots, m$ satisfying the fundamental relation

$$AV_k = V_{k+1} H_{k+1,k}.$$

The GMRES methods computes the approximation x_k by solving the least-squares problem, which may be achieved using QR factorization (decomposition into a prod-

uct of orthogonal and upper triangular matrices) of $H_{k+1,k}$ in conjunction with Givens rotations.

3.2.2. Preconditioned GMRES

Here we present a couple of improvements for a standard GMRES method, specifically preconditioned variants [34]:

- GMRES with right preconditioning (right version is preferred, since it guarantees that the residual norm is *non-increasing* for increasing iterations):

Algorithm 3 PRECGMRES(m)

1. **Start and Initialization:**
   Choose initial guess x_0, dimension m of the Krylov subspace, preconditioner M and the residual norm tolerance ε;
2. **Arnoldi process:**
   a) Compute $r_0 = b - Ax_0,\ \beta = \|r_0\|,\ v_1 = r_0/\beta$;
   b) **for** $j = 1, \dots, m$ **do**
      - Compute $z_j := M^{-1}v_j,\ w := Az_j$;
      - **for** $i = 1, \dots, j$ **do**
        - $h_{i,j} := (w, v_i)$;
        - $w = w - h_{i,j}v_i$;
      - Compute $h_{j+1,j} = \|w\|,\ v_{j+1} = w/h_{j+1,j}$;

   c) Define $V_m = [v_1, \dots, v_m]$;
3. **Construct the approximate solution:**
   a) solve $\min_{y_m \in \mathbb{R}^m} \|\beta e_1 - H_{m+1,m}y_m\|$;
   b) $x_m := x_0 + M^{-1}V_m y_m$;
4. **Restart:**
   if $\|b - Ax_m\| < \varepsilon$
   Stop the algorithm, since the converged solution is found;
   else
   $x_0 := x_m$;
   Go to step 2;

The solution in the step 3. b) of the algorithm 3 is obtained by calculating a linear combination of the preconditioned vectors,

$$z_i = M^{-1}v_i, \quad i = 1, \dots, m.$$

Note that we do not need to save all these vectors, since they are constructed using the same preconditioner M. Furthermore, we only have to apply M^{-1}

to $V_m x_m$. Now we are interested in varying of the preconditioning matrix at every step of the algorithm, i.e. z_j would be defined by

$$z_j = M_j^{-1} v_j,$$

and we save them to update x_m in the step 3, that leads to the "flexible" modification of the previous algorithm.

- GMRES with variable preconditioning (flexible scheme):

Algorithm 4 FGMRES(m)

1. **Start and Initialization:**
   Choose initial guess x_0, dimension m of the Krylov subspace, preconditioners M_j and the residual norm tolerance ε;
2. **Arnoldi process:**
   a) Compute $r_0 = b - Ax_0,\ \beta = \|r_0\|,\ v_1 = r_0/\beta$;
   b) **for** $j = 1, \dots, m$ **do**
      – Compute $z_j := M_j^{-1} v_j,\ w := Az_j$;
      – **for** $i = 1, \dots, j$ **do**
         • $h_{i,j} := (w, v_i)$;
         • $w = w - h_{i,j} v_i$;
      – Compute $h_{j+1,j} = \|w\|,\ v_{j+1} = w/h_{j+1,j}$;
   c) Define $Z_m = [z_1, \dots, z_m]$;
3. **Construct the approximate solution:**
   a) solve $\min_{y_m \in \mathbb{R}^m} \|\beta e_1 - H_{m+1,m} y_m\|$;
   b) $x_m := x_0 + Z_m y_m$;
4. **Restart:**
   if $\|b - Ax_m\| < \varepsilon$
   Stop the algorithm, since the converged solution is found;
   else
   $x_0 := x_m$;
   Go to step 2;

3.3. Construction of deflation vectors

Here we consider the most common techniques to construct the deflation matrix.

3.3.1. Ritz deflation

Due to the size and complexity of the matrix under consideration instead of the exact values approximated eigenvectors and eigenvalues are calculated using

various numerical methods. A widely used technique for accomplishing this is called the *Rayleigh-Ritz procedure* in combination with *Arnoldi algorithm*.

Rayleigh-Ritz procedure. The Rayleigh-Ritz procedure is an iterative projection method for the general eigenvalue problem

$$Ax = \lambda x, \quad A \in \mathbb{C}^{n\times n}.$$

It is based on the *Arnoldi iteration* and starts with a unit norm vector $v_1 \in \mathbb{C}$ and yields after k steps a decomposition of the form

$$AV_k = V_{k+1}H_{k+1,k} = V_kH_{k,k} + h_{k+1,k}v_{k+1}e_k^T,$$

where e_k is the k-th canonical vector in $\mathbb{R}^k$.

There are several mathematically equivalent variants of this algorithm, in particular "classical" and "modified" Gram-Schmidt variants. In each case the columns of $V_k \in \mathbb{C}^{n\times k}$ form an orthonormal basis of the k-th Krylov subspace generated by A and v_1, i.e.,

$$\mathcal{K}_k(A, v_1) = \text{span}\{v_1, Av_1, ..., A^{k-1}v_1\},$$

where the final subspace $\mathcal{K}_m(A, v_1)$ is invariant under A, and $H_{k+1,k} \in \mathbb{C}^{k+1,k}$ is an upper Hessenberg matrix with positive subdiagonal elements, so that $H_{k+1,k}$ is unreduced. The Arnoldi algorithm terminates (in exact arithmetic) in step $k = m$ with $v_{m+1} = 0$ and $h_{m+1} = 0$ if and only if v_1 is of grade m with respect to A, i.e., m is the smallest integer such that vectors $v, Av, ..., A^{m-1}v$ are linearly independent, but $v, Av, ..., A^m v$ are linearly dependent. In this step we have

$$AV_m = V_mH_{m,m}$$

so that the columns of V_m span the A-invariant subspace $\mathcal{K}_m(A, v_1)$ and the eigenvalues of $H_{m,m}$ are eigenvalues of A.

In the following we assume that $1 \le k \le m$. The idea is to use eigenpairs of the intermediate Hessenberg matrices $H_{k,k}$, $k = 1, 2, 3, ...$, for the approximation of eigenpairs of A. If $(\hat{\lambda}_j^{(k)}, z_j^{(k)})$ is an eigenpair of $H_{k,k}$, then $\hat{\lambda}_j^{(k)}$ is called a *Ritz value* of A with respect to the subspace $\mathcal{K}_k(A, v_1)$ and $\hat{x}_j^{(k)} := V_k z_j^{(k)}$ is the corresponding *Ritz vector*. Now the residual

$$r_j^{(k)} := A\hat{x}_j^{(k)} - \hat{\lambda}_j^{(k)}\hat{x}_j^{(k)}$$

satisfies

$$V_k^H r_j^{(k)} = V_k^H(A\hat{x}_j^{(k)} - \hat{\lambda}_j^{(k)}\hat{x}_j^{(k)}) = H_{k,k}z_j^{(k)} - \hat{\lambda}_j^{(k)}z_j^{(k)} = 0,$$

since $V_k^H AV_k = H_{k,k}$ is a unitary similarity transformation with $V_k^H V_k = I$, and hence each Ritz pair $(\hat{\lambda}_j^{(k)}, \hat{x}_j^{(k)})$ in the Arnoldi method satisfies two conditions

$$\hat{x}_j^{(k)} \in \mathcal{K}_k(A, v_1), \quad r_j^{(k)} \perp \mathcal{K}_k(A, v_1),$$

where the second condition is called *Galerkin orthogonal projection problem*.

Seeing that Ritz pairs tend to approximate eigenvalues and eigenvectors of A, d approximated eigenvectors associated with the d smallest Ritz values may be taken as the columns of matrix Z.

However, according to [35] *Harmonic Ritz vectors* is better option, if the approximation of extreme eigenvalues is required.

3.3.2. Harmonic Ritz deflation

Imposing the *Petrov-Galerkin orthogonality conditions* on the residual

$$r_j^{(k)} \perp A\mathcal{K}_k(A, v_1),$$

Harmonic Ritz vectors are obtained by solving the equation

$$(AV_k)^H r_j^{(k)} = (AV_k)^H (A\hat{x}_j^{(k)} - \hat{\lambda}_j^{(k)} \hat{x}_j^{(k)}) = 0,$$

which can be overwritten either as the harmonic eigenvalue problem or the generalized eigenvalue problem. The latter is preferable to be used for solving, since it does not require to store the Hessenberg matrix $H_{k,k}$.

It is important to remark that the required number of Harmonic Ritz vectors to be computed for the deflation subspace Z is much smaller than dimension of matrix A, which makes this procedure relatively inexpensive.

3.3.3. Physics-based deflation

Subdomain deflation. This approach is closely related to the domain decomposition and multigrid methods, where the discretized computational domain Ω is divided into non-overlapping subdomains Ω_i, which are accumulated into single cells using projector operators P_1 and P_2. Deflation vectors z_j forming columns of the deflation matrix Z are piecewise-constant, orthogonal and disjoint and are defined as

$$(z_j)_i = \begin{cases} 1, & x_i \in \Omega_i, \\ 0, & x_i \in \Omega \setminus \Omega_i, \end{cases}$$

where x_i is a grid point in the discretized domain Ω_i.

Levelset deflation. Levelset deflation takes into account geometry and geological properties of the reservoir and decomposes it into different parts with similar characteristics, provided in matrix coefficients, in order to guarantee that eigenvectors associated with extreme eigenvalues are well approximated.

Subdomain-level deflation. This method of choosing deflation vectors combines two previously described ideas. First of all, initial domain is decomposed into subdomains using certain criteria. Thereafter, levelset deflation is applied for each subdomain, reflecting jumps between the high permeability and low permeability nodes in case of reservoir simulation.

A simple example of physics-based deflation for a 4×4 grid is given in Figure 3.1. In each node (black and red squares) the values correspond to the values in the first deflation vector, whereas dash and dotted lines indicate borders between

computational domains and separate subdomains with high contrasts in the PDE coefficients.

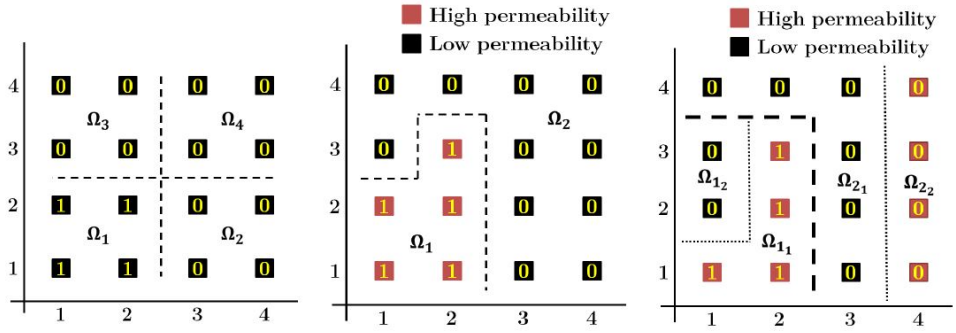


Figure 3.1: Subdomain, levelset and subdomain-levelset deflation [36].

There are some other options of constructing deflations vectors available in the literature [37].

4

Multiscale Methods

4.1. Introduction

Key concept. Although standard preconditioners improve the convergence rate, after a few iterations they produce smooth low frequency errors, corresponding to extreme eigenvalues, which are reduced very slowly. One of the most effective methods to deal with this problem is a multigrid idea comprising following steps:

1. Transformation of the initial system discretized on a fine grid into a coarse grid using a restriction operator;
2. Solving the problem on a coarse grid (coarse-grid correction);
3. Interpolation of the correction to the fine grid using a prolongation operator.

On a coarse grid low frequency errors behave like high frequency errors, and hence are easily removable, but they also cause new high frequency errors on a fine grid, which should be treated using smoothing (relaxation) operators after the coarse grid procedure is completed.

4.1.1. Upscaling Methods

Upscaling, also called homogenization, methods have been widely used to solve large scale subsurface flow problems. The main idea of upscaling is to compute effective properties on a coarse scale by some preliminary steps, and then solve the coarse-scale equations. The governing equations for subsurface flow can be split into the flow problem (pressure equation) and the transport problem (saturation equations). For detailed surveys of these we refer to [38].

In spite of the fact that multiscale methods are similar to upscaling techniques, there also exist some significant differences. The purpose of multiscale approach is to accurately approximate the fine-scale solutions, whereas upscaling modelling focuses on obtaining the approximation to the coarse-scale solutions. In other

words, the multiscale approach is a local-global concept based on the construction of global solutions providing global coarse scale information, while the local solutions are used to obtain fine scale information. Moreover, in multiscale methods the coarse-scale system is obtained numerically and dynamically using current fine-scale information rather than coarse-scale quantities computed beforehand. Since no prior flow scenarios are assumed, multiscale modelling is process independent. Therefore multiscale methods are generally suitable for any sort of reservoir simulation. In addition, different from multiphase upscaling techniques, in multiscale modelling one avoids the inconsistency and non-physical features on the coarse scale in the process of coupling local and global problems [39].

4.1.2. Dual-Grid Methods

Generally, in the most trivial scenario multiscale methods utilize only two grids to provide an efficient numerical algorithm for a reservoir simulation. In some early papers authors proposed a few methods, where both coarse- and fine-scale flow information are obtained without directly solving the full fine-scale problem. There is a family of such methods called ”dual-grid” methods, which can be easily distinguished from multiscale methods due to some substantial differences between them.

However, there is one common drawback for all discussed dual-grid methods: the fine- and coarse-scale problems are not coupled properly. In particular, any improvements in the local fine-scale solutions have a quite limited impact on the coarse-scale solution, since the latter is almost independent of the local fine-scale solution. On the contrary, the coarse-scale approximations have a large influence on the fine-scale solutions, i.e., degradation of the coarse-scale solutions leads to the highly bad results on the fine scale. In addition, dual-grid methods demonstrate extremely poor performance for highly heterogeneous permeability fields [40].

4.1.3. Multiscale Finite Element Method (MsFEM)

Recently developed by Hou and Wu [41] multiscale method for the flow problem has much stronger coupling between the fine and coarse scales. The main idea of the MsFEM is to integrate the fine-scale information into a coarse-scale system via special basis functions. Unlike traditional finite-element methods, these basis functions, representing the fine-scale information, of the MsFEM are usually obtained as numerical solutions of localized boundary-value problems that reflects the fine-scale properties. The MsFEM solves the underlying elliptic problem

$$\nabla \cdot (\lambda \cdot \nabla u) = f, \quad \text{on} \quad \Omega.$$

In terms of classical finite-element approach, the weighting functions space $\mathcal{V}^h$ is spanned by the basis functions as

$$\mathcal{V}^h = \text{span}\{\phi_K^i;\ i = 1, ..., n;\ K \in \mathcal{K}^h \subset H_0^1(\Omega)\},$$

where ϕ_K^i denotes the basis function associated with node i in element K, n is the number of nodes in element K, $\mathcal{K}^h$ is an element partition of domain Ω and $H_0^1(\Omega)$

is the Hilbert functional space defined on Ω. These basis functions are solved locally in each element with reduced boundary condition, such that

$$\begin{cases} \nabla \cdot (\lambda \cdot \nabla \phi_K^i) = 0, & \text{in } \Omega_k \\ \nabla_{\|} \cdot (\lambda \cdot \nabla \phi_K^i)_{\|} = 0, & \text{on } \partial\Omega_k \\ \phi_K^i(x_j) = \delta_{ij} & \forall j \in 1, \ldots, n, \end{cases}$$

where the subscript $\|$ indicates the vector (or operator) projected along the tangential direction of the element boundary $\partial\Omega_k$, and Ω_k is the domain of element K, superscript i denotes one node of that element, and x_j represents the coordinate of node j. After locally solving equation above, the coarse-scale system can be constructed by the basis functions, in a similar way as conventional finite-element methods. Once the coarse-scale solution is obtained, the fine-scale approximation can be calculated using the basis functions and the coarse-scale solution at node i, i.e. u_i, as

$$u(x) = \sum_{i=1}^{n} \phi_K^i(x) u_i, \quad \text{if } x \in \Omega_k.$$

It was also pointed out that large errors occur due to resonance when the scale of oscillations in the fine-scale coefficient is close to the scale of the grid. In addition, the resonance error can be eliminated by improving the boundary conditions of the basis functions. In order to tackle this issue, they proposed an oversampling technique that imposes the reduced boundary condition on a sampled domain, which is larger than the coarse element and then the basis functions are computed on that sampled domain. They show that a good choice of the boundary condition determines that the local characteristics are well sampled into the basis functions and can significantly improve the accuracy of multiscale methods. The investigations reported in [42] demonstrate that the MsFEM has a good convergence rate for the elliptic equation with highly oscillatory coefficients.

The major drawback of this method for reservoir simulation is that it does not provide a mass conservative velocity field, which is a crucial element in achieving accurate solutions of the transport problem, since existing multiscale methods use a sequential strategy. Later, Chen and Hou [43] presented a multiscale formulation based on a mixed finite-element method and demonstrated the importance of a locally conservative algorithm for transport simulation. This family of Mixed Multiscale Finite-Element Methods (MMsFEM) [44] can offer mass conservative velocity fields for both fine- and coarse-scale grids.

4.1.4. Multiscale Volume Element Method (MsFVM)

In order to obtain approximate solutions that are strictly locally mass conservative on the fine scale, the Multiscale Finite Volume Method (MsFVM) [45] was developed. Compared with the finite element formulation, the MsFVM yields mass conservative solutions using a smaller number of degrees of freedom.

In MsFVM basis functions are utilized to capture the fine-scale information, which are identical to those of the MsFEM. The approximate pressure solution obtained

from MsFVM guarantees local mass conservation, which can be used to reconstruct the velocity field by solving local elliptic problems on primal coarse grids with Neumann boundary conditions. Recent developments of the MsFVM include incorporating the effects of compressibility, gravity and capillary, complex wells, faults, fractures, three-phase flow using the black-oil model and compositional displacements [46]. Moreover, for efficiency's sake the method has been improved by adaptive computation of the basis functions for multiphase, time-dependent displacement problems [47]. Zhou and Tchelepi [48] proposed the Operator-Based Multiscale Method (OBMM), which employs constructed in an algebraic manner prolongation and restriction operators in order to capture the fine-scale information. Applying these two operators to the original fine-scale mass balance equation leads to a coarse-scale system, which can be solved for the coarse-scale pressure field. Then, a fine-scale approximation can be obtained by prolongation of the coarse-scale solution. This algebraic formulation reduces computational costs for problems defined on unstructured grids and allows to incorporate complex reservoir physics for easy integration of the method into existing solvers.

In spite of the good performance for a large number of test cases, the original MsFVM deteriorates for channelized permeability fields and large anisotropy [49]. As in all multiscale methods, in the MsFVM framework, the coarse system is obtained by using basis functions, which are numerically computed on local domains (with assumed boundary conditions). The accuracy of the MsFVM is therefore strongly dependent on the quality of the local boundary conditions. For some very challenging problems, the method fails to provide accurate solutions. To resolve these limitations, the iterative MsFVM (i-MsFVM) was introduced by Hajibeygi et al. [50], where the MsFVM solution is iteratively improved by locally computed correction functions together with a fine-scale smoother. The i-MsFV method converges to the fine-scale reference solution, and a conservative velocity field can be constructed after any iteration level. The i-MsFVM reduced the MsFVM errors for many challenging problems; however, for highly heterogeneous and anisotropic cases, it did not perform satisfactorily. The weak MsFVM coarse-scale operator is the main reason of this issue, which has been overcome by development of the Two-Stage Algebraic Multiscale Linear Solver (TAMS) [51]. To provide the conservative solution, when the full procedure is done, the MsFVM operator is employed as the last step in the TAMS algorithm. Having a two steps structure, high- and low-frequency errors are handled by smoother and multiscale preconditioner in the local and global stages, respectively. Obviously, combination of the stages allows to resolve all kinds of frequency errors. Also it is important to note that TAMS is appropriate for both finite volume and finite element schemes and does not contain correction function like other MsFVMs.

5

Multiscale Restriction-Smoothed Basis Method (MsRSB)

5.1. Introduction

In the previous chapter a number of various multiscale methods were introduced as an improvement of classical upscaling technique in order to accelerate reservoir simulation and provide better scaling for the Poisson-type equations modelling flow in porous media. These methods relies on the construction of a set of prolongation (basis functions) and restriction operators which map between unknowns corresponding to fine-grid cells honouring the petrophysical properties of the geological model and coarse-grid variables used for a dynamic simulation. One needs to solve local flow equations to compute prolongation operators numerically, which are used to construct a reduced coarse-scale system describing the macro-scale displacement arising due to global forces [52]. Unlike other methods, basis functions are obtained by *restricted smoothing*: starting from a constant (i.e., equal 1 in the coarse block cells and 0 otherwise), prolongation operators are computed iteratively on the fine-scale grid in a such way as to be consistent with the local properties of the differential operators. It was shown that the multiscale restriction-smoothed basis (MsRSB) method is as robust and accurate as existing multiscale techniques and even outperforms them for some reservoir simulation test cases due to its three benefits [53]:

1. Both fine- and coarse-scale grids have unstructured topology and general polyhedral geometry. It makes method applicable for complex models involving heterogeneous domains with long coherent structures with high contrasts, since the coarse partition is adapted to geological features of the reservoir;

2. It is not required to expensively recompute local basis functions to account for transient behaviour: only a few extra steps needed to update existing basis functions;
3. Method is suitable for any flow problem where pressure equation may be isolated.

5.2. Multiscale formulation

Like in all multiscale techniques, MsRSB method starts from a fine grid $\{\Omega_i\}_{i=1}^n$, which is partitioned into the coarse grid $\{\bar{\Omega}_j\}_{j=1}^m$ $(m < n)$, where each fine cell is contained in only one coarse block. Then, numerical prolongation P and restriction R operators are defined as mappings between fine cells and coarse blocks and represented as sparse matrices of sizes $n \times m$ and $m \times n$, respectively:

$$P : \{\bar{\Omega}_j\}_{j=1}^m \to \{\Omega_i\}_{i=1}^n,$$
$$R : \{\Omega_i\}_{i=1}^n \to \{\bar{\Omega}_j\}_{j=1}^m.$$

Let us denote by p_f and p_c pressure values computed on the fine and coarse grids, which are connected via the prolongation operator,

$$p_f = Pp_c. \tag{5.1}$$

Generally, pressure solution on the coarse grid does not coincide with the exact solution, but can be an accurate approximation, which is calculated more efficiently than solving the initial problem on a fine grid.

Inserting (5.1) into the initial linear system $Ap = q$ and applying restriction operator R, we derive a linear system for pressure on the coarse grid,

$$R(A(Pp_c)) = (RAP)p_c = A_c p_c = Rq = q_c.$$

Basically, there two main ways to determine the restriction operator, either a Galerkin operator or control volume summation operator, i.e.,

$$R_G = P^T,$$
$$(R_{CV})_{ij} = \begin{cases} 1, & \text{if } x_j \in \bar{\Omega}_i, \\ 0, & \text{otherwise.} \end{cases}$$

5.2.1. Construction of basis functions

In most multiscale methods basis functions forming prolongation operators are obtained by numerically solving local flow problems, which are defined similar to the global problem with modified boundary conditions and account for corresponding local features. However, in MsRSB method basis functions are constructed iteratively, and here we will describe this process.

For each coarse block basis functions are initially defined as characteristic functions,

$$P_{ij}^0 = \begin{cases} 1, & \text{if fine cell } i \text{ belongs to the coarse block } j, \\ 0, & \text{otherwise,} \end{cases}$$

Obviously, such a choice is made for convenience sake, namely constant functions are trivial to construct and handle, and they also provide partition of unity. We then define a local smoothing repeated iteration,

$$P_j^{n+1} = (I - \omega D^{-1} A) P_j^n, \tag{5.2}$$

where j denotes the column index of the prolongation operator, $\omega \in (0, 1]$ is a relaxation factor and D is a diagonal matrix that contains the diagonal entries of (weakly) diagonally dominant system matrix A. Note that $\omega = 2/3$ is the optimal relaxation parameter for Jacobi's method applied to Poisson's equation with constant coefficients. Better choices of ω will speed up convergence of the basis construction, but are not necessarily obvious for general problems. By iterating on the prolongation operator, we try to reduce $\|AP\|_1$ as much as possible to achieve relatively smooth residual error. In general, (5.2) determines an increment for each basis function based on the local error and amends this increment until a convergence criterion is satisfied as well as to guarantee that the support of the basis functions is inside already defined acceptable regions. This update is also used to determine convergence of the basis construction procedure. Since in every iteration the cell value is modified based on the topological neighbours, the support of the basis functions will gradually increase step-by-step and eventually cover the whole computational domain. These support regions are needed to restrict the basis functions expansion outside of the coarse grid block.

5.3. Iterative multiscale formulation

An utilization of the prolongation operator (iteratively constructed or even defined by piecing together localized flow solutions) has a negative influences on the solution, namely local high-frequency errors show up in the fine-scale approximation. This is issue is handled by applying a smoother, which can be combined with the multiscale solve to formulate an iterative solver consisting of two steps:

1. multiscale increment is computed iteratively;
2. a few smoother operations are completed to limit the error of the approximation obtained in step 1.

The whole procedure eventually reduces the fine-scale residual to zero [54].

To determine the full iterative scheme let us define the defect d at each multiscale iteration k as

$$d^k = b - Ax^k,$$

where x^k is the approximation of the pressure solution (at step k), A is a given coefficient matrix and b is a right-hand side.

Then, having initial guess zero we apply smoother S to the defect,

$$y^k = S(d^k).$$

Hence, the next approximation in the iterative process can be written as a sum of the current solution, coarse correction (ensures that the coarse-scale conservative

property is not removed) and smoothed update, i.e.

$$x^{k+1} = x^k + P\left(A_c^{-1} R\left(d^k - Ay^k\right)\right) + y^k.$$

Here one pass of the smoother along with the coarse correction is called a *multiscale cycle*.

Note that the smoother S should be inexpensive for the updates, hence the incomplete LU-factorization with zero fill-in (ILU(0)) is usually used, but a few Jacobi iterations may be applied for problems where only some local error needs to be reduced.

Finally, it is important to note there are some recent developments related to meshless multiscale methods [55] which can be considered as alternative to the above multiscale methods.

6

Adaptive Deflated Multiscale Solvers (ADMS)

6.1. Motivation

Existing multiscale solvers use a sequence of aggressive restriction, coarse-grid correction and prolongation operators to handle low-frequency modes on the coarse grid. High-frequency errors are resolved by employing a smoother on the fine grid. Deflation preconditioning improves matrix properties, i.e., damps slowly varying errors, corresponding to extreme eigenvalues, in the linear solver residuals. Various *Adapted Deflated Multiscale Solvers* are proposed in order to detect the low-frequency modes instead of relying on the residual map and complement today's stet-of-the-art advanced iterative multiscale strategies.

6.2. Various forms of ADMS

6.2.1. Fully ADMS

Applying the restriction operator R to the preconditioned deflated system on the fine grid

$$P_1 A^f p^f = P_1 b^f$$

and expressing the pressure on the fine grid as an approximation on the coarse grid prolonged to the fine grid, we obtain the following formulation of the *fully ADMS (F-ADMS)*:

$$\begin{aligned} P_1 A^f p^f = P_1 b^f \quad &\Leftrightarrow \quad R P_1 A^f P \hat{p}^c = R P_1 b^f, \text{ where } P\hat{p}^c = p^f \\ &\Rightarrow \quad \hat{p}^c = (R P_1 A^f P)^{-1} R P_1 b^f \\ &\Rightarrow \quad p^f \approx Z E^{-1} Z^T b^f + P_2 P \hat{p}^c = \underbrace{[Z E^{-1} Z^T + P_2 P (R P_1 A^f P)^{-1} R P_1]}_{M^{-1}_{F-ADMS}} b^f \end{aligned}$$

6.2.2. Decoupled ADMS

The simplest way to utilize the multiscale solver along with the deflation method is to use the operator M_{ASM}^{-1} based on Additive Schwarz Method with deflation correction [56]:

$$\mathrm{M}_{ASM}^{-1} = ZE^{-1}Z^T + \sum R_i^T \left(R_i A R_i^T\right) R_i,$$

where R_i is the restriction operator to the overlapping domain.

In light of the structure of the operator M_{ASM}^{-1}, it is possible to use the *decoupled ADMS (D-ADMS)* constructed as follows

$$p^f \approx \underbrace{\left[ZE^{-1}Z^T + P(RA^fP)^{-1}R\right]}_{M_{D-ADMS}^{-1}} b^f$$

6.2.3. Mixed ADMS

Mixed ADMS (M-ADMS) employs an enriched set of basis functions to map between fine and coarse scales. This extended set involves the conventional multiscale local basis functions and globally constructed deflation vectors. Hence, the global prolongation operator P is constructed such that it consists of original prolongation operator and deflation operator, whereas there is only one option for the restriction operator R:

$$\hat{P} = [P; Z], \quad R = \hat{P}^T.$$

Consequently, the solution on the fine grid is defined as

$$p^f \approx \underbrace{\hat{P}\left(\hat{R}A^f\hat{P}\right)^{-1}\hat{R}}_{M_{M-ADMS}^{-1}} b^f.$$

7

Results

In this chapter we will present the description of test cases, discuss the obtained results of numerical experiments and draw corresponding conclusions. Basically, we focus on solving the pressure equation (1.8) in the form of system of linear equations (1.15) resulting from the discretization and linearization of the governing PDEs of subsurface flow. For this purpose we use various ADMS accelerated by the right-preconditioned GMRES, where the preconditioner consists of deflation matrix (and operators in case of Fully ADMS) and/or multiscale mapping operators, and compare the required number of iterations for convergence among them.

7.1. ADMS parameters

Deflation. To construct the deflation matrix we use techniques described in chapter 3.

Smoother and Preconditioner. *Jacobi* and *ILU(0)* (see section 2.5) are selected as smoother and preconditioner in case where only deflation is used (without any multiscale solver).

Grid partitioning. To formulate the problem on the coarse scale, prolongation and restriction operators (used for deflation matrix construction and multiscale basis functions generation) are obtained via the MsRSB method (see chapter 5), while the multigrid transformation can be performed in two ways:

- traditional domain subdivision into equal rectangles or squares;
- **METIS partitioning.** If we deal with the highly heterogeneous computational domain, which contains regions with large jumps in permeability coefficients, then the software package *METIS* [57] may be used to obtain coarse mesh partitioning honouring these discontinuities.

Figure 7.1: Subdomain partitioning into 16 subdomains using METIS [56].

Coarse linear solver. For solving the coarse-scale problem we utilize built-in MATLAB iterative solvers (sections 3.2):

- GMRES;
- BiConjugate Gradients stabilized method (BiCGSTAB).

Passing all necessary parameters to embedded MRST routines, we extract the linear system matrix A and corresponding right-hand side vector b to solve the pressure equation using our developed methods. Since the deflation matrix along with the multiscale mapping operators can be constructed in a few different ways and comprise of changing number of vectors, we describe certain options used in ADMS for each model problem separately.

7.2. Test cases

The three model problems that used to test ADMS are:

- "Islands" (high-permeability inclusions);
- Fractured reservoir;
- SPE10 layer [58];

For each scenario we will provide a detailed specification, that contains physical parameters of the system and the iterative solver data including standard GMRES input as well as the deflation matrix, multiscale basis functions and function handles for the smoother and coarse level linear solver if needed. Considered test cases are good examples of real reservoirs with highly heterogeneous porous media.

7.2.1. "Islands" model problem

Here we consider a 100×100 Cartesian grid, which has a physical size of 10×10 meters squared, with 64 high-permeability islands located symmetrically with respect to the centre of the domain (see Fig. 7.2):

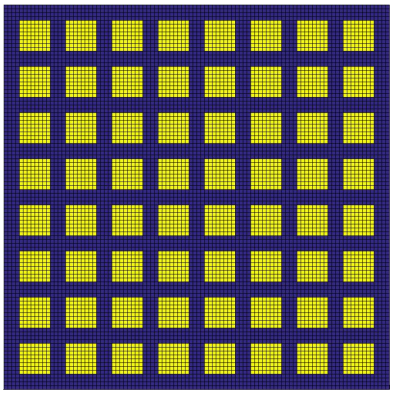


Figure 7.2: 2D example with 64 high heterogeneities islands.

Then we initialize an incompressible single phase fluid model of viscosity $1\ cP$ and density $1014\ kg/m^3$ (typical water properties). After that we impose flux and pressure boundary conditions by specifying a Neumann condition with total inflow of $5000\ m^3/day$ on the left side and a Dirichlet condition with fixed pressure of $50\ bar$ on the right side. To drive the flow, we use a fluid source at the south-west corner and a fluid sink at the north-east corner of the model. More precisely, we set the source terms such that a unit time corresponds to the injection of one pore volume of fluids. As the last step, we establish the initial state with a pressure value of zero and a unit fluid saturation, that completes setup of the model, which leads to a 10000×10000 matrix with 49600 non-zero entries and a diagonal sparsity pattern:

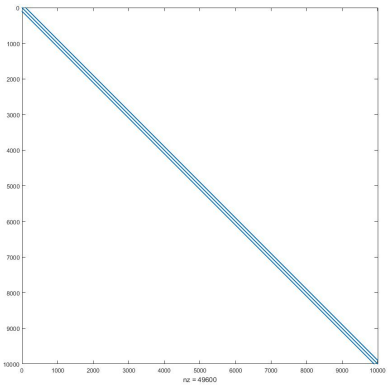


Figure 7.3: "Islands" example matrix sparsity pattern.

Deflation. For method validation the following deflation matrices are used:

1. Z_1 - *physics-based deflation*: contains 64 vectors of the size 10000, where each vector represents only one high-permeability island, specifically 1 in the corresponding island coordinates and 0 otherwise;
2. Z_2 - *physics-based deflation*: similar approach as in Z_1, but now 0 in the island coordinates and 1 otherwise;
3. Z_{12} - *physics-based deflation*: consists of Z_1 and the first vector from Z_2;
4. Z_{21} - *physics-based deflation*: consists of Z_2 and the first vector from Z_1;
5. Z_{ms} - *multiscale basis functions*: contains multiscale basis functions as columns;
6. Z_{db} - *domain-based*: similar to the *physics-based deflation* approach, but here we put 1 into all cells, which belong to the certain subdomain, and 0 into all other cells;
7. Z_{mix} - *mixed deflation*: consists of vectors obtained from different approaches (e.g. physics-based deflation and multiscale basis functions);
8. Z_t - *theoretical*: consists of determined beforehand number of the exact eigenvectors corresponding to the smallest magnitude eigenvalues.

In this scenario standard GMRES method (without deflation and multiscale complements) requires 500 iterations to solve the pressure equation with desired tolerance 10^{-7}. As regards ADMS which for sure outperform GMRES, we present results for the right-preconditioned version, then for decoupled, fully and mixed modifications. Note that in all following tables the number in brackets indicates number of vectors in the corresponding matrix.

Right preconditioner.

Here we consider the constructed preconditioner as a classical right preconditioner for GMRES.

As a benchmark, i.e. the most efficient way to deflate extreme eigenvalues, we examine influence of the deflation matrix Z_t, which allows method to converge with the computed number of iterations (see table 7.1):

deflation matrix	no. iterations
$Z_t(16)$	181
$Z_t(32)$	142
$Z_t(64)$	57
$Z_t(65)$	51

Table 7.1: Linear solver iterations using Z_t

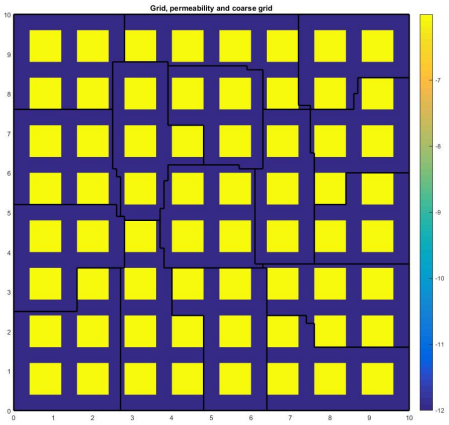
Now we would like to determine the best option for the deflation matrix, which can be easily and cheaply constructed.

The required number of iterations for the restarted GMRES, right-preconditioned only by physics-based deflation, are shown in the table 7.2:

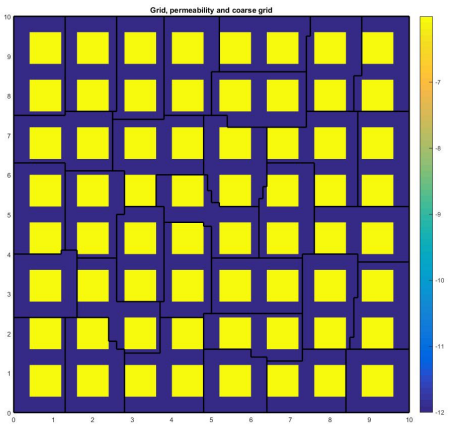
deflation matrix	no. iterations
$Z_1(64)$	103
$Z_2(64)$	93
$Z_{12}(65)$	88
$Z_{21}(65)$	88

Table 7.2: Linear solver iterations using physics-based deflation

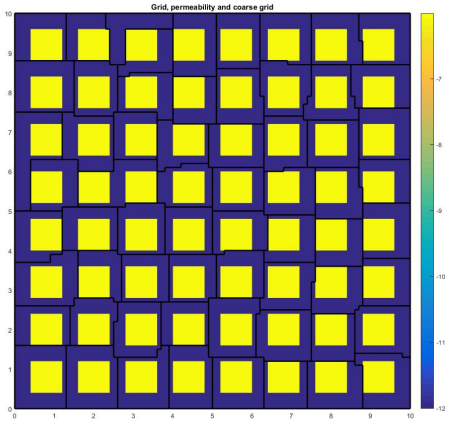
For the domain decomposition we employ 3 METIS-based variants of grid partitioning - into 16, 32 and 64 subdomains:



(a) Partitioning into 16 subdomains



(b) Partitioning into 32 subdomains



(c) Partitioning into 64 subdomains

Figure 7.4: METIS-based grid partitioning.

For these cases the required number of iterations for GMRES, right-preconditioned only by deflation consisting of multiscale basis functions or domain-based vectors, are presented in the table 7.3:

deflation matrix	**no. subdomains**		
	16	*32*	*64*
Z_{ms}	255	203	44
Z_{db}	285	256	59
$Z_{mix}[Z_1\ Z_{ms}]$	51	36	23
$Z_{mix}[Z_1\ Z_{db}]$	49	38	29

Table 7.3: Linear solver iterations using Z_{ms}, Z_{db} and mixed deflation

Note that in the corresponding deflation matrices the number of columns coincide with the chosen number of subdomains, while the mixed version contains vectors from both components. As we can see from the tables above, the best option for the deflation would be usage of mixed deflation.

To investigate the performance of the multigrid idea, we again decompose the computational domain and apply only multiscale operators as a preconditioner for the GMRES, which gives us,

no. subdomains	**no. iterations**
16	95
32	86
48	87
64	20

Table 7.4: Linear solver iterations using multiscale operators

Decoupled ADMS

When we unite deflation and multiscale operators into single preconditioner, we eventually obtain the simplest form of the ADMS:

number of subdomains	deflation matrix							
	Z_1	Z_2	Z_{12}	Z_{21}	Z_{ms}	Z_{db}	$Z_{mix}[Z_1\ Z_{ms}]$	$Z_{mix}[Z_1\ Z_{db}]$
16	37	38	38	38	95	134	39	36
32	28	29	28	28	77	315	27	28
64	14	13	13	13	16	40	14	12

Table 7.5: Linear solver iterations for decoupled ADMS

Fully ADMS

In this section we look at the fully ADMS. Since the only change, compared to the right preconditioner and decoupled versions, is made on the coarse linear solver level, the difference in performance is observed exceptionally in application of both deflation and multiscale operators. In the following table 7.6 we present results for different choices of deflation matrix and grid partitioning:

number of subdomains	deflation matrix			
	Z_1	Z_2	Z_{12}	Z_{21}
16	10	10	9	9
32	10	9	7	7
64	10	6	6	6

Table 7.6: Linear solver iterations for fully ADMS

Mixed ADMS

In this type of ADMS instead of using standard multiscale operators, we complement (by adding corresponding columns) prolongation operator with different physics-based deflation matrices and transpose it to get the restriction operator. Therefore, ”including” deflation into multigrid idea we obtained that the method converges approximately as fast as fully ADMS:

number of subdomains	deflation matrix			
	Z_1	Z_2	Z_{12}	Z_{21}
16	15	16	14	16
32	13	12	12	12
64	11	10	11	10

Table 7.7: Linear solver iterations for mixed ADMS

To summarize the difference in the performance between various preconditioners, we provide the comparison graph for the case with 32 subdomains and physics-based deflation matrix.

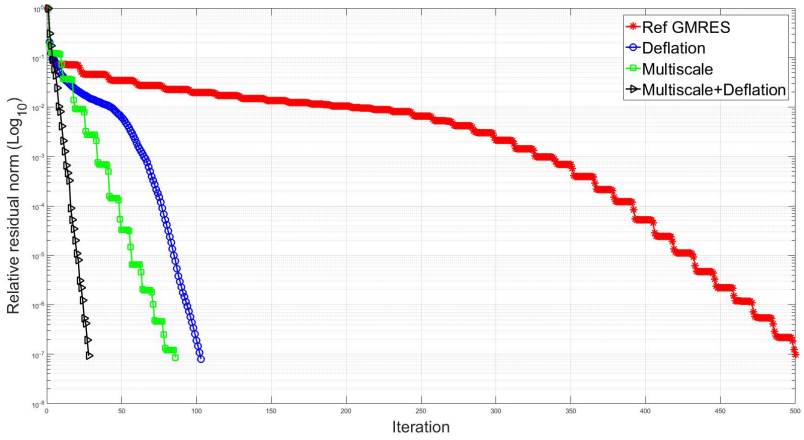


Figure 7.5: Preconditioners performance comparison using METIS-based partitioning into 32 subdomains and Z_1.

7.2.2. Fractured reservoir

In this model problem we consider $2D$ example of the fractured reservoir, where fractures have an essential influence on the fluid flow due to its increased permeability. More specifically, we have generated 30 fractures randomly distributed over the 100×100 domain with a homogeneous permeability which is 10^6 times greater than reservoir permeability (Fig. 7.6).

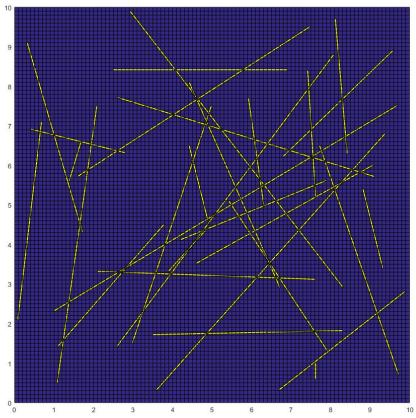


Figure 7.6: Fractured reservoir.

As regards physical features of the system, namely fluid characteristics and boundary conditions, we use the same setup as described in the previous model problem. As an output of the TPFA-type discretization routine we obtain a $10277 \times$

10277 matrix (here the last 277 rows and columns correspond to fractures properties) with the following sparsity pattern:

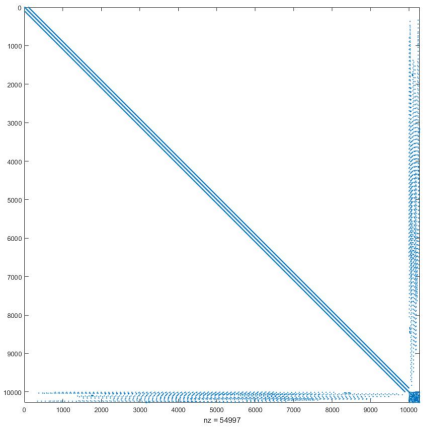
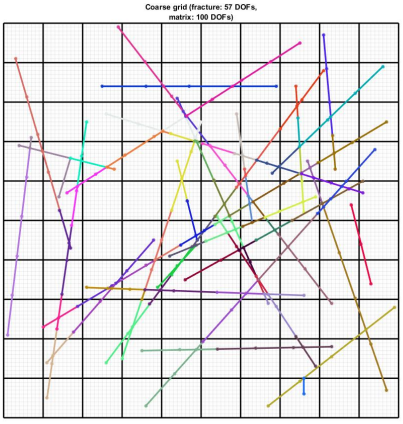


Figure 7.7: Fractured reservoir matrix sparsity pattern.

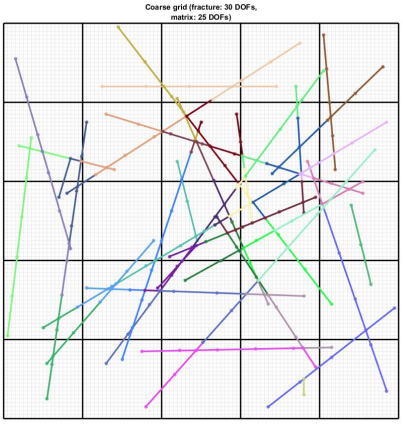
Deflation. For the deflation-based preconditioner we utilize these matrices:

1. Z_1 - *physics-based deflation*: contains 30 vectors of the size 10277, where every vector corresponds to only one fracture, especially 1 in cells crossed by this fracture and 0 otherwise;
2. Z_2 - *physics-based deflation*: similar approach as in Z_1, but now 0 in the intersected cells and 1 otherwise;
3. Z_{12} - *physics-based deflation*: consists of Z_1 and the first vector from Z_2;
4. Z_{21} - *physics-based deflation*: consists of Z_2 and the first vector from Z_1;
5. Z_{ms} - *multiscale basis functions*: contains multiscale basis functions as columns;
6. Z_{db-g} - *grid-based*: here we put 1 into all cells, which belong to the certain subdomain, and 0 into all other cells;
7. Z_{db-f} - *fracture-based*: similar to the *grid-based deflation* approach, but here we put 1 into all cells, which belong to the certain set of fractures pieces after grid partitioning, and 0 into all other cells;
8. Z_{db} - *domain-based*: contains vectors from both Z_{db-g} and Z_{db-g};
9. Z_{mix} - *mixed deflation*: consists of vectors obtained from different approaches (e.g. physics-based deflation and multiscale basis functions);
10. Z_t - *theoretical*: consists of determined beforehand number of the exact eigenvectors corresponding to the smallest magnitude eigenvalues.

In the figure below there is a couple of possible ways to partition the grid and fractures (every colour denotes separate set of fractures):



(a) 57 fracture and 100 grid subdomains



(b) 30 fracture and 25 grid subdomains

Figure 7.8: Grid- and fracture-based partitioning

While standard GMRES method needs 100 iterations to solve the pressure equation with desired tolerance 10^{-9}, we will now present the deflation- and multiscale-based preconditioned version as well as ADMS performance.

Right preconditioner

First of all, we applied the theoretical type of deflation, i.e. using $Z_t(30)$, and found out that the method converges with 48 iterations.

Secondly, we establish the required number of the iterations for all other deflation matrices:

- physics-based deflation:

deflation matrix	no. iterations
$Z_1(30)$	85
$Z_2(30)$	71
$Z_{12}(31)$	70
$Z_{21}(31)$	70

Table 7.8: Linear solver iterations using physics-based deflation

- multiscale-based deflation (*f* and *g* stand for the amount of fractures and grid subdomains respectively):

deflation matrix	grid and fractures partitioning				
	30 *f* & 25 *g*	30 *f* & 16 *g*	21 *f* & 16 *g*	57 *f* & 100 *g*	16 *f* & 100 *g*
Z_{ms}	45	52	51	32	34
Z_{db-f}	84	84	85	85	85
Z_{db-g}	94	96	96	78	78
Z_{db}	59	62	63	43	48

Table 7.9: Linear solver iterations using multiscale (grid- and fractures-based) deflation

- mixed deflation:

deflation matrix	grid and fractures partitioning				
	21 *f* & 25 *g*	16 *f* & 100 *g*	16 *f* & 16 *g*	57 *f* & 25 *g*	12 *f* & 16 *g*
$Z_{mix}[Z_1\ Z_{ms}]$	44	32	50	43	50
$Z_{mix}[Z_1\ Z_{db-g}]$	60	51	64	60	64
$Z_{mix}[Z_2\ Z_{ms}]$	45	32	50	43	50
$Z_{mix}[Z_2\ Z_{db-g}]$	60	51	64	60	64

Table 7.10: Linear solver iterations using mixed deflation

The performance of only multiscale-based preconditioner is reflected in the table below:

fractures and grid partitioning	no. iterations
16 *f* & 25 *g*	67
21 *f* & 16 *g*	66
16 *f* & 100 *g*	53
12 *f* & 16 *g*	42
9 *f* & 16 *g*	33
9 *f* & 25 *g*	29
9 *f* & 100 *g*	25

Table 7.11: Linear solver iterations using multiscale-based preconditioner

As we can observe from the table 7.11, even quite rough decomposition has a considerable effect on the convergence rate.

Decoupled ADMS

Finally, we demonstrate the results obtained after applying the full preconditioner (both deflation and multiscale components):

fractures and grid partitioning	**deflation matrix**					
	Z_1	Z_2	Z_{ms}	$Z_t(30)$	$Z_{mix}[Z_1\ Z_{ms}]$	$Z_{mix}[Z_1\ Z_{db-g}]$
9 *f* & 16 *g*	35	33	27	19	33	35
9 *f* & 25 *g*	29	27	30	18	27	27
9 *f* & 100 *g*	26	24	18	18	18	23

Table 7.12: Linear solver iterations for ADMS as a right preconditioner for GMRES

To complete the investigation of the first ADMS modification, we provide the convergence plot showing relative residual norms depending on the iteration number:

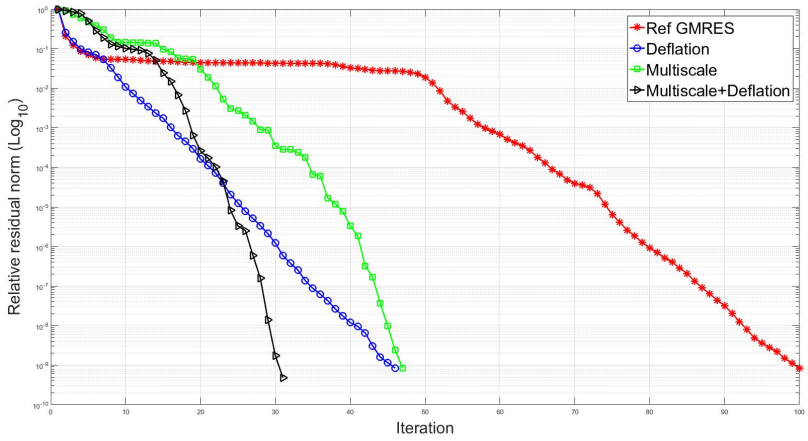


Figure 7.9: Preconditioners performance comparison using Z_{ms}.

7.2.3. SPE10 layer

The second model of the *SPE10* project is a highly heterogeneous reservoir, characterized by large permeability variations (up to 6 orders of magnitude), with a regular geometry described on a Cartesian grid with $60 \times 220 \times 85$ $(1,122,000)$ cells, which has the physical size of $1200 \times 2200 \times 170$ (ft), where 1 $foot$ is $0,3048$ $meters$. For the detailed description of the *SPE Comparative Solution Project* one can visit:

http://www.spe.org/web/csp/datasets/set02.htm

For testing purposes we consider the *third* layer (permeability distribution is shown in figure 7.10) of the reservoir, which is a part of a shallow-marine Tarbert formation [59].

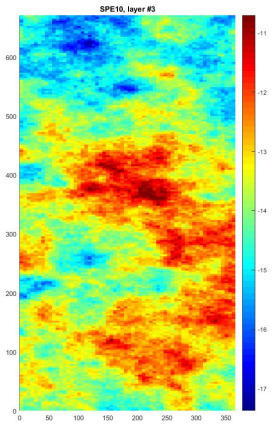
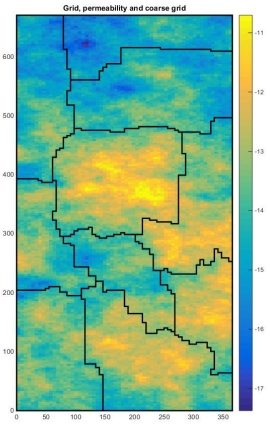


Figure 7.10: Permeability distribution of the 3rd layer in the SPE10 project, model 2.

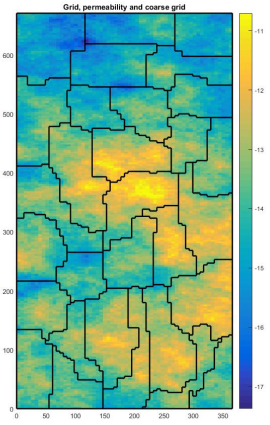
In this scenario we make use of 3 deflation matrices (the way to construct them is explained in 2 previous model problems):

- Z_{ms} - *multiscale basis functions*: contains multiscale basis functions as columns;
- Z_{db} - *domain-based*: 1 in cells belonging to the certain subdomain and 0 otherwise;
- Z_t – *theoretical*: consists of determined beforehand number of the exact eigenvectors corresponding to the smallest magnitude eigenvalues.

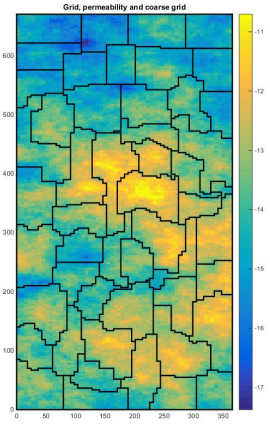
To partition the grid, we subdivide the domain by METIS-based routine into $10, 30$ and 50 subdomains, respectively. In the figure 7.11 these decompositions are presented:



(a) Partitioning into 10 subdomains



(b) Partitioning into 30 subdomains

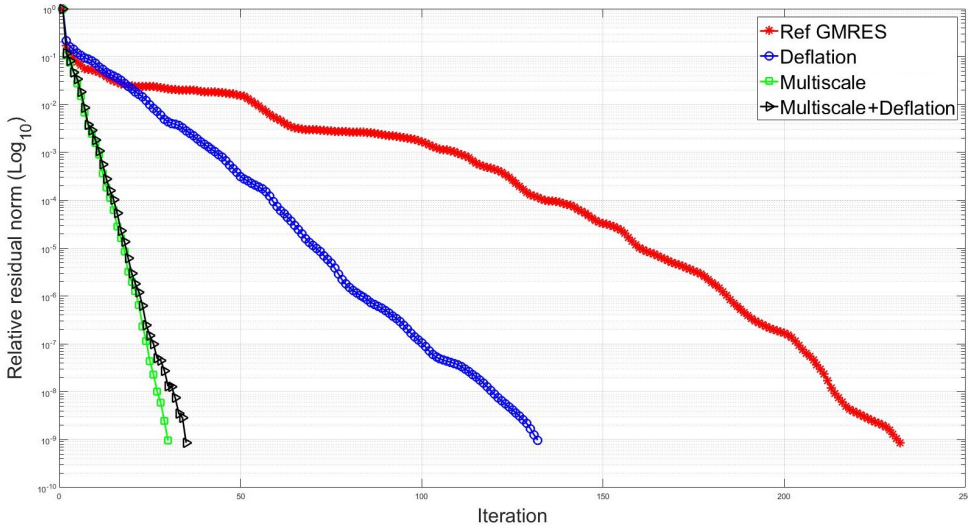


(c) Partitioning into 50 subdomains

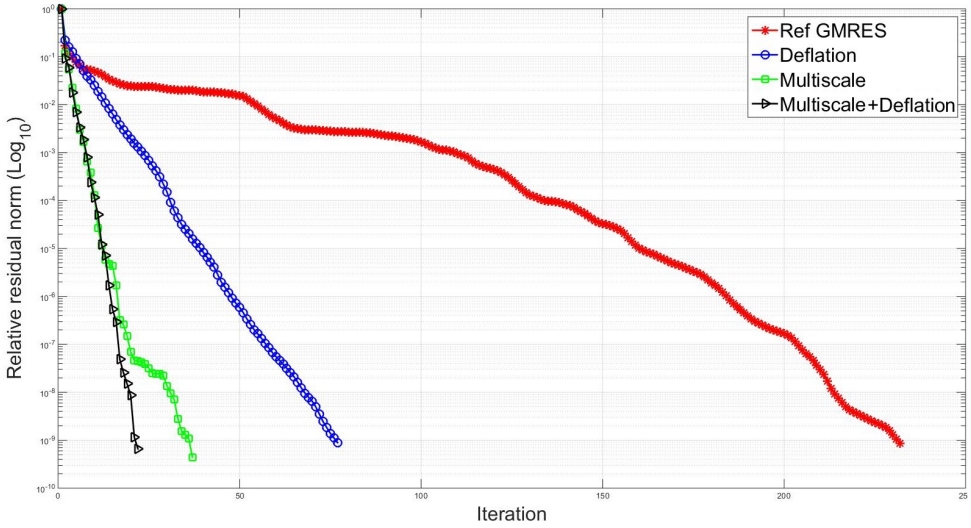
Figure 7.11: METIS-based grid partitioning.

Decoupled ADMS

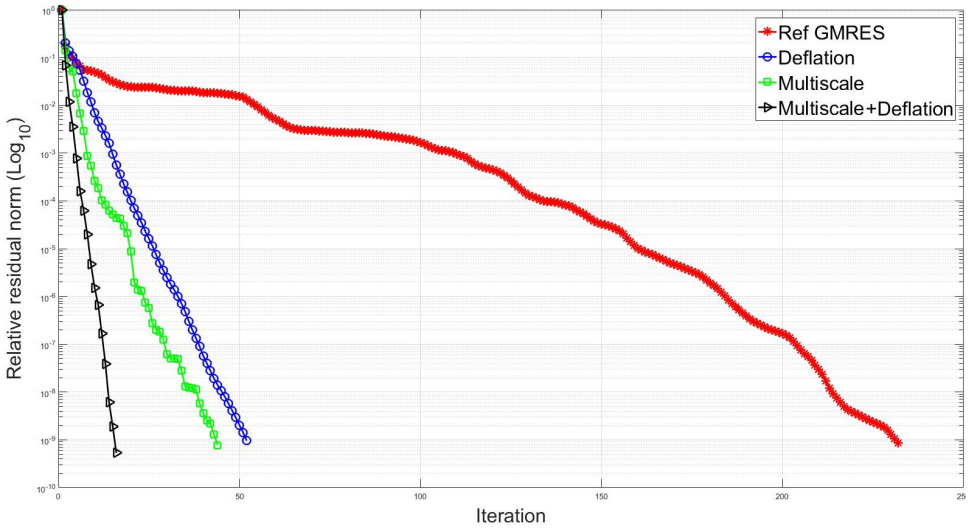
For every type of decomposition, we performed several tests and discovered that in fact the usage of Z_{ms} as a deflation matrix outperforms (requires less number of iterations to converge) even preconditioner with Z_t. Moreover, if we increase the number of subdomains, then the dominance of the combined preconditioner becomes more conspicuous, which is reflected in the figure below:



(a) METIS-based partitioning into 10 subdomains



(b) METIS-based partitioning into 30 subdomains



(c) METIS-based partitioning into 50 subdomains

Figure 7.12: Preconditioners performance comparison using Z_{ms} for various domain decompositions.

In general, for a such heterogeneous reservoir it is complicated to construct physics-based deflation matrix, which makes Z_{ms} optimal option, whereas the partitioning scheme is also highly problem dependent, i.e. the optimal number of subdomains has to be defined during thorough analysis of the computational domain in order to obtain the balance between computational effort and fast convergence rate.

Fully ADMS

This version of the preconditioner allows method to converge in 5 and even 2 iterations for grid partitioning into 5 and 7 subdomains, respectively, which means it is the fastest option among all modifications.

References

[1] H. Walter and B. Epple, *Numerical Simulation of Power Plants and Firing Systems* (Springer, 2016).

[2] R. Hill, *Elastic properties of reinforced solids: some theoretical principles,* Journal of the Mechanics and Physics of Solids **11**, 357 (1963).

[3] K. Aziz and A. Settari, *Petroleum reservoir simulation* (Chapman & Hall, 1979).

[4] J. Bear, *Dynamics of fluids in porous media* (Courier Corporation, 2013).

[5] S. Whitaker, *Flow in porous media i: A theoretical derivation of darcy's law,* Transport in Porous Media **1**, 3 (1986).

[6] Z. Chen, G. Huan, and Y. Ma, *Computational methods for multiphase flows in porous media* (SIAM, 2006).

[7] D. W. Peaceman, *Fundamentals of numerical reservoir simulation. 1977,* .

[8] C. E. Brennen, *Fundamentals of multiphase flow* (Cambridge university press, 2005).

[9] C. T. Crowe, J. D. Schwarzkopf, M. Sommerfeld, and Y. Tsuji, *Multiphase flows with droplets and particles* (CRC press, 2011).

[10] M. Muskat, R. D. Wyckoff, *et al.*, *Flow of homogeneous fluids through porous media,* (1937).

[11] J. A. Trangenstein and J. B. Bell, *Mathematical structure of the black-oil model for petroleum reservoir simulation,* SIAM Journal on Applied Mathematics **49**, 749 (1989).

[12] I. Aavatsmark, *Interpretation of a two-point flux stencil for skew parallelogram grids,* Computational geosciences **11**, 199 (2007).

[13] I. Aavatsmark, *An introduction to multipoint flux approximations for quadrilateral grids,* Computational Geosciences **6**, 405 (2002).

[14] R. A. Klausen and R. Winther, *Convergence of multipoint flux approximations on quadrilateral grids,* Numerical methods for partial differential equations **22**, 1438 (2006).

[15] M. G. Edwards and C. F. Rogers, *Finite volume discretization with imposed flux continuity for the general tensor pressure equation,* Computational Geosciences **2**, 259 (1998).

[16] I. Aavatsmark, G. Eigestad, and R. Klausen, *Numerical convergence of the mpfa o-method for general quadrilateral grids in two and three dimensions,* Compatible spatial discretizations , 1 (2006).

[17] R. A. Klausen and R. Winther, *Robust convergence of multi point flux approximation on rough grids,* Numerische Mathematik **104**, 317 (2006).

[18] G. Moog, *Advanced discretization methods for flow simulation using unstructured grids,* Department of Energy Resources Engineering, Stanford University CA (2013).

[19] K.-A. Lie, S. Krogstad, I. S. Ligaarden, J. R. Natvig, H. M. Nilsen, and B. Skaflestad, *Open-source matlab implementation of consistent discretisations on complex grids,* Computational Geosciences **16**, 297 (2012).

[20] I. Aavatsmark and G. Eigestad, *Numerical convergence of the mpfa o-method and u-method for general quadrilateral grids,* International journal for numerical methods in fluids **51**, 939 (2006).

[21] J. Nordbotten and I. Aavatsmark, *Monotonicity conditions for control volume methods on uniform parallelogram grids in homogeneous media,* Computational Geosciences **9**, 61 (2005).

[22] J. M. Nordbotten and G. T. Eigestad, *Discretization on quadrilateral grids with improved monotonicity properties,* Journal of computational physics **203**, 744 (2005).

[23] K.-A. Lie, *An introduction to reservoir simulation using matlab: user guide for the matlab reservoir simulation toolbox (mrst). sintef ict,* (2016).

[24] Y. Saad, *Iterative methods for sparse linear systems* (SIAM, 2003).

[25] J. Ruge and K. Stuben, *Algebraic multigrid (amg),* In: McCormick, S. (Ed.) Multigrid Methods, 3, SIAM, Philadelphia, PA , 73 (1987).

[26] T. B. Jönsthövel, A. A. Lukyanov, E. D. Wobbes, and C. Vuik, *Monotone non-galerkin algebraic multigrid method applied to reservoir simulations,* in *ECMOR XV-15th European Conference on the Mathematics of Oil Recovery* (2016).

[27] H. Klie, M. Wheeler, K. Stueben, and T. Clees, *Deflation amg solvers for highly ill-conditioned reservoir simulation problems,* in *Reservoir Simulation Symposium, doi:10.2118/105820-MS* (2007).

[28] J. Erhel, K. Burrage, and B. Pohl, *Restarted gmres preconditioned by deflation,* Journal of computational and applied mathematics **69**, 303 (1996).

[29] L. Mansfield, *Damped jacobi preconditioning and coarse grid deflation for conjugate gradient iteration on parallel computers,* SIAM Journal on Scientific and Statistical Computing **12**, 1314 (1991).

[30] R. A. Nicolaides, *Deflation of conjugate gradients with applications to boundary value problems,* SIAM Journal on Numerical Analysis **24**, 355 (1987).

[31] J. Tang and C. Vuik, *On deflation and singular symmetric positive semi-definite matrices,* Journal of computational and applied mathematics **206**, 603 (2007).

[32] C. Vuik, A. Segal, and J. Meijerink, *An efficient preconditioned cg method for the solution of a class of layered problems with extreme contrasts in the coefficients,* Journal of Computational Physics **152**, 385 (1999).

[33] Y. Saad and M. H. Schultz, *Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems,* SIAM Journal on scientific and statistical computing **7**, 856 (1986).

[34] Y. Saad, *A flexible inner-outer preconditioned gmres algorithm,* SIAM Journal on Scientific Computing **14**, 461 (1993).

[35] R. B. Morgan, *Gmres with deflated restarting,* SIAM Journal on Scientific Computing **24**, 20 (2002).

[36] J. van der Linden, T. Jönsthövel, A. A. Lukyanov, and C. Vuik, *The parallel subdomain-levelset deflation method in reservoir simulation,* Journal of Computational Physics **304**, 340 (2016).

[37] A. A. Lukyanov, J. van der Linden, T. B. Jönsthövel, and C. Vuik, *Meshless subdomain deflation vectors in the preconditioned krylov subspace iterative solvers,* in *ECMOR XV-14th European Conference on the Mathematics of Oil Recovery* (2016).

[38] C. L. Farmer, *Upscaling: a review,* International Journal for Numerical Methods in Fluids **40**, 63 (2002).

[39] M. Pal, S. Lamine, K.-A. Lie, S. Krogstad, *et al.*, *Multiscale method for simulating two-and three-phase flow in porous media,* in *SPE Reservoir Simulation Symposium* (Society of Petroleum Engineers, 2013).

[40] P. Audigane and M. J. Blunt, *Dual mesh method for upscaling in waterflood simulation,* Transport in Porous Media **55**, 71 (2004).

[41] T. Y. Hou and X.-H. Wu, *A multiscale finite element method for elliptic problems in composite materials and porous media,* Journal of Computational Physics **134**, 169 (1997).

[42] T. Hou, X.-H. Wu, and Z. Cai, *Convergence of a multiscale finite element method for elliptic problems with rapidly oscillating coefficients,* Mathematics of Computation of the American Mathematical Society **68**, 913 (1999).

[43] Z. Chen and T. Hou, *A mixed multiscale finite element method for elliptic problems with oscillating coefficients,* Mathematics of Computation **72**, 541 (2003).

[44] J. E. Aarnes, S. Krogstad, and K.-A. Lie, *A hierarchical multiscale method for two-phase flow based upon mixed finite elements and nonuniform coarse grids,* Multiscale Modeling & Simulation **5**, 337 (2006).

[45] P. Jenny, S. Lee, and H. Tchelepi, *Multi-scale finite-volume method for elliptic problems in subsurface flow simulation,* Journal of Computational Physics **187**, 47 (2003).

[46] H. Hajibeygi and P. Jenny, *Multiscale finite-volume method for parabolic problems arising from compressible multiphase flow in porous media,* Journal of Computational Physics **228**, 5129 (2009).

[47] P. Jenny, S. H. Lee, and H. A. Tchelepi, *Adaptive fully implicit multi-scale finite-volume method for multi-phase flow and transport in heterogeneous porous media,* Journal of Computational Physics **217**, 627 (2006).

[48] H. Zhou, H. A. Tchelepi, *et al.*, *Operator-based multiscale method for compressible flow,* in *SPE Reservoir Simulation Symposium* (Society of Petroleum Engineers, 2007).

[49] V. Kippe, J. E. Aarnes, and K.-A. Lie, *A comparison of multiscale methods for elliptic problems in porous media flow,* Computational Geosciences **12**, 377 (2008).

[50] H. Hajibeygi, G. Bonfigli, M. A. Hesse, and P. Jenny, *Iterative multiscale finite-volume method,* Journal of Computational Physics **227**, 8604 (2008).

[51] H. Zhou, H. A. Tchelepi, *et al.*, *Two-stage algebraic multiscale linear solver for highly heterogeneous reservoir models,* SPE Journal **17**, 523 (2012).

[52] K. Lie, O. Møyner, J. Natvig, A. Kozlova, K. Bratvedt, S. Watanabe, and Z. Li, *Successful application of multiscale methods in a real reservoir simulator environment,* in *ECMOR XV-15th European Conference on the Mathematics of Oil Recovery* (2016).

[53] O. Møyner and K.-A. Lie, *A multiscale restriction-smoothed basis method for high contrast porous media represented on unstructured grids,* Journal of Computational Physics **304**, 46 (2016).

[54] Y. Wang, H. Hajibeygi, and H. A. Tchelepi, *Algebraic multiscale solver for flow in heterogeneous porous media,* Journal of Computational Physics **259**, 284 (2014).

[55] A. A. Lukyanov and C. Vuik, *Parallel fully implicit smoothed particle hydrodynamics based multiscale method,* in *ECMOR XV-15th European Conference on the Mathematics of Oil Recovery* (2016).

[56] V. Dolean, P. Jolivet, F. Nataf, N. Spillane, and H. Xiang, *Two-level domain decomposition methods for highly heterogeneous darcy equations. connections with multiscale methods,* Oil & Gas Science and Technology–Revue d'IFP Energies nouvelles **69**, 731 (2014).

[57] G. Karypis and V. Kumar, *Metis: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices,* University of Minnesota, Department of Computer Science and Engineering, Army HPC Research Center, Minneapolis, MN (1998).

[58] M. Christie, M. Blunt, *et al.*, *Tenth spe comparative solution project: A comparison of upscaling techniques,* in *SPE Reservoir Simulation Symposium* (Society of Petroleum Engineers, 2001).

[59] S. Davies, N. Dawers, A. McLeod, and J. Underhill, *The structural and sedimentological evolution of early synrift successions: the middle jurassic tarbert formation, north sea,* Basin Research **12**, 343 (2000).