RAPID COMPOSITIONAL SIMULATION USING MODEL ORDER REDUCTION AND MACHINE LEARNING TECHNIQUE

A Dissertation

by

JAE WOOK LEE

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2020

Major Subject: Petroleum Engineering

ABSTRACT

Fast and reliable reservoir simulation is a key for the successful decision making in integrated reservoir studies. Large and complex multiphase reservoir models usually require expensive computational infrastructure. Physics-based model order reduction (MOR) methods, especially snapshots-based methods such as the proper orthogonal decomposition (POD), have been introduced and applied especially for mitigating the computational cost of black oil models in workflows that require multiple calls of the reservoir simulator. However, only a limited number of methods have looked deeper at the effectiveness of these techniques to multiphase and compositional simulation where expensive flash and phase equilibrium calculations are added to the level of complexities associated with obtaining robust solutions. In this work, we develop coupled physics-based and artificial neural network (ANN)-based MOR techniques for rapid compositional simulations that accelerate calibrating of phase equilibrium during expensive computations. We base our framework on the so-called the POD-DEIM, which uses the discrete empirical interpolation method (DEIM) step to overcome the cost associated with the nonlinear terms.

Rapid flash calculation can be accomplished by use of machine learning method such as ANN. The fully connected trained network yields reliable estimation for the solutions of the composition related variables. Therefore, the process for obtaining the solutions for the flash calculation is substituted without the expensive computation of

Newton-Raphson iteration.

In this study, we introduce a new formulation for the POD-DEIM method applied to a compositional simulator. The new formulation allows the tracking and approximation of each component individually as opposed to only pressures and saturations. We test the robustness of the POD-DEIM method integrated with the ANN-based rapid flash calculation to reduce the computational cost for multi-phase, multi-components 3D reservoir model. Our results show that the POD-DEIM technique enables us to approximate the conventional model with high levels of accuracy up to more than 99%. And it also enables a faster simulation due to the reduced order system. The reduced order modeling using the POD-DEIM reduces the CPU time of the compositional simulation by around 14% comparing to the fine scale model. Machine learning method makes the model get to the solutions much faster without a solver for the Newton-Raphson method. ANN-based MOR accelerates the flash calculation and the coupled the POD-DEIM and ANN technique reduces the CPU time of the compositional simulation by around 14.5% comparing to the fine scale model. When the rapid flash calculation using ANN is combined with the POD-DEIM, one can save the overall simulation time in both solving the system and calculating the EOS-based equilibrium equations.

DEDICATION

To my beloved parents, wife, and son for their endless love, support, and patience throughout my academic journey and my life.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisor, Dr. Gildin, for his guidance and support. He was not only an excellent academic mentor, but also a life mentor who showed passion on research work as well as a great attitude as a scholar. His encouragement and faith in me led me to this accomplishment.

I would like to also thank my committee members, Dr. Killough, Dr. Barrufet, and Dr. Efendiev for their valuable feedbacks.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supervised by a dissertation committee consisting of Professor Eduardo Gildin (chair), John Killough, and Maria Barrufet in the Department of Petroleum Engineering and Professor Yalchin Efendiev in the Department of Mathematics.

In Chapter 2, the in-house compositional simulator originally developed by Dr. Ernesto Valbuena Olivares was modified by the student and used for the reduced order modeling.

All other work conducted for the dissertation was completed by the student independently.

Funding Sources

Jae Wook Lee's Ph.D. works have been supported by Energi Simulation (ES) through the ES Research Chair at Texas A&M University.

NOMENCLATURE

A	area, ft2
κ_{ij}	binary interaction coefficient between components i and j
p_{wf}	bottomhole flowing pressure for perforated gridblock, psia
z_c	central gridblock depth, ft
p_{oC}	central gridblock oil phase pressure, psia
Z_α	compressibility factor of phase α, dimensionless
p_{c_i}	critical pressure of component i, psia
T_{c_i}	critical temperature of component i, °F or R
A	cubic EOS coefficient
B	cubic EOS coefficient
ρ	density, lb/ft3
D	diameter, in or ft
b	EOS mixture parameter (linear mixing rule)
$(a\alpha)$	EOS mixture parameter (quadratic mixing rule)
a_i	EOS parameter for component i
b_i	EOS parameter for component i
K_i	equilibrium ratio of component i
ε	error tolerance
$\hat{\varphi}_i^\alpha$	fugacity coefficient of component i in phase α

$\hat{f}_i^\alpha$	fugacity of component i in phase α
R_{fug_i}	fugacity residual for component i
p_{cgo}	gas-water capillary pressure, psi
R	gas constant, 10.7316 ft3.psi/°R/lbmol or 8.31446 J/K/mol
S_g	gas saturation, V/V fraction
g_c	gravity constant
V_b	gridblock rock bulk volume, ft3
Δx	gridblock size in x-direction
$\Delta \mathrm{y}$	gridblock size in y-direction
$\Delta \mathrm{z}$	gridblock size in z-direction
R_i	hydrocarbon component residual, lbmol/day
T_η	interblock geometric transmissibility
$\lambda_{\alpha\eta}$	interblock phase α mobility
$\vec{\vec{J}}$	Jacobian matrix
L	length, ft
f_l	liquid molar fraction
x_i	liquid molar fraction of component i
W	mass of water per unit volume, lb/ft3
$\tilde{\rho}_\alpha$	molar density of phase α, lbmol/ft3
$\mathcal{v}_\alpha$	molar volume of phase α, ft3/lbmol

M_{w_i}	molecular weight of component i, lb/lbmol
M_w^{α}	molecular weight of phase α, lb/lbmol
z_{η}	neighbor gridblock depth, ft
$p_{o\eta}$	neighbor gridblock oil phase pressure, psia
$\dot{m}_{s/s}$	net mass rate from source/sink
$\dot{n}_{i_{s/s}}$	net molar rate of component i from source/sink
n_c	number of hydrocarbon components
F_i	number of moles of component i per unit volume, lbmol/ft3
$n_{gridblocks}$	number of reservoir gridblocks
p_{cow}	oil-water capillary pressure, psi
p_o	oil phase pressure, psia
k_{rog}	oil relative permeability at actual gas saturation and connate water saturation
k_{row}	oil relative permeability at actual water saturation
k_{rocw}	oil relative permeability at connate water saturation
S_o	oil saturation, V/V fraction
Φ_{α}	potential of phase α, psia
p	pressure, psia
R_{RR}	Rachford-Rice residual
p_{ref}	reference pressure, psia
$k_{r\alpha}$	relative permeability of phase α, dimensionless

$\vec{\vec{k}}$	rock permeability tensor, mD
ϕ	rock porosity, V/V fraction
R_{sat}	saturation (volume constraint) residual
T	temperature, °F or R
t	time, day
Δt	time step size, day
Δ_τ	time differential operator
z_i	total molar fraction of component i
F	total number of moles per unit volume, lbmol/ft3
$a_{\alpha\eta}$	transmissibility of phase α between central gridblock and neighbor gridblock in η direction, lbmol/day/psi v
β	unit conversion factor
f_v	vapor molar fraction
y_i	vapor molar fraction of component i
$\vec{R}$	vector of residuals
$\vec{x}$	vector of unknowns
$\vec{V}$	velocity of fluid, ft/day
μ_α	viscosity of phase α, cP
V	volume, STB or ft3
q_α	volumetric rate of phase α, STB/day, MSCF/day, or ft3/day

$q_{w_{s/s}}$	volumetric water rate from a well, ft3/day
R_w	water residual, lb/day
S_w	water saturation, V/V fraction
B_w	water volumetric factor, bbl/STB
BHP	well bottomhole pressure (at reference depth), psia
WI_α	well index for phase α, lbmol/day/psi
WI_{geom}	well index geometric component
r_w	wellbore radius, ft

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Page

1. INTRODUCTION

1.1. Background

Reservoir simulation predicts the flow of fluids by using numerical models. Although the underlying equations for any porous media flow simulation are based on conservation of mass, typically, two families of reservoir simulators are used based on the rock-fluid structure of a particular case study. Black oil models are used to study the fluid model which can be expressed as a function of pressure and saturation with fixed composition. While compositional models are used when either the in-place or injected fluid causes fluid properties to be dependent on composition (Young and Stephenson 1983). As the industry drills the wells deeper in which accompanied by high pressure and high temperature, there has been more needs for compositional simulation. Furthermore, the importance of the enhanced oil recovery using gas injection and the development of unconventional reservoirs increase the demand for compositional reservoir modeling.

As the reservoir simulation plays an important role in reservoir engineering, especially to understand reservoir dynamics and to forecast production in more complex reservoirs, methods to reduce simulation turnarounds, e.g., the computational time and the nonlinearity of problems in high fidelity reservoir simulation models, become paramount to fast decision-making processes. Particularly, compositional reservoir simulation includes phase equilibrium calculation and many numbers of variables, therefore it requires higher computational infrastructure. Due to the rapid improvement of high-performance computing one can solve large problems much faster than ever before (Figure 1.1), however, the efforts to develop techniques making simulation models more efficient and more concise should be accompanied by the development of hardware in order to achieve an optimized simulation.

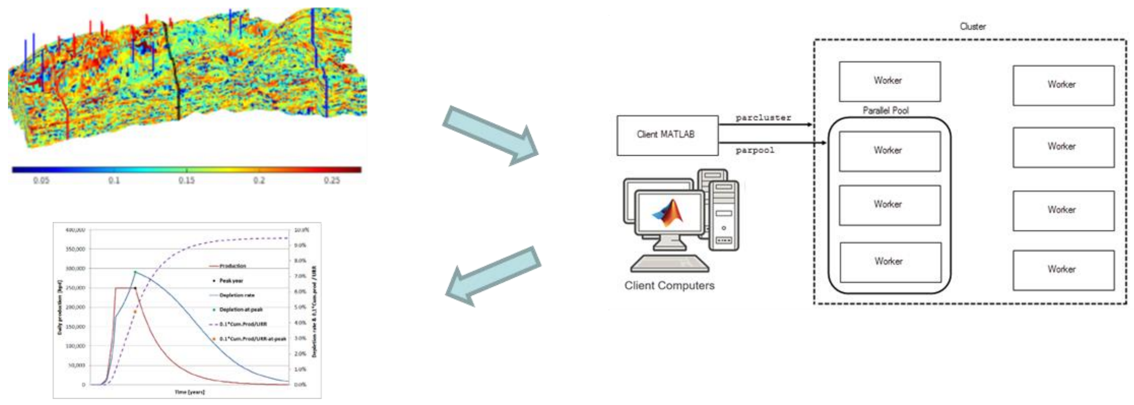


Figure 1.1 Parallel computing for high fidelity reservoir model. Improvement of high-performance computing allows more complex reservoir simulation with a reduced CPU time.

To this end, there have been many different techniques to reduce the computational cost for simulation models. Upscaling for flow computation or multi-scale method are the examples of the methods which yield approximations that preserve local characteristics of the fluid model. In addition, many model reduction approaches and rapid simulation techniques such as proper orthogonal decomposition-discrete empirical interpolation method (POD-DEIM), proper orthogonal decomposition-trajectory piecewise linearization (POD-TPWL), which retain the global features, have been introduced and applied to mostly black oil models.

In this study, we focus on two different MOR methods, one is physics-based MOR and the other is ANN-based MOR technique. The physics-based MOR is a technique for reducing the computational complexity of mathematical models in numerical simulations. It is one of the most appealing approaches for faster numerical simulation. Hence, various MOR methodologies have been introduced in the field of reservoir simulation, and they have shown very good performance and accuracy in the approximation to the original fully implicit reservoir simulation. ANN-based MOR is realized by machine learning technique. A trained model based on a given dataset provides prediction when a new data is employed. Reservoir simulation enhanced by machine learning is one of the most promising field of study in petroleum engineering, since the derived proxy models provide direct solutions for the given problems, i.e., without being intrusive to the simulator code, at a very small amount of CPU time, in contrast with the physics-based MOR.

1.2. Literature Review

1.2.1 Multi-phase Compositional Reservoir Simulation

The system of equations in black oil models can be explained as a function of pressure and saturation, and phase composition is considered constant. In contrast with black oil models, the compositional models handle each hydrocarbon component separately.

There are many different ways to build compositional reservoir models depending on the approaches how to set the variables and the system of equations.

Molar variables method was introduced using saturation constraint and n_c phase equilibrium equations which account for the state when the phase fugacities for each component are equal (Fussell and Fussell 1979). Variables for this method are pressure, phase fraction, phase composition, and overall composition of each component.

Natural variables method used variables such as pressure, phase saturation, and phase composition of each component (Coats 1980). The characteristic of this method is that one needs to replace component mole fraction in the oil phase with another primary variable if the oil phase disappears in the process of simulation (He et al. 2014).

Another molar variables method was introduced using the similar variables as the previous molar formulation (Young and Stephenson 1983). The difference between the two molar methods is that Young and Stephenson's method computes the composition of the second (oil) phase using the composition of the first (gas) phase.

Volume-balance formulation used extensive molar variables (Acs et al. 1985). The sequence of computations in simulation is similar to the molar formulations.

Even though each formulation has its own advantage in analyzing nonlinear behaviors, the natural variables method or molar variables method is the preferred approach for isothermal compositional simulation. In this study, we base molar variables method.

1.2.2. Model Order Reduction

Compositional simulators essentially include phase flash calculations with equilibrium checking in order to obtain thermodynamically consistent models. Therefore, the system of equations in compositional models comprises of a greater number of variables in addition to pressure and saturation, and it results in a larger system to solve. The additional nonlinear terms in the equations are also an issue since they disrupt the efficiency of calculations. In essence, they are more complex and computationally expensive processes than black oil models. All these complexities in compositional model pose challenges to attain accurate solutions in a timely manner.

Over the decades, many efforts have been made in the field of reservoir simulation in order to achieve faster and more efficient simulations while retaining the reliability and accuracy.

The combination of POD and DEIM is one of the widely used and robust MOR methods (Chaturantabut and Sorensen 2011). POD reduces the size of the system to be solved, and DEIM contributes to approximate the nonlinear terms for faster computation by supplementing the disadvantage of solving the full-order model when applying POD. These snapshot-based methods work in a two-step process. In the training step one can obtain the snapshots, which are the solutions of the states (pressures, saturations, compositions) at each time step for the full simulation model. These solutions are to be used to derive the POD basis in a later step. Similarly, the DEIM basis is obtained from the snapshots of nonlinear functions. Then the test case, i.e., simulation with inputs different than the training, is utilized to validate if the reduced model works for the different well control schedule cases and to compare the speed of the simulation run time.

POD-DEIM has been used in black oil models to reduce the complexity of the system as well as to approximate the mobility related the nonlinear term (Gildin et al. 2013). This POD-DEIM approach has been extended to the localized model reduction (LPOD-LDEIM) by computing several local subspaces (Ghasemi and Gildin 2015). Yoon et al. proposed hyper-reduction that consists of state reduction, constraint reduction, and nonlinearity treatment based on POD, Petrov-Galerkin projection, and “gappy” reconstruction (Yoon et al. 2016). Also, the use of the trajectory-based DEIM (TDEIM) to approximate the nonlinear terms has been introduced (Tan et al. 2017). They interpolate the perturbed term which is defined as the difference between the test and the training terms instead of the original nonlinear term. Figure 1.2 shows the schematic diagram of the studies for physics-based MOR in black oil models. All the MOR methods work in a two-step process which are offline processing and online processing. For example, the physics-based MORs for black oil models such as POD-DEIM, LPOD-LDEIM reduce the orders of states, residuals, and Jacobians in material balance based on the training controls. And these MOR basis that are pre-computed in offline are used in online process for new controls.

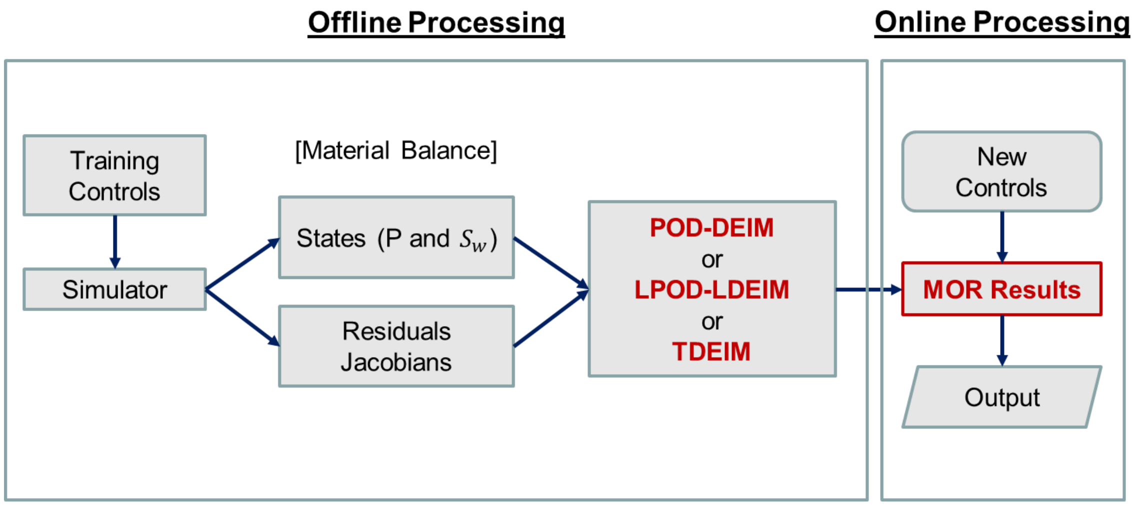


Figure 1.2 Physics-based MOR: Black oil models. In offline processing, physics-based MORs reduce the orders in material balance, and the pre-computed MOR basis are utilized for new controls in online processing.

When it comes to compositional models, there has been an attempt to apply the MOR using the technique combining POD and trajectory-piecewise linearization (TPWL) (Cardoso and Durlofsky 2010a; He and Durlofsky 2014). This method uses linearization around the states generated during previous training runs. Figure 1.3 illustrates the schematic diagram of physics-based MOR in compositional models. Physics-based MOR applied to compositional simulation follows the same structure as of MOR implemented for black oil models. However, in compositional simulators, the conventional flash calculation is necessary to track individual components, which may impose challenges to attain accurate solutions after the application of MOR. To overcome this limitation, a recent study applied the POD to solve the flow equation and the DEIM is used to solve the Peng-Robinson equation of state, in the calculation of compressible single-phase gas reservoir flow (Li et al. 2020).

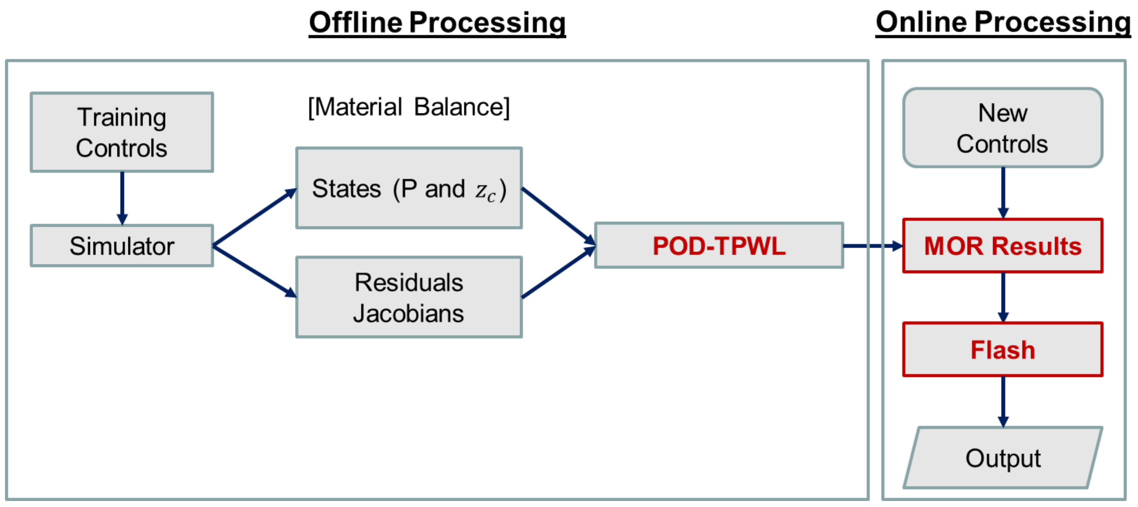


Figure 1.3 Physics-based MOR: Compositional models. Physics-based MORs in compositional simulation reduce the orders in material balance, however the conventional flash calculation is still necessary in online processing.

For the data-driven MOR, artificial (recurrent) neural networks (RNNs) based reservoir simulation was developed to capture nonlinear interaction between wells, fluid and rock to predict production rates. Figure 1.4 displays that data-driven MOR replaces the entire material balance equations, and the results are utilized for the new control cases.

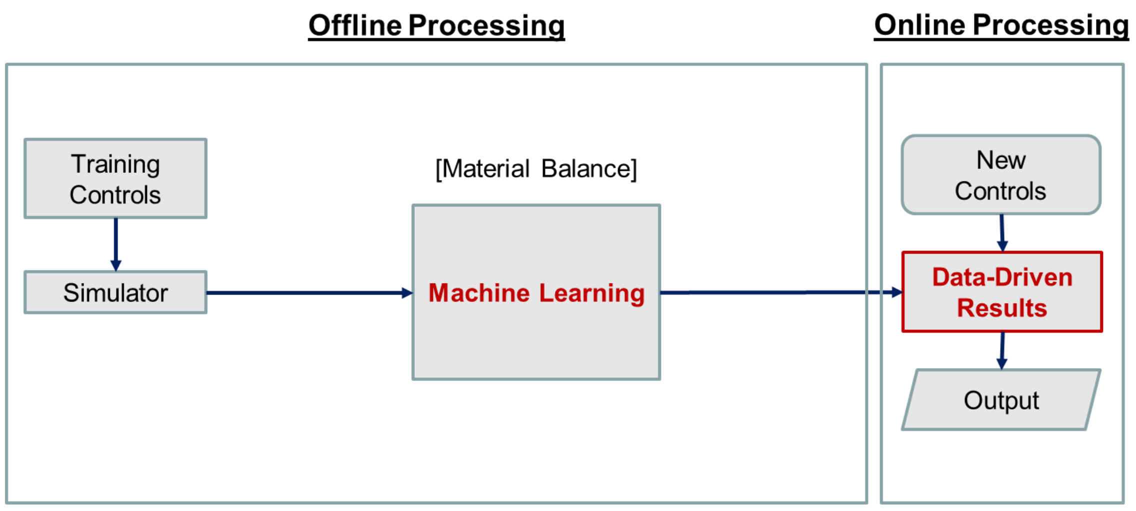


Figure 1.4 Data-driven MOR: Black oil models. Machine learning techniques can substitute the entire material balance without solving the system iteratively.

The limitations of these studies stem from the fact that the physics-based MOR were applied mostly to the black oil models. A few studies applied the physics-based MOR to the compositional models. The POD-TPWL technique for the compositional model was only used for the mass balance calculation while performing the flash calculation in a conventional way. Furthermore, a recent study of the POD-DEIM for the compositional model (Lee and Gildin 2020) also had a limitation such as the POD-DEIM were used separately in solving the flow equation and solving the equation of state, thus they were unable to completely reduce the nonlinearities of the model. In addition, data-driven MOR has not been entirely validated in the compositional cases.

1.2.3. Machine Learning

Phase equilibrium calculations that are based on the equation of state (EOS) is essential to predict accurate phase behavior in compositional simulation. However, phase calculation process requires expensive computational cost through multiple iterations due to its nature of nonlinearity. Moreover, as the number of components increases the computational time increases accordingly.

The efforts for rapid flash calculation have been presented in various studies. The reduced flash calculation by decomposing the binary interaction parameters (BIPs) into two parts using a simple quadratic expression shows speedup by a factor of 3 to 20 over conventional flash calculations, depending on the number of components (Li and Johns 2006). For non-intrusive methods, data-driven model reduction has been applied in material balance type modeling such as the capacitance resistance model (CRM) (Yousef et al. 2005; Weber et al. 2009), artificial neural networks (ANN), and surrogate reservoir models (SRMs) (Mohaghegh et al. 2012). These methods are not based on the physics of the problem therefore, they usually require significant amount of dataset and time to train the model. Even if one can have a well-structured model with reliable dataset, there still exists a possibility of obtaining the unexpected solutions. However, recent advancement of technology on machine learning enables us to train the model more efficiently and rapidly with providing highly accurate solutions.

In compositional models, machine learning techniques such as ANN can be utilized for the rapid simulations that accelerate calibrating of phase stability test and phase splitting during expensive computations. Neural networks predict the vapor/liquid equilibrium K values for light hydrocarbon mixtures (Habiballah et al. 1996). ANN can accelerate flash calculation by predicting the stability for phase stability test and by providing a good initial guess for phase splitting calculations without the need to iteratively solve the nonlinear equations which are computationally expensive (Gaganis and Varotsis 2012; Wang et al. 2019). Proxy flash calculation using deep neural network enables to speed up the time-consuming flash calculation in tight oil and shale gas reservoirs (Wang et al. 2019). The schematic diagram in Figure 1.5 shows that the data-driven flash calculation can accelerate the conventional flash calculation in new control cases, however it still requires computationally expensive material balance.

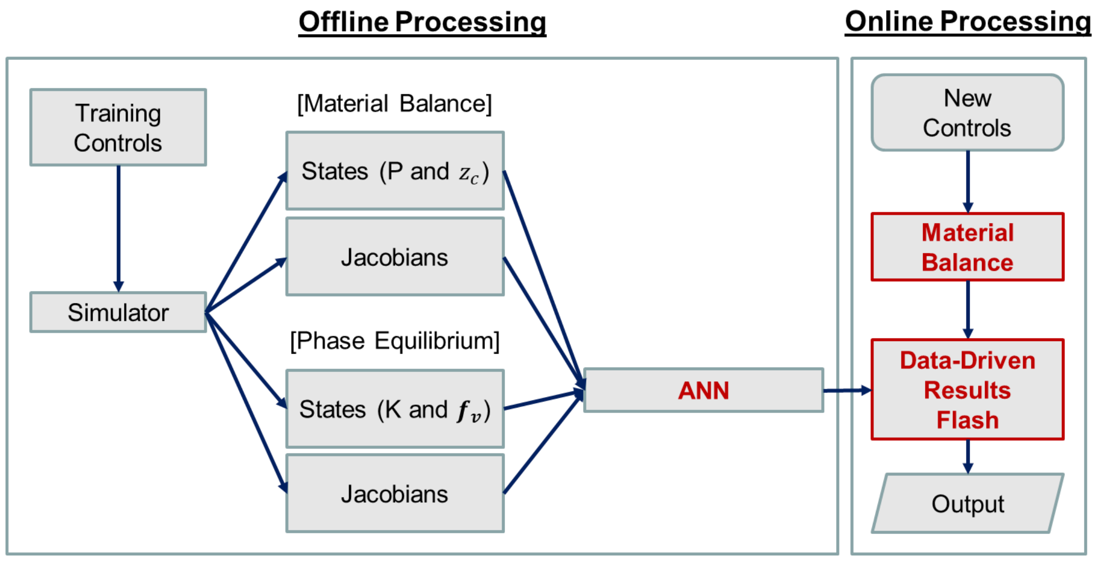


Figure 1.5 Data-driven MOR: Compositional models. ANN-based MORs that are automated in offline accelerate the conventional flash calculation, however computationally expensive material balance is still required.

The proposed method in this study aims at reducing the system of material balance equations using physics-based MOR and, simultaneously, simplifying the phase equilibrium calculation using ANN, as shown in Figure 1.6. Specifically, the POD-DEIM reduces the computational cost of the mass balance calculation using a reduced subspace by POD and a selected small set of gridblocks for evaluating nonlinear terms by DEIM. In addition, ANN accelerates the process of solving the states with respect to the flash calculation. Consequently, the online processing for the new controls can be more efficient and, thus faster.

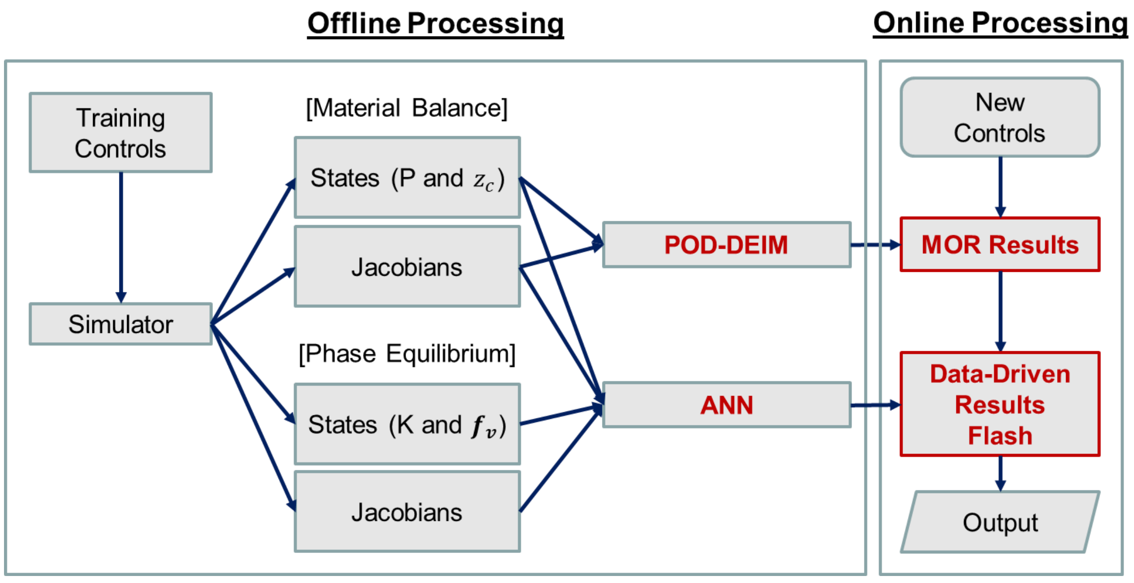


Figure 1.6 Physics- and ANN-based MOR: Compositional models. Combining physics and ANN-based MORs can speed up in both material balance and flash calculation.

1.3. Research Scope

To sum up, two main reasons have motivated us to this study. One is that the POD-DEIM has only been validated to the black oil models, and to the best of our knowledge, has not been applied to true compositional models; and the other is that the machine learning techniques can enhance the ability of the flash calculations in compositional models to achieve further speedups when integrated with MOR. To this end, the principal objectives of this thesis are to:

- Develop a multi-phase, multi-component reservoir simulator in heterogeneous porous media based on the concept of a modified molar variables formulation;
- Develop a new framework to implement physics-based reduced order modeling using the POD-DEIM for faster flow compositional simulation tracking components individually;
- Create a stand-alone ANN model to estimate the values related to the flash calculation;
- Develop a new framework to embed the ANN model into the POD-DEIM applied simulator for further speedups.

1.4. Thesis Outline

In this study, the proposed MOR technique combines POD with DEIM for the compositional model to reduce the dimension of the system and computational time.

This thesis is organized as follows: In Chapter 2, we first introduce the characteristics of the compositional reservoir simulation and present a developed compositional simulator based on a modified molar variables formulation. In Chapter 3, the proposed MOR techniques, which are POD and DEIM will be presented. In order to apply POD-DEIM to the compositional simulation, we show the modification of the typical compositional model by using primary and secondary variables. Numerical examples will be followed to demonstrate the performance of the proposed methodology. Chapter 4 introduces the basic concepts of machine learning technique using ANN, and the development of the ANN-based flash calculation is described. The applications using POD-DEIM with ANN-based flash calculation are followed. Finally, in the very last chapter we discuss the conclusions and suggestions for future work.

2. COMPOSITIONAL RESERVOIR SIMULATION

In general, methods for compositional reservoir modeling can be divided into two categories: mass balance methods and volume balance methods. Mass balance method is based on the mass conservation of hydrocarbon components and water, and it utilizes the hydraulic diffusivity equation to account for the mass transport between the grid blocks. On the other hand, volume balance method is based on the premise that pore is filled with hydrocarbon and water. Since pore volume is only a function of pressure, formulation for volume balance method is expressed as a function of pressure and compositions for setting the total fluid volume.

In this study, we focus on the mass balance method to build a multi-phase, multi-component reservoir model. More details about formulating the underlying equations for the fully implicit compositional reservoir simulator in fine scale (full-order model) are described in this chapter.

2.1. Model Assumptions

To have a simulation simple while satisfying the underlying physics, one can make assumptions in relation to the reservoir formulation as below.

- There are oil, gas, and water phases in the porous media.
- Reservoir has no flow boundaries.
- Rock is slightly compressible.

- Gravity effect is considered.
- We have n_c number of hydrocarbon components in the model.
- For the Equation of State (EOS), vapor-liquid equilibrium (VLE) by Peng-Robinson equation of state (PR-EOS) is used.
- No interaction is assumed between water and hydrocarbon components in VLE calculation.
- There are no chemical reactions or adsorption between components.
- Reservoir is isothermal.

Simulations in this study are based on the finite difference method (FDM) with discretization in space and time. The fully implicit reservoir simulator is developed, and the system of equations are given to obtain the solutions at each iteration and at each time step. Here, we build on the work done by Valbuena Olivares (2015).

2.2. Multi-Phase Flow Equations in Porous Media

The system of equations in a compositional simulation has a total of $3n_c + 15$ unknown variables (Table 2.1), and these unknown variables are solved using constitutive governing equations which are mass balance, momentum conservation, and equation of state. The related auxiliary equations support the governing equations and explain the additional physics occurring in the given simulation model. Furthermore, the total number of variables can be reduced by introducing alternate formulations, so that one can have more compact simulator having less numerical complexity.

Table 2.1 Unknown variables per gridblock.

Unknown Variable	Description	Number
p_α	Phase pressure	3
S_α	Phase saturation	3
z_i	Total mole composition	n_c
x_i	Liquid phase mole composition	n_c
y_i	Gas phase mole composition	n_c
ρ_α	Phase density	3
μ_α	Phase viscosity	3
k_α	Phase permeability	3
		Total $\boldsymbol{3n_c + 15}$

The general multi-phase flow equations are derived from two conservation laws, which are mass conservation equation and momentum conservation equation, and they are reviewed in this section.

2.2.1. Mass Conservation Equation

A mass conservation equation or a continuity equation (also called mass balance or material balance) explains the conservation of mass in the physical system by accounting for that the rate change of mass in a material is zero. In other words, if mass is to be conserved, the accumulation of mass inside the material (control volume) should be equal to the net influx of mass through the surface (Eq. 2.1).

$$\begin{aligned}(Mass\ flow\ into\ element) - (Mass\ flow\ out\ of\ element) \\ = (Rate\ of\ change\ of\ mass\ inside\ the\ element) \\ + (Source/Sink)\end{aligned} \tag{2.1}$$

This relationship can be written as follows considering an element of porous material, where we have a single-phase flow through a unit of control volume (Eq. 2.2).

$$\frac{\partial \rho}{\partial t} = \nabla \cdot \left(\rho \vec{V}\right) + \dot{m}_{V_{\frac{s}{s}}} = 0 \tag{2.2}$$

Where,

ρ: fluid density, [lb/ft^3]

t: time, day

$\vec{V}$: fluid velocity, [ft/day]

$\dot{m}_{V_{\frac{s}{s}}}$: well sink/source, [lb/day/ft^3]

In case of compositional models having multi-component, one needs to define mass fraction of a component in a phase. And mass balance of hydrocarbon components can be expressed using moles of components relationship instead of using mass relationship, as shown in Eq. 2.3.

$$\frac{\partial \tilde{\rho}_i}{\partial t} = \nabla \cdot \left(\tilde{\rho}_i \vec{V}\right) + \dot{n}_{iV_{\frac{s}{s}}} = 0 \tag{2.3}$$

Where,

$\tilde{\rho}_i$: molar density of component i, [lbmol/ft^3]

t: time, day

$\vec{V}$: fluid velocity, [ft/day]

$\dot{n}_{iV_{\frac{s}{s}}}$: sink/source, [lb/day/ft^3]

2.2.2. Momentum Conservation Equation

Momentum is defined as the multiplication of the mass and its velocity. And the conservation of momentum states that the momentum of an isolated system is a constant, which means that the vector sum of the momentum in a system cannot be changed by interactions.

In multi-phase flow in porous media, the momentum equation is expressed by Darcy's law (Eq. 2.4).

$$\vec{u}_\alpha = -\beta \frac{k_{r\alpha}}{\mu_\alpha} \vec{\vec{k}} (\nabla \Phi_\alpha) \tag{2.4}$$

Where,

$\vec{u}_\alpha$: velocity of phase α, [ft/day]

β: conversion factor for field units, 0.00633

$k_{r\alpha}$: relative permeability of phase α, [dimensionless]

μ_α: viscosity of phase α, [cP]

$\vec{\vec{k}}$: permeability tensor, [mD]

Φ_α: potential of phase α, [psia]

In multi-phase flow model, the momentum conservation and mass conservation equations are combined with a set of transport equations, and it forms a general multi-phase flow equation. The differential form of this flow equations yields hydrocarbon hydraulic diffusivity equation as shown in Eq. 2.5.

$$\nabla \cdot \left[\beta_c \vec{\vec{k}} A \left(x_i \frac{k_{ro}}{\mu_o} \nabla \Phi_o + y_i \frac{k_{rg}}{\mu_g} \nabla \Phi_g \right)\right] = V_b \frac{\partial}{\partial t}[\phi F_i] + \dot{n}_i, i = 1 to\, n_c \qquad (2.5)$$

Where,

$k_{ro}\ and\ k_{ro}$: oil and gas relative permeability, [dimensionless]

$\mu_o\ and\ \mu_g$: oil and gas viscosity, respectively, [cP]

$x_i\ and\ y_i$: liquid and vapor molar fractions of component i, respectively, [lbmol/lbmol]

$\vec{\vec{k}}$: rock permeability tensor, [mD]

Φ_o: oil potential, [psia]

Φ_g: gas potential, [psia]

A: area perpendicular to flow direction, [ft^2]

V_b: gridblock rock bulk volume, [ft^3]

ϕ: rock porosity, [ft^3/ ft^3]

F_i: number of moles of component i per unit pore volume, [lbmol/ ft^3]

$\dot{n}_i$: molar rate of component i from a well, [lbmol/ day]

n_c: the number of hydrocarbon components

β_c = 0.00633: the conversion constant for field units

Eq. 2.6 describes the differential form of the flow equation for water hydraulic diffusivity.

$$\nabla \cdot \left[\beta_c \vec{\vec{k}} A \frac{k_{rw}}{\mu_w} \nabla \Phi_w\right] = V_b \frac{\partial}{\partial t}[\phi \mathrm{W}] + \rho_w q_w \qquad (2.6)$$

Where,

k_{rw}: water relative permeability, [dimensionless]

μ_w: water viscosity, [cP]

$\vec{\vec{k}}$: rock permeability tensor, [mD]

Φ_w: water potential, [psia]

A: area perpendicular to flow direction, [ft^2]

V_b: gridblock rock bulk volume, [ft^3]

ϕ: rock porosity, [ft^3/ ft^3]

W: mass of water rate from a well, [ft^3/ day]

ρ_w: water density, [lb/ft^3]

β_c = 0.00633: the conversion constant for field units

2.2.3. Energy Conservation Equation

The equation for energy balance can be added when the thermal processes affect the system of flow equations, and a variable for total energy or temperature need to be included to describe the impact of the thermal treatment. The detailed energy conservation equation can be found in the thermal recovery literatures (Prats 1982; Green and Wilhite 1998; Fanchi 2006).

Since consideration of the energy balance equation imposes additional nonlinear terms to the system, it requires more computationally intensive solutions with substantial additional complexities. Therefore, in many realistic and practical systems, thermal processes are neglected by assuming isothermal approximation. In this study, we also assume the isothermal condition, and therefore the energy conservation term or variable is not included in the overall framework of the compositional simulation.

2.3. Constraint Equations

2.3.1. Phase Equilibrium Relationships

Equation of state (EOS) is a mathematical relationship of state variables such as pressure, volume, temperature at a given set of physical conditions. EOS is utilized in describing compositions of fluid mixture and in properties of fluids. There are several different forms of cubic EOS, for example Van der Waals EOS, Redlich-Kwong EOS (RK EOS), Soave-Redlich-Kwong EOS (SRK EOS) (Redlich and Kwong 1949; Soave 1972). In this study, we use Peng-Robinson EOS (PR EOS) to compute compositions and properties of fluid. The PR EOS was developed in 1976 as a modification of Van der Waals EOS (Peng and Robinson 1976). The advantage of PR EOS is that it enables us to have more accurate predictions of the volumetric behavior of the coexisting phases in vapor-liquid equilibrium (VLE) calculations with the improved liquid density values and equilibrium ratios, comparing to the SRK EOS. VLE describes the distribution of a chemical components between the vapor phase and the liquid phase. Through VLE calculation, one can get the concentration of phases as well as the properties of fluid such as molecular weight, density, and others. Phase equilibrium relationships such as VLE can be expressed in terms of the fugacities or K values as shown in Eq. 2.7 through Eq. 2.8.

$$f_{oi} = f_{gi}, i = 1, \cdots, n_c \tag{2.7}$$

$$y_i = K_i x_i, i = 1, \cdots, n_c \qquad (2.8)$$

Where,

f_{oi}: fugacity of liquid phase of component i

f_{gi}: fugacity of vapor phase of component i

x_i: liquid molar fraction of component i

y_i: vapor molar faction of component i

K_i: equilibrium ratio of component i

The constraint equations for compositional models arise from the requirement that the sum of the mass or mole fraction in each phase should be unity, as shown in Eq. 2.9 through Eq. 2.11.

$$\sum_{i=1}^{n_c} x_i = 1 \qquad (2.9)$$

$$\sum_{i=1}^{n_c} y_i = 1 \qquad (2.10)$$

$$\sum_{i=1}^{n_c} z_i = 1 \qquad (2.11)$$

More detailed descriptions on the phase equilibrium calculation and PR EOS are presented in chapter 4.

2.3.2. Saturation Equation

The constraints equation on saturation ensures the pore space is completely filled with fluids and summation of each phase saturations is unity (Eq. 2.12).

$$S_o + S_g + S_w = 1 \tag{2.12}$$

Here, S_o, S_g, and S_w are the saturation of oil, gas and water. The overall composition can be defined as presented in Eq. 2.13.

$$z_i = (1 - v)x_i + vy_i, i = 1, \cdots, n_c \tag{2.13}$$

Therefore, the above saturation constraint equation can be rewritten as shown in Eq. 2.14.

$$1 - F\left[\frac{(1 - v)}{\rho_o} + \frac{v}{\rho_g}\right] - \frac{W}{\rho_w} = 0 \tag{2.14}$$

2.4. Types of Compositional Simulation

Over the decades, various types of compositional models have been developed. Several most widely used formulations for general compositional reservoir simulation are introduced and compared in this section (Voskov and Tchelepi 2012). They are essentially based on the EOS and the fully implicit method with the full Jacobian matrix as a function of primary and secondary variables to solve the nonlinear system.

Natural variables formulation (Coats 1980) is one of the most well-known isothermal compositional simulation using the unknown variables of pressure, phase saturation, phase composition of each component.

Molar variables formulation (Young and Stephensen 1983) uses pressure, phase fraction, phase composition for gas phase, and the overall composition for each component as the unknown variables. The unknowns are obtained similarly in the way of solving the natural variables formulation except phase density and phase saturation calculation. The advantage of molar formulation is that it is independent of the appearance or disappearance of a particular hydrocarbon phase.

Another molar variable formulation (Chien et al 1985) is a variant of the original molar variables method in which the equilibrium ratio, K_i is introduced instead of the phase fraction, x_i. This formulation was more linearized by Wang et al. (1997) by using $\ln K_i$ instead of K_i in the thermodynamic equilibrium equation. The unknown variables for the formulation are pressure, phase fraction, natural logarithm of phase equilibrium ratio, and the overall phase composition of each component.

Overall composition variables formulation (Collins et al. 1992) uses pressure, and composition of each component for the unknowns. This formulation is essentially identical to the molar formulation if instantaneous thermodynamic equilibrium is assumed. But the advantage of this method is that every gridblock has the consistent form of variables and equation, therefore solutions can be achieved in more concise manner.

Finally, the volume balance formulation (Acs et al. 1985) uses molar variables instead of overall composition variables.

In this study, we utilize a fully implicit method with a modified molar variables formulation which is composed of the primary and the secondary variables, as a base case. We adapt the formulation given by Wang et al. (1997) with a modification.

2.5. Choice of Independent Variables

Since we assume three phases (oil, water, gas) and multi-component reservoir model in this study, the system of equations are functions of pressure, phase variables, and component variables. As depicted in Table 2.1, a total $3n_c + 15$ unknown variables need to be included in the reservoir system (see section 2.2 for more details). There are various ways to select the type and the number of independent variables to form a system of equations depending on the chosen methodology. In this study, the way of selecting the reservoir variables follows Wang et al. (1997)'s molar variables formulation, which is also a variant of conventional Young and Stephenson's and Chien et al.'s method. As stated in section 2.4, the advantage of molar variables formulation is that it is not affected by the appearance or disappearance of a particular hydrocarbon phase. Furthermore, using $\ln K_i$ instead of K_i makes the thermodynamic equilibrium equation more linear.

In particular, there is an additional reason why we choose a modified molar variables formulation of Wang et al. in this study. The application of the MOR methods introduced in the following chapter, is based on the solution of the state variables in the system. Therefore, molar formulation is well suitable for applying the MOR method considered here since it maintains the variables regardless of the phase change. More details about the concept of MOR method and its application to the molar variables based compositional model will be discussed in a later chapter.

Table 2.2 displays that the system consists of $2n_c + 3$ independent variables for each grid blocks used in this study.

Table 2.2 Independent variables for reservoir system of equations.

Variable	Description	Number
$\ln K_i$	Natural logarithm of equilibrium ratio	n_c
p_o	Oil-phase pressure	1
F_i	Moles of component i per pore volume	n_c
W	Mass of water per pore volume	1
f_v	Molar vapor fraction	1
		Total $2n_c + 3$

As seen in Tables 2.1, the total number of variables to be solved for each discretized gridblock aggregates to $3n_c + 15$. This is due to the following. The $n_c + 12$ unknown variables and equations in addition to $2n_c + 3$ independent variables and equations are required to construct a full system comprised of $3n_c + 15$, so the auxiliary equations are employed to complete it.

2.6. Residual Equations

The numerical solution of the partial differential equations described in Eq. 2.5 and 2.6 are usually obtained by the finite differences (volume) discretization method. The reader is referred to Aziz and Settari (1986), Ertekin et al. (2001) for a complete mathematical derivation of the discretized equations. Here, it suffices to say that given the solution of the nonlinear equations by the Newton-Raphson method, we need to obtain the residual and Jacobian matrices to be used in the solver portion of the numerical solution (see section 2.10 for more details). The residual equations can be derived from differential material balances describing component flow, phase equilibrium, and constraints equations that are explained in section 2.2 and 2.3.

Material balance equations are discretized in time and space by use of finite difference, specifically space is discretized using central difference in the block-centered geometry with 6-directions (E-east, W-west, N-north, S-south, T-top, B-bottom), and time discretization is conducted by backward difference for fully implicit method.

Residual equations for hydrocarbon component and water are shown in Eq. 2.15 and Eq. 2.16.

$$R_i = \frac{V_b}{\Delta t}\left[F_i^n \emptyset'(p_{oC}^{n+1} - p_{oC}^n) + \emptyset^{n+1}(F_i^{n+1} - F_i^n)\right]$$
$$- \sum_{\eta=1}^{6}\left[a_{o\eta}^{n+1} x_i^{n+1}(\Delta p_{o\eta} - \gamma_{o\eta}^{n+1}\Delta z_\eta)\right]$$
$$- \sum_{\eta=1}^{6}\left[a_{g\eta}^{n+1} y_i^{n+1}(\Delta p_{o\eta} - \gamma_{g\eta}^{n+1}\Delta z_\eta)\right] \tag{2.15}$$
$$- \sum_{\eta=1}^{6}\left[(a_{g\eta} y_i p'_{cgo_\eta} S'_{g\eta})^{n+1}\Delta F_i\right]$$
$$+ WI_o^{n+1} x_i^{n+1}\left(p_{oC}^{n+1} - p_{wf}^{n+1}\right)$$
$$+ WI_g^{n+1} y_i^{n+1}\left(p_{oC}^{n+1} + p_{cgo}^{n+1} - p_{wf}^{n+1}\right)$$

$$R_w = \frac{V_b}{\Delta t}\left[W^n \phi'(p_{oC}^{n+1} - p_{oC}^n) + \phi^{n+1}(W^{n+1} - W^n)\right]$$
$$- \sum_{\eta=1}^{6}\left[a_{w\eta}^{n+1}\left(\Delta p_{o\eta}^{n+1} - \gamma_{w\eta}^{n+1}\Delta z_\eta\right) - \frac{a_{w\eta}^{n+1} p'^{(n+1)}_{cow_\eta}}{\rho_{w\eta}^{n+1}}\Delta W_\eta^{n+1}\right] \tag{2.16}$$
$$+ WI_w^{n+1}\left(p_{oC}^{n+1} - p_{cow}^{n+1} - p_{wf}^{n+1}\right)$$

Where,

R_i: hydrocarbon component residual, [lbmol/day]

R_w: water residual, [lb/day]

Δt: size of time step, [day]

$\emptyset'$: porosity time chord-slope, [1/psi]

p_o: oil phase pressure, [lbmol/day]

$a_{\alpha\eta}$: transmissibility of phase α between central gridblock and neighboring gridblock in η direction, [lbmol/day/psi]

p_c': capillary pressure spatial chord-slope, [psi]

S'_g: gas saturation spatial chord-slope, [fraction]

WI_α: well index of phase α, [lbmol/day/psi]

p_{wf}: bottomhole flowing pressure, [psia]

n: current time level [dimensionless]

$n+1$: next time level [dimensionless]

Phase transmissibility $a_{\alpha\eta}$ in each residual equation is defined as the product of the geometric interblock transmissibility and phase mobility between center gridblock and a neighbor gridblock in η flux direction (Eq. 2.17).

$$a_{\alpha\eta} = T_\eta \lambda_{\alpha\eta} = T_\eta \left(k_{r\alpha}\frac{\tilde{\rho}_\alpha}{\mu_\alpha}\right)_\eta, \eta = E, W, N, S, B, T \tag{2.17}$$

The reader is referred to section 2.6 for the definition of the set (E-east, W-west, N-north, S-south, T-top, B-bottom). In addition, capillary pressure needs to be considered in the residual equations, however since it is expressed as a function of saturation which is not the primary independent variable in the system, some treatment is required to rewrite it with the concept of spatial chord-slope of capillary pressure and mass (mole) of phase (component). Eq. 2.18 and Eq. 2.19 are the examples of chord-slope for capillary pressure.

$$p'_{cow_\eta} = \frac{p_{cow_\eta} - p_{cow_c}}{S_{w_\eta} - S_{w_C}} \tag{2.18}$$

$$p'_{cgo_\eta} = \frac{p_{cgo_\eta} - p_{cgo_C}}{S_{g_\eta} - S_{g_C}} \tag{2.19}$$

The detailed process of discretization for hydrocarbon component and water are referred in Appendix A.

Component fugacity residual is one of the equations related to VLE. Thermodynamic equilibrium or VLE for a compositional fluid is reached when the phase fugacities for each component are equal. When it comes to the chemical potential concept, this implies that the molecular transfer rate from liquid to vapor phase equals the molecular transfer rate from vapor to liquid phase for all components (Valbuena Olivares 2015). Therefore, VLE allows us to estimate compositions of fluid mixtures and fluid properties such as fugacities. Since the equilibrium ratio K_i is the ratio of vapor molar fraction to liquid molar fraction, which can also be rewritten as the ratio of fugacity coefficient of vapor phase to fugacity coefficient of liquid phase, final form of fugacity residual is expressed as follows (Eq. 2.20).

$$R_{fug_i} = \ln K_i + \ln \hat{\varphi}_i^v - \hat{\varphi}_i^l \ ; \ \ i = 1 \ to \ n_c \tag{2.20}$$

Where,

$\hat{\varphi}_i^v$: fugacity coefficient of component i in vapor phase

$\hat{\varphi}_i^l$: fugacity coefficient of component i in liquid phase

Saturation constraint residual implies that the summation of saturations of all fluids which fill the pore of rock should be unity. The following residual equation in Eq. 2.21 explains the pore is fully saturated with oil, gas, and water.

$$R_{sat} = F\left[\frac{1-f_v}{\tilde{\rho}_o} + \frac{f_v}{\tilde{\rho}_g}\right] + \frac{W}{\tilde{\rho}_w} - 1 \tag{2.21}$$

Residual for correct molar vapor fraction by the Rachford-Rice (1952) is another residual equation arises from VLE (Eq. 2.22). This residual is used to calculate the vapor molar fraction f_v for a mixture of composition z_i at known K values.

$$R_{RR} = \sum_{i=1}^{n_c} \frac{z_i(K_i - 1)}{1 + f_v(K_i - 1)} \tag{2.22}$$

2.7. Auxiliary Equations

As mentioned in section 2.2 and 2.5, there are essentially $3n_c$+15 unknown variables in the system, however the use of auxiliary relationships can reduce the number of independent variables, resulting in reducing the complexity of the system with a smaller number of nonlinear equations. Several auxiliary – or supplemental – equations are specified in this section.

2.7.1. Rate Equations

In general, a reservoir simulator needs to include rate equations for modeling wells and phase potential calculation. For the wells modeling that represents the fluid flow between well and gridblocks, Peaceman's equation is utilized in this study with the use of well index (WI) concept (Peaceman 1978). Rate equation is written as Eq. 23 using well index, which is defined as the multiplication of $WI_{geometry}$ and WI_{fluid}.

$$WI_{\alpha} = WI_{geometry} WI_{fluid} \tag{2.23}$$

$WI_{geometry}$ is a constant that doesn't change in the process of simulation and can be pre-computed, and WI_{fluid} is represented by dynamic phase mobility for the perforated gridblock (Eq. 2.24).

$$WI_{\alpha} = WI_{geometry} \lambda_{\alpha C} \tag{2.24}$$

Calculation of geometric well index for a vertical well is shown in Eq. 2.25 through Eq. 2.26.

$$WI_{geom} = 2\pi \beta_c \frac{\Delta z \sqrt{k_x k_y}}{\ln\left(\frac{r_o}{r_w}\right) + S} \tag{2.25}$$

With,

$$r_o = 0.28 \frac{\left(\Delta x^2 \sqrt{\frac{k_y}{k_x}} + \Delta y^2 \sqrt{\frac{k_x}{k_y}}\right)^{\frac{1}{2}}}{\sqrt[4]{\frac{k_y}{k_x}} + \sqrt[4]{\frac{k_x}{k_y}}} \tag{2.26}$$

Where,

Δx, Δy, Δz: gridblock dimensions, [ft]

k_x, k_y, k_z: permeability in each coordinate direction, [mD]

r_w: wellbore radius, [ft]

S: well skin, [dimensionless]

Dynamic phase mobility for a producing well is computed using fluid properties and relative permeability of the perforated gridblocks (Eq. 2.27), and the calculation of dynamic phase mobility for an injection well is based on total fluid mobility, as shown in Eq. 2.28 (Schlumberger, 2014).

$$\lambda_{\alpha C_{Producer}} = \left(k_{r\alpha} \frac{\tilde{\rho}_\alpha}{\mu_\alpha}\right)_C \tag{2.27}$$

$$\lambda_{\alpha C_{Injector}} = \left(\frac{k_{ro}}{\mu_o} + \frac{k_{rg}}{\mu_g} + \frac{k_{rw}}{\mu_w}\right)_C \tilde{\rho}_{\alpha C} \tag{2.28}$$

Consequently, the rate equation can be expressed as water mass rate and hydrocarbon component molar rate by use of the well index concept, as shown in Eq. 2.29 and Eq. 2.30, respectively.

$$\dot{m}_{w_{\underline{\underline{s}}}} = \rho_w q_{w_{\underline{\underline{s}}}} = WI_w\left(p_{oC} - p_{cow} - p_{wf}\right) \tag{2.29}$$

$$\dot{n}_{i_{\underline{\underline{s}}}} = WI_o x_i\left(p_{oC} - p_{wf}\right) + WI_g y_i\left(p_{oC} + p_{cgo} - p_{wf}\right) \tag{2.30}$$

Another auxiliary equation for rate equations is phase potential calculation (Eq. 2.31).

$$\Phi_\alpha = p_\alpha - \gamma_\alpha z = p_\alpha - g_c \rho_\alpha z \tag{2.31}$$

Where,

p_α: pressure of phase α, [psia]

ρ_α: density of phase α, [lb/ft^3]

z: depth, [ft]

g_c: gravity constant

An example of calculation of interblock water phase potential between central and neighboring gridblocks is shown in Eq. 2.32.

$$\begin{aligned}\Delta\Phi_{w\eta} = (\Phi_{w\eta} - \Phi_{wC}) &= \Delta p_{o\eta} - \Delta p_{cow_\eta} - \gamma_{w\eta}\Delta z_\eta \\ &= (p_{o\eta} - p_{oC}) - (p_{cow_\eta} - p_{cow_C}) \\ &- g_c\left(\frac{\rho_{w\eta} + \rho_{wC}}{2}\right)(z_\eta - z_C)\end{aligned} \tag{2.32}$$

The subscripts C and η indicates central gridblock and neighboring gridblocks, respectively.

2.7.2. Rock-Fluid Interaction

Two auxiliary equations to express rock-fluid interaction are relative permeability and capillary pressure. Three-phase relative permeability is used when oil, water, and gas are flowing in a porous media simultaneously. Relative permeability curves are provided by laboratory tests, and correlations are used instead of direct measurements since three-phase relative permeabilities are difficult to measure. The correlation used in this study is based on the assumptions that the water relative permeability curve (k_{rw}) obtained for a water-oil system depends only on water saturation, and the gas relative permeability curve (k_{rg}) obtained for a gas-oil system depends only on gas saturation (Fanchi 2006). The three-phase oil relative permeability (k_{ro}) is then calculated as a function of water and gas relative permeability (Eq. 2.33).

$$k_{ro} = k_{rocw}\left[\left(\frac{k_{row}}{k_{rocw}} + k_{rw}\right)\left(\frac{k_{rog}}{k_{rocw}} + k_{rg}\right) - k_{rw} - k_{rg}\right] \tag{2.33}$$

Where,

k_{ro}: oil relative permeability

k_{rocw}: oil relative permeability at irreducible water saturation

k_{row}: oil relative permeability at actual water saturation

k_{rog}: oil relative permeability at actual gas saturation and irreducible water saturation

k_{rw}: water relative permeability at actual water saturation

k_{rg}: gas relative permeability at actual gas saturation

Eq. 2.33 is based on the methods by Stone (1973), and Dietrich and Bondor (1976). One needs to assure that the relative permeability constraints, for example $k_{row}(1 - S_{wr}) = k_{rog}(S_o + S_w = 1.0)$ for the irreducible saturation S_{wr} and $S_g = 0$, are satisfied to get the realistic values.

Capillary pressure is defined as the difference between the pressure of a non-wetting phase $p_{non-wetting}$ and the pressure of a wetting phase $p_{wetting}$, as shown in Eq. 2.34, and is used to determine initial fluid contact and transition zones of two immiscible fluids.

$$p_c = p_{non-wetting} - p_{wetting} \tag{2.34}$$

Oil-water capillary pressure and gas-oil capillary pressure can be expressed as Eq. 2.35 and Eq. 2.36 respectively.

$$p_{cow} = p_o - p_w \tag{2.35}$$

$$p_{cgo} = p_g - p_o \tag{2.36}$$

2.7.3. Other Auxiliary Equations

Saturation-dependent rock-fluid interaction data, rock compressibility equation, and viscosity of oil and gas can also be used for auxiliary equations to complement the solutions of the system. Volume translation method is also applied in this study to match the hydrocarbon phase volumetric properties obtained from EOS to the measured properties from laboratory (Péneloux et al. 1982).

2.8. Vector of Independent Variables, Residuals, and Jacobians

We use the Newton-Raphson method to solve the system of nonlinear equations in this study. Newton's method can be derived by considering the linear approximation of the nonlinear vector-valued functions, given an initial guess. Ultimately, the method yields a solution of large-scale linear systems. The initial guess for the independent variables is obtained from the solution of previous time step to form the vector of unknown variables $\vec{x}$, and the residual vector $\vec{R}$ is computed based on the vector of independent variables and boundary conditions. The Jacobian matrix $\vec{J}$ is formed by taking the partial derivatives of the residual with respect to the unknown variable. Usually numerical derivatives are used. Finally, the system of equations is solved by use of various solvers such as direct method, conjugate gradient method (CG), generalized minimum residual method (GMRES), and biconjugate gradients stabilized method (BiCGSTAB), from MATLAB® libraries (Valbuena Olivares 2015) in this study.

The vector of independent variables can be setup in any order, but in this study, it follows the ascending order of $\vec{\imath}$, $\vec{\jmath}$, $\vec{k}$, one after another, and is expressed as shown in Eq. 2.37. Eq. 2.38 presents the size of $\vec{x}$.

$$\vec{x} = \begin{bmatrix} \ln \vec{K}_{i\,\forall(\vec{\imath},\vec{\jmath},\vec{k})} \\ p_{o\,\forall(\vec{\imath},\vec{\jmath},\vec{k})} \\ \vec{F}_{i\,\forall(\vec{\imath},\vec{\jmath},\vec{k})} \\ W_{\forall(\vec{\imath},\vec{\jmath},\vec{k})} \\ f_{v\,\forall(\vec{\imath},\vec{\jmath},\vec{k})} \end{bmatrix} \tag{2.37}$$

$$size(\vec{x}) = (2n_c + 3) \times number\ of\ gridblocks \tag{2.38}$$

Similarly, the vector of residuals is structured in accordance with the order of $\vec{x}$ (Eq. 2.39). The size of $\vec{R}$ is equal to the size of $\vec{x}$ as in (Eq. 2.40).

$$\vec{R} = \begin{bmatrix} \vec{R}_{fug_i\,\forall(\vec{\imath},\vec{\jmath},\vec{k})} \\ R_{sat\,\forall(\vec{\imath},\vec{\jmath},\vec{k})} \\ \vec{R}_{i\,\forall(\vec{\imath},\vec{\jmath},\vec{k})} \\ R_{w\,\forall(\vec{\imath},\vec{\jmath},\vec{k})} \\ R_{RR\,\forall(\vec{\imath},\vec{\jmath},\vec{k})} \end{bmatrix} \tag{2.39}$$

$$size(\vec{R}) = (2n_c + 3) \times number\ of\ gridblocks \tag{2.40}$$

Jacobian matrix is the matrix of all residuals' first-order partial derivatives. There are two ways of computing Jacobian; one is numerical Jacobian and the other is analytical Jacobian. We use numerical Jacobian which is easier to implement in the numerical simulation. The size of Jacobian is shown in Eq. 2.41.

$$size\left(\vec{J}\right) = size\left(\vec{R}\right) \times \, size(\vec{x}) \tag{2.41}$$

Since $size\left(\vec{R}\right)$ equals $size(\vec{x})$, the shape of Jacobian is square as seen in Eq. 2.42.

$$\vec{J} = \left[\frac{\partial \vec{R}}{\partial \vec{x}}\right]$$

$$= \begin{bmatrix}
\frac{\partial R_{fug_1}}{\partial \ln K_1} & \cdots & \frac{\partial R_{fug_1}}{\partial \ln K_{n_c}} & \frac{\partial R_{fug_1}}{\partial p_o} & \frac{\partial R_{fug_1}}{\partial F_1} & \cdots & \frac{\partial R_{fug_1}}{\partial F_{n_c}} & \frac{\partial R_{fug_1}}{\partial W} & \frac{\partial R_{fug_1}}{\partial f_v} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\frac{\partial R_{fug_{n_c}}}{\partial \ln K_1} & \cdots & \frac{\partial R_{fug_{n_c}}}{\partial \ln K_{n_c}} & \frac{\partial R_{fug_{n_c}}}{\partial p_o} & \frac{\partial R_{fug_{n_c}}}{\partial F_1} & \cdots & \frac{\partial R_{fug_{n_c}}}{\partial F_{n_c}} & \frac{\partial R_{fug_{n_c}}}{\partial W} & \frac{\partial R_{fug_{n_c}}}{\partial f_v} \\
\frac{\partial R_{sat}}{\partial \ln K_1} & \cdots & \frac{\partial R_{sat}}{\partial \ln K_{n_c}} & \frac{\partial R_{sat}}{\partial p_o} & \frac{\partial R_{sat}}{\partial F_1} & \cdots & \frac{\partial R_{sat}}{\partial F_{n_c}} & \frac{\partial R_{sat}}{\partial W} & \frac{\partial R_{sat}}{\partial f_v} \\
\frac{\partial R_1}{\partial \ln K_1} & \cdots & \frac{\partial R_1}{\partial \ln K_{n_c}} & \frac{\partial R_1}{\partial p_o} & \frac{\partial R_1}{\partial F_1} & \cdots & \frac{\partial R_1}{\partial F_{n_c}} & \frac{\partial R_1}{\partial W} & \frac{\partial R_1}{\partial f_v} \\
\vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\
\frac{\partial R_{n_c}}{\partial \ln K_1} & \cdots & \frac{\partial R_{n_c}}{\partial \ln K_{n_c}} & \frac{\partial R_{n_c}}{\partial p_o} & \frac{\partial R_{n_c}}{\partial F_1} & \cdots & \frac{\partial R_{n_c}}{\partial F_{n_c}} & \frac{\partial R_{n_c}}{\partial W} & \frac{\partial R_{n_c}}{\partial f_v} \\
\frac{\partial R_w}{\partial \ln K_1} & \cdots & \frac{\partial R_w}{\partial \ln K_{n_c}} & \frac{\partial R_w}{\partial p_o} & \frac{\partial R_w}{\partial F_1} & \cdots & \frac{\partial R_w}{\partial F_{n_c}} & \frac{\partial R_w}{\partial W} & \frac{\partial R_w}{\partial f_v} \\
\frac{\partial R_{RR}}{\partial \ln K_1} & \cdots & \frac{\partial R_{RR}}{\partial \ln K_{n_c}} & \frac{\partial R_{RR}}{\partial p_o} & \frac{\partial R_{RR}}{\partial F_1} & \cdots & \frac{\partial R_{RR}}{\partial F_{n_c}} & \frac{\partial R_{RR}}{\partial W} & \frac{\partial R_{RR}}{\partial f_v}
\end{bmatrix} \tag{2.42}$$

2.9. Compositional Simulation Workflow

The flow chart in Figure 2.1 displays the overall compositional simulation workflow used in most commercial software. Firstly, input data is read, and then the parallel computing is set up. Parallel computing can accelerate the computations by breaking down larger problems into smaller problems and computing them using multiple processors simultaneously. Miscellaneous calculations such as geometric transmissibility calculation and well index calculation are performed. After the initialization of independent variables using the given initial conditions, time steps for fully implicit method using Newton-Raphson iteration are solved until final simulation time. There are several subroutines in the workflow, and one example is the flash calculation in Newton-Raphson iteration. The detailed procedure for the flash calculation is discussed in Chapter 4.

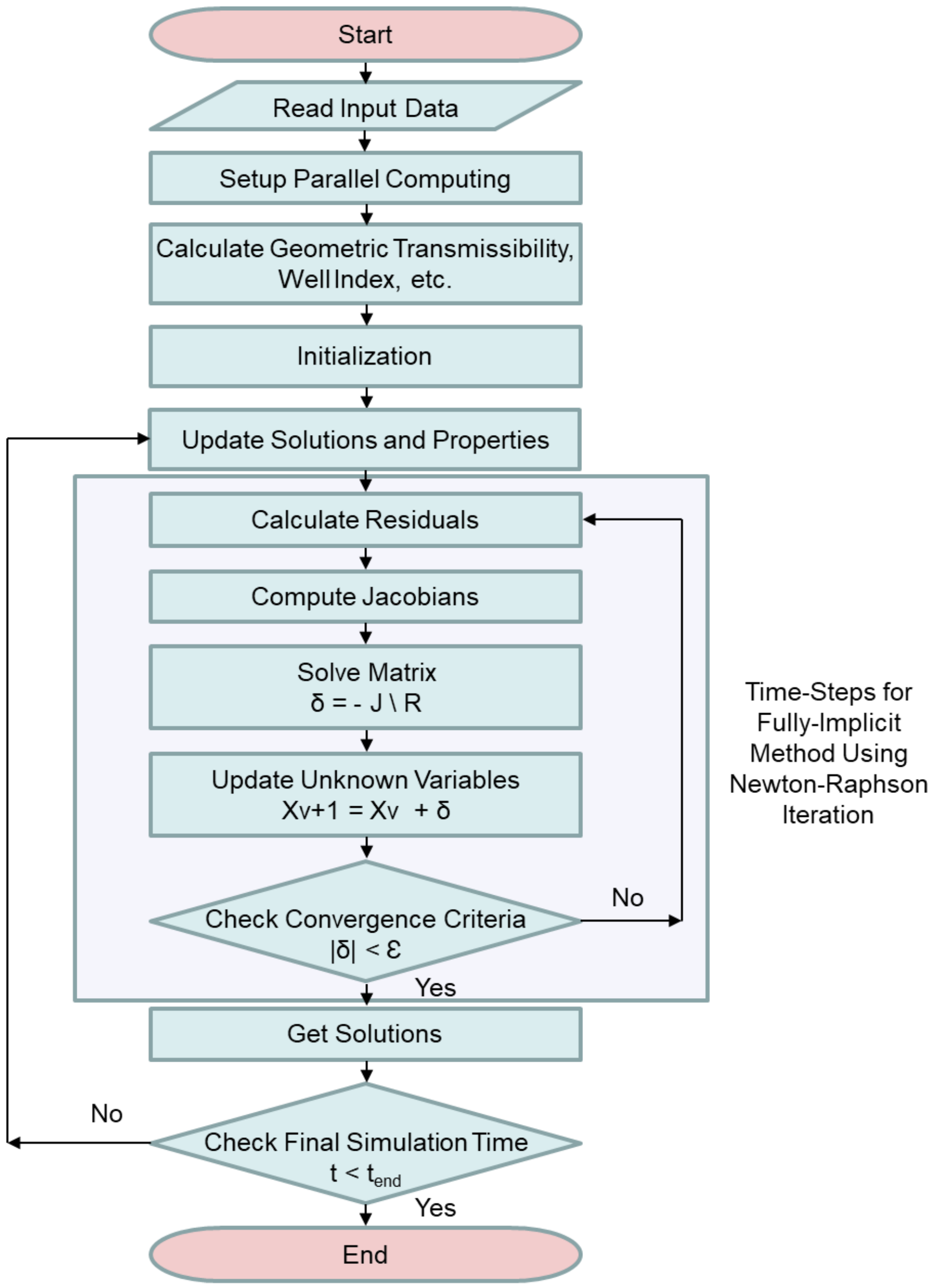

Start
Read Input Data
Setup Parallel Computing
Calculate Geometric Transmissibility, Well Index, etc.
Initialization
Update Solutions and Properties
Calculate Residuals
Compute Jacobians
Solve Matrix $\delta = - J \backslash R$
Update Unknown Variables $X_{v+1} = X_v + \delta$
Check Convergence Criteria $|\delta| < \varepsilon$
No
Yes
Time-Steps for Fully-Implicit Method Using Newton-Raphson Iteration
Get Solutions
Check Final Simulation Time $t < t_{end}$
No
Yes
End

Figure 2.1 Compositional simulation workflow adapted from the work done by Valbuena (2015). Time steps for fully implicit method using Newton-Raphson iteration are solved until final simulation time. Several subroutines such as flash calculation is included in the process of time step.

2.10. Matrix Solvers

There are multiple ways to solve the matrix-type system of equations. One can use direct methods which is expensive for large-scale computations, or iterative methods which is relatively cheaper than direct solvers. Examples of Krylov subspaces in iterative methods are CG (Hestenes and Stiefel 1952; Saad 2003), GMRES (Saad and Schultz 1986; Saad 2003), and BiCGSTAB (van der Vorst 1992). The text by Chen (2005) describes several preconditioning techniques used in conjunction with iterative methods. In this study, we use a direct solver (LU Decomposition) as a default matrix solver, that usually requires less than 5 iterations to convergence. This selection is purely based on the size of problems studied here, but implementation of iterative methods is the way to go for larger problems.

2.11. Physical Limits and Convergence Criteria

Newton-Raphson iteration stops when the error meets the convergence criteria. In order to get the reliable and acceptable solution from the iteration process, we use multiple physical limits and convergence criteria as follows.

$$p > 14.7 \tag{2.43}$$

$$0 \leq S_\alpha \leq 1 \tag{2.44}$$

$$\left\|\vec{R}\right\|_2 \rightarrow 0 \tag{2.45}$$

$$\|p^{k+1} - p^k\|_2 \leq \varepsilon_p, \varepsilon_p = 1.47 \tag{2.46}$$

$$\left\|\hat{f}_i^{l^{k+1}} - \hat{f}_i^{l^k}\right\|_2 \leq \varepsilon_f, \varepsilon_f = 0.001 \tag{2.47}$$

$$\left\|\hat{f}_i^{v^{k+1}} - \hat{f}_i^{v^k}\right\|_2 \leq \varepsilon_f, \varepsilon_f = 0.001 \tag{2.48}$$

Note that the convergence criteria for the norm of pressure change, phase fugacity change is referred from Schlumberger (2014).

3. PROPOSED NEW WORKFLOW OF PHYSICS-BASED MODEL ORDER REDUCTION FOR COMPOSITIONAL SIMULATION*

Model order reduction techniques aim to reduce the computational cost of large-scale simulation models by finding an approximation to the solution using a less complex model. We should emphasize that here, reduction in model complexity is achieved by reducing the number of variables in the system of equation, or by approximating the attributes of gridblocks in case of reservoir simulation. Furthermore, the full-order system can be approximated by either linear or nonlinear model depending on the nature of the system.

Proper Orthogonal Decomposition (POD) is one of the well-established and most fundamentally used MOR method in the field of reservoir simulation owing to its simplicity, accuracy and overall efficiency. We should note that although the POD method is endowed with many positive qualities, its robustness is not well studied to yield always convergence results in our particular application. Many variants of the method have been introduced to overcome some of its drawbacks, such as the lack of material balance guarantees, the proper selection of basis and general convergence issues. The reader is referred to Gildin et al. (2013) and Ghasemi et al. (2015) for more information.

One of the additional works introduce to extend the POD to highly nonlinear systems is the Discrete Empirical Interpolation Method (DEIM) (Chaturantabut and

*Part of this section is reprinted with permission from SPE-198946-MS "A New Framework for Compositional Simulation Using Reduced Order Modeling Based on POD-DEIM" by J. Lee and E. Gildin, July 2020, SPE Latin American and Caribbean Petroleum Engineering Conference, Society of Petroleum Engineers.

Sorensen 2010). DEIM approximates the nonlinear properties by interpolating them from certain selected gridblock locations in the reservoir domain. In this case, the Jacobian and residuals do not need to be reconstructed entirely for every Newton step, that is, only a small portion needs to be evaluated. Therefore, DEIM is efficiently utilized to reduce the complexity of the nonlinear system when combined with POD technique (Ghasemi et al. 2015). In what follows, we describe mathematically the POD-DEIM method and its modifications for the compositional simulation shown in Chapter 2.

3.1 Proper Orthogonal Decomposition (POD)

POD is a mathematical method originated from statistical analysis that uses an orthogonal transformation to convert a given set of variables to a linearly independent set of variables which is called a POD basis. So, the main idea of POD is to reduce the dimension of a given data set consisting of many variables that are correlated with each other while retaining the variation present in the dataset to the maximum extent. POD is based on the singular value decomposition (SVD) (Golub and Van Loan 1996) and it is also called principal component analysis (PCA) or other different names depending on the field of application.

POD has been investigated and successfully implemented in reservoir simulation by Cardoso et al. (2010a), Gildin et al. (2013), among others, as a way of reducing the computational cost when computing the solutions of the reservoir state (pressures and saturations). In this study, we follow the standard snapshot-based POD formulation and its main formulation and implementation is presented below.

The overall workflow starts with the run of many (3 to 4 from our experience) full-scale simulations to obtain the system's snapshots which are the solutions of the independent variables at each time step by running a high fidelity full-order model with training inputs. More explicitly, suppose that the simulation model has M gridblocks and is run for N time steps. Each snapshot can be expressed as N column vectors having length M, that is $M \times N$ matrix (Eq. 3.1 and Figure 3.1).

$$A = (a_1, \cdots, a_N) \tag{3.1}$$

Given the correlations in the saved snapshot, we then use the well-known SVD to the snapshot matrix to obtain basis for the space spanned by the snapshots, as in Eq.3.2.

$$A = U\Sigma V^T \tag{3.2}$$

In this case, the left singular vector U can be used to project the high order system into a smaller subsystem, as it can be seen from Figure 3.2. The R value in Figure 3.2 is selected based on the energy level of the states. And the reduced U matrix is used as the POD basis.

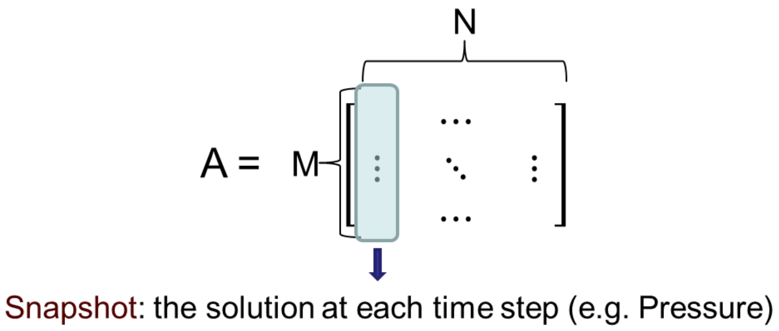


Figure 3.1 Snapshot matrix for POD. M: number of gridblock, N: number of time step. Each column of snapshot matrix indicates the solutions of unknowns at each time step. Reprinted with permission from SPE-198946-MS.

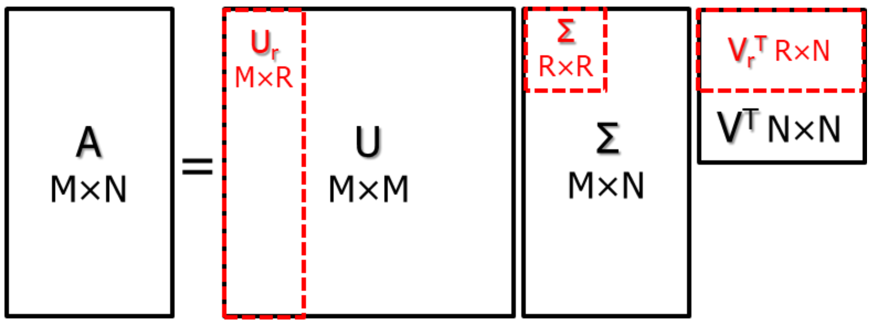
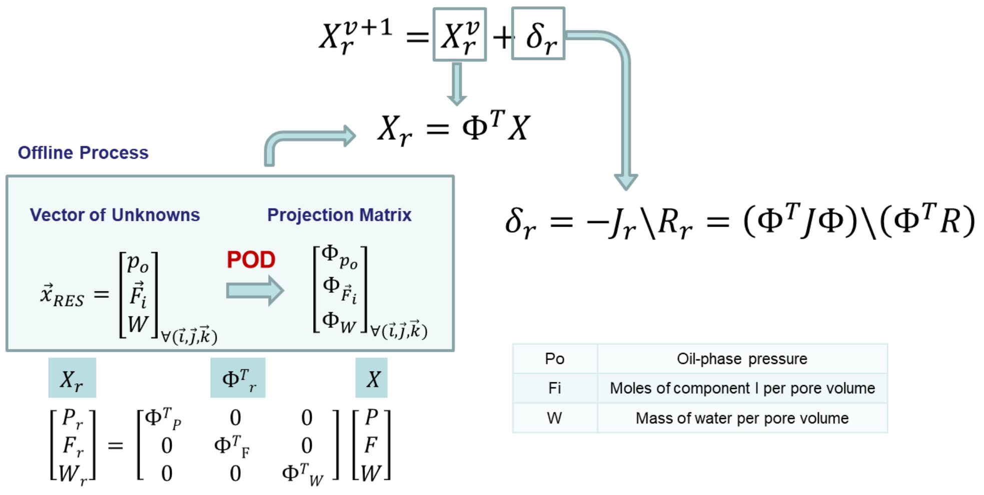


Figure 3.2 Illustration of POD basis. The reduced U matrix, U_r is the POD basis that is used to project the high-order system into a smaller subsystem. Reprinted with permission from SPE-198946-MS.

The POD basis is applied in the process of Newton Raphson iterations as depicted in Figure 3.3. In this case, the truncated vector of U, namely Φ is used to project the state, residual, and Jacobian into a smaller matrix.



$X_r^{v+1} = \boxed{X_r^v} + \boxed{\delta_r}$
$X_r = \Phi^T X$
Offline Process
Vector of Unknowns
Projection Matrix
POD
$\vec{x}_{RES} = \begin{bmatrix} p_o \\ \vec{F}_i \\ W \end{bmatrix}_{\forall(\vec{i},\vec{j},\vec{k})}$
$\begin{bmatrix} \Phi_{p_o} \\ \Phi_{\vec{F}_i} \\ \Phi_W \end{bmatrix}_{\forall(\vec{i},\vec{j},\vec{k})}$
$\delta_r = -J_r \backslash R_r = (\Phi^T J \Phi) \backslash (\Phi^T R)$
X_r
$\Phi^T{}_r$
X
$\begin{bmatrix} P_r \\ F_r \\ W_r \end{bmatrix} = \begin{bmatrix} \Phi^T{}_P & 0 & 0 \\ 0 & \Phi^T{}_F & 0 \\ 0 & 0 & \Phi^T{}_W \end{bmatrix} \begin{bmatrix} P \\ F \\ W \end{bmatrix}$

Po	Oil-phase pressure
Fi	Moles of component I per pore volume
W	Mass of water per pore volume

Figure 3.3 Application of POD in Newton Raphson iterations. State, residual, Jacobian are projected into the reduced subspace by the projection matrix Φ (POD basis U_r). Reprinted with permission from SPE-198946-MS.

The overall procedure of the POD algorithm is summarized in Table 3.1.

Table 3.1 Procedure of the POD algorithm.

```
Algorithm 1: POD Procedure

1:  while the error criteria are not met
2:  $R = R(x^{n+1}, x^n, u^{n+1}),\ J = J(x^{n+1}, x^n, u^{n+1})$
3:  $J_r = \psi^{\mathrm{T}} J \Phi,\ R_r = \psi^{\mathrm{T}} R$
4:  $\delta z = -J_r^{-1} R_r$
5:  $\delta \tilde{x} = \Phi \delta z$
6:  $x^{n+1} \leftarrow x^{n+1} + \delta \tilde{x}$
7:  End
```

3.2 Discrete Empirical Interpolation Method (DEIM)

Even though we reduce the system by projecting the states into a smaller subspace using POD, the performance of the POD cannot be maximized to completely reduce the dimension of the system because POD still requires the nonlinear-function evaluations in the computation of residuals and Jacobian matrices on all the gridblocks. In general, a large portion of the CPU time is spent on computing the nonlinear equations, so it is essential to find ways to reduce the computational time for the nonlinear terms. Therefore, we apply DEIM to approximate these nonlinear functions for further system reduction. DEIM is based on the greedy algorithm and is an enhancement of POD that simplifies the evaluation of the nonlinear term in the reduced model. DEIM approximates a linear or nonlinear function by means of an interpolatory projection of a few selected global snapshots of the function. The idea is to represent a function over the full domain while using only information in some locations. DEIM is briefly introduced in this section, and more details about DEIM can be found in Chaturantabut and Sorensen (2010).

Suppose $f(\tau) \in \mathbb{R}^n$ denotes a nonlinear function in a high-dimensional space. The function $f(\tau)$ can be linearly approximated by projecting it into a subspace using the basis U_f and low-dimensional coefficient $c(\tau)$ (Eq. 3.3).

$$f(\tau) \approx U_f c(\tau) \tag{3.3}$$

Similar to the POD basis, the basis matrix U_f is obtained by POD based on the snapshots of the nonlinear terms. To determine $c(\tau)$, a selection matrix P for the sampling points where the nonlinear terms are evaluated needs to be defined (Eq. 3.4).

$$P = \left(e_{\wp_1}, \cdots, e_{\wp_m}\right) \in \mathbb{R}^{n\times m} \tag{3.4}$$

Where, $e_{\wp_i} = [0, \cdots, 0, 1, 0, \cdots, 0]^T \in \mathbb{R}^n$ is a column of an identity matrix corresponding to the indices $\wp_i$. If we multiply P^T on both side of (3.3), we can rewrite $f(\tau)$ as follows (Eq. 3.5).

$$f(\tau) \approx U_f c(\tau) = U_f \left(P^T U_f\right)^{-1} P^T f(\tau) \tag{3.5}$$

Here we assume that $P^T U_f$ is nonsingular. $P^T f(\tau)$ can also be replaced by $f_{\wp_m}(\tau)$, therefore the final form of the expression is shown in Eq. 3.6.

$$f(\tau) \approx \tilde{f}(\tau) = U_f \left(P^T U_f\right)^{-1} P^T f(\tau) = U_f \left(P^T U_f\right)^{-1} \begin{pmatrix} f_{\wp_1}(\tau) \\ f_{\wp_2}(\tau) \\ \vdots \\ f_{\wp_m}(\tau) \end{pmatrix} \tag{3.6}$$

Figure 3.4 is an example of DEIM that represents the selected locations where the nonlinear functions are computed, and it is obviously noticeable that the computational cost can be significantly reduced when comparing the full-order model.

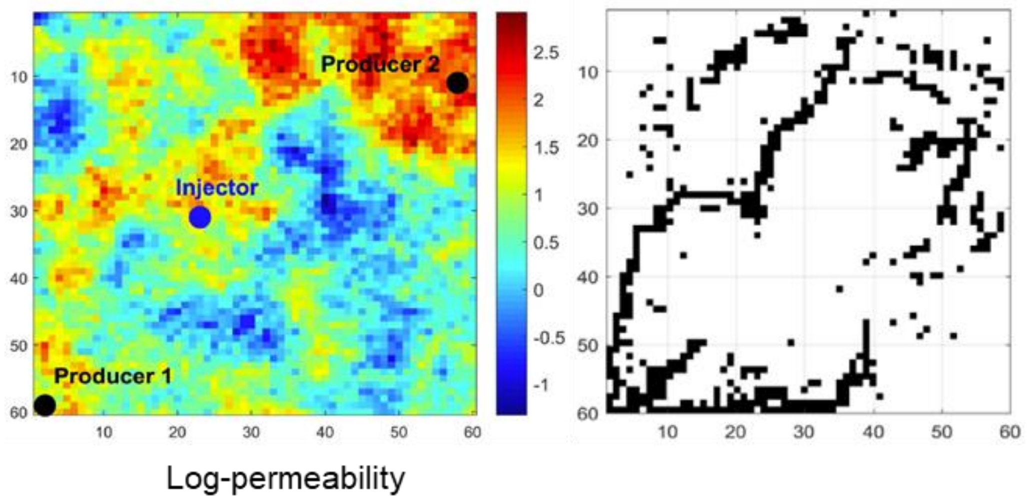


Figure 3.4 Example of DEIM locations (Tan et al. 2017). Left: map of log-permeability. Right: DEIM sampling locations. DEIM locations are selected by the greedy algorithm.

The overall DEIM procedure is summarized in Table 3.2. The interpolation indices $\wp_m$ is decided by the greedy algorithm as shown in Algorithm 2. The greedy algorithm is a type of heuristic problem-solving algorithms that seeks the global optimal solution by choosing the local optimal solution at each decision-making stage. In DEIM, the greedy algorithm sorts the column of the DEIM basis U_f in terms of the importance of the component. That is to say, the most important column is located at the first column and the least one is at the last column. The first sampling point from the first column of the basis matrix is simply the point with the largest absolute value. And then at the next column, the point with the largest absolute error between the actual function value and the estimated value, $max\{|u_{f_\ell} - U_f c|\}$, is chosen for the next location in order to improve the performance of the interpolation. This process is repeated until it reaches the last column of the basis matrix.

This greedy algorithm is commonly used when determining the selection points not only for the DEIM but also for the Gauss-Newton approximate tensor (GNAT) which is a modification of Newton's method to approximate nonlinear residual and left subspace basis by computing only a small subset of rows (Chaturantabut and Sorensen 2010; Carlberg et al. 2011, 2013).

Table 3.2 Procedure of the DEIM algorithm.

Algorithm 2: DEIM Procedure (Chaturantabut and Sorensen 2010)

```
1:  DEIM basis $U_f = (u_{f_1}, \cdots, u_{f_m}) \in \mathbb{R}^{n\times m}$
2:    $(|\rho|, \wp_1) = max\{|u_{f_1}|\}$
3:    $U_f = (u_{f_1}), P = (e_{\rho_1})$
4:  **for** $\ell = 2, \cdots, m$
5:    $(P^T U_f)c = P^T u_{f_\ell}$
6:    $r = u_{f_\ell} - U_f c$
7:    $(|\rho|, \wp_1) = max\{|r|\}$
8:    $U_f \leftarrow (U_f u_{f_\ell}), P \leftarrow (P e_{\rho_\ell})$
9:  **end**
10:   $\vec{\wp} = (\wp_1, \cdots, \wp_m)^T$
```

3.3. POD-DEIM for Compositional Simulation

The concept of POD is very straightforward, but the implementation to highly nonlinear systems require additional manipulations and tuning. In particular, a modified version of the POD is necessary to handle efficiently compositional simulation.

As seen in the flow chart of Newton Raphson iteration (Figure 3.5), we may collect and save snapshots from the fine scale training run. However, since we update the solutions of two independent variables (ln K_i and f_v) using flash calculation after solving the system of equations, they are not the direct solutions from the system of equations. More specifically, if the regular variables are used in the compositional simulator the variables derived from flash calculation are calibrated in the Newton Raphson iteration loop. Then, the solutions of these flash calculation variables are updated based on the solutions of other variables such as oil phase pressure once they are obtained from the Newton Raphson iteration. Therefore, one cannot use the POD at this point since the snapshots for POD basis should be derived from the solutions of the system of equations. In other words, POD basis of flash calculation variables should not be updated after we obtain the solutions.

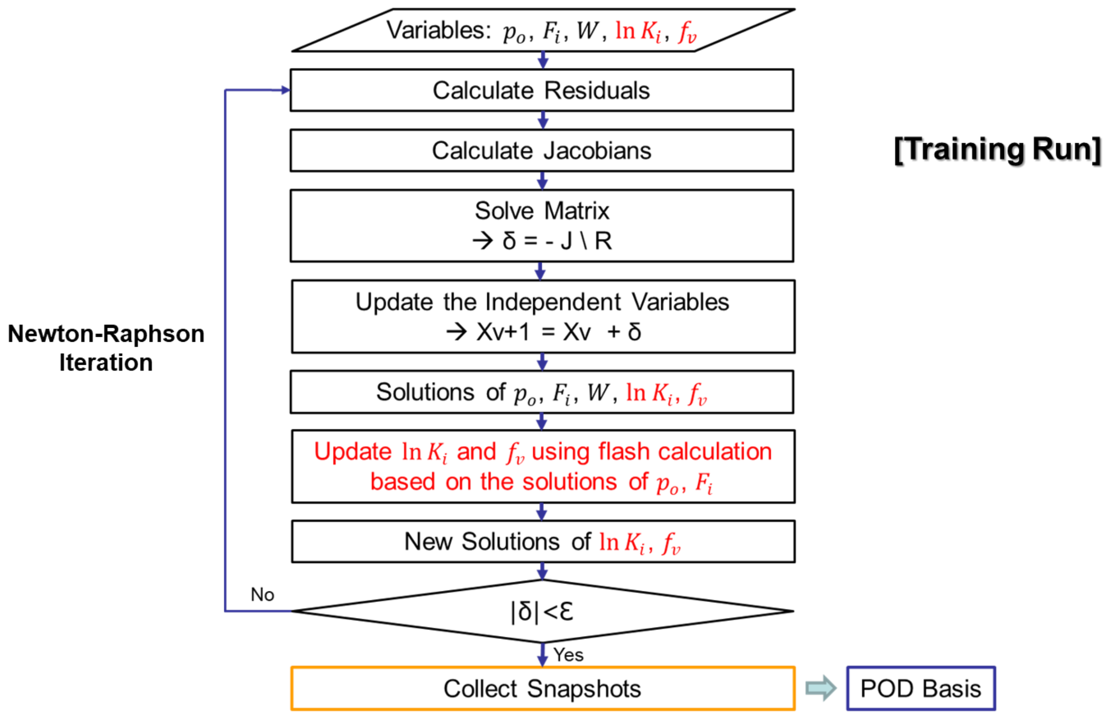

Variables: p_o, F_i, W, $\ln K_i$, f_v
Calculate Residuals
Calculate Jacobians
[Training Run]
Solve Matrix → δ = - J \ R
Newton-Raphson Iteration
Update the Independent Variables → Xv+1 = Xv + δ
Solutions of p_o, F_i, W, $\ln K_i$, f_v
Update $\ln K_i$ and f_v using flash calculation based on the solutions of p_o, F_i
New Solutions of $\ln K_i$, f_v
No
|δ|<Ɛ
Yes
Collect Snapshots
POD Basis

Figure 3.5 Flowchart for collecting snapshot in training schedules in a general compositional simulator based on a modified molar formulation. Two independent variables $\ln K_i$ and f_v are updated after convergence in Newton-Raphson method, which violates the concept of snapshot that must be the solution of the system.

Therefore, if we run a test case using the POD basis obtained from a training case, obviously we get wrong solutions for K value and molar vapor fraction, which are the independent variables related to flash calculation, because their POD bases are not based on the solutions of the system of equations (Figure 3.6). This occurs even when we run the same schedules of training case for the test case. The POD basis must be based on the solutions of the system of equations.

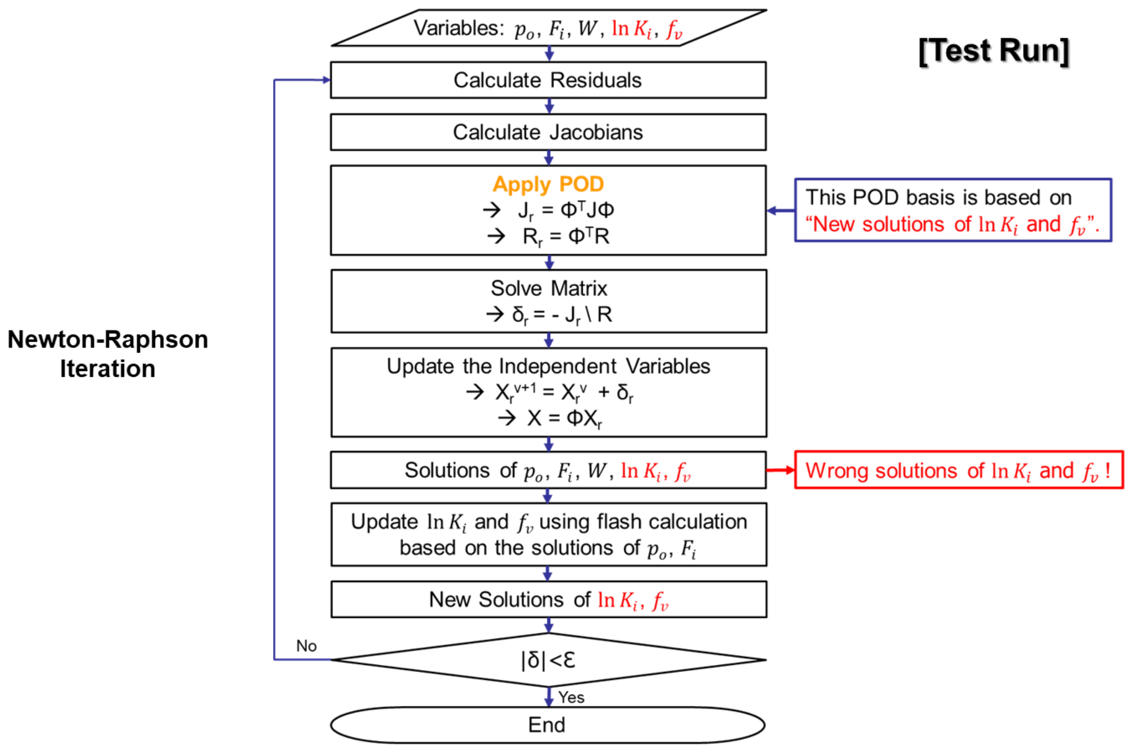
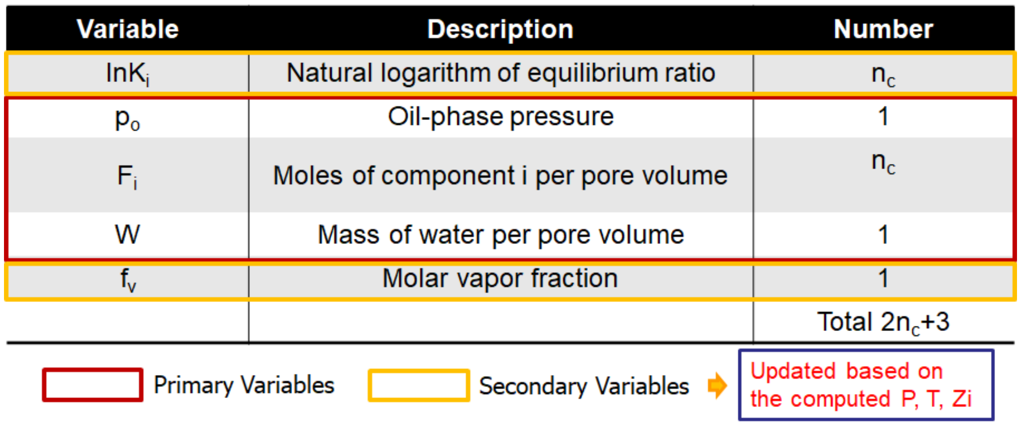

[Test Run]
Variables: p_o, F_i, W, $\ln K_i$, f_v
Calculate Residuals
Calculate Jacobians
Apply POD
→ $J_r = \Phi^T J \Phi$
→ $R_r = \Phi^T R$
This POD basis is based on "New solutions of $\ln K_i$ and f_v".
Solve Matrix
→ $\delta_r = - J_r \backslash R$
Newton-Raphson Iteration
Update the Independent Variables
→ $X_r^{v+1} = X_r^v + \delta_r$
→ $X = \Phi X_r$
Solutions of p_o, F_i, W, $\ln K_i$, f_v
Wrong solutions of $\ln K_i$ and f_v !
Update $\ln K_i$ and f_v using flash calculation based on the solutions of p_o, F_i
New Solutions of $\ln K_i$, f_v
No
$|\delta|<\mathcal{E}$
Yes
End

Figure 3.6 Flowchart of the POD application using snapshot from training schedules for test run in a general compositional simulator based on a modified molar formulation. Wrong solutions are obtained since the POD basis is not based on the system of equations.

To this end, we modify the general molar variables formulation by dividing the independent variables into primary and secondary variables, so we can apply the POD-DEIM method using the physically correct variable. Basically, primary variables are defined for the mass balance calculation whereas the secondary variables are related to the flash calculation as shown in Figure 3.7.



Variable	Description	Number
$\ln K_i$	Natural logarithm of equilibrium ratio	n_c
p_o	Oil-phase pressure	1
F_i	Moles of component i per pore volume	n_c
W	Mass of water per pore volume	1
f_v	Molar vapor fraction	1
		Total $2n_c+3$

🟥 Primary Variables 🟨 Secondary Variables ➔ Updated based on the computed P, T, Zi

Figure 3.7 Primary and secondary variables. Secondary variables are the independent variables for the flash calculation. The POD is applicable with respect to primary variables. Reprinted with permission from SPE-198946-MS.

Using this new formulation, the state variables along with the residual and Jacobian matrices of primary and secondary terms are shown in Eq. 3.7 through Eq. 3.12. Even though we separate primary and secondary variables and equations, residuals of the primary equations are still the functions of both primary and secondary variables. For example, the mass balance equation for hydrocarbon component is dependent on phase compositions. Therefore, we need to recast the original matrices to handle correctly the dependencies for the secondary constraints.

$$\vec{x}_{RES\_primary} = \begin{bmatrix} p_o \\ \vec{F}_i \\ W \end{bmatrix}_{\forall(\vec{\imath},\vec{\jmath},\vec{k})} \tag{3.7}$$

$$\vec{R}_{RES\_primary} = \begin{bmatrix} R_{sat} \\ \vec{R}_i \\ R_w \end{bmatrix}_{\forall(\vec{\imath},\vec{\jmath},\vec{k})} \tag{3.8}$$

$$\vec{\vec{J}} = \left[\frac{\partial \vec{R}_{RES}}{\partial \vec{x}_{RES}}\right]_{(\vec{\imath},\vec{\jmath},\vec{k})} = \begin{bmatrix} \frac{\partial R_{sat}}{\partial x_{p_o}} & \frac{\partial R_{sat}}{\partial x_{F_1}} & \cdots & \frac{\partial R_{sat}}{\partial x_{F_{n_c}}} & \frac{\partial R_{sat}}{\partial W} \\ \frac{\partial R_1}{\partial x_{p_o}} & \frac{\partial R_1}{\partial x_{F_1}} & \cdots & \frac{\partial R_1}{\partial x_{F_{n_c}}} & \frac{\partial R_1}{\partial W} \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ \frac{\partial R_{n_c}}{\partial x_{p_o}} & \frac{\partial R_{n_c}}{\partial x_{F_1}} & \cdots & \frac{\partial R_{n_c}}{\partial x_{F_{n_c}}} & \frac{\partial R_{n_c}}{\partial W} \\ \frac{\partial R_w}{\partial x_{p_o}} & \frac{\partial R_w}{\partial x_{F_1}} & \cdots & \frac{\partial R_w}{\partial x_{F_{n_c}}} & \frac{\partial R_w}{\partial W} \end{bmatrix}_{(\vec{\imath},\vec{\jmath},\vec{k})} \tag{3.9}$$

$$\vec{x}_{RES\_secondary} = \begin{bmatrix} \ln \vec{K_\imath} \\ f_v \end{bmatrix}_{\forall(\vec{\imath},\vec{\jmath},\vec{k})} \tag{3.10}$$

$$\vec{R}_{RES\_secondary} = \begin{bmatrix} \vec{R}_{fug_i} \\ R_{RR} \end{bmatrix}_{\forall(\vec{\imath},\vec{\jmath},\vec{k})} \tag{3.11}$$

$$\vec{\vec{J}}_{Secondary} = \left[\frac{\partial \vec{R}_{RES}}{\partial \vec{x}_{RES}}\right]_{(\vec{\imath},\vec{\jmath},\vec{k})} = \begin{bmatrix} \frac{\partial R_{fug_1}}{\partial \ln K_1} & \cdots & \frac{\partial R_{fug_1}}{\partial \ln K_{n_c}} & \frac{\partial R_{fug_1}}{\partial f_v} \\ \vdots & \cdots & \vdots & \vdots \\ \frac{\partial R_{fug_{n_c}}}{\partial \ln K_1} & \cdots & \frac{\partial R_{fug_{n_c}}}{\partial \ln K_{n_c}} & \frac{\partial R_{fug_{n_c}}}{\partial f_v} \\ \frac{\partial R_{RR}}{\partial \ln K_1} & \cdots & \frac{\partial R_{RR}}{\partial \ln K_{n_c}} & \frac{\partial R_{RR}}{\partial f_v} \end{bmatrix}_{(\vec{\imath},\vec{\jmath},\vec{k})} \tag{3.12}$$

Thus, the chain rule is used to reconstruct the revised Jacobian matrix. In a fully implicit framework, the global system of primary equations is solved, and then the local system of secondary equations is solved on a gridblock basis for secondary variables, while keeping the primary variables fixed. Therefore, once the secondary equations are solved, all fluid and solid properties are updated accordingly. The secondary equations are assumed to be fully converged for all Newton iterations. This is achieved by subjecting the solution of the primary equations to the constraints based on the secondary equations. Hence, the chain rule can be used to reconstruct the Jacobian matrix as shown in Eq. 3.13 through 3.17.

$$J = \frac{dR_p\left(X_p, X_s\right)}{dX_p} = \frac{\partial R_p}{\partial X_s}\frac{dX_s}{dX_p} + \frac{\partial R_p}{\partial X_p} \tag{3.13}$$

$$\frac{dR_s\left(X_p, X_s\right)}{dX_p} = \frac{\partial R_s}{\partial X_s}\frac{dX_s}{dX_p} + \frac{\partial R_s}{\partial X_p} \tag{3.14}$$

$$\frac{dX_s}{dX_p} = -\frac{\partial R_s}{\partial X_s}\backslash\frac{\partial R_s}{\partial X_p} \tag{3.15}$$

$$\frac{dR_p\left(X_p, X_s\right)}{dX_p} = -\frac{\partial R_p}{\partial X_s}\left(\frac{\partial R_s}{\partial X_s}\backslash\frac{\partial R_s}{\partial X_p}\right) + \frac{\partial R_p}{\partial X_p} \tag{3.16}$$

$$J = -Jps(Jss\backslash Jsp) + Jpp \tag{3.17}$$

Here, R_p and R_s represent residuals corresponding to the primary and secondary equations, and X_p and X_s denote the sets of primary and secondary variables. The final form of Jacobian J in Eq. 3.17 can be expressed using Jpp, Jps, Jsp, Jss that represent $\frac{\partial R_p}{\partial X_p}$, $\frac{\partial R_p}{\partial X_s}$, $\frac{\partial R_s}{\partial X_p}$, $\frac{\partial R_s}{\partial X_s}$, respectively.

Saturation constraint residual, hydrocarbon component residual, and water residual are the residuals corresponding to the primary equations for the POD-DEIM application. These residual equations are composed of three different terms, which are accumulation term, convective flow term, source/sink term. Since residuals contain many nonlinear terms, our approach for collecting the snapshot of nonlinear terms at every converged time step involves separating snapshot into an accumulation term, a convective flow term, and a source/sink term. These three terms have a quite different physical meanings and are non-zeros at convergence. Therefore, they are suitable for obtaining the DEIM basis (Figure 3.8, Figure 3.9).

$$
\begin{aligned}
R_i = {} & \boxed{\frac{V_b}{\Delta t}[F_i^n \phi'(p_{oc}^{n+1} - p_{oc}^n) + \phi^{n+1}(F_i^{n+1} - F_i^n)]} \longrightarrow \textit{Accumulation} \\
& \boxed{\begin{aligned} & - \sum_{\eta=1}^{6} [a_{o\eta}^{n+1} x_i^{n+1} (\Delta p_{o\eta} - \gamma_{o\eta}^{n+1} \Delta z_\eta)] \\ & - \sum_{\eta=1}^{6} [a_{g\eta}^{n+1} y_i^{n+1} (\Delta p_{o\eta} - \gamma_{g\eta}^{n+1} \Delta z_\eta)] \\ & - \sum_{\eta=1}^{6} \left[\left(a_{g\eta} y_i p'_{cgo_\eta} S'_{g_\eta} \right)^{n+1} \Delta F_i \right] \end{aligned}} \longrightarrow \textit{Convective flow} \\
& \boxed{\begin{aligned} & + WI_o^{n+1} x_i^{n+1} (p_{oc}^{n+1} - p_{wf}^{n+1}) \\ & + WI_g^{n+1} y_i^{n+1} (p_{oc}^{n+1} + p_{cgo}^{n+1} - p_{wf}^{n+1}) \end{aligned}} \longrightarrow \textit{Well Source/Sink}
\end{aligned}
$$

Figure 3.8 Division of hydrocarbon component residual to collect snapshot for DEIM. Three different terms have different physical meanings and are non-zeros at convergence.

$$
\begin{aligned}
R_w = {} & \boxed{\frac{V_b}{\Delta t}[W^n \phi'(p_{oc}^{n+1} - p_{oc}^n) + \phi^{n+1}(W^{n+1} - W^n)]} \longrightarrow \textit{Accumulation} \\
& \boxed{- \sum_{\eta=1}^{6} \left[a_{w\eta}^{n+1} (\Delta p_{o\eta}^{n+1} - \gamma_{w\eta}^{n+1} \Delta z_\eta) - \frac{a_{w\eta}^{n+1} p'^{(n+1)}_{cow_\eta}}{\rho_{w\eta}^{n+1}} \Delta W_\eta^{n+1} \right]} \longrightarrow \textit{Convective flow} \\
& \boxed{+ WI_w^{n+1} (p_{oc}^{n+1} - p_{cow}^{n+1} - p_{wf}^{n+1})} \longrightarrow \textit{Well Source/Sink}
\end{aligned}
$$

Figure 3.9 Division of water residual to collect snapshot for DEIM. Three different terms have different physical meanings and are non-zeros at convergence.

To make the snapshot matrix of hydrocarbon component residual and water residual simple, we concatenate snapshots of some nonlinear terms as shown in Eq. 3.18.

$$X_g = \left[g_{accum}^1, g_{accum}^2, \dots, g_{accum}^{n_s}, g_{conv}^1, g_{conv}^2, \dots, g_{conv}^{n_s}\right] \tag{3.18}$$

Here, we concatenate only accumulation terms and convective flow terms since source term is the negative sum of the accumulation and convective flow term. This formulation was introduced by Tan et al. (2017) and subsequently expanded in the work by Jiang and Durlofsky (2018). Recalling the DEIM algorithm, we apply DEIM method using projection basis and indices from this concatenated snapshot (Eq. 3.19 through Eq. 3.21).

$$X_g = U\Sigma V^T \tag{3.19}$$

$$g(x) \approx U(P^T U)^{-1} P^T g(x) \tag{3.20}$$

$$\begin{gathered} g_p(x) \approx U_p\left(P_p^T U_p\right)^{-1} P_p^T g_p(x) \\ where, p = Hydrocarbon\ Component,\ Water \end{gathered} \tag{3.21}$$

The flow chart in Figure 3.10 summarizes the overall offline and online process of the POD-DEIM algorithm. In offline processing, snapshots of primary variables are collected for the POD basis and snapshots of primary residual equations are obtained for the DEIM basis from a fine scale model with training schedules. And then in online processing, mass balance calculation is performed in a reduced-order system by the POD-

DEIM method. Note that the secondary variables and equations for flash calculation remain in a full-order system.

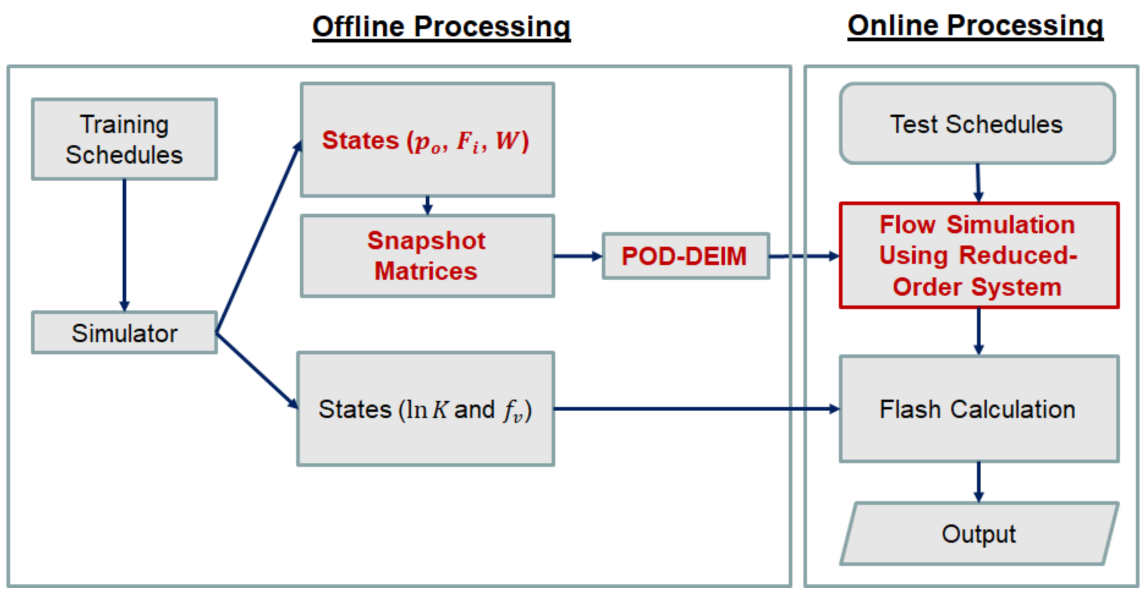


Figure 3.10 Flow chart of the coupled POD and DEIM approaches. Snapshots for the POD basis and the DEIM basis are acquired in offline processing. In online processing, mass balance calculation is performed in a reduced-order system by the POD-DEIM method.

The theoretical computational complexity of the full-order model and reduced model based on POD and POD-DEIM are compared in Table 3.3. (Chaturantabut 2012). Note that $\alpha(n)$ denotes the floating-point operation per second (Flops) for evaluating the nonlinear function F at n components, and $\alpha^d(n)$ is the Flops for evaluating derivative of the nonlinear function F at n components. Practically, the CPU time may not be always proportional to the predicted Flops, since there are many other factors affecting the actual CPU time (Gilbert 1992). Nevertheless, this analysis is meaningful to investigate the relative computational times and the performance of the POD-DEIM.

Table 3.3 Comparison of the computational complexity for each Newton iteration in different systems. n: number of gridblocks, k: number of POD basis, and m: number of DEIM basis.

System	**Computation in Newton iteration**	**Complexity (1 iteration)**	**Total complexity**
Full	$G(y) = Ay + F(y)$ $J(y) = A + diag\{F'(y)\}$ $y \leftarrow y - J(y)^{-1}G(y)$	$2n^2 + \alpha(n) + n\ or\ cn + \alpha(n)$ $n^2 + \alpha^d(n)\ or\ n + \alpha^d(n)$ $O(n^3)\ or\ O(n^2)$ c ~ nonzero per row of A	$O(n^3)$ Sparse:$O(n^2)$

Table 3.3 Continued

System	Computation in Newton iteration	Complexity (1 iteration)	Total complexity
POD	$\hat{G}(y) = \hat{A}\hat{y} + V^T F(V\hat{y})$ $\hat{J}(y) = \hat{A} + V^T diag\{F'(V\hat{y})\}V$ $\hat{y} \leftarrow \hat{y} - \hat{J}(y)^{-1}\hat{G}(y)$	$2k^2 + \alpha(n) + k + 4nk$ $k^2 + \alpha^d(n) + 4nk + 2nk^2$ $O(k^3)$	$O(k^3 + nk^2)$
POD-DEIM	$\hat{G}(y) = \hat{A}\hat{y} + BF(V_{\vec{\wp}}\hat{y})$ $\hat{J}(y) = \hat{A} + Bdiag\{F'(V_{\vec{\wp}}\hat{y})\}V_{\vec{\wp}}$ $\hat{y} \leftarrow \hat{y} - \hat{J}(y)^{-1}\hat{G}(y)$ Where $B = V^T U U_{\vec{\wp}}^{-1}$, $U_{\vec{\wp}} = P^T U, V_{\vec{\wp}} = P^T V$	$2k^2 + \alpha(m) + k + 4mk$ $k^2 + \alpha^d(m) + 4mk + 2mk^2$ $O(k^3)$	$O(k^3 + mk^2)$

3.4. Applications and Results

The application of the proposed reduced order modeling using POD-DEIM in the developed compositional simulator is discussed in this chapter. Here we investigate the ability of our novel MOR formulation to speed up the computation and efficiency along with its accuracy to handle multiple components as compare to the conventional full-order model. Hence, we develop the fully implicit compositional reservoir simulator that is coded in MATLAB®, and then, POD-DEIM is applied to the developed simulator.

When it comes to the POD-DEIM, the snapshot-based method works in a two-step process: training case and test case. Essentially, one can retrieve the necessary information – snapshots which are the solutions of independent variables at each time step - from the full-order training case to derive the POD basis. And the DEIM basis is also collected from the snapshots of the solutions of the nonlinear functions in this step. Then, the POD-DEIM is applied for the test case which has a different well control schedules from the training schedules. The results of the test case using POD-DEIM are compared to those of full-order test case in test schedules to validate the applicability of the POD-DEIM method.

3.4.1. Model Description

A synthetic reservoir model with heterogeneous and anisotropic permeability distribution is created to test the proposed MOR methods. The reservoir model has $15 \times 15 \times 3$ gridblocks of equal dimensions with length and width of 88 ft, thickness of 30 ft.

In order to investigate the variation of each component as time changes, it is necessary to run the gas injection case such as nitrogen injection or the enhanced oil recovery using CO_2 injection. However, as an initial step to validate the proposed MOR method in the compositional model, we assume the waterflooding scenario in this study to make the cases as simple as possible. Thus, we have one injector and one producer as illustrated in Figure 3.11, and properties of the model are shown in Table 3.4.

There exist 3 phases - oil, water, gas, and eight components are assumed in the model. They are CO_2, $C_1 - N_2$, $C_2 - C_3$, C_4, C_5, C_6, C_{7+}, $Asphaltene$. The detailed fluid EOS Parameters for flash calculation is displayed in Appendix C. Sum of the concentration of each component should be equal to 1.

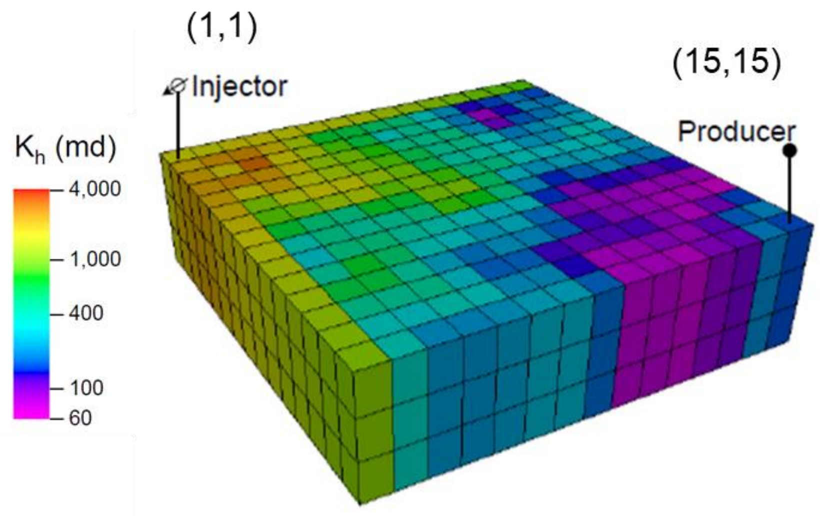


Figure 3.11 Reservoir model permeability map (Valbuena Olivares 2015). Injection well and production well are located at grid block (1, 1) and (15, 15) respectively.

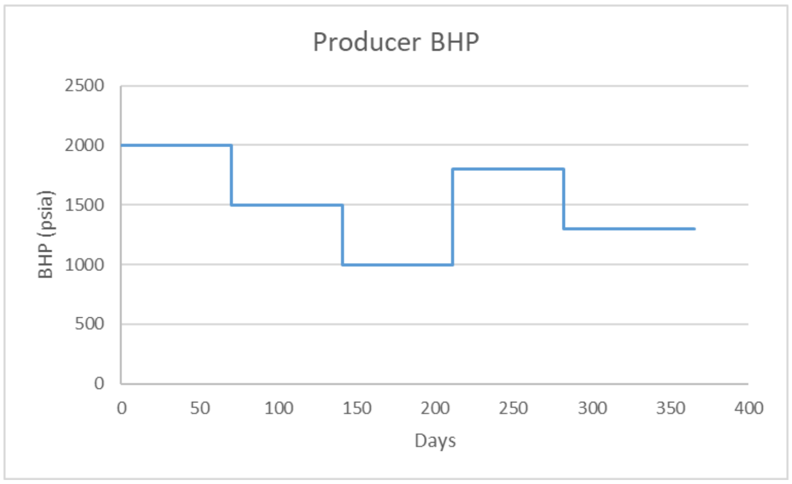
Table 3.4 Reservoir model properties (Valbuena Olivares 2015).

Property	Value
Reservoir top depth, ft	2,665
Porosity, fraction	0.2
k_h Geometric mean, mD	400.2
k_v/k_h, fraction	0.1
Initial water saturation, fraction	0.35
Initial gas saturation, fraction	0
Initial pressure, psia	8,868
Temperature, °F	200
Water density at SC, lb/ft^3	63.0
Rock compressibility, $psia^{-1}$	4x10-6
Reference pressure, psia	5,868
Oil in place, MMSTB	3.30
Gas in place, MMSCF	929.48
Water in place, MMSTB	1.94
Simulated time, days	365

3.4.2. Validation of the Developed Simulator

In this study, the specifications of the computer are Intel(R) Xeon(R) CPU E5-1660 v3 with 3.00 GHz for the processor, and 64.0 GB RAM. For the parallel processing, 8 cores in a single processor are used for computing.

For the first step, we compare our developed simulator with the commercial software Schlumberger Eclipse (Schlumberger 2014) to verify its performance. The producer well is controlled by bottom hole pressure, and the injector well is controlled by injection rate, as shown in Figure 3.12. We simulate the reservoir for 365 days' time span, and Figure 3.13 represents the comparisons for reservoir pressure, oil production rate, water production rate, and gas production rate, for both the in-house simulator and Eclipse. As can be seen in this picture, there are excellent agreements to the results of the commercial software. Since even the peaks caused by the abrupt injection in a very short amount of time are accurately realized in the developed simulator, our in-house simulator is then deemed accurate for further studies using reduced-order models.



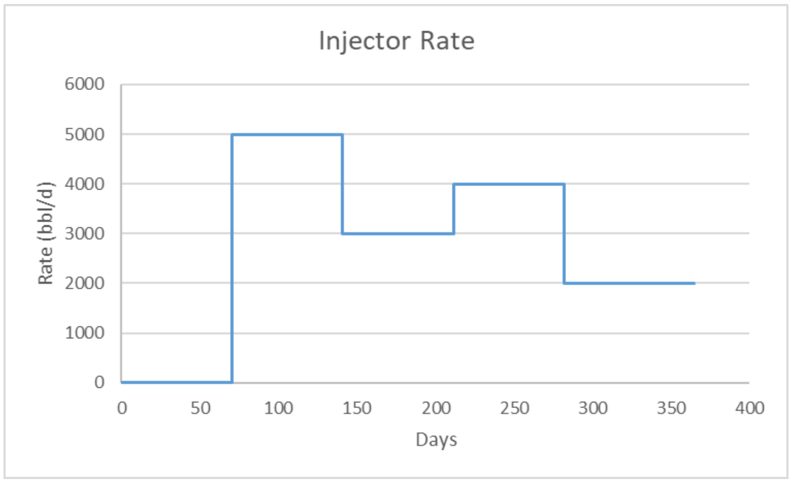


Figure 3.12 Well schedules used for validating the developed model. Time span of the simulation is 365 days. The producer well is controlled by bottom hole pressure, and the injector well is controlled by injection rate. Reprinted with permission from SPE-198946-MS.

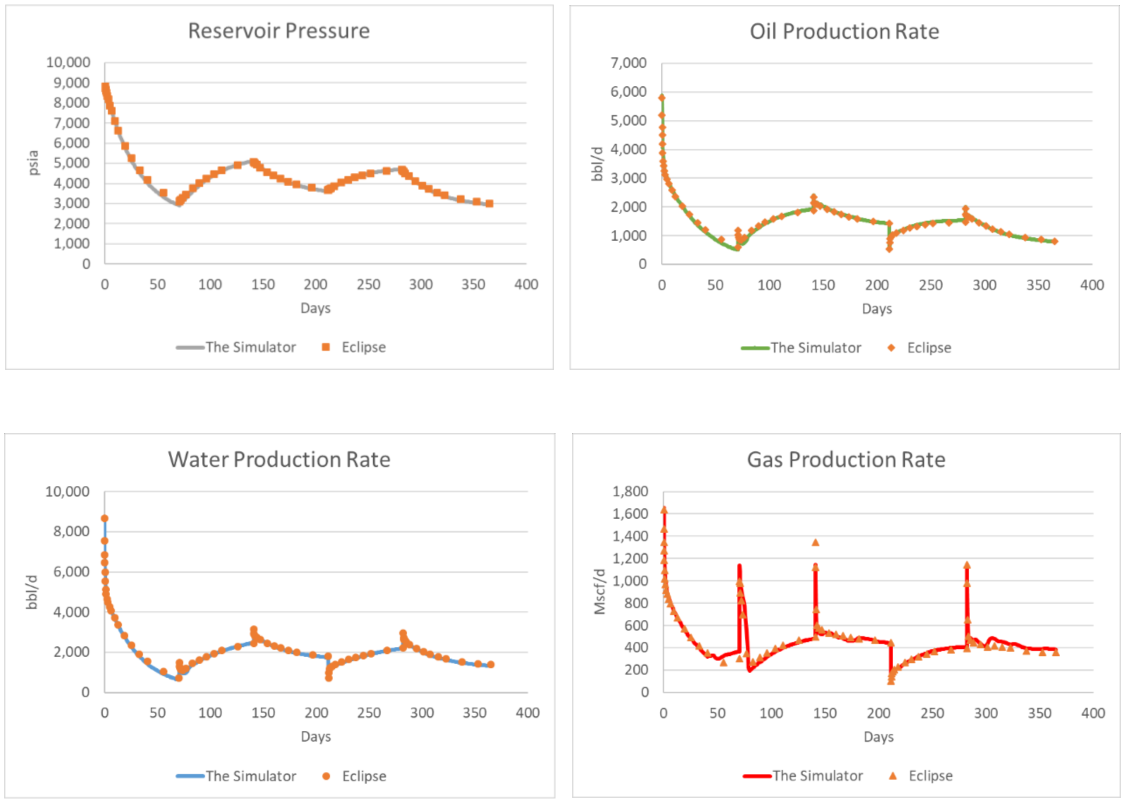


Figure 3.13 Comparison of the performance between the Eclipse and the developed simulator. The developed simulator shows excellent match to the result of the commercial software. Reprinted with permission from SPE-198946-MS.

3.4.3. Application of POD-DEIM

As mentioned earlier, we use a two-step process to obtain the reduced-order model. The training case is utilized for capturing snapshots, whereas test case is used for testing the proposed MOR method under different well schedules.

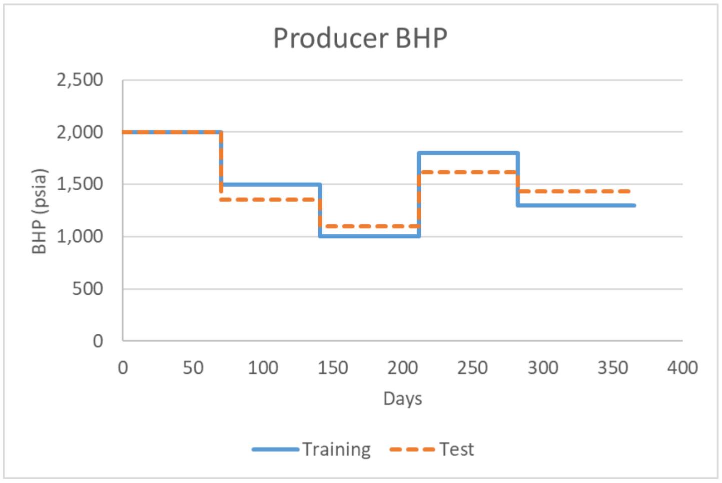
In our examples, both training and test schedules are run for 365 days, and their initial time step (dt) is set to 0.1 day. We use a variable time step, therefore, if convergence is achieved within a specified maximum number of iterations, time step size is doubled up to a user-defined size. If convergence is not accomplished within a maximum number of iterations, time step size is reduced by half, and initial guess and iteration process is reset.

In order to have enough number of snapshots with respect to the POD-DEIM basis, maximum time step for the training case is set to 1 day in training case, whereas that of test case is set to 15 days. Consequently, the training case has 378 time steps, and test case has 58 time steps (Table 3.5).

Table 3.5 Comparison of training case and test case. The POD-DEIM is validated using the test cases that have different well schedules to the training case. The training case has 378 time steps to obtain the enough number of snapshots.

	Training Case	**Test Case**
Purpose	Capture and save snapshots (POD-DEIM Basis)	Validate POD-DEIM using different well schedules
Duration of Simulation	365 days	365 days
Initial dt	0.1 day	0.1 day
Maximum dt	1 day	15 days
Number of Time Step	378	58

The well schedules are perturbed by 10% for the test case to verify the performance of the POD-DEIM model as shown in Figure 3.14.



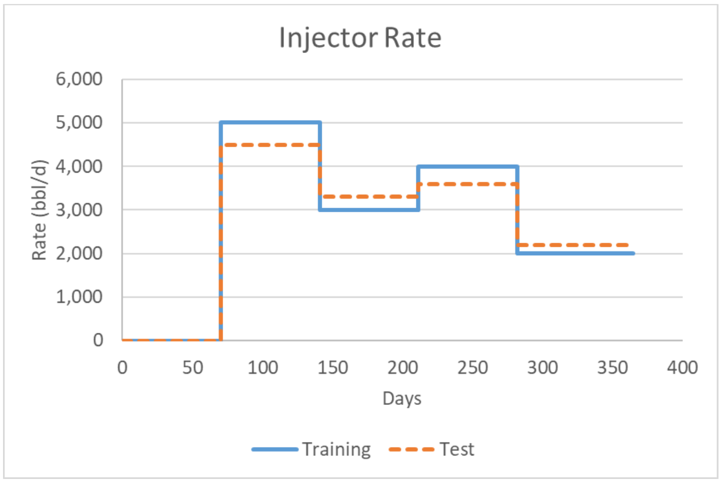


Figure 3.14 Original and perturbed well schedules used for training and test case. The well schedules of the training case are perturbed by 10% for the test case. Reprinted with permission from SPE-198946-MS.

We validate the POD-DEIM by applying it to three different test cases (Table 3.6). All three cases share the same POD basis having dimension of 445. There are 10 independent variables in 675 gridblocks (15×15×3), therefore the dimension of the full-order system is 6,750. The POD basis, namely projection matrix Φ is selected by the fraction of total energy to be captured, which is 99.99% in this study, and then the dimension of the POD basis becomes 6,750 x 445. If we multiply this POD basis by the full-order system, the dimension of the reduced system becomes 445 as well. We change the residuals selected for DEIM and the number of DEIM location for each case in order to verify the performance and sensitivity of the POD-DEIM.

Table 3.6 Three cases for test run. The DEIM is applied to the different residuals with different number of sampling location for each test case.

	Case 1	**Case 2**	**Case 3**
Dimension of POD Basis	6,750 × 445	6,750 × 445	6,750 × 445
Residual Selected for DEIM	R_{sat}	$R_1, \cdots, R_8$	R_{sat} $R_1, \cdots, R_8$ R_w
Number of DEIM Location	100	150	300

Because we do not have any prior knowledge about the appropriate number of sampling locations for the DEIM, we implement sensitivity analysis using three different cases by changing the number of sampling locations as well as the type of residual to apply the DEIM.

In case 1, POD is applied to all primary variables and DEIM is applied only to the saturation constraint residual with 100 sampling points out of 675 total gridblocks. The selected DEIM locations are presented in Figure 3.15. As the critical dynamic features arise around the wells, most of the grid cells are selected near the wells.

POD reduces the dimension of the system from 6,750 to 445, and DEIM enables us to compute 6,175 nonlinear functions out of 6,750. To sum up, about 8.5% of nonlinear function computation is reduced.

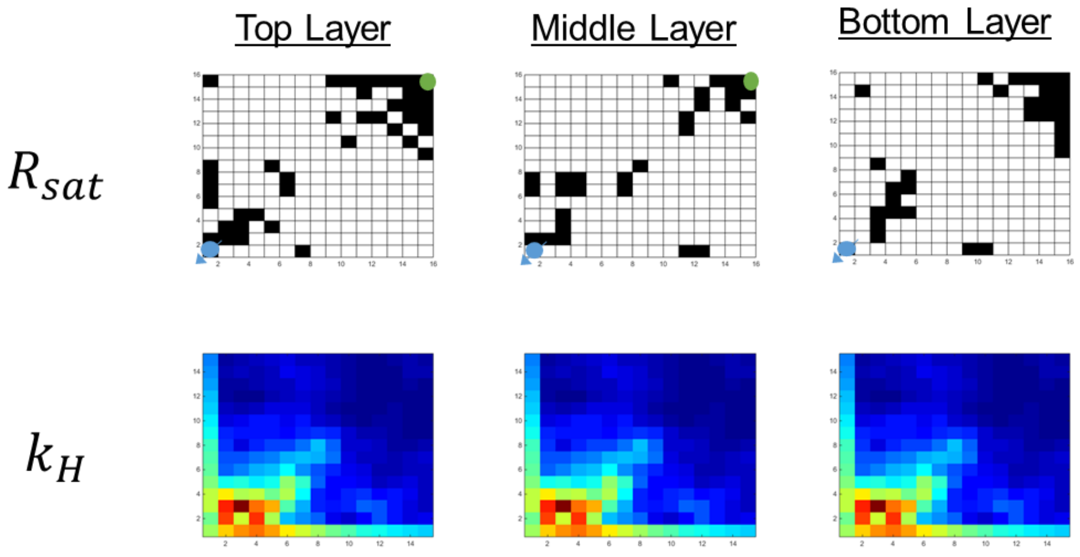


Figure 3.15 Sampling location for the DEIM in case 1. The selected 100 locations of the DEIM for R_{sat} are shown.

Figure 3.16 shows the performance of the original simulator and the simulator with POD-DEIM. We compare reservoir pressure, oil, water, gas production rate. The orange dashed lines are the simulation result from the original simulator using training schedules, and black solid line is the result from the original simulator using test schedules. Dots are the results of POD-DEIM application in test schedules. The well schedules are perturbed by 10% for the test schedules, therefore we can see the variation between the training and test schedules. All results show very good estimation within less than 0.1% error.

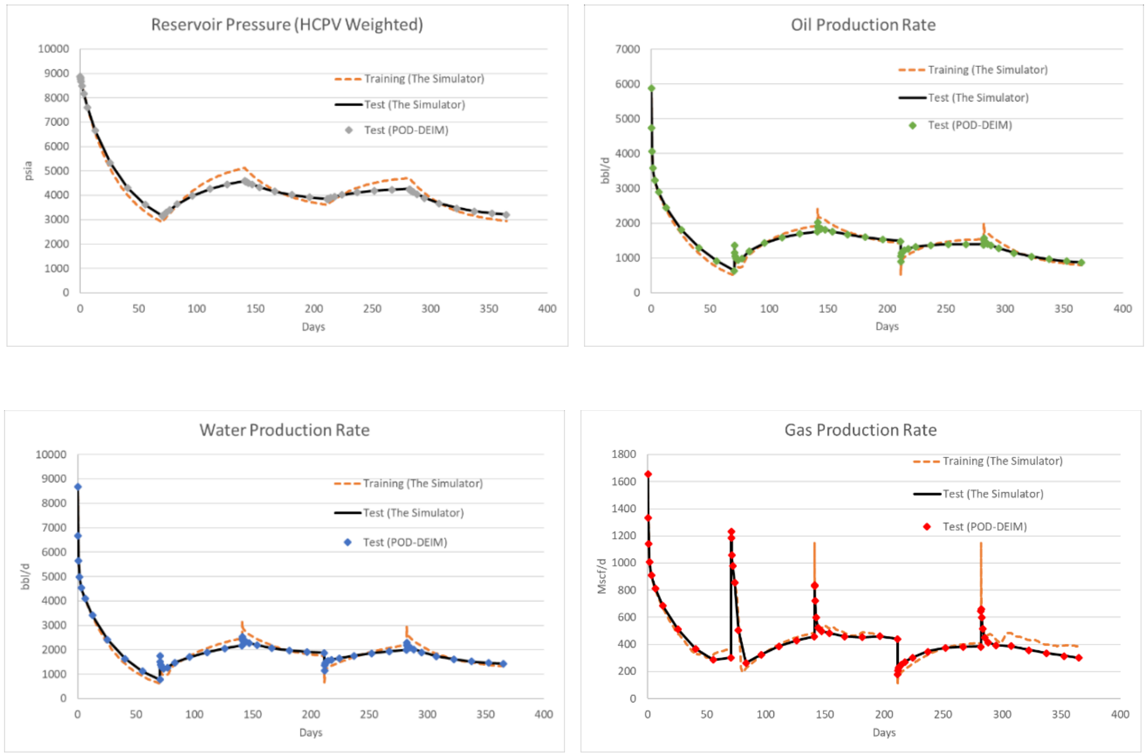


Figure 3.16 Comparison of the performance of training and case 1. Results of the POD-DEIM application show very good estimation within less than 0.1% error.

Table 3.7 describes the reduction of simulation run time by the POD-DEIM in a single time step. For case 1, when we apply POD-DEIM, simulation run time of a single time step is reduced around 0.7% (Figure 3.17). Note that about 98% of the total CPU time is spent in Jacobian setup. Even though the dimension of the system is reduced by POD, the number of locations for nonlinear function evaluation is reduced a little by DEIM, thus we can only see a small amount of time reduction in this case. Time reduction in Jacobian setup and residual calculation arises from DEIM, and reduction in primary matrix solver mostly results from POD.

Table 3.7 Simulation run time of a single time step for the full-order model and the reduced order model in case 1. The POD-DEIM reduces the CPU time for Jacobian setup, matrix solver (primary), and residual calculation.

	Full-Order (sec)	POD-DEIM (sec)
Jacobian Setup	335.0	332.9
Matrix Solver (Primary)	2.4	2.0
Matrix Solver (Secondary)	1.2	1.1
Linear/Nonlinear Eq.	3.0	2.1
Other	1.1	2.0

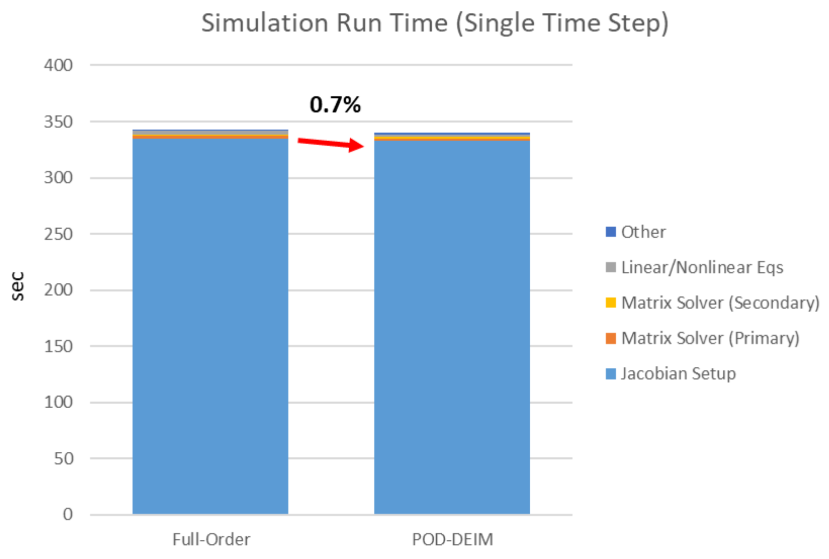


Figure 3.17 Comparison of simulation run time of a single time step in case 1. Note that about 98% of the total CPU time is spent in Jacobian setup. Simulation run time of a single time step is reduced by 0.7% using the POD-DEIM.

Table 3.8 shows the reduction of simulation run time by the POD-DEIM application in 1-year simulation in case 1. The total CPU time reduction of 1-year simulation is about 7% (Figure 3.18). More time reduction than a single time step mostly comes from the fact that some of time steps require a smaller number of iterations to convergence.

Table 3.8 Simulation run time of 1-year for the full-order model and for the reduced order model in case 1. Total simulation run time is reduced by the POD-DEIM application.

	Full-Order (sec)	POD-DEIM (sec)
Total Simulation Run Time (1 Year)	20,788.4	18,556.9

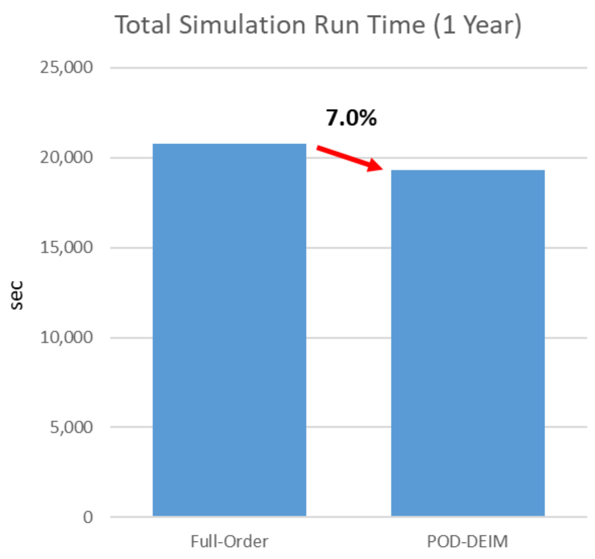


Figure 3.18 Comparison of simulation run time of 1-year in test case 1. The total CPU time reduction of 1-year simulation is about 7% by using the POD-DEIM.

Case 2 is the case where we change the residuals applying DEIM and the number of sampling locations. The POD is applied to all primary variables and the DEIM is applied only to the hydrocarbon component residuals with 150 sampling points out of 675 total gridblocks (Figure 3.19). The POD reduces the dimension of the system to 445 as it does in case 1, and we compute 2,550 nonlinear functions out of 6,750 by taking advantage of the DEIM. In sum, about 62.2% of nonlinear function computation is reduced.

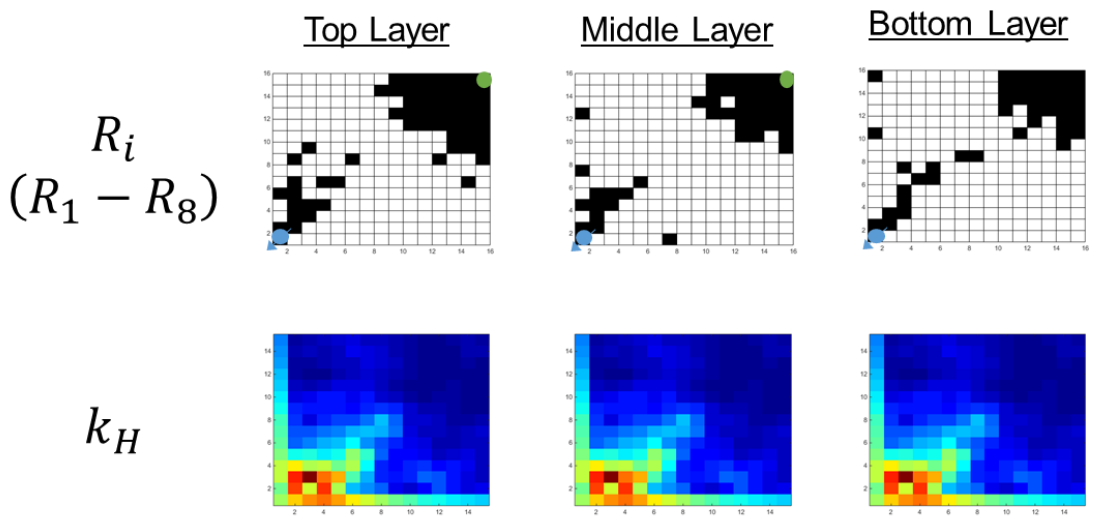


Figure 3.19 Sampling location for the DEIM in case 2. The selected 150 locations of the DEIM for R_i are shown.

The performance of the original simulator and the simulator with POD-DEIM are compared for case 2 (Figure 3.20). Case 2 also replicates very similar results to the full-order simulator for test input with less than 0.1% error.

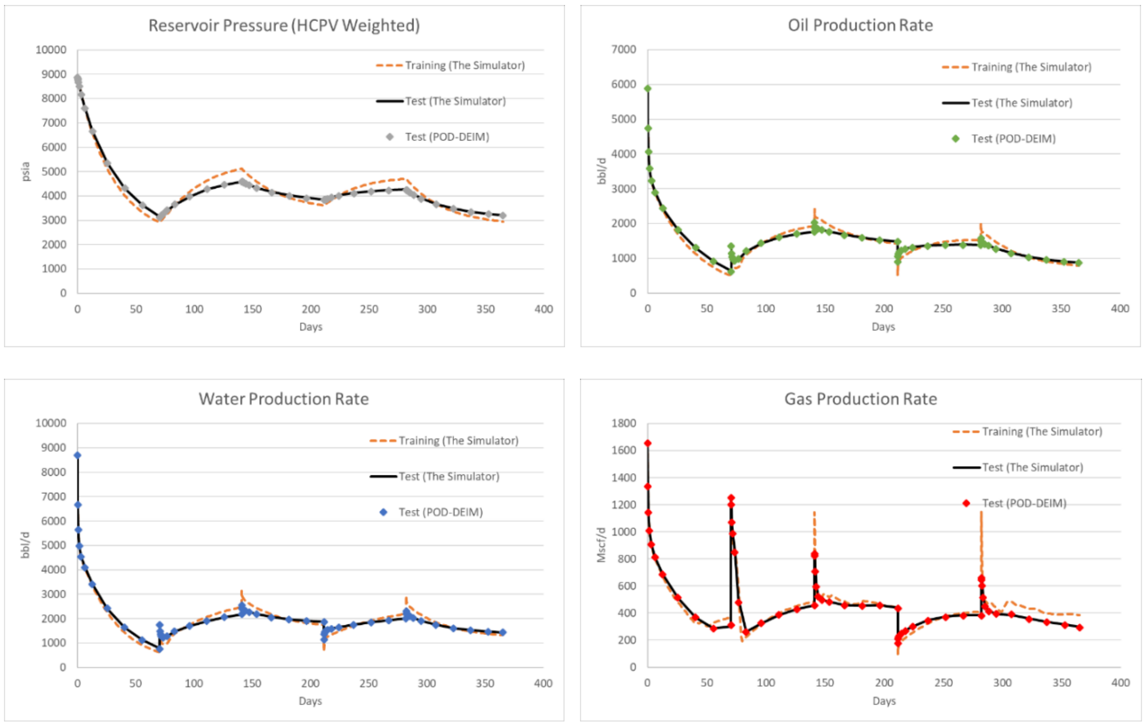


Figure 3.20 Comparison of the performance of training and case 2. Results of the POD-DEIM application show excellent estimation within less than 0.1% error.

Table 3.9 displays the reduction of simulation run time by the POD-DEIM in a single time step in case 2. For case 2, when we apply POD-DEIM, simulation run time of a single time step is reduced by 4.2% (Figure 3.21). Since in case 2, the number of locations for nonlinear function evaluation is even more reduced than case 1, we can see further time reduction in Jacobian setup and residual computation. In addition, POD reduces the elapsed time for primary matrix solver by 60%.

Table 3.9 Simulation run time of a single time step for the full-order model and the reduced order model in case 2. Further time reduction than case 1 is shown in Jacobian setup and residual computation due to the larger time reduction in nonlinear function evaluation by the DEIM.

	Full-Order (sec)	POD-DEIM (sec)
Jacobian Setup	335.0	322.1
Matrix Solver (Primary)	2.4	1.0
Matrix Solver (Secondary)	1.2	1.1
Linear/Nonlinear Eq.	3.0	2.0
Other	1.1	2.0

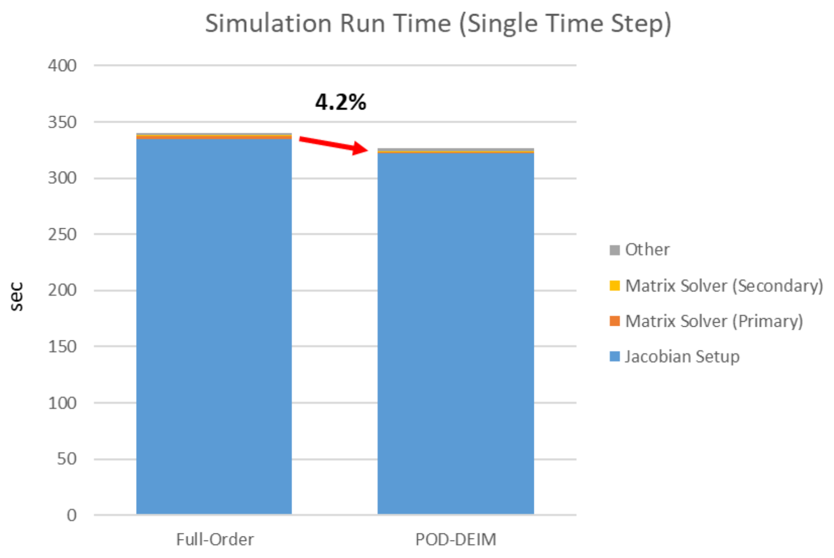


Figure 3.21 Comparison of simulation run time of a single time step in test case 2. Simulation run time of a single time step is reduced by 4.2% by the POD-DEIM application.

Table 3.10 describes the reduction of simulation run time by the POD-DEIM application in 1-year simulation in case 2. Total CPU time reduction of 1-year simulation is about 14.1%, that is more than twice time reduction comparing to case 1 (Figure 3.22).

Table 3.10 Simulation run time of 1 year for the full-order model and for the reduced order model in case 2. The reduction of total simulation run time in case 2 is much greater than case 1 since the number of nonlinear function evaluation is further reduced by the DEIM in case 2.

	Full-Order (sec)	POD-DEIM (sec)
Total Simulation Run Time (1 Year)	20,788.4	17,851.8

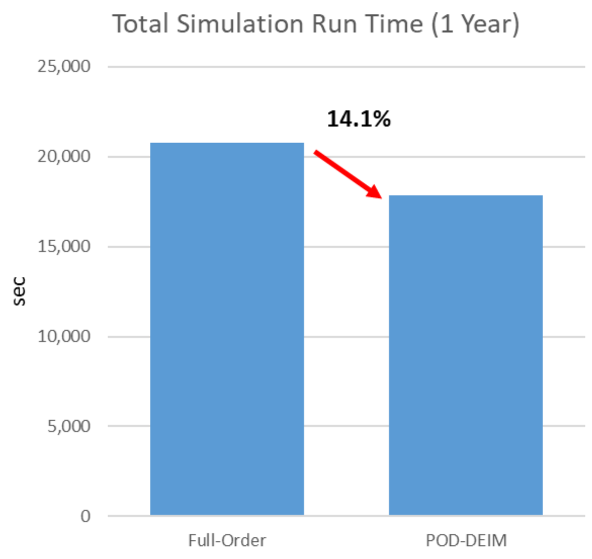


Figure 3.22 Comparison of simulation run time of 1 year in test case 2. Total CPU time reduction of 1-year simulation is about 14.1% by the POD-DEIM application.

Now we try even more reduction in DEIM sampling location by applying the DEIM to all residuals as illustrated in Figure 3.23 through Figure 3.25. When we select 100 or 200 points out of 675, simulation runs okay but requires much greater number of time steps due to the convergence issue. It seems that the convergence issue arises from the fact that we are approximating too many nonlinear functions at the same time. Therefore, we set case 3 as the case applying the POD-DEIM to all primary variables and residuals with 300 sampling points out of 675 total gridblocks. Case 3 has a little greater number of sampling points than case 2, but it is remarkable in the sense that we estimate all residuals using DEIM. The POD reduces the dimension of the system to 445 as case 1 and 2, and we compute 3,000 nonlinear functions out of 6,750 using DEIM. In sum, about 55.6% of nonlinear function computation is reduced.

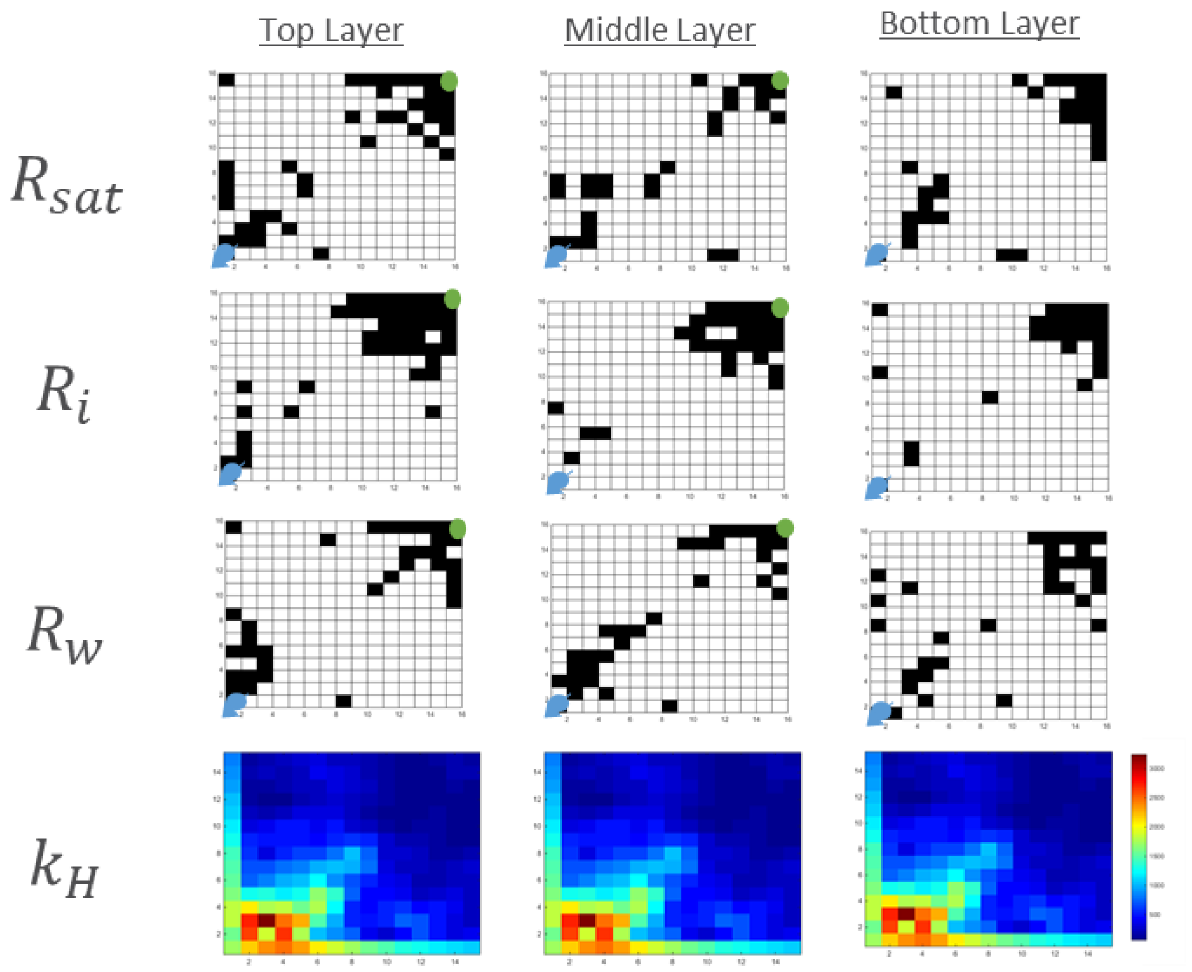


Figure 3.23 Sampling location for the DEIM in case 3. The selected 100 locations of the DEIM for all residuals are shown.

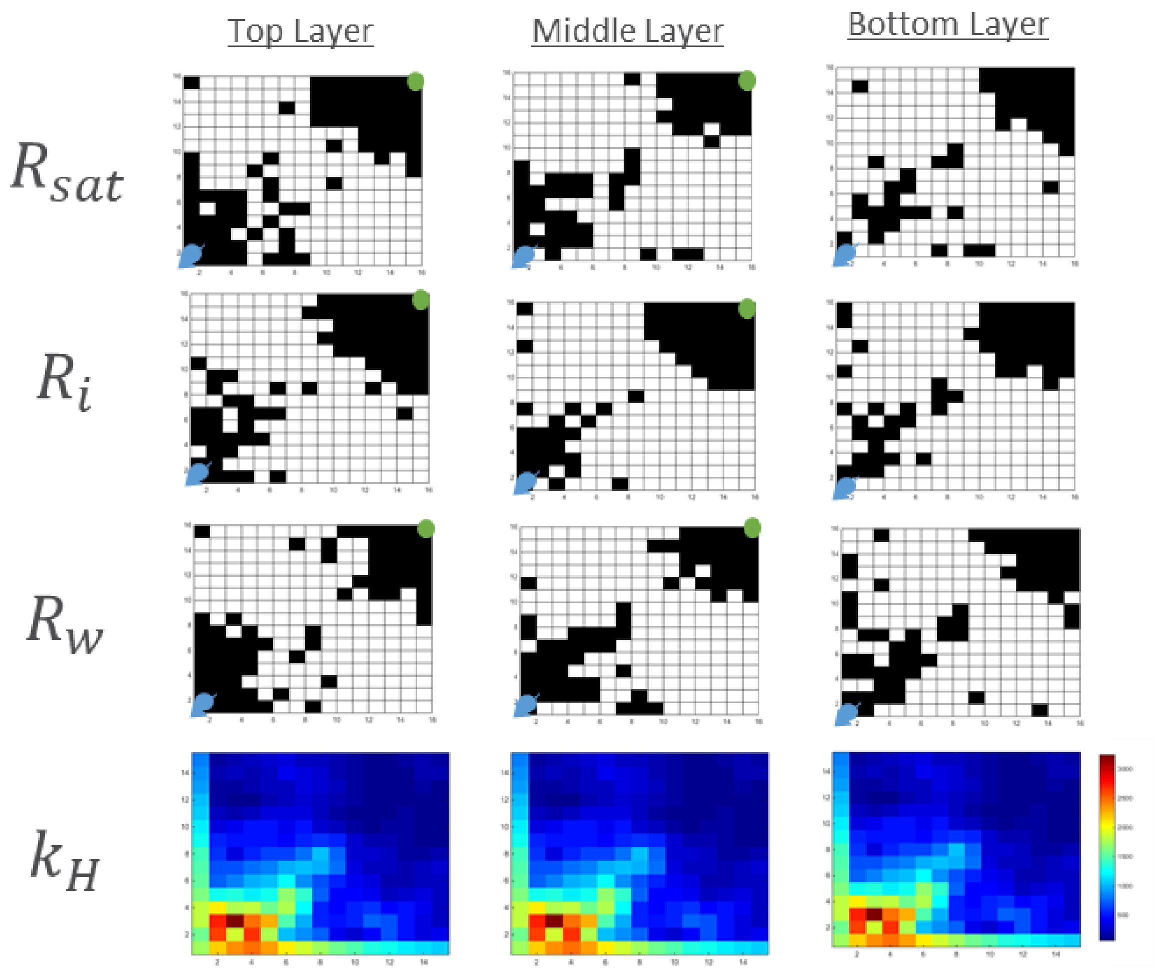


Figure 3.24 Sampling location for the DEIM in case 3. The selected 200 locations of the DEIM for all residuals are shown.

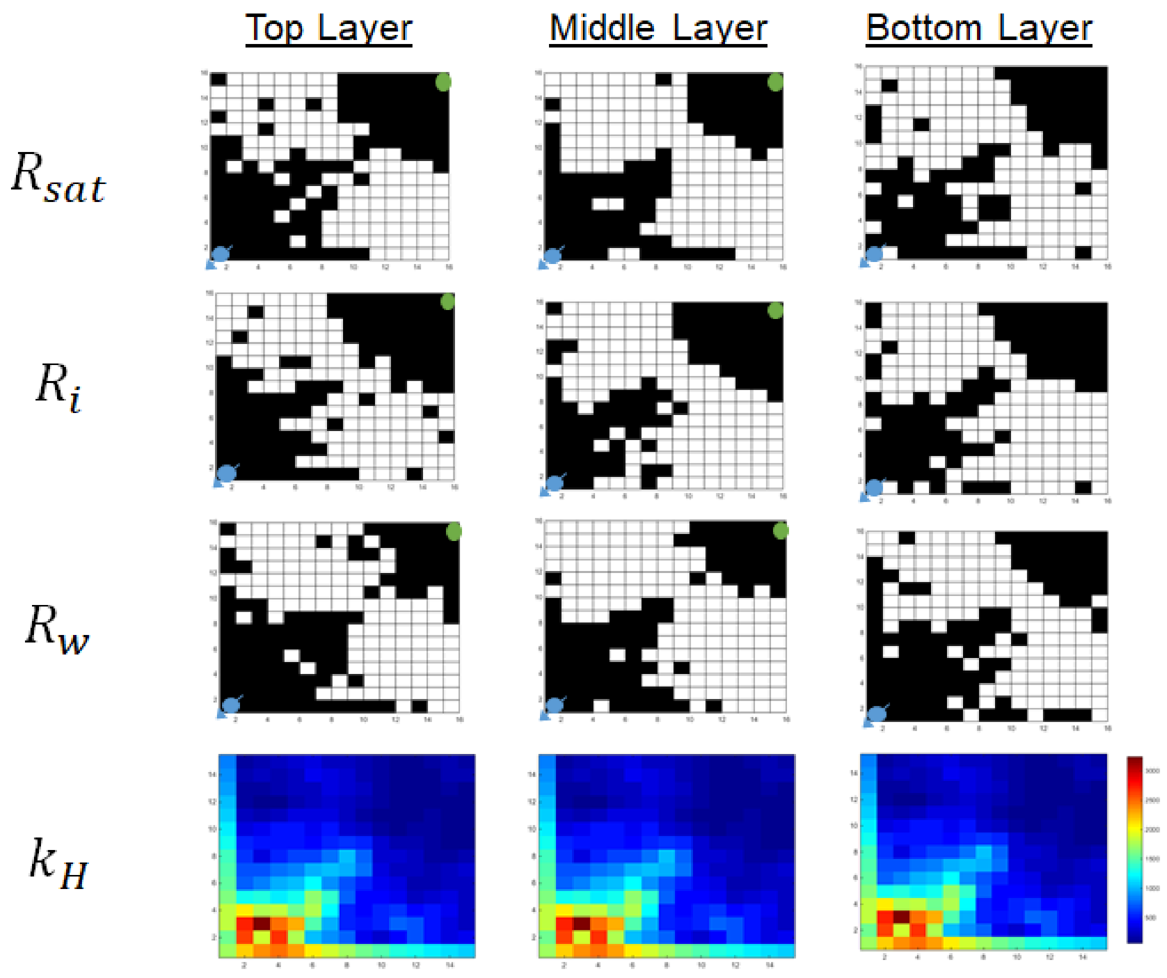


Figure 3.25 Sampling location for the DEIM in case 3. The selected 300 locations of the DEIM for all residuals are shown. Reprinted with permission from SPE-198946-MS.

The performance of the original simulator and the simulator with POD-DEIM are compared for case 3 (Figure 3.26). Case 3 also shows a strong match to the full-order simulator in test schedules with less than 0.1% error.

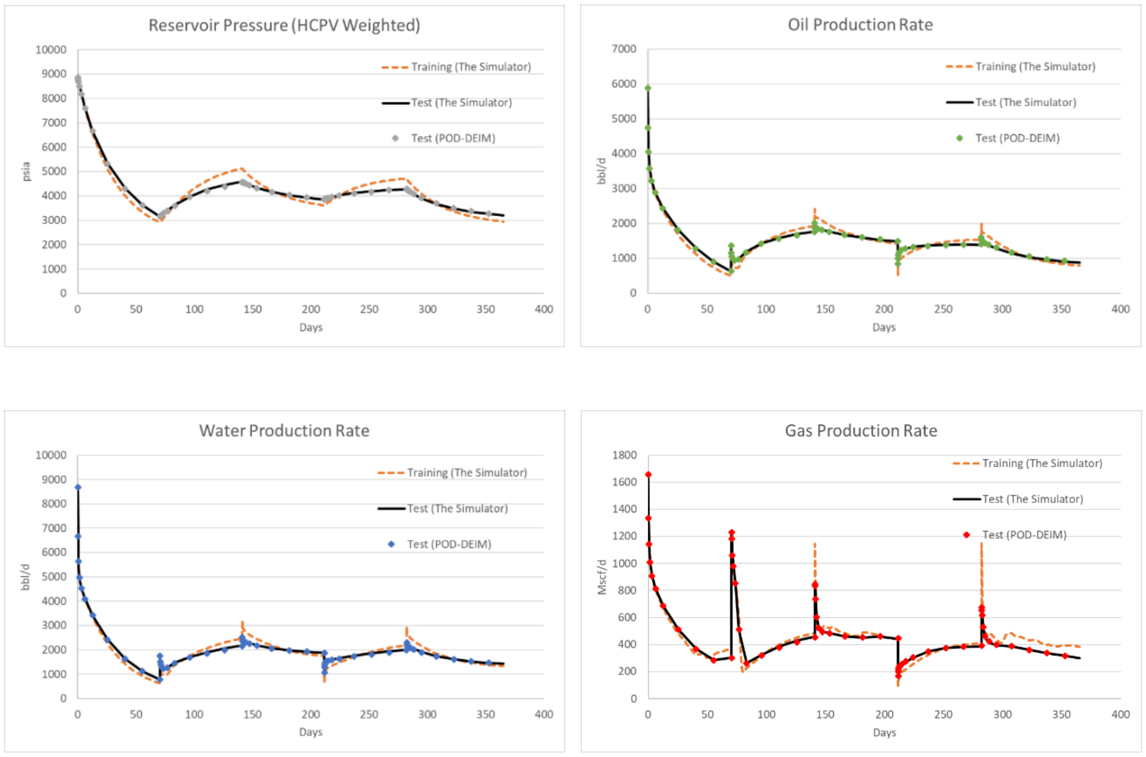
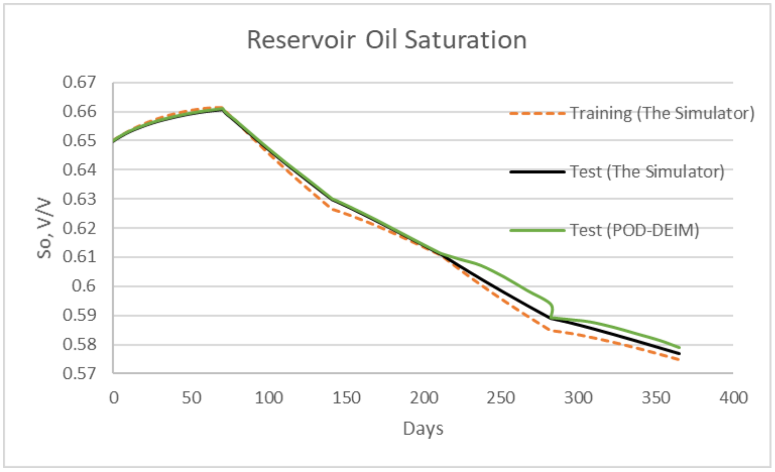
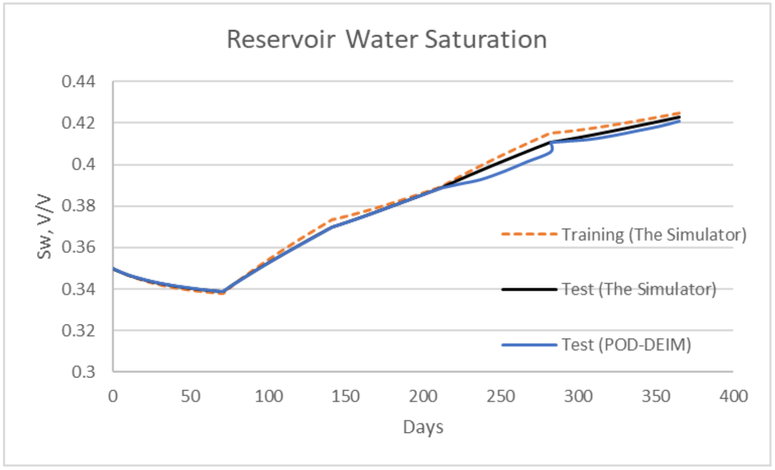


Figure 3.26 Comparison of the performance of training and case 3. Results of the POD-DEIM application show strong match within less than 0.1% error. Reprinted with permission from SPE-198946-MS.

Phase saturations in a full-order model and in a reduced model are compared to validate the consistency for three phase waterflooding scenario in Figure 3.27. They show a good match overall.





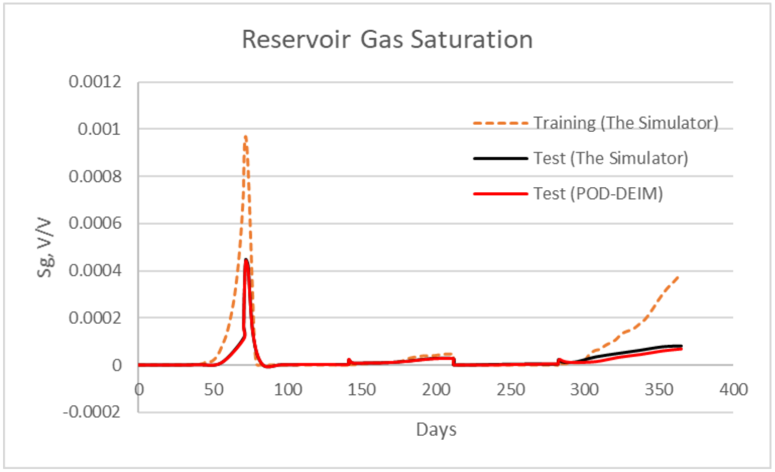


Figure 3.27 Comparison of phase saturation for training and test case 3. Comparisons shows a good match overall. Reprinted with permission from SPE-198946-MS.

The sum of moles of each component per pore volume is compared. As shown in Figure 3.28, compositions in the MOR applied case are almost identical to those in the original simulator. This implies that the POD-DEIM retains the compositional properties of the full-order model with high accuracy.

Since we assume the three phase waterflooding scenario in this study, the dynamic changes in compositions over time are not seen in the graph. However, because the simple compositional simulation with waterflooding using the proposed MOR method is validated, we can also extend it to gas injection scenarios to test the proposed MOR method.

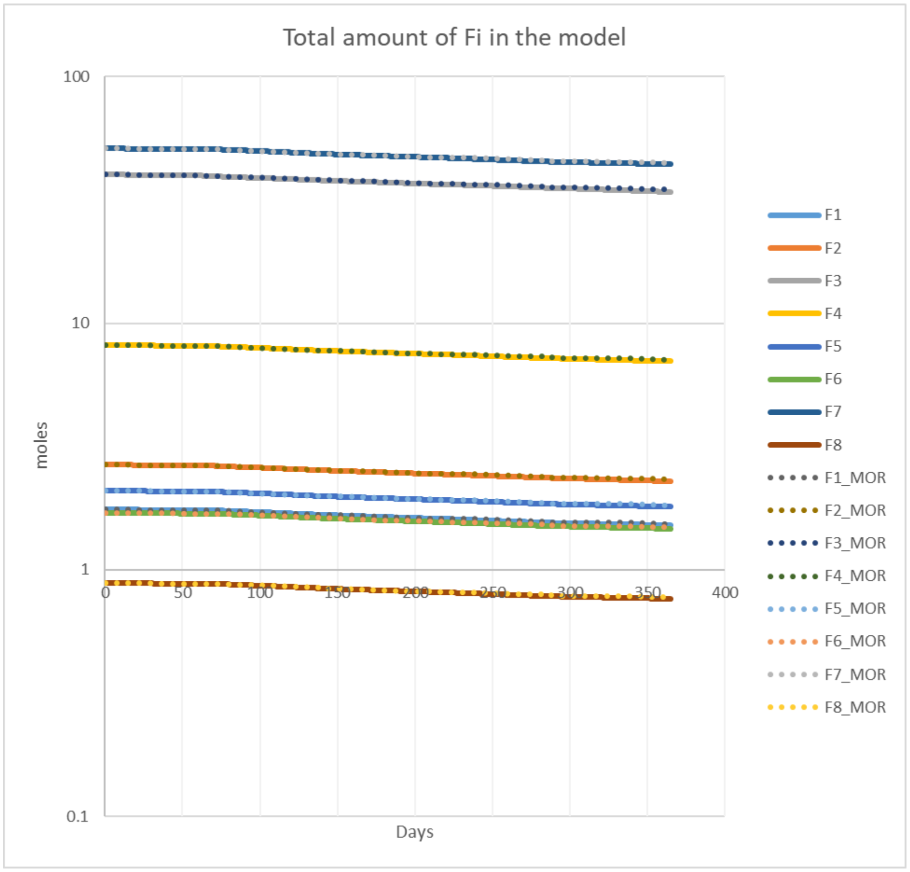


Figure 3.28 Comparison of compositions. Variation of compositions over time in test case 3 are almost identical to that in the original simulator. The POD-DEIM retains the compositional properties of the full-order model with high accuracy. Reprinted with permission from SPE-198946-MS.

Table 3.11 displays the reduction of simulation run time by the POD-DEIM in a single time step in case 3. For case 3, when we apply the POD-DEIM, simulation run time of a single time step is reduced by 3.3% (Figure 3.29). It is a little less time reduction than case 2 because case 3 has a little more DEIM sampling points than case 2.

Table 3.11 Simulation run time of a single time step for the full-order model and the reduced order model in case 3. Time reduction in case 3 is a little less than case 2 because case 3 has a little more DEIM sampling points than case 2.

	Full-Order (sec)	POD-DEIM (sec)
Jacobian Setup	335.0	324.7
Matrix Solver (Primary)	2.4	1.2
Matrix Solver (Secondary)	1.2	1.2
Linear/Nonlinear Eq.	3.0	2.1
Other	1.1	2.1

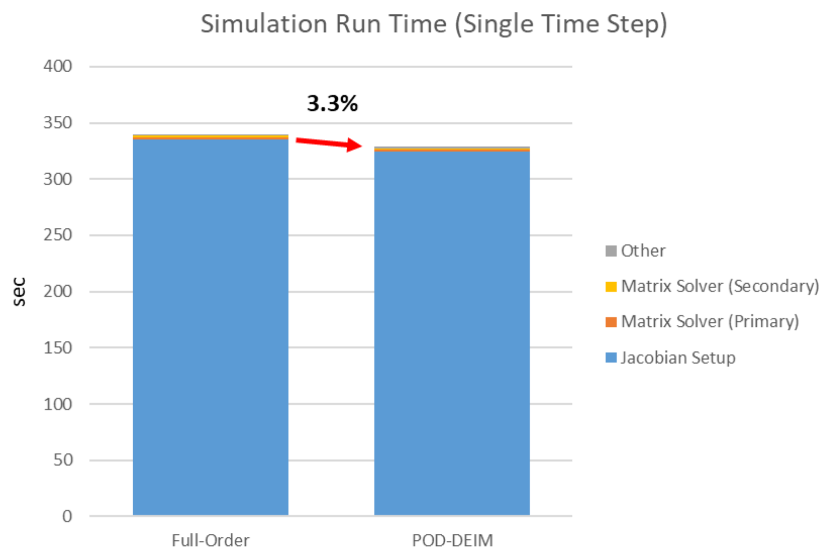


Figure 3.29 Comparison of simulation run time of a single time step in test case 3. Simulation run time of a single time step is reduced by 3.3% by the POD-DEIM application.

Table 3.12 shows the reduction of simulation run time by the POD-DEIM application in 1-year simulation in case 3. The total CPU time reduction of 1-year simulation is about 9.0% as shown in Figure 3.30.

Table 3.12 Simulation run time of 1-year for the full-order model and for the reduced order model. Reduction of total CPU time in case 3 is also a little less than that in case 2.

	Full-Order (sec)	POD-DEIM (sec)
Total Simulation Run Time (1 Year)	20,788.4	18,922.6

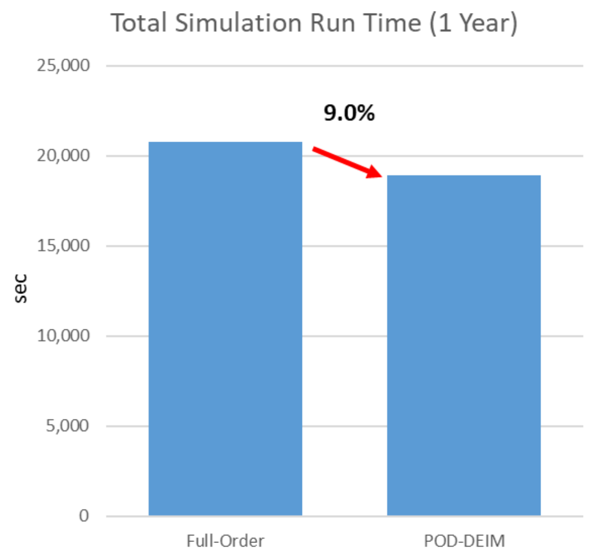


Figure 3.30 Comparison of simulation run time of 1-year in case 3. The total CPU time reduction of 1-year simulation in case 3 is about 9.0% by using the POD-DEIM.

In conclusion, one can confirm that the computational time is decreased by up to 14% when we apply the POD-DEIM in the developed compositional model. This speedup mainly results from the reduced computational time in the evaluation of the linear and nonlinear functions. However, although the computational time for mass balance is reduced by using the POD-DEIM method, it is not capable of applying the POD-DEIM for the flash calculation due to the characteristics of the molar variables formulations. Since the flash calculation is composed of many nonlinear terms and requires expensive computational cost, an additional methodology must be accompanied by the POD-DEIM to tackle this problem.

4. MACHINE LEARNING FOR PHASE EQUILIBIRUM CALCULATIONS

Machine learning is generally defined as the study of computer algorithms that improve automatically through experience (Mitchell 1997). Machine learnings, which are the subsets of artificial intelligence, as illustrated in Figure 4.1, have been promising techniques in many of the field. There has been controversy on the credibility of machine learning and its range of application, however it is now one of the widely used techniques for statistics and engineering, among other areas, by proving its ability to produce high reliability and accuracy.

Machine learning techniques are also extensively used in petroleum engineering in terms of prediction of reservoir performance, well design, well logging analysis, etc. This chapter investigate the applicability of machine learning techniques to the flash calculation in compositional simulation since the flash calculation is computationally expensive part containing complex nonlinear equations.

Firstly, the basic concept and the general procedure of flash calculation are introduced. Then, machine learning is briefly introduced, and more specifically ANNs are reviewed for the application in flash calculation. In this study, ANN is utilized to predict the equilibrium ratios (K values) and molar vapor fraction (f_v) in phase equilibrium calculation using reservoir pressure and fluid composition as inputs. Finally, stand-alone flash calculation case to investigate the performance of the applied ANN is shown, and rapid flash calculation integrated with POD-DEIM is reviewed for a robust reduced order modeling.

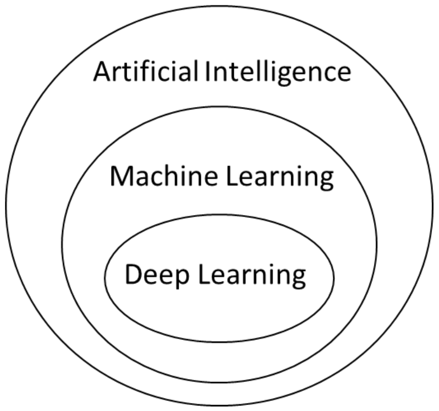


Figure 4.1 Hierarchy in artificial intelligence. Machine learning is a subset of artificial intelligence, and deep learning is a technique for realizing machine learning.

4.1. Phase Equilibrium Calculations

This section describes the fundamentals of phase equilibrium calculation and the conventional flash calculation process based on equations of state (EOSs).

4.1.1. Equilibrium Conditions

In the chemical potential concept, thermodynamic equilibrium is the state when the molecular transfer rate from liquid to vapor phase is equal to the molecular transfer rate from vapor to liquid phase, for all components. Since fugacity is a measure of the chemical potential, at thermodynamic equilibrium conditions, the fugacities for each component in the liquid-vapor mixture are equal (Eq. 4.1).

$$\hat{f}_i^l = \hat{f}_i^v, i = 1, \cdots, n_c \tag{4.1}$$

If we apply natural logarithm on both sides, Eq. 4.1 can be re-written as Eq. 4.2.

$$\ln\left(\frac{\hat{f}_i^v}{\hat{f}_i^l}\right) = 0, i = 1, \cdots, n_c \tag{4.2}$$

And by using the fugacity coefficient, the fugacity for a component in a mixture is re-written as Eq. 4.3 and Eq. 4.4 for vapor phase and liquid phase respectively.

$$\hat{f}_i^v = y_i \hat{\varphi}_i^v p \tag{4.3}$$

$$\hat{f}_i^l = x_i \hat{\varphi}_i^l p \tag{4.4}$$

Namely, Eq. 4.3 and Eq. 4.4 must be equal from the relationship in Eq. 4.1, as shown in Eq. 4.5.

$$y_i \hat{\varphi}_i^v p = x_i \hat{\varphi}_i^l p \tag{4.5}$$

Since the equilibrium ratio is defined as the proportion of vapor to liquid molar composition, the equilibrium condition in Eq. 4.5 can be written as Eq. 4.6.

$$K_i = \frac{y_i}{x_i} = \frac{\hat{\varphi}_i^l}{\hat{\varphi}_i^v} \tag{4.6}$$

The above equation can provide the fugacity residual (Eq. 4.7) when combining it with Eq. 4.2 through Eq. 4.4.

$$\ln K_i + \ln \hat{\varphi}_i^v - \ln \hat{\varphi}_i^l = 0 \tag{4.7}$$

4.1.2. Formulations for Flash Calculations

Two main formulations for flash calculations are discussed in this section: First, minimization of Gibbs free energy and second, direct solution of the nonlinear equations. After deriving the expression for the Gibbs free energy, several different algorithms for the minimization of the Gibbs free energy and the direct solution of the nonlinear equations are presented.

Gibbs free energy is defined as a thermodynamic potential associated with a chemical reaction that can be used to do reversible or maximum work at a constant temperature and pressure. The molar Gibbs free energy of a multicomponent, multiphase system is given as Eq. 4.8 (Okuno 2009).

$$\underline{G} = \sum_{j=1}^{N_p} \sum_{i=1}^{N_c} \beta_j x_{ij} \bar{G}_{ij} \tag{4.8}$$

Where,

β_j: mole fraction of phase j

x_{ij}: mole fraction of component i in phase j

$\bar{G}_{ij}$: partial molar Gibbs free energy of component i in phase j

N_p: number of phases

N_c: number of components

If we combine Eq. 4.3 and 4.4 with Eq. 4.8, the molar Gibbs free energy is expressed as shown in Eq. 4.9.

$$\begin{aligned} \underline{G} &= RT\sum_{j=1}^{N_p}\sum_{i=1}^{N_c}\beta_j x_{ij}\ln\left(x_{ij}\varphi_{ij}\right) + \underline{G}^{IG}\sum_{j=1}^{N_p}\sum_{i=1}^{N_c}\beta_j x_{ij} \\ &= RT\sum_{j=1}^{N_p}\sum_{i=1}^{N_c}\beta_j x_{ij}\ln\left(x_{ij}\varphi_{ij}\right) + \underline{G}^{IG} \end{aligned} \tag{4.9}$$

Where,

$\underline{G}^{IG}$: the molar Gibbs free energy of the ideal gas

Since $\underline{G}^{IG}$ depends on pressure and temperature that are fixed in the flash calculation, Eq. 4.9 is rewritten as Eq. 4.10. Hence, the minimization of Gibbs free energy is equivalent to the minimizing the Eq. 4.10.

$$\underline{G_R} = RT\sum_{j=1}^{N_p}\sum_{i=1}^{N_c}\beta_j x_{ij}\ln\left(x_{ij}\varphi_{ij}\right) \tag{4.10}$$

Where,

$\underline{G_R}$: the molar Gibbs free energy of the real gas

The function $\underline{G_R}$ is minimized subject to the conditions of Eq. 4.11 through Eq. 4. 14 using iterative methods.

$$\sum_{i=1}^{N_p} \beta_j x_{ij} = z_i \tag{4.11}$$

$$\sum_{i=1}^{N_c} z_i = 1.0 \tag{4.12}$$

$$\textstyle\sum_{i=1}^{N_p} \beta_j = 1.0 \text{ and } \beta_j \geq 0 \text{ for } j = 1, \cdots, N_p \tag{4.13}$$

$$\textstyle\sum_{i=1}^{N_c} x_{ij} = 1.0 \text{ and } x_{ij} \geq 0 \text{ for } i = 1, \cdots, N_c \text{ and } j = 1, \cdots, N_p \tag{4.14}$$

Gibbs free energy is the thermodynamic requirement for the phase equilibrium. In other words, the correct solution to the phase equilibrium problem must satisfy not only the necessary condition which is the fugacity equilibrium but also the global minimum of the Gibbs free energy. Hence, the minimization method for Gibbs free energy that can guarantee the global minimum is utilized due to its reliability (Teh and Rangaiah 2002). Examples of the iterative method for the minimization of the Gibbs free energy are successive substitution (SS), dominant eigenvalue method (DEM), and Newton's method (Pan and Firoozabadi 2003; Firoozabadi et al. 2007). The detailed derivation and procedure can be referred in the work by Pan and Firoozabadi (2003). The advantage of the SS is it eventually converges by keeping an appropriate direction however, it is extremely slow in the critical region. And Newton's method converges quadratically, but only locally. Therefore, the combined SS-Newton method in the critical region is used in practice.

The fugacity equilibriums that can be expressed in terms of equilibrium ratio (K value) and component mass balances form the set of equations for the direct solution of the nonlinear equations (Teh and Rangaiah 2002). Therefore, it is also called K value method and the mass balance approach. In this method, the governing equations consisting of mass balance, fugacity equilibrium, and mole fraction summation are solved numerically. The fugacity equilibrium is a necessary condition for the minimization of the Gibbs free energy, and its solution can be local minimum or global minimum. Therefore, the direct solution of the nonlinear equations may not guarantee the minimum of Gibbs free energy. Nonetheless, due to its flexibility that satisfies the mass balance only when convergence is achieved, the direct solution of the nonlinear equations is used in most vapor-liquid phase split computations (Okuno 2009). The same iterative methods in the minimization of Gibbs free energy can also be used for the direct solution method, and Figure 4.2 illustrates the process of the direct solution of the nonlinear equations.

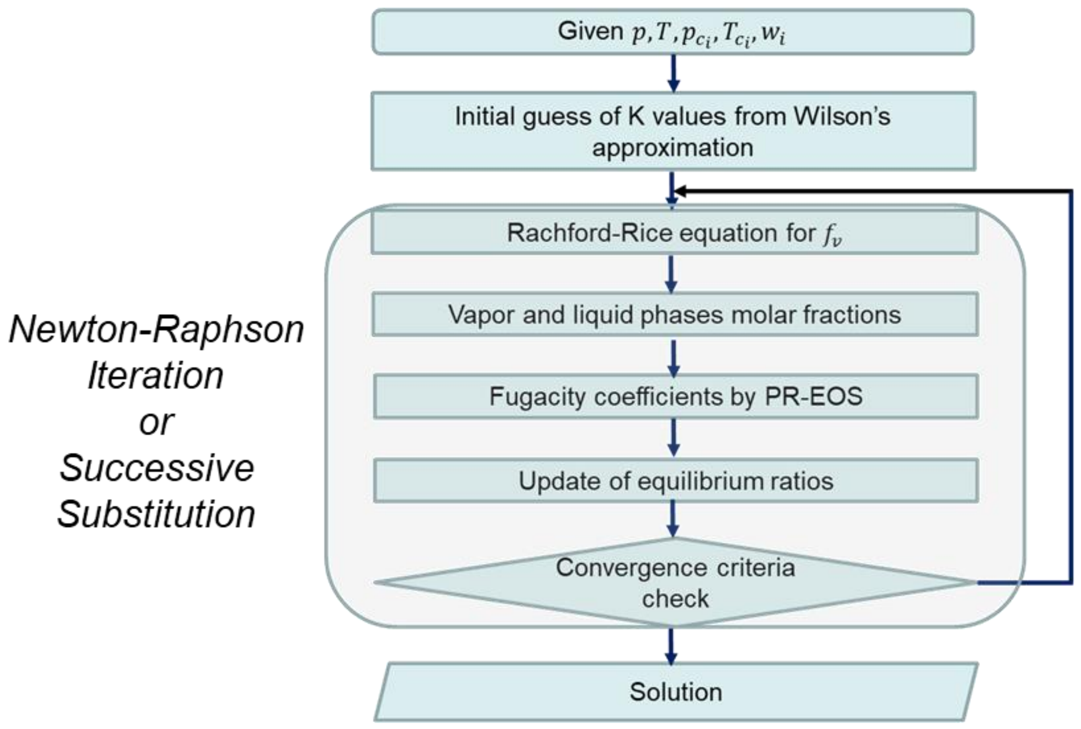

Given $p, T, p_{c_i}, T_{c_i}, w_i$
Initial guess of K values from Wilson's approximation
Rachford-Rice equation for f_v
Vapor and liquid phases molar fractions
Fugacity coefficients by PR-EOS
Update of equilibrium ratios
Convergence criteria check
Solution
Newton-Raphson Iteration or Successive Substitution

Figure 4.2 Process of conventional flash calculation using direct solution of the nonlinear equations. Flash calculation can be achieved through the numerical solution.

In this study, we use the direct solution of the nonlinear equations with successive substitutions, and the detailed procedures and equations of the method is discussed in section 4.4.

4.1.3. Equations of State

In this study, fluid properties and phase fugacities in vapor-liquid calculations are based on the Peng-Robinson equation of state (PR EOS). The equation form of the PR EOS is as shown in Eq. 4.15.

$$p = \frac{RT}{v - b} - \frac{(a\alpha)}{v^2 + 2vb - b^2} \tag{4.15}$$

Eq. 4.15 can be expressed using the compressibility factor $Z = \frac{pV}{nRT}$ as shown in Eq. 4.16.

$$Z^3 - (1 - B)Z^2 + (A - 3B^2 - 2B)Z - (AB - B^2 - B^3) = 0 \tag{4.16}$$

The detailed equations and procedure for PR EOS are described in Appendix B.

4.2. Basics of Machine Learning

Machine learning, a subset of artificial intelligence, has a wide range of applicability in various field of study, thus has been deemed one of the appealing prediction techniques.

This section briefly introduces the basic concepts of machine learning including its history, types, and applications.

4.2.1. Concept of Machine Learning

Machine learning is a field of computer science and a type of artificial intelligence that has evolved from pattern recognition and computational learning theory. In 1959, Arthur Samuel (1959) defined machine learning as a “Field of study that gives computers the ability to learn without being explicitly programmed”. After the definition of machine learning by Arthur Samuel, in 1997 Tom Mitchell (1997) provides a more modern definition:” A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E”. In other words, machine learning is a field of study to acquire new knowledge by providing a computer with a data set and by making it study as humans do. The goal of machine learning is to develop programs using algorithm that teach computers to learn and grow when exposed to new data, without any assistance of human.

4.2.2. History of Machine Learning

Machine learning was first studied in 1950’s from the field of games such as Chess or Go. And the concept of perceptron was first introduced in 1957. In 1969, Multi-layer perceptron was introduced, however as layers get more complex calculation becomes more complex, therefore it couldn’t solve the parameter value. In 1970’s and 1980’s, with the development of genetic algorithm, backpropagation, and others, machine learning began to flourish, and currently the breakthrough of deep learning, which is a technique for realizing machine learning based on artificial neural networks, drive artificial intelligence boom.

4.2.3. Types of Machine Learning

In general, there are 3 types of machine learning: supervised learning, unsupervised learning, and reinforcement learning.

Supervised learning is a process to study a labeled dataset as input and its corresponding output. The goal of the supervised learning is to approximate the mapping function from the input to the output, thus one can predict the output when the new input data is provided. Examples are classification and regression.

Unsupervised learning studies only an input without an output, and most data mining techniques such as clustering, anomaly detection, association, and autoencoders are the examples of unsupervised learning.

Reinforcement learning is a training to make a sequence of decisions through trial and error. It differs from the supervised learning in that it doesn't need the labelled input/output. Therefore, reinforcement learning is performed without the information of relationship between an input and an output, and it focuses on maximizing the total reward.

4.2.4. Applications of Machine Learning

Machine learning is utilized in almost all fields in computer science, and is also applied in object (letters, face, etc.) recognition, language processing, voice recognition, search engine, bioinformatics, computer games, and robotics.

In particular, in petroleum engineering, machine learning is widely used in well logging estimation, drilling automation, reservoir simulation, and in performance prediction workflows.

4.3. Basics of Artificial Neural Networks

This section describes the general concept of Artificial Neural Networks (ANNs) which is used to develop the networks in this study. Specifically, feed forward neural networks with backpropagation is discussed in detail.

4.3.1. Concept of Artificial Neural Networks

Algorithm is a set of rules to reach the solution of a given problem. ANNs can be defined as a type of algorithms that aim to mimic the human brain's decision-making process. As it becomes available to take advantage of more data, the field of neural networks are growing faster and bigger than ever. In conjunction with its rapid development, deep learning which is a branch of machine learning that uses various types of neural networks is also flourishing.

ANNs are networks of simple processing elements operating in parallel that maps an input space to an output space (Priddy and Keller 2005). ANNs are also known as feed forward neural networks (FNNs) since inputs are processed only in the forward direction. FNNs consist of 3 layers - input, hidden, and output - are composed of neurons. Neurons receive information from previous neurons, and process the information using activation function which contains weights and bias, as shown in Eq. 4.17 and Eq. 4.18 (Priddy and Keller 2005; Aggarwal 2018; Goodfellow et al. 2016).

$$\alpha = f(v) \tag{4.17}$$

$$v = \sum_{i=0}^{N_i} w_i x_i + b_i \tag{4.18}$$

Where,

α: output of a neuron

f: activation function

v: net stimuli of a neuron

N_i: number of inputs

w_i: weight

x_i: input value

b_i: bias

Figure 4.3 illustrates an example of the fully connected feed forward neural network.

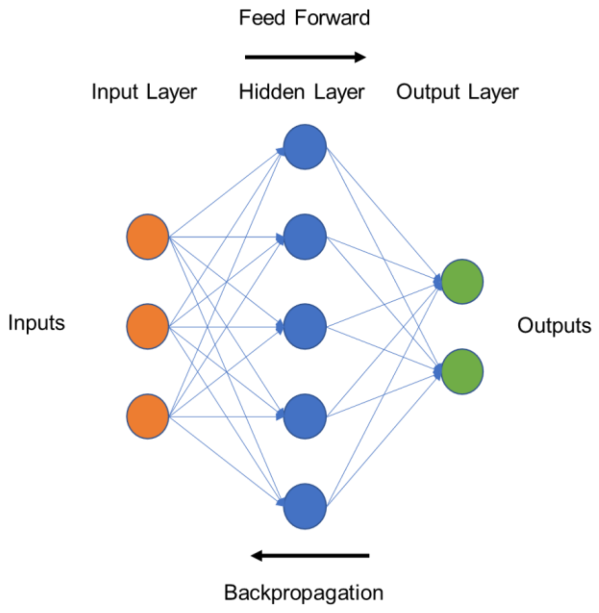


Figure 4.3 An example of ANNs. A fully connected FNN using backpropagation is illustrated.

There are many different types of activation functions depending on the purpose of the neural network. In particular, the nonlinear activation function helps ANNs be capable of learning any complex nonlinear function and weights that map any input to the output. Hence, modern neural network models use nonlinear activation functions for learning and modeling complex data. The examples of nonlinear activation functions are shown in Eq. 4.19 through Eq. 4.23.

$$\text{Sigmoid: } f(v) = \frac{1}{1+e^{-v}} \tag{4.19}$$

Tanh: $f(v) = \frac{e^{2v}-1}{e^{2v}+1}$ (4.20)

ReLU: $f(v) = max\{v, 0\}$ (4.21)

Softmax: $f(v) = \frac{e^{v_j}}{\sum_{j=1}^{J} e^{v_j}}$ for $j = 1, \cdots, J$ (4.22)

Hard Tanh: $f(v) = max\{min[v, 1], 0\}$ (4.23)

The model generalization is a process that trains the neural network to make accurate predictions by adjusting the weights for the given training data set. The training data is composed of set of paired input and output. And the weights in ANNs are adjusted based on the errors between the target and the prediction. The errors are evaluated by the loss function, and the mean square error J_{MSE} and the mean absolute error J_{MAE} are two commonly used loss functions (Eq. 4.24 and Eq. 4.25).

$$J_{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y}_i)^2 \quad (4.24)$$

$$J_{MAE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y}_i) \quad (4.25)$$

Where,

n: number of data samples

y_i: target value at i

$\bar{y}_i$: predicted value at i

The adjusted weights can be obtained by the backpropagation algorithm. Backpropagation algorithm adjusts the weights to find the accurate prediction by minimizing the value of loss function. The detailed procedure of backpropagation is described in Eq. 4.26 through Eq. 4.37 (Priddy and Keller 2005; Haykin 1994; Mejia 2019). In order to minimize the loss function, the gradient descent error minimization in terms of weights is introduced as shown in Eq. 4.26, and Eq. 4.26 can be rewritten using the chain rule as shown in Eq. 4.27.

$$\frac{\partial J}{\partial w_{ij}^{\ell}} \equiv 0 \tag{4.26}$$

$$\frac{\partial J}{\partial w_{ij}^{\ell}} = \frac{\partial J}{\partial v_{ij}^{\ell}} \frac{\partial v_{ij}^{\ell}}{\partial w_{ij}^{\ell}} \tag{4.27}$$

If we define the first term on the right in Eq. 4.27 as Eq. 4.28, and the second term on the right as Eq. 4.29 using Eq. 4.18, then Eq. 4.26 can be expressed as Eq. 4.30.

$$\Delta_{ij}^{\ell} = -\frac{\partial J}{\partial v_{ij}^{\ell}} \tag{4.28}$$

$$\frac{\partial v_{ij}^{\ell}}{\partial w_{ij}^{\ell}} = \frac{\partial}{\partial w_{ij}^{\ell}} \sum_{i=0}^{N_i} \left(w_{ij}^{\ell} x_{ij}^{\ell-1} \right) + b_{ij}^{\ell} = x_{ij}^{\ell-1} \tag{4.29}$$

$$\frac{\partial J}{\partial w_{ij}^{\ell}} = -\Delta_{ij}^{\ell} x_{ij}^{\ell-1} \tag{4.30}$$

The first term on the right-hand side of Eq. 4.27 is also expanded using the chain rule (Eq. 4.31), and it is expressed as Eq. 4.32 by using Eq. 4.17.

$$\frac{\partial J}{\partial v_{ij}^{\ell}} = \frac{\partial J}{\partial x_{ij}^{\ell}} \frac{\partial x_{ij}^{\ell}}{\partial v_{ij}^{\ell}} \tag{4.31}$$

$$\frac{\partial x_{ij}^{\ell}}{\partial v_{ij}^{\ell}} = \frac{\partial}{\partial v_{ij}^{\ell}} f^{\ell}\left(v_{ij}^{\ell}\right) = f'^{\ell}\left(v_{ij}^{\ell}\right) \tag{4.32}$$

Considered the mean square error, the final form of the derivative of the output error is expressed as shown in Eq. 4.33.

$$\frac{\partial J}{\partial x_{ij}^{\ell}} = \frac{\partial}{\partial x_{ij}^{\ell}} \left[\frac{1}{n} \sum_{i=1}^{n} (\bar{x}_i - x)^2 \right] = -(\bar{x}_i - x)\ for\ \ell = L \tag{4.33}$$

Where,

L: output layer

Since the variation of weight in a certain layer affects the next layer, one can describe the derivative of the output error as a sum of all the variations in the downstream network (Eq. 4.34).

$$\frac{\partial J}{\partial x_{ij}^{\ell}} = \sum_{k=1}^{N^{\ell+1}} \frac{\partial J}{\partial v_{ij}^{\ell+1}} \frac{\partial v_{ij}^{\ell+1}}{\partial x_{ij}^{\ell+1}}\ when\ \ell < L\ (Hidden\ Layers) \tag{4.34}$$

If we substitute the first and second terms on the right-hand side of Eq. 4.34 by using Eq. 4.18, Eq. 4.28 and Eq. 4.29, then Eq. 4.34 can be expressed as shown in Eq.

4.35.

$$\frac{\partial J}{\partial x_{ij}^{\ell}} = -\sum_{k=1}^{N^{\ell+1}} \Delta_{ik}^{\ell+1} w_{ik}^{\ell+1} \ when\ \ell < L\ (Hidden\ Layers) \tag{4.35}$$

The delta terms for output layer and hidden layer are defined as Eq. 4.36 and Eq. 4.37. As seen in Eq. 4.37, in order to obtain the delta term of hidden layer, the computation of the delta of output layer must be preceded. In other words, backpropagation stands for the error propagate backwards in the process of neural networks.

$$\Delta_{ij}^{L} = f'^{L}\left(v_{ij}^{L}\right)(y_i - \bar{y}_i)\ where\ \ell = L\ (Output\ Layer) \tag{4.36}$$

$$\Delta_{ij}^{\ell} = f'^{\ell}\left(v_{ij}^{\ell}\right)\sum_{k=1}^{\ell+1} \Delta_{ik}^{\ell+1} w_{ik}^{\ell+1} \ when\ \ell < L\ (Hidden\ Layers) \tag{4.37}$$

One can find various resources for ANNs using backpropagation algorithm such as MATLAB® Toolbox or Theano, TensorFlow, Keras in Python. We used MATLAB® Toolbox for the estimation of the variables in flash calculation in this study.

4.3.2. Data Treatment

Data preparation is necessary to avoid undertraining or overfitting. Whereas undertraining results from the range of the data and the sensitivity of the model, overfitting occurs when the network memorizes the training examples but fails to learn to generalize to new situations. Both cases present poor performance with the testing data.

Data normalization minimizes the bias of the networks by transforming the values of the variables to a consistent scale when the range of the variables are on different scales. Many different data normalization techniques can be found in the literatures (Priddy and Keller 2005; Gulli 2017), but the most commonly used technique is the min-max normalization.

In the min-max normalization, the matrices of a data set of both input and output are processed by normalizing the minimum and maximum values. In this study, we use the *mapminmax* keyword in MATLAB to map row minimum and maximum values to [-1, 1]. Eq. 4.38 describes the data normalization of *mapminmax*.

$$y = (y_{max} - y_{min}) \times \frac{(x - x_{min})}{(x_{max} - x_{min})} + y_{min} \tag{4.38}$$

4.4. ANNs for Flash Calculation

In chapter 3, the mass balance terms in residual equations including primary variables are reduced by using POD-DEIM method. As explained in chapter 3, the secondary variables - $\ln K_i$ and f_v which are related to the phase equilibrium equations - are not able to be reduced through POD-DEIM due to its nature of the molar variables formulations. However, phase equilibrium calculation is quite complex and consumes many portions of the total CPU time in a compositional simulation. In particular, the flash calculation which is composed of lots of nonlinear terms becomes the reason of expensive computational cost for phase equilibrium calculation. To tackle this problem, there have been many efforts for the rapid flash calculation. While the most of these efforts were focused on reducing the number of variables in the flash calculation in the past, more recent researches take advantage of the state-of-the-art machine learning techniques such as ANNs. The complex process of solving the phase equilibrium problem can thus be simplified, and faster solutions by ANNs can be obtained. As a result, the overall computational time of compositional model is reduced.

4.4.1. Formulation and Algorithm for Flash Calculation

The flash calculation can be implemented using successive substitution method or Newton-Raphson iteration (Michelsen and Mollerup 2007). The successive substitution method is used in this study, and procedures of the algorithm are summarized as follows.

1. Specify pressure (p), temperature (T), component critical pressure (p_{c_i}) and temperature (T_{c_i}), acentric factor (w_i), and molar fraction of the mixture.
2. Calculate an initial guess for the equilibrium ratio (K values) using Wilson's approximation in Eq. 4.39.

$$\ln K_i = \ln \frac{p_{c_i}}{p} + 5.373(1 + w_i)\left(1 - \frac{T_{c_i}}{T}\right) \tag{4.39}$$

3. Solve Eq. 4.40 for the vapor molar fraction f_v.

$$\sum_{i=1}^{n_c}(y_i - x_i) = \sum_{i=1}^{n_c} \frac{z_i(K_i - 1)}{1 + f_v(K_i - 1)} = 0 \tag{4.40}$$

4. Compute vapor and liquid molar fractions using Eq. 4.41 and Eq. 4.42.

$$x_i = \frac{z_i}{1 + f_v(K_i - 1)} \tag{4.41}$$

$$y_i = \frac{z_i K_i}{1 + f_v(K_i - 1)} \tag{4.42}$$

5. Calculate the fugacity coefficient for each component i and phase α using Eq. 4.43 through Eq. 4.45.

$$\ln(\hat{\varphi}_i^\alpha) = \frac{B_i}{B}(Z^\alpha - 1) - \ln(Z^\alpha - B^\alpha) + \frac{A^\alpha}{2\sqrt{2}B^\alpha}\left[\frac{B_i}{B^\alpha} - \frac{2\sum_{j=1}^{n_c} x_j^\alpha\,(a\alpha)_{ij}}{(a\alpha)^\alpha}\right]\ln(\beta^\alpha) \tag{4.43}$$

Where,

$$(a\alpha)_{ij} = \sqrt{a_i a_j \alpha_i \alpha_j}\left(1 - \kappa_{ij}\right) \tag{4.44}$$

$$\beta^\alpha = \frac{Z^\alpha + \left(1 + \sqrt{2}\right)B^\alpha}{Z^\alpha - \left(1 - \sqrt{2}\right)B^\alpha} \tag{4.45}$$

6. Update the equilibrium ratio K as shown in Eq. 4.46.

$$K_i^{k+1} = K_i^k\left(\frac{\hat{\varphi}_i^v}{\hat{\varphi}_i^l}\right) \tag{4.46}$$

7. Check for convergence using the criteria shown in Eq. 4.47 and repeat from Step-2 unless the criteria are met.

$$\sum_{i=1}^{n_c} \frac{z_i(K_i - 1)}{1 + f_v(K_i - 1)} \leq \varepsilon \tag{4.47}$$

4.4.2. Methodology for Application of ANNs in Flash Calculation

The solutions of two independent variables, $\ln K_i$ and f_v, that are related to flash calculation are obtained from two-step process in our deveoped simulator. In the first step, the system of equations for the second variables (variable for flash calculation) is solved in the Newton-Raphson iteration along with the primary variables (variables for mass balance), so that the solution of primary variables satisfies the condition of secondary variables. Then, in the second step, the solution of secondary variables is updated using flash calculation based on the obtained converged solution of primary variables such as pressure p and moles of each component F_i.

When it comes to the CPU time of the developed compositional simulator, the first step requires much more time than the second step. In addition, due to the sensitivity of the output of flash calculation, substituting the entire flash calculation with ANNs is very challenging. Therefore, we apply ANNs to replace the first step of solving the system of equations for logarithm of equilibrium ratio $\ln K_i$ and molar vapor fraction f_v, in this study. The obtained estimations from ANNs are then used for the input of flash calculation in the second step.

4.5. Applications and Results

As a first step to train the ANN models, a database is established considering the application range of input data and compositions constraint. The parameters for the input set are moles of components and pressure, since flash calculation is a function of composition and pressure. In this study, the number of components in the mixture is eight, and the summation of each component concentration must be unity. For pressure data, we vary the pressure from 3,400 to 8,900 psia considering the initial reservoir pressure of 8,868 psia in the developed simulator. The pressure intervals are set to be 100 psia, and concentration intervals are set to be 0.1. Hence, the total of 45,936 dataset is created for training, and min-max data normalization is performed to scale the variables to the range of -1 to 1. Once the database is generated, it is randomly split into 3 different datasets: 70% of training dataset, 15% of validation dataset, and 15% of testing.

There can be many different combinations of settings to build an ANN architecture by varying the number of hidden layers, number of neurons per hidden layer, and type of activation function. In this study, we adapt the number of hidden layers and the number of neurons in hidden layer from the ANN architecture created by Mejia (2019) which show excellent estimations for our output. Table 4.1 summarizes the settings of ANN, and Figure 4.4 illustrates the Tanh activation function that are used in this study.

Table 4.1 The developed ANN architecture.

Setting	Value
Activation function in Hidden layers	Tanh
Number of hidden layers	2
Number of neurons per hidden layer	20
Input	Pressure and mixture composition
Outputs	Molar vapor fraction and Natural logarithm of equilibrium ratio

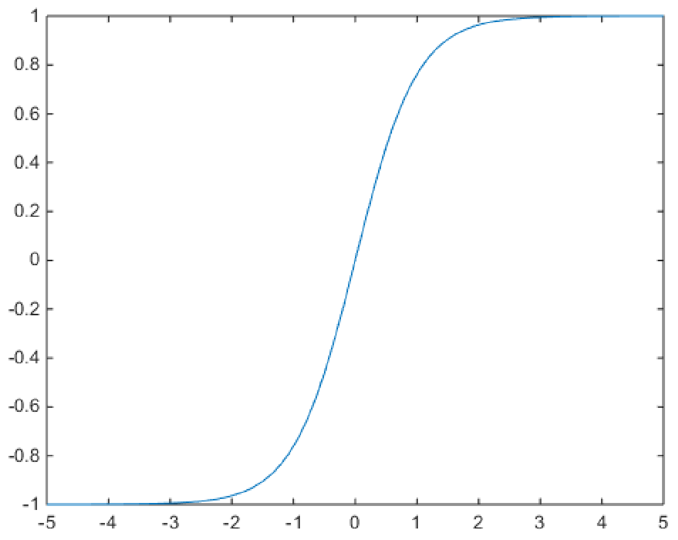


Figure 4.4 Tanh activation function used in the ANN.

The network for predicting flash calculation variables consists of an input layer, two hidden layers, and an output layer. Figure 4.5 shows the developed ANN architecture.

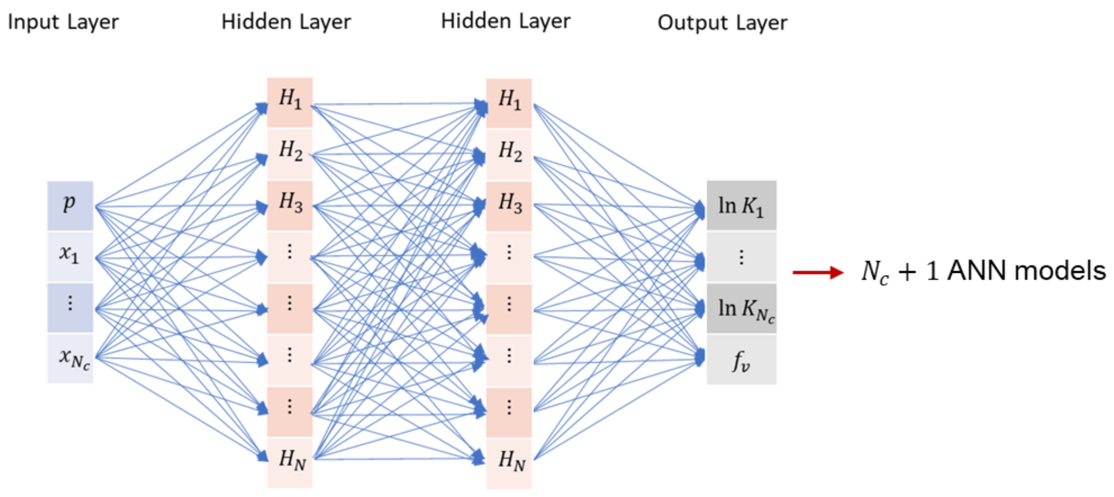


Figure 4.5 Artificial neural network model for flash calculation. Inputs are pressure and mixutre composition, and outputs are equilibrium ratio ($\ln K_i$) and molar vapor fraction (f_v). Since there are eight components in our model, nine ANN models are necessary for the prediction of each output.

Prior to the application of the developed ANNs into our in-house compositional simulator, the stand-alone flash calculation using ANNs is investigated to verify the performance of the developed ANNs. The stand-alone case study shows that the trained ANN models yield the highly accurate estimation to the target value, as shown in Table 4.2.

Table 4.2 Comparison between the target value from conventional flash calculation and the estimation from flash calculation using the developed ANN. Estimated values show a strong match to the target values.

	Target	Estimation	Accuracy
$\ln K_1$	-4.9783	-4.9820	99.93%
$\ln K_2$	-0.4171	-0.4201	99.29%
$\ln K_3$	0.6007	0.6000	99.89%
$\ln K_4$	-1.5858	-1.5875	99.89%
$\ln K_5$	-3.1528	-3.1459	99.78%
$\ln K_6$	-4.1146	-4.1145	99.99%
$\ln K_7$	-17.9604	-17.9617	99.99%
$\ln K_8$	-41.1543	-41.2108	99.86%
f_v	0.0002	0.0000	99.98%

We utilize the MATLAB® Toolbox to create, train and validate the ANN. Figure 4.6 displays the network created by MATLAB® Toolbox. Figure 4.7 and Figure 4.8 show the regression plots and error performance plots, respectively. Regression plots show around 99% of accuracy on prediction by the developed ANNs, which means the application of the ANNs to flash calculation is consistent and of practical use. Error performance plots indicate the best performance taken from the epoch with the lowest validation error.

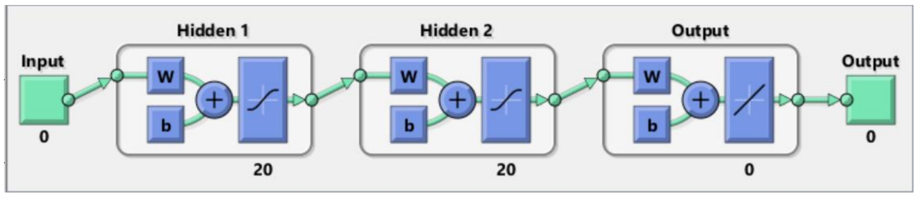


Figure 4.6 Function fitting neural network by MATLAB® Toolbox.

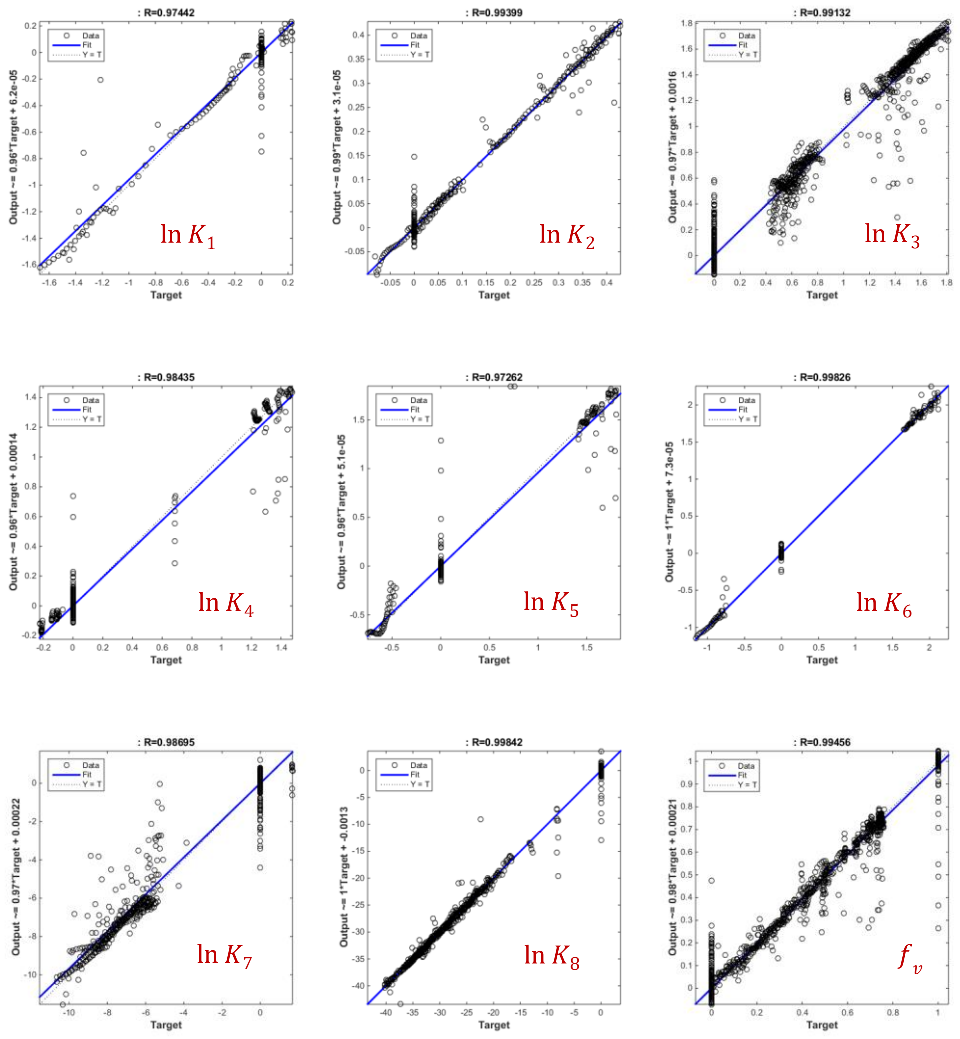


Figure 4.7 Regression plots. Around 99% of accuracy on prediction by the developed ANNs implies the application of the ANNs to flash calculation is practical.

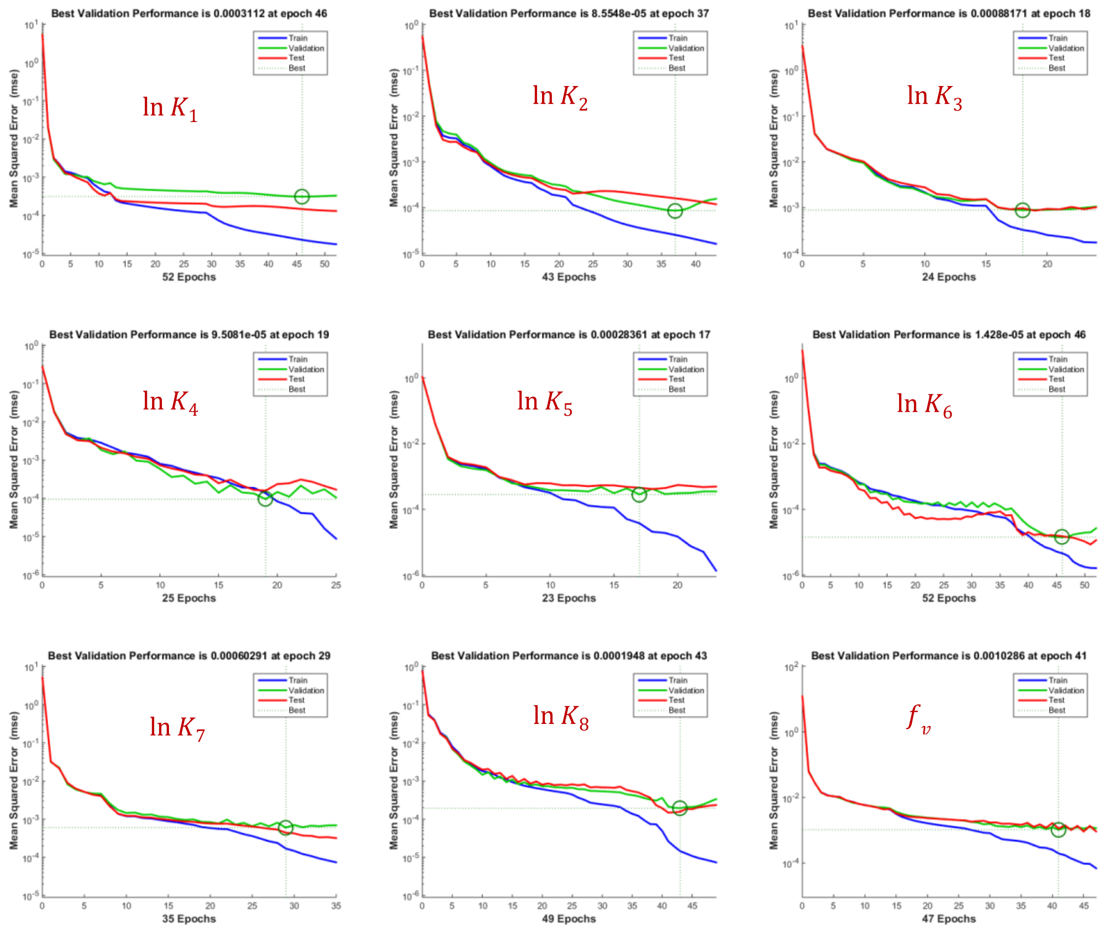


Figure 4.8 Error performance plots. Each parameter shows different number of epochs to take the best performance.

Owing to the robustness of the ANN applied to the stand-alone flash, we extended its application to the full reduced-order model workflow. In this case, we combined the ANN into the POD-DEIM framework. In particular, we started by using the previous described case 2. In case 2, the performance of the POD-DEIM with ANN model shows an excellent match to the full-order simulator in test schedules with less than 0.1% error (Figure 4.9).

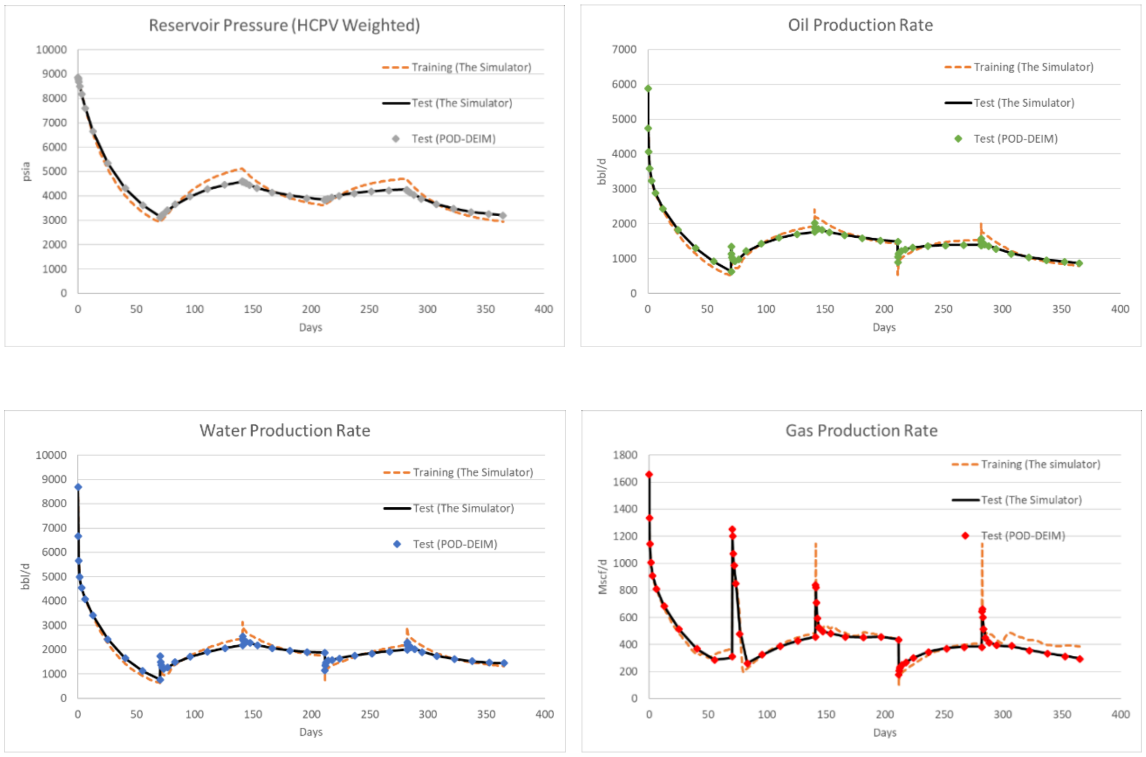


Figure 4.9 Comparison of training and test case 2. The performance of the POD-DEIM with ANN model shows an excellent match to the full-order simulator.

Table 4.3 and Figure 4.10 summarize the simulation run time reduction in a single time step in case 2. When we apply both POD-DEIM and ANN, simulation run time of a single time step is reduced by 4.5%, that is 0.4% more reduction comparing to the POD-DEIM only case. Time reduction in Jacobian setup and residual calculation arises from the DEIM, and reduction in primary matrix solver mostly results from the POD application. Since we substitute the process of solving the system by ANNs, no time is consumed in secondary matrix solver.

Table 4.3 Simulation run time of a single time step for the full-order model and the reduced order model in case 2. Application of ANNs eliminates the CPU time for solving secondary variables.

	Full-Order (sec)	POD-DEIM with ANNs (sec)
Jacobian Setup	335.0	321.8
Matrix Solver (Primary)	2.4	1.0
Matrix Solver (Secondary)	1.2	-
Linear/Nonlinear Eq.	3.0	2.1
Other	1.1	2.2

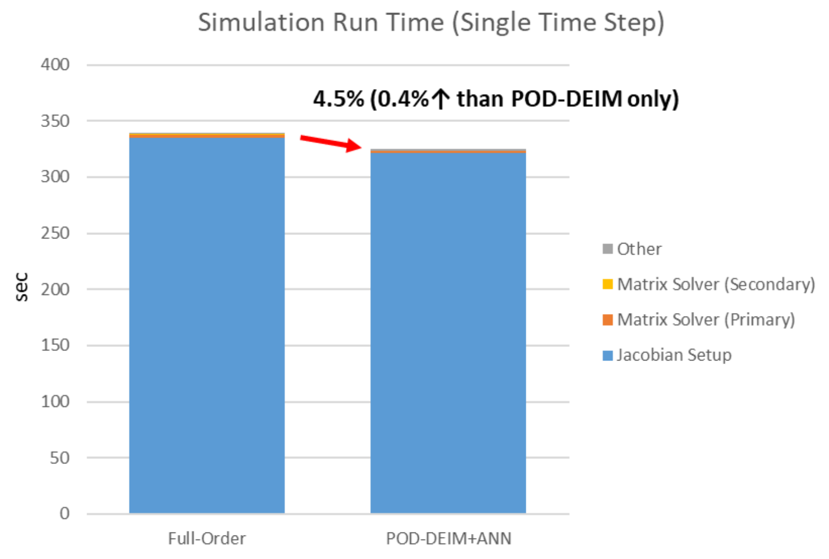


Figure 4.10 Comparison of simulation run time of a single time step in test case 2. Simulation run time of a single time step is reduced by 4.5% by using the POD-DEIM with ANNs.

The total CPU time reduction of 1-year simulation is about 14.6%, that is 0.5%

further reduction than the POD-DEIM only case (Table 4.4 and Figure 4.11).

Table 4.4 Simulation run time of 1-year for the full-order model and for the reduced order model in case 2. The total CPU time is reduced more when the ANNs are integrated with the POD-DEIM.

	Full-Order (sec)	POD-DEIM with ANNs (sec)
Total Simulation Run Time (1 Year)	20,788.4	17,762.3

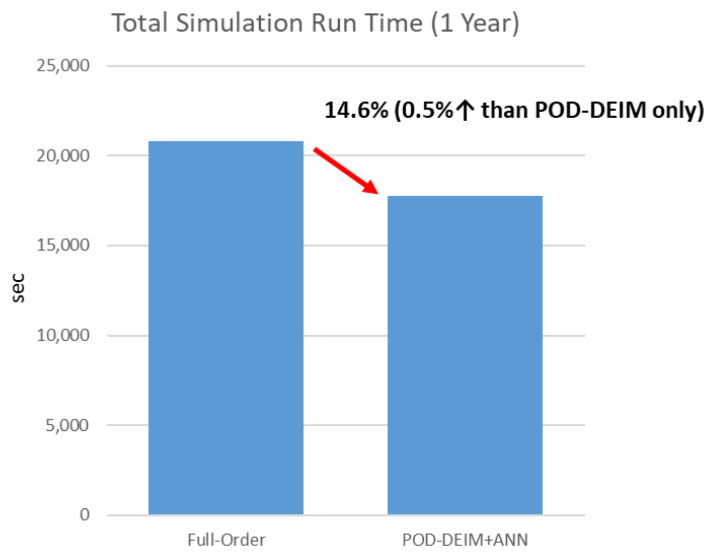


Figure 4.11 Comparison of simulation run time of 1-year in test case 2. The total CPU time reduction of 1-year simulation is about 14.6% by using the POD-DEIM with ANNs.

Coupled POD-DEIM and ANN is tested in case 3. In case 3, the performance of the POD-DEIM with ANN models also shows a strong estimation with less than 0.1% error (Figure 4.12).

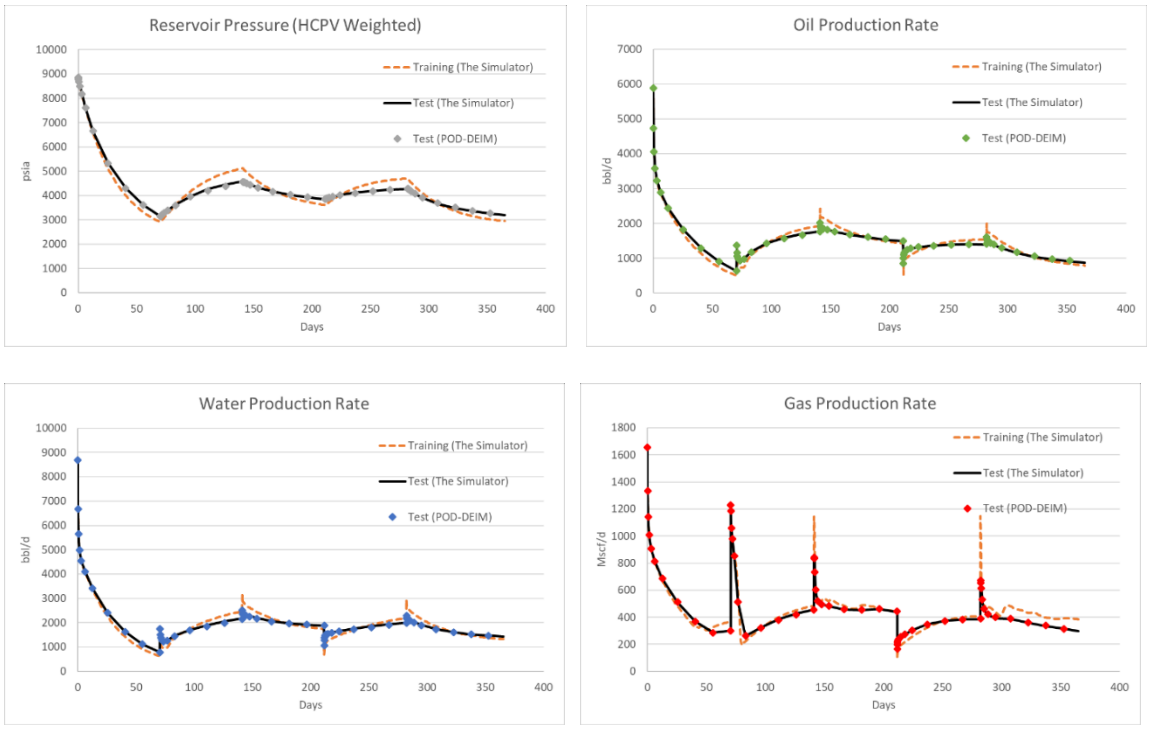


Figure 4.12 Comparison of training and test case 3. The performance of the POD-DEIM with ANN models shows a strong estimation with less than 0.1% error.

Table 4.5 and Figure 4.13 summarize the simulation run time reduction in a single time step in case 3. For case 3, simulation run time of a single time step is reduced about 3.8%, which is 0.5% more reduction comparing to the POD-DEIM only case. As seen in case 2, secondary matrix solver is skipped as we introduce ANN-based flash calculations.

Table 4.5 Simulation run time of a single time step for the full-order model and the reduced order model in case 3. Application of ANNs eliminates the CPU time for solving secondary variables.

	Full-Order (sec)	POD-DEIM (sec)
Jacobian Setup	335.0	324.4
Matrix Solver (Primary)	2.4	1.1
Matrix Solver (Secondary)	1.2	-
Linear/Nonlinear Eq.	3.0	2.1
Other	1.1	2.0

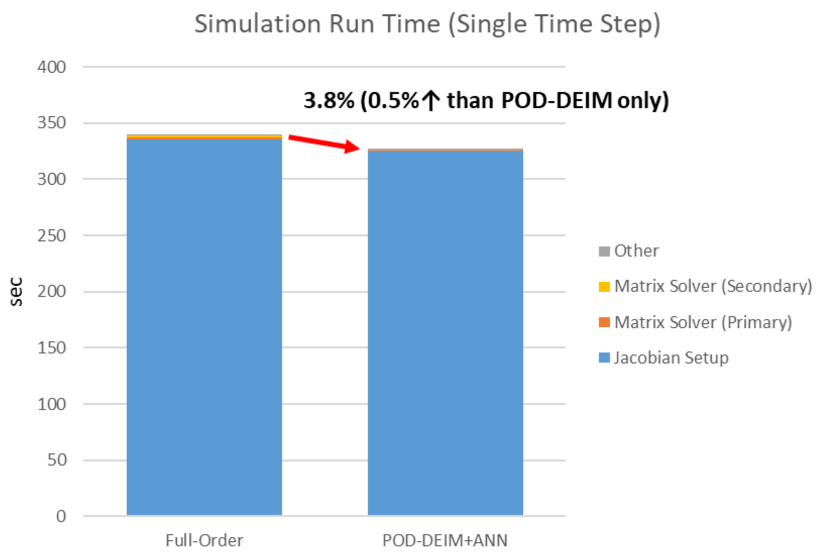


Figure 4.13 Comparison of simulation run time of a single time step in test case 3. Simulation run time of a single time step is reduced by 3.8% by using the POD-DEIM with ANNs.

The total CPU time reduction of 1-year simulation is about 10.7%, that is 1.7%

further reduction than POD-DEIM only case (Table 4.6 and Figure 4.14).

Table 4.6 Simulation run time of 1-year for the full-order model and for the reduced order model in case 3. The total CPU time is reduced more when the ANNs are integrated with the POD-DEIM.

	Full-Order (sec)	POD-DEIM (sec)
Total Simulation Run Time (1 Year)	20,788.4	18,556.9

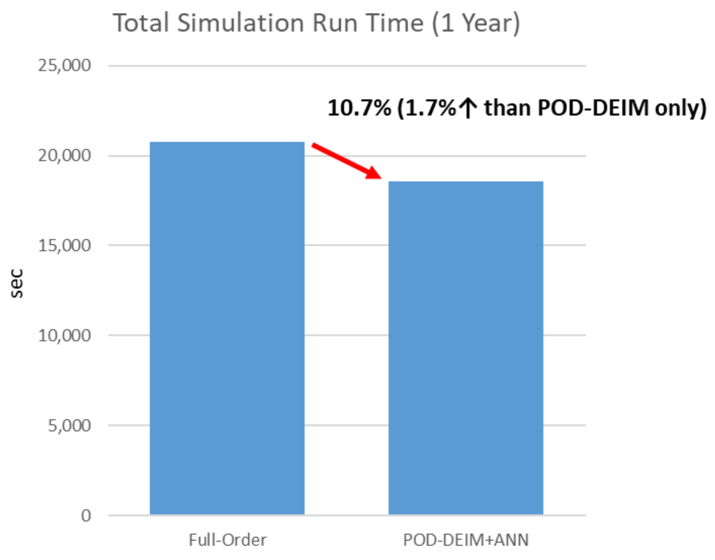


Figure 4.14 Comparison of simulation run time of 1-year in test case 3. The total CPU time reduction of 1-year simulation is about 10.7% by using the POD-DEIM with ANNs.

In conclusion, one can see that the additional reduction in computational time is achieved by applying the ANNs for flash calculation. As mentioned in chapter 1, although there have been many efforts applying the MOR methods or machine learning techniques for the faster computation in reservoir simulation, they were only capable of reducing the complexity and nonlinearity in either the mass balance calculation or the flash calculation. The robustness of this study is that the developed new framework combining the physics-based MOR (POD-DEIM) and the ANN-based MOR (ANNs for flash calculation) reduces the system of material balance and accelerate the flash calculation, simultaneously. By using this new framework, the reduction of computational time in both the mass balance calculation and the flash calculation can be achieved in compositional reservoir simulation.

5. CONCLUSTIONS AND FUTURE WORK

The demand for the compositional reservoir simulation has been increased as we develop reservoirs in deep water, unconventional gas reservoirs, and conventional reservoirs using gas injection. Nevertheless, the compositional simulation has been very challenging due to the complexity comparing to the black oil simulation and its expensive computational cost. In this study, we investigated the methodologies to enhance the efficiency of numerical computation and came up with a new framework for a faster but highly accurate compositional simulation using not only the physics-based MOR but also machine learning-based method. The robustness of the proposed new framework was validated in the various test cases.

This chapter reviews the findings from the study and summarizes the conclusions and the contributions. Furthermore, some recommendations and future works for further improvements are presented.

5.1. POD-DEIM

In this work, we proposed a new framework for implementing reduced order modeling techniques for multi-phase, multi-component simulation.

First, we developed our own compositional simulator based on the modified molar variables formulation method. We showed that our in-house simulator is as accurate as a commercial software. In order to apply the proposed MOR method, we additionally modified the simulator to have primary and secondary variables, so that we could apply the MOR to the terms that are related to the primary variables, independent of the flash calculation related secondary variables.

Then, we developed the reduced-order complexity-reduction techniques for simulations of multiphase compositional model. This technique reduces the system by projecting the governing on the subspace spanned by their POD-DEIM modes. DEIM in this case, is combined with POD to reduce the online computational cost for the nonlinear terms and to make it independent of the fine grid.

We presented numerical examples of 3D reservoir model to test the performance and accuracy of the reduced order model. We tested 3 different cases by changing the number of sampling locations for DEIM. The result of the proposed MOR technique proves its robustness on accuracy by showing the excellent agreement to the conventional simulator with the errors less than 0.1%. As we reduce the number of DEIM sampling locations, the CPU time also reduced accordingly. And the reduced order modeling using the POD-DEIM reduces the CPU time of the compositional simulation by maximum around 14% comparing to the fine scale model. This is a modest speedup, but we believe that in case of more complex physics, such as miscible gas injection, we would obtain speedups compared as the black oil formulations.

In summary, one can efficiently use the proposed model order reduction methods when performing fast and reliable compositional simulation.

5.2. Rapid Flash Calculation

Rapid flash calculation using the ANN-based machine learning technique was developed. The developed neural networks enabled us to substitute the process of solving the secondary independent variables in Newton Raphson iterations. Since this approach still retained the process of conventional flash calculation, it had the robustness that the simulations for flash calculation were conducted based on both physics and machine learning technique.

The result of the proposed ANN integrated with the physics-based MOR technique showed excellent agreement to the conventional simulator with the errors of less than 0.1%.

ANN-based MOR accelerated the flash calculation, and the coupled POD-DEIM and ANN technique reduced the CPU time of the compositional simulation by around 0.5% more than the POD-DEIM only cases. And the total CPU time when applying both POD-DEIM and ANN reduced by maximum around 14.5% comparing to the fine scale model.

5.3. Suggestions for Future Work

5.3.1. POD-DEIM

In this study, we implemented the compositional simulation with waterflooding scenario in order to make the cases as simple as possible. However, one cannot see the variation of compositions in the model through waterflooding. Therefore, gas injection such as nitrogen injection or enhanced oil recovery by CO_2 or enriched gas injection need to be investigated for more realistic and practical validation of the proposed methods. This will be able to be achieved by extending the developed compositional simulator for the gas injection scenarios, and we will leave it for future work.

The results of POD-DEIM application in compositional simulation showed case-sensitiveness. We tested the POD-DEIM concept in the case that 8 different components exist in the model, which makes somewhat too complicated to solve the system. In other words, having 8 components in the model implies that the total primary independent variables that need to be solved are 10, including pressure and mass of water. Therefore, solutions of 10 variables are estimated by POD-DEIM, and it results in the incorrect solutions at the next time step. Consequently, this prevented us to select much smaller number of DEIM sampling location. Thus, it resulted in only about 15% reduction in the total simulation time. Only 8 components case are investigated in this study, however it is suggested to test the cases with a smaller number of components in the model which are expected to show better performance and time efficiency by the POD-DEIM application.

As mentioned in chapter 2, one can build a compositional simulator using different set of variables and different system of equations. Even though the POD-DEIM technique is applied to the simulator based on a modified molar variables formulation in this study, it seems worthwhile to test them in any different type of compositional simulator.

5.3.2. Rapid Flash Calculation

CPU time for calling a trained network in order to predict the new output is affected by the number of hidden layers as well as the number of neurons in the hidden layers. In this study, 2 hidden layers and 20 neurons in each hidden layer are used to yield a best performance for accuracy, however one can try to reduce them and optimize the network so that the required CPU time to call a trained network can be shorter.

Due to the sensitiveness of the flash calculation it was unable to substitute the entire flash calculation with ANN, however one can try to apply ANN to replace the process of solving flow equations (expressed by primary independent variables) so that the solutions could be directly obtained from the input data without expensive computational cost of the Newton Raphson iterations.

REFERENCES

Acs, G., Doleschall, S., and Farkas, E. 1985. General Purpose Compositional Model. SPE Journal 25 (4): 543-553.

Aggarwal, C.C. 2018. Neural networks and deep learning. Cham: Springer International Publishing.

Aziz, K. and Settari, A. 1986. Petroleum Reservoir Simulation. Elsevier Applied Science Publishers.

Cardoso, M.A. and Durlofsky, L.J. 2010a. Linearized Reduced-Order Models for Subsurface Flow Simulation. Journal of Computational Physics 229 (3): 681-700. http://dx.doi.org/10.1016/j.jcp.2009.10.004.

Carlberg, K., Bou-Mosleh, C., and Farhat, C. 2011. Efficient non-linear model reduction via a least-squares Petrov-Galerkin projection and compressive tensor approximations. International Journal for Numerical Methods in Engineering, 86(2):155-181.

Carlberg, K., Farhat, C., Cortial, J., and Amsallem, D. 2013. The GNAT method for nonlinear model reduction: effective implementation and application to computational fluid dynamics and turbulent flows. Journal of Computational Physics, 242:623-647.

Chaturantabut, S. 2012. Nonlinear model reduction via discrete empirical interpolation. Diss., Rice University. https://hdl.handle.net/1911/70218.

Chaturantabut, S. and Sorensen, D.C. 2010. Nonlinear model reduction via discrete empirical interpolation. SIAM Journal on Scientific Computing, 32(5):2737-2764.

Chaturantabut, S. and Sorensen, D.C. 2011. Application of POD and DEIM on Dimension Reduction of Non-Linear Miscible Viscous Fingering in Porous Media. Mathematical and Computer Modeling of Dynamical Systems 17 (4): 337-53. http://dx.doi.org/10.1080/1387.954.2011.547660.

Chen, K. 2005. Matrix preconditioning techniques and applications, Cambridge University Press.

Chien, M.C.H., Lee, S.T., and Chen, W.H., 1985. A new fully implicit compositional simulator. SPE 13385.

Coats, K.H. 1980. An Equation of State Compositional Model. SPE Journal (October 1980): 363-376. DOI: 10.2118/8284-PA.

Collins, D.A., Nghiem, L.X., Li, Y.-K., and Grabonstotter, J.E., 1992. An efficient approach to adaptive implicit compositional simulation with an equation of state. SPERE 7(2), 259–264.

Concus, P. and Golub, G.H. 1976. A generalized conjugate gradient method for nonsymmetric systems of linear equations, in R. Glowinski and P.L. Lions eds., Lecture Notes in Economics and Mathematical Systems 134, 56–65, Springer.

Dietrich, J.K. and Bondor, P.L. 1976. Three-phase Oil Relative Permeability Models. Paper presented at SPE Annual Fall Technical Conference and Exhibition, New Orleans, Louisiana, 3-6 October. SPE-6044-MS.

Ertekin, T., Abou-Kassem, J.H., and King, G.R. 2001. Basic Applied Reservoir Simulation. Society of Petroleum Engineers.

Fanchi, J.R. 2006. Principles of Applied Reservoir Simulation. Gulf Professional Publishing.

Firoozabadi, A., Haugen, K.B., and Sun, L. 2007. Three-Phase Equilibrium Calculations for Compositional Simulation. Society of Petroleum Engineers. doi:10.2118/106045-MS.

Fussell, L.T. and Fussell, D.D., 1979. An iterative technique for compositional reservoir models. SPEJ 211–220 (August).

Gaganis, V. and Varotsis, N. 2012. Machine Learning Methods to Speed up Compositional Reservoir Simulation. EAGE Annual Conference & Exhibition incorporating SPE Europec. Society of Petroleum Engineers, June.

Gene H. Golub and Charles F. Van Loan. 1996. Matrix computations (3rd ed.). Johns Hopkins University Press, USA.

Ghasemi, M. and Gildin, E. 2015. Localized Model Reduction in Porous Media Flow. IFAC-Papers Online 48 (6): 242-47. http://dx.doi.org/10.1016/j.ifacol.2015.08.038.

Ghasemi, M., Yang, Y., Gildin, E., Efendiev, Y., and Calo, V. 2015. Fast Multiscale Reservoir Simulations using POD-DEIM Model Reduction. Society of Petroleum Engineers. doi:10.2118/173271-MS.

Gilbert, J.R., Moler, C., and Schreiber, R. 1992. Sparse matrices in matlab: design and implementation. SIAM J. Matrix Anal. Appl., 13(1):333-356.

Gildin, E., Ghasemi, M., Romanovskay, A., and Efendiev, Y. 2013. Nonlinear Complexity Reduction for Fast Simulation of Flow in Heterogeneous Porous Media. SPE Reservoir Simulation Symposium. Society of Petroleum Engineers, 2013. http://dx.doi.org/10.2118/163618-MS.

Golub, G.H. and Van Loan, C.F. 1996. Matrix computations (3rd ed.). MD, Baltimore: The Johns Hopkins Univ. Press.

Goodfellow, I., Bengio, Y., and Courville, A. 2016. Deep learning. MIT press.

Green, D.W. and Willhite, G.P. 1998. Enhanced Oil Recovery, Richardson, TX: Society of Petroleum Engineers.

Gulli, A., and Pal, S. 2017. Deep Learning with Keras. Packt Publishing Ltd.

Habiballah, W.A., Startzman, R.A., and Barrufet, M.A. 1996. Use of Neural Networks for Prediction of Vapor/Liquid Equilibrium K-Values for Light-Hydrocarbon Mixtures. SPE Reservoir Engineering, 11(02), 121-126. https://doi.org/10.2118/28597-PA.

Haykin, S. 1994. Neural networks: a comprehensive foundation. Prentice Hall PTR.

He, J. and Durlofsky, L.J. 2014. Reduced-Order Modeling for Compositional Simulation by Use of Trajectory Piecewise Linearization. SPE Journal 19, no. 5: 858-72. http://dx.doi.org/10.2118/163634-PA.

Hestenes, M.R. and Stiefel, E. 1952. Methods of conjugate gradients for solving linear problems, J. Res. Nat. Bur. Stand. 49, 409–436.

Jiang, R. and Durlofsky, L.J. 2018. Implementation and Detailed Assessment of a GNAT Reduced-Order Model for Subsurface Flow Simulation. Journal of Computational Physics. 379. 10.1016/j.jcp.2018.11.038.

Lee, J. and Gildin, E. 2020. A New Framework for Compositional Simulation Using Reduced Order Modeling Based on POD-DEIM. SPE Latin American and Caribbean Petroleum Engineering Conference. Society of Petroleum Engineers, July.

Li, J. Fan, X., Wang, Y. et al. 2020. A POD-DEIM reduced model for compressible gas reservoir flow based on the Peng-Robinson equation of state. Journal of Natural Gas Science and Engineering, Volume 79. 103367. ISSN 1875-5100. https://doi.org/10.1016/j.jngse.2020.103367.

Li, Y., Johns, R. T. 2006. Rapid Flash Calculations for Compositional Simulation. SPE Annual Technical Conference and Exhibition. Society of Petroleum Engineers, October.

Mejia, J.L.H. 2019. Application of Artificial Neural Networks for Rapid Flash Calculations. University of Texas at Austin, Austin, Texas, U.S.A. Retrieved from https://hdl.handle.net/2152/78612.

Michelsen, M. and Mollerup, J. 2007. Thermodynamic Models: Fundamentals & Computational Aspects. Trans. Ed Stenby, E. Denmark: Tie-Line Publications. Second Edition. ISBN 87-989961-3-4.

Mitchell, T. 1997. Machine Learning. McGraw Hill.

Mohaghegh, S. D., Liu, J. S., Gaskari, R., Maysami, M., and Olukoko, O. A. 2012. Application of Well-Base Surrogate Reservoir Models (SRMs) to Two Offshore Fields in Saudi Arabia, Case Study. Society of Petroleum Engineers. doi:10.2118/153845-MS.

Okuno, R. 2009. Modeling of phase behavior for gas flooding simulation. PhD dissertation, the University of Texas at Austin, Austin, Texas, U.S.A. Retrieved from http://hdl.handle.net/2152/10585.

Pan, H. and Firoozabadi, A. 2003. Fast and Robust Algorithm for Compositional Modeling: Part II - Two-Phase Flash Computations. Society of Petroleum Engineers. doi:10.2118/87335-PA.

Peaceman, D.W. 1978. Interpretation of Well-Block Pressure in Numerical Reservoir Simulation. Society of Petrloeum Engineers Journal 18 (3): 183-94. http://dx.doi.org/10.2118/6893-PA.

Péneloux, A., Rauzy, E., and Fréze, R. 1982. A Consistent Correction for Redlich-Kwong-Soave Volumes. Fluid Phase Equilibria 8 (1): 7-23. DOI: http://dx.doi.org/10.1016/0378-3812(82)80002-2.

Peng, D. -Y. and Robinson, D. B. 1976. A New Two-Constant Equation of State. Industrial & Engineering Chemistry Fundamentals, 15 (1), 59-64. DOI: 10.1021/i160057a011.

Prats, M. 1982. Thermal Recovery, SPE Monograph Series, Richardson TX: Society of Petroleum Engineers.

Priddy, K. L. and Keller, P. E. 2005. Artificial neural networks: an introduction (Vol. 68). SPIE press.

Rachford, H.H., Jr. and Rice, J.D. 1952. Procedure for Use of Electronic Digital Computers in Calculating Flash Vaporization Hydrocarbon Equilibrium. Petroleum Transactions 195 (Technical Note 136): 327-328. DOI: 10.2118/952327-G.

Redlich, O., Kwong, J. N. S. 1949. On the Thermodynamics of Solutions. V. An Equation of State. Fugacities of Gaseous Solutions. Chemical Reviews. 44 (1): 233–244. doi:10.1021/cr60137a013. ISSN 0009-2665. PMID 18125401.

Saad, Y. 2003. Iterative Methods for Sparse Linear Systems, second edition. Society for Industrial and Applied Mathematics.

Saad, Y. and Schultz, M.H. 1986. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J. Sci. Statist. Comput. 7, 856–869.

Samuel, A.L. 1959. Some Studies in Machine Learning Using the Game of Checkers. IBM J. Res. Dev., 3, 210-229.

Schlumberger. Eclipse Technical Description and Reference Manual 2014.1. Schlumberger.

Soave, G. 1972. Equilibrium constants from a modified Redlich-Kwong equation of state. Chemical Engineering Science. 27 (6): 1197–1203. doi:10.1016/0009-2509(72)80096-4.

Stone, H.L. 1973. Estimation of Three-Phase Relative Permeability and Residual Oil Data. Journal of Canadian Petroleum Technology, pp. 53ff.

Tan, X., Gildin, E., Trehan, S., Yang, Y., Hoda, N. 2017. Trajectory-Based DEIM TDEIM Model Reduction Applied to Reservoir Simulation. SPE Reservoir Simulation Conference. Society of Petroleum Engineers. https://doi.org/10.2118/182600-MS.

Teh, Y. S., Rangaiah, G. P. 2002. A Study of Equation-Solving and Gibbs Free Energy Minimization Methods for Phase Equilibrium Calculations. Chemical Engineering

Research and Design, Volume 80, Issue 7, Pages 745-759. ISSN 0263-8762. https://doi.org/10.1205/026387602320776821.

Valbuena Olivares, E. 2015. Production Performance Modeling Through Integration of Reservoir and Production Network with Asphaltene Deposition. Doctoral dissertation, Texas A & M University. Available electronically from http: / /hdl.handle.net /1969 .1 /155139.

Valbuena Olivares, E., Barrufet, M., & Killough, J. 2015. Forecasting Production Performance in Reservoir-Network Coupled Systems with Asphaltene Modeling. Society of Petroleum Engineers. doi:10.2118/174795-MS.

van der Vorst, H.A. 1992. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. SIAM J. Sci. Statist. Comput. 13. 631–644.

Voskov, D. and Tchelepi, H. 2012. Comparison of nonlinear formulations for two-phase multi-component EoS based simulation. Journal of Petroleum Science and Engineering. s 82–83. 101–111. 10.1016/j.petrol.2011.10.012.

Wang, K., Luo, J., Yan, L. et al. 2019. Artificial Neural Network Accelerated Flash Calculation for Compositional Simulations. Society of Petroleum Engineers. doi:10.2118/193896-MS.

Wang, P., Yotov, I., Wheeler, M. et al. 1997. A new generation EOS compositional reservoir simulator: part 1-formula-tion and discretization. SPE 37979.

Wang, S., Sobecki, N., Ding, D., Wu, Y.-S., and Zhu, L. 2019. Accelerated Compositional Simulation of Tight Oil and Shale Gas Reservoirs Using Proxy Flash Calculation. Society of Petroleum Engineers. doi:10.2118/193878-MS.

Weber, D., Edgar, F.T., Lake, L.W. et al. 2009. Improvements of the capacitance resistive modeling and optimization of large scale reservoirs. In SPE-121299-MS, editor, SPE Western Regional Meeting, San Jose, California.

Yang, Y., Ghasemi, M., Gildin, E., Efendiev, Y. and Calo, V. 2016. Fast Multiscale Reservoir Simulation with POD-DEIM Model Reduction. SPE Journal, June. http://dx.doi.org/10.2118/173271-PA.

Yoon, S., Alghareeb, Z. M., and Williams, J. 2014. Development of Reduced-order Oil Reservoir Models using Localized DEIM. SPE Annual Technical Conference and Exhibition. Society of Petroleum Engineers. https://doi.org/10.2118/170741-MS.

Yoon, S., Alghareeb, Z. M., and Williams, J. 2016. Hyper-Reduced-Order Models for Subsurface Flow Simulation. https://doi.org/10.2118/181740-PA.

Young, L. C. and Stephenson, R. E. 1983. A Generalized Compositional Approach for Reservoir Simulation. Society of Petroleum Engineers Journal. 23. 727-742. 10.2118/10516-PA.

Yousef, A. A., Gentil, P. H., Jensen, J. L., and Lake, L. W. 2005. A Capacitance Model To Infer Interwell Connectivity From Production and Injection Rate Fluctuations. Society of Petroleum Engineers. doi:10.2118/95322-MS.

APPENDIX A

Residuals for hydrocarbon component and water using molar variables formulation in compositional simulation are detailed in this appendix. The derivation of the residual is based on the mass balance equations for hydrocarbon component and water, and the finite difference method is applied for each hydraulic diffusivity equation (Valbuena Olivares 2015).

A.1. Hydrocarbon Component Diffusivity Equation

The diffusivity equation for hydrocarbon component can be expressed in the differential form as follows (Eq. A.1).

$$\nabla \cdot \left[\beta_c \vec{\vec{k}} A \left(x_i \frac{k_{ro}}{\mu_o} \nabla \Phi_o + y_i \frac{k_{rg}}{\mu_g} \nabla \Phi_g \right)\right] = V_b \frac{\partial}{\partial t}[\phi F_i] + \dot{n}_i, i = 1 to\, n_c \qquad \text{(A.1)}$$

Where,

k_{ro} and k_{ro}: oil and gas relative permeability, [dimensionless]

μ_o and μ_g: oil and gas viscosity, [cP]

x_i and y_i: liquid and vapor molar fractions of component i respectively, [lbmol/lbmol]

$\vec{\vec{k}}$: rock permeability tensor, [mD]

Φ_o and Φ_g: oil and gas potential, [psia]

A: area perpendicular to flow direction, [ft^2]

V_b: gridblock rock bulk volume, [ft^3]

ϕ: rock porosity, [ft^3/ ft^3]

F_i: number of moles of component i per unit pore volume, [lbmol/ ft^3]

$\dot{n}_i$: molar rate of component i from a well, [lbmol/ day]

n_c: the number of hydrocarbon components

β_c = 0.00633: the conversion constant for field units

The equation is comprised of three different terms to make up the mass balance relationship. The term on the left-hand side is convective flow, and the first term on the right represents accumulation. The last term on the right-hand side displays well source/sink. Each term can be expressed in the finite difference form using block-centered grid discretization in space and backward discretization in time for the fully implicit method.

A.1.1. Hydrocarbon Component Convective Flow

Convective flow term is a summation of fluxes from a gridblock to six different flux directions of neighboring gridblocks. For hydrocarbon component, fluxes of oil and gas phase are summed up as follows (Eq. A.2).

$$\nabla \cdot \left[a_o x_i \nabla \Phi_o + a_g y_i \nabla \Phi_g\right]$$
$$\begin{aligned} &= a_{oE} x_{iE} \nabla \Phi_{oE} + a_{oW} x_{iW} \nabla \Phi_{oW} + a_{oN} x_{iN} \nabla \Phi_{oN} \\ &+ a_{oS} x_{iS} \nabla \Phi_{oS} + a_{oB} x_{iB} \nabla \Phi_{oB} + a_{oT} x_{iT} \nabla \Phi_{oT} \\ &+ a_{gE} y_{iE} \nabla \Phi_{gE} + a_{gW} y_{iW} \nabla \Phi_{gW} + a_{gN} y_{iN} \nabla \Phi_{gN} \\ &+ a_{gS} y_{iS} \nabla \Phi_{gS} + a_{gB} y_{iB} \nabla \Phi_{gB} + a_{gT} y_{iT} \nabla \Phi_{gT} \end{aligned} \tag{A.2}$$

Where,

$a_{o\eta}$: oil transmissibility, $\eta = E, W, N, S, B, T$

$a_{g\eta}$: gas transmissibility, $\eta = E, W, N, S, B, T$

And oil and gas interblock transmissibility can be expressed as the multiplication of geometric transmissibility (T_η) and oil/gas mobility ($\lambda_{o\eta}\ and\ \lambda_{g\eta}$) (Eq. A.3 and Eq. A.4).

$$a_{o\eta} = T_\eta \lambda_{o\eta} \tag{A.3}$$

$$a_{g\eta} = T_\eta \lambda_{g\eta} \tag{A.4}$$

The interblock geometric transmissibility for the block-centered discretization is defined as the multiplication of permeability and area divided by the distance between the gridblocks. For the six neighboring gridblocks, the geometric transmissibility can be written as follows (Eq. A.5 through Eq. A.10), and these can be pre-computed before the time stepping since they are considered constant in the simulation.

$$T_E = 2\beta_c \left[\frac{(k\Delta y\Delta z)_i(k\Delta y\Delta z)_{i+1}}{(k\Delta y\Delta z)_i\Delta x_{i+1} + (k\Delta y\Delta z)_{i+1}\Delta x_i}\right] \tag{A.5}$$

$$T_W = 2\beta_c \left[\frac{(k\Delta y\Delta z)_i(k\Delta y\Delta z)_{i-1}}{(k\Delta y\Delta z)_i\Delta x_{i-1} + (k\Delta y\Delta z)_{i-1}\Delta x_i}\right] \tag{A.6}$$

$$T_N = 2\beta_c \left[\frac{(k\Delta x\Delta z)_j(k\Delta x\Delta z)_{j+1}}{(k\Delta x\Delta z)_j\Delta y_{j+1} + (k\Delta x\Delta z)_{j+1}\Delta y_j}\right] \tag{A.7}$$

$$T_S = 2\beta_c \left[\frac{(k\Delta x\Delta z)_j(k\Delta x\Delta z)_{j-1}}{(k\Delta x\Delta z)_j\Delta y_{j-1} + (k\Delta x\Delta z)_{j-1}\Delta y_j}\right] \tag{A.8}$$

$$T_B = 2\beta_c \left[\frac{(k\Delta y\Delta \mathrm{x})_k(k\Delta y\Delta x)_{k+1}}{(k\Delta y\Delta x)_k\Delta z_{k+1} + (k\Delta y\Delta x)_{k+1}\Delta z_k}\right] \tag{A.9}$$

$$T_T = 2\beta_c \left[\frac{(k\Delta y\Delta x)_k(k\Delta y\Delta x)_{k-1}}{(k\Delta y\Delta x)_k\Delta z_{k-1} + (k\Delta y\Delta x)_{k-1}\Delta z_k}\right] \tag{A.10}$$

Viscosity and density for oil/gas that are used to calibrate oil/gas mobility are volume-weighted and mass-weighted arithmetic averages as shown in Eq. A.11 and Eq. A.12.

$$\lambda_{o\eta} = \left(k_{ro}\frac{\tilde{\rho}_o}{\mu_o}\right)_\eta \tag{A.11}$$

$$\lambda_{g\eta} = \left(k_{rg}\frac{\tilde{\rho}_g}{\mu_g}\right)_\eta \tag{A.12}$$

Oil potential in block-centered gird method is expressed using oil phase pressure p_o and height difference Δz (Eq. A.13 and Eq. A.14).

$$\Delta\Phi_{o\eta} = \left(\Phi_{o\eta} - \Phi_{oC}\right) = \Delta \mathrm{p}_{o\eta} - \gamma_{o\eta}\Delta \mathrm{z}_{\eta} \tag{A.13}$$

$$\Delta\Phi_{o\eta} = \left(p_{o\eta} - p_{oC}\right) - g_c\left(\frac{\rho_{o\eta} + \rho_{oC}}{2}\right)\left(\mathrm{z}_{\eta} - \mathrm{z}_C\right) \tag{A.14}$$

Gas potential is also a function of oil phase pressure and height difference (Eq. A.15); however, the capillary pressure of oil and gas needs to be considered in this case (Eq. A.16).

$$\Delta\Phi_{g\eta} = \left(\Phi_{g\eta} - \Phi_{gC}\right) = \Delta \mathrm{p}_{o\eta} + \Delta \mathrm{p}_{cgo_{\eta}} - \gamma_{g\eta}\Delta \mathrm{z}_{\eta} \tag{A.15}$$

$$\Delta\Phi_{g\eta} = \left(p_{o\eta} - p_{oC}\right) + \left(p_{cgo_{\eta}} - p_{cgo_C}\right) - g_c\left(\frac{\rho_{g\eta} + \rho_{gC}}{2}\right)\left(\mathrm{z}_{\eta} - \mathrm{z}_C\right) \tag{A.16}$$

The final form of hydrocarbon component convective flow term can be rearranged and simplified using chain rule and chord-slope of capillary pressure as shown in Eq. A.17. The chord-slope of capillary pressure enables us to replace capillary pressure equation that are expressed as a function of saturation with the equation that is a function of water mass per pore volume, which is one of the primary independent variables.

$$
\begin{aligned}
\nabla \cdot \left[a_o x_i \nabla\Phi_o + a_g y_i \nabla\Phi_g\right] \\
&= \sum_{\eta=1}^{6}\left[a_{o\eta} x_i\left(\Delta p_{o\eta} - \gamma_{o\eta}\Delta z_\eta\right)\right] \\
&+ \sum_{\eta=1}^{6}\left[a_{g\eta} y_i\left(\Delta p_{o\eta} - \gamma_{g\eta}\Delta z_\eta\right)\right] \\
&+ \sum_{\eta=1}^{6}\left[a_{g\eta} y_i {p'}_{cgo_\eta} {S'}_{g_\eta}\Delta F_i\right]
\end{aligned}
\tag{A.17}
$$

A.1.2. Hydrocarbon Component Accumulation

Accumulation term is discretized in time using backward difference in order to constitute a fully implicit system of equations, and its final form can be expressed using chain rule and chord-slope similarly to the convective flow term (Eq. A.18 and Eq. A.19).

$$V_b \frac{\partial}{\partial t}[\phi F_i] = \frac{V_b}{\Delta t}[\Delta_\tau(\phi F_i)] = \frac{V_b}{\Delta t}[F_i^n \Delta_\tau(\phi) + \phi^{n+1}\Delta_\tau(F_i)] \tag{A.18}$$

$$V_b \frac{\partial}{\partial t}[\phi F_i] = \frac{V_b}{\Delta t}[F_i^n \phi'(p^{n+1} - p^n) + \phi^{n+1}(F_i^{n+1} - F_i^n)] \tag{A.19}$$

A.1.3. Hydrocarbon Component Residual

By combining all three terms which are discretized in space and in time, the final form of hydrocarbon component residual can be shown as follows (Eq. A.20).

$$R_i = \frac{V_b}{\Delta t}\left[F_i^n \phi'^{(p_{oC}^{n+1} - p_{oC}^n)} + \phi^{n+1}(F_i^{n+1} - F_i^n)\right]$$

$$- \sum_{\eta=1}^{6}\left[a_{o\eta}^{n+1} x_i^{n+1}\left(\Delta p_{o\eta} - \gamma_{o\eta}^{n+1}\Delta z_\eta\right)\right]$$

$$- \sum_{\eta=1}^{6}\left[a_{g\eta}^{n+1} y_i^{n+1}\left(\Delta p_{o\eta} - \gamma_{g\eta}^{n+1}\Delta z_\eta\right)\right] \tag{A.20}$$

$$- \sum_{\eta=1}^{6}\left[\left(a_{g\eta} y_i p'_{cgo_\eta} S'_{g_\eta}\right)^{n+1} \Delta F_i\right]$$

$$+ WI_o^{n+1} x_i^{n+1}\left(p_{oC}^{n+1} - p_{wf}^{n+1}\right)$$

$$+ WI_g^{n+1} y_i^{n+1}\left(p_{oC}^{n+1} + p_{cow}^{n+1} - p_{wf}^{n+1}\right)$$

A.2. Water Diffusivity Equation

The diffusivity equation for water can be expressed in the differential form as Eq. A.21 in a similar way to the hydrocarbon component diffusivity equation.

$$\nabla \cdot \left[\beta_c \vec{\vec{k}} A \frac{k_{rw}}{\mu_w} \nabla \Phi_w\right] = V_b \frac{\partial}{\partial t}[\phi \mathrm{W}] + \rho_w q_w \tag{A.21}$$

Where,

k_{rw}: water relative permeability, [dimensionless]

μ_w: water viscosity, [cP]

$\vec{\vec{k}}$: rock permeability tensor, [mD]

Φ_w: water potential, [psia]

A: area perpendicular to flow direction, [ft^2]

V_b: gridblock rock bulk volume, [ft^3]

ϕ: rock porosity, [ft^3/ ft^3]

W: mass of water rate from a well, [ft^3/ day]

ρ_w: water density, [lb/ft^3]

β_c = 0.00633: the conversion constant for field units

The equation is also composed of three different terms to make up the mass balance relationship, which are convective flow, accumulation, and well source/sink. Each term can be expressed in the finite difference form using central difference discretization in space and backward discretization in time for the fully implicit method similarly to the hydrocarbon component residual.

A.2.1. Water Convective Flow

Convective flow term for water is a summation of fluxes from a center gridblock to six different flux directions of neighboring gridblocks that are discretized in space using central difference (Eq. A.22).

$$\nabla \cdot [a_w \nabla \Phi_w] = a_{wE}\nabla\Phi_{wE} + a_w\mathrm{w}\nabla\Phi_{wW} + a_{wN}\nabla\Phi_{wN} + a_{wS}\nabla\Phi_{wS} + a_{wB}\nabla\Phi_{wB} + a_{wT}\nabla\Phi_{wT} \tag{A.22}$$

And water interblock transmissibility can be expressed as the multiplication of geometric transmissibility (T_η) and water mobility ($\lambda_{w\eta}$) (Eq. A.23).

$$a_{w\eta} = T_\eta \lambda_{w\eta} \tag{A.23}$$

As explained in A.1.1. the interblock geometric transmissibility for the block-centered discretization is the multiplication of permeability and area divided by the distance between the gridblocks, and these can be pre-computed before the time stepping since they are considered constant in the simulation (Eq. A.24 through Eq. A.29).

$$T_E = 2\beta_c\left[\frac{(k\Delta y\Delta z)_i(k\Delta y\Delta z)_{i+1}}{(k\Delta y\Delta z)_i\Delta x_{i+1} + (k\Delta y\Delta z)_{i+1}\Delta x_i}\right] \tag{A.24}$$

$$T_W = 2\beta_c\left[\frac{(k\Delta y\Delta z)_i(k\Delta y\Delta z)_{i-1}}{(k\Delta y\Delta z)_i\Delta x_{i-1} + (k\Delta y\Delta z)_{i-1}\Delta x_i}\right] \tag{A.25}$$

$$T_N = 2\beta_c\left[\frac{(k\Delta x\Delta z)_j(k\Delta x\Delta z)_{j+1}}{(k\Delta x\Delta z)_j\Delta y_{j+1} + (k\Delta x\Delta z)_{j+1}\Delta y_j}\right] \tag{A.26}$$

$$T_S = 2\beta_c\left[\frac{(k\Delta x\Delta z)_j(k\Delta x\Delta z)_{j-1}}{(k\Delta x\Delta z)_j\Delta y_{j-1} + (k\Delta x\Delta z)_{j-1}\Delta y_j}\right] \tag{A.27}$$

$$T_B = 2\beta_c\left[\frac{(k\Delta y\Delta \mathrm{x})_k(k\Delta y\Delta x)_{k+1}}{(k\Delta y\Delta x)_k\Delta z_{k+1} + (k\Delta y\Delta x)_{k+1}\Delta z_k}\right] \tag{A.28}$$

$$T_T = 2\beta_c \left[\frac{(k\Delta y\Delta x)_k (k\Delta y\Delta x)_{k-1}}{(k\Delta y\Delta x)_k \Delta z_{k-1} + (k\Delta y\Delta x)_{k-1} \Delta z_k} \right] \tag{A.29}$$

Viscosity and density for water that are used to compute water mobility are volume-weighted arithmetic averages (Eq. A.30).

$$\lambda_{w\eta} = \left(k_{rw} \frac{\rho_w}{\mu_w} \right)_\eta \tag{A.30}$$

Water potential in block-centered gird method is expressed using oil phase pressure p_o and height difference Δz with the capillary pressure of oil and water (Eq. A.31 and Eq. A.32).

$$\Delta\Phi_{w\eta} = \left(\Phi_{w\eta} - \Phi_{wC}\right) = \Delta p_{o\eta} - \Delta p_{cow_\eta} - \gamma_{w\eta}\Delta z_\eta \tag{A.31}$$

$$\begin{aligned} \Delta\Phi_{w\eta} = \left(p_{o\eta} - p_{oC}\right) - \left(p_{cow_\eta} - p_{cow_C}\right) \\ - g_c \left(\frac{\rho_{w\eta} + \rho_{wC}}{2} \right) \left(z_\eta - z_C\right) \end{aligned} \tag{A.32}$$

The final form of water convective flow term can be rearranged and simplified using chain rule and chord-slope of capillary pressure as shown in Eq. A.33.

$$\nabla \cdot [a_w \nabla \Phi_w] = \sum_{\eta=1}^{6} \left[a_{w\eta}\left(\Delta p_{o\eta} - \gamma_{w\eta}\Delta z_\eta\right) - \frac{a_{w\eta} {p'}_{cow_\eta}}{\rho_{w\eta}} \Delta W_\eta \right] \tag{A.33}$$

Where the chord-slope of capillary pressure for oil-water and the difference of water mass per pore volume between center block and neighboring block are displayed as follows (Eq. A.34 and Eq. A.35).

$$p'_{cow_\eta} = \frac{p_{cow_\eta} - p_{cow_c}}{S_{w_\eta} - S_{w_C}} \tag{A.34}$$

$$\Delta W_\eta = W_\eta - W_C \tag{A.35}$$

A.2.2 Water Accumulation

Water accumulation term is discretized in time using backward difference in order to form a fully implicit system of equations, and its final form can be expressed using chain rule and chord-slope similarly to the water convective flow term (Eq. A.36).

$$V_b \frac{\partial}{\partial t}[\phi \mathrm{W}] = \frac{V_b}{\Delta t}[\Delta_\tau(\phi \mathrm{W})] = \frac{V_b}{\Delta t}[W^n \Delta_\tau(\phi) + \phi^{n+1}\Delta_\tau(\mathrm{W})] \tag{A.36}$$

The above equation is further discretized using the chain rule in porosity as follows (Eq. A.37).

$$\Delta_\tau(\phi) = \frac{\partial \phi}{\partial t} = \frac{\partial \phi}{\partial p}\frac{\partial p}{\partial t} = \frac{\phi^{n+1} - \phi^n}{p^{n+1} - p^n}\Delta_\tau(p) = \phi' \Delta_\tau(p) \tag{A.37}$$

And the final form is shown as Eq. A.38:

$$V_b \frac{\partial}{\partial t}[\phi \mathrm{W}] = \frac{V_b}{\Delta t}[W^n \phi'(p^{n+1} - p^n) + \phi^{n+1}(W^{n+1} - W^n)] \tag{A.38}$$

A.2.3. Water Residual

By combining convective flow, accumulation, and well source/sink terms which are discretized in space and in time, the final form of water residual can be shown as follows (Eq. A.39).

$$
\begin{aligned}
R_w = \frac{V_b}{\Delta t}\left[W^n \phi' (p_{oC}^{n+1} - p_{oC}^{n}) + \phi^{n+1}(W^{n+1} - W^n)\right] \\
- \sum_{\eta=1}^{6}\left[a_{w\eta}^{n+1}\left(\Delta p_{o\eta}^{n+1} - \gamma_{w\eta}^{n+1}\Delta z_\eta\right) - \frac{a_{w\eta}^{n+1} {p'}_{cow_\eta}^{(n+1)}}{\rho_{w\eta}^{n+1}}\Delta W_\eta^{n+1}\right] \quad \text{(A.39)} \\
+ WI_w^{n+1}\left(p_{oC}^{n+1} - p_{cow}^{n+1} - p_{wf}^{n+1}\right)
\end{aligned}
$$

APPENDIX B

The Peng-Robinson cubic equation of state (PR EOS) (Peng and Robinson, 1978) with the van der Waals mixing rule is detailed in this appendix. It is generally used for the solution of the fugacity coefficient in vapor-liquid equilibrium (VLE). There are several different types of the equations of state, however Redlich-Kwong or Peng-Robinson is more popular for the correlation of fluid properties (Young and Stephenson,1983).

The equation of the form of PR EOS is shown in Eq. B.1.

$$p = \frac{RT}{v - b} - \frac{(a\alpha)}{v^2 + 2vb - b^2} \tag{B.1}$$

Eq. B.1 can be expressed using the compressibility factor $Z = \frac{pV}{nRT}$ as shown in Eq. B.2 with Eq. B.3 through Eq. B.11.

$$Z^3 - (1 - B)Z^2 + (A - 3B^2 - 2B)Z - (AB - B^2 - B^3) = 0 \tag{B.2}$$

With,

$$A = \frac{p(a\alpha)}{(RT)^2} \tag{B.3}$$

$$B = \frac{pb}{RT} \tag{B.4}$$

$$(a\alpha) = \sum_i \sum_j x_i x_j \sqrt{a_i a_j \alpha_i \alpha_j}\left(1 - k_{ij}\right) \tag{B.5}$$

$$b = \sum_i x_i b_i \tag{B.6}$$

$$a_i = 0.45724 \frac{R^2 T_{ci}^2}{p_{ci}^2} \tag{B.7}$$

$$b_i = 0.07780 \frac{R T_{ci}}{p_{ci}} \tag{B.8}$$

$$\alpha_i = \left[1 + m_i \left(1 - \sqrt{\frac{T}{T_{ci}}}\right)\right]^2 \tag{B.9}$$

$$m_i = 0.37464 + 1.54226\omega_i - 0.2699\omega_i^2 \; for \; \omega_i < 0.49 \tag{B.10}$$

$$m_i = 0.379642 + 1.48503\omega_i - 0.164423\omega_i^2 + 0.01667\omega_i^3 \quad for \; \omega_i \geq 0.49 \tag{B.11}$$

APPENDIX C

Table C.1 Fluid EOS parameters for flash calculation.

	z_i	Molecular Weight (lbm/lbmol)	T_c(°F)	p_{crit} (psia)	Acentric factor	Parachor	Volume shift
CO2	0.0246	44.01	87.6	1070.2	0.225	0	-0.01313
C1-N2	0.3694	16.22750162	-118.6883866	664.7012182	0.008493774	0	-0.09426
C2-C3	0.0752	37.59375	152.1195479	658.6289894	0.127082447	0	-0.07819
C4	0.0193	58.1	295.984456	544.5440415	0.187803109	0	-0.01798
C5	0.0157	72.2	377.4757962	490.2070064	0.239687898	0	-0.00501
C6	0.0162	86	453.5	477.2	0.275	0	-0.02941
C7+	0.47145	320	1089	180.8	1.022	0	0.00974
Asphaltene	0.00815	800	2105	178.3	1.441	0	0.01648

Table C.2 Binary interaction coefficients (BIC) for fluid EOS characterization.

BIC	CO2	C1-N2	C2-C3	C4	C5	C6	C7+	Asphaltene
CO2	0							
C1-N2	0	0						
C2-C3	0	0	0					
C4	0	0	0	0				
C5	0	0	0	0	0			
C6	0	0	0	0	0	0		
C7+	0	0.053	0	0	0	0	0	
Asphaltene	0	0.135	0.135	0.135	0.135	0	0	0