INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2015; **103**:1–30
Published online 17 March 2015 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/nme.4874

Constraint reduction procedures for reduced-order subsurface flow models based on POD–TPWL

J. He*,† and L. J. Durlofsky

Department of Energy Resources Engineering, Stanford University, Stanford, CA 94305, USA

SUMMARY

The properties and numerical performance of reduced-order models based on trajectory piecewise linearization (TPWL) and proper orthogonal decomposition (POD) are assessed. The target application is subsurface flow modeling, although our findings should be applicable to a range of problems. The errors arising at each step in the POD–TPWL procedure are described. The impact of constraint reduction on accuracy and stability is considered in detail. Constraint reduction entails projection of the overdetermined system into a low-dimensional subspace, in which the system is solvable. Optimality conditions for constraint reduction, in terms of error minimization, are derived. Galerkin and Petrov–Galerkin projections are shown to correspond to optimality in norms that involve weighting with the Jacobian matrix. Two new treatments, inverse projection and weighted inverse projection, are suggested. These methods minimize error in appropriate norms, although they require substantial preprocessing computations. Numerical results are presented for oil reservoir simulation problems. Galerkin projection provides reasonable accuracy for simpler oil–water systems, although it becomes unstable in more challenging cases. Petrov–Galerkin projection is observed to behave stably in all cases considered. Weighted inverse projection also behaves stably, and it provides the highest accuracy. Runtime speedups of 150–400 are achieved using these POD–TPWL models. Copyright © 2015 John Wiley & Sons, Ltd.

Received 14 March 2014; Revised 25 November 2014; Accepted 19 December 2014

KEY WORDS: reduced-order model; proper orthogonal decomposition; POD; trajectory piecewise linearization; TPWL; reservoir simulation; subsurface flow; model order reduction; stability; surrogate model; proxy model

1. INTRODUCTION

The development of reduced-order modeling procedures for nonlinear problems is a topic of great interest in many application areas. The issues and approaches considered in this paper are relevant to a wide range of problems, although our focus here is on subsurface flow modeling—specifically oil reservoir simulation. Within that setting, detailed finite-volume-based flow simulators, which track the movement of multiple components in multiple phases through porous subsurface formations, are typically used to model the production of oil and gas. Important applications within this area, such as production optimization, uncertainty assessment and data assimilation, require large numbers of simulation runs. These applications, like many in other engineering fields, are extremely demanding computationally using standard full-order simulations, and they could benefit greatly from the use of fast, accurate, and robust reduced-order models (ROMs).

Essentially, the ROMs considered in this work include three key components: state reduction, nonlinearity treatment, and constraint reduction. State reduction entails the expression of full-order states (i.e., the vector of state variables in all grid blocks in the model) in terms of a small set

*Correspondence to: J. He, Chevron Energy Technology Company, 1500 Louisiana Street, Houston, TX 77002, USA.
†E-mail: nervhjc@gmail.com

of reduced variables. Nonlinearity treatment involves the approximate representation of nonlinear effects. Approaches include the construction of approximate/reduced nonlinear terms or Jacobian matrices and/or the use of some type of (piecewise) linearization procedure. Constraint reduction, which is the focus of this work, is required because, after the introduction of state reduction, there are many more equations than unknowns. It is the constraint reduction matrix that defines the low-dimensional subspace in which the residue of the original system is driven to zero. As we will see, the choice of this matrix can have a large impact on the accuracy and stability of the resulting ROM.

The use of state reduction is based on the assumption that the state vectors of the full-order system essentially lie in a lower-dimensional subspace. This is often a reasonable assumption because the states that can arise are defined through initial conditions and system dynamics, which are not infinitely variable. With this assumption, a basis for the subspace, which projects the full-order (high-fidelity) state into a low-order representation, can be constructed. In many ROM procedures, including the one considered here, the state reduction basis matrix $\mathbf{\Phi}$ is constructed through the use of proper orthogonal decomposition (POD). With this approach, a data matrix, containing as its columns 'snapshots' (solution vectors) computed during 'training' simulations, is first constructed. The left singular vectors of the singular value decomposition of the data matrix define the columns of the basis matrix $\mathbf{\Phi}$. POD-based ROMs have been used in a number of application areas [1–5], including subsurface flow simulation [6, 7]. Other approaches, such as balanced truncation [8–10] and Krylov subspace methods [11–14], have also been successfully applied.

State reduction procedures decrease the number of unknowns that must be determined at each time step in a dynamic simulation. However, for nonlinear time-variant problems, the speedups achieved through the use of state reduction alone are typically quite modest. Specifically, for reservoir simulation problems, speedup factors of at most 10 have been achieved using this approach [6, 7]. This is because some of the order-reduction computations have a computational complexity that scales with the dimension of the full-order problem. If such computations are performed at each (nonlinear) iteration at every time step, as is the case in [6, 7], the observed speedup will be limited.

Various treatments have been proposed to further accelerate ROMs for nonlinear problems. These include the discrete empirical interpolation method (DEIM) and trajectory piecewise linearization (TPWL). DEIM, first proposed by Chaturantabut and Sorensen [15, 16], reduces the dimension of nonlinear functions in the governing partial differential equation (PDE) using an empirically derived basis. During the inline (runtime) stage, reduced-order nonlinear functions are determined through computations involving only a small number of grid blocks, which greatly reduces inline computational demands. Carlberg *et al.* [17] further extended the method to treat nonlinear algebraic systems obtained from the application of Newton's method. They applied a compressive tensor approximation to enable the fast construction of reduced Jacobian matrices, which were then used in inline computations. DEIM has been applied successfully for different applications [16–18], although its implementation does require nonlinear terms to be evaluated at particular grid blocks during inline processing. This is intrusive with respect to the full-order simulator, which could pose a problem in the use of DEIM with general-purpose reservoir simulators. We note finally that a prototype DEIM procedure has been developed for reservoir flow [19], although only small two-dimensional models have thus far been considered.

Trajectory piecewise linearization, proposed by Rewienski and White [20], handles nonlinearity by constructing local (piecewise) linearizations around previously simulated (training) solutions. Because new (test) runs entail linearization around training 'points', the order-reduction computations can all be performed offline (i.e., in a preprocessing step). Thus, the inline computations involve only low-dimensional linear solutions. TPWL has been combined with POD and applied for a number of subsurface flow problems. These include oil–water models [21–23], idealized thermal simulation cases [24], compositional systems [25], and ensemble-based data assimilation [26]. Construction of the POD–TPWL model for reservoir simulation problems requires preprocessing (offline) computations equivalent to about three to four full-order simulations, although runtime speedups of 200–1000 were reported in the studies noted earlier.

As indicated earlier, constraint reduction entails the projection of the overdetermined system into a low-dimensional subspace in which the residue is driven to zero. This subspace is defined by the constraint reduction matrix $\mathbf{\Psi}$. For ROMs based on Krylov subspace or balanced truncation

DOI: 10.1002/nme

(including balanced POD [27, 28], in which POD is used to approximate the Gramians in the balanced truncation method), the appropriate constraint reduction matrix is provided from theory [29]. For POD-based methods, there is no unambiguous choice for $\mathbf{\Psi}$, and different approaches have been used.

In the initial POD–TPWL method for reservoir simulation [21], a Galerkin projection scheme [30], in which $\mathbf{\Psi} = \mathbf{\Phi}$ (recall that $\mathbf{\Phi}$ is the dimension-reduction matrix), was applied. However, as shown in [23], this approach can lead to numerical stability problems in some cases. In [23], a procedure was devised to select the columns in $\mathbf{\Phi}$ to improve system stability. This approach was shown to perform well for the oil–water cases considered, but it does not guarantee stability. Bond and Daniel [31] proposed that the constraint reduction matrix $\mathbf{\Psi}$ be designed to guarantee the stability of the reduced system through satisfaction of Lyapunov stability criteria. However, if only stability is considered, the ROM may be inaccurate. If accuracy is also taken into account, a matrix optimization problem must be solved to obtain the optimal $\mathbf{\Psi}$, and this is very expensive for large systems. A Petrov–Galerkin projection scheme was recently used for POD-based DEIM by Carlberg *et al.* [17]. This approach provided numerical stability at reasonable computational cost (although stability is still not guaranteed). A Petrov–Galerkin procedure was also used in [32] for linear model reduction, and recently in [25] for POD–TPWL compositional reservoir simulation. This approach has not, however, been studied systematically within the context of POD–TPWL.

In this work, we assess the accuracy and stability of various constraint reduction treatments for POD–TPWL models. The approaches presented should be relevant for POD–TPWL procedures in a range of application areas, although our implementation and numerical results are for subsurface flow problems. Following a brief assessment of the POD–TPWL errors that arise from state reduction and linearization, we discuss the characteristics of several constraint reduction procedures. Optimality conditions for these approaches, which are based on error minimization, are presented. The methods considered include Galerkin projection, Petrov–Galerkin projection, and two new methods, inverse projection (IP) and weighted IP (WIP). We also provide linear stability criteria for POD–TPWL models. The numerical accuracy and stability of the different constraint reduction procedures are compared for oil–water and oil–gas compositional flow examples. Our results demonstrate the relative advantages of the different approaches and suggest directions for future research.

This paper proceeds as follows. In Section 2, we briefly discuss the reservoir simulation problems targeted in this work. In Section 3, the POD–TPWL model is derived, and the error incurred at each step is discussed. Optimal constraint reduction procedures are derived in Section 4, and stability requirements are discussed in Section 5. In Section 6, the performance of Galerkin projection and Petrov–Galerkin projection are investigated in detail for three test cases. The two new constraint reduction methods, IP and WIP, are developed in Section 7. Numerical results for these approaches are also presented. A summary of our findings and suggestions for future work are provided in Section 8.

2. PROBLEM DESCRIPTION

Our specific interest here is in the simulation of oil–water and oil–gas compositional systems. Oil–water systems are commonly used to model oil production driven by the injection of water (referred to as waterflooding), while compositional systems are used to model enhanced oil recovery processes, which often involve the injection of gas, as well as CO_2 storage operations. Our descriptions here are brief; for more details, see, e.g., [21, 29, 33–37].

The governing equations for oil–water systems consist of statements of mass conservation for oil and water, combined with Darcy's law, which relates the flow of each fluid phase to the pressure gradient. These equations include accumulation, flux, and source/sink terms and can be written as

$$\frac{\partial}{\partial t}\left(\phi \rho_j S_j\right) - \nabla \cdot \left[\rho_j \lambda_j \mathbf{k}\left(\nabla p_j - \rho_j g \nabla D\right)\right] + q_j^w = 0, \quad j = o,\ w, \tag{1}$$

DOI: 10.1002/nme

where the subscript j designates the fluid phase (o indicates oil and w water). Here, t is time, ϕ is porosity (volume fraction of the pore space), ρ_j is the phase density, S_j is the phase saturation (volume fraction of phase j within the pore space) , λ_j is the phase mobility (which is typically a nonlinear function of S_j), $\mathbf{k}$ is the absolute permeability tensor ($\mathbf{k}$ is essentially a flow conductivity and is a property of the rock), p_j is the phase pressure, g is the gravitational acceleration, D is the depth, and q_j^w is the source/sink term (the superscript w indicates that this term is driven by wells). The system is closed by adding the saturation constraint ($S_o + S_w = 1$) and the capillary pressure relationship ($p_c(S_w) = p_o - p_w$). The primary unknowns in Equation (1) are often taken to be the oil phase pressure p_o and water phase saturation S_w. Other quantities (p_w and S_o) can be easily computed block by block once p_o and S_w are determined. See [21] for more details on the oil–water problem formulation in the context of POD–TPWL. Note that equations of the form of Equation (1) arise in many problems involving flow and transport.

For compositional systems, we track a total of n_c components (as opposed to two components in oil–water systems). The governing equations for oil–gas compositional systems resemble Equation (1), in that they entail statements of mass conservation for each component and Darcy's law for each phase, although they are complicated by the fact that components partition between the oil and gas phases. Phase equilibrium equations for each component are therefore additionally required to determine the fraction of each component in each phase. For isothermal compositional systems with n_c components in two phases, there are a total of $2n_c + 4$ unknown variables in each grid block [25]. Practical systems are typically modeled with ∼4–10 components, so computational demands for large models can be substantial.

In compositional models, there are, however, only n_c primary equations and n_c primary unknowns for each grid block. This set of equations must be solved as a fully coupled system. The remaining $n_c + 4$ unknowns decouple and can be solved block by block. The n_c primary equations are typically the mass conservation equations [34, 35, 37]. Various choices for the n_c primary variables are possible. Most common is the so-called natural formulation, in which the primary variables consist of the oil-phase pressure p_o and $n_c - 1$ phase-dependent variables (such as the mole fraction of component c in the gas phase). The natural formulation, however, requires variable switching when a phase disappears. This introduces complications in the context of reduced-order modeling because it is much more straightforward for the ROM procedure to treat the same types of variables in the test and training simulations. We thus apply the less commonly used molar formulation [37], in which the primary variables are p_o and $n_c - 1$ overall mole fractions. These quantities are well defined at all times in all grid blocks, so variable switching is not required.

The governing equations and detailed discretizations differ for oil–water and compositional problems. Nonetheless, the fully implicit discretized representations, for a wide range of problems including these, can be written as a general set of nonlinear algebraic equations in the following form:

$$\mathbf{g}^{n+1} = \mathbf{g}\left(\mathbf{x}^{n+1}, \mathbf{x}^n, \mathbf{u}^{n+1}\right) = \mathbf{0}. \tag{2}$$

Here, $\mathbf{g}$ is the residual vector we seek to drive to zero, n and $n + 1$ denote time level, $\mathbf{u}$ is the set of control parameters, and $\mathbf{x}$ designates the state vector (primary variables in each grid block). We denote the number of grid blocks as n_b and the dimension of the state vector $\mathbf{x}$ (i.e., the total number of primary variables) as n_v. In oil–water systems, $\mathbf{x}$ contains oil pressure p (from here on, we use p in place of p_o) and water saturation S_w in each grid block, so $n_v = 2n_b$. In compositional systems, $\mathbf{x}$ contains pressure p and the overall mole fraction, designated z_c, for $n_c - 1$ components. In this case, $n_v = n_c n_b$. In Equation (2), $\mathbf{u}$ is the set of specified control parameters that drive the oil recovery process. These are taken here to be the pressures of injection or production wells (referred to as bottom-hole pressures or BHPs), although they could also be injection or production flow rates. In either case, these terms enter through the source terms in the governing equations (e.g., q_j^w in Equation (1)) and thus strongly impact the numerical solutions. In all cases, $\mathbf{x}^n$ is known from the previous time step or the initial condition, and the goal is to compute $\mathbf{x}^{n+1}$.

DOI: 10.1002/nme

In the full-order simulation, Equation (2) is solved using Newton's method. This entails, at each iteration, the solution of the high-dimensional linear system

$$\mathbf{J}\boldsymbol{\delta} = -\mathbf{g}, \tag{3}$$

where $\mathbf{J}$ is the Jacobian matrix, given by $\mathbf{J} = \frac{\partial \mathbf{g}}{\partial \mathbf{x}}$ evaluated at the current estimate of $\mathbf{x}^{n+1}$, and $\boldsymbol{\delta} = \mathbf{x}^{n+1,\nu+1} - \mathbf{x}^{n+1,\nu}$ is the update vector, where ν designates iteration. Convergence is achieved once an appropriate norm of $\mathbf{g}^{n+1}$ is less than a specified tolerance, and the solution states are then designated $\mathbf{x}^{n+1}$.

For practical reservoir simulation problems, $n_v \sim O(10^4 - 10^6)$. In addition, the high degree of nonlinearity of Equation (2) can result in substantial numbers of Newton iterations, small time steps, and frequent time-step cuts. The combination of nonlinearity and high dimensionality leads to very large computational demands, especially when thousands or tens of thousands of simulations must be performed, as may be the case for production optimization computations. The POD–TPWL approach we now describe can provide a much more efficient (although approximate) solution of Equation (2).

3. POD–TPWL MODEL AND ASSESSMENT OF ERROR

In this section, we will consider general POD–TPWL models, applicable to a wide range of systems. We will assess the errors incurred at each step of the POD–TPWL procedure. These include linearization error, state reduction error, constraint reduction error, and error propagated from the previous time step (which we relate to stability). Most aspects of this discussion are quite general, although some are specific to the particular models and fluid systems under consideration. Detailed derivations of POD–TPWL models for oil–water [21, 23] and compositional systems [25] have been presented previously and should be consulted for more details.

Error in ROMs has been analyzed by a number of investigators. Rathinam and Petzold [38], for example, presented an error analysis for POD-based reduced-order ordinary differential equation (ODE) systems. Chaturantabut and Sorensen [39] provided a state-space error estimate for nonlinear model reduction based on POD–DEIM. Our discussion here will be focused on POD–TPWL.

In the following discussion, the POD–TPWL equations are derived from the original system of equations (Equation (2)) by introducing a series of approximations. Solutions associated with different levels of approximation will be denoted $\mathbf{x}_\alpha^{n+1}$ ($\alpha = 1, 2, 3, 4$), with increasing α indicating a more approximate solution. Consistent with this, we denote the exact solution to the full set of nonlinear algebraic equations (Equation (2)) as $\mathbf{x}_0^{n+1}$. The solution that contains linearization error is designated $\mathbf{x}_1^{n+1}$; the solution that contains linearization error and state reduction error is denoted $\mathbf{x}_2^{n+1}$; the solution that contains linearization, state reduction, and constraint reduction error is designated $\mathbf{x}_3^{n+1}$; and the solution that contains all of these errors plus error propagated from the previous time step is denoted $\mathbf{x}_4^{n+1}$.

3.1. Trajectory piecewise linearization

In order to construct the POD–TPWL model, we must first perform one or more full-order 'training' simulations for some specific control parameters, which we designate $\mathbf{u}^{i+1}$. This corresponds to generating solutions to the following equation:

$$\mathbf{g}^{i+1} = \mathbf{g}(\mathbf{x}^{i+1}, \mathbf{x}^i, \mathbf{u}^{i+1}) = \mathbf{0}, \quad i = 0, \ldots, n_t - 1, \tag{4}$$

where n_t is the number of time steps. Note that we use superscripts i and $i+1$ to indicate sequential 'points' (in time) in a training simulation; i.e., $\mathbf{x}^i$ and $\mathbf{x}^{i+1}$ are the solutions of the training simulation at time steps i and $i+1$.

In order to construct the solution for a new set of controls (designated $\mathbf{u}^{n+1}$), rather than solve Equation (2) iteratively using Newton's method, we instead represent the new residual vector $\mathbf{g}^{n+1}$

DOI: 10.1002/nme

in terms of a Taylor series expansion around the training solution. We refer to the new simulation as a 'test' simulation. Neglecting higher-order terms, we write

$$\mathbf{g}^{n+1} = \mathbf{0} \approx \mathbf{g}^{i+1} + \frac{\partial \mathbf{g}^{i+1}}{\partial \mathbf{x}^{i+1}}\left(\mathbf{x}^{n+1} - \mathbf{x}^{i+1}\right) + \frac{\partial \mathbf{g}^{i+1}}{\partial \mathbf{x}^{i}}\left(\mathbf{x}^{n} - \mathbf{x}^{i}\right) + \frac{\partial \mathbf{g}^{i+1}}{\partial \mathbf{u}^{i+1}}\left(\mathbf{u}^{n+1} - \mathbf{u}^{i+1}\right). \quad (5)$$

Here, $\mathbf{x}^n$ indicates the solution at the previous time level in the test simulation, $\mathbf{x}^{n+1}$ indicates the test solution that we wish to compute, and $\mathbf{x}^i$ and $\mathbf{x}^{i+1}$ are sequential solutions in the training simulation.

From Equation (4), we know that $\mathbf{g}^{i+1} = \mathbf{0}$. This allows us to express Equation (5), after some rearrangement, as follows:

$$\mathbf{J}^{i+1}\mathbf{x}^{n+1} = \mathbf{J}^{i+1}\mathbf{x}^{i+1} - \left[\mathbf{A}^{i+1}\left(\mathbf{x}_0^n - \mathbf{x}^i\right) + \mathbf{B}^{i+1}\left(\mathbf{u}^{n+1} - \mathbf{u}^{i+1}\right)\right], \quad (6)$$

where

$$\mathbf{J}^{i+1} = \frac{\partial \mathbf{g}^{i+1}}{\partial \mathbf{x}^{i+1}}, \quad \mathbf{A}^{i+1} = \frac{\partial \mathbf{g}^{i+1}}{\partial \mathbf{x}^{i}}, \quad \mathbf{B}^{i+1} = \frac{\partial \mathbf{g}^{i+1}}{\partial \mathbf{u}^{i+1}}. \quad (7)$$

Here, $\mathbf{J}^{i+1} \in \mathbb{R}^{n_v \times n_v}$ is the Jacobian matrix at time step $i+1$ (evaluated upon convergence) of the training simulation, $\mathbf{A}^{i+1} \in \mathbb{R}^{n_v \times n_v}$, and $\mathbf{B}^{i+1} \in \mathbb{R}^{n_v \times n_u}$, where n_u is the dimension of the control vector $\mathbf{u}$ (typically, there are significantly fewer wells than grid blocks, so $n_u << n_v$). Note that $\mathbf{x}_0^n$, the exact solution of Equation (2) at the previous time step, appears in Equation (6). This is because the error we are now considering corresponds to the error incurred in a single time step. Later, in Section 3.4, we will incorporate the error propagated from the previous time step into our analysis.

We denote the solution to Equation (6) as $\mathbf{x}_1^{n+1}$. This equation approximates the high-dimensional nonlinear system in Equation (2) as a high-dimensional linear system. The (linearization) error incurred in this step is referred to as $\mathbf{e}_{01}^{n+1}$, where $\mathbf{e}_{01}^{n+1} = \mathbf{x}_0^{n+1} - \mathbf{x}_1^{n+1}$.

3.2. Proper orthogonal decomposition

Equation (6) is linear, but it is still expressed in the high-dimensional space. To reduce the number of unknowns, we now apply POD. This enables us to write $\mathbf{x} = \boldsymbol{\Phi}\boldsymbol{\xi}$, where $\boldsymbol{\Phi} \in \mathbb{R}^{n_v \times l}$ is the basis matrix and $\boldsymbol{\xi} \in \mathbb{R}^{l \times 1}$ is the reduced state vector. Because $l << n_v$, the system state can be expressed in terms of a relatively small number of variables.

Proper orthogonal decomposition has been used in the context of reduced-order modeling by a number of researchers (e.g., [1–7]). In POD, the columns of the basis matrix $\boldsymbol{\Phi}$ are the leading singular vectors of the snapshot matrices $\mathbf{X}$. Snapshot matrices contain, as their columns, the solution vectors computed during one or more training simulations. In this work, we typically use two or three training runs to provide a sufficient number of snapshots for the POD basis construction.

As discussed in [21] and [25], we apply POD separately to pressure snapshots and water saturation snapshots (in oil–water problems) or to pressure snapshots and overall mole fraction snapshots (in compositional problems). For oil–water systems, we have

$$\mathbf{x} = \begin{bmatrix} \mathbf{p} \\ \mathbf{S}_w \end{bmatrix} \approx \boldsymbol{\Phi}\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\Phi}_p & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Phi}_S \end{bmatrix} \begin{bmatrix} \boldsymbol{\xi}_p \\ \boldsymbol{\xi}_S \end{bmatrix}, \quad (8)$$

where $\boldsymbol{\Phi}_p \in \mathbb{R}^{n_b \times l_p}$ and $\boldsymbol{\Phi}_S \in \mathbb{R}^{n_b \times l_S}$ are the basis matrices for pressure and water saturation, respectively; $\boldsymbol{\xi}_p \in \mathbb{R}^{l_p \times 1}$ and $\boldsymbol{\xi}_S \in \mathbb{R}^{l_S \times 1}$ are the reduced state vectors for pressure and water saturation, respectively; and $\boldsymbol{\Phi} \in \mathbb{R}^{n_v \times l}$, where $l = l_p + l_S$, is the basis matrix for the entire state vector $\boldsymbol{\xi}$. Note that, in general, the number of columns in $\boldsymbol{\Phi}_p$ differs from that in $\boldsymbol{\Phi}_S$ (i.e., $l_p \neq l_S$).

Similarly, for compositional systems, we have

$$\mathbf{x} = \begin{bmatrix} \mathbf{p} \\ \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_{n_c-1} \end{bmatrix} \approx \boldsymbol{\Phi}\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\Phi}_p & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Phi}_z \end{bmatrix} \begin{bmatrix} \boldsymbol{\xi}_p \\ \boldsymbol{\xi}_z \end{bmatrix}, \quad (9)$$

DOI: 10.1002/nme

where $\mathbf{\Phi}_z \in \mathbb{R}^{n_b(n_c-1)\times l_z}$ is the basis matrix for the overall mole fraction variables, $\boldsymbol{\xi}_z \in \mathbb{R}^{l_z\times 1}$ is the corresponding reduced state vector, and $\mathbf{\Phi} \in \mathbb{R}^{n_v\times l}$, where $l = l_p + l_z$, is the basis matrix for the entire state vector $\boldsymbol{\xi}$. Procedures for specifying l_p and l_S, or l_p and l_z, which can be based on 'energy' criteria or stability considerations (the latter are discussed in Section 5 and illustrated in Section 6.4), are described in [21, 23, 25]. We note finally that all columns of the submatrices in $\mathbf{\Phi}$, as well as the columns of the overall $\mathbf{\Phi}$ matrix, are orthonormal (meaning $\mathbf{\Phi}^T\mathbf{\Phi} = \mathbf{I}$).

We now introduce the reduced representation $\mathbf{x} = \mathbf{\Phi}\boldsymbol{\xi}$ to the right-hand side of Equation (6), which gives

$$\mathbf{J}^{i+1}\mathbf{x}^{n+1} = \mathbf{J}^{i+1}\mathbf{\Phi}\boldsymbol{\xi}^{i+1} - \left[\mathbf{A}^{i+1}\mathbf{\Phi}\left(\boldsymbol{\xi}_0^n - \boldsymbol{\xi}^i\right) + \mathbf{B}^{i+1}\left(\mathbf{u}^{n+1} - \mathbf{u}^{i+1}\right)\right]. \tag{10}$$

The solution to Equation (10) is denoted $\mathbf{x}_2^{n+1}$. The additional error incurred in this step is due to state reduction and can be expressed as $\mathbf{e}_{12}^{n+1} = \mathbf{x}_1^{n+1} - \mathbf{x}_2^{n+1}$. Note that $\boldsymbol{\xi}_0^n$ is the reduced representation of $\mathbf{x}_0^n$, the true solution at time step n; i.e., $\boldsymbol{\xi}_0^n = \mathbf{\Phi}^T\mathbf{x}_0^n$ (note that the subscript convention for $\boldsymbol{\xi}$ corresponds to that used for $\mathbf{x}$). Therefore, $\boldsymbol{\xi}_0^n$ is not at the same level of approximation as $\mathbf{x}^{n+1}$ ($= \mathbf{x}_2^{n+1}$) in Equation (10). This is the case because $\boldsymbol{\xi}_0^n$ is the projection of the true solution of Equation (2) at time step n, while $\mathbf{x}_2^{n+1}$ includes both linearization error and state reduction error. For simplicity, we denote the right-hand side of Equation (10) as $\mathbf{b}^{n+1}$ (this notation will be used in the subsequent analysis). Then Equation (10) becomes simply $\mathbf{J}^{i+1}\mathbf{x}^{n+1} = \mathbf{b}^{n+1}$.

3.3. Constraint reduction

Applying the POD representation to the left-hand side of Equation (10) results in an overdetermined system, which has n_v equations but only l unknowns. This approximation introduces a residual term $\mathbf{r}$, which appears as follows:

$$\mathbf{J}^{i+1}\mathbf{\Phi}\boldsymbol{\xi}^{n+1} = \mathbf{J}^{i+1}\mathbf{\Phi}\boldsymbol{\xi}^{i+1} - \left[\mathbf{A}^{i+1}\mathbf{\Phi}\left(\boldsymbol{\xi}_0^n - \boldsymbol{\xi}^i\right) + \mathbf{B}^{i+1}\left(\mathbf{u}^{n+1} - \mathbf{u}^{i+1}\right)\right] + \mathbf{r}. \tag{11}$$

In general, there is no solution for $\boldsymbol{\xi}^{n+1}$ that can render the residual term to be identically zero. Therefore, Equation (11) is usually solved by requiring $\mathbf{r}$ to be zero in an l-dimensional subspace whose basis matrix is denoted $\mathbf{\Psi}^{i+1}$ (i.e., $(\mathbf{\Psi}^{i+1})^T\mathbf{r} = \mathbf{0}$). The matrix $\mathbf{\Psi}^{i+1} \in \mathbb{R}^{n_v\times l}$ is called the constraint reduction matrix, also referred to as the left projection matrix or the test function. Premultiplying Equation (11) by $(\mathbf{\Psi}^{i+1})^T$, with $(\mathbf{\Psi}^{i+1})^T\mathbf{r} = \mathbf{0}$, yields, after some rearrangement,

$$\boldsymbol{\xi}^{n+1} = \boldsymbol{\xi}^{i+1} - \left(\mathbf{J}_r^{i+1}\right)^{-1}\left[\mathbf{A}_r^{i+1}\left(\boldsymbol{\xi}_0^n - \boldsymbol{\xi}^i\right) + \mathbf{B}_r^{i+1}\left(\mathbf{u}^{n+1} - \mathbf{u}^{i+1}\right)\right], \tag{12}$$

where the reduced derivative matrices are defined as

$$\mathbf{J}_r^{i+1} = \left(\mathbf{\Psi}^{i+1}\right)^T\mathbf{J}^{i+1}\mathbf{\Phi}, \quad \mathbf{A}_r^{i+1} = \left(\mathbf{\Psi}^{i+1}\right)^T\mathbf{A}^{i+1}\mathbf{\Phi}, \quad \mathbf{B}_r^{i+1} = \left(\mathbf{\Psi}^{i+1}\right)^T\mathbf{B}^{i+1}. \tag{13}$$

Here, $\mathbf{J}_r^{i+1} \in \mathbb{R}^{l\times l}$ is the reduced Jacobian matrix, $\mathbf{A}_r^{i+1} \in \mathbb{R}^{l\times l}$, and $\mathbf{B}_r^{i+1} \in \mathbb{R}^{l\times n_u}$.

The choice of $\mathbf{\Psi}^{i+1}$ is not unique. In previous work involving POD–TPWL models for oil–water systems, Galerkin projection was applied [21, 23, 26], meaning we take $\mathbf{\Psi}^{i+1} = \mathbf{\Phi}$. As discussed earlier, this can lead to unstable POD–TPWL models [23]. Recent work has demonstrated that Petrov–Galerkin projection, where $\mathbf{\Psi}^{i+1} = \mathbf{J}^{i+1}\mathbf{\Phi}$, represents a viable (and more numerically stable) alternative [17, 25]. In this work, we will investigate the accuracy and stability of these, and other, constraint reduction methods.

Equation (12) defines the POD–TPWL model given the true reduced solution ($\boldsymbol{\xi}_0^n$) at the previous time step. This equation can also be written as

$$\boldsymbol{\xi}^{n+1} = \left(\mathbf{J}_r^{i+1}\right)^{-1}\left(\mathbf{\Psi}^{i+1}\right)^T\mathbf{b}^{n+1}, \tag{14}$$

with $\mathbf{b}^{n+1}$ as defined earlier. We refer to $\boldsymbol{\xi}^{n+1}$ in Equations (12) and (14) as $\boldsymbol{\xi}_3^{n+1}$. The full-order solution at this level of approximation, denoted $\mathbf{x}_3^{n+1}$, can be reconstructed as $\mathbf{x}_3^{n+1} = \mathbf{\Phi}\boldsymbol{\xi}_3^{n+1}$. The error incurred in this step is defined as $\mathbf{e}_{23}^{n+1} = \mathbf{x}_2^{n+1} - \mathbf{x}_3^{n+1}$ and is referred to as the constraint reduction error.

DOI: 10.1002/nme

3.4. Error propagation

The term $\boldsymbol{\xi}_0^n$ in Equation (12) is the projection of the true solution $\mathbf{x}_0^n$, which is generally not available. In an actual POD–TPWL computation, $\boldsymbol{\xi}_0^n$ is replaced by $\boldsymbol{\xi}^n$, which corresponds to the POD–TPWL solution at time step n. We thus write

$$\boldsymbol{\xi}^{n+1} = \boldsymbol{\xi}^{i+1} - \left(\mathbf{J}_r^{i+1}\right)^{-1}\left[\mathbf{A}_r^{i+1}\left(\boldsymbol{\xi}^n - \boldsymbol{\xi}^i\right) + \mathbf{B}_r^{i+1}\left(\mathbf{u}^{n+1} - \mathbf{u}^{i+1}\right)\right]. \tag{15}$$

Equation (15) defines the POD–TPWL model we actually solve. By combining linearization and order reduction, the original high-order nonlinear system of equations has been transformed to a linear expression in l variables (l is typically ∼100–1000), which can be evaluated in seconds [21, 26].

To apply Equation (15), we must first determine the saved state and control point $(\boldsymbol{\xi}^{i+1}, \boldsymbol{\xi}^i, \mathbf{u}^{i+1})$ around which to linearize. This is usually accomplished by minimizing a measure of distance, denoted $d^{n,j}$, between the test solution $\boldsymbol{\xi}$ at time step n and any saved point $\boldsymbol{\xi}^j$ in the training simulation, that is, $i = \arg\min_j (d^{n,j})$. Distance definitions are typically problem specific. Here, we use the definition given in [25], which entails a weighted combination of the relative difference in dimensionless time (pore volume injected in this context) and in the reduced water saturation or total mole fraction states ($\boldsymbol{\xi}_S$ or $\boldsymbol{\xi}_z$). See [25] for further details.

In Equation (15), $\boldsymbol{\xi}^n$ corresponds to the POD–TPWL solution at the previous time step, which we designate $\boldsymbol{\xi}_4^n$. The solution to Equation (15) is thus designated $\boldsymbol{\xi}_4^{n+1}$, and its corresponding full-order state is $\mathbf{x}_4^{n+1}$. The difference between $\boldsymbol{\xi}_0^n$ and $\boldsymbol{\xi}_4^n$ can be expressed as

$$\boldsymbol{\xi}_0^n - \boldsymbol{\xi}_4^n = \boldsymbol{\Phi}^T(\mathbf{x}_0^n - \mathbf{x}_4^n) = \boldsymbol{\Phi}^T\mathbf{e}_{04}^n. \tag{16}$$

The fact that $\mathbf{e}_{04}^n$ is nonzero results in a difference between the reduced solutions $\boldsymbol{\xi}_3^{n+1}$ and $\boldsymbol{\xi}_4^{n+1}$ and thus a difference between the full-order solutions $\mathbf{x}_3^{n+1}$ and $\mathbf{x}_4^{n+1}$. The error incurred at this step is defined as $\mathbf{e}_{34}^{n+1} = \mathbf{x}_3^{n+1} - \mathbf{x}_4^{n+1}$, which can be viewed as the error inherited from the previous time step. It is given by

$$\begin{aligned}\mathbf{e}_{34}^{n+1} &= \mathbf{x}_3^{n+1} - \mathbf{x}_4^{n+1}\\ &= \boldsymbol{\Phi}\left(\boldsymbol{\xi}_3^{n+1} - \boldsymbol{\xi}_4^{n+1}\right)\\ &= -\boldsymbol{\Phi}\left(\mathbf{J}_r^{i+1}\right)^{-1}\mathbf{A}_r^{i+1}\left(\boldsymbol{\xi}_0^n - \boldsymbol{\xi}_4^n\right)\\ &= \mathbf{M}^{i+1}\mathbf{e}_{04}^n,\end{aligned} \tag{17}$$

where $\mathbf{M}^{i+1} = -\boldsymbol{\Phi}(\mathbf{J}_r^{i+1})^{-1}\mathbf{A}_r^{i+1}\boldsymbol{\Phi}^T$. The third step in Equation (17) corresponds to subtracting Equation (15) (for $\boldsymbol{\xi}_4^{n+1}$) from Equation (12) (for $\boldsymbol{\xi}_3^{n+1}$).

3.5. Total error of the POD–TPWL model

The total error $\mathbf{e}_{04}^{n+1}$ of the POD–TPWL solution $\mathbf{x}_4^{n+1}$ (reconstructed from the solution of Equation (15), $\boldsymbol{\xi}_4^{n+1}$) relative to the true solution of Equation (2), $\mathbf{x}_0^{n+1}$, can be expressed as

$$\begin{aligned}\mathbf{e}_{04}^{n+1} &= \mathbf{x}_0^{n+1} - \mathbf{x}_4^{n+1}\\ &= (\mathbf{x}_0^{n+1} - \mathbf{x}_1^{n+1}) + (\mathbf{x}_1^{n+1} - \mathbf{x}_2^{n+1}) + (\mathbf{x}_2^{n+1} - \mathbf{x}_3^{n+1}) + (\mathbf{x}_3^{n+1} - \mathbf{x}_4^{n+1})\\ &= \mathbf{e}_{01}^{n+1} + \mathbf{e}_{12}^{n+1} + \mathbf{e}_{23}^{n+1} + \mathbf{e}_{34}^{n+1}\\ &= \mathbf{e}_{01}^{n+1} + \mathbf{e}_{12}^{n+1} + \mathbf{e}_{23}^{n+1} + \mathbf{M}^{i+1}\mathbf{e}_{04}^n.\end{aligned} \tag{18}$$

In other words, the total error $\mathbf{e}_{04}^{n+1}$ is the sum of the linearization error $\mathbf{e}_{01}^{n+1}$, the state reduction error $\mathbf{e}_{12}^{n+1}$, the constraint reduction error $\mathbf{e}_{23}^{n+1}$, and the error propagated from the previous time step $\mathbf{M}^{i+1}\mathbf{e}_{04}^n$. The norm of the total error is bounded by the sum of the norms of these four error components; that is,

$$\left\|\mathbf{e}_{04}^{n+1}\right\| \leqslant \left\|\mathbf{e}_{01}^{n+1}\right\| + \left\|\mathbf{e}_{12}^{n+1}\right\| + \left\|\mathbf{e}_{23}^{n+1}\right\| + \left\|\mathbf{M}^{i+1}\mathbf{e}_{04}^n\right\|. \tag{19}$$

DOI: 10.1002/nme

The linearization error $\mathbf{e}_{01}^{n+1}$ is related to the nonlinearity of the problem and the distance between the current solution and the point in the training run used for linearization. This error component can be reduced by using additional training simulations for linearization (or occasional retraining), by using a better point selection scheme, or by modeling higher-order terms [40]. The second-order terms (SOTs) appearing in the Taylor-series expansion of Equation (2) involve sparse third-order tensors multiplied by two vectors of differences. For example, the SOT with respect to $(\mathbf{x}^{i+1}, \mathbf{u}^{i+1})$ can be written as

$$\text{SOT} = \left(\mathbf{x}^{n+1} - \mathbf{x}^{i+1}\right)^T \frac{\partial^2 \mathbf{g}^{i+1}}{\partial \mathbf{x}^{i+1} \partial \mathbf{u}^{i+1}} \left(\mathbf{u}^{n+1} - \mathbf{u}^{i+1}\right). \tag{20}$$

These second-order derivatives are not usually available in subsurface flow simulators, although it should be possible to construct them using simulators based on automatic differentiation, such as Stanford's Automatic Differentiation-based General Purpose Research Simulator (AD-GPRS) [41]. In addition, for specific applications, some of the SOTs vanish. For example, for reservoir simulation problems, the cross terms involving second derivatives with respect to $(\mathbf{x}^i, \mathbf{x}^{i+1})$ and $(\mathbf{x}^i, \mathbf{u}^{i+1})$ are usually zero. Furthermore, for models with standard well treatments in which well pressures (BHPs) are the control parameters, the second-order derivative with respect to $\mathbf{u}^{i+1}$ is zero. Thus, only some of the SOTs would need to be considered.

The state reduction error $\mathbf{e}_{12}^{n+1}$ results from applying order reduction to the states. Equations (10) and (6) are linear systems with the same coefficient matrix but different right-hand sides. Therefore, $\|\mathbf{e}_{12}^{n+1}\| \leqslant \kappa \|\mathbf{x}_2^{n+1}\| \|\Delta\mathbf{b}\| / \|\mathbf{b}^{n+1}\|$, where κ is the condition number of $\mathbf{J}^{i+1}$, $\mathbf{b}^{n+1}$ is the right-hand side of Equation (10), and $\Delta\mathbf{b}$ is the difference between the right-hand sides of Equations (10) and (6) [42]. The magnitude of $\|\Delta\mathbf{b}\|$ is determined by the state reduction basis matrix $\mathbf{\Phi}$. Therefore, within the framework of POD, the state reduction error $\mathbf{e}_{12}^{n+1}$ can be decreased by improving the quality of $\mathbf{\Phi}$. This can be accomplished, for example, by collecting more snapshots from additional training simulations. Another approach, proposed in [23], is to keep variables in important grid blocks (e.g., well blocks) in the full-order space, meaning these variables are not represented via $\mathbf{x} = \mathbf{\Phi}\boldsymbol{\xi}$. This treatment was shown to provide more accurate results in many cases.

The error terms $\mathbf{e}_{01}^{n+1}$ and $\mathbf{e}_{12}^{n+1}$ are incurred prior to the introduction of constraint reduction in Equation (12), so they do not depend on the matrix $\mathbf{\Psi}^{i+1}$. The error $\mathbf{e}_{23}^{n+1}$ is however incurred by the state and constraint reduction at time step $n+1$, so it does depend on $\mathbf{\Psi}^{i+1}$. The error $\mathbf{e}_{34}^{n+1} = \mathbf{M}^{i+1}\mathbf{e}_{04}^{n}$ is the error propagated from the previous time step. It also depends on $\mathbf{\Psi}^{i+1}$ via the matrix $\mathbf{M}^{i+1}$. An optimal choice of $\mathbf{\Psi}^{i+1}$ should minimize the sum of $\mathbf{e}_{23}^{n+1}$ and $\mathbf{e}_{34}^{n+1}$. The optimal $\mathbf{\Psi}^{i+1}$ will however depend on the error at the previous time step ($\mathbf{e}_{04}^{n}$), which is, in general, not available. Therefore, our strategy here is to choose a constraint reduction matrix $\mathbf{\Psi}^{i+1}$ that minimizes the one-step error term $\mathbf{e}_{23}^{n+1}$, while also ensuring that the resulting POD–TPWL model behaves stably, which means the error from previous time steps will not grow unphysically with time.

4. OPTIMAL CONSTRAINT REDUCTION PROCEDURES

We now derive optimal constraint reduction matrices that minimize the one-step error $\mathbf{e}_{23}^{n+1}$ in different norms. In our development here, because we only consider one time step, we drop the superscripts $i+1$ and $n+1$.

4.1. General development

The optimality condition can be written as

$$\mathbf{\Psi}^* = \arg\min_{\mathbf{\Psi}} \|\mathbf{e}_{23}\|_{\mathbf{\Theta}}^2 = \arg\min_{\mathbf{\Psi}} \|\mathbf{x}_2 - \mathbf{x}_3\|_{\mathbf{\Theta}}^2, \tag{21}$$

where $\mathbf{\Psi}^*$ designates the optimum (we drop the superscript $*$ in subsequent equations). Here, $\|\cdot\|_{\mathbf{\Theta}}$ is a norm defined as $\|\mathbf{e}\|_{\mathbf{\Theta}} = \sqrt{\mathbf{e}^T\mathbf{\Theta}\mathbf{e}}$, with $\mathbf{e} \in \mathbb{R}^{n_v \times 1}$ and $\mathbf{\Theta} \in \mathbb{R}^{n_v \times n_v}$, where $\mathbf{\Theta}$ is a symmetric

DOI: 10.1002/nme

positive definite (SPD) matrix and n_v is the dimension of the full-order state $\mathbf{x}$. The requirement for $\boldsymbol{\Theta}$ to be SPD ensures that the minimum of $\|\mathbf{e}\|^2_{\boldsymbol{\Theta}}$ is uniquely $\mathbf{e} = \mathbf{0}$.

We denote the objective function of the minimization problem in Equation (21) as $f(\boldsymbol{\Psi})$; that is, $f(\boldsymbol{\Psi}) = \|\mathbf{e}_{23}\|^2_{\boldsymbol{\Theta}} = \|\mathbf{x}_2 - \mathbf{x}_3\|^2_{\boldsymbol{\Theta}}$. From Equation (10), we have $\mathbf{x}_2 = \mathbf{J}^{-1}\mathbf{b}$, while $\mathbf{x}_3$ can be reconstructed from the solution of Equation (12) as $\mathbf{x}_3 = \boldsymbol{\Phi}\boldsymbol{\xi}_3$ (which we denote simply as $\boldsymbol{\Phi}\boldsymbol{\xi}$ in this section). Therefore, $f(\boldsymbol{\Psi})$ can be expressed as

$$\begin{aligned} f(\boldsymbol{\Psi}) &= \|\mathbf{x}_2 - \mathbf{x}_3\|^2_{\boldsymbol{\Theta}} \\ &= \left\|\mathbf{J}^{-1}\mathbf{b} - \boldsymbol{\Phi}\boldsymbol{\xi}\right\|^2_{\boldsymbol{\Theta}} \\ &= (\mathbf{J}^{-1}\mathbf{b} - \boldsymbol{\Phi}\boldsymbol{\xi})^T\boldsymbol{\Theta}(\mathbf{J}^{-1}\mathbf{b} - \boldsymbol{\Phi}\boldsymbol{\xi}). \end{aligned} \tag{22}$$

The first-order optimality condition for the minimization problem in Equation (21) is $\frac{\partial f}{\partial \boldsymbol{\Psi}} = \mathbf{0}$. From the chain rule and the fact that $\boldsymbol{\xi} = \boldsymbol{\xi}(\boldsymbol{\Psi})$ (because $\boldsymbol{\xi}$ depends on $\boldsymbol{\Psi}$ via $\mathbf{J}_r$, $\mathbf{A}_r$, and $\mathbf{B}_r$), we have $\frac{\partial f}{\partial \boldsymbol{\Psi}} = \frac{\partial f}{\partial \boldsymbol{\xi}}\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{\Psi}}$, in which case $\frac{\partial f}{\partial \boldsymbol{\xi}} = \mathbf{0}$ is sufficient to conclude that $\frac{\partial f}{\partial \boldsymbol{\Psi}} = \mathbf{0}$. The condition $\frac{\partial f}{\partial \boldsymbol{\xi}} = \mathbf{0}$ gives

$$\boldsymbol{\Phi}^T\boldsymbol{\Theta}\boldsymbol{\Phi}\boldsymbol{\xi} - \boldsymbol{\Phi}^T\boldsymbol{\Theta}\mathbf{J}^{-1}\mathbf{b} = \mathbf{0} \tag{23}$$

or, equivalently,

$$\boldsymbol{\xi} = \left(\boldsymbol{\Phi}^T\boldsymbol{\Theta}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^T\boldsymbol{\Theta}\mathbf{J}^{-1}\mathbf{b}. \tag{24}$$

We also have a direct expression for $\boldsymbol{\xi}$ from Equation (14):

$$\boldsymbol{\xi} = \left(\boldsymbol{\Psi}^T\mathbf{J}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Psi}^T\mathbf{b}. \tag{25}$$

Equations (25) and (24) must both hold for an arbitrary vector $\mathbf{b}$. Therefore, the following matrix equation for $\boldsymbol{\Psi}$ applies:

$$\left(\boldsymbol{\Psi}^T\mathbf{J}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Psi}^T = \left(\boldsymbol{\Phi}^T\boldsymbol{\Theta}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^T\boldsymbol{\Theta}\mathbf{J}^{-1}. \tag{26}$$

The solution to Equation (26) is, by inspection,

$$\boldsymbol{\Psi}^T = \boldsymbol{\Phi}^T\boldsymbol{\Theta}\mathbf{J}^{-1}. \tag{27}$$

Note that $\boldsymbol{\Psi}^T = \mathbf{C}\boldsymbol{\Phi}^T\boldsymbol{\Theta}\mathbf{J}^{-1}$, for any full-rank $\mathbf{C} \in \mathbb{R}^{l\times l}$, is also a solution to Equation (26). The choice of $\mathbf{C}$ will not affect the solution for $\boldsymbol{\xi}$ because $\mathbf{C}^{-1}\mathbf{C}$ immediately appears in Equation (25) (here, we take $\mathbf{C} = \mathbf{I}$).

Equation (27) provides a general solution for $\boldsymbol{\Psi}^T$ that satisfies the optimality condition in Equation (21). The solution for $\boldsymbol{\Psi}^T$ in Equation (27) depends on the matrix $\boldsymbol{\Theta}$ and in general involves the matrix $\mathbf{J}^{-1}$. In practice, special choices of $\boldsymbol{\Theta}$ that eliminate $\mathbf{J}^{-1}$ are often employed. Examples are Galerkin projection and Petrov–Galerkin projection, which we will describe in detail later. Other choices for $\boldsymbol{\Theta}$, which do not eliminate $\mathbf{J}^{-1}$, will lead to higher computational cost, but they may have better theoretical properties and display better numerical performance. We will discuss two such treatments, IP and WIP, in Section 7.

DOI: 10.1002/nme

4.2. Galerkin projection

Galerkin projection corresponds to the case where $\boldsymbol{\Theta} = \mathbf{J}$. Equation (27) then becomes $\boldsymbol{\Psi}^T = \boldsymbol{\Phi}^T$. Galerkin projection has been used in many previous POD-based order-reduction procedures [4, 6, 16, 21, 23, 43–45].

Galerkin projection enforces so-called Galerkin orthogonality, which means that the residual vector of Equation (10) is orthogonal to the state subspace $\boldsymbol{\Phi}$. It also minimizes the objective function $(\mathbf{e}_{23})^T\mathbf{J}\mathbf{e}_{23}$. When the matrix $\mathbf{J}$ is SPD, this projection method is optimal in the sense defined in Equation (21). However, if $\mathbf{J}$ is not SPD, $(\mathbf{e}_{23})^T\mathbf{J}\mathbf{e}_{23}$ will not be a norm definition for $\mathbf{e}_{23}$ because it could be negative. In subsurface flow simulations involving multiple phases, the Jacobian matrix is in general not SPD. Therefore, Galerkin projection is in general not a strict minimizer of the norm of $\mathbf{e}_{23}$.

However, despite this theoretical limitation, previous applications of POD for subsurface flow [4, 6, 16, 21, 23, 43–45] using Galerkin projection often show reasonable accuracy, especially in terms of the well production and injection rates. This may be due to the fact that the Galerkin orthogonality condition requires the residual vector to be orthogonal to the columns of $\boldsymbol{\Phi}$. Recall that the columns of $\boldsymbol{\Phi}$ capture the variations in the states because they are computed through application of POD. Therefore, variables with larger variation tend to have larger weights in the basis vectors. As a result, the corresponding equations are weighted more heavily and are thus solved more accurately. In reservoir simulation applications, variables with large variation often correspond to well blocks and to blocks in the near-well regions. The additional weighting applied to these blocks may enable Galerkin projection to provide accurate well rate predictions.

4.3. Petrov–Galerkin projection

If $\boldsymbol{\Theta}$ is taken as $\mathbf{J}^T\mathbf{J}$, Equation (27) becomes $\boldsymbol{\Psi}^T = \boldsymbol{\Phi}^T\mathbf{J}^T$. This projection method, called Petrov–Galerkin projection, has been used in [32] for order reduction of large-scale systems with high-dimensional parametric input and in [17] for order reduction within the context of DEIM. Petrov–Galerkin projection has some interesting properties. First, because $\mathbf{J}^T\mathbf{J}$ is SPD, the resulting $\boldsymbol{\Psi}$ satisfies Equation (21) and minimizes the norm of $\mathbf{e}_{23}$, with the norm defined as $\|\mathbf{e}\|^2_{\mathbf{J}^T\mathbf{J}} = \mathbf{e}^T\mathbf{J}^T\mathbf{J}\mathbf{e}$. Second, Petrov–Galerkin projection is equivalent to solving the normal equation for the overdetermined system in Equation (11), which minimizes the 2-norm of the residual vector.

The optimality condition for Petrov–Galerkin projection can also be interpreted in another way. With the Petrov–Galerkin method, Equation (21) is equivalent to

$$\boldsymbol{\Psi}^* = \arg\min_{\boldsymbol{\Psi}} \|\mathbf{e}_{23}\|^2_{\mathbf{J}^T\mathbf{J}} = \arg\min_{\boldsymbol{\Psi}} \|\mathbf{J}\mathbf{e}_{23}\|_2^2 . \tag{28}$$

In other words, this approach minimizes the 2-norm of $\mathbf{J}\mathbf{e}_{23}$. By writing the singular value decomposition of $\mathbf{J}$ as $\mathbf{J} = \mathbf{U}_J\boldsymbol{\Sigma}_J\mathbf{V}_J^T$, where $\mathbf{U}_J$ is an orthogonal matrix containing as its columns the left-singular vectors of $\mathbf{J}$, $\boldsymbol{\Sigma}_J$ is a diagonal matrix with the singular values of $\mathbf{J}$ on the diagonal, and $\mathbf{V}_J$ is an orthogonal matrix containing as its columns the right-singular vectors of $\mathbf{J}$ (or equivalently, the eigenvectors of $\mathbf{J}^T\mathbf{J}$), Equation (28) can be expressed as

$$\boldsymbol{\Psi}^* = \arg\min_{\boldsymbol{\Psi}} \left\| \boldsymbol{\Sigma}_J\mathbf{V}_J^T\mathbf{e}_{23} \right\|_2^2 . \tag{29}$$

Equation (29) indicates that the Petrov–Galerkin method transforms the error vector into a coordinate defined by $\mathbf{V}_J$, weights each element of the new vector with the corresponding singular value, and then minimizes the 2-norm of the weighted transformed error vector. In subsurface flow simulation, the Jacobian matrix $\mathbf{J}$ is usually very ill-conditioned, and its singular values vary widely, spanning up to 10 orders of magnitude. Therefore, the weighting scheme using singular values can be highly skewed. In other words, some components in the error vector will be very strongly weighted, while others may be very weakly weighted. Therefore, although the Petrov–Galerkin method is optimal in the sense defined in Equation (21), the skewed weighting embedded in the method may result in inaccuracy in the resulting POD–TPWL model for some quantities.

DOI: 10.1002/nme

5. STABILITY CRITERIA

In addition to accuracy and efficiency considerations, the constraint reduction matrix $\mathbf{\Psi}$ should also be selected to ensure the stability of the resulting POD–TPWL method. In theory, we could attempt to construct an SPD matrix $\mathbf{\Theta}$ in Equation (27) that also satisfies the Lyapunov stability criteria. This would provide a guaranteed stable and optimally accurate $\mathbf{\Psi}$. This approach, however, would entail coupling Equation (27) with the Lyapunov equations and solving the resulting matrix equations. Such an approach is unlikely to be computationally tractable for high-dimensional subsurface flow problems. Therefore, in this work, we start with constraint reduction methods that satisfy Equation (27) and then consider methods to assess and enhance their stability.

The stability of ROMs based on TPWL was first considered in [20], which addressed order reduction of nonlinear ODE systems. In that work, the piecewise linear reduced-order ODE system was deemed stable when the coefficient matrix of the linear ODE system at each linearization step is a Hurwitz matrix (all eigenvalues have negative real part). Bond and Daniel [31] considered ROMs for linear time-invariant ODE systems and proposed that $\mathbf{\Psi}$ be chosen to guarantee stability by satisfying the Lyapunov equations of the reduced system. However, their approach involves solving a matrix optimization problem, which for our models would entail an optimization of $n_v \times l$ variables under constraints for each time step of the training simulation. As indicated earlier, this amount of computation will be prohibitive for models of reasonable size.

He *et al.* [23] considered the stability of POD–TPWL for fully discretized (PDE) systems in the context of reservoir simulation. There it was shown that, in order for the POD–TPWL model to be stable, the amplification factor at each time step of a particular matrix should be less than 1. This analysis will now be described within the framework of error assessment.

The stability of the POD–TPWL formulation can be analyzed from the error propagation expression (Equation (18)), which can be rewritten as

$$\mathbf{e}_{04}^{n+1} = \mathbf{M}^{i+1}\mathbf{e}_{04}^{n} + \mathbf{e}_{03}^{n+1}, \tag{30}$$

where

$$\mathbf{M}^{i+1} = -\mathbf{\Phi}\left(\mathbf{J}_r^{i+1}\right)^{-1}\mathbf{A}_r^{i+1}\mathbf{\Phi}^T, \tag{31}$$

with $\mathbf{M}^{i+1} \in \mathbb{R}^{n_v \times n_v}$ (note that $\mathbf{M}^{i+1}$ appeared originally in Equation (17)). Equation (30) indicates that the total error of the POD–TPWL model at time step $n+1$ is the total error at time step n amplified by the matrix $\mathbf{M}^{i+1}$, which is referred to as the amplification matrix, plus the one-step error $\mathbf{e}_{03}^{n+1}$ ($\mathbf{e}_{03}^{n+1} = \mathbf{e}_{01}^{n+1} + \mathbf{e}_{12}^{n+1} + \mathbf{e}_{23}^{n+1}$) incurred at time step $n+1$. It is important to observe that $\mathbf{M}^{i+1}$ in Equation (31) is defined as a projection of the matrix product $-(\mathbf{J}_r^{i+1})^{-1}\mathbf{A}_r^{i+1}$, whose dimension is $l \times l$. Therefore, although $\mathbf{M}^{i+1} \in \mathbb{R}^{n_v \times n_v}$, its rank is at most l.

We denote the spectral radius of $\mathbf{M}^{i+1}$ as γ^{i+1} and refer to it as the amplification factor. For a matrix $\mathbf{M}$ that does not vary with time (i.e., $\mathbf{M}^{i+1} = \mathbf{M}$), the error $\mathbf{e}_{04}^{n}$ in Equation (30) will not grow exponentially if and only if the amplification factor γ^{i+1} is less than or equal to 1 [46]. In a piecewise linear system, $\mathbf{M}^{i+1}$ generally changes at each time step. However, the amplification factor γ^{i+1} was still shown to be a strong indicator of stability. Specifically, it was demonstrated in [23] that an isolated γ^{i+1} that is greater than 1 may amplify the error at a specific time step and create a spike in the solution. Several consecutive time steps with $\gamma^{i+1} > 1$ may cause the solution to become unphysical. In practice, however, the occasional occurrence of γ^{i+1} values that only slightly exceed 1 does not appear to cause numerical instability. Therefore, to ensure that the error does not amplify over time, we require that γ^{i+1} not exceed 1 by more than a small threshold; e.g., $\gamma^{i+1} < 1.05$. This will be used as the criterion to assure (essentially) stable POD–TPWL behavior in this work. We note that [20] and [47] also used the spectral radius of an amplification matrix as a stability indicator for piecewise linear reduced-order ODE systems.

Because $\mathbf{M}^{i+1}$ is a high-dimensional matrix, its spectral radius can be expensive to compute. Fortunately, because of the rank deficiency of $\mathbf{M}^{i+1}$ ($\mathbf{M}^{i+1}$ has a maximum rank of l), we do not

DOI: 10.1002/nme

need to analyze this high-dimensional matrix to assess POD–TPWL stability. Rather, we define $\mathbf{M}_r^{i+1} \in \mathbb{R}^{l \times l}$ as

$$\mathbf{M}_r^{i+1} = -\left(\mathbf{J}_r^{i+1}\right)^{-1} \mathbf{A}_r^{i+1}. \tag{32}$$

We now show that $\mathbf{M}^{i+1}$ and $\mathbf{M}_r^{i+1}$ have the same nonzero eigenvalues and thus the same spectral radius.

From the definitions of $\mathbf{M}^{i+1}$ and $\mathbf{M}_r^{i+1}$ in Equations (31) and (32), we see that $\mathbf{M}^{i+1} = \mathbf{\Phi}\mathbf{M}_r^{i+1}\mathbf{\Phi}^T$. In addition, using the fact that $\mathbf{\Phi}$ is orthonormal (i.e., $\mathbf{\Phi}^T\mathbf{\Phi} = \mathbf{I}$), we have

$$\mathbf{M}_r^{i+1} = (\mathbf{\Phi}^T\mathbf{\Phi})\mathbf{M}_r^{i+1}(\mathbf{\Phi}^T\mathbf{\Phi}) = \mathbf{\Phi}^T(\mathbf{\Phi}\mathbf{M}_r^{i+1}\mathbf{\Phi}^T)\mathbf{\Phi} = \mathbf{\Phi}^T\mathbf{M}^{i+1}\mathbf{\Phi}. \tag{33}$$

Using the preceding expressions, the fact that the matrices $\mathbf{M}^{i+1}$ and $\mathbf{M}_r^{i+1}$ have the same nonzero eigenvalues and thus the same spectral radius can be shown as follows. Let $\mathbf{y}$ be an eigenvector of $\mathbf{M}^{i+1}$ with nonzero eigenvalue λ; that is, $\mathbf{M}^{i+1}\mathbf{y} = \lambda\mathbf{y}$. Using this, along with the relationship between $\mathbf{M}^{i+1}$ and $\mathbf{M}_r^{i+1}$, and premultiplying by $\mathbf{\Phi}^T$, we have

$$\mathbf{\Phi}^T\mathbf{M}^{i+1}\mathbf{y} = \mathbf{\Phi}^T\mathbf{\Phi}\mathbf{M}_r^{i+1}\mathbf{\Phi}^T\mathbf{y} = \mathbf{M}_r^{i+1}\mathbf{\Phi}^T\mathbf{y} = \lambda\mathbf{\Phi}^T\mathbf{y}. \tag{34}$$

This means $\mathbf{\Phi}^T\mathbf{y}$ is an eigenvector of $\mathbf{M}_r^{i+1}$ with eigenvalue λ. Similarly, if $\boldsymbol{\eta}$ is an eigenvector of $\mathbf{M}_r^{i+1}$ with nonzero eigenvalue λ, we have

$$\mathbf{M}^{i+1}\mathbf{\Phi}\boldsymbol{\eta} = \mathbf{\Phi}\mathbf{M}_r^{i+1}\mathbf{\Phi}^T\mathbf{\Phi}\boldsymbol{\eta} = \mathbf{\Phi}\mathbf{M}_r^{i+1}\boldsymbol{\eta} = \lambda\mathbf{\Phi}\boldsymbol{\eta}, \tag{35}$$

which means $\mathbf{\Phi}\boldsymbol{\eta}$ is an eigenvector for $\mathbf{M}^{i+1}$ with eigenvalue λ. Therefore, $\mathbf{M}^{i+1}$ and $\mathbf{M}_r^{i+1}$ have the same nonzero eigenvalues and thus the same spectral radius γ^{i+1}. In practice, γ^{i+1} can thus be calculated efficiently by computing the largest eigenvalue of the low-dimensional matrix $\mathbf{M}_r^{i+1}$, as described in [23].

Given the derivative matrices $\mathbf{J}^{i+1}$ and $\mathbf{A}^{i+1}$ and simulation results for all training simulations, the amplification matrix only depends on the number of reduced variables (l_p and l_S for oil–water problems and l_p and l_z for compositional problems), which determine $\mathbf{\Phi}$ and the constraint reduction matrix $\mathbf{\Psi}$. For general choices of l_p and l_S (or l_z) and $\mathbf{\Psi}$, the resulting POD–TPWL model can be unstable. It was shown in [23] that, for oil–water problems, the stability behavior of the POD–TPWL model using Galerkin projection can be very sensitive to the choice of l_p and l_S. Improved results for oil–water problems were achieved in [23] by finding the (l_p, l_S) combination that provides the best stability properties. This approach entails specifying minimum and maximum values for l_p and l_S, calculating the maximum amplification factor for all time steps i in the training run ($\max_i \gamma^i$) for all (l_p, l_S) combinations considered, and selecting the (l_p, l_S) combination that provides the lowest $\max_i \gamma^i$. This can be accomplished efficiently (in low-dimensional space) because the reduced derivative matrices $\mathbf{J}_r^{i+1}$ and $\mathbf{A}_r^{i+1}$ for different (l_p, l_S) combinations are just submatrices of the reduced derivative matrices for the largest values of l_p and l_S considered. Note that this (l_p, l_S) selection method will only be effective when $\max_i \gamma^i$ is below the specified maximum (e.g., 1.05).

This procedure provides an offline indication of POD–TPWL solution stability using only low-order computations. This stability indicator can be applied for any constraint reduction procedure, and we will use it for all of the approaches considered in this paper. Along these lines, the stability of POD–TPWL using Petrov–Galerkin projection has not, to our knowledge, been previously studied. In Section 6, we will see that Petrov–Galerkin projection provides much better stability than Galerkin projection for challenging cases. In addition, in Section 7, we will show that the two new projection methods introduced here, IP and WIP, also provide POD–TPWL models that behave stably.

DOI: 10.1002/nme

6. NUMERICAL IMPLEMENTATION AND RESULTS

We now describe some key aspects of the POD–TPWL implementation and the way in which we quantify error. Numerical results are then presented for two oil–water models and one compositional model. Additional results and discussion can be found in [29].

6.1. POD–TPWL implementation

The POD–TPWL method has been implemented for compatibility with Stanford's AD-GPRS [41]. AD-GPRS was modified to output the state and derivative information required to construct the POD–TPWL model.

During the offline (preprocessing) stage, two or three training simulations are performed using AD-GPRS for specific sets of input (BHP) control parameters. One of the training runs is designated the primary training simulation. This run provides state vectors and derivative matrices for use in subsequent (inline) computations. The other (secondary) training runs are used only to provide snapshots for the construction of the basis matrix $\mathbf{\Phi}$. Secondary training runs are needed because the number of snapshots from a single simulation run, which corresponds to the number of time steps in that run, is typically not enough to provide a high-quality POD basis matrix. More specifically, we have found that around 300 snapshots are needed to construct the POD basis for the problems considered in this paper. Given that a typical run entails 100–200 time steps, this corresponds to two to three training runs. We do not apply any special procedures to determine the controls used in the primary or secondary training runs (these controls are all generated randomly over the range of interest), although it is possible that better POD–TPWL accuracy could be achieved through the use of a more formal approach.

In the basic implementation of the offline procedure, the state vector at each time step of each training simulation, as well as the derivative matrices $\mathbf{J}^i$, $\mathbf{A}^i$, and $\mathbf{B}^i$ at each time step of the primary training simulation, are saved to disk. The POD basis is then constructed from the snapshot matrices. Finally, the reduced states ($\boldsymbol{\xi}^i = \mathbf{\Phi}^T \mathbf{x}^i$) and derivative matrices are formed, with the latter computed using Equation (13).

If full-order derivative matrices are written as output, the storage requirements for the preprocessing computations can be very large. In [25], a reduced-storage offline procedure is described, in which only the reduced-order matrices $\mathbf{J}_r^i$, $\mathbf{A}_r^i$, and $\mathbf{B}_r^i$ are written to disk. This approach requires that the primary training simulation be run twice—once to provide snapshots before construction of the basis matrix $\mathbf{\Phi}$ and once after. The computational cost for the second of these runs can be reduced substantially, however, as the converged states are already known (meaning no iteration or linear solutions are required). See [25] for more details on both the basic and reduced-storage offline procedures.

In the inline stage, Equation (15) is evaluated for control parameters $\mathbf{u}^{n+1}$ that differ from those used in the training runs. After the reduced state $\boldsymbol{\xi}^{n+1}$ is computed, the full-order state $\mathbf{x}^{n+1}$ can be reconstructed at selected locations (e.g., well blocks), and other quantities of interest, such as the phase flow rates for each well, can be calculated. For compositional problems, the calculation of well flow rate additionally requires that a flash calculation be performed at the well blocks to determine secondary variables such as oil saturation. More details can be found in [25].

As indicated earlier, in addition to performing the training simulations, in the offline stage, we must also output detailed information at each time step of the primary training simulation, construct the POD basis, and reduce the training states and derivatives. The reduction of the state vectors and derivative matrices constitutes the majority of the additional offline overhead. For both the Galerkin and Petrov–Galerkin constraint reduction procedures, this offline cost is approximately equal to the cost of an additional full-order simulation. Once the POD–TPWL model is constructed, however, an inline run typically takes only a few seconds. Thus, POD–TPWL is most suitable for use in applications requiring a large number of simulations with different input control parameters, as is the case for production optimization computations.

DOI: 10.1002/nme

6.2. *Error definitions*

In order to assess the accuracy of POD–TPWL models, we compute the mismatch (error) in well flow rates between the full-order AD-GPRS solution (Q_{full}) and the POD–TPWL simulation (Q_{tpwl}). Phase flow-rate errors for a particular well are computed at each time step, then integrated over time, and then normalized by the time-integrated flow rate for that well from the full-order solution. Errors from all wells of the same type (injection or production) are then averaged to provide overall error values. For example, the overall average error for oil production rate, designated E_o, is computed as

$$E_o = \frac{1}{n_{pw}} \sum_{j=1}^{n_{pw}} \frac{\int_0^T |Q^j_{o,full} - Q^j_{o,tpwl}| \, \mathrm{d}t}{\int_0^T Q^j_{o,full} \, \mathrm{d}t}, \tag{36}$$

where subscript o designates oil, superscript j indicates a particular production well, n_{pw} is the total number of production wells, and T is the total simulation time. The integration is performed using the trapezoidal rule.

For oil–water models, we compute the error in oil production rate E_o, water production rate E_w, and water injection rate E_{iw}. For compositional models, we calculate the error for oil production rate E_o, gas production rate E_g, and gas injection rate E_{ig}. These errors are computed using expressions analogous to Equation (36).

We also calculate the average state error over all time steps and all grid blocks. For example, the average error in pressure E_p is defined as

$$E_p = \frac{1}{n_t n_b} \sum_{i=1}^{n_t} \sum_{k=1}^{n_b} \left| p^i_{k,full} - p^i_{k,tpwl} \right|, \tag{37}$$

where n_t is the number of time steps in the simulation, n_b is the number of grid blocks, $p^i_{k,full}$ is the full-order pressure solution for block k at time step i, and $p^i_{k,tpwl}$ is the analogous quantity for the POD–TPWL model, constructed through application of $\mathbf{x} = \mathbf{\Phi}\boldsymbol{\xi}$. Similar expressions are used for other variables. For oil–water models, we compute the average error in pressure and water saturation (E_p and E_S) and, for compositional models, the average error in pressure and mole fraction of the injected component (E_p and E_z).

6.3. *Case 1: oil–water flow with equal phase densities*

The reservoir model for cases 1 and 2, shown in Figure 1, is a portion of the so-called Stanford VI reservoir model developed in [48]. The model represents a fluvial depositional system, with high-permeability (sand) channels embedded in a low-permeability background shale. The dimensions of

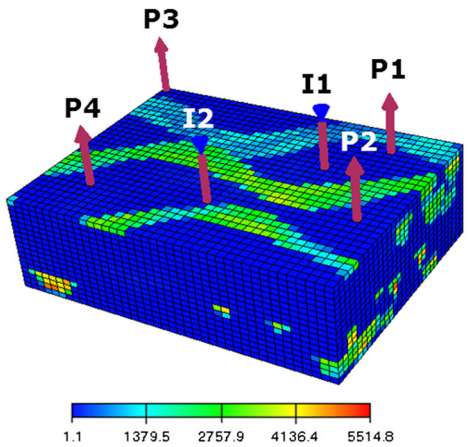


Figure 1. Reservoir model for cases 1 and 2 (permeability in the x-direction is shown).

DOI: 10.1002/nme

the grid are $30 \times 40 \times 17$, for a total of 20,400 cells. The model contains two injection wells and four production wells, as indicated in Figure 1. The production wells are perforated (open to flow) in the upper five layers and the injection wells in the lower three layers. The wells are controlled through specification of time-varying BHP.

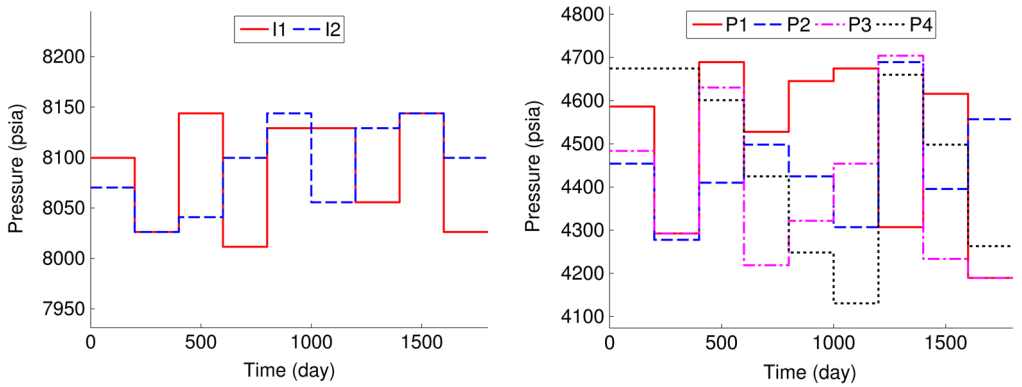
The relative permeability functions, which quantify the relative amounts of water and oil flow in each grid block, are as follows:

$$k_{rw} = \left(\frac{S_w - S_{wc}}{1 - S_{or} - S_{wc}} \right)^2, \quad k_{ro} = \left(\frac{S_o - S_{or}}{1 - S_{or} - S_{wc}} \right)^2, \tag{38}$$

where oil saturation $S_o = 1 - S_w$. The parameters S_{wc} and S_{or}, specified to be 0.02 and 0.3, respectively, account for the fact that both phases cease to flow below some threshold saturation. The oil and water viscosities (μ_o and μ_w) are 3 and 1 cp, respectively. These, together with the relative permeability functions, define the phase mobility functions in Equation (1) (specifically, $\lambda_o = k_{ro}/\mu_o$ and $\lambda_w = k_{rw}/\mu_w$). The oil and water phase densities are both specified to be 1000 kg/m^3. The initial reservoir pressure is 5880 psi (405.4 bar), and the initial water saturation is 0.1.

Three training simulations are performed to construct the POD–TPWL model, one of which is used as the primary training run. The BHP controls of the two injectors and four producers are shown in Figure 2. Well settings are varied every 200 days and are generated randomly over a prescribed range. We take this range to be relatively narrow in the examples in this paper in order to focus on constraint reduction error and stability behavior. From the three training simulations, 334 snapshots are collected to construct the basis matrix $\mathbf{\Phi}$. We use 70 reduced pressure variables and 100 reduced water saturation variables ($l_p = 70$, $l_S = 100$). Figure 3 shows the randomly generated BHPs for the test case, which differ from the training-run BHPs. We note that the training and test-case BHP profiles are meant to resemble those generated during a computational optimization procedure, where the goal is to determine the time-varying BHPs that maximize a prescribed measure of reservoir performance.

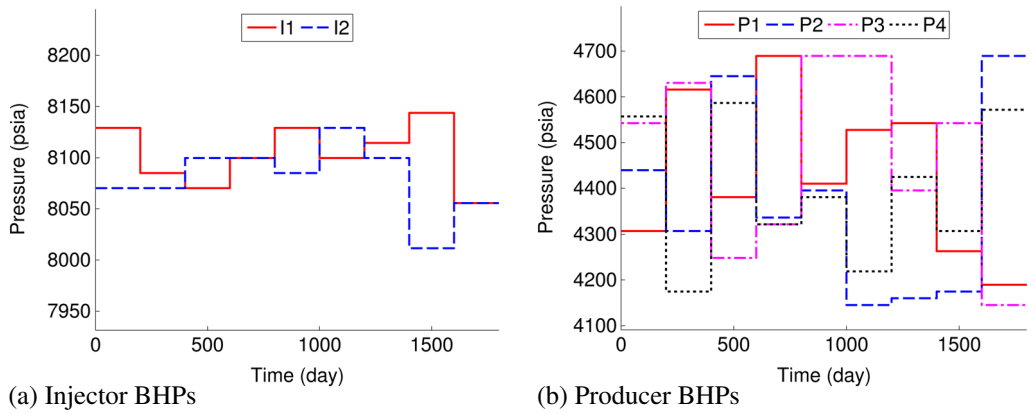
As discussed in [23], for oil–water systems with equal phase densities, the POD–TPWL model with Galerkin projection tends to behave stably for most (l_p, l_S) combinations. Figure 4 shows the amplification factor γ^i at each time step i for both Galerkin projection (GLK) and Petrov–Galerkin projection (PG). The label 'GLK_70_100' in the legend indicates the result using Galerkin projection with 70 reduced pressure variables and 100 reduced water saturation variables. The labels for other constraint reduction methods follow this format. It is clear from Figure 4 that the time-varying γ^i for both methods for this case are very close to 1. Indeed, they are below 1 for the entire simulation period, meaning that the resulting POD–TPWL models are always stable. Note that there are small spikes in γ^i every 200 days. These are due to the very small time steps used in the training



(a) Injector BHPs (b) Producer BHPs

Figure 2. Time-varying bottom-hole pressures (BHPs) for the primary training simulation for cases 1 and 2. (a) Injector BHPs. (b) Producer BHPs.

DOI: 10.1002/nme



(a) Injector BHPs (b) Producer BHPs

Figure 3. Time-varying bottom-hole pressures (BHPs) for the test simulation for cases 1 and 2. (a) Injector BHPs. (b) Producer BHPs.

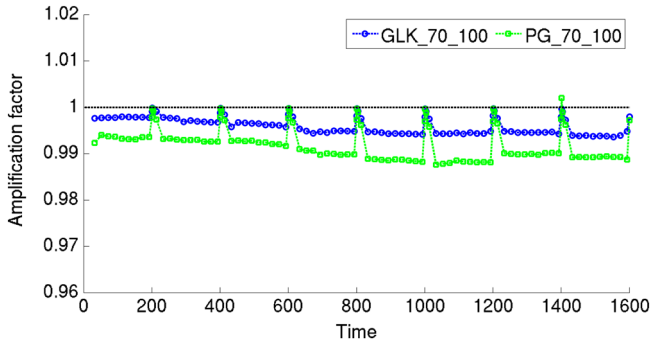


Figure 4. Amplification factor γ^i for each time step in case 1. GLK, Galerkin projection; PG, Petrov–Galerkin projection.

run when the BHPs are changed. For an infinitely small time step, the Jacobian matrix **J** will equal the negative of the **A** matrix, and the resulting amplification matrix will be the identity matrix, with γ^i of 1. We note that, although we show results only for $l_p = 70$ and $l_S = 100$, similar stable performance can be observed for different l_p and l_S over a reasonable range (as will be illustrated later).

We now compare the performance of the Galerkin and Petrov–Galerkin methods for case 1. Figure 5 shows the oil production rate for producer 1, and Figure 6 shows the water injection rate for injector 1. The black dotted line depicts the result from the primary training simulation, about which we linearize. Shown in red is the full-order (reference) AD-GPRS solution for the test case. It is clear that the test and training solutions differ. Galerkin and Petrov–Galerkin projection results are shown in blue (open circles) and green (open squares), respectively. Both approaches provide accurate results in this case, although Galerkin projection can be seen to be slightly more accurate for water injection (see, e.g., results at early time in Figure 6). The results in Figures 5 and 6 are representative of those for the other wells.

Table I summarizes the flow rate errors and average state errors from POD–TPWL using the two constraint reduction methods. For the same values of l_p and l_S, Galerkin projection provides better accuracy for this case than Petrov–Galerkin projection in all five error measures. This may be due to inaccuracy resulting from the skewed weighting inherent in Petrov–Galerkin projection, as discussed in Section 4.3.

For this example, the full-order parallelized AD-GPRS simulation requires about 290 s to run on a cluster node with eight cores (dual quad-core Nehalem™). The POD–TPWL model, with either Galerkin or Petrov–Galerkin projection, requires approximately 0.7 s on a single core of the same processor. This corresponds to a runtime speedup of a factor of about 400. As noted earlier,

DOI: 10.1002/nme

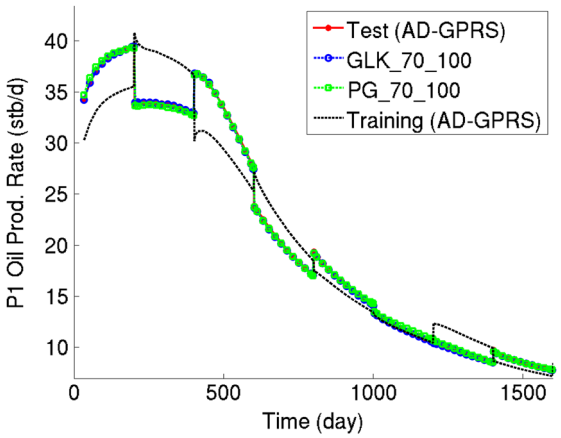


Figure 5. Oil production rate for producer 1 in case 1. Results for test (AD-GPRS), GLK_70_100 and PG_70_100 essentially overlay one another. AD-GPRS, Automatic Differentiation-based General Purpose Research Simulator; GLK, Galerkin projection; PG, Petrov–Galerkin projection.

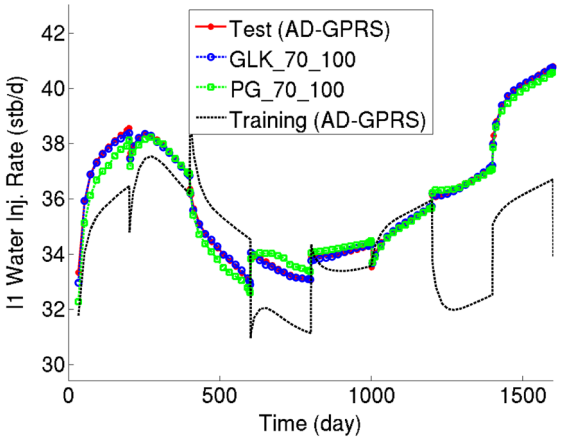


Figure 6. Water injection rate for injector 1 in case 1. Results for test (AD-GPRS) and GLK_70_100 essentially overlay one another. AD-GPRS, Automatic Differentiation-based General Purpose Research Simulator; GLK, Galerkin projection; PG, Petrov–Galerkin projection.

Table I. Summary of error for case 1.

	E_o	E_w	E_{iw}	E_p (psi)	E_S
GLK_70_100	0.0054	0.0058	0.0015	1.57	0.00066
PG_70_100	0.0113	0.0156	0.0052	7.07	0.00130

GLK, Galerkin projection; PG, Petrov–Galerkin projection.

preprocessing requires three training simulations plus overhead computations equal to about an additional full-order simulation.

6.4. Case 2: oil–water flow with unequal phase densities

In case 1, both projection methods were stable. However, stability may become an issue when more complicated physics is introduced. This will be illustrated in case 2, which is identical to case 1 except that now the oil and water phase densities are set to 800 and 1000 kg/m^3, respectively. The difference in density results (physically) in gravity-driven countercurrent flow, which leads to changes in the structure of the Jacobian matrix because of upstream weighting [33]. As in case 1, we perform three training simulations to construct the POD–TPWL model; one of these is the primary

DOI: 10.1002/nme

training run. The time-varying BHP controls for the training and test simulations are the same as in case 1 (Figures 2 and 3). From the three training simulations, 335 snapshots are collected. We again take $l_p = 70$ and $l_S = 100$.

Figure 7 shows the oil production rate for producer 1, and Figure 8 shows the water injection rate for injector 1. It can be seen that the results using Petrov–Galerkin projection match the true test-case solution fairly closely, although some differences are evident. The results using Galerkin projection, by contrast, display substantial fluctuations in the early stage of the simulation. Similar fluctuations for the Galerkin projection run are also observed in flow rates for the other wells (not shown here).

The different performances of the Galerkin and Petrov–Galerkin projections for this case can be explained by Figure 9, which shows the amplification factor γ^i at each time step for the two methods. For the Petrov–Galerkin method, the value of γ^i is below or very near 1 for the entire simulation period. Thus, the Petrov–Galerkin method performs stably throughout the simulation. Again, the peaks visible every 200 days are due to the small time steps used when the BHP controls change. For the Galerkin method, the value of γ^i is much larger than 1 at early time, and at some time steps, it is as high as 14.5 (note that the vertical axis only extends to 1.5). This instability causes the fluctuations evident in Figure 8. At later time, the amplification factor decreases to around 1, and the fluctuations disappear.

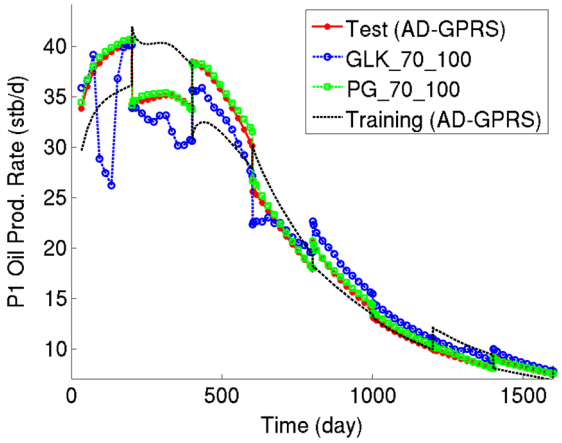


Figure 7. Oil production rate for producer 1 in case 2. AD-GPRS, Automatic Differentiation-based General Purpose Research Simulator; GLK, Galerkin projection; PG, Petrov–Galerkin projection.

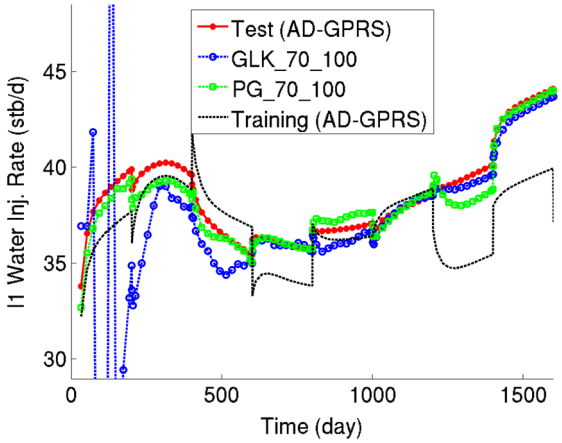


Figure 8. Water injection rate for injector 1 in case 2. AD-GPRS, Automatic Differentiation-based General Purpose Research Simulator; GLK, Galerkin projection; PG, Petrov–Galerkin projection.

DOI: 10.1002/nme

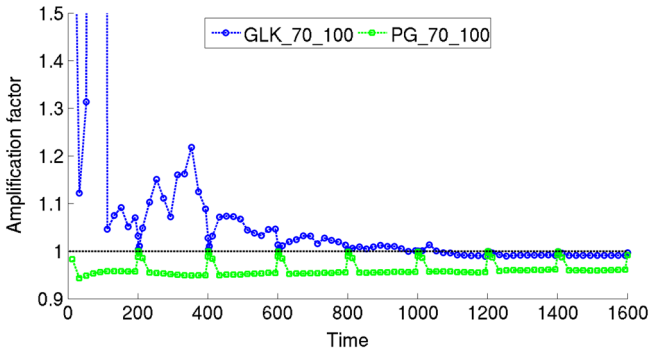
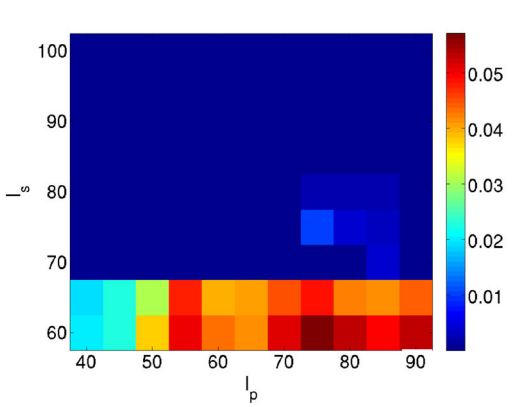
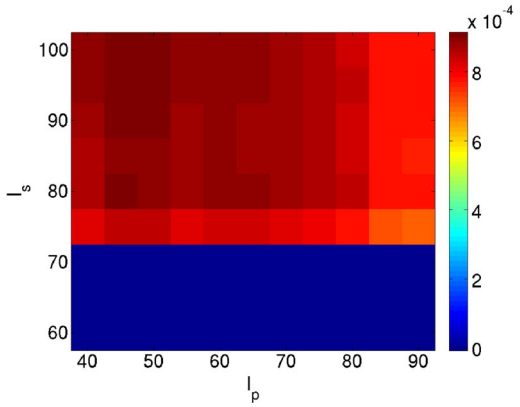


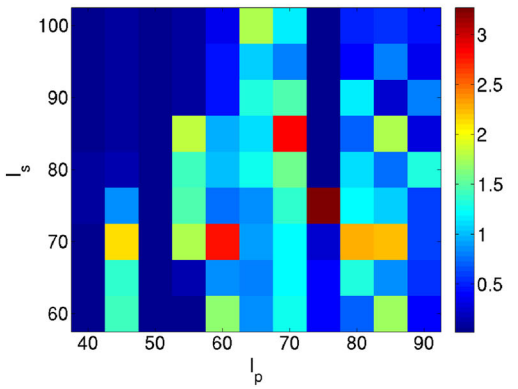
Figure 9. Amplification factor γ^i for each time step in case 2. GLK, Galerkin projection; PG, Petrov–Galerkin projection.



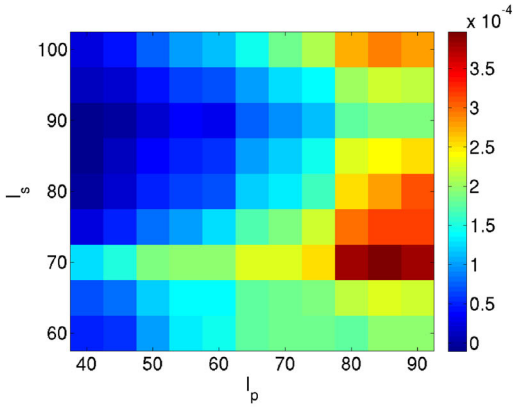
(a) Galerkin projection for Case 1 (minimum value on color bar: 4.8×10^{-6})



(b) Petrov-Galerkin projection for Case 1 (minimum value on color bar: -8.0×10^{-6})



(c) Galerkin projection for Case 2 (minimum value on color bar: 1.7×10^{-2})



(d) Petrov-Galerkin projection for Case 2 (minimum value on color bar: -1.1×10^{-5})

Figure 10. Maps of $\log_{10}(\max_i \gamma^i)$ for cases 1 and 2. (a) Galerkin projection for case 1 (minimum value on color bar: 4.8×10^{-6}). (b) Petrov–Galerkin projection for case 1 (minimum value on color bar: -8.0×10^{-6}). (c) Galerkin projection for case 2 (minimum value on color bar: 1.7×10^{-2}). (d) Petrov–Galerkin projection for case 2 (minimum value on color bar: -1.1×10^{-5}).

DOI: 10.1002/nme

Note that the stability results in Figure 9 are specifically for $l_p = 70$ and $l_S = 100$. As discussed earlier, the choice of l_p and l_S affects stability behavior. Figure 10 depicts in log scale the values of $\max_i \gamma^i$ for different combinations of l_p and l_S for Galerkin (lower left) and Petrov–Galerkin (lower right) projection methods for case 2. These plots will be referred to as stability maps. Note that the limits of the color bars for the two plots are very different (the minimum value for each color bar, which corresponds to the minimum value in the corresponding map, is indicated). As discussed earlier, for a problem with a constant matrix $\mathbf{M}$ in which time tends to infinity, we would require $\log_{10}(\max_i \gamma^i) < 0$ to assure stability. However, in practice, stable behavior in POD–TPWL models is observed as long as $\log_{10}(\max_i \gamma^i)$ is close to zero (e.g., $\log_{10}(\max_i \gamma^i) \lesssim 0.02$ for our examples; this precise value may be problem/control-setting dependent). This is the case for two reasons. First, because our models entail $O(100)$ time steps, very small growth rates do not lead to unbounded errors. Second, because the amplification matrix ($\mathbf{M}^{i+1}$ or $\mathbf{M}_r^{i+1}$) varies from time step to time step and because Figure 10 depicts the maximum $\log_{10}\gamma^i$ over all time steps, the requirement that $\log_{10}(\max_i \gamma^i)$ be less than zero is overly strict. In the following, we thus refer to cases for which $\max_i \gamma^i < 1.05$, which corresponds to $\log_{10}(\max_i \gamma^i) < 0.021$, as behaving stably.

The $\log_{10}(\max_i \gamma^i)$ for Galerkin projection for case 2 is displayed in Figure 10(c). It can be seen from the plots that, for Galerkin projection POD–TPWL, stability is very sensitive to the choice of (l_p, l_S), and there is no clear trend. Many of the l_p and l_S combinations lead to instability (as they correspond to large $\max_i \gamma^i$), while there are some l_p and l_S combinations for which the POD–TPWL model should behave stably. On the other hand, as shown in Figure 10(d), for Petrov–Galerkin projection, the value of $\log_{10}(\max_i \gamma^i)$ for any combination of l_p and l_S is very close to zero (the largest value being 0.0004). Thus, this method behaves stably for all combinations of l_p and l_S considered.

Also shown in Figure 10 are the stability maps for Galerkin (upper left) and Petrov–Galerkin (upper right) for case 1 (equal phase densities). It is clear that for case 1, Petrov–Galerkin projection should behave stably for all (l_p, l_S) considered, and Galerkin projection should behave stably over a large range of l_p and l_S. A comparison of Figure 10(a) and (c) demonstrates that the inclusion of more complicated physics has the potential to adversely affect the performance of POD–TPWL models that apply Galerkin projection for constraint reduction. This problem may be related to the fact that the Jacobian matrix is not SPD, as discussed in Section 4.2.

Because the stability characteristics of POD–TPWL models that apply Galerkin projection can be sensitive to the choice of l_p and l_S, the model can be 'stabilized' (in a practical sense) through the careful selection of l_p and l_S, assuming there exists an (l_p, l_S) combination with $\log_{10}(\max_i \gamma^i) < 0.021$. Such an approach was applied in [23]. For the current example, for $l_p = 75$ and $l_S = 80$, the value of $\log_{10}(\max_i \gamma^i)$ is below this threshold (Figure 10(c)), which suggests that the resulting POD–TPWL model will behave stably. Figure 11 shows the water injection rate for injector 1 for both projection methods with $l_p = 75$ and $l_S = 80$. It is clear that the results using Galerkin projection no longer exhibit fluctuations (compare Figures 8 and 11) and that the Petrov–Galerkin results continue to be stable, as would be expected from Figure 10(d). These results illustrate the efficacy of constructing the POD–TPWL stability maps shown in Figure 10.

Table II shows the error for the Petrov–Galerkin method and the stably behaving Galerkin method. The results indicate that the stabilized Galerkin projection is more accurate for this case (with the exception of E_S), which is consistent with the observations for case 1.

6.5. Case 3: compositional simulation

Case 3 involves a four-component system in which we model the injection of CO_2 into an oil reservoir. The original fluid in place, in terms of overall mole fractions, consists of 0.01 CO_2, 0.11 of the C_1 component, 0.29 of the C_4 component, and 0.59 of the C_{10} component. Pure CO_2 is injected. The reservoir model for this case is defined on a $32 \times 40 \times 8$ grid, which translates to 10,240 grid blocks and thus 40,960 primary variables ($10{,}240 \times 4$). The permeability field, shown in Figure 12, was generated geostatistically using sequential Gaussian simulation within the SGeMS geological modeling package [49]. The model contains four producers at the four corners and one injector in

DOI: 10.1002/nme

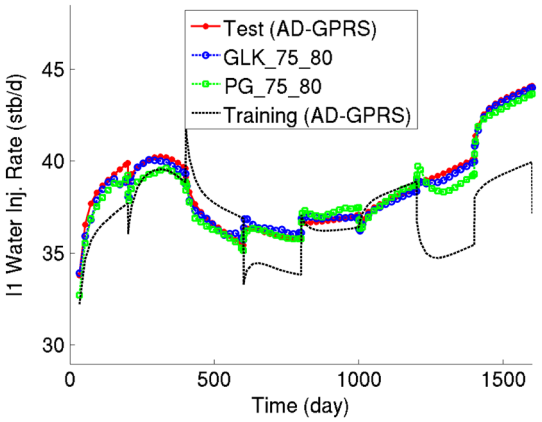


Figure 11. Water injection rate for injector 1 in case 2, with l_p and l_S selected based on Figure 10(c). AD-GPRS, Automatic Differentiation-based General Purpose Research Simulator; GLK, Galerkin projection; PG, Petrov–Galerkin projection.

Table II. Summary of error for case 2.

	E_o	E_w	E_{iw}	E_p (psi)	E_S
GLK_75_80	0.0216	0.0152	0.0050	13.6	0.00212
PG_75_80	0.0225	0.0255	0.0112	15.0	0.00204

GLK, Galerkin projection; PG, Petrov–Galerkin projection.

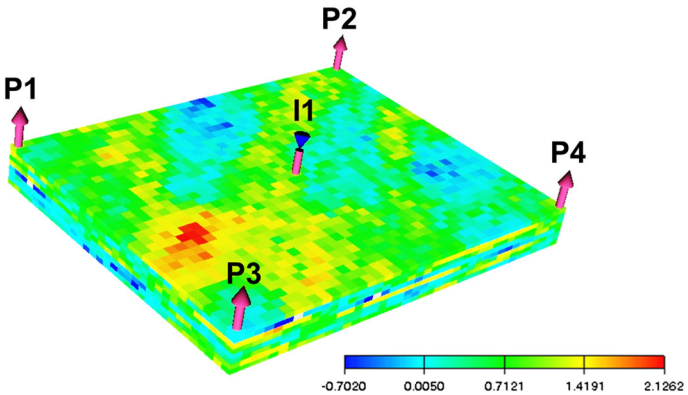


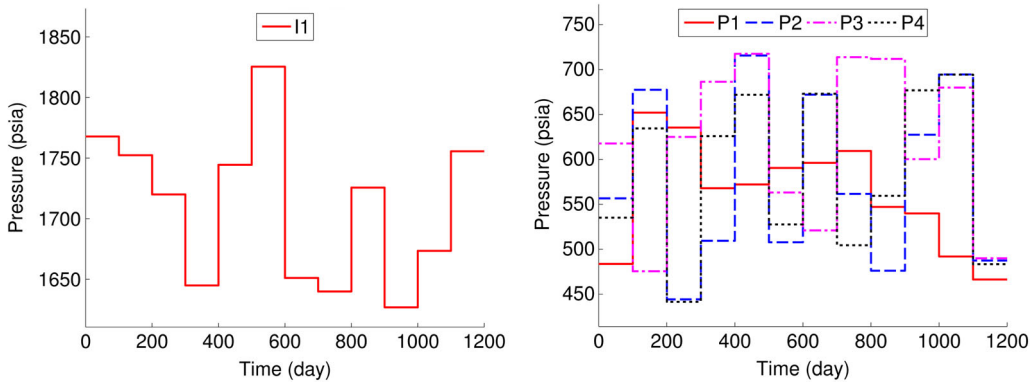
Figure 12. Reservoir model for case 3 (log-permeability is shown).

the middle, forming a five-spot pattern. The producers are perforated in the lower four layers and the injector in the upper four layers.

We perform two training runs, one of which is the primary training simulation, to construct the POD–TPWL model. The BHP controls for the injector and four producers are shown in Figure 13. These BHPs are varied every 100 days and are generated randomly. From the two training simulations, 275 snapshots are collected to build the basis matrix $\mathbf{\Phi}$. We specify $l_p = 120$ and $l_z = 150$. Figure 14 shows the BHPs for the test case, which are also randomly generated.

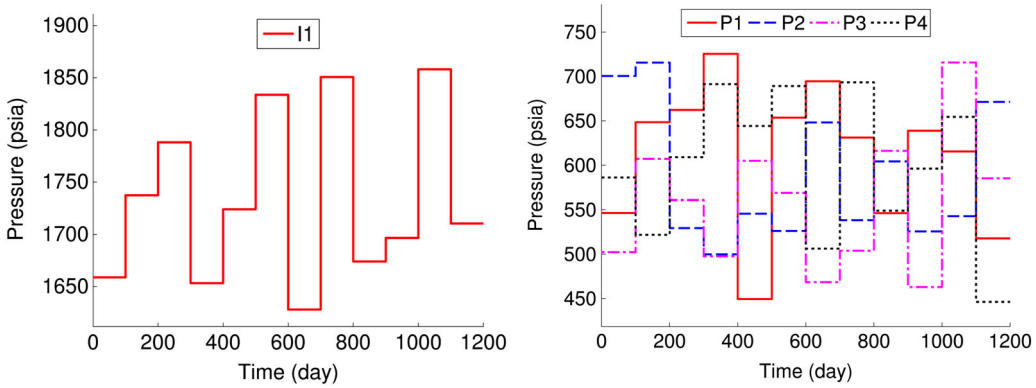
Figure 15 shows the stability maps for the Galerkin and Petrov–Galerkin methods for this case. Note that the scales of the color bars for the two figures are very different. For Petrov–Galerkin projection, all (l_p, l_z) combinations lead to models that behave stably. For Galerkin projection, for all (l_p, l_z) combinations considered, $\log_{10}(\max_i \gamma^i)$ is always much larger than 0 (the lowest $\max_i \gamma^i$ value being about 27). This means that for any of these (l_p, l_z) combinations, the resulting POD–TPWL model will have one or more unstable steps. Therefore, for this case, we cannot apply

DOI: 10.1002/nme



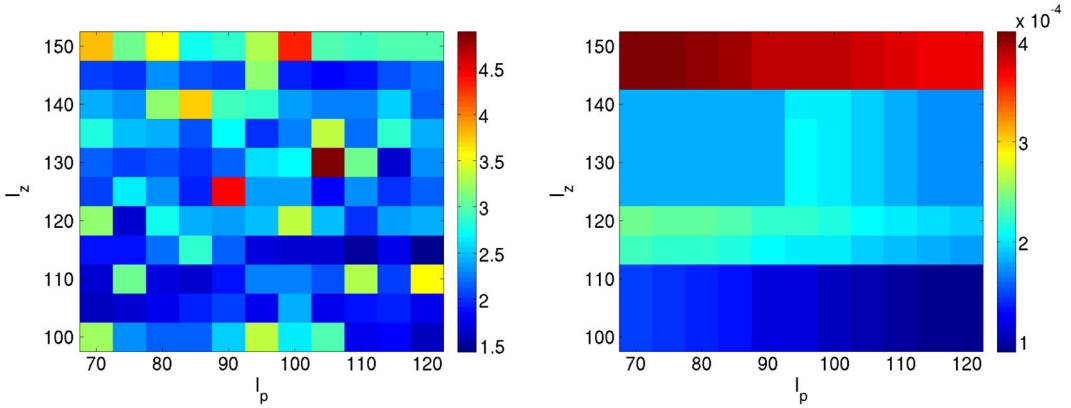
(a) Injector BHPs (b) Producer BHPs

Figure 13. Time-varying bottom-hole pressures (BHPs) for the primary training simulation for case 3. (a) Injector BHPs. (b) Producer BHPs.



(a) Injector BHPs (b) Producer BHPs

Figure 14. Time-varying bottom-hole pressures (BHPs) for the test simulation for case 3. (a) Injector BHPs. (b) Producer BHPs.



(a) Galerkin projection (minimum value on color bar: 1.43)

(b) Petrov-Galerkin projection (minimum value on color bar: 8.9×10^{-5})

Figure 15. Maps of $\log_{10}(\max_i \gamma^i)$ for case 3. (a) Galerkin projection (minimum value on color bar: 1.43). (b) Petrov–Galerkin projection (minimum value on color bar: 8.9×10^{-5}).

DOI: 10.1002/nme

the basis selection procedure used for case 2 to construct a stably behaving POD–TPWL model using Galerkin projection.

In addition, although not shown here, for all (l_p, l_z) combinations considered, the Galerkin method displays $\log_{10}(\max_i \gamma^i) > 0$ for all time steps. The reason that compositional systems are less stable than oil–water systems with Galerkin projection is not entirely clear, but it may be due, at least in part, to the large density difference between the oil and gas phases (which leads to a strong gravity-driven countercurrent flow) and to the high nonlinearity resulting from complex phase behavior. This issue should be investigated in future work.

Figures 16 and 17 show the oil and gas production rates for producer 1, while Figure 18 displays the gas injection rate for injector 1. Test-case results using Petrov–Galerkin projection are shown along with results using WIP, which will be discussed in the next section. Only results for the first 600 days of the simulation are shown. This allows us to focus on the period where gas production rates are increasing and the most error occurs. The POD–TPWL model exhibits reasonable overall accuracy, although errors are evident for different quantities at different times. For example, the oil production rate in Figure 16 displays some inaccuracy from 300 to 400 days, as do gas injection rates (Figure 18) over the first 200 days. The solution, however, clearly behaves stably, and the results are of sufficient accuracy to be useful for many applications. These results are representative of those for other wells. Runtime speedup for this case is around a factor of 150, which is lower than that for cases 1 and 2. The lower speedup here results from the flash calculation, which is

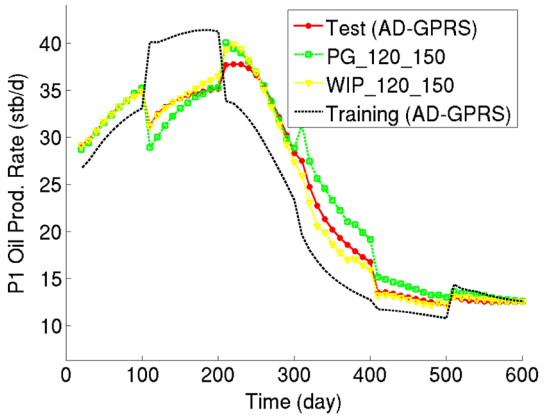


Figure 16. Oil production rate for producer 1 in case 3. AD-GPRS, Automatic Differentiation-based General Purpose Research Simulator; PG, Petrov–Galerkin projection; WIP, weighted inverse projection.

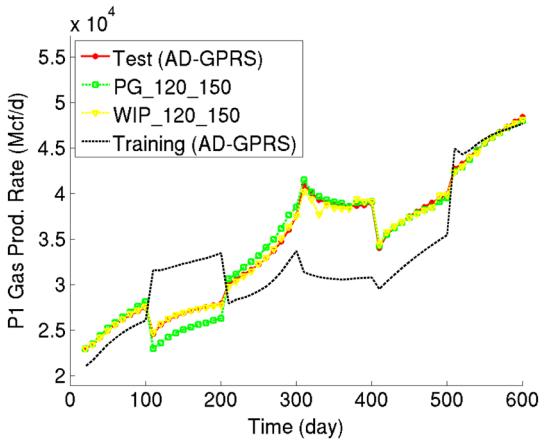


Figure 17. Gas production rate for producer 1 in case 3. AD-GPRS, Automatic Differentiation-based General Purpose Research Simulator; PG, Petrov–Galerkin projection; WIP, weighted inverse projection.

DOI: 10.1002/nme

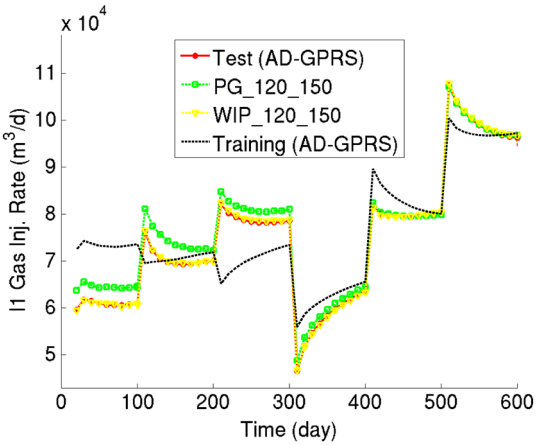


Figure 18. Gas injection rate for injector 1 in case 3. AD-GPRS, Automatic Differentiation-based General Purpose Research Simulator; PG, Petrov–Galerkin projection; WIP, weighted inverse projection.

not handled efficiently in our current POD–TPWL model. This cost can, however, be significantly reduced through an efficient implementation.

7. IP AND WIP CONSTRAINT REDUCTION METHODS

In Section 4, we showed that, to minimize constraint reduction error $\mathbf{e}_{23}$, optimal constraint reduction methods should be of the form $\boldsymbol{\Psi}^T = \boldsymbol{\Phi}^T\boldsymbol{\Theta}\mathbf{J}^{-1}$. We use $\boldsymbol{\Theta} = \mathbf{J}$ in the case of Galerkin projection and $\boldsymbol{\Theta} = \mathbf{J}^T\mathbf{J}$ in the case of Petrov–Galerkin projection. Both choices eliminate the $\mathbf{J}^{-1}$ term and are thus very efficient to compute. Other choices for $\boldsymbol{\Theta}$ also exist, however. Although they typically entail higher computational costs, they may have theoretical advantages and display superior performance. In this section, we consider two such projection methods, IP and WIP.

7.1. Method development

An intuitive choice for $\boldsymbol{\Theta}$ is the identity matrix. Then the optimal projection matrix is given by $\boldsymbol{\Psi}^T = \boldsymbol{\Phi}^T\mathbf{J}^{-1}$. We refer to this approach as the IP method.

An important property of the IP method is that it minimizes the 2-norm of the constraint reduction error $\mathbf{e}_{23}$. That is, it satisfies

$$\boldsymbol{\Psi}^* = \arg\min_{\boldsymbol{\Psi}} \|\mathbf{e}_{23}\|_2^2 . \tag{39}$$

Compared with the norms appearing in the optimality conditions for the Galerkin and Petrov–Galerkin methods, the 2-norm appearing here is preferable because it does not depend on the specific structure of $\mathbf{J}$.

In many reservoir simulation applications, such as production optimization, we are particularly interested in the states in well blocks because these values directly affect injection and production rates. We can assign larger weights to the elements of the error vector corresponding to these blocks in Equation (21) by defining $\boldsymbol{\Theta} = \mathbf{W}^T\mathbf{W}$, where $\mathbf{W}$ is a diagonal weighting matrix. The resulting projection matrix is $\boldsymbol{\Psi}^T = \boldsymbol{\Phi}^T\mathbf{W}^T\mathbf{W}\mathbf{J}^{-1}$. We refer to this method as WIP. WIP degenerates to IP when $\mathbf{W} = \mathbf{I}$. WIP minimizes the weighted error vector in the 2-norm,

$$\boldsymbol{\Psi}^* = \arg\min_{\boldsymbol{\Psi}} \|\mathbf{W}\mathbf{e}_{23}\|_2^2 . \tag{40}$$

In the examples in this work involving WIP, we assign a value of 5 to the diagonal elements of $\mathbf{W}$ corresponding to well block states and 1 to the remaining diagonal elements. This 5-to-1 ratio

DOI: 10.1002/nme

was determined through limited numerical experimentation. The values of $\mathbf{W}$ can be further tuned to improve accuracy in particular grid blocks.

The IP and WIP methods offer theoretical advantages by virtue of their optimality conditions. Achieving this benefit, however, requires additional computational effort. This is because, for both the IP and WIP methods, the $\mathbf{J}^{-1}$ term in the expression for $\mathbf{\Psi}$ is not multiplied by $\mathbf{J}$, so it does not cancel as in the Galerkin and Petrov–Galerkin procedures.

In practice, $\mathbf{J}^{-1}$ does not need to be calculated explicitly. The matrix $\mathbf{\Psi}$ only appears in the calculation of the reduced derivatives in Equation (13). Substituting Equation (27) into Equation (13) (and dropping the superscript $i+1$), we have

$$\mathbf{J}_r = \mathbf{\Phi}^T \mathbf{\Theta} \mathbf{\Phi}, \quad \mathbf{A}_r = \mathbf{\Phi}^T \mathbf{\Theta} \mathbf{J}^{-1} \mathbf{A} \mathbf{\Phi}, \quad \mathbf{B}_r = \mathbf{\Phi}^T \mathbf{\Theta} \mathbf{J}^{-1} \mathbf{B}. \tag{41}$$

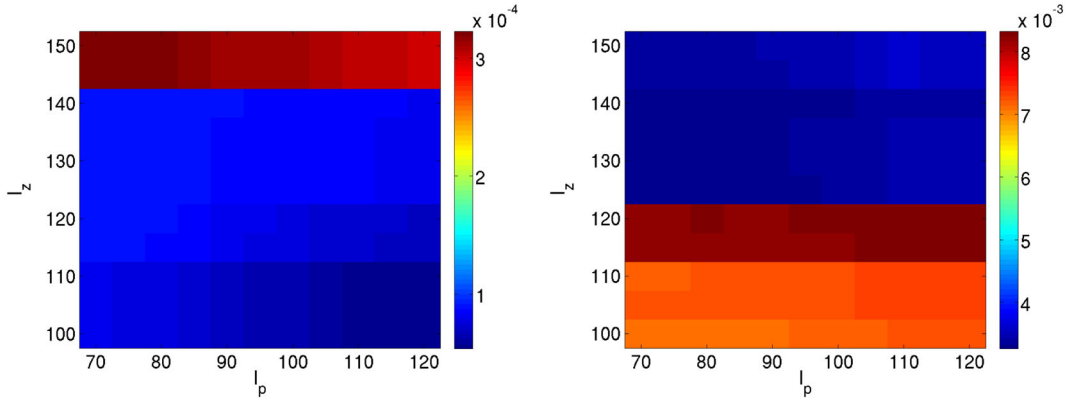
Interestingly, the calculation of $\mathbf{J}_r$ now does not involve $\mathbf{J}$. The term $\mathbf{J}^{-1}\mathbf{A}\mathbf{\Phi}$ in matrix $\mathbf{A}_r$ can be calculated by solving $\mathbf{J}\mathbf{x}_A = \mathbf{A}\mathbf{\Phi}$, which is an $n_v \times n_v$ system with l right-hand sides, where l is the total number of reduced variables. Similarly, the term $\mathbf{J}^{-1}\mathbf{B}$ in matrix $\mathbf{B}_r$ can be calculated by solving $\mathbf{J}\mathbf{x}_B = \mathbf{B}$, which is an $n_v \times n_v$ system with n_u right-hand sides, where n_u is the dimension of the control parameter $\mathbf{u}$. In total, we will thus need to solve the high-dimensional linear system with $l + n_u$ right-hand sides at each time step. For a case with $l = 200$ and $n_u = 10$, the cost of constructing $\mathbf{A}_r$ and $\mathbf{B}_r$ for the IP or WIP method is the equivalent of about 10–20 full-order simulations for a compositional problem and 20–40 full-order simulations for an oil–water problem (this difference arises because full-order compositional simulations usually require more Newton iterations and time-step cuts than oil–water problems). Preprocessing for IP and WIP is thus much more expensive than for Galerkin and Petrov–Galerkin methods (although runtimes, and thus runtime speedup, are comparable). However, in applications where the POD–TPWL model can be used in place of hundreds of full-order runs, the IP and WIP methods may be viable options.

We note finally that the computational cost of IP and WIP may be reduced significantly by approximating $\mathbf{J}^{-1}$ with another matrix $\mathbf{Q}$, so that we can use $\mathbf{Q}$ in place of $\mathbf{J}^{-1}$. Potential choices for $\mathbf{Q}$ are the various preconditioners used for solving $\mathbf{J}\mathbf{x} = \mathbf{b}$. This should be considered in future work.

7.2. *Numerical results using IP and WIP*

The IP and WIP methods will now be applied to cases 1–3. The problem setups are identical to those described earlier.

In terms of stability, the IP and WIP methods behave stably for all three cases, and their stability characteristics are not sensitive to the choice of l_p and l_S (or l_z). As an example, Figure 19 shows the stability maps for the IP and WIP methods for case 3, which was the most challenging case for



(a) IP method (minimum value on color bar: 5.4×10^{-5}) (b) WIP method (minimum value on color bar: 3.3×10^{-3})

Figure 19. Maps of $\log_{10}(\max_i \gamma^i)$ for inverse projection (IP) and weighted inverse projection (WIP) for case 3. (a) IP method (minimum value on color bar: 5.4×10^{-5}). (b) WIP method (minimum value on color bar: 3.3×10^{-3}).

DOI: 10.1002/nme

the Galerkin method. It can be seen that for both methods, $\log_{10}(\max_i \gamma^i)$ is very close to zero for all combinations of l_p and l_z considered.

To assess their accuracy, results using IP and WIP are now compared with those using the Galerkin and Petrov–Galerkin methods for cases 1–3. Table III summarizes the errors for case 1. In this case, the injection and production rate results using IP are less accurate than those using the Galerkin method. However, the IP method provides better results in terms of the average state error for both pressure and water saturation. This is because the IP method is, by design, optimal in minimizing the constraint reduction error globally, rather than at the well blocks. WIP, which places additional weight on the well blocks, improves upon the IP injection and production rate results (especially for E_{iw}) while maintaining state errors similar to those of the IP method.

Tables IV and V present the errors for cases 2 and 3. For case 2, the IP method provides more accurate results than the Galerkin and Petrov–Galerkin methods in all five metrics. The WIP method provides the most accurate well rate predictions among all methods tested and leads to average state errors that are again comparable with those of the IP method. For case 3, the IP and WIP methods continue to provide more accurate results than those obtained using Petrov–Galerkin projection.

Figures 16–18 display the performance of the WIP method relative to the Petrov–Galerkin method for case 3. It is clear that WIP leads to improved results compared with Petrov–Galerkin projection, most notably in oil production rate from 300 to 400 days, in gas production rate from 100 to 200 days, and in gas injection rate before 300 days.

The results in this section indicate that the use of (weighted) IP for constraint reduction leads to POD–TPWL models that are more accurate than those generated using Galerkin or Petrov–Galerkin projection. In addition, in common with Petrov–Galerkin projection, IP and WIP provide POD–TPWL models that behave stably. The IP and WIP methods, as currently implemented, require much more preprocessing than the Petrov–Galerkin procedure, so the overall level of speedup they

Table III. Summary of error for case 1, with IP and WIP.

	E_o	E_w	E_{iw}	E_p (psi)	E_S
GLK_70_100	0.0054	0.0058	0.0015	1.57	0.00066
PG_70_100	0.0113	0.0156	0.0052	7.07	0.00130
IP_70_100	0.0059	0.0064	0.0044	0.62	0.00057
WIP_70_100	0.0050	0.0062	0.0013	0.58	0.00060

GLK, Galerkin projection; IP, inverse projection; PG, Petrov–Galerkin projection; WIP, weighted inverse projection.

Table IV. Summary of error for case 2, with IP and WIP.

	E_o	E_w	E_{iw}	E_p (psi)	E_S
GLK_75_80	0.0216	0.0152	0.0050	13.6	0.00212
PG_75_80	0.0225	0.0255	0.0112	15.0	0.00204
IP_75_80	0.0058	0.0077	0.0041	1.43	0.00084
WIP_75_80	0.0055	0.0076	0.0013	0.759	0.00085

GLK, Galerkin projection; IP, inverse projection; PG, Petrov–Galerkin projection; WIP, weighted inverse projection.

Table V. Summary of error for case 3, with IP and WIP.

	E_o	E_g	E_{ig}	E_p (psi)	E_z
PG_120_150	0.0400	0.0160	0.0187	5.68	0.00894
IP_120_150	0.0239	0.0103	0.0081	1.50	0.00393
WIP_120_150	0.0219	0.0085	0.0077	1.41	0.00395

IP, inverse projection; PG, Petrov–Galerkin projection; WIP, weighted inverse projection.

DOI: 10.1002/nme

provide is limited. Future work should target the fast construction of the reduced matrices $\mathbf{A}_r$ and $\mathbf{B}_r$. Progress in this direction would enable the efficient use of IP and WIP for large models.

8. CONCLUDING REMARKS

In this paper, we assessed the various errors that arise in POD–TPWL models. These errors derive from state reduction, nonlinearity treatment (accomplished here using piecewise linearization), and constraint reduction. Our focus was on constraint reduction, which is applied to project the overdetermined high-dimensional system of equations into a low-dimensional subspace. Once constraint reduction and state reduction (accomplished here through the use of POD) are applied, the low-dimensional set of equations is fully determined. The choice of constraint reduction procedure was shown to affect POD–TPWL error and stability behavior. In previous work on the use of POD–TPWL for subsurface flow, Galerkin projection has mainly been applied, although recent work has used Petrov–Galerkin projection for compositional reservoir simulation [25]. Here, we showed that both the Galerkin and Petrov–Galerkin projection methods can be derived from the optimality condition of the constraint reduction error, although the Galerkin method does not satisfy the optimality condition when the Jacobian matrix is not SPD (which it is not in oil–water or compositional flow problems).

The performance of the Galerkin and Petrov–Galerkin methods was compared for two oil–water cases and one oil–gas compositional case. For oil–water systems with gravity-driven countercurrent flow, it was observed that the stability behavior of Galerkin projection is sensitive to the number of reduced variables (l_p, l_S) used. In these cases, it may however be possible to find an (l_p, l_S) combination for which Galerkin projection behaves stably. For compositional simulation, the Galerkin method was unstable for any of the (l_p, l_z) combinations tested. By contrast, the Petrov–Galerkin method was shown to behave stably for all cases considered (both here and in [25]). In addition, its stability characteristics were not found to be sensitive to the number of reduced variables used. Therefore, for complex reservoir simulation problems, Petrov–Galerkin projection appears to be more reliable compared with Galerkin projection. This observation is consistent with the findings in [17], where Galerkin and Petrov–Galerkin projections were compared for ROMs for turbulent flow and nonlinear structural dynamics problems. We note, however, that theoretical guarantees regarding the stability of Petrov–Galerkin projection for nonlinear problems do not, to our knowledge, exist.

There are other methods that satisfy the optimality conditions for constraint reduction, and two such procedures were considered. IP minimizes the 2-norm of the constraint reduction error, while WIP minimizes the 2-norm of the error vector with extra weight at selected blocks. The WIP method is by design optimal for well flow rate calculations (when well blocks are weighted more heavily), and it was shown to provide the best overall accuracy among all of the constraint reduction methods considered. However, because they involve the inverse Jacobian matrix $\mathbf{J}^{-1}$, the IP and WIP methods incur high computational costs for the construction of the POD–TPWL model. Specifically, preprocessing (overhead) costs for these methods are equivalent to about 10–40 full-order simulations. Improving the efficiency of these methods should be a topic for future research.

Although our specific focus in this paper was on subsurface flow, a variety of problems are governed by conservation laws similar to Equation (1), so many of our detailed findings should be more broadly relevant. In addition, because most aspects of the development presented in this paper are not specific to a particular application, we expect our general approaches and findings to be applicable to a range of problems modeled using POD–TPWL. More specifically, the step-by-step error assessment, optimality results for the various constraint reduction treatments, stability analysis, and the procedure for low-dimensional stability map construction should all be applicable for POD–TPWL methods in general. Our findings regarding the stability advantages of Petrov–Galerkin projection relative to Galerkin projection should hold for other problems in which the Jacobian matrix is not SPD. We also believe that the IP and WIP approaches for constraint reduction may provide higher accuracy for a range of applications (although they require efficiency improvements, as noted earlier). A few treatments are, however, application specific. These include the selection of the number of training runs and the controls applied in these runs, the number of snapshots required,

DOI: 10.1002/nme

the detailed construction of the basis matrix $\mathbf{\Phi}$, and the specific definition of 'distance' used to determine the nearest saved state.

Future work should consider the development of constraint reduction methods that are optimally accurate and guaranteed to be stable by, for example, combining the optimality condition presented in this work with the Lyapunov stability equation. The quantification and reduction of the other errors that arise in POD–TPWL models should also be addressed. The linearized treatment could conceivably be improved by incorporating (or estimating) higher-order corrections at selected locations and times. For subsurface flow applications, it will additionally be of interest to consider cases with more wells (e.g., 10–100). With larger numbers of wells, there is more variability in the states that can occur in the model. This suggests that more snapshots, from more training runs, will be needed to represent the system. The development of techniques for the systematic design of training runs may enable the POD–TPWL model to provide sufficient accuracy, with reasonable overhead, for practical cases.

ACKNOWLEDGEMENTS

We thank the U.S. Department of Energy National Energy Technology Laboratory (award DE-FE0009051, administered through Battelle Memorial Institute), Stanford University (through a Stanford Graduate Fellowship), and the industrial affiliates of the Stanford Smart Fields Consortium, for funding this work. We are grateful to Hamdi Tchelepi for useful discussions and suggestions.

REFERENCES

1. Sirovich L. Analysis of turbulent flows by means of the empirical eigenfunctions. *Fluid Dynamics Research* 1991; **8**(1–4):85–100.
2. Bui-Thanh T, Damodaran M, Willcox K. Aerodynamic data reconstruction and inverse design using proper orthogonal decomposition. *AIAA Journal* 2004; **42**(8):1505–1516.
3. Vermeulen PTM, Heemink AW, Stroet CBMT. Reduced models for linear groundwater flow models using empirical orthogonal functions. *Advances in Water Resources* 2004; **27**:57–69.
4. Cai L, White RE. Reduction of model order based on proper orthogonal decomposition for lithium-ion battery simulations. *Journal of the Electrochemical Society* 2009; **156**(3):A154–A161.
5. Liberge E, Hamdouni A. Reduced order modelling method via proper orthogonal decomposition (POD) for flow around an oscillating cylinder. *Journal of Fluids and Structures* 2010; **26**(2):292–311.
6. van Doren JFM, Markovinović R, Jansen JD. Reduced-order optimal control of water flooding using proper orthogonal decomposition. *Computational Geosciences* 2006; **10**:137–158.
7. Cardoso MA, Durlofsky LJ, Sarma P. Development and application of reduced-order modeling procedures for subsurface flow simulation. *International Journal for Numerical Methods in Engineering* 2009; **77**(9):1322–1350.
8. Moore B. Principal component analysis in linear systems: Controllability, observability, and model reduction. *IEEE Transactions on Automatic Control* 1981; **26**:17–31.
9. Heijn T, Markovinović R, Jansen JD. Generation of low-order reservoir models using system-theoretical concepts. *SPE Journal* 2004; **9**(2):202–218.
10. Condon M, Ivanov R. Empirical balanced truncation of nonlinear systems. *Journal of Nonlinear Science* 2004; **14**(5):405–414.
11. Gallivan K, Grimme E, Dooren PV. A rational Lanczos algorithm for model reduction. *Numerical Algorithms* 1996; **12**(1):33–63.
12. Vasilyev D, Rewienski M, White J. A TBR-based trajectory piecewise-linear algorithm for generating accurate low-order models for nonlinear analog circuits and MEMS. *Dac '03: Proceedings of the 40th Conference on Design Automation*, Anaheim, California, USA, 2003; 490–495.
13. Yang YJ, Shen KY. Nonlinear heat-transfer macromodeling for MEMS thermal devices. *Journal of Micromechanics and Microengineering* 2005; **15**(2):408–418.
14. Vasilyev D, Rewienski M, White J. Macromodel generation for BioMEMS components using a stabilized balanced truncation plus trajectory piecewise-linear approach. *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems* 2006; **25**(2):285–293.
15. Chaturantabut S, Sorensen DC. Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal on Scientific Computing* 2010; **32**(5):2737–2764.
16. Chaturantabut S, Sorensen DC. Application of POD and DEIM to dimension reduction of nonlinear miscible viscous fingering in porous media. *Mathematical and Computer Modelling of Dynamical Systems* 2011; **17**(4):337–353.
17. Carlberg K, Bou-Mosleh C, Farhat C. Efficient non-linear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations. *International Journal for Numerical Methods in Engineering* 2011; **86**(2):155–181.

DOI: 10.1002/nme

18. Kellems A, Chaturantabut S, Sorensen DC, Cox SJ. Morphologically accurate reduced order modeling of spiking neurons. *Journal of Computational Neuroscience* 2010; **28**:477–494.
19. Gildin E, Ghasemi M, Romanovskay A, Efendiev Y. Nonlinear complexity reduction for fast simulation of flow in heterogeneous porous media (SPE paper 163618). *SPE Reservoir Simulation Symposium*, The Woodlands, Texas, USA, February 2013.
20. Rewienski M, White J. A trajectory piecewise-linear approach to model order reduction and fast simulation of nonlinear circuits and micromachined devices. *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems* 2003; **22**(2):155–170.
21. Cardoso MA, Durlofsky LJ. Linearized reduced-order models for subsurface flow simulation. *Journal of Computational Physics* 2010; **229**(3):681–700.
22. Cardoso MA, Durlofsky LJ. Use of reduced-order modeling procedures for production optimization. *SPE Journal* 2010; **15**(2):426–435.
23. He J, Sætrom J, Durlofsky LJ. Enhanced linearized reduced-order models for subsurface flow simulation. *Journal of Computational Physics* 2011; **230**:8313–8341.
24. Rousset M AH, Huang CK, Klie H, Durlofsky LJ. Reduced-order modeling for thermal recovery processes. *Computational Geosciences* 2014; **18**:401–415.
25. He J, Durlofsky LJ. Reduced-order modeling for compositional simulation by use of trajectory piecewise linearization. *SPE Journal* 2014; **19**(5):858 –872.
26. He J, Sarma P, Durlofsky LJ. Reduced-order flow modeling and geological parameterization for ensemble-based data assimilation. *Computers & Geosciences* 2013; **55**:54–69.
27. Lall S, Marsden JE, Glavaški S. A subspace approach to balanced truncation for model reduction of nonlinear control systems. *International Journal of Robust and Nonlinear Control* 2002; **12**(6):519–535.
28. Willcox K, Peraire J. Balanced model reduction via the proper orthogonal decomposition. *AIAA journal* 2002; **40**(11):2323–2330.
29. He J. Reduced-order modeling for oil–water and compositional systems, with application to data assimilation and production optimization. *Ph.D. Thesis*, 2013.
30. Berkooz G, Titi EZ. Galerkin projections and the proper orthogonal decomposition for equivariant equations. *Physics Letters A* 1993; **174**(1–2):94–102.
31. Bond BN, Daniel L. Guaranteed stable projection-based model reduction for indefinite and unstable linear systems. *Proceedings of the IEEE/ACM International Conference on Computer-aided Design*, San Jose, California, USA, 2008; 728–735.
32. Bui-Thanh T, Willcox K, Ghattas O. Model reduction for large-scale systems with high-dimensional parametric input space. *SIAM Journal on Scientific Computing* 2008; **30**(6):3270–3288.
33. Aziz K, Settari A. *Petroleum Reservoir Simulation*. Blitzprint, Ltd.: Calgary, Canada, 2002.
34. Coats KH. An equation of state compositional model. *SPE Journal* 1980; **20**(5):363–376.
35. Young LC, Stephenson RE. A generalized compositional approach for reservoir simulation. *SPE Journal* 1983; **23**(5):727–742.
36. Gerritsen MG, Durlofsky LJ. Modeling fluid flow in oil reservoirs. *Annual Review of Fluid Mechanics* 2005; **37**: 211–238.
37. Voskov DV, Tchelepi HA. Comparison of nonlinear formulations for two-phase multi-component EoS based simulation. *Journal of Petroleum Science and Engineering* 2012; **82–83**:101–111.
38. Rathinam M, Petzold LR. A new look at proper orthogonal decomposition. *SIAM Journal on Numerical Analysis* 2003; **41**(5):1893–1925.
39. Chaturantabut S, Sorensen DC. A state space error estimate for POD-DEIM nonlinear model reduction. *SIAM Journal on Numerical Analysis* 2012; **50**(1):46–63.
40. Chen Y, White J. A quadratic method for nonlinear model order reduction. *Proceedings of the International Conference on Modeling and Simulation of Microsystems*, San Diego, California, 2000; 477–480.
41. Zhou Y. Parallel general-purpose reservoir simulation with coupled reservoir models and multi-segment wells. *Ph.D. Thesis*, 2012.
42. Cheney EW, Kincaid DR. *Numerical Mathematics and Computing*. Cengage Learning: Boston, Massachusetts, 2012.
43. Markovinović R, Jansen JD. Accelerating iterative solution methods using reduced-order models as solution predictors. *International Journal for Numerical Methods in Engineering* 2006; **68**(5):525–541.
44. Ravindran SS. A reduced-order approach for optimal control of fluids using proper orthogonal decomposition. *International Journal for Numerical Methods in Fluids* 2000; **34**(5):425–448.
45. Cao Y, Zhu J, Navon IM, Luo Z. A reduced-order approach to four-dimensional variational data assimilation using proper orthogonal decomposition. *International Journal for Numerical Methods in Fluids* 2007; **53**(10):1571–1583.
46. Gelfand I. Zur theorie der charactere der abelschen topologischen gruppen. *Matematicheskii Sbornik* 1941; **9**(51): 49–50.
47. Bond BN, Daniel L. Stable reduced models for nonlinear descriptor systems through piecewise-linear approximation and projection. *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems* 2009; **28**(10): 1467–1480.
48. Castro SA. A probabilistic approach to jointly integrate 3D/4D seismic, production data and geological information for building reservoir models. *Ph.D. Thesis*, 2007.
49. Remy N, Boucher A, Wu J. *Applied Geostatistics with SGeMS: A User's Guide*. Cambridge University Press: Cambridge, England, 2008.

DOI: 10.1002/nme