Journal of Computational Physics 229 (2010) 681–700

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Linearized reduced-order models for subsurface flow simulation

M.A. Cardoso *,[1], L.J. Durlofsky

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

ARTICLE INFO

Article history:
Received 18 December 2008
Received in revised form 16 July 2009
Accepted 3 October 2009
Available online 17 October 2009

Keywords:
Proper orthogonal decomposition (POD)
Trajectory piecewise linearization (TPWL)
Reservoir simulation
Subsurface flow
Optimization
Model order reduction
Reduced-order model
Two-phase flow

ABSTRACT

A trajectory piecewise linearization (TPWL) procedure for the reduced-order modeling of two-phase flow in subsurface formations is developed and applied. The method represents new pressure and saturation states using linear expansions around states previously simulated and saved during a series of preprocessing training runs. The linearized representation is projected into a low-dimensional space, with the projection matrix constructed through proper orthogonal decomposition of the states determined during the training runs. The TPWL model is applied to two example problems, containing 24,000 and 79,200 grid blocks, which are characterized by heterogeneous permeability descriptions. Extensive test simulations are performed for both models. It is shown that the TPWL model provides accurate results when the controls (bottom hole pressures of the production wells in this case) applied in test simulations are within the general range of the controls applied in the training runs, even though the well pressure schedules for the test runs can differ significantly from those of the training runs. This indicates that the TPWL model displays a reasonable degree of robustness. Runtime speedups using the procedure are very significant–a factor of 100–2000 (depending on model size and whether or not mass balance error is computed at every time step) for the cases considered. The preprocessing overhead required by the TPWL procedure is the equivalent of about four high-fidelity simulations. Finally, the TPWL procedure is applied to a computationally demanding multiobjective optimization problem, for which the Pareto front is determined. Limited high-fidelity simulations demonstrate the accuracy and applicability of TPWL for this optimization. Future work should focus on error estimation and on stabilizing the method for large models with significant density differences between phases.

1. Introduction

The development of reduced-order modeling procedures has received significant attention in recent years. Reduced-order models (ROMs) are particularly well suited for use in minimization procedures, where the forward model must be run many times (e.g., hundreds or thousands) for the determination of optimal design or operating parameters. Our particular interest here is in subsurface flow problems, specifically oil reservoir simulation. Other closely related applications include aquifer management and the geological storage of carbon dioxide.

A number of the previous ROM procedures developed for subsurface flow applications are based on the use of proper orthogonal decomposition (POD). This entails performing one or more high-fidelity training simulations, saving snapshots (state vectors) at a number of time steps from these simulations, and then constructing a set of basis functions from these snapshots. The basis functions are generated through a singular value decomposition of the snapshot matrix, which is analogous to

* Corresponding author.
E-mail addresses: marcocardoso@petrobras.com.br (M.A. Cardoso), lou@stanford.edu (L.J. Durlofsky).
[1] Now at Petróleo Brasileiro S.A., Petrobras.

0021-9991/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2009.10.004

an eigen-decomposition of the covariance matrix formed from the snapshots. Efficiency is achieved because only relatively few basis functions must be retained; i.e., flow solutions can be represented in terms of just the leading basis vectors.

POD procedures were first introduced by Lumley [1] to identify coherent structures in dynamical systems and have subsequently been applied in a number of application areas (e.g., [2–7]). Vermeulen et al. [8] appear to be the first to have applied POD techniques within the context of subsurface flow problems. They achieved substantial speedups for groundwater flow involving a single (water) component. In this case, however, the governing equation is linear (or nearly linear). Jansen and coworkers [9,10] applied POD techniques to two-phase (oil–water) flow. Here, due to differing phase viscosities and phase interference (relative permeability) effects, the problem is nonlinear and the reported speedups were much more modest; e.g., a factor of 3 or less.

In recent work targeting oil–water reservoir simulation problems, we incorporated new procedures designed to enhance the basic POD approach (Cardoso et al. [11]). These included a snapshot clustering procedure [12] and a missing point estimation (MPE) technique [13]. MPE acts to eliminate rows from the basis matrix and thus accelerates the requisite matrix–matrix multiplications. We incorporated the POD techniques into a general purpose research simulator and applied them to geologically realistic models containing 60,000 grid blocks. Speedups of up to about a factor of 10 were achieved.

There are however several limitations that impact the degree of speedup attainable from standard POD procedures for this nonlinear problem. Consider a system containing N_c grid blocks (which corresponds to $2N_c$ unknowns for the oil–water problem) and suppose that this system can be represented using ℓ basis functions. For specificity, in an example in our earlier study [11], $N_c = 60{,}000$ and $\ell = 40$. The standard POD procedure works at the level of the linear solver, reducing the sparse $2N_c \times 2N_c$ linear system to a full $\ell \times \ell$ system. Solver time is thus reduced substantially. Computational requirements for other operations, by contrast, such as the construction of the Jacobian matrix (which must be performed at each iteration of every time step) are not reduced at all through the use of the standard POD technique. The application of MPE procedures does eliminate the need for some rows of the Jacobian matrix (and thus they need not be constructed), but the required number of rows in the Jacobian would still be expected to be a reasonable fraction of N_c. In addition, a full implementation of MPE requires modifications throughout the simulator. This may be complicated if the MPE procedure is incorporated into a comprehensive general purpose simulator.

We also observed that standard POD procedures are less robust and/or require more basis functions as the problem becomes more nonlinear, as occurs with strong gravitational effects and with highly nonlinear relative permeability functions [11]. In these cases, the speedups offered by standard POD techniques will be quite modest at best.

Recent linearization procedures, specifically the trajectory piecewise linearization (TPWL) introduced by Rewienski and White [14,15], appear to offer a means for addressing some of the limitations in standard POD techniques highlighted above. The TPWL approach combines reduced-order modeling with linearization of the governing equations. For the implementation described in this paper, TPWL proceeds as follows. First, two training simulations are performed, from which the states and converged Jacobian matrices are saved. A basis matrix is then constructed from the states using proper orthogonal decomposition, and this basis is used to form reduced states and reduced Jacobian matrices. In subsequent simulations, new states are represented in terms of linear expansions around previously simulated (and saved) states and Jacobians. Very high degrees of computational efficiency are achieved because all of these computations are performed in reduced space. TPWL has been successfully applied in a number of application areas, including computational fluid dynamics [16], heat-transfer modeling [17], biomedical micro-electromechanical systems (BioMEMS) [18], electronic circuits [19,20], and moving electromagnetic devices [21]. Stabilized TPWL methods have also been recently presented by, e.g., [22,23]. TPWL procedures do not, however, appear to have been considered for oil reservoir simulation or for any closely related subsurface flow applications.

Our goal in this paper is to develop and apply TPWL procedures for subsurface flow problems. Toward this end we formulate a TPWL representation for oil–water flow and present extensive results for challenging subsurface flow problems. Our examples involve highly heterogeneous geological models and include different phase densities (in one case) and nonlinear relative permeabilities. Numerical results demonstrate that, for test simulations that are within the general range of the training runs, the TPWL procedure can provide predictions in close agreement with high-fidelity simulations with runtime speedups of a factor of 100–2000. The computational overhead required to construct the TPWL model is the equivalent of about four high-fidelity simulations.

This paper proceeds as follows. We first present the governing equations for oil–water flow and develop the linearized representation. POD procedures and their use in conjunction with the linearized oil–water model are then described. Next, some implementation issues are discussed and the overall TPWL algorithms are provided. We then present simulation results for two cases, containing 24,000 and 79,200 grid blocks. Accurate results for oil and water production and water injection rates are achieved for both cases for a wide variety of well control schedules. We additionally apply the TPWL representation for a multiobjective optimization problem and demonstrate close correspondence with selected high-fidelity computations. Finally, additional issues and outstanding challenges are discussed and concluding remarks are presented.

2. Trajectory piecewise linearization (TPWL) for subsurface flow

In this section we present the equations for two-phase subsurface flow and briefly describe the basic finite volume discretization. The TPWL procedure is then developed in detail. This includes a short description of the construction of the POD basis matrix, which is required for the TPWL representation. We note that there are several heuristic components in our TPWL formulation, which are discussed in the development that follows.

2.1. Oil–water flow equations and discretization

Subsurface flow models are derived by combining mass conservation equations with the multiphase version of Darcy's law. For the oil–water case considered here, there is no mass transfer between phases; i.e., the oil component resides only in the oil phase and the water component only in the water phase. Then, for each component/phase j (with $j = o$ denoting oil and $j = w$ water), we have:

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

where $\mathbf{k}$ is the absolute permeability (assumed to be a diagonal tensor), $\lambda_j = k_{rj}/(\mu_j B_j)$ is the phase mobility, k_{rj} is the relative permeability to phase j, μ_j is the phase viscosity, B_j is the formation volume factor for phase j (defined below), p_j is phase pressure, ρ_j is the phase density, g is gravitational acceleration, D is depth, t is time, ϕ is porosity (void fraction of the rock), S_j is saturation and q_j^w is the source/sink term. The formation volume factor is defined as the ratio of the volume of phase j at reservoir conditions to the phase volume at reference (stock tank) conditions. Designating the reference density of phase j as ρ_j^0, it follows that $B_j = \rho_j^0/\rho(p)$. In the case of incompressible flow, density does not vary with pressure and $B_j = 1$. The general two-phase flow description is completed through the saturation constraint ($S_o + S_w = 1$) and by specifying a capillary pressure relationship; i.e., $p_c(S_w) = p_o - p_w$, which relates the phase pressures.

The two-phase flow description entails four equations and four unknowns (p_o, p_w, S_o, S_w). We select p_o and S_w as primary unknowns (which we generally refer to simply as p and S). Once these are computed, p_w and S_o can be readily determined from the capillary pressure relationship and saturation constraint.

We now briefly discuss the standard finite volume representation for Eq. (1) (see [24,25] for more details). More specialized procedures; e.g., mixed finite element-finite volume [26] or multiscale techniques [27], could also be applied. We designate the spatial directions as (ξ, η, ζ) rather than the usual (x, y, z) since we will use $\mathbf{x}$ later to designate the states (pressure and saturation). Fig. 1 shows a portion of a one-dimensional grid. For simplicity, here we neglect capillary pressure (so $p_o = p_w = p$), treat horizontal flow in the ξ-direction (for which $\nabla D = 0$), and assume the grid block dimensions ($\Delta\xi$, $\Delta\eta$, $\Delta\zeta$) are constant. We consider fully implicit discretizations. For this case the discretized form for the flow terms (left hand side of Eq. (1)) is

$$\frac{\partial}{\partial \xi}\left[k\lambda_j\left(\frac{\partial p}{\partial \xi}\right)\right] \approx \left\{(T_j)_{i-1/2}^{n+1}\left[p_{i-1}^{n+1} - p_i^{n+1}\right] + (T_j)_{i+1/2}^{n+1}\left[p_{i+1}^{n+1} - p_i^{n+1}\right]\right\}\frac{1}{V}, \tag{2}$$

where subscript j indicates phase and i designates grid block, superscript $n+1$ specifies the next time step, and $V = \Delta\xi\Delta\eta\Delta\zeta$ is the volume of grid block i. The transmissibility $(T_j)_{i-1/2}^{n+1}$ relates flow in phase j to the difference in pressure between grid blocks $i-1$ and i and is given by:

$$(T_j)_{i-1/2}^{n+1} = \left(\frac{kA}{\Delta\xi}\right)_{i-1/2}\left(\frac{k_{rj}}{B_j\mu_j}\right)_{i-1/2}^{n+1}, \tag{3}$$

where $A = \Delta\eta\Delta\zeta$ is the area of the common face between blocks $i-1$ and i. The interface permeability $k_{i-1/2}$ is computed as the harmonic average of k_i and k_{i-1} and the $k_{rj}/(B_j\mu_j)$ term is upstream weighted. The transmissibility $(T_j)_{i+1/2}^{n+1}$ is defined analogously. The flow terms can be seen to introduce nonlinearity into the system as $(T_j)_{i\pm1/2}^{n+1}$ are functions of pressure and saturation and multiply terms involving pressure.

The first term on the right hand side of Eq. (1) represents mass accumulation and is represented discretely for block i as:

$$\frac{\partial}{\partial t}\left(\phi\frac{S_j}{B_j}\right) \approx \frac{1}{\Delta t}\left[\left(\phi\frac{S_j}{B_j}\right)_i^{n+1} - \left(\phi\frac{S_j}{B_j}\right)_i^{n}\right], \tag{4}$$

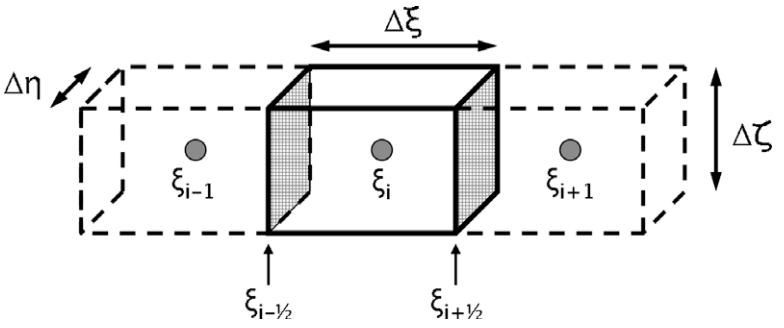

Fig. 1. Portion of a one-dimensional grid.

where Δt is the time interval. For incompressible systems ϕ is constant and $B_j = 1$, in which case Eq. (4) is given by:

$$\frac{\partial}{\partial t}\left(\phi \frac{S_j}{B_j}\right) \approx \frac{\phi_i}{\Delta t}\left[(S_j)_i^{n+1} - (S_j)_i^n\right]. \tag{5}$$

The second term on the right hand side of Eq. (1) represents the source/sink term. In reservoir simulation, the sources and sinks correspond to wells which are modeled using a well equation:

$$\left(q_j^w\right)_i^{n+1} = W_i(\lambda_j)_i^{n+1}(p_i^{n+1} - p_i^w), \tag{6}$$

where $(q_j^w)_i^{n+1}$ is the flow rate of phase j from block i into the well (or vice versa) at time $n+1$, p_i^{n+1} is grid block pressure at time $n+1$, p_i^w is the wellbore pressure for well w in grid block i, and W_i is the well index. For a vertical well that fully penetrates block i, W_i is given by [28]:

$$W_i = \left[\frac{2\pi k \Delta\zeta}{\ln(r_0/r_w)}\right]_i, \tag{7}$$

where r_w is the wellbore radius and $r_0 \approx 0.2\Delta\xi$. See [28] for expressions for W_i for more general cases. Note that, if a well is operating under bottom hole pressure (BHP) control (i.e., well pressure rather than flow rate is prescribed), this is represented in the simulation model by specifying p_i^w in Eq. (6). In three-dimensional models, BHP for a particular well is specified at the uppermost well block and p_i^w for lower blocks is computed using a $\rho g \Delta D$ correction. More general well models also exist; see [29] for discussion.

The discrete representation of the three-dimensional case is a straightforward generalization of the one-dimensional discretization presented above. The resulting nonlinear system can be expressed as follows:

$$\mathbf{g}(\mathbf{x}^{n+1}, \mathbf{u}^{n+1}) = \mathbf{F}(\mathbf{x}^{n+1}) + \mathbf{A}(\mathbf{x}^{n+1}, \mathbf{x}^n) + \mathbf{Q}(\mathbf{x}^{n+1}, \mathbf{u}^{n+1}) = 0, \tag{8}$$

where $\mathbf{g}$ represents the residual vector, which we seek to drive to zero, $\mathbf{x}$ designates the system states (pressure and saturation), $\mathbf{u}$ indicates the system controls (well BHPs), and $\mathbf{F}$, $\mathbf{A}$ and $\mathbf{Q}$ represent the convective, accumulation and source/sink terms, respectively. See [24] for detailed expressions (though note that the signs of some terms in Eq. (8) differ from those in [24]). This fully implicit system is nonlinear and is solved through application of Newton's method, with the Jacobian matrix given by $\partial\mathbf{g}/\partial\mathbf{x}$.

2.2. Linearization of governing equations

The linearized model can be developed by expanding the governing equations around a particular state $\mathbf{x}_0$ which corresponds to a set of controls $\mathbf{u}_0$. In order to express the linearized model at the level of the governing partial differential equations, it is useful to rewrite Eq. (1) as

$$\frac{\partial \mathbf{a}(\mathbf{x})}{\partial t} + \mathbf{f}(\mathbf{x}) + \mathbf{q}(\mathbf{x}, \mathbf{u}) = 0, \tag{9}$$

where $\mathbf{x} = [p, S]^T$, $\mathbf{a} = [\phi S_o/B_o, \phi S_w/B_w]^T$, $\mathbf{q} = [q_o^w, q_w^w]^T$ and

$$\mathbf{f} = -\begin{bmatrix} \nabla\cdot[\lambda_o \mathbf{k}(\nabla p - \rho_o g \nabla D)] \\ \nabla\cdot[\lambda_w \mathbf{k}(\nabla p - \rho_w g \nabla D)] \end{bmatrix}, \tag{10}$$

where we take $p_o = p_w$ (i.e., $p_c = 0$). Now, expanding around $(\mathbf{x}_0, \mathbf{u}_0)$

$$\frac{\partial}{\partial t}\left[\mathbf{a}_0 + (\mathbf{a}_\mathbf{x})_0(\mathbf{x} - \mathbf{x}_0)\right] + \mathbf{f}_0 + (\mathbf{f}_\mathbf{x})_0(\mathbf{x} - \mathbf{x}_0) + \mathbf{q}_0 + (\mathbf{q}_\mathbf{x})_0(\mathbf{x} - \mathbf{x}_0) + (\mathbf{q}_\mathbf{u})_0(\mathbf{u} - \mathbf{u}_0) = 0. \tag{11}$$

Here, the subscript 0 means evaluated at $\mathbf{x}_0$ (or $(\mathbf{x}_0,\mathbf{u}_0)$, as appropriate), $\mathbf{a}_\mathbf{x} = \partial\mathbf{a}/\partial\mathbf{x}$ (and similarly for $\mathbf{f}_\mathbf{x}$ and $\mathbf{q}_\mathbf{x}$) and $\mathbf{q}_\mathbf{u} = \partial\mathbf{q}/\partial\mathbf{u}$. Noting that $(\mathbf{x}_0, \mathbf{u}_0)$ satisfies Eq. (9), we can simplify Eq. (11) to

$$\frac{\partial}{\partial t}\left[(\mathbf{a}_\mathbf{x})_0(\mathbf{x} - \mathbf{x}_0)\right] + (\mathbf{f}_\mathbf{x})_0(\mathbf{x} - \mathbf{x}_0) + (\mathbf{q}_\mathbf{x})_0(\mathbf{x} - \mathbf{x}_0) + (\mathbf{q}_\mathbf{u})_0(\mathbf{u} - \mathbf{u}_0) = 0. \tag{12}$$

This equation is now linear in $(\mathbf{x} - \mathbf{x}_0)$. Given $\mathbf{x}_0$ and $\mathbf{u}_0$, it can be used to compute $\mathbf{x}(t)$. Deviations from the initial trajectory $(\mathbf{x}_0)$ are driven by the $(\mathbf{q}_\mathbf{u})_0(\mathbf{u} - \mathbf{u}_0)$ term.

The numerical implementation of Eq. (12) can be developed by starting with the discrete representation, Eq. (8). We expand around previously simulated (saved) states and corresponding controls, designated as $(\mathbf{x}^{sv}, \mathbf{u}^{sv})$. These states are saved from a small number (two in the examples here) of preprocessing 'training' simulations [14]. Expanding Eq. (8) gives:

$$\mathbf{g}(\mathbf{x}^{n+1}, \mathbf{u}^{n+1}) = \mathbf{g}(\mathbf{x}^{sv}, \mathbf{u}^{sv}) + \left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}}\right)_{sv}(\mathbf{x}^{n+1} - \mathbf{x}^{sv}) + \left(\frac{\partial \mathbf{g}}{\partial \mathbf{u}}\right)_{sv}(\mathbf{u}^{n+1} - \mathbf{u}^{sv}) + \cdots, \tag{13}$$

where $\mathbf{u}^{n+1}$ is the new set of controls (which are specified), $\mathbf{x}^{n+1}$ is the new state we wish to determine, and both $(\partial\mathbf{g}/\partial\mathbf{x})_{sv}$ and $(\partial\mathbf{g}/\partial\mathbf{u})_{sv}$ are matrices evaluated at $(\mathbf{x}^{sv}, \mathbf{u}^{sv})$. The detailed information associated with $(\partial\mathbf{g}/\partial\mathbf{x})_{sv}$ is saved along with the states during the training runs, though in a reduced form as described below.

Given the solution at time step n ($\mathbf{x}^n$), our goal is to represent $\mathbf{x}^{n+1}$ in the form of Eq. (13), with higher-order terms neglected. In the absence of information about the higher-order derivatives, error will be minimized by expanding around the saved state that is the 'closest' to $\mathbf{x}^{n+1}$. We designate the state that is closest to $\mathbf{x}^n$ as $\mathbf{x}^i$. It is then reasonable to assume that the closest saved state to $\mathbf{x}^{n+1}$ (the state we seek to determine) is $\mathbf{x}^{i+1}$, which is the saved state that follows $\mathbf{x}^i$.

There are various criteria for determining the closest saved state, and we will specify our particular approach below. Note that the 'distance' between the new and saved controls need not be considered in this determination as $\mathbf{g}$ is linear in $\mathbf{u}$ for the well model used here (Eq. (6)). Thus, the linearized treatment with respect to $\mathbf{u}$ in Eq. (13) is exact regardless of the distance between $\mathbf{u}^{n+1}$ and $\mathbf{u}^{i+1}$.

We can now proceed with expansions of the form of Eq. (13) for each of the terms in Eq. (8). Expanding around $\mathbf{x}^{i+1}$, we have, for the convective effects

$$\mathbf{F}^{n+1} \approx \mathbf{F}^{i+1} + \frac{\partial \mathbf{F}^{i+1}}{\partial \mathbf{x}^{i+1}}(\mathbf{x}^{n+1} - \mathbf{x}^{i+1}), \tag{14}$$

where $\mathbf{F}^{n+1} = \mathbf{F}(\mathbf{x}^{n+1})$ (and similarly for $\mathbf{F}^{i+1}$) and $\partial \mathbf{F}^{i+1}/\partial \mathbf{x}^{i+1}$ designates the flow term portion of the Jacobian matrix for the state $\mathbf{x}^{i+1}$. Note that the controls do not appear in the flow terms.

The accumulation term depends on the states at time levels n and $n+1$. We thus expand around both $\mathbf{x}^{i+1}$ and $\mathbf{x}^i$, which gives:

$$\mathbf{A}(\mathbf{x}^{n+1}, \mathbf{x}^n) \approx \mathbf{A}^{i+1}(\mathbf{x}^{i+1}, \mathbf{x}^i) + \frac{\partial \mathbf{A}^{i+1}}{\partial \mathbf{x}^{i+1}}(\mathbf{x}^{n+1} - \mathbf{x}^{i+1}) + \frac{\partial \mathbf{A}^{i+1}}{\partial \mathbf{x}^i}(\mathbf{x}^n - \mathbf{x}^i). \tag{15}$$

From here on $\mathbf{A}^{i+1}(\mathbf{x}^{i+1},\mathbf{x}^i)$ will be designated simply as $\mathbf{A}^{i+1}$.

The source/sink term $\mathbf{Q}$ depends on both $\mathbf{x}^{n+1}$ and the new controls $\mathbf{u}^{n+1}$. The source/sink term can thus be represented as

$$\mathbf{Q}(\mathbf{x}^{n+1}, \mathbf{u}^{n+1}) \approx \mathbf{Q}^{i+1}(\mathbf{x}^{i+1}, \mathbf{u}^{i+1}) + \frac{\partial \mathbf{Q}^{i+1}}{\partial \mathbf{x}^{i+1}}(\mathbf{x}^{n+1} - \mathbf{x}^{i+1}) + \frac{\partial \mathbf{Q}^{i+1}}{\partial \mathbf{u}^{i+1}}(\mathbf{u}^{n+1} - \mathbf{u}^{i+1}). \tag{16}$$

Recall that the controls in our case are the well bottom hole pressures, p_i^w. Because of the way in which the controls appear in the source term (Eq. (6)), we can compute $\mathbf{Q}(\mathbf{x}^{i+1}, \mathbf{u}^{n+1})$ directly. In other words, this term can be evaluated using the new controls $\mathbf{u}^{n+1}$ and not the controls $\mathbf{u}^{i+1}$ (which are the controls associated with the saved state $\mathbf{x}^{i+1}$). Thus we can write

$$\mathbf{Q}(\mathbf{x}^{n+1}, \mathbf{u}^{n+1}) \approx \mathbf{Q}(\mathbf{x}^{i+1}, \mathbf{u}^{n+1}) + \frac{\partial \mathbf{Q}^{i+1}}{\partial \mathbf{x}^{i+1}}(\mathbf{x}^{n+1} - \mathbf{x}^{i+1}). \tag{17}$$

This representation eliminates the need to construct $\partial \mathbf{Q}^{i+1}/\partial \mathbf{u}^{i+1}$ or to save it from the training simulations. An alternate representation that does incorporate the $\partial \mathbf{Q}^{i+1}/\partial \mathbf{u}^{i+1}$ term is presented below.

Applying Eqs. (14), (15) and (17) into Eq. (8) we have:

$$\begin{aligned}\mathbf{g}(\mathbf{x}^{n+1}, \mathbf{u}^{n+1}) \approx \mathbf{F}^{i+1} &+ \frac{\partial \mathbf{F}^{i+1}}{\partial \mathbf{x}^{i+1}}(\mathbf{x}^{n+1} - \mathbf{x}^{i+1}) + \mathbf{A}^{i+1} + \frac{\partial \mathbf{A}^{i+1}}{\partial \mathbf{x}^{i+1}}(\mathbf{x}^{n+1} - \mathbf{x}^{i+1}) + \frac{\partial \mathbf{A}^{i+1}}{\partial \mathbf{x}^i}(\mathbf{x}^n - \mathbf{x}^i) + \mathbf{Q}(\mathbf{x}^{i+1}, \mathbf{u}^{n+1}) \\ &+ \frac{\partial \mathbf{Q}^{i+1}}{\partial \mathbf{x}^{i+1}}(\mathbf{x}^{n+1} - \mathbf{x}^{i+1}).\end{aligned} \tag{18}$$

Defining the Jacobian matrix as:

$$\mathbf{J}^{i+1} = \frac{\partial \mathbf{F}^{i+1}}{\partial \mathbf{x}^{i+1}} + \frac{\partial \mathbf{A}^{i+1}}{\partial \mathbf{x}^{i+1}} + \frac{\partial \mathbf{Q}^{i+1}}{\partial \mathbf{x}^{i+1}}, \tag{19}$$

enables us to express Eq. (18) more concisely

$$\mathbf{g}(\mathbf{x}^{n+1}, \mathbf{u}^{n+1}) \approx \mathbf{F}^{i+1} + \mathbf{A}^{i+1} + \frac{\partial \mathbf{A}^{i+1}}{\partial \mathbf{x}^i}(\mathbf{x}^n - \mathbf{x}^i) + \mathbf{Q}(\mathbf{x}^{i+1}, \mathbf{u}^{n+1}) + \mathbf{J}^{i+1}(\mathbf{x}^{n+1} - \mathbf{x}^{i+1}) = 0. \tag{20}$$

Now, setting the residual using the new controls at the new state to zero ($\mathbf{g}(\mathbf{x}^{n+1}, \mathbf{u}^{n+1}) = 0$), we arrive at an equation that can be solved to determine $\mathbf{x}^{n+1}$:

$$\mathbf{J}^{i+1}(\mathbf{x}^{n+1} - \mathbf{x}^{i+1}) = -\left[\mathbf{F}^{i+1} + \mathbf{A}^{i+1} + \frac{\partial \mathbf{A}^{i+1}}{\partial \mathbf{x}^i}(\mathbf{x}^n - \mathbf{x}^i) + \mathbf{Q}(\mathbf{x}^{i+1}, \mathbf{u}^{n+1})\right]. \tag{21}$$

An alternate representation for Eq. (21) can also be constructed. This representation has the advantage of not requiring the $\mathbf{F}^{i+1}$, $\mathbf{A}^{i+1}$ and $\mathbf{Q}(\mathbf{x}^{i+1}, \mathbf{u}^{n+1})$ terms but it does require the $\partial \mathbf{Q}^{i+1}/\partial \mathbf{u}^{i+1}$ term. The relative advantages of this alternate representation compared to Eq. (21) depend on which terms are most readily available, or easiest to compute, in the simulator used to generate the TPWL model, and on the well model that is applied. To generate the alternate representation, we express $\mathbf{Q}(\mathbf{x}^{i+1}, \mathbf{u}^{n+1})$ as

$$\mathbf{Q}(\mathbf{x}^{i+1}, \mathbf{u}^{n+1}) \approx \mathbf{Q}(\mathbf{x}^{i+1}, \mathbf{u}^{i+1}) + \frac{\partial \mathbf{Q}^{i+1}}{\partial \mathbf{u}^{i+1}}(\mathbf{u}^{n+1} - \mathbf{u}^{i+1}). \tag{22}$$

Inserting this representation into Eq. (21), and noting that $\mathbf{g}^{i+1} = \mathbf{F}^{i+1} + \mathbf{A}^{i+1} + \mathbf{Q}^{i+1} = 0$, we have

$$\mathbf{J}^{i+1}(\mathbf{x}^{n+1} - \mathbf{x}^{i+1}) = -\left[\frac{\partial \mathbf{A}^{i+1}}{\partial \mathbf{x}^i}(\mathbf{x}^n - \mathbf{x}^i) + \frac{\partial \mathbf{Q}^{i+1}}{\partial \mathbf{u}^{i+1}}(\mathbf{u}^{n+1} - \mathbf{u}^{i+1})\right]. \tag{23}$$

Because the source terms in our problem are linear in the controls (Eq. (6)), TPWL models based on the above linearizations (Eqs. (21) and (23)) give identical results. More general well models, which represent wellbore flow effects, can also be applied (see, e.g., [29]). These models are nonlinear, so in this case we would expect to see some differences between results using Eqs. (21) and (23).

2.3. Construction of the POD basis matrix

Eqs. (21) and (23) provide a representation for new states in terms of saved information, but they are of only limited use in their current forms. This is because $\mathbf{J}^{i+1}$ and $\partial\mathbf{A}^{i+1}/\partial\mathbf{x}^i$ are, although sparse, of dimensions $2N_c \times 2N_c$, and the other terms in Eq. (21) are vectors of dimension $2N_c$ (recall that N_c is the number of grid blocks in the high-fidelity model). We therefore proceed by applying proper orthogonal decomposition (POD) to construct an orthonormal basis that can be used to project the high-fidelity linearized models to a lower-dimensional representation. As discussed in the Introduction, POD can be applied directly to the general simulation equations to provide a POD-based reduced-order model (as in [9,11]). Here, by contrast, we apply POD in conjunction with the linearized representation. We note that many previous TPWL implementations [14,17–20] used Krylov techniques to construct the reduced-order basis, though POD was used by [16,21].

The basic POD procedure is presented in detail in [9,11] so our description here will be brief. The first step in POD is to simulate the high-fidelity system one or more times with prescribed sequences for the controls $\mathbf{u}$. These runs are known as training simulations. The pressure and saturation states at each time step are saved as solution 'snapshots,' designated $\mathbf{x}_p$ and $\mathbf{x}_s$ respectively (note that in our actual implementation we construct the POD basis using oil potential rather than pressure – see Section 2.5 for further discussion of this issue). These snapshots also represent the saved states used in the TPWL procedure. The snapshots from all training simulations are assembled in the data matrices $\mathbf{X}_p$ and $\mathbf{X}_s$, where each column corresponds to a single pressure (or oil potential) or saturation snapshot. The POD procedure is applied separately to $\mathbf{X}_p$ and $\mathbf{X}_s$. In the description below, we drop the p and s subscripts for conciseness.

Assume we have a total of k pressure snapshots and k saturation snapshots, meaning that the data matrix $\mathbf{X}$ is of dimensions $N_c \times k$ (typically $k \ll N_c$). The POD basis matrix is given by the eigenfunctions of the matrix $\mathbf{X}\mathbf{X}^T$. This matrix is of dimensions $N_c \times N_c$, so eigen-decomposition will be very expensive for large N_c. The eigenvectors of $\mathbf{X}\mathbf{X}^T$, however, are identical to the left singular vectors of $\mathbf{X}$, so we instead perform a singular value decomposition (SVD) of $\mathbf{X}$. The singular values (σ_i) of $\mathbf{X}$ are related to eigenvalues (λ_i) of $\mathbf{X}\mathbf{X}^T$ via $\sigma_i = \lambda_i^{1/2}$.

If the basis matrix $\mathbf{\Phi}$ contains as its columns all of the eigenvectors of $\mathbf{X}\mathbf{X}^T$ (left singular vectors of $\mathbf{X}$) that correspond to nonzero eigenvalues, it will be of dimensions $N_c \times k$. However, there is very little 'energy' associated with many of the eigenvectors (columns of $\mathbf{\Phi}$) and these can be eliminated with little effect. The energy E_ℓ associated with the ℓ most significant eigenvectors is quantified as $E_\ell = \sum_{i=1}^{\ell} \lambda_i$, where the eigenvalues are ordered in decreasing λ_i. The total energy of the system (E_t) is computed by setting $\ell = k$ in this computation. By specifying the fraction of the total energy that we wish to capture in $\mathbf{\Phi}$, a reasonable value for $\ell \leqslant k$ can be determined. Other criteria, involving the rate of decay of λ_i with increasing i, can also be used to select ℓ. In actual applications, some amount of iteration may be required to determine appropriate values of ℓ_p and ℓ_s.

After applying the above procedure to both $\mathbf{X}_p$ and $\mathbf{X}_s$, the basis matrices $\mathbf{\Phi}_p$ (of dimensions $N_c \times \ell_p$) and $\mathbf{\Phi}_s$ ($N_c \times \ell_s$) are combined to produce one basis matrix $\mathbf{\Phi}$ of dimensions $2N_c \times \ell$, where $\ell = \ell_p + \ell_s$. The exact way in which these matrices are combined depends on the ordering of the unknowns – see Eq. (13) in [11] for the detailed form. The $\mathbf{\Phi}$ matrix can now be used to relate a high-dimensional state $\mathbf{x}$ to a reduced state $\mathbf{z}$; i.e.,

$$\mathbf{x} \approx \mathbf{\Phi}\mathbf{z}, \tag{24}$$

where $\mathbf{z}$ is of dimension ℓ. Because $\mathbf{\Phi}$ is orthonormal, we also have $\mathbf{z} = \mathbf{\Phi}^T\mathbf{x}$.

In the POD-based ROMs for subsurface flow presented in [9,11], at each iteration of each time step the full Jacobian matrix $\mathbf{J}$ is constructed and subsequently projected into reduced space to provide the reduced Jacobian $\mathbf{J}_r$, where $\mathbf{J}_r = \mathbf{\Phi}^T\mathbf{J}\mathbf{\Phi}$. A linear system of dimension ℓ is then solved to compute the solution update in reduced space. As indicated in the Introduction, this procedure can provide a degree of speedup (at most about a factor of 10) relative to high-fidelity computations but it is limited due to the need for these costly computations at each iteration. As we will see below, the inline portion of the TPWL procedure does not require these calculations, so much greater runtime speedups can be achieved.

2.4. TPWL representation with POD basis matrix

We now use $\mathbf{\Phi}$ to project Eqs. (21) and (23) into reduced space. We first consider Eq. (21). Premultiplying both sides of Eq. (21) by $\mathbf{\Phi}^T$ and approximating $\mathbf{x}$ as $\mathbf{\Phi}\mathbf{z}$ gives:

$$\mathbf{\Phi}^T\mathbf{J}^{i+1}\mathbf{\Phi}(\mathbf{z}^{n+1} - \mathbf{z}^{i+1}) = -\mathbf{\Phi}^T\left[\mathbf{F}^{i+1} + \mathbf{A}^{i+1} + \frac{\partial\mathbf{A}^{i+1}}{\partial\mathbf{x}^i}\mathbf{\Phi}(\mathbf{z}^n - \mathbf{z}^i) + \mathbf{Q}(\mathbf{x}^{i+1}, \mathbf{u}^{n+1})\right]. \tag{25}$$

This equation can be rewritten more concisely by defining $\mathbf{J}_r^{i+1} = \mathbf{\Phi}^T\mathbf{J}^{i+1}\mathbf{\Phi}$, $(\partial\mathbf{A}^{i+1}/\partial\mathbf{x}^i)_r = \mathbf{\Phi}^T(\partial\mathbf{A}^{i+1}/\partial\mathbf{x}^i)\mathbf{\Phi}$, $\mathbf{F}_r^{i+1} = \mathbf{\Phi}^T\mathbf{F}^{i+1}$, $\mathbf{A}_r^{i+1} = \mathbf{\Phi}^T\mathbf{A}^{i+1}$, and $\mathbf{Q}_r = \mathbf{\Phi}_w^T\mathbf{Q}$. The resulting equation can be solved directly for $\mathbf{z}^{n+1}$

$$\mathbf{z}^{n+1} = \mathbf{z}^{i+1} - \left(\mathbf{J}_r^{i+1}\right)^{-1}\left[\mathbf{F}_r^{i+1} + \mathbf{A}_r^{i+1} + \left(\frac{\partial\mathbf{A}^{i+1}}{\partial\mathbf{x}^i}\right)_r(\mathbf{z}^n - \mathbf{z}^i) + \mathbf{Q}_r(\mathbf{x}^{i+1}, \mathbf{u}^{n+1})\right]. \tag{26}$$

In Eq. (26), $\mathbf{J}_r^{i+1}$ and $(\partial \mathbf{A}^{i+1}/\partial \mathbf{x}^i)_r$ are full matrices of dimension $\ell \times \ell$ and the other terms are vectors of dimension ℓ. The matrices $(\mathbf{J}_r^{i+1})^{-1}$ and $(\partial \mathbf{A}^{i+1}/\partial \mathbf{x}^i)_r$ and the vectors $\mathbf{z}^{i+1}$, $\mathbf{F}_r^{i+1}$ and $\mathbf{A}_r^{i+1}$ can be precomputed because they are constructed from the reduced basis $\mathbf{\Phi}$ and information saved during the training runs. The vector $\mathbf{Q}_r(\mathbf{x}^{i+1},\mathbf{u}^{n+1})$, by contrast, cannot be precomputed because it depends on the controls $\mathbf{u}^{n+1}$, which are not known when the training simulations are performed. This term can be efficiently constructed, however, because high-fidelity information is required only for the small fraction of grid blocks that contain wells. After the well block entries to $\mathbf{Q}(\mathbf{x}^{i+1}, \mathbf{u}^{n+1})$ are computed, this vector can be projected into the reduced space using $\mathbf{Q}_r(\mathbf{x}^{i+1}, \mathbf{u}^{n+1}) = \mathbf{\Phi}_w^T \mathbf{Q}(\mathbf{x}^{i+1}, \mathbf{u}^{n+1})$, where $\mathbf{\Phi}_w^T$ includes only those rows of the basis matrix that correspond to grid blocks containing wells.

The reduced-order version of Eq. (23) can be constructed by proceeding in a similar manner. The result is

$$\mathbf{z}^{n+1} = \mathbf{z}^{i+1} - \left(\mathbf{J}_r^{i+1}\right)^{-1} \left[\left(\frac{\partial \mathbf{A}^{i+1}}{\partial \mathbf{x}^i} \right)_r (\mathbf{z}^n - \mathbf{z}^i) + \left(\frac{\partial \mathbf{Q}^{i+1}}{\partial \mathbf{u}^{i+1}} \right)_r (\mathbf{u}^{n+1} - \mathbf{u}^{i+1}) \right], \tag{27}$$

where $(\partial \mathbf{Q}^{i+1}/\partial \mathbf{u}^{i+1})_r = \mathbf{\Phi}_w^T (\partial \mathbf{Q}^{i+1}/\partial \mathbf{u}^{i+1})$.

The actual TPWL implementations accomplished in this work are for the case of incompressible fluid and rock. For such systems, with reference to Eq. (5), the accumulation term becomes

$$\mathbf{A}(\mathbf{x}^{n+1}, \mathbf{x}^n) = \mathbf{D}(\mathbf{x}^{n+1} - \mathbf{x}^n) \text{ and } \mathbf{A}(\mathbf{x}^{i+1}, \mathbf{x}^i) = \mathbf{D}(\mathbf{x}^{i+1} - \mathbf{x}^i), \tag{28}$$

which gives $\partial \mathbf{A}^{i+1}/\partial \mathbf{x}^{i+1} = -\partial \mathbf{A}^{i+1}/\partial \mathbf{x}^i = \mathbf{D}$, where $\mathbf{D}$ is a constant matrix.

We note finally that another variant of TPWL, which represents $\mathbf{Q}(\mathbf{x}^{n+1}, \mathbf{u}^{n+1})$ in terms of an expansion around $\mathbf{Q}(\mathbf{x}^n, \mathbf{u}^{n+1})$, is discussed in [30]. This representation performs well in many cases (including those presented in Section 3) but has been found to be less accurate for problems involving strong density differences and variable injector pressures (this appears to be due to the additional reconstruction error and instability associated with this TPWL variant). Thus TPWL models based on Eq. (26) or (27) are evidently the most promising for general subsurface flow simulations. In all results presented here, the TPWL model defined by Eq. (27) is applied.

2.5. Implementation issues

We now comment briefly on aspects of the numerical implementation and on some of the heuristic treatments employed. The high-fidelity simulations are all performed using Stanford's general purpose research simulator, GPRS [31,29]. The simulator was modified to output the Jacobian information from the training runs, as required by the TPWL procedure. The output files occupy substantial disk space, as the matrices for the high-fidelity model can be quite large. Following the simulation of the training runs, the reduced basis $\mathbf{\Phi}$ is constructed, the reduced matrices are generated, and the reduced Jacobian matrices are inverted. The computational requirements for constructing the basis and generating the required matrices are comparable to the time required to perform the training simulations. These operations are however now performed using Matlab [32], so it is likely that they can be accelerated.

If these output files are too large for the available disk space, then an alternate procedure can be applied in which the training simulations are run twice. In the first set of training runs, only the states are saved. Using these states, the basis matrix $\mathbf{\Phi}$ is constructed. Then, in the second set of runs, the basis matrix is input to the simulator and the reduced matrices (e.g., $\mathbf{J}_r$) are generated and output. Using this approach we avoid outputting and storing any high-fidelity ($2N_c \times 2N_c$) matrices, though the number of training simulations doubles (this extra cost is somewhat mitigated by time savings associated with reduced output to disk).

In forming the $\mathbf{\Phi}_p$ matrix (described in Section 2.3), we have investigated the use of snapshots of either pressure or oil potential. Oil potential, designated φ_o, combines pressure and gravitational effects and is given by $\varphi_o = p/\rho_o - gD$. In the POD-based reduced-order modeling procedure developed in [11], for cases with significant difference in density between phases, enhanced accuracy may be observed when the $\mathbf{\Phi}$ matrix is computed using snapshots of φ_o rather than p. For the TPWL results presented in Section 3, however, we achieved essentially identical results using either approach. The results actually presented use snapshots of oil potential. For the determination of ℓ_p and ℓ_s, we first compute upper estimates for these quantities by specifying E_ℓ/E_t for oil potential and saturation states as 0.999999 and 0.9999, respectively. A few tests are then performed, using lower values of ℓ_p and ℓ_s, to find values that provide an appropriate degree of accuracy.

The TPWL solutions (Eq. (26) or (27)) are implemented as Matlab procedures. Our formulation generally avoids the need to perform any computations using the high-fidelity model. However, in the following section we do report on the mass balance error observed using TPWL for some cases. For such calculations, the high-fidelity model must be reconstructed (using $\mathbf{x} \approx \mathbf{\Phi z}$) and computations then performed using it. This will of course reduce the speedup achieved using TPWL.

The TPWL procedure represents new solutions, designated $\mathbf{x}^{n+1}$, in terms of expansions around saved solutions, designated $\mathbf{x}^{i+1}$. As described above, this requires that we first determine the 'closest' saved state $\mathbf{x}^i$ to the solution at the previous time step, $\mathbf{x}^n$. It would be time consuming if this determination was performed using high-fidelity states. Therefore, we investigated various approaches for finding the closest state using only reduced state information. The best accuracy was achieved by determining the saved state that minimizes $|\mathbf{z}_s^n - \mathbf{z}_s^i|$, where $\mathbf{z}_s^n = \mathbf{\Phi}_s^T \mathbf{x}_s^n$ and $\mathbf{z}_s^i = \mathbf{\Phi}_s^T \mathbf{x}_s^i$ designate reduced saturation states

(we only need to include the first five components of $\mathbf{z}$ in computing $|\mathbf{z}_s^n - \mathbf{z}_s^i|$ as the component magnitudes decrease quickly). Inclusion of the reduced pressure states in this determination did not provide additional accuracy. This is likely because the problem is linear in pressure but nonlinear in saturation. Thus, the basic linearization (Eq. (13)) is less accurate for saturation than pressure, which renders TPWL solutions more sensitive to distance from the saturation state than to distance from the pressure state. We note that the distance between $\mathbf{z}_s^{n+1}$ and $\mathbf{z}_s^{i+1}$ can be checked after the TPWL time step, and if there is a saved state that is closer to $\mathbf{z}_s^{n+1}$, the time step can be repeated (this approach was not implemented here).

The basic TPWL algorithms are summarized below. Algorithm 1 presents the steps in the offline portion of the TPWL procedure. Steps 2–4 involve simulating and saving information from the training simulations. Steps 6–13 entail the construction of the $\mathbf{\Phi}$ matrix, which is formed from $\mathbf{\Phi}_p$ and $\mathbf{\Phi}_s$. Steps 15–17 generate reduced-space quantities that are needed for either TPWL representation. Steps 19–21 are applied if the TPWL model is defined by Eq. (26), while step 24 is followed if the TPWL model is defined by Eq. (27).

Algorithm 2 defines the inline part of the TPWL procedure. Step 1 prescribes the sequence of controls $\mathbf{u}$, and the model is run until the specified final time. Note that global reconstruction (step 6) need only be applied if material balance errors are being computed. Otherwise, only the well states are required (step 8).

Algorithm 1. TPWL offline processing

1	**for** each training simulation **do**
2	Set controls $\mathbf{u}$
3	Run high-fidelity training simulation (GPRS)
4	Save snapshots $\mathbf{x}_p$ and $\mathbf{x}_s$ and Jacobian matrices $\mathbf{J}$
5	**end for**
6	Assemble data matrices $\mathbf{X}_p$ and $\mathbf{X}_s$
7	**for** $\mathbf{X}_p$ and $\mathbf{X}_s$ **do**
8	Perform SVD($\mathbf{X}$)
9	$\lambda_i = \sigma_i^2$
10	Determine ℓ
11	Construct $\mathbf{\Phi}$
12	**end for**
13	Combine $\mathbf{\Phi}_p$ and $\mathbf{\Phi}_s$ to form full $\mathbf{\Phi}$
14	**for** each training simulation **do**
15	$\mathbf{z} = \mathbf{\Phi}^T\mathbf{x}$
16	$\mathbf{J}_r^{i+1} = \mathbf{\Phi}^T\mathbf{J}^{i+1}\mathbf{\Phi}$
17	$(\partial\mathbf{A}^{i+1}/\partial\mathbf{x}^i)_r = \mathbf{\Phi}^T(\partial\mathbf{A}^{i+1}/\partial\mathbf{x}^i)\mathbf{\Phi}$
18	**for** Eq. (26) **do**
19	$\mathbf{F}_r^{i+1} = \mathbf{\Phi}^T\mathbf{F}^{i+1}$
20	$\mathbf{A}_r^{i+1} = \mathbf{\Phi}^T\mathbf{A}^{i+1}$
21	$\mathbf{Q}_r = \mathbf{\Phi}_w^T\mathbf{Q}$
22	**end for**
23	**for** Eq. (27) **do**
24	$(\partial\mathbf{Q}^{i+1}/\partial\mathbf{u}^{i+1})_r = \mathbf{\Phi}_w^T(\partial\mathbf{Q}^{i+1}/\partial\mathbf{u}^{i+1})$
25	**end for**
26	**end for**

Algorithm 2. TPWL inline processing

1	Set new controls $\mathbf{u}$
2	**while** $t < t_{final}$ **do**
3	Find $\mathbf{z}_s^i$ closest to $\mathbf{z}_s^n$
4	Solve linearized model (Eq. (26) or (27)) to compute new state $\mathbf{z}^{n+1}$
5	**if** material balance error required **then**
6	$\mathbf{x}^{n+1} \approx \mathbf{\Phi}\mathbf{z}^{n+1}$
7	**else**
8	$\mathbf{x}_w^{n+1} \approx \mathbf{\Phi}_w\mathbf{z}^{n+1}$
9	**end if**
10	**end while**

3. Simulation results

In this section we apply the TPWL procedure to two reservoir models –one containing 24,000 grid blocks and the other containing 79,200 grid blocks. We consider the performance of the TPWL models for a variety of test runs and also apply TPWL within the context of a multiobjective production optimization problem.

3.1. Reservoir model 1

3.1.1. Model description

This reservoir model, shown in Fig. 2, is a modified portion of a channelized reservoir model presented in [33] (the full model is often referred to as SPE 10). The model is three-dimensional and contains 24,000 grid blocks, with $n_x = 60$, $n_y = 80$ and $n_z = 5$, where n_x, n_y and n_z designate the number of grid blocks in each direction (we now refer to the spatial directions as (x, y, z)). There are four production wells (designated P1–P4) and two water injection wells (designated I1 and I2).

Permeability is taken to be a diagonal tensor, with $k_x = k_y$. The mean k_x is 418 mD and the mean porosity is 0.203. The vertical permeability (k_z) is prescribed as $k_z = 0.3k_x$ in the channels and $k_z = 10^{-3}k_x$ in the non-channel regions. The initial oil and water saturations are 0.8 and 0.2, respectively and the residual oil (S_{or}) and water (S_{wr}) saturations are 0.2. For oil we set $\rho_o = 45$ lb/ft^3, $\mu_o = 3.0$ cp; for water, $\rho_w = 60$ lb/ft^3, $\mu_w = 0.5$ cp. The system is incompressible and capillary pressure effects are neglected. The relative permeabilities for the oil and water phases are specified as:

$$k_{ro}(S_w) = k_{ro}^0 \left(\frac{1 - S_w - S_{or}}{1 - S_{wr} - S_{or}} \right)^a, \tag{29}$$

$$k_{rw}(S_w) = k_{rw}^0 \left(\frac{S_w - S_{wr}}{1 - S_{wr} - S_{or}} \right)^b, \tag{30}$$

where k_{ro}^0 and k_{rw}^0 are the endpoint relative permeabilities. Here we set $k_{ro}^0 = k_{rw}^0 = 1$ and $a = b = 2$.

3.1.2. Training runs

The first step in the TPWL procedure is to perform a small number of training runs using the high-fidelity model to generate the states and Jacobian matrices (GPRS is used for these simulations). For this reservoir model two training simulations are performed. In all of these simulations the injection wells are prescribed to maintain constant bottom hole pressure (BHP) of 10,000 psia. This condition will be maintained in subsequent test runs (our focus here is on varying the production well BHPs). For the production wells, the BHPs are prescribed to follow schedules that are expected to represent approximately the operating conditions that will be encountered in subsequent simulations. We have not yet established a fully systematic approach for prescribing the training run BHPs, so we instead apply the following heuristic procedure (see [11] for further discussion of this issue).

In the first training simulation, the BHPs of all four production wells are changed every 200 days, as indicated in Fig. 3(a), such that the total flow rate (oil + water) of each production well is approximately the same and approximately constant in time for each 1000 day interval. Specifically, over the first 1000 days, the total flow rate per well is 2000 stb/d (stock tank barrels per day), from 1000 to 2000 days it is 2300 stb/d, from 2000 to 3000 days it is 2700 stb/d, from 3000 to 4000 days it is 3100 stb/d and from 4000 to 5000 days it is 3300 stb/d. In the second training schedule, the BHPs of the production wells are varied every 200 days, randomly and independently, between 1000 and 4000 psia, as illustrated in Fig. 3(b) for P1 and P2 (wells P3 and P4 display similar BHP profiles). Thus with the training runs we attempt to capture constant flow rate behavior (run 1) and more variable effects (run 2). Other training sequences could of course be devised, though as we will see below this set of training runs generally results in accurate predictions in the subsequent test runs.

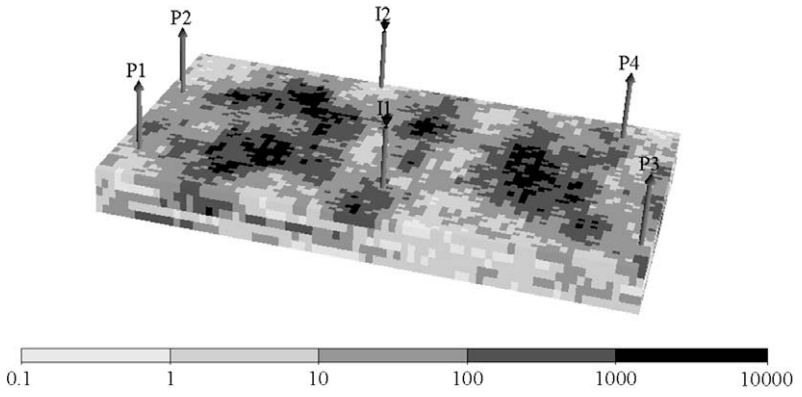

Fig. 2. Synthetic reservoir model (24,000 grid blocks) with four production wells and two injection wells. Permeability in x-direction (in mD) is shown.

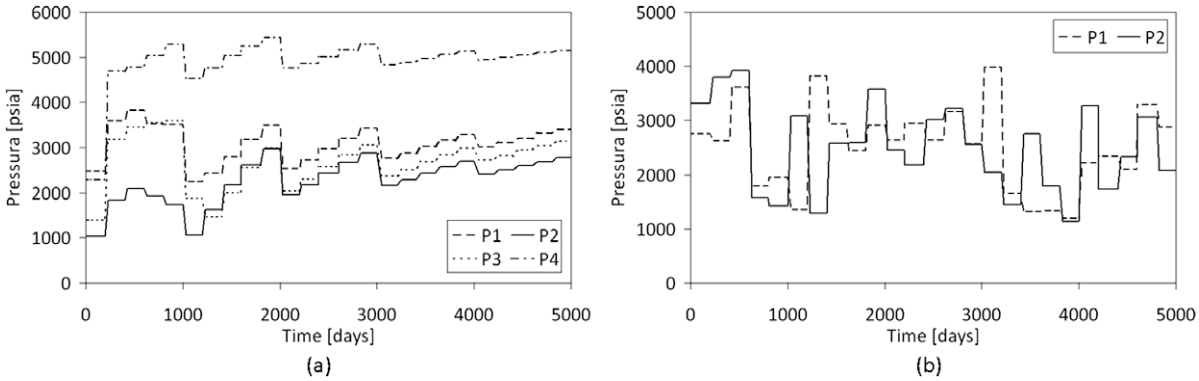

Fig. 3. Producer BHP schedules for (a) training run 1 and (b) training run 2.

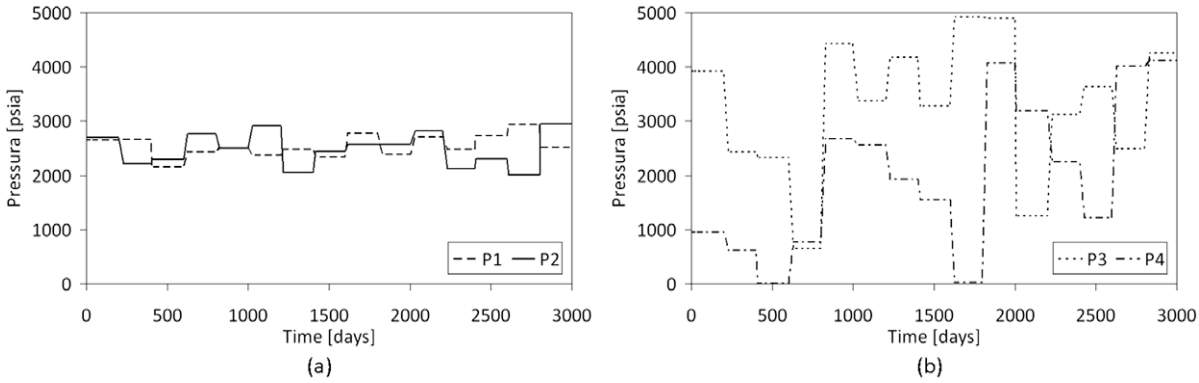

Fig. 4. Producer BHP schedules for (a) P1 and P2 for test A and (b) P3 and P4 for test E.

In the training runs, we simulate reservoir performance for a total of 5000 days with a maximum time step of 30 days. From the two training runs 391 pressure and saturation snapshots and Jacobian matrices were recorded. The POD basis matrix was constructed following the approach described in Section 2.3. The reduced basis $\mathbf{\Phi}$ used for most of the runs in this section contains a total of 400 columns, 250 of which correspond to pressure states and 150 to saturation states.

3.1.3. Test runs

We perform a total of eight test simulations using a variety of specifications for producer BHPs. Five tests (designated A–E) are performed with random schedules. In test A the production well BHPs vary from 2000 to 3000 psia, for test B from 1500 to 3500 psia, for test C from 1000 to 4000 psia, for test D from 500 to 4500 psia, and for test E from 0 to 5000 psia. Fig. 4(a) presents the BHPs for the production wells P1 and P2 for test A and Fig. 4(b) shows the BHPs for wells P3 and P4 for test E. Note that the BHPs for tests A–C are within the range of the training runs, while those for tests D and E are outside of this range.

Another set of three tests (F–H) is performed by modifying the producer BHPs used in training run 2 by varying amounts. Fig. 5(a) shows that for tests F and G the BHPs of production well P1 are consistently below the BHPs used in training run 2. On the other hand, for test H the BHPs are consistently above those used for training. The same is true for wells P2 and P3. For production well P4, which is the well with the highest flow rate, in tests F–H we prescribe BHPs below that of training run 2, as indicated in Fig. 5(b) (test H entails the largest deviation from the training run and is therefore the most challenging for the TPWL model). Fig. 5(c) presents the BHPs for all of the production wells in test F and Fig. 5(d) depicts those for test H.

3.1.4. Error measures

The accuracy of the TPWL simulations can be assessed both visually and by computing the average error, relative to the reference high-fidelity simulation, for the oil and water production rates and the water injection rate. We compute these errors as in our previous paper [11]. Rates from the TPWL simulation are interpolated to correspond to the simulation times in the high-fidelity simulation. We avoid interpolating across discontinuities in the well BHP. Our procedure for computing error in oil production rate is as follows. For each production well m, at every simulated time step i, the oil production rates in the high-fidelity simulation ($Q_{o,hf}^{m,i}$) and in the TPWL simulation ($Q_{o,tpwl}^{m,i}$) are computed. The time-average error for each

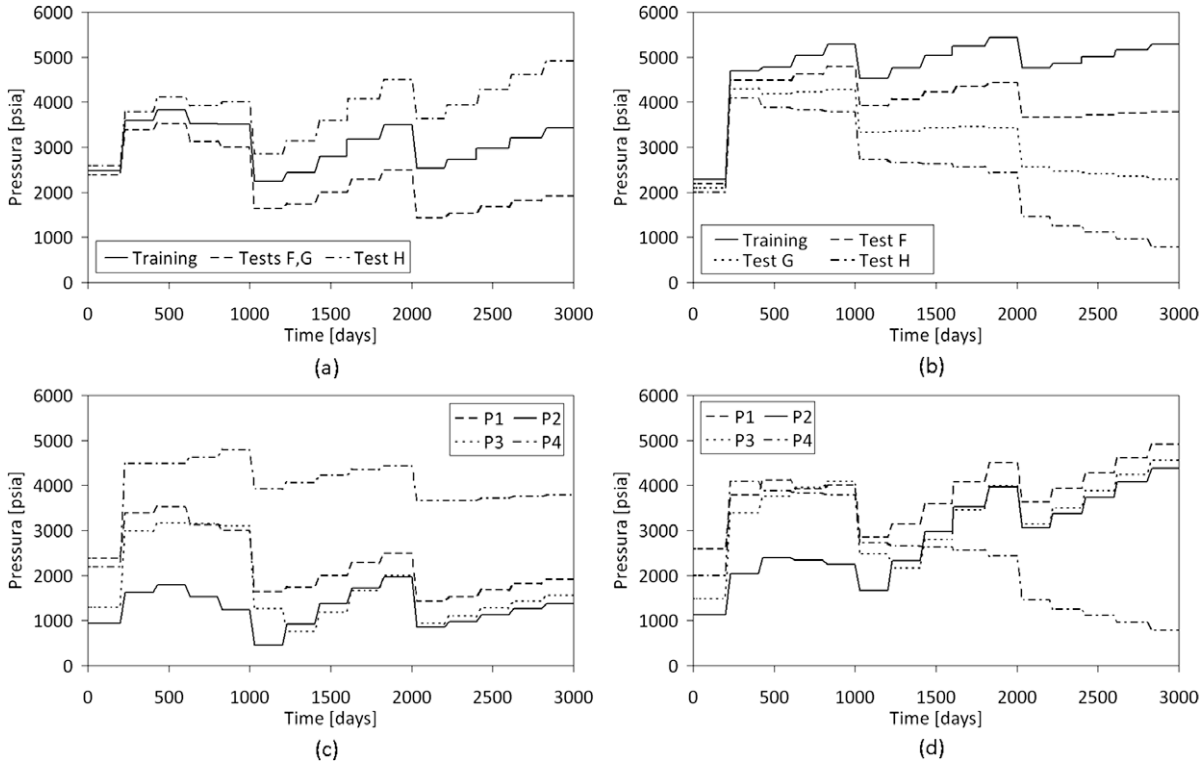

Fig. 5. Producer BHP schedules for training run 2 and tests F–H: (a) P1, (b) P4, (c) test F – all wells and (d) test H – all wells.

well, designated E^m, is calculated as the average of the absolute differences, normalized by the time-average oil flow rate $\overline{Q}^m_{o,hf}$ for the well:

$$E^m = \frac{1}{n_t \overline{Q}^m_{o,hf}} \sum_{i=1}^{n_t} |Q^{m,i}_{o,hf} - Q^{m,i}_{o,tpwl}|, \tag{31}$$

where n_t is the total number of time steps. The overall average error, designated E, is computed by averaging E^m over all wells:

$$E = \frac{1}{n_w} \sum_{m=1}^{n_w} E^m, \tag{32}$$

where n_w is the number of production wells. The same procedure is applied to compute average errors in water production and water injection rates (for water injection we average over the two injection wells).

The material balance errors for the oil and water components are also of interest. These are computed as follows for the incompressible case considered here. For oil, at every time step i, the volume of oil in place (V^i_o) and the cumulative production of oil (Q^i_o) are computed (note that this requires that we reconstruct the high-fidelity solution). The oil material balance error at time step i (MBE^i_o) is given by:

$$\mathrm{MBE}^i_o = \frac{V^0_o - (V^i_o + Q^i_o)}{V^0_o}, \tag{33}$$

where V^0_o is the volume of oil in place at initial time. For the water component, the volume in place (V^i_w), the cumulative production (Q^i_w) and the cumulative injection (I^i_w) are similarly updated at each time step and the material balance error is computed as:

$$\mathrm{MBE}^i_w = \frac{V^0_w - (V^i_w + Q^i_w - I^i_w)}{V^0_w}, \tag{34}$$

where V^0_w is the volume of water originally in place. The material balance errors reported below are the maximum MBE^i_o and MBE^i_w computed during the course of the TPWL simulation.

3.1.5. Model 1 simulations

We now present results illustrating and quantifying the performance of TPWL for the test runs. Fig. 6 demonstrates the accuracy of the TPWL model relative to the high-fidelity model for test C. The producer BHP schedules are shown in Fig. 6(a).

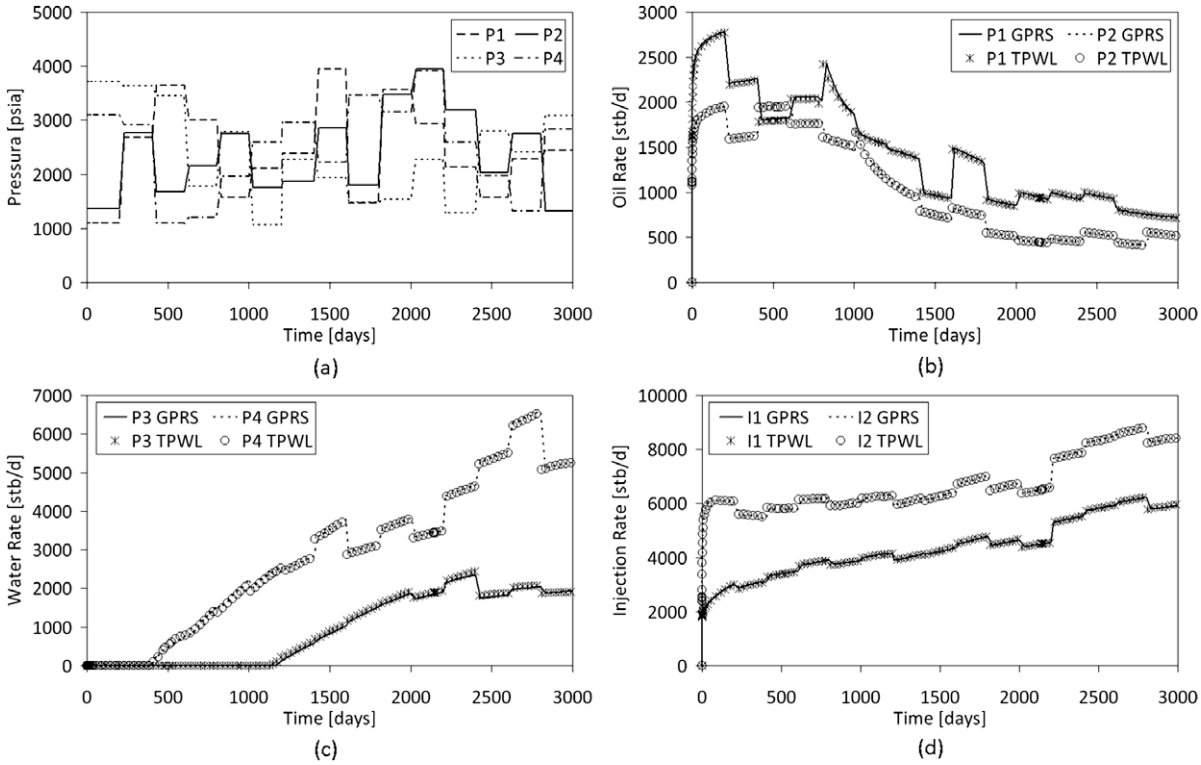

Fig. 6. Results for test C: (a) BHPs for each production well, (b) oil rate for production wells P1 and P2, (c) water rate for production wells P3 and P4 and (d) injection rates for wells I1 and I2.

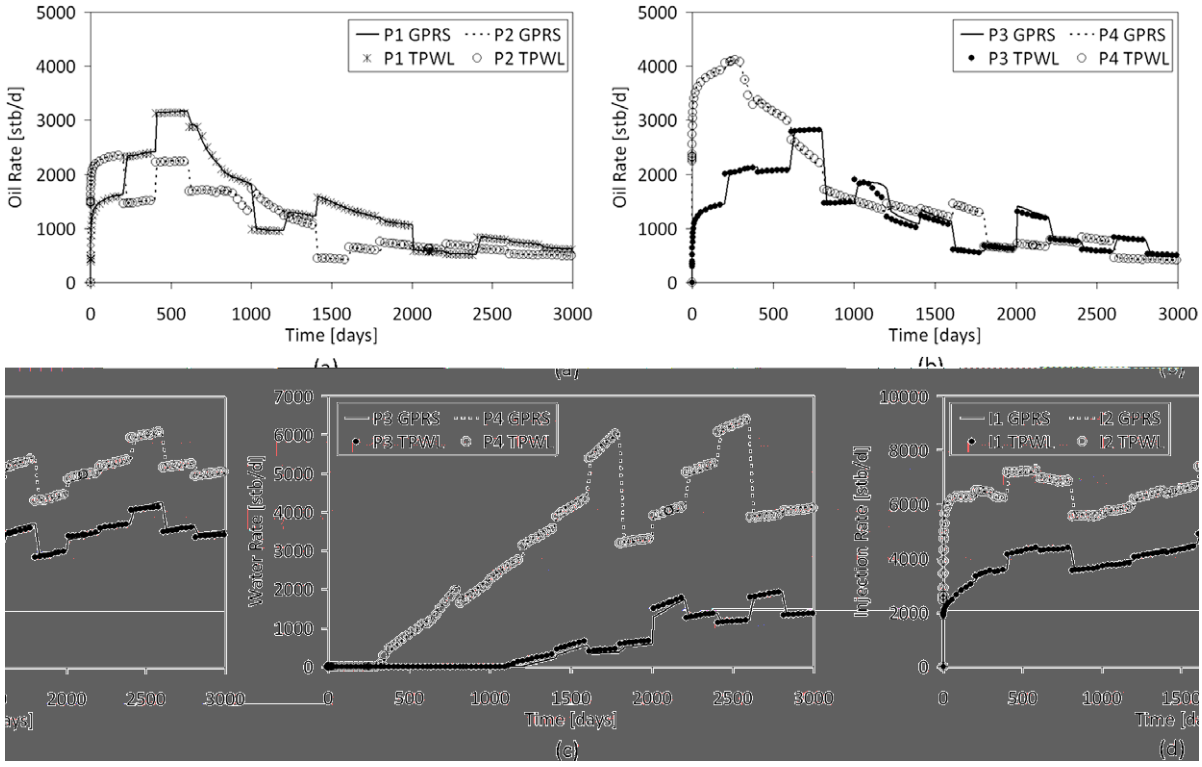

Fig. 7. Results for test E: (a) oil rate for production wells P1 and P2, (b) oil rate for production wells P3 and P4, (c) water rate for production wells P3 and P4 and (d) injection rates for wells I1 and I2.

Table 1
Errors in TPWL solutions for tests A–E.

	Test A	Test B	Test C	Test D	Test E
Oil prod.	0.0080	0.0126	0.0147	0.0187	0.0236
Water prod.	0.0176	0.0255	0.0247	0.0343	0.0819
Water inj.	0.0049	0.0057	0.0075	0.0098	0.0113
MBE oil	0.0005	0.0009	0.0009	0.0014	0.0030
MBE water	0.0015	0.0014	0.0051	0.0035	0.0140

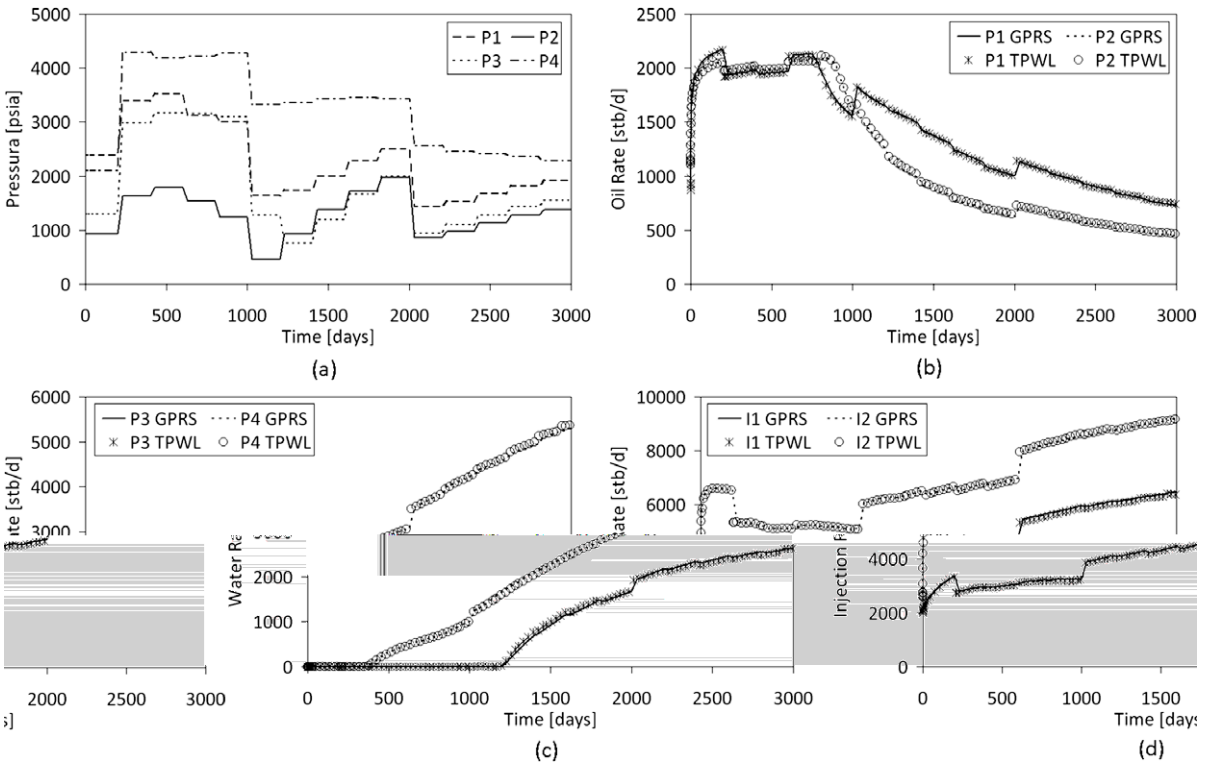

Fig. 8. Results for test G: (a) BHPs for each production well, (b) oil rate for production wells P1 and P2, (c) water rate for production wells P3 and P4 and (d) injection rates for wells I1 and I2.

It is apparent that these well BHPs differ significantly from any of the training runs, though they are within the range of the training BHPs. Fig. 6(b) presents the oil rates for wells P1 and P2, while Fig. 6(c) shows the water rates for wells P3 and P4. Fig. 6(d) depicts the injection rates for wells I1 and I2. It is evident from these figures that the TPWL results, depicted by $\times$'s and circles, are in very close agreement with the reference high-fidelity (GPRS) solutions, depicted by solid and dashed curves. Though not shown, results of similar accuracy are achieved for the oil rates for wells P3 and P4 and the water rates for wells P1 and P2. Thus these results demonstrate the viability of the TPWL method for modeling subsurface flow.

We next consider test E. The TPWL model for test E would not be expected to be as accurate as in the previous case because the well BHPs vary over a larger range. In addition, these well BHPs are outside of the range of the training runs. Thus this is a challenging test case. Fig. 7(a) presents the oil rates for wells P1 and P2, while Fig. 7(b) shows the oil rates for wells P3 and P4. Although acceptable accuracy is achieved over most of the simulation, well P3 displays error in oil rate between 1000 and 1500 days. Fig. 7(c) shows the water production rates for wells P3 and P4. Some error in the water rate for well P3 is evident. Results of similar accuracy are observed for wells P1 and P2. Fig. 7(d) presents the injection rates for wells I1 and I2, which are seen to be quite accurate. This case illustrates that the TPWL model can lose some accuracy if the test runs involve well settings (and thus states) that are outside the range of the training runs. However, the magnitude of the discrepancy between the TPWL model and the high-fidelity simulation is not excessive, suggesting that the TPWL model is reasonably robust.

The average oil and water production rate errors, the average water injection rate error, and the maximum material balance errors for oil and water, for tests A–E, are presented in Table 1. From the results in this table, it is evident that the errors in production and injection rates increase consistently as we proceed from test A to test E (thus test C represents in some sense an 'average' result). For the mass balance errors, there is not a monotonic increase as we proceed from test A to test

Table 2
Errors in TPWL solutions for tests F–H.

	Test F	Test G	Test H
Oil prod.	0.0066	0.0095	0.0136
Water prod.	0.0165	0.0244	0.0520
Water inj.	0.0033	0.0051	0.0068
MBE oil	0.0001	0.0004	0.0016
MBE water	0.0039	0.0061	0.0177

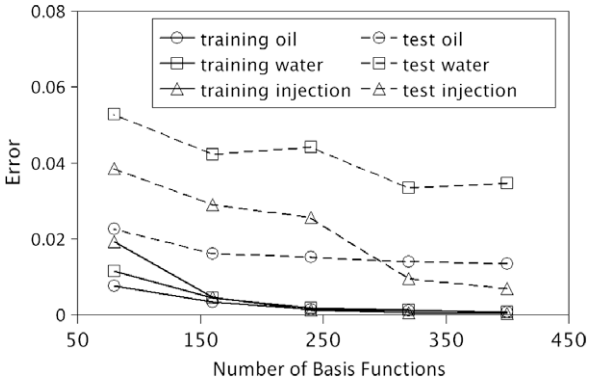

Fig. 9. Errors in the TPWL simulations for different numbers of basis functions.

E, though test E does provide the largest errors. The general level of the production and injection rate errors is relatively small, though we reiterate that they are average errors and significant instantaneous errors are observed in some cases.

We now consider the second set of test runs. Results for test G are shown in Fig. 8. Fig. 8(a) presents the BHP schedules prescribed for each production well, Fig. 8(b) shows the oil rates for wells P1 and P2, Fig. 8(c) depicts the water rates for wells P3 and P4, and Fig. 8(d) provides the injection rates for wells I1 and I2. Results for oil rates for P3 and P4 and water rates for P1 and P2 are comparable. For all of these results, the agreement between the TPWL and the high-fidelity models is quite close.

The average well rate errors and maximum material balance errors for tests F–H are presented in Table 2. It is evident that these errors increase consistently from test F to test H. The errors are generally comparable in magnitude to those presented in Table 1 for tests A–E, again indicating the overall level of accuracy and robustness of the TPWL model for this example.

The results above demonstrate that, for the cases considered, the TPWL model is able to provide simulation results in reasonable agreement with high-fidelity simulations, particularly when the controls (producer BHPs in our case) are kept within the range of the training simulations. The computational requirements for the high-fidelity (GPRS) and TPWL simulations are, however, considerably different. The GPRS simulations require about 430 s of CPU time, while the TPWL runs consume only about 0.85 s if we do not compute the material balance errors at every time step. This represents a runtime speedup of about 500. If we compute the material balance errors at every time step, the TPWL computations require about 4 s, which reduces the runtime speedup to about a factor of 100. It is not necessary to compute the material balance errors at every time step, however, and if they are computed relatively infrequently (e.g., every 10–20 time steps), the additional computation will not be excessive.

The speedups achieved by the TPWL model are significantly greater than those attained by previous POD-based reduced-order modeling approaches for this problem. Specifically, as indicated in the Introduction, POD-based ROMs [9–11] achieved at most about a factor of 10 speedup. The much larger speedups observed here are due to the fact that the TPWL model avoids many of the computations required by the POD-based ROM procedures, as discussed in Section 2.3.

As the computational costs associated with the TPWL model (both the runtime and preprocessing calculations) increase with increasing ℓ (number of columns in $\mathbf{\Phi}$), it is useful to assess the impact of ℓ on the accuracy and timing of the TPWL model. To quantify these effects, we perform TPWL simulations for the two training runs and the eight test runs using different sized $\mathbf{\Phi}$ matrices. Fig. 9 presents the average errors for oil and water production rates and water injection rates using different numbers of basis functions. For these computations, we use prescribed percentages (from 20% to 100%) of the ℓ_p and ℓ_s used in the results presented above (recall that for those results $\ell_p = 250$, $\ell_s = 150$). From the figure, it is evident that TPWL maintains reasonable accuracy in oil and water production and water injection rates, even with fairly small ℓ_p and ℓ_s. Runtime speedups for these models are as follows (material balance error is not computed): for $\ell = 400$, speedup is 500 (as indicated above); for $\ell = 320$, speedup is 642; for $\ell = 240$, speedup is 777; for $\ell = 160$, speedup is 873; and for $\ell = 80$, speedup is 995.

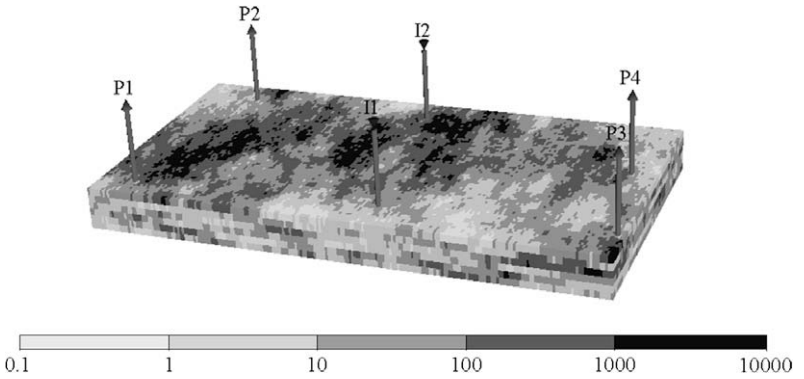

Fig. 10. Synthetic reservoir model containing 79,200 grid blocks with four production wells and two injection wells. Permeability in *x*-direction (in mD) is shown.

3.2. Reservoir model 2

3.2.1. Model description

We now consider a larger model in order to demonstrate the applicability of the TPWL representation for models approaching sizes encountered in practical reservoir simulation. Like the reservoir model used in the examples above, this system is derived from [33]. The model, again containing four production wells and two water injection wells, is of dimensions $60 \times 220 \times 6$, for a total of 79,200 grid blocks (see Fig. 10). All fluid properties are the same as those used previously except for the oil and water densities, which we prescribe as $\rho_o = \rho_w = 55$ lb/ft^3.

For the case of fluid densities that differ significantly, the accuracy of the overall method can be more sensitive to the number of basis functions (ℓ_p and ℓ_s) used in the construction of $\mathbf{\Phi}_p$ and $\mathbf{\Phi}_s$ than in the case of $\rho_o = \rho_w$ (J. He and J. Saetrom, private communication). This sensitivity is related to potential instabilities in the TPWL model, and these instabilities appear to increase with increasing model size. This general issue is discussed further in Section 4, where we present results for reservoir model 2 with $\rho_o \neq \rho_w$. In future work we will investigate the use of stabilized TPWL procedures (e.g., [22,23]) for cases with significant density differences.

3.2.2. Training and testing runs

For this reservoir model we perform two training runs, both with the injection wells prescribed to operate at a constant bottom hole pressure of 8000 psia. For the first training run the BHPs of the four production wells are held constant at 1000 psia. In the second training run the BHPs are varied randomly and independently over a range of 1000–4000 psia. The training runs were simulated for 5000 days with a maximum time step of 30 days. A total of 415 pressure and saturation snapshots and Jacobian matrices were recorded. The POD basis contains a total of 400 columns, 150 of which correspond to pressure and 250 to saturation.

We perform a total of five test simulations (designated I–M) using random producer well BHP schedules. In test I the production well BHPs range from 2000 to 3000 psia, for test J from 1500 to 3500 psia, for test K from 1000 to 4000 psia, for test L from 500 to 4500 psia, and for test M from 0 to 5000 psia.

3.2.3. Model 2 simulations

Fig. 11 demonstrates the ability of the TPWL representation to reproduce results for test K. Fig. 11(a) presents the BHP schedule for each production well, Fig. 11(b) shows the oil rate for wells P3 and P4, Fig. 11(c) depicts the water rate for wells P1 and P2 and Fig. 11(d) provides the injection rates for wells I1 and I2. TPWL results are generally quite accurate, though error in water rate in well P2 is evident just after the breakthrough.

Errors in the TPWL solutions for tests I through M are presented in Table 3. Recall that producer BHPs for tests I–K are within the range of the training runs, while those for tests L and M are outside of this range. The errors in Table 3 are overall higher, but generally of comparable magnitude, relative to those presented in Table 1 for tests A–E. Errors clearly trend higher as we proceed from test I to test M, as expected, though the increase is not strictly monotonic. Water production errors are fairly high for tests K–M. These results reiterate that the TPWL model provides accurate results within an appropriate range, but that solution accuracy can degrade outside of the range of the training runs.

The runtime speedups offered by TPWL relative to the high-fidelity simulations are even larger in this case than in the previous example. Specifically, the GPRS simulations for this model required about 30 min of CPU time, while each TPWL run required only 0.9 s if material balance errors were not computed. These timings correspond to a runtime speedup factor of about 2000. If material balance errors are computed at every time step, the TPWL runs require about an order of magnitude more computation, so the speedup relative to GPRS is reduced to about a factor of 200.

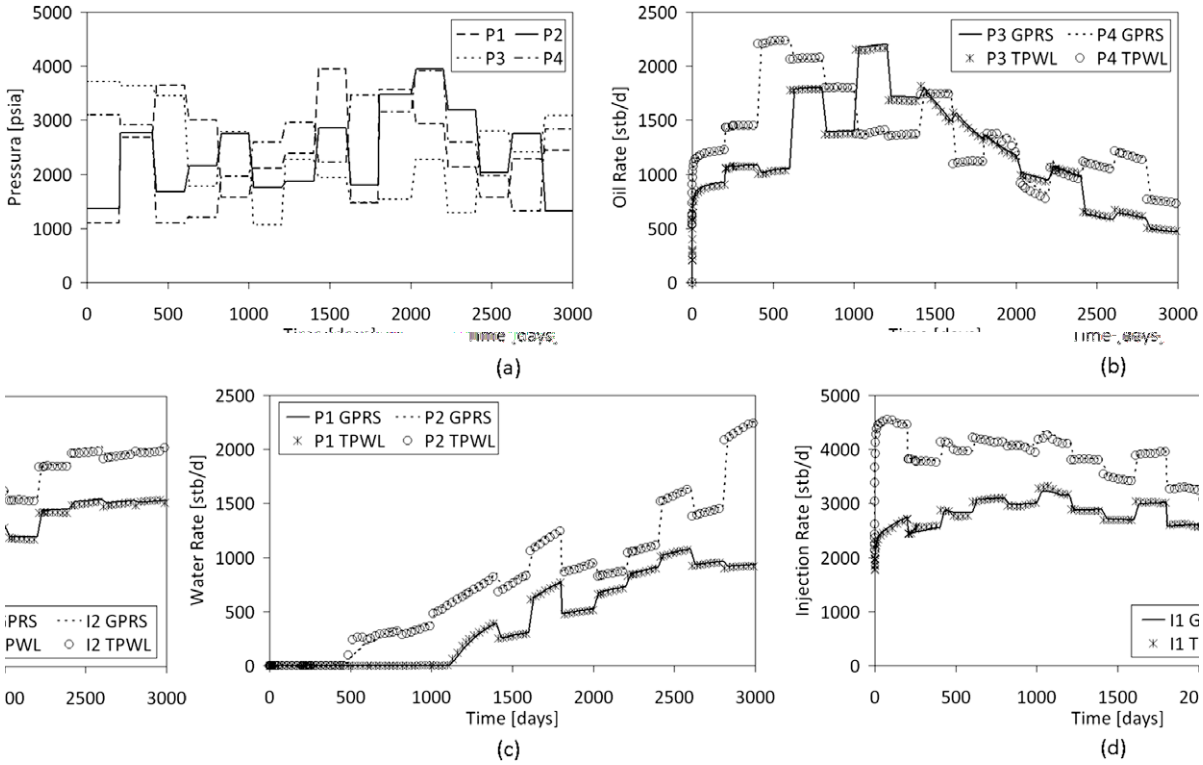

Fig. 11. Results for test K: (a) BHPs for each production well, (b) oil rate for production wells P3 and P4, (c) water rate for wells P1 and P2 and (d) injection rates for wells I1 and I2.

Table 3
Errors in TPWL solutions for tests I–M.

	Test I	Test J	Test K	Test L	Test M
Oil prod.	0.0097	0.0139	0.0145	0.0221	0.0404
Water prod.	0.0236	0.0211	0.0639	0.0946	0.1852
Water inj.	0.0040	0.0078	0.0118	0.0141	0.0214
MBE oil	0.0014	0.0012	0.0020	0.0029	0.0040
MBE water	0.0041	0.0036	0.0096	0.0104	0.0217

3.3. Optimization using TPWL

A target application for this TPWL method is as a fast (approximate) function evaluator in optimization procedures. Within the context of oil reservoir management, one is often interested in maximizing oil production or the net present value of the project. In another paper [34], we apply TPWL for single-objective optimization. In that case we are able to compare optimization results using the high-fidelity model with those using TPWL, and close agreement in optimized net present value is observed (even though the control sequences differ slightly between the two cases).

Here we address a more challenging multiobjective optimization in which we seek to maximize cumulative oil production while minimizing cumulative water injection. Thus the goal of the optimization is to determine the Pareto front on a plot of cumulative oil produced versus cumulative water injected. We address this problem by combining (using linear weighting) the two objectives into a single objective function, which we then maximize. By varying the relative weighting of the two terms, the convex portion of the Pareto front can be generated. Specifically, we maximize the net present value (NPV) of the reservoir over a period of 3000 days. The discount rate is 10% and the cost of both the produced and injected water is fixed at \$10/stb. The oil price is varied from \$10/stb to \$150/stb in increments of \$2/bbl, and the optimization is performed for each oil price. By varying oil price we vary the weighting of the two terms. The solution of this optimization problem is computationally demanding if the high-fidelity model is used because the optimization procedure must be run for each different oil price.

For this optimization, we consider reservoir model 1 (containing 24,000 grid blocks) and optimize the BHPs of the four producer wells. The minimum and maximum producer BHPs are prescribed as 1000 psia and 4000 psia respectively. These

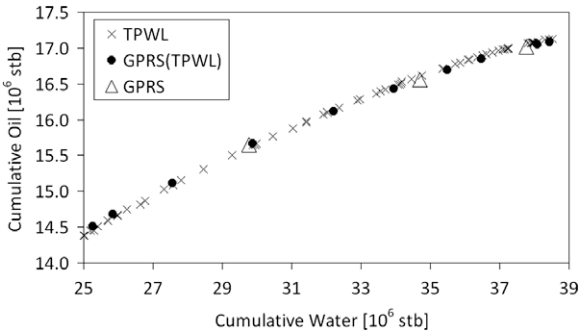

Fig. 12. Optimized cumulative oil production versus cumulative water injected.

Table 4
Errors in TPWL solutions (no reduction) for tests I–M.

	Test I	Test J	Test K	Test L	Test M
Oil prod.	0.0041	0.0033	0.0067	0.0153	0.0148
Water prod.	0.0203	0.0131	0.0337	0.0686	0.0755
Water inj.	0.0014	0.0012	0.0013	0.0056	0.0050
MBE oil	0.0007	0.0005	0.0009	0.0028	0.0024
MBE water	0.0018	0.0017	0.0031	0.0047	0.0178

BHPs are updated every 250 days. As the simulation is run for 3000 days, there are a total of 48 control variables (12 for each production well). We further specify that the maximum change in BHP from one control step to the next is 250 psia. For the optimization, we use a gradient-based procedure ("fmincon" in Matlab [32]), with the gradients computed using numerical finite difference. A total of four iterations of the optimization are performed, meaning that each point on the curve requires 196 simulations (a few more simulations may be required in the line search).

The TPWL model is formed as follows. In the first training run, the BHPs of all production wells are kept constant at 1000 psia. For the second training run, we use the random case applied in Section 3.1.2 (depicted in Fig. 3(b)). A reduced basis $\mathbf{\Phi}$ is then constructed with $\ell_p = 250$ and $\ell_s = 150$.

The optimization results are presented in Fig. 12, where we plot cumulative oil produced versus cumulative water injected. The $\times$'s represent TPWL solutions (71 points were generated), while the solid circles represent high-fidelity (GPRS) results computed using the optimal well settings as determined from the TPWL optimizations. For a few points (shown as triangles) we perform the full optimization using the high-fidelity model. The general correspondence between these three sets of points is quite close, indicating the applicability of the TPWL procedure for this application.

The speedup achieved by TPWL for this case is very significant. A total of about 13,916 simulations are required for this optimization. The TPWL computations required 0.85 s per run, while the high-fidelity simulations consumed 430 s per run. Thus the full optimization entailed 3.3 h using TPWL. The TPWL overhead, which is the equivalent of about four high-fidelity simulations, required about 0.5 h (so total TPWL computation time was about 3.8 h). We did not perform the full optimization using the high-fidelity model – had we done this, it would have required about 69.3 days.

4. Additional issues

The results presented in the previous section demonstrate the potential of the TPWL method for subsurface flow modeling. Reasonable accuracy was achieved for two realistic example cases for test simulations in which the production well controls (BHPs) were quite different from those used in the training simulations. The method was also shown to be applicable for use in production optimization problems. There remain, however, several issues that must be addressed before the TPWL procedure can be used for a wide range of practical subsurface flow problems.

4.1. Linearization and projection errors

Because the method is based on linearization around previously simulated (saved) states, degradation in accuracy is to be expected when higher-order terms in the expansion (Eq. (13)) become significant. Error also arises from the projection into reduced space and the reconstruction in physical space. We can assess the relative contributions of these two errors by running the linearized model without applying any reduction, which corresponds to solving Eq. (21) or (23). Solution of these equations is time consuming and thus not very useful for practical simulations, but it does elucidate the sources of TPWL model error.

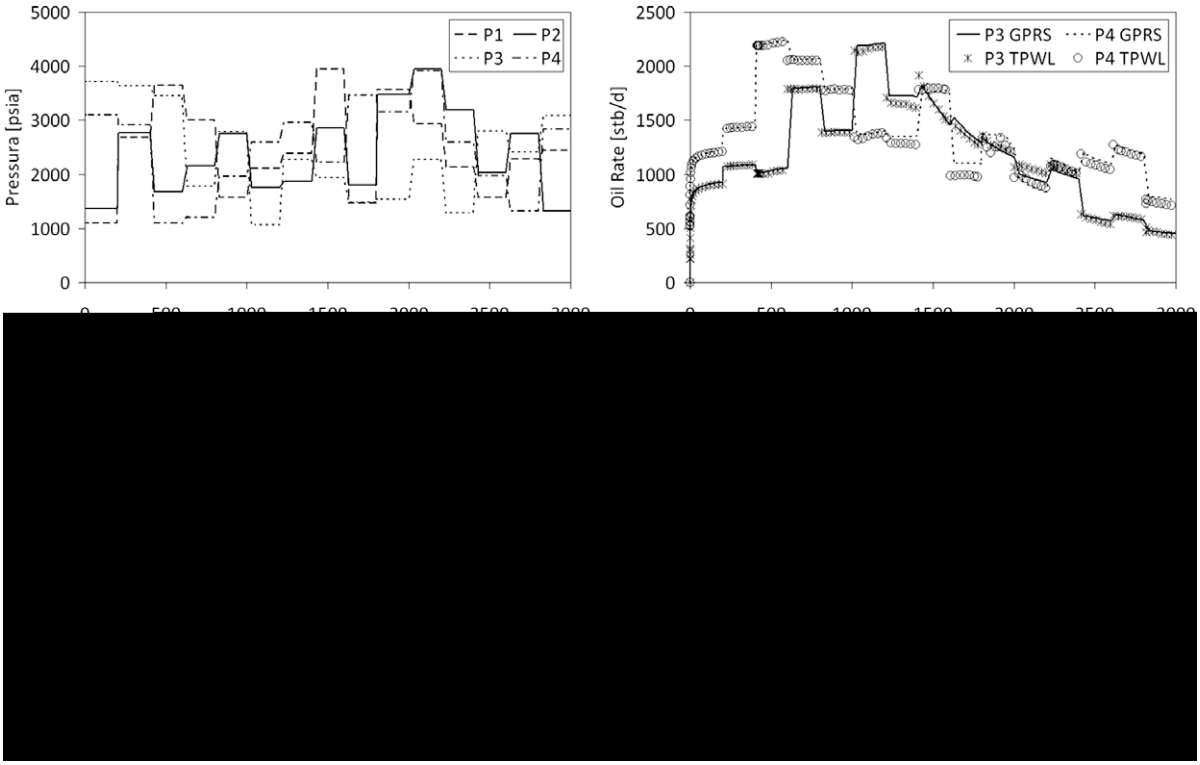

Fig. 13. Results for test K with density difference: (a) BHPs for each production well, (b) oil rate for production wells P3 and P4, (c) water rate for wells P1 and P2 and (d) injection rates for wells I1 and I2.

We performed TPWL simulations with no reduction (by solving Eq. (23)) for all test runs for both reservoir models. Representative results, for test runs I–M (reservoir model 2), are presented in Table 4. Comparison of the errors in Table 4 to the errors shown earlier for the same test runs with reduction (Table 3) demonstrate that model accuracy is in all cases higher when reduction is not applied, as would be expected. For some quantities the errors are significantly less for the non-reduced model (e.g., water production rate error for test M, which decreases from 0.1852 to 0.0755), though in other cases the errors are quite close. These results, in addition to those for test runs A–H, indicate that some of the error in the TPWL results shown in Section 3 derives from the linearization but that a significant portion of the error arises from projection and reconstruction. Thus improved TPWL procedures should target both of these sources of error.

In practical settings, it will be useful to develop efficient estimators for these errors as well as heuristics that indicate when the TPWL solution is likely to degrade. Preliminary results suggest that empirical error estimates that relate error in oil and water production rates and injection rate to mass balance error and some measure of the distance between the saved state $\mathbf{x}^{i+1}$ and the computed state $\mathbf{x}^{n+1}$ (both of which can be readily computed) are of value in this regard, though further quantification is necessary.

4.2. Stability issues for systems with density differences

As noted in Section 3.2.1, TPWL accuracy for reservoir model 2 (containing 79,200 cells) degrades when strong density differences are included. We illustrate this effect by considering a case that is identical to reservoir model 2 except now we specify ρ_o=45 lb/ft^3 and ρ_w=60 lb/ft^3 (in the earlier results we used $\rho_o = \rho_w$= 55 lb/ft^3). The training runs are identical to those used in the earlier results. TPWL results for this case for test run K are shown in Fig. 13. Errors are clearly larger in this case than for the case of equal densities (Fig. 11). Larger errors can be observed in more extreme cases; e.g., for controls that are outside the range of the training runs or for greater density differences. We also observe increased sensitivity of TPWL model accuracy to the number of basis functions (ℓ_p and ℓ_s) used in the construction of $\mathbf{\Phi}_p$ and $\mathbf{\Phi}_s$ for cases with significant density differences.

The different behaviors observed for equal versus unequal densities appear to be related to the stability of the linearized representation. A preliminary investigation (J. He and J. Saetrom, private communication) shows that large TPWL errors are associated with local instabilities in the TPWL equation. Specifically, defining an appropriate amplification matrix based on Eq. (27) (or Eq. (23) in the case of no model reduction), we observe that local 'spikes' in TPWL model error appear when the spectral radius of the amplification matrix exceeds 1 (which corresponds to an unstable representation). Instability in

reduced-order models has been noted and discussed by previous investigators (e.g., [35,22,23]) for problems in other application areas.

The different characteristics of the TPWL models for the two problems likely arise from the more complex physics that appear in cases with unequal densities. These cases display more localized (high-frequency) variability in the saturation field, which occurs as a result of the local competition between convective and gravitational effects. The saturation fields in this case may resemble discrete random fields, in contrast to the smoother fields observed for the equal density case. In addition, for the equal density case oil and water always flow in the same direction, while for unequal densities counter-current flow can occur in the vertical direction. This modifies the structure of the Jacobian matrix (due to upstream weighting). Thus there are key physical and numerical differences between the two cases, and in future work we will attempt to relate the observed differences in stability to these effects.

Other researchers have addressed ROM and TPWL stability issues that arise in other application areas, and they have suggested a number of remedies which we plan to investigate. These include the use of Fourier model reduction [35] and improved TPWL weighting functions and projection schemes [22,23]. Along these lines, in preliminary tests we have observed improved results using basis switching and the optimization of ℓ_p and ℓ_s. Our results also suggest that some amount of retraining, to assure that the test simulations are sufficiently 'close' to training runs, will yield improved results. The goal-oriented optimization procedure for constructing the basis matrix presented in [36] and the training scheme suggested in [37] may be useful in this regard.

4.3. Optimization and additional applications

Within the context of optimization, error estimators could be used to determine when model retraining is required. Retraining could then be accomplished by performing one or more new high-fidelity simulations using control schedules consistent with those at the current stage of the optimization. This would provide additional saved states which should be closer to those encountered during subsequent stages of the optimization. Overall speedups will not, however, be as substantial if retraining is performed. A related approach involving a two-stage optimization, in which variable injector pressures were also considered, is presented in [34]. In that application, the TPWL model was trained using two (heuristic) training simulations and the optimization was performed. The model was then retrained using the 'optimal' settings determined in this first stage and the optimization was repeated. This approach was shown to provide optimization results and well settings in reasonably close overall agreement with those from the high-fidelity model.

Finally, it will be of interest to extend the TPWL technique to more complex subsurface flow problems. This could include the modeling of thermal operations which, like compositional systems, are often very expensive to simulate. For such cases, the development of reduced-order models that retain a reasonable level of accuracy could enable the application of computational production optimization for realistic problems.

5. Concluding remarks

In this paper we developed a trajectory piecewise linearization (TPWL) procedure for the modeling of two-phase flow in subsurface formations. This reduced-order modeling technique entails representation of new states in terms of linear expansions around states previously simulated and saved during a series of preprocessing training runs. The method is efficient because all computations are performed in a low-dimensional space determined through proper orthogonal decomposition of the pressure (or oil potential) and saturation states encountered during the training simulations.

We applied the method to two models, containing 24,000 and 79,200 grid blocks, derived from the SPE 10 reservoir description. We observed reasonable levels of accuracy in TPWL computations for oil and water production rates and water injection rates for test cases in which the production well bottom hole pressures were within the general range of the training simulations. For such cases (tests A–C, I–K), average relative errors in oil and water production rates were at most 0.0147 and 0.0639 respectively. The TPWL model is not mass conservative, though for tests A–C and I–K, maximum mass balance errors were 0.0020 or less for oil and 0.0096 or less for water. As expected, when the production well BHP schedule in the test runs differed significantly from those of the training runs, the accuracy of the TPWL model degraded. In many cases error was still acceptable, though large errors were observed for some quantities (e.g., water production rate in test M).

The preprocessing overhead required for the construction of the TPWL model corresponds to about the time required to simulate four high-fidelity models. The TPWL model provided runtime speedups of about 500-2000 for the cases considered if mass balance error was not computed. Speedups will be less dramatic if mass balance error is assessed frequently (though in practice this error need be computed only occasionally) or if retraining is performed.

We also applied the TPWL procedure to a multiobjective optimization in which the goal was to maximize cumulative oil production while minimizing cumulative water injection. TPWL results for the Pareto front were in agreement with limited high-fidelity computations, demonstrating the potential applicability of the TPWL procedure for computationally demanding optimization problems. Future work should address issues relating to the efficient estimation of TPWL model error, the development of a stabilized TPWL model for cases with different phase densities, and retraining of the TPWL model during the course of the optimization.

Acknowledgments

We are grateful to Huanquan Pan for modifying Stanford's general purpose research simulator (GPRS) to output information required by the TPWL representations and to David Echeverria Ciaurri for useful discussions and suggestions regarding Matlab optimization routines. We also thank Jincong He and Jon Saetrom for insightful comments and for providing us with preliminary results from their investigation of the performance of TPWL for problems with strong density differences. Finally, we thank two anonymous reviewers for their constructive suggestions on the initial version of this paper.

References

[1] J.L. Lumley, Atmospheric turbulence and radio wave propagation, Journal of Computational Chemistry 23 (13) (1967) 1236–1243.
[2] P. Astrid, A. Verhoeven, Application of least squares MPE technique in the reduced order modeling of electrical circuits, in: 17th International Symposium on Mathematical Theory of Networks and Systems, Kyoto, Japan, 2006.
[3] D. Zheng, K.A. Hoo, M.J. Piovoso, Low-order model identification of distributed parameter systems by a combination of singular value decomposition and the Karhunen-Loève expansion, Industrial & Engineering Chemistry Research 41 (6) (2002) 1545–1556.
[4] Y. Cao, J. Zhu, Z. Luo, I.M. Navon, Reduced order modeling of the upper tropical Pacific ocean model using proper orthogonal decomposition, Computers & Mathematics with Applications 52 (8–9) (2006) 1373–1386.
[5] T. Bui-Thanh, M. Damodaran, K. Willcox, Aerodynamic data reconstruction and inverse design using proper orthogonal decomposition, AIAA Journal 42 (8) (2004) 1505–1516.
[6] M. Meyer, H.G. Matthies, Efficient model reduction in non-linear dynamics using the Karhunen-Loève expansion and dual-weighted-residual methods, Computational Mechanics 31 (2003) 179–191.
[7] S.S. Ravindran, Adaptive reduced-order controllers for thermal flow system using proper orthogonal decomposition, SIAM Journal of Scientific Computing 23 (6) (2002) 1924–1942.
[8] P.T.M. Vermeulen, A.W. Heemink, C.B.M.T. Stroet, Reduced models for linear groundwater flow models using empirical orthogonal functions, Advances in Water Resources 27 (2004) 57–69.
[9] J.F.M. van Doren, R. Markovinović, J.D. Jansen, Reduced-order optimal control of water flooding using proper orthogonal decomposition, Computational Geosciences 10 (2006) 137–158.
[10] T. Heijn, R. Markovinović, J.D. Jansen, Generation of low-order reservoir models using system-theoretical concepts, SPE Journal 9 (2) (2004) 202–218.
[11] M.A. Cardoso, L.J. Durlofsky, P. Sarma, Development and application of reduced-order modeling procedures for subsurface flow simulation, International Journal for Numerical Methods in Engineering 77 (9) (2009) 1322–1350.
[12] J. Burkardt, M. Gunzburger, H.C. Lee, Centroidal Voronoi Tessellation-based reduced-order modeling of complex systems, SIAM Journal of Scientific Computing 28 (2) (2006) 459–484.
[13] P. Astrid, Reduction of process simulation models: a proper orthogonal decomposition approach, Ph.D. Thesis, Eindhoven University of Technology, 2004.
[14] M.J. Rewienski, A trajectory piecewise-linear approach to model order reduction of nonlinear dynamical systems, Ph.D. Thesis, Massachusetts Institute of Technology, 2003.
[15] M. Rewienski, J. White, 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 22 (2) (2003) 155–170.
[16] D. Gratton, K. Willcox, Reduced-order, trajectory piecewise-linear models for nonlinear computational fluid dynamics, in: 34^{th} AIAA Fluid Dynamics Conference and Exhibit, Portland, Oregon, USA, 2004.
[17] Y.J. Yang, K.Y. Shen, Nonlinear heat-transfer macromodeling for MEMS thermal devices, Journal of Micromechanics and Microengineering 15 (2) (2005) 408–418.
[18] D. Vasilyev, M. Rewienski, J. White, 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 25 (2) (2006) 285–293.
[19] S.K. Tiwary, R.A. Rutenbar, Faster, parametric trajectory-based macromodels via localized linear reductions, in: ICCAD '06: Proceedings of the 2006 IEEE/ACM International Conference on Computer-aided Design, 2006.
[20] N. Dong, J. Roychowdhury, General-purpose nonlinear model-order reduction using piecewise-polynomial representations, IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems 27 (2) (2008) 249–264.
[21] M.N. Albunni, V. Rischmuller, T. Fritzsche, B. Lohmann, Model-order reduction of moving nonlinear electromagnetic devices, IEEE Transactions on Magnetics 44 (7) (2008) 1822–1829.
[22] B.N. Bond, L. Daniel, Stabilizing schemes for piecewise-linear reduced order models via projection and weighting functions, in: 2007 IEEE/ACM International Conference on Computer-Aided Design, San Jose, California, USA, 2007.
[23] B.N. Bond, L. Daniel, Guaranteed stable projection-based model reduction for indefinite and unstable linear systems, in: 2008 IEEE/ACM International Conference on Computer-Aided Design, San Jose, California, USA, 2008.
[24] K. Aziz, A. Settari, Fundamentals of Reservoir Simulation, Elsevier Applied Science Publishers, 1986.
[25] M.G. Gerritsen, L.J. Durlofsky, Modeling fluid flow in oil reservoirs, Annual Review of Fluid Mechanics 37 (2005) 211–238.
[26] L.J. Durlofsky, A triangle based mixed finite element-finite volume technique for modeling two phase flow through porous media, Journal of Computational Physics 105 (2) (1993) 252–266.
[27] P. Jenny, S.H. Lee, H.A. Tchelepi, Adaptive fully implicit multi-scale finite-volume method for multi-phase flow and transport in heterogeneous porous media, Journal of Computational Physics 217 (2) (2006) 627–641.
[28] D.W. Peaceman, Interpretation of well-block pressures in numerical reservoir simulation with nonsquare grid blocks and anisotropic permeability, SPE Journal 23 (3) (1983) 531–543.
[29] Y. Jiang, A flexible computational framework for efficient integrated simulation of advanced wells and unstructured reservoir models, Ph.D. Thesis, Stanford University, 2007.
[30] M.A. Cardoso, Development and application of reduced-order modeling procedures for reservoir simulation, Ph.D. Thesis, Stanford University, 2009.
[31] H. Cao, Development of techniques for general purpose simulators, Ph.D. Thesis, Stanford University, 2002.
[32] MathWorks, The MathWorks, Inc., <http://www.mathworks.com>, 2008.
[33] M.A. Christie, M.J. Blunt, Tenth SPE comparative solution project: a comparison of upscaling techniques, SPE Reservoir Evaluation & Engineering (4) (2001) 308–317.
[34] M.A. Cardoso, L.J. Durlofsky, Use of reduced-order modeling procedures for production optimization (SPE paper 119057), SPE Journal, in press.
[35] K. Willcox, A. Megretski, Fourier series for accurate, stable, reduced-order models in large-scale linear applications, SIAM Journal on Scientific Computing 26 (3) (2005) 944–962.
[36] T. Bui-Thanh, K. Willcox, O. Ghattas, B. van, B. Waanders, Goal-oriented, model-constrained optimization for reduction of large-scale systems, Journal of Computational Physics 224 (2) (2007) 880–896.
[37] B.N. Bond, L. Daniel, A piecewise-linear moment-matching approach to parameterized model-order reduction for highly nonlinear systems, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 26 (12) (2007) 2116–2129.