Journal of Petroleum Science and Engineering 211 (2022) 110175

Contents lists available at ScienceDirect

Journal of Petroleum Science and Engineering

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

Application of physics informed neural networks to compositional modeling

Thelma Anizia Ihunde, Olufemi Olorode *

Louisiana State University, USA

ARTICLE INFO

Keywords:
Physics-informed neural network
Compositional modeling
Artificial intelligence
Phase equilibrium calculations
Physics-constrained deep learning

ABSTRACT

Compositional modeling is essential when simulating any process that involves significant changes in the composition of reservoir fluids. This includes modeling the flow of multicomponent hydrocarbons in pipes, surface facilities, and subsurface rocks. However, the rigorous thermodynamics approach to obtain phase composition is computationally expensive. So, various researchers have considered using machine learning models trained with rigorous phase-equilibrium (flash) calculations to improve computational speed.

Unlike previous publications that apply classical deep learning (DL) models to flash calculations, this work will demonstrate the first attempt to incorporate thermodynamics constraints into the training of these models to ensure that they honor physical laws. To this end, we generated one million different compositions with a space-filling mixture design and performed two-phase flash to obtain the corresponding phase compositions. We performed seven-fold cross-validation to ensure reliable estimates of model accuracy. We compared the physics-constrained and standard DL model results to quantify the ability of our approach to honor physical constraints.

The evaluation of our physics-informed neural network (PINN) model compared to a standard DL model shows that we can incorporate physical constraints without a considerable reduction in model accuracy. Based on the test data, our model evaluation results indicate that both PINN and standard DL models achieve coefficients of determination of 97%. In contrast, the root-mean-square error of the physics-constraint errors in the PINN model is at least two times smaller than in the standard DL model. To further demonstrate that our PINN model outperforms the DL model in terms of honoring physical constraints, we generate phase envelopes using the overall compositions predicted using the PINN and DL models for several fluid mixtures in the test data. These results show the importance of incorporating the thermodynamic constraints into DL models.

1. Introduction

Compositional modeling involves stability analysis, where the stability of a fluid mixture in a single-phase state is determined at a given pressure and temperature. Next, phase equilibrium calculations (also known as flash) are performed to determine the mole fractions of the fluid components in the liquid and vapor phases (for two-phase flash). These calculations are needed in every representative elementary volume (REV) and time step during compositional fluid flow simulation in porous media (Coats, 1980; Pal and Mandal, 2021; Young and Stephenson, 1983) and in pipes (Furukawa et al., 1986; Gould, 1979). Therefore, various researchers have proposed iterative and non-iterative approaches to accelerate flash calculations in compositional reservoir simulation. Some of the iterative techniques include the minimization of Gibb's free energy (Nichita et al., 2002), dimensionality reduction (Firoozabadi and Pan, 2000; Nichita and Graciaa, 2011; Pan & Firoozabadi, 2001), and solving the Rachford-Rice equation by minimizing a non-monotonic convex function (Okuno et al., 2010). Some non-iterative techniques include interpolating data from look-up tables (Voskov and Tchelepi, 2009; Belkadi et al., 2011; Wu et al., 2015).

In recent years, researchers have leveraged machine learning (ML) advancements to improve the speed and accuracy of compositional simulation (Gaganis and Varotsis, 2012, 2014). This has been accomplished primarily through the use of supervised ML algorithms such as support vector machines (SVMs) (Gaganis and Varotsis, 2012, 2014), relevance vector machines (RVMs) (Kashinath et al., 2018), and deep neural networks (DNNs) (Kashinath et al., 2018, K. Wang et al., 2019a, 2019b; Li et al., 2019; S. Wang et al., 2019a, 2019b). Considering that the use of trained DNNs to predict flash calculation results has been shown to be over two order of magnitudes faster than the standard

* Corresponding author.
E-mail address: folorode@lsu.edu (O. Olorode).

https://doi.org/10.1016/j.petrol.2022.110175
Received 29 September 2021; Received in revised form 21 December 2021; Accepted 12 January 2022
Available online 15 January 2022

iterative flash procedure (Li et al., 2019), Wang et al. (2020) showed that it yields an excellent match when implemented in compositional reservoir simulators. Although these supervised ML algorithms have been used to create accurate proxy models that are trained with data from rigorous compositional models, it is essential to note that these algorithms can generate predictions that do not honor the underlying physics for phase equilibrium. This is because the loss function in these ML models is simply a function of the difference between the training data and the model prediction of phase mole fractions (x_i and y_i) and liquid fractions. So, there is a considerable probability of generating model predictions that have low loss function values but do not yield the given overall mole fractions when combined according to the interphase mass balance equation.

To address the challenge of honoring physical laws during the training of DL models, several authors have developed frameworks that enable the integration of physics into deep learning. The most common approach involves adding governing partial differential equations (PDEs), initial and boundary conditions as penalties in the loss function (Raissi and Karniadakis, 2018; Raissi et al., 2019; and Haghighat and Juanes, 2021). Others have also demonstrated the potential of implementing PDE constraints as extensions to the computational graph for the deep neural network (Huang et al., 2020; Xu and Darve, 2020). However, no publication has incorporated physical constraints into the machine learning models trained with data from two-phase flash. To address this limitation, we presented the initial attempt of incorporating physical constraints into DL models for two-phase flash calculations in Ihunde and Olorode (2021).

Although deep learning has been applied successfully to various problems such as computer vision and image classification, autonomous driving, speech recognition, and recommender systems, its application to physics or engineering problems is still relatively new. Considering that PDEs govern several physical processes, it is easy to see why most of the previous publications on PINNs focus on incorporating PDEs, initial and boundary conditions into the training of these models. However, a few authors (Daw et al., 2021; Kashinath et al., 2021) have presented the use of PINNs to include other physical laws that are not PDEs. The critical requirement is that the physics-based equations added to the standard or empirical loss function are continuous and differentiable (Daw et al., 2021). This work evaluates the feasibility of including thermodynamics constraints into DL models via the penalty approach. We used this approach instead of the other techniques that enforce physical constraints via custom neural network architectures (Beucler et al., 2019; Jiang et al., 2020) because it is efficient and straightforward to implement and interpret. Although we refer to this approach as PINN, it is also commonly referred to as the physics-constrained deep learning (PCDL) or physics-guided neural network (PGNN) model.

This paper summarizes the procedure for phase equilibrium calculations, which is used to generate the data for training the PINN and DL models. Next, we discuss deep learning and how we incorporate physics into these models. Finally, we conclude with a discussion of our PINN model results compared to the standard DL model results.

1.1. Phase equilibrium (flash) calculations

The conventional approach to solve phase-equilibrium equations involves satisfying the equality of fugacity, interphase mass balance, and component balance constraints. This is achieved using the successive substitution and/or the Newton-Raphson iteration. Given a fluid mixture with N components, the cubic equations of state (EOS) is used to describe its phase behavior. The model parameters of the EOS include the critical temperature (T_c), critical pressure (P_c), acentric factor (ω), and molecular weight (M) of each hydrocarbon component (Firoozabadi, 2016). The binary interaction coefficient (bic) between each pair of hydrocarbon components is also a required model parameter.

For a two-phase liquid/vapor system, the pressure (P), temperature (T), and overall mole fraction of each fluid component (z_i) completely define the state of the fluid mixture. The output of the flash calculation includes the liquid fraction (L) or vapor fraction (V), and the mole fractions of the liquid and vapor phases, which are x_i and y_i, respectively.

1.1.1. Equal fugacity constraint

At chemical equilibrium, the chemical potential of each hydrocarbon component in the liquid phase must be the same as that in the vapor phase. Considering that fugacity is a proxy for chemical potential, chemical equilibrium is achieved when the fugacity of each component in the liquid phase is the same as that in the vapor phase. For components $i = 1$ ...., N, the equal fugacity constraint is given as:

$$f_{Li}(x_i, P_L, T) = f_{Vi}(y_i, P_V, T), \quad [1]$$

where f_{Li} and f_{Vi} are the fugacities of each hydrocarbon component in the liquid and vapor phases, whereas P_L and P_V are the pressures of the liquid and vapor phases. When this constraint is satisfied, the chemical potential of each fluid component in both phases is the same. This work computes the fugacities as detailed in Peng and Robinson (1976) and Firoozabadi (2016)

1.1.2. Interphase mass balance constraint

The overall mole fraction of each component (z_i) can be written as a function of the phase mole fractions (x_i and y_i) and vapor fraction as follows:

$$z_i = Vy_i + (1 - V)x_i. \quad [2]$$

By combining the definition of the vapor-liquid equilibrium factor ($k_i = y_i/x_i$) with Eq. [2], we obtain the following expressions for the phase mole fractions:

$$x_i = \frac{z_i}{V(k_i - 1) + 1}, \quad [3]$$

$$y_i = \frac{z_i k_i}{V(k_i - 1) + 1} = k_i x_i. \quad [4]$$

The Wilson's correlation (Wilson, 1968) gives the initial K-values (k_i) to be used in the iterative solution of the Rachford-Rice (RR) equation.

Combining Eqs. [3] and [4], we obtain the Rachford-Rice (RR) equation:

$$RR = \sum_{i=1}^{N} (y_i - x_i) = \sum_{i=1}^{N} \frac{z_i(k_i - 1)}{V(k_i - 1) + 1} = 0. \quad [5]$$

Solving the Rachford-Rice (RR) equation yields the vapor mole fraction, V.

1.1.3. Component balance constraint

The component balance constraint is the final constraint required in phase-equilibrium calculations. It ensures that the overall mole fractions and phase mole fractions of every component sum up to one. That is,

$$\sum_{j=1}^{n_c} x_j = 1, \quad \sum_{j=1}^{n_c} y_j = 1, \quad \sum_{j=1}^{n_c} z_j = 1, \quad [6]$$

where n_c is the total number of components in the mixture.

1.2. Generation of training and test data

To generate the data that is used to train the machine learning models presented in this work, we employ the two-phase flash algorithm summarized in Fig. 1. This algorithm starts with a Michelsen stability test (Michelsen, 1982a, 1982b) at the given pressure and temperature to determine if the fluid mixture is stable in the single-phase state or not. If the fluid is stable in the single-phase state, there is no need to compute the phase mole fractions from the Rachford-Rice equation. Otherwise,

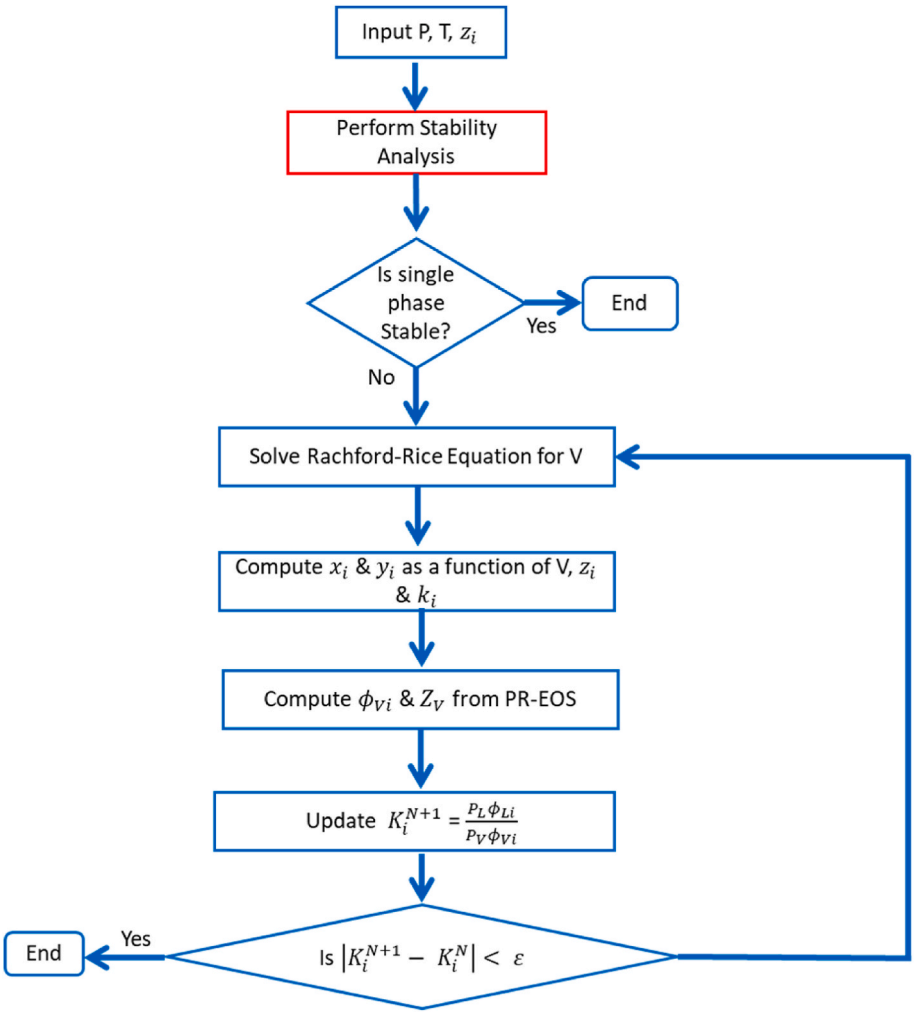

Fig. 1. Flow chart illustrates the algorithm for two-phase equilibrium calculations.

we solve the RR equation using the successive substitution and/or Newton-Raphson's method(s) to obtain the vapor fraction. We then estimate the phase mole fractions from Eqs. [3] and [4], using an initial estimate of K-value from Wilson's correlation (Wilson, 1968). Finally, we update the K-value and repeat the process of solving the RR equation until the change in the K-value between two successive iterations becomes negligible. The final value of the phase mole-fractions is taken as the x_i and y_i values.

Although the approach presented is applicable regardless of the number of components, this work consists of one million different three-component fluid mixtures, which are generated using a space-filling mixture design. The MATLAB Reservoir Simulation Toolkit (MRST) (Lie, 2019) is used to perform stability analysis and isothermal two-phase flash, which yields the corresponding phase mole fractions and vapor fraction at 100 different pressures between 14.7 psi to 5000 psi. A smaller subset of the data is used to train, validate, and test the models built in this work. Table 1 shows the first six rows of a randomly selected subset of the data. The first four columns (unshaded columns) represent the overall composition and pressure, which are the input variables for the flash calculations. The temperature is not provided in the table because this work focuses on isothermal applications as proof of the concept. So, we kept the temperature of all fluid mixtures at a constant value of 353 K (176 °F). The grey-shaded columns in the table represent the output variables obtained from the two-phase flash computations. This work aims to train deep neural networks (with and without physics constraints) using the input data (white columns in Table 1) to predict the output data. The subscripts 1, 3, and 14 in the overall and phase compositions indicate that the fluids studied are based on three-component mixtures, which consist of methane (C_1), propane (C_3), and tetradecane (C_{14}).

The one million fluid mixtures generated were split in ratio 70:15:15 for training, validation, and testing, respectively. The training and

Table 1
Excerpt from the dataset used to train the ML models.

$z_{methane}$	$z_{propane}$	z_{C14}	Pressure (psi)	Vapor Fraction	$x_{methane}$	$x_{propane}$	x_{C14}	$y_{methane}$	$y_{propane}$	y_{C14}
0.030	0.852	0.118	4345	0	0.030	0.852	0.118	0	0	0
0.245	0.397	0.358	1626	0	0.245	0.397	0.358	0	0	0
0.903	0.094	0.003	367	1	0	0	0	0.903	0.094	0.003
0.684	0.315	0.001	1676	1	0	0	0	0.684	0.315	0.001
0.686	0.137	0.177	820	0.297	0.592	0.157	0.251	0.909	0.088	0.004
0.613	0.220	0.168	1072	0.645	0.217	0.311	0.472	0.831	0.169	0.000

validation datasets are used during the training process while the test dataset is withheld and unseen by the models. To generate a reliable generalized model, the data used to train the classification model needs to have a balanced class distribution of the three possible phase states—liquid, vapor, and two-phase. This is important because a model trained on an imbalanced dataset will preferentially perform better on the class with the highest number of observations.

1.3. Deep neural networks (DNN)

This section provides a summary of the key components of DNNs. For a more detailed introduction, the reader is referred to classical textbooks (Goodfellow et al., 2016; Trask, 2019; Weidman, 2019) on this active field of research. Here, we start by defining a neural network as a nonparametric machine learning model that can be trained on any nonlinear data. It is a collection of nodes or units that are connected by a directed link. For example, a link from unit i to unit j propagates an activation, a_i from i to j. The activation a_i is equivalent to input, x_i in the case of an external input activation and y_j in the case of an external output activation. Each link's numerical weight ($w_{i,j}$) determines the strength and sign of the connection between the pair of units.

It is worth noting that each unit has a dummy input ($a_0 = 1$) with a corresponding weight ($w_{0,j}$), which is referred to as the bias. We compute the output of each unit by applying an activation function (σ) to the weighted sum of its inputs as follows:

$$a_j = \sigma\left(\sum_{i=0}^{n} w_{i,j} a_i\right). \quad [7]$$

The activation function could be a rectified linear unit (ReLU), a hard threshold, hyperbolic tangent, or a sigmoid function. This gives the neural network the flexibility to represent any nonlinear function. The activation function used in the output unit also ensures that the output phase mole fractions range between zero and one. For a neural network with m layers, the activation a_j from Eq. [7] can be combined with other activations in a feed-forward network to obtain the final output of the network:

$$\widehat{y_j} = \sum^{m} \circ \sum^{m-1} \circ \ldots \circ \sum^{1} \circ \sum^{0} (a_j), \quad [8]$$

where

$$\sum^{i} (a_j) = \sigma\left(\sum_{i=0}^{n} w_{i,j} a_i\right). \quad [9]$$

To train this multilayer neural network, we use the gradient of the loss function to update the weights of all the units in all the layers $i = 1 \ldots m$. This is achieved using a backpropagation algorithm that minimizes the loss function (L). The loss function used in this work is the mean-squared error (MSE), which is given as:

$$L(\mathbf{w}) = \frac{1}{N} \sum \left(y_j - \widehat{y_j} \right)^2, \quad [10]$$

where y_j is the actual output value and $\widehat{y_j}$ is the predicted output value. To minimize this function using a gradient-based (or stochastic gradient) method, we need to find the function's gradient with respect to the weights. This is achieved as follows (Russell and Norvig, 2016):

$$\frac{\partial L_k}{\partial w_{i,j}} = -2(y_k - a_k)\frac{\partial a_k}{\partial w_{i,j}} = -2 \triangle_k w_{j,k} \sigma'\left(\sum_{i=0}^{n} w_{i,j} a_i\right) a_i, \quad [11]$$

where $\triangle_k$ is a modified error, which is defined as:

$$\triangle_k = \left(y_j - \widehat{y_j} \right) \sigma'\left(\sum_{i=0}^{n} w_{i,j} a_i\right). \quad [12]$$

The weights are then updated using a stochastic gradient descent algorithm in batch mode. However, the simple gradient descent update of the weights and biases is given as:

$$w_{i,j}^{new} = w_{i,j}^{old} + l \frac{\partial L_k}{\partial w_{i,j}}. \quad [13]$$

Here, l is the learning rate or weight decay parameter that controls the change in the weights from one iteration to another. At initial conditions, the weights and biases are set to small random values and then changed as the network learns (Shmueli et al., 2017).

1.4. Introduction of the physics-informed neural networks

As discussed in the introduction, this work employs the penalty-based approach to augment the standard loss functions with additional physics-based terms. As in Raissi et al. (2019), this approach extends the standard loss function in Eq. [10] to include the mean-squared error (MSE) that is associated with the governing PDEs, as well as initial and boundary conditions as follows:

$$L = MSE_1 + MSE_2 + MSE_3, \quad [14]$$

\where MSE_1 , MSE_2 , and MSE_3 are the MSEs associated with the standard DL, PDE, and boundary conditions, respectively. Note that Raissi et al. (2019) also presented a version of the equation that incorporated the loss associated with the PDE (MSE_2)only instead of both MSE_2 and MSE_3 .

Here, we point out that simply summing up the MSEs implicitly assumes equal weights in this multi-objective optimization problem. Considering that each of the MSEs in this equation could vary in magnitude, there is no guarantee that all three MSEs will be minimized to the same degree. On the contrary, the largest of these three MSEs typically gets minimized the most. In this work, we propose using a weighted summation of the standard DL MSE (Eq. [10]) and the thermodynamics constraints discussed previously. Of the three thermodynamics constraints, we only consider the interphase mass balance and component balance constraints because the equality of fugacity is computationally expensive.

Generating training data from the iterative solution of the Rachford-Rice equation is analogical to the use of a numerical or analytical model to create the training data for standard PDE-based PINNs like in Raissi et al. (2019) and Haghighat and Juanes (2021). Similarly, we use the interphase mass balance and component balance constraints instead of the PDEs in these previous publications. This work aims to demonstrate the feasibility of incorporating physical constraints that do not have to be PDEs.

The modified loss function used in this work is given as:

$$L = \lambda_1 MSE_1 + \lambda_2 MSE_2 + \lambda_3 MSE_3, \quad [15]$$

where these three MSEs are given as:

$$MSE_1 = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} \left(y_{i,j} - \widehat{y_{i,j}} \right)^2, \quad [16]$$

$$MSE_2 = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{n_c} \left(x_{i,j}(1 - V_i) + y_{i,j} V_i - z_{i,j} \right)^2, \quad [17]$$

$$MSE_3 = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{n_c} \left(x_{ij} - y_{ij} \right)^2, \quad [18]$$

and λ_1, λ_2, and λ_3 are the corresponding weights. Note that the inner summation in Eq. [16] is over the total number of variables M, whereas it is over the total number of components n_c in Eqs. [17] and [18]. A comparison of the equations for MSE_2 and MSE_3 with the equations for the interphase mass balance and component balance constraints (Eqs.

[2] and [6]) indicates that these equations correspond to both constraints in the homogeneous form. In contrast, MSE_1 is the data misfit, which is based on the difference between the model and the training data, as in Eq. [10]. To evaluate the significance of MSE_2 and MSE_3 without one masking the effect of the other, we only run them in isolation in this work. This implies that we have two main scenarios—one with a loss function that only involves MSE_1 and MSE_2, and another which involves MSE_1 and MSE_3.

1.4.1. Model evaluation metrics

We evaluate the effectiveness of the PINN model at honoring the physical constraints using the root mean squared error (RMSE). For the interphase mass balance constraint, we compute the RMSE as:

$$RMSE_2 = \sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{n_c}\left\{\left(z_{i,j} - \left[\widehat{x}_{i,j}(1-\widehat{V}_i) + \widehat{y}_{i,j}\widehat{V}\right]\right)^2\right\}}{N}} \quad [19]$$

Considering that the overall mole-fractions provided are guaranteed to sum up to one, the component balance constraint will consist of two different RMSEs—one for the liquid phase ($RMSE_L$) and the other for the vapor phase ($RMSE_V$). These two RMSEs are computed as:

$$RMSE_L = \sqrt{\frac{\sum_{i=1}^{n}\left(\left(\sum_{j=1}^{n_c}\widehat{x}_{i,j}\right) - 1\right)^2}{N}} \quad [20]$$

$$RMSE_V = \sqrt{\frac{\sum_{i=1}^{n}\left(\left(\sum_{j=1}^{n_c}\widehat{y}_{i,j}\right) - 1\right)^2}{N}} \quad [21]$$

For simplicity, we combine these two RMSEs by adding them together to obtain the RMSE for the component balance constraint as follows:

$$RMSE_3 = RMSE_L + RMSE_V\ . \quad [22]$$

1.5. Implementation of the physics-informed neural networks

This work focuses on developing a PINN regression model for predicting the phase compositions and vapor fraction. We implement the PINN models using the SciANN Python package presented in Haghighat and Juanes (2021). A fully connected feed-forward deep neural network is constructed with the loss function modified to account for the physical constraints. The inputs and outputs of the PINN model used in this work are shown in Fig. 2, which is a sketch of the neural network. In this figure, z_i, x_i, and y_i, represent the overall, liquid-phase, and gas-phase mole fractions, whereas, "isLiquid" and "isGas" are dummy variables that indicate whether the fluid exists in the liquid, gas, or two-phase state. Note that the two-phase state is not provided to avoid the well-known "dummy-variable trap" (Suits, 1957). Table 2 summarizes the specification of the deep neural network.

1.5.1. K-fold cross-validation

Due to the stochastic nature of the random initial neural network weights and the stochastic gradient optimizers used to train the model, the predicted outputs typically change even if the same model is trained several times on the same input data. So, we perform seven-fold cross-validation, where seven DL models with the same parameters are trained and tested on different subsets of the training data. This allows us to quantify the model's performance in terms of its mean and standard deviation, leading to better estimates of its performance on unseen data. In each of the seven folds, 6/7 of the training and validation data is used to train the model. In contrast, the remaining 1/7 of the data is used to validate the model's performance. So, we obtain seven different performance estimates from each of the specified DNN model parameters. These are used to generate the box plots presented in the next section. Additionally, the seven models generated with each combination of model parameters are saved and combined using an equal weighting scheme to form an ensemble model. This technique, which is known as model averaging or bootstrap aggregating ("bagging" for short), helps to reduce the generalization error (Breiman, 1996) and typically outperforms the single best-performing model (Goodfellow et al., 2016).

1.6. Discussion of results

1.6.1. PINNs with interphase mass balance constraint

This section compares the results from a PINN model to those from a standard DL model. The use of the weighted sum of MSEs allows us to obtain the results of a standard DL model by setting the values of λ_2 and

Table 2
Specification of the neural network.

Dataset	Training: 700,000 Testing: 150,000 Validation: 150,000
Network	Input layer has 6 neurons; 4 hidden layers have 128 neurons each; output layer has 7 neurons
Batch size	256
Number of epochs	300
Optimizer	Adam
Modified Loss function	$\lambda_1 L_1 + \lambda_2 L_2 + \lambda_3 L_3$
Hidden layer activation function	ReLU
Output layer activation function	Sigmoid
Metric	MSE

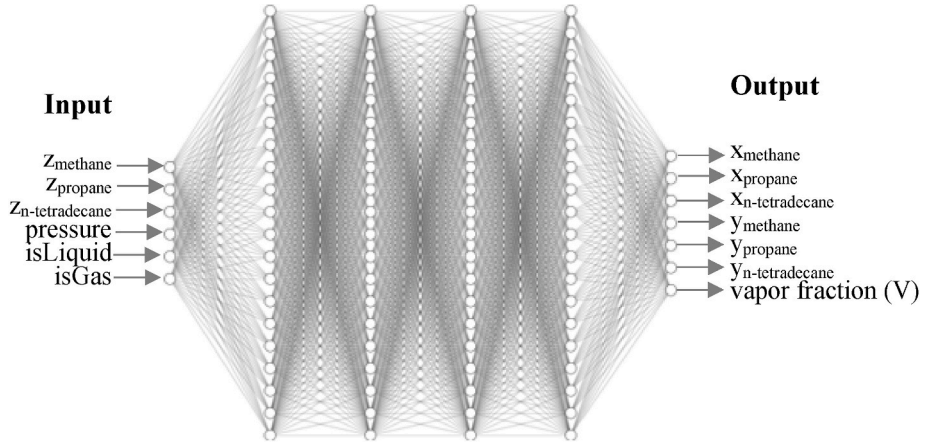

Fig. 2. Neural network sketch shows the hidden layers and input and output variables of the DNN and PNN models.

λ_3 to zero because λ_1 is set to a constant value of one in all cases. So, to include only the interphase mass balance constraint, we simplify Eq. [15] as follows:

$$L = MSE_1 + \lambda_2 MSE_2 . \quad [23]$$

It is essential to determine the value of λ_2 that minimizes the errors associated with the data misfit (MSE_1) and the interphase mass balance constraint (MSE_2). To this end, Fig. 3 presents a box plot of the RMSE of the interphase mass balance constraint and R^2 value on the primary and secondary Y-axes, respectively. These values are plotted against weight λ_2 on the X-axis to find the optimum value of $\lambda_2$where the RMSE of the interphase mass balance constraint is minimized, and the R^2 is still very high. The RMSE measures the degree to which the model honors the laws governing phase equilibrium, whereas the R^2 quantifies the model's overall accuracy. The weight (λ_2) is set to values ranging between 0 and 10, with the first box at a weight of zero in Fig. 3 being essentially a standard DL model with no physics constraint.

As mentioned in the previous section, the box plots in Fig. 3 are generated using the seven estimates of RMSE and R^2 from the seven-fold cross-validation. Although we tried a computationally expensive multi-objective optimization of the weights, the approach of systematically increasing the weights while observing the changes in the RMSE and R^2 values yields valuable insights into the model performance at different λ_2 values. From Fig. 3, we observe that the RMSE of the interphase mass balance error drops from 2.7% to 1.2% when the weight λ_2 is increased from 0 to 2, after which the R^2 value declines more appreciably. Finally, we conclude from this figure that the optimum weight to be used in the interphase mass balance constraint is ~1.73. This is because the RMSE only decreases slightly while the R^2 value decreases significantly when λ_2 is increased above 1.73.

Table 3 presents a comparison between the PINN and DNN models using the R^2 and RMSE of the data misfit and the two physics constraints. In this table, "PINN1" and "DNN1" refer to the model results with/without the interphase mass balance constraint, whereas "PINN2" and "DNN2" refer to corresponding results with/without the component balance constraint. Additionally, we obtained the tabulated from the best single PINN and best single DNN models instead of the ensemble model results. This standard practice when benchmarking models helps avoid the natural effect of improved model performance due to model averaging (Goodfellow et al., 2016).

Table 3
Comparison of PINN to DNN model.

	PINN1	DNN1	PINN2	DNN2
Overall Model R^2	0.9658	0.9663	0.9683	0.9663
Overall RMSE	0.0356	0.0399	0.03887	0.0399
Physics RMSE	0.0118	0.0265	0.00015	0.00126

Table 3 shows that the RMSEs for the two physical constraints are lower in the PINN models than in their corresponding DNN models. This indicates that the physics-based constraints included in the PINN models results in model predictions that honor these physical laws, whereas DNN models only minimize the empirical loss function. Although the R^2 and overall RMSE are the same in "DNN1" and "DNN2" because they refer to the same model, the RMSEs are different. This is because the physics RMSE in DNN1 refers to the interphase mass balance RMSE (Eq. [19]).

Whereas that of DNN2 refers to the component balance RMSE (Eq. [22]). Despite the fact that the overall RMSE and R^2 are approximately the same in the PINN1 and DNN1 models, the interphase mass balance RMSE for the PINN1 model is 55% less than that of the DNN1 model. So, we can conclude that the PINN model honors the interphase mass balance constraints at the selected λ_2 value of 1.73. Additionally, this result and Eq. [19] clearly show that unlike the overall RMSE and R^2 that simply compare model predictions to test data, the physics RMSE for PINN1 directly quantifies the degree to which the predicted phase compositions and vapor fraction honors the interphase mass balance.

1.6.2. *PINNs with component balance constraint*

Here, we discuss the results from training PINNs with a component balance constraint. In this case, we simplify the loss function in Eq. [15] as follows:

$$L = MSE_1 + \lambda_3 MSE_3 . \quad [24]$$

Eqs. [23] and [24] implicitly show that the two physical constraints were incorporated and optimized in isolation. This is to keep the optimization of these weights simple to interpret and implement. Fig. 4 presents a box plot of the RMSE of the component balance error and R^2 of the model against λ_3. It shows that the RMSE drops from 0.15% to 0.02% while the R^2 remains fairly constant as λ_3 is increased from zero to 2.24. So, the optimum weight for the component balance constraint is

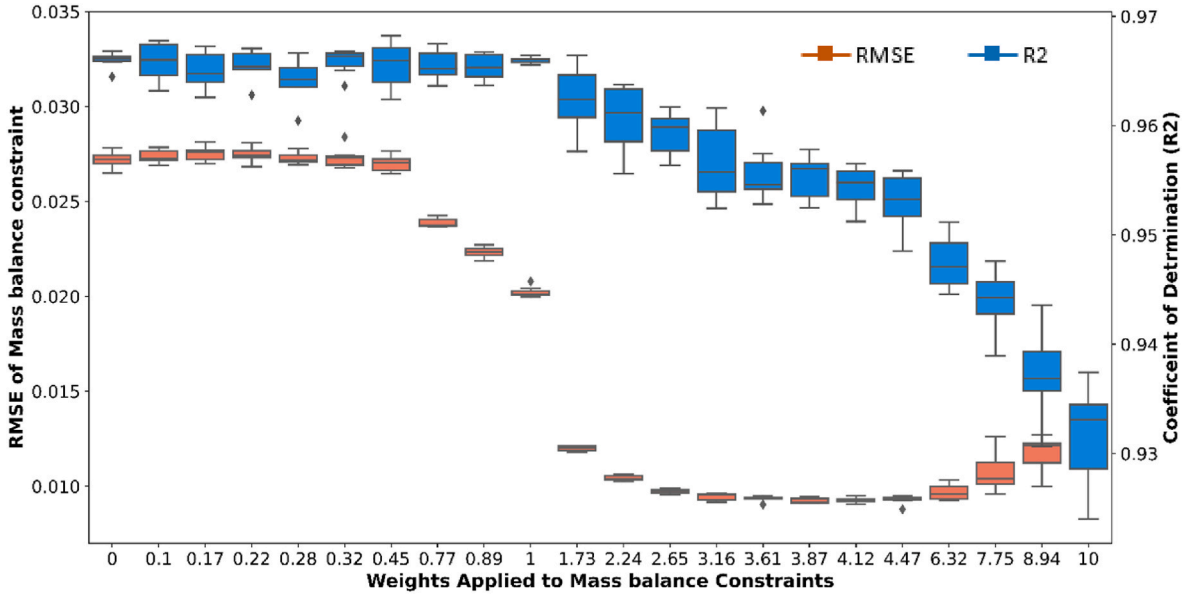

Fig. 3. Box plots show how the model accuracy and physics constraint errors (indicated by the R^2 and RMSE, respectively) vary with the weights applied to the interphase mass balance constraint.

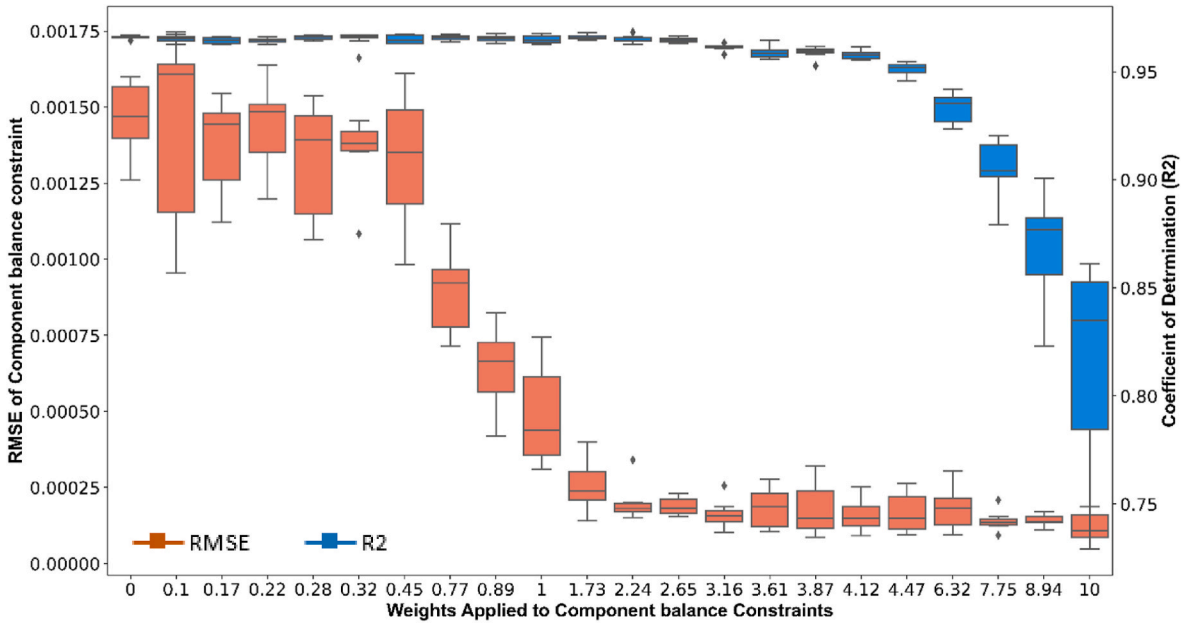

Fig. 4. Box plots show how the model accuracy and physics constraint errors (indicated by the R^2 and RMSE, respectively) vary with the weights applied to the component balance constraint.

~2.24. Table 3 shows that the component balance RMSE for PINN2 is 88% less than that for DNN2. This indicates that the PINN model honors the component balance constraints at the selected λ_3 value of 2.24. Unlike the overall model RMSE and R^2, Eqs. [19] through [21] show clearly that the component balance and interphase mass balance RMSEs are not computed relative to the test data. They are entirely functions of the predicted data, making them excellent metrics to quantify the degree to which the predicted phase compositions and vapor fraction honor physics constraints and not just the degree to which they match the test data.

Comparing the RMSEs in Fig. 3 to those in Fig. 4 and Table 3 shows that the component balance RMSEs are one order of magnitude lower than the interphase mass balance RMSEs. This could be attributed to the ability of standard DNN models to infer that the phase compositions should sum to one without implementing it into the loss function explicitly. In contrast, the interphase mass balance constraint is more complex because it is a linear combination of the predicted vapor fraction, overall, and phase mole fractions. Additionally, the component balance error could be much smaller because the DNN models can learn the component balance constraint from the implicit component balance in the overall mole fractions in the training data. Therefore, the remainder of this paper focuses on PINNs with the interphase mass balance constraint only.

The DNN and PINN models are used to predict the x_i, y_i, and V of the 150,000 fluid mixtures in the test dataset. The predicted fluid mixtures are then ranked in percentiles based on each mixture's interphase mass balance errors in the PINN model, which is approximately the same as the DNN model's ranking. The overall composition of the fluid mixtures at the 99th, 75th, 50th, 25th, and 1st error percentiles are used to create the phase diagrams in Fig. 5. The other two curves in the pressure-temperature (P-T) phase diagrams are based on the same fluid mixtures but with different overall compositions. The "actual" overall composition is simply taken from the z_i values for the corresponding fluid mixture in the test data to obtain the dotted red lines. However, for the DNN and PINN models, the overall mole fractions were computed from the predicted phase mole fractions using Eq. [2]. Table 4 shows the total and interphase mass balance errors associated with each fluid mixture at the outlined percentiles. The total error is the sum of the absolute difference between the predicted phase compositions and vapor fractions from the PINN and DNN models and their corresponding values in the test data. Similarly, the "physics 1 error" is the absolute value of "$z_i - Vy_i - (1 - V)x_i$" from Eq. [2]. This table shows that the PINN model outperforms the DNN model at all percentiles.

The P-T phase diagrams in Fig. 5(a) are based on model predictions of the overall mole fraction for a fluid mixture at pressure and temperature conditions where the composition exists in the single-phase liquid state. Under such conditions, the estimated z_i values exactly match the actual z_i values because the x_i's are identical to the z_i values, whereas y_i and V are zeros. So, in Fig. 5(a), the DNN and PINN models also match the actual P-T phase diagram exactly. The remaining images in Fig. 5 show the corresponding phase diagrams for two-phase fluid mixtures at the specified percentiles. Although these phase diagrams cover a wide range of temperature and pressure values, the input data used to create them correspond to a single point in the P-T phase diagram. So, it is unrealistic to expect a machine learning model trained to predict the fluid properties for a distinct fluid mixture at only one pressure and temperature to match the actual phase behavior over a wide pressure and temperature range over which it is not trained.

The results in Fig. 5 show that the prediction of the phase diagram based on the PINN model outperforms the DNN model in all cases. The black vertical line highlights the isothermal temperature (of 176 °F) at which all the fluid compositions were specified. As expected, the deviation of the models from the actual saturation pressure at this temperature increases as the error percentile decreases. Although the R^2 and RMSE for the models in Table 3 do not indicate a significant difference in the results, Table 4 and the phase diagrams for the predicted phase compositions show that incorporating physics with PINNs yields predictions that more accurately describe the phase behavior of compositional fluid mixtures. This is because incorporating physics-based constraints via custom loss functions is known to act as a physics-based regularization that helps solve ill-posed problems and prevent overfitting (Kashinath et al., 2021). Finally, the DNN models show excellent performance metrics, but the predictions may not honor the thermodynamics constraints of phase equilibrium, which are not imposed during the model training. PINNs address this limitation and yield better model predictions that honor the physical constraints of phase equilibrium. Additionally, our timing results show that the trained PINN models are 145 times faster than the standard iterative flash procedure.

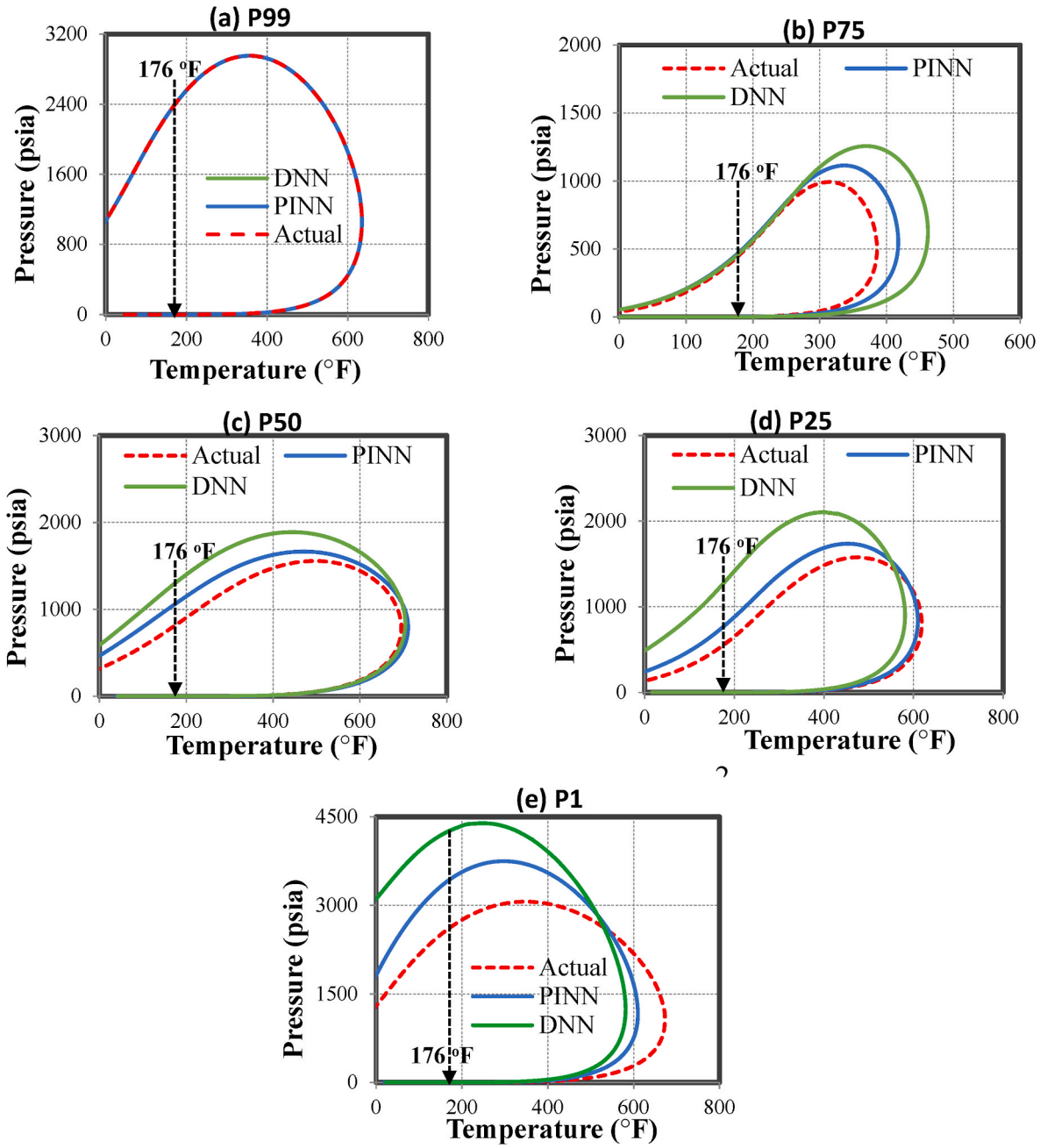

Fig. 5. Phase envelopes for compositions at different percentiles indicate that the PINN model yields a better description of the phase behavior than the DNN model.

Table 4
Summary of the performance of the DNN and PINN models at specific percentiles.

Percentile	PINN Total error	PINN Physics I error	DNN Total error	DNN Physics I error
99	8.60E-05	2.00E-06	1.20E-04	7.00E-05
75	1.33E-03	1.20E-05	3.91E-03	7.30E-04
50	1.22E-02	2.16E-03	2.37E-02	2.17E-02
25	7.37E-02	1.47E-02	1.35E-01	2.19E-02
1	4.07E-01	1.34E-02	4.13E-01	2.83E-02

2. Conclusions

This work presents the incorporation of thermodynamics constraints into deep learning models for two-phase flash calculations. It is achieved by modifying the standard loss function to include these physics constraints as a weighted sum of mean-squared errors. We performed a sensitivity study that showed the change in the overall R^2 and RMSE with increasing weights to determine the optimum weights used in the custom loss function. We generated one million unique fluid mixtures from a space-filling mixture design to obtain the data needed in this work. These mixtures' pressure, temperature, and overall composition were used to predict the flash output variables. Of the one million fluid mixtures, 150,000 were withheld for testing, whereas the remaining were partitioned between training and validation in a seven-fold cross-validation.

The results show that incorporating thermodynamics constraints into PINNs yields model predictions with over a 55% reduction in the physics constraint errors (RMSE) when compared with DNNs. Although the RMSE of the physics constraints reduced significantly, the difference between the overall R^2 value of the PINN and DNN models was

negligible (<0.05%). This indicates that using PINNs results in model predictions that honor physical constraints without reducing overall model accuracy. Finally, we presented phase diagrams that show that the PINN model significantly outperforms the DNN model in predicting compositional fluid behavior.

CRediT author statement

Thelma Ihunde: Data curation, Writing — original draft preparation, Visualization, Investigation, and Validation. **Olufemi Olorode:** Supervision, Conceptualization, Methodology, Software, Writing—Reviewing and Editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors recognize the support of the Craft and Hawkins Department of Petroleum Engineering at Louisiana State University. This work was facilitated using the computational resources and software packages available in the department and at the Center for Computation and Technology.

Nomenclature

f_{Li}	fugacity of component i in the liquid phase
f_{Vi}	fugacity of component i in the vapor phase
k_i	vapor-liquid equilibrium factor
L	loss function
MSE	mean square error, which is used as the loss function
P	pressure
$RMSE$	root mean square error, which is used as a metric
R^2	coefficient of determination, which is used as a metric
T	temperature
V	vapor fraction
x_i	mole fraction of component, i in the liquid phase
y_i	mole fraction of component, i in the gas phase
z_i	overall mole fraction of component, i.

References

Belkadi, A., Yan, W., Michelsen, M.L., Stenby, E.H., 2011. Comparison of two methods for speeding up flash calculations in compositional simulations. In: SPE Reservoir Simulation Symposium.
Beucler, T., Rasp, S., Pritchard, M., Gentine, P., 2019. In: Achieving Conservation of Energy in Neural Network Emulators for Climate Modeling arXiv preprint arXiv: 1906.06622.
Breiman, L., 1996. Bagging predictors. Mach. Learn. 24 (2), 123–140.
Coats, K.H., 1980. An equation of state compositional model. Soc. Petrol. Eng. J. 20 (5), 363–376.
Daw, A., Karpatne, A., Watkins, W., Read, J., Kumar, V., 2021. Physics-guided Neural Networks (Pgnn): an Application in Lake Temperature Modeling arXiv preprint arXiv:1710.11431.
Firoozabadi, A., 2016. Thermodynamics and Applications in Hydrocarbon Energy Production. McGraw-Hill Education.
Firoozabadi, A., Pan, H., 2000. Fast and robust algorithm for compositional modeling: Part i-stability analysis testing. In: SPE Annual Technical Conference and Exhibition.
Furukawa, H., Shoham, O., Brill, J., 1986. Predicting Compositional Two-phase Flow Behavior in Pipelines.
Gaganis, V., Varotsis, N., 2012. Machine learning methods to speed up compositional reservoir simulation. In: SPE Europec/EAGE Annual Conference.
Gaganis, V., Varotsis, N., 2014. An integrated approach for rapid phase behavior calculations in compositional modeling. J. Petrol. Sci. Eng. 118, 74–87.
Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT press.
Gould, T.L., 1979. Compositional two-phase flow in pipelines. J. Petrol. Technol. 31 (3), 373–384.
Haghighat, E., Juanes, R., 2021. Sciann: a keras/tensorflow wrapper for scientific computations and physics-informed deep learning using artificial neural networks. Comput. Methods Appl. Mech. Eng. 373, 113552.
Huang, D.Z., Xu, K., Farhat, C., Darve, E., 2020. Learning constitutive relations from indirect observations using deep neural networks. J. Comput. Phys. 416, 109491.
Ihunde, T.A., Olorode, O., 2021. Application of physics informed neural networks to compositional modelling. In: SPE/AAPG/SEG Asia Pacific Unconventional Resources Technology Conference.
Jiang, C.M., Kashinath, K., Prabhat, Marcus, P., 2020. Enforcing hard physical constraints in cnns through differentiable pde layer. In: ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations.
Kashinath, A., Szulczewski, M.L., Dogru, A.H., 2018. A fast algorithm for calculating isothermal phase behavior using machine learning. Fluid Phase Equil. 465, 73–82.
Kashinath, K., Mustafa, M., Albert, A., Wu, J., Jiang, C., Esmaeilzadeh, S., Singh, A., 2021. Physics-informed machine learning: case studies for weather and climate modelling. Philos. Trans. Royal Soc. A 379 (2194), 20200093.
Li, Y., Zhang, T., Sun, S., 2019. Acceleration of the NVT flash calculation for multicomponent mixtures using deep neural network models. Ind. Eng. Chem. Res. 58 (27), 12312–12322.
Lie, K.-A., 2019. An Introduction to Reservoir Simulation Using MATLAB/GNU Octave: User Guide for the MATLAB Reservoir Simulation Toolbox (MRST). Cambridge University Press.
Michelsen, M.L., 1982a. The isothermal flash problem. Part I. Stability. Fluid Phase Equil. 9 (1), 1–19.
Michelsen, M.L., 1982b. The isothermal flash problem. Part II. Phase-split calculation. Fluid Phase Equil. 9 (1), 21–40.
Nichita, D.V., Gomez, S., Luna, E., 2002. Multiphase equilibria calculation by direct minimization of Gibbs free energy with a global optimization method. Comput. Chem. Eng. 26 (12), 1703–1724.
Nichita, D.V., Graciaa, A., 2011. A new reduction method for phase equilibrium calculations. Fluid Phase Equil. 302 (1–2), 226–233.
Okuno, R., Johns, R.T., Sepehrnoori, K., 2010. A new algorithm for Rachford-Rice for multiphase compositional simulation. SPE J. 15 (2), 313–325.
Pal, N., Mandal, A., 2021. Compositional Simulation Model and History-Matching Analysis of Surfactant-Polymer-Nanoparticle (SPN) Nanoemulsion Assisted Enhanced Oil Recovery. Journal of the Taiwan Institute of Chemical Engineers.
Pan, H., Firoozabadi, A., 2001. Fast and robust algorithm for compositional modeling: part ii-two-phase flash computations. In: SPE Annual Technical Conference and Exhibition.
Peng, D.-Y., Robinson, D.B., 1976. A new two-constant equation of state. Ind. Eng. Chem. Fundam. 15 (1), 59–64.
Raissi, M., Karniadakis, G.E., 2018. Hidden physics models: machine learning of nonlinear partial differential equations. J. Comput. Phys. 357, 125–141.
Raissi, M., Perdikaris, P., Karniadakis, G.E., 2019. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. 378, 686–707.
Russell, S.J., Norvig, P., 2016. Artifical Intelligence: A Modern Approach. Pearson.
Shmueli, G., Bruce, P.C., Yahav, I., Patel, N.R., Lichtendahl Jr., K.C., 2017. Data Mining for Business Analytics: Concepts, Techniques, and Applications in R. John Wiley & Sons.
Suits, D.B., 1957. Use of dummy variables in regression equations. J. Am. Stat. Assoc. 52 (280), 548–551.
Trask, A.W., 2019. Grokking Deep Learning. Simon and Schuster.
Voskov, D.V., Tchelepi, H.A., 2009. Tie-simplex based mathematical framework for thermodynamical equilibrium computation of mixtures with an arbitrary number of phases. Fluid Phase Equil. 283 (1–2), 1–11.
Wang, K., Luo, J., Wei, Y., Wu, K., Li, J., Chen, Z., 2019a. Artificial neural network assisted two-phase flash calculations in isothermal and thermal compositional simulations. Fluid Phase Equil. 486, 59–79.

Wang, K., Luo, J., Wei, Y., Wu, K., Li, J., Chen, Z., 2020. Practical application of machine learning on fast phase equilibrium calculations in compositional reservoir simulations. J. Comput. Phys. 401, 109013.
Wang, S., Sobecki, N., Ding, D., Zhu, L., Wu, Y.-S., 2019b. Accelerating and stabilizing the vapor-liquid equilibrium (VLE) calculation in compositional simulation of unconventional reservoirs using deep learning based flash calculation. Fuel 253, 209–219.
Weidman, S., 2019. Deep Learning from Scratch: Building with Python from First Principles. O'Reilly Media, Inc.
Wilson, G., 1968. A modified redlich–kwong eos, application to general physical data calculations. In: AIChE 65th National Meeting, p. 15c.
Wu, Y., Kowitz, C., Sun, S., Salama, A., 2015. Speeding up the flash calculations in two-phase compositional flow simulations–The application of sparse grids. J. Comput. Phys. 285, 88–99.
Xu, K., Darve, E., 2020. Physics Constrained Learning for Data-Driven Inverse Modeling from Sparse Observations arXiv preprint arXiv:2002.10521.
Young, L.C., Stephenson, R.E., 1983. A generalized compositional approach for reservoir simulation. Soc. Petrol. Eng. J. 23 (5), 727–742.