CHAPTER SEVEN

Recent progress in accelerating flash calculation using deep learning algorithms

Contents

7.1 Accelerated flash calculation using deep learning algorithm with experimental data as input

Vapor–liquid equilibrium (VLE) is of essential importance in modeling the multiphase and multicomponent flow simulation for a number of engineering processes. Knowledge of the equilibrium conditions in mixtures can be obtained from data collected in direct experiment or using thermodynamic models, including activity coefficients at low system pressure or fugacity coefficients at high system pressure. In the last two decades the application of equilibrium calculation using equations of state (EOSs) that describes mixing rules with experience coefficients has been proposed and widely discussed. A realistic EOS, for example, Peng–Robinson, is generally considered as an appropriate thermodynamic model to correlate and predict VLE conditions, due to the long-time improvement developed with applications in different aspects. The calculation procedures using EOS have been extensively studied and modified. The EOS parameters, describing concentration, combination, and interaction between binary mixtures, can decide the accuracy in correlating the VLE process. In practice, such

Reservoir Simulations.
DOI: https://doi.org/10.1016/B978-0-12-820957-8.00007-1

parameters are generally obtained by fitting experimental data under the temperature at which VLE is required. However, in most EOS calculations, iterations are needed, which makes it less suitable for time-sensitive applications. For cases such as phase equilibrium in underground heated flow, as the concentration of many possible components varies greatly in a wide range, it is difficult to correlate them and predict the parameters exactly from experiments. Some very small concentrations of components are essential parts, which make the parameter calculation more difficult. In reality a hydrocarbon mixture may consist of tens to hundreds of species, which is far more than the number of components in numerical simulations. Thus in order to accelerate phase equilibrium calculation, what first comes to mind is to lump the fluid mixture into a smaller number of pseudocomponents without losing much accuracy of the EOS model. Clearly, the fewer the number of components is, the more computational time can be saved and meanwhile the more accuracy will be lost. Despite this, the speedup given by reducing the number of components is barely satisfactory, therefore stimulating a large batch of researchers to develop acceleration strategies for phase equilibrium calculation over the past two decades (Zhu et al., 2019; Zhang et al., 2015, 2017; Mathias, 1983; Pedersen et al., 2006; Li and Firoozabadi, 2012; Kou and Sun, 2015, 2018a, 2018b; Li et al., 2019a,b; Baker et al., 1982; Shen et al., 2018; Tao et al., 2019; Sun et al., 2017).

7.1.1 Introduction on artificial neural network

Since the breakthrough of AlexNet in 2012, deep learning has made profound impact on both the industry and the academia. Not only has it revolutionized the computer vision field, improving the performance of image recognition and object detection dramatically, and the natural language processing field, setting new record for speed recognition and language translation, but it has also enabled machines to reach human level intelligence in some certain tasks, such as the Go game. In addition to those well-known tasks that deep learning is especially expert in, deep learning has been applied to a broad range of problems, such as protein binding affinity prediction, enzyme function prediction, structure superresolution reconstruction, the third-generation sequencing modeling, particle accelerator data analysis, and modeling brain circuits. Such a great potential of deep learning comes from its significant performance improvement over the traditional machine learning algorithms, such as support vector machine. The traditional machine learning methods usually consider just one layer nonlinear combination of the input features, while the deep learning method can consider ultracomplex nonlinear combination of the input features by taking advantage of multiple hidden layers. During training, the back-propagation algorithm increases the weight of the feature combination which is useful for the final classification or regression problem to emphasize the useful features while decreases the weight of those

unrelated feature combinations. In spite of the universal approximation theorem, which states that we can approximate any continuous function using a feed-forward network with a single hidden layer containing a finite number of neurons, the success of deep learning shows the potential of fitting VLE using multilayer neural networks. In principle, deep learning can serve as a general function approximator, which can approximate the underlying physical process of VLE in an implicit way. Meanwhile, with such approximation, the matureness of both the hardware and software development in the deep learning field can be very helpful to accelerate the original complex flash calculation.

Artificial neural networks (ANNs) are computational models designed to incorporate and extract key features of the original inputs. Deep neural networks usually refer to those ANNs that consist of multiple hidden layers. A deep fully connected neural network is applied to model the VLE. Following the input layer, a number of fully connected hidden layers, with a certain number of nodes, stack over the other, whose final output is fed into another fully connected layer, which is the final output layer. Since we are fitting X and Y in our model, the final output layer contains two nodes, each of which predicts the value of one of the two variables. The activation function of this layer is fixed as linear. Naturally the proposed ANN input variables include critical pressure (P_c), critical temperature (T_c), and acentric factor (ω) of the components comprising the mixture. As a result, the eight variables in Fig. 7.1 are the above three factors for each of the two components in the mixture, the temperature, and the pressure. The required binary mixture experimental VLE data were gathered from the Korea Thermophysical Properties Data Bank (KDB), of 1332 data points in total, with supplementary selection of consistency and applicability. As instructed on the database,

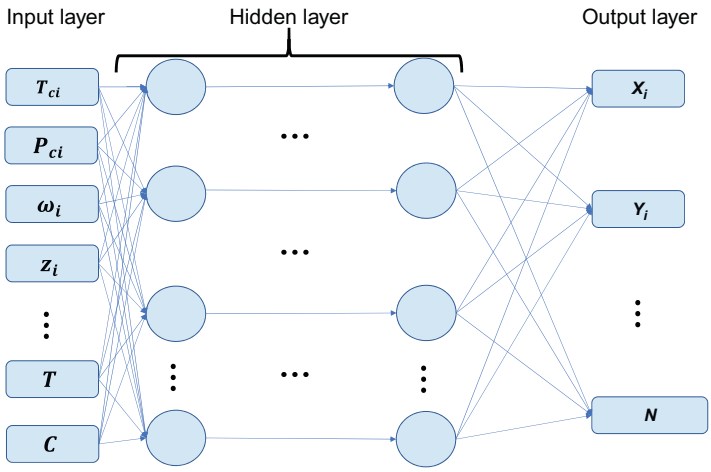

Figure 7.1 Neural network structure to model phase equilibrium.

the expected mean relative error of the experimental data we used for training and validating the model is around 20%. A large range of pressures and temperatures are considered while ensuring that the mixture does not enter into a critical state, which is to confirm that a two-phase condition is ensured.

Due to the high complexity of the neural network model and the limited number of data (only 1332 records in total), the trained model is subject to overfitting. To deal with the common and most serious issue in the deep learning field, we adopted weight decay as well as dropout to handle the problem. The model initialization can also influence the final result significantly. We utilized Xavier initializer to perform the model initialization. The whole package is developed using TFlearn. Trained on a workstation with one Maxwell Titan X card, the model converged in 10 minutes. A simplified flowchart of our network model working process in each node is presented in Fig. 7.2.

Formally, for the ith hidden layer, let a_i denote the input of the layer, and y_i to denote the output of the layer. Then we have

$$y_i = f_i(W_i \times a_i + b_i), \tag{7.1}$$

where W_i is the weight; b_i is the bias; and f_i is the activation functions of the ith layer. For a network with multiple layers the output of one hidden layer is the input of the next layer. For example, we can represent the network in Fig. 7.1 as

$$o = f_3(W_3 \times f_2(W_2 \times f_1(W_1 * x_1 + b_1) + b_2) + b_3), \tag{7.2}$$

where $o = (X, T)$; f_1, f_2, f_3 are the activation functions; W_1, W_2, W_3 are the weights for each layer; b_1, b_2, b_3 are the bias terms of each layer.

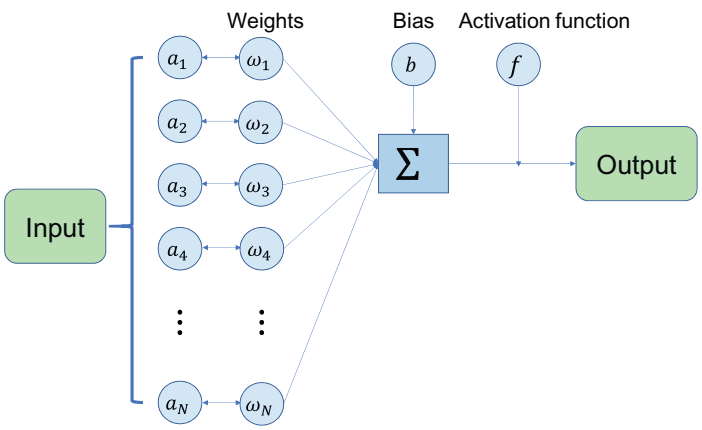

Figure 7.2 The flowchart of regression process in each node.

7.1.2 Technique explanation in artificial neural network

Here are the short explanations of the techniques used to obtain a practical network:

1. Weight decay: Overfitting is usually a serious issue in the deep learning field, which means that the learned model has almost perfect performance on the training data while performing poorly on the validation or testing data. The main reason of overfitting in this field is that the model itself is composed of too many parameters, while we do not have enough training data, that is, the model is overparameterized. In order to prevent the overfitting issue from hurting the model's performance, we usually apply additional constraint on the model's parameters to reduce the freedom of the model. In general, if the model is overfitted, the norm of the weight parameters is often very large. As a result, one way to avoid overfitting is to add an additional constraint on the norm of the weight parameters and penalize large weights. In practice, we can add a regularization term, which is related to the norm of the weights, in the loss function to make the model fit the training data and penalize large weights at the same time. Formally, the original loss function for deep learning, which is the mean squared loss in our problem, can be formulated as

$$L = \frac{1}{N}\sum_{n=1}^{N} \left\| o - \hat{o} \right\|^2, \tag{7.3}$$

where N is the total number of training data; o is the output of the model; $\hat{o}$ is the observed value. After adding the L2 weight decay term the loss function becomes:

$$L = \frac{1}{N}\sum_{n=1}^{N} \left\| o - \hat{o} \right\|^2 + \lambda \left\| \mathbf{W} \right\|_2^2, \tag{7.4}$$

where $\mathbf{W}$ is the whole set of weight parameters of the model; λ is the regularization coefficient, that is, how much we penalize over the large weights.

2. Dropout: Dropout is a very efficient method for dealing with overfitting in neural networks. This method reduces the freedom of the network by discarding nodes and connections of the model during the training stage. For example, if we apply the dropout technique to a certain layer with the keep probability as P $(0 < P < 1)$, then, during each training stage, each node of that layer would first be evaluated independently with the probability of P being kept or the probability of $1 - P$ being discarded. If the nodes are discarded, all the nodes and connections are discarded from the model. After the dropout procedure, the reduced network is trained during the training stage. After that certain training stage, the discarded nodes are inserted to the model with the original weights and the model enters the next training cycle.

3. Xavier initializer: The initialization of the neural network model is of vital importance, which can affect the convergence speed and even the final model's performance. If the weights are initialized with very small values, the variance of the input signal vanishes across different layers and eventually drops to a very low value, which reduces the model complexity and may hurt the model's performance. If the weights are initialized with very large values, the variance of the input signal tends to increase rapidly across different layers. That may cause gradient vanishing or explosion, which increases the difficulty of training a working model. Since we usually initialize the weights with a Gaussian distribution, to control the variance of the signal, it is desirable to initialize the weights with a variance δ to make the variance of the output of a layer the same as that of the input of the layer.
4. Batch normalization: Training deep learning model is notoriously time-consuming, because of the large number of parameters belonging to different layers. Not only is the optimization for such a large number of parameters internally time-consuming, but there are some undesirable properties of the multilayer model which makes the convergence process slow. One property of the deep learning method is that the distribution of each layer's input might change because the parameters of the previous layer are usually changed during training, which is usually referred to as "internal covariate shift." To solve the problem, batch normalization is proposed. In addition to normalize the original input of the model, which is the input of the first layer, this technique makes the normalization part of the model and performs normalization on hidden layers for each training batch during the training stage. Batch normalization enables larger learning rates and can accelerate the convergence speed by 10 times.
5. Activation functions: The activation function is where the nonlinearity and the expressiveness power of deep neural network models come from. There are numerous activation functions: rectified linear unit (ReLU), parametric ReLU (PReLU), TanH, sigmoid, softplus, softsign, leaky ReLU, exponential linear unit (ELU), and scaled ELU (SELU).

7.1.3 Case study

In order to accelerate and optimize the original flash calculation using successive substitution method (SSM), an attempt has been made to use the deep learning method for the VLE calculation of the systems C1−C7 mixtures, including methane, ethane, propane, *N*-butane, *N*-pentane, *N*-hexane, and *N*-heptane. Two other accelerating methods, Newton's method and sparse grids method, are also introduced and used as a comparison. Physical properties of each component are listed in Table 7.1. Essentially, as the Gibbs phase rule stipulates, two intensive properties are required to completely

Table 7.1 Physical properties of the seven components investigated for binary phase equilibrium in this case.

Component	ω	T_c (K)	P_c (bar)
C_1	0.0115	190.6	46
C_2	0.0908	305.4	48.84
C_3	0.1454	369.8	42.46
C_4	0.1886	421.09	37.69
C_5	0.2257	467.85	34.24
C_6	0.2564	521.99	34.66
C_7	0.285	557.09	32.62

Table 7.2 CPU time comparisons among different vapor–liquid equilibrium methods.

Method	CPU time (s)	Acceleration
SSM	2503.32	None
Newton	1201.76	2.082
Sparse grids	5.11	489.823
Deep learning	1.22	2051.639

SSM, Successive substitution method.

describe a binary two-phase system at equilibrium conditions. Temperature and pressure are two such thermodynamic intensive properties conventionally selected, because of the relative ease with which they can be measured. Alongside temperature and pressure the acentric factor is also generally included in VLE phase equilibrium calculations to account for nonsphericity of molecules. The required C1–C7 binary mixture experimental VLE data were gathered from the KDB, of totaling 1332 data points, with supplementary selection of consistency and applicability. As instructed on the database, the expected mean relative error of the experimental data we used for training and validating the model is around 20%. A large range of pressures and temperatures are considered while ensuring that the mixture does not enter into a critical state, which is to confirm that a two-phase condition is ensured.

A selected model based on the above analysis has been used in a flash calculation case, and the results are compared with three other methods, SSM, Newton's method and sparse grids method. The binary components in this case are set as methane and propane, with temperature constant at 226K and pressure of 11 values changing from 6 to 77 bar. The CPU time used for each method is listed in Table 7.2.

It should be noted that the initial guess of Newton's method is the result of SSM, which means that it will take much less time to converge. Besides, the time used to generate the surrogate model in the sparse grids method is neglected, which means that the total real CPU time for sparse grids are much higher. In fact, SSM is applied

to get the initial data model. For the deep learning model, the CPU time for data training is also neglected, but this training time is only 15.72 seconds, which is much lower than other ones. Meanwhile, the trained model can be repeatedly used for different binary components and the conditions of temperature and pressure. It can be concluded that the deep learning method is much more efficient than the traditional SSM, and also faster than the other two acceleration methods. It is easy to expect better efficiency of sparse grids and deep learning method in large-scale calculation, as the model of the two can be repeatedly used in different cases, but in SSM and Newton's method everything will start from scratch.

Except for efficiency, the accuracy of our optimized deep learning model is also proved in our calculation. We collected the experimental data at certain temperature, composition and pressure conditions as the ground truth and compared with the results from different calculation methods. It is noted that as the sparse grids method is based on a surrogate model generated from SSM, the results are neglected in the comparison. It can be referred from Fig. 7.3 that all the results calculated from these methods match the experimental data well, although not perfectly. Generally speaking, SSM will show an obvious error at some points, but results from Newton's method are much better as they converge from the result of SSM. The results of Newton's method is much better, but still less accurate than Deep Learning model. In summary, the optimized deep learning model has much better CPU time efficiency while conserving the similar accuracy of other flash calculation methods. Similar property can be also detected in the binary phase equilibrium calculation results of ethane and pentane at constant temperature 310.93K and varying with pressure (Fig. 7.4).

It can be concluded from the case study that the proposed models can serve the purpose of being close first estimates for more thermodynamically rigorous VLE calculation procedures. However, overfitting is obvious in the model construction process and results in relatively high prediction errors in some cases. Thus it is still necessary to

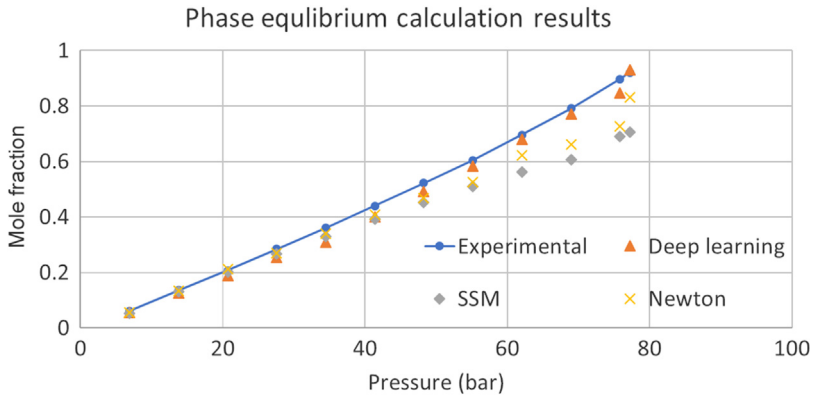

Figure 7.3 Accuracy comparisons among different phase equilibrium calculation methods. The relationship among different calculation methods and experimental data as ground truth.

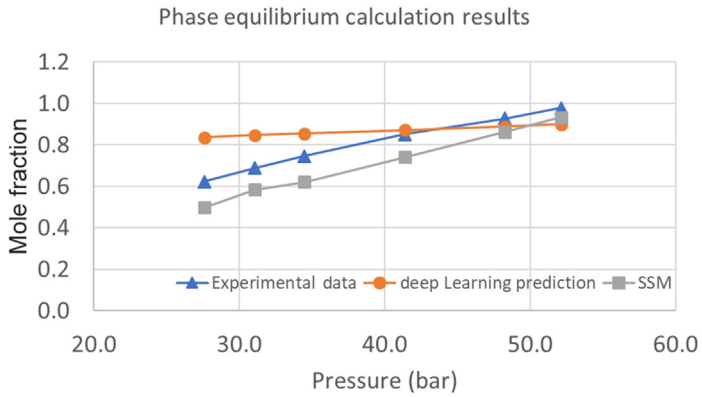

Figure 7.4 The flowchart of regression process in each node.

construct a larger set of experimental binary VLE data, which is hard to collect based on current database. In addition to the a large amount of time spent on repeated typing, another problem is that the tested data are in varieties of units and in some cases, the results can even be different for the same components' mixture. Another way to get large amount of data is to use flash calculation with EOS. However, it is commonly acknowledged that the convergence and accuracy of current flash calculation methods cannot be ensured. Besides, sometimes the results even have no physical meanings and we need to exclude them manually. Thus the application of flash calculation results used as input data should be studied based on the selection and optimization of the calculation method, which remains to be the future work. Compared with traditional flash calculation method, our deep learning model will show better stability, which means that it can always ensure a reasonable result with acceptable error. The problem of manual handling in the process of flash calculation should also be treated carefully. Based on the fact that experimental data are limited, it is expected that new flash calculation methods can be developed to overcome the above problems in current methods. Verifications should also be performed to prove the reliability of the flash algorithms. For example, with the increased calculation capability, it may be possible to extend the deep learning methods developed here to field-scale study in the future.

7.2 Accelerated flash calculation using deep learning algorithm with flash data as input

Compared to the classical NPT flash calculation, the NVT flash exhibits some advantages and has attracted a lot of attention from researchers in recent years. Without inverting the EOS, the NVT formulation has a unique solution and thus

eliminates the root-selection procedure that usually takes place in NPT flash problems. Additionally, volumes of pure substances under saturation pressure cannot be uniquely determined using the NPT formulation, since all states (two phases, single vapor phase, or single liquid phase) of a pure substance share the same pressure on the phase boundary. Similar behavior has been observed in multicomponent mixtures with three or four phases as well. Despite the fact that the NPT flash is the most commonly used flash technique in compositional simulators, numerous efforts have been made to improve the performance of the NVT flash calculation and also extend its applications. To model the dynamic process from any nonequilibrium state to the equilibrium state, Kou (Kou and Sun, 2018) established evolution equations for mole numbers and volume, which were solved by a well-designed energy-stable numerical algorithm. In addition to the aforementioned bulk phase flash problems the confined phase behaviors at constant moles, volume, and temperature have been studied by taking into account adsorption, capillary pressure, or confinement effect resulting from the interaction between fluid molecules and pore walls. In this section, the training data for the deep learning model are provided by the NVT flash calculation. Three real reservoir fluids are investigated, including the five-component Bakken oil, eight-component EagleFord1 oil, and 14-component EagleFord2 oil. The compositional parameters for each reservoir fluid are presented in the Supporting Information. A deep neural network is established with five activation layers, each of which contains 100 nodes, and a total of 4000 iterations. "ReLU" is chosen as the activation function. It is worth mentioning that the performance of this network configuration has been validated in the previous section. To investigate the effect of data size on the performance of the deep neural network model, we calculate equilibrium results of the NVT flash for the EagleFord1 oil on the same computational domain with 51×51, 71×71, 101×101, 151×151, 201×201, and 301×301 uniform grids. In addition, the results of the Bakken oil and EagleFord2 oil are computed on the specified concentration and temperature intervals, which are uniformly divided into 301×301 grids. All eight data sets are used to train the proposed neural network, and the efficiency and accuracy of the trained model are tested. One key effort of this study is to investigate the possibility in achievement of both stability test and phase split calculation by a single neural network model, which is different from the conventional two-step framework that all the preceding research follows based on machine learning models.

7.2.1 Deep learning model training

The compositional properties of fluid components, overall molar concentration, and temperature are used as the input, and the proposed deep neural network predicts mole fractions of components in both vapor and liquid phases. The key parameters of the model are the weights of each activation layer, which control the model

prediction under the given input data. In the beginning, those weights are initialized randomly, implying that the model initially yields useless results. To approximate the NVT flash calculation by the deep learning model, we optimize the weight parameters to fit the equilibrium mole fraction of vapor and liquid components. In the following, 90% of the data are used to train our network model, while the remaining 10% data are used for validation unless otherwise noted.

Table 7.3 presents the training data size (Ntrain), testing data size (Ntest), training time (ttrain) and testing time (ttest) of the deep neural network with different data sets. Here ttest represents the time that the trained model spent on estimating equilibrium mole fractions for the flash problem of the same data size. For instance, it takes 7.9 seconds for the trained model to predict the mole fractions of the EagleFord1 oil on a 201×201 grid. In addition, the mean absolute error (ε_a) and relative error (ε_r) are presented in Table 7.1 as well. Clearly, for the EagleFord1 oil, as the number of input data becomes larger, the training time significantly increases, while the testing time does not change too much. Furthermore, we observe that both absolute and relative prediction errors continue to decrease with the data size increasing. Under the same data volume, it seems the more components are involved, the larger prediction error the deep neural network model exhibits. However, the Bakken oil makes an exception and yields greater error than the Eagle Ford oils. This might be attributed to the underneath correlations between different components differing from the investigated fluid mixtures so that the trained network model yields different accuracy. Essentially, the composition of the Bakken oil is quite different from the compositions of the two Eagle Ford samples, the latter of which exhibit some similarities to some extent. This may explain why our observation disagrees with the expectation. Table 7.4 compares the computational time of NVT flash calculations to the testing time of the deep neural network for the EagleFord1 oil with different data sizes. It can be seen that the testing time is much less than the computational time of flash calculations. When the data size reaches 301×301, the trained model makes predictions 244 times faster than the iterative flash calculation.

Table 7.3 Performance of different data source size.

Training data source	Training time	Test time	Mean absolute error	Mean relative error
EagleFord1 (51×51)	42	4.1	0.02946	0.03582
EagleFord1 (71×71)	134	5.9	0.01930	0.02410
EagleFord1 (101×101)	235	6.3	0.01378	0.01723
EagleFord1 (151×151)	512	7.1	0.01301	0.01647
EagleFord1 (201×201)	1232	7.9	0.01261	0.01580
EagleFord1 (301×301)	2531	8.6	0.01252	0.01571
EagleFord2 (301×301)	2468	8.9	0.01684	0.01964
Bakken (301×301)	2352	9.2	0.02362	0.02885

Table 7.4 Comparison of computational time of the iterative flash calculation and testing time of the deep neural network for the EagleFord1 oil with different data sizes.

Data size	Deep learning CPU time	Flash calculation CPU time
51×51	4.1	61
71×71	5.9	122
101×101	6.3	237
151×151	7.1	547
201×201	7.9	1021
301×301	8.6	2100

7.2.2 Phase splitting test

In the phase splitting calculations, phase stability analysis is always included in most algorithms (Li et al., 2019b). Such tests are needed to determine whether the fluid mixture is stable in given thermodynamic conditions or will split into more phases, which is prior to further multiphase flow and transport investigations. The tangent plane distance, also known as TPD function, is often located by local minimization methods using multiple guesses or direct searching methods and used as the criteria to test the phase stability. In general, the investigated fluid mixture is considered to be unstable with negative TPD functions, and phase splitting may occur in that thermodynamic condition. The trial phase compositions after the phase splitting process is the initialized phase equilibrium conditions for deeper calculation, and several implementation approaches have been proposed for VT-scheme phase splitting tests as well as the whole phase equilibrium calculations. However, these algorithms are designed for unconfined spaces in previous references, which are not capable of dealing with unconventional reservoirs. The basic mechanism of using deep neural network to estimate phase equilibrium conditions is to represent the underlying correlations between input and out thermodynamic properties, which are previously described using EOSs. These correlations are obtained and unearthed with a process analogous to the biological nervous system, while the capability to solve certain engineering problems is determined by the network structure and *hyperparameter* tuning. A fully connected deep ANN is applied in this paper and the network main structure is the same as the schematic diagram illustrated in Fig. 7.1. A small modification is added to the previous structure that we reduce half the original output parameters by introducing a coefficient φ describing the proportion between liquid phase and vapor phase mole fractions. Only the mole fraction of liquid components Y_i remains, and the total output parameters can be reduced a lot in this manner especially for complex fluid mixture with a large number of components, which we believe can improve greatly the training efficiency. This new network structure is illustrated by a schematic diagram as shown in Fig. 7.5. The input parameters remain the same as previous as this is currently the best manner to represent the critical thermodynamic properties of certain various components.

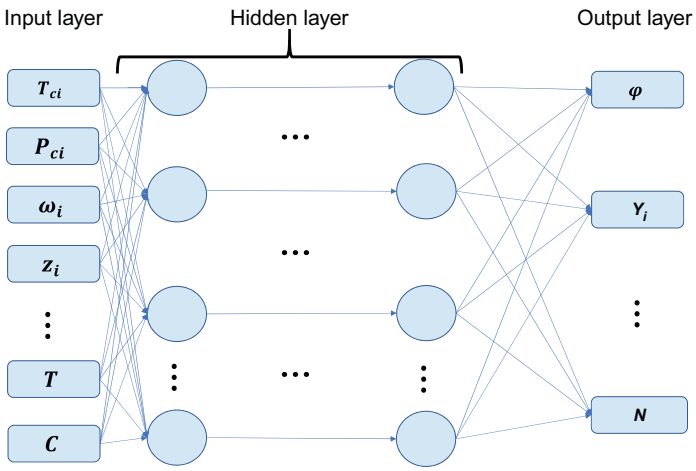

Figure 7.5 Optimized network structure.

A significant highlight of our deep learning algorithm stands on training the neural network to automatically detect the total phase numbers existing in the fluid mixture under certain thermodynamic environment conditions at equilibrium conditions so as to avoid the separate stages of additional stability test. Under this guideline of simultaneously completing phase splitting and phase stability test together at the same time of predicting the vapor and liquid molar compositions, the validation of prediction accuracy can be illustrated by the comparison with the flash calculation data as the ground truth. As shown in Fig. 7.6, the total phase numbers existing in the fluid mixture at equilibrium under the specified overall concentrations of 10 mol/m^3 predicted by the deep learning algorithms meet well with that from the flash calculation data. With temperature increasing, the two-phase mixture will transfer to single vapor phase, and this reasonable phase transition process can be captured successfully by both the flash calculation scheme and deep learning algorithm.

Supercritical fluid, which denotes the substance at a temperature and pressure above its critical point without the existence of distinct vapor and liquid phases, is become increasingly popular in current engineering researches due to the specific potential applications in chemical extraction, dry cleaning, water oxidation or gasification, and carbon capture and storage. Because pressure is not needed as a constant precondition in NVT flash calculation schemes, our accelerated phase equilibrium estimation algorithm is capable of capturing this special mechanism. By checking the value of N and φ, we can determine the supercritical area in certain temperature ranges. As shown in Fig. 7.7 under the specified overall concentrations of 5854.15 mol/m^3, supercritical fluid can be detected by a special label of N as 3, and the perfect match between NVT flash data and deep learning result validates the capability of our deep learning algorithm to detect the supercritical fluid phenomena.

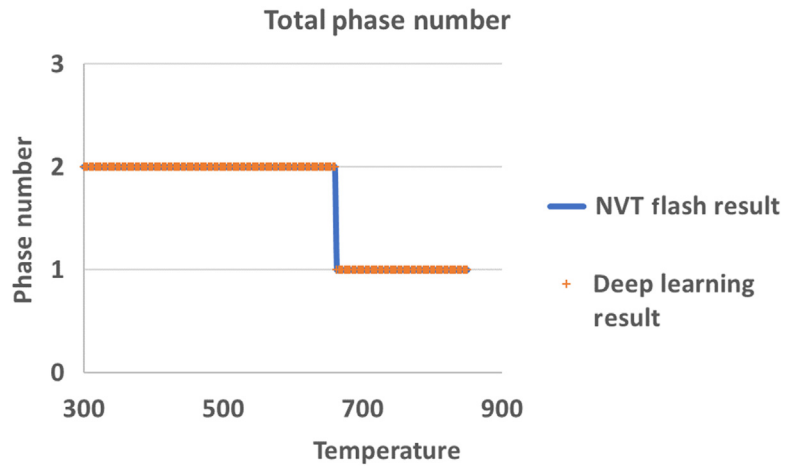

Figure 7.6 Total phase numbers existing in the fluid mixture at equilibrium under the specified overall concentrations of 10 mol/m^3.The temperature unit is "K".

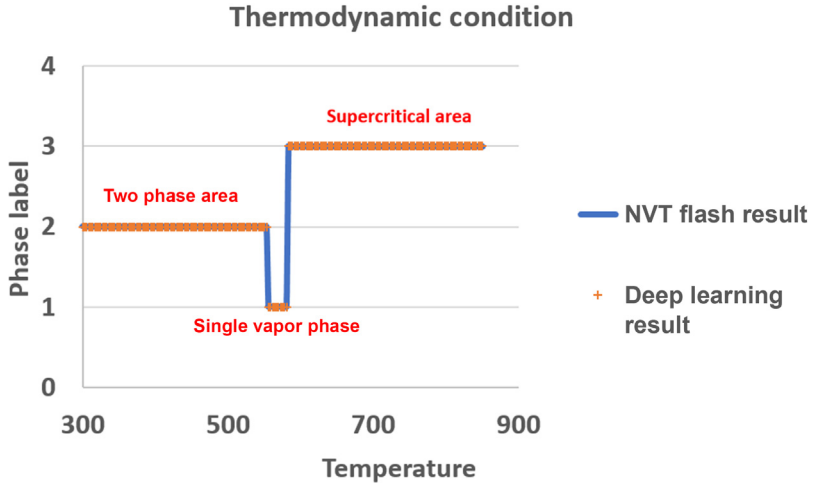

Figure 7.7 Thermodynamic conditions in the fluid mixture at equilibrium under the specified overall concentrations of 5854.15 mol/m^3.The temperature unit is "K".

7.2.3 Network optimization

Phase equilibrium calculation without considering capillary pressure is tested first. However, in order to meet the possible variation caused by output parameter changes, these network *hyperparameters* are optimized and a significant difference has been detected on the number of hidden layers. As shown in Fig. 7.8 the network performance with seven hidden layers is much better than that of five hidden layers, which indicates that we need more hidden layers to capture accurately the thermodynamic correlations in

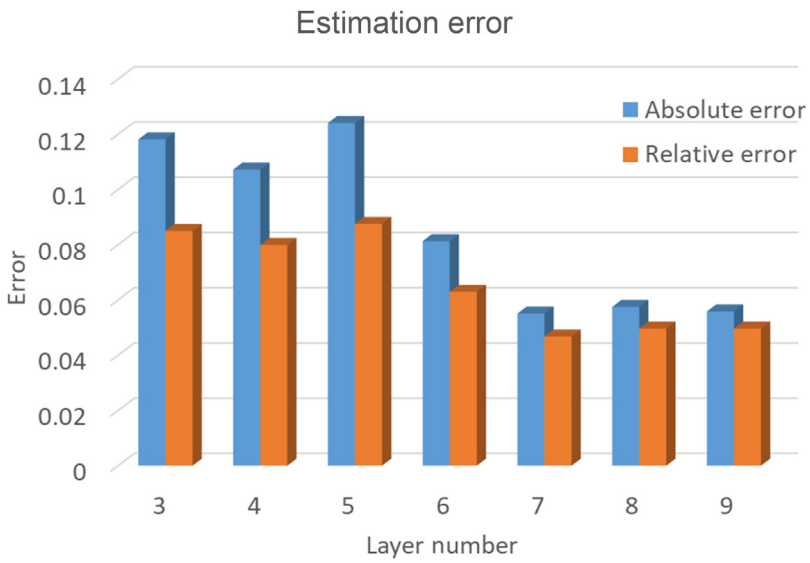

Figure 7.8 Estimation error using deep neural networks with different hidden layers.

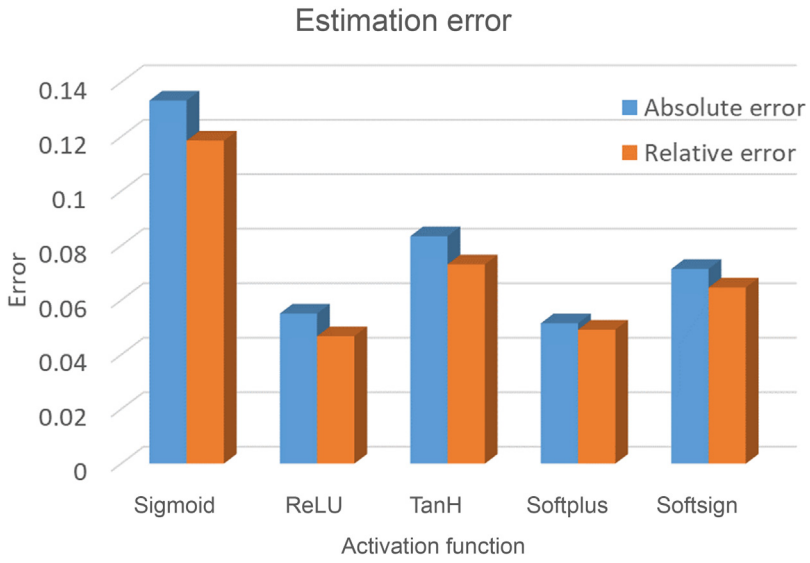

Figure 7.9 Estimation error using deep neural networks with different activation functions.

this network structure. Other network configurations remain the similar performance using previous selections, for example, the estimation error of deep neural networks with different activation functions is illustrated in Fig. 7.9 and "ReLU" is still the best in relative errors. It is interesting to see that the absolute estimation error of the network using "ReLU" is larger than that using "softplus" but the relative error of "ReLU" is smaller.

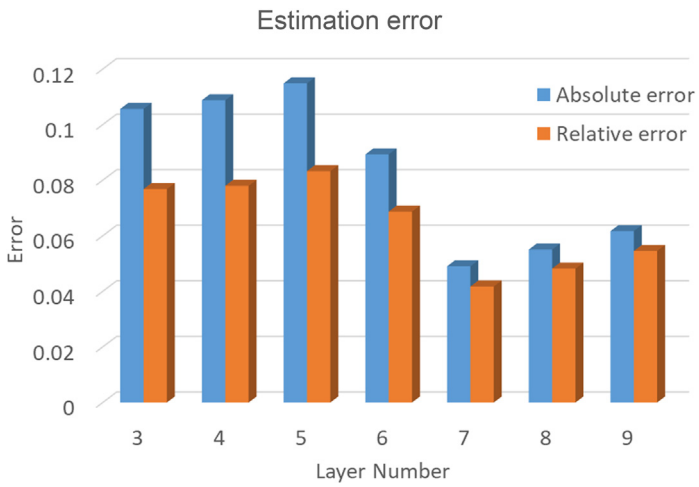

Figure 7.10 Estimation error using deep neural networks with different hidden layers considering capillary pressure.

A possible explanation of this phenomenon can be attributed by the random selections of the training and test samples in the total input data in these two cases.

If phase equilibrium calculations considering the effect of capillary pressure are investigated, the network *hyperparameters* are tuned to optimize the performance under this specific mechanism. To show the wide applicability of the optimized network configurations obtained by previous tuning, the performance of using different numbers of layers is compared and illustrated in Fig. 7.10. It can be referred that the additional considered capillarity mechanisms impact slightly on the performance of deep neural networks with various features, but the best selection still remains the same.

7.3 Realistic case studies

In this section, we will show the capability of our thermodynamic phase equilibrium algorithms and acceleration methods using deep learning algorithms on realistic engineering cases. Parameters used in the NVT flash calculation algorithms and deep learning training and testing are listed in Tables 7.5–7.7 to be referred by the readers.

Case 1: Pure CO_2 equilibrium calculation. Conditions:

T in [220, 320]; % temperature [K]
C in [0 32000]; % overall molar concentration [mol/m^3]
zi = 1; % overall mole fraction
Kij = 0; % binary interaction coefficient
Results: Figs. 7.11–7.14

Table 7.5 Parameters used in example 1−5.

Components	P_c (MPa)	T_c (K)	ω	M_w (g/mol)	[P][a]
N_2	3.390	126.21	0.0390	28.01	41.0
CO_2	7.375	304.14	0.2390	44.01	78.0
C_1	4.599	190.56	0.0110	16.04	77.3
nC_5	3.370	469.70	0.2510	72.15	233.9
C_6	3.012	507.40	0.2960	86.20	271.0
nC_{10}	2.110	617.70	0.4890	142.28	433.5
PC_1	5.329	333.91	0.1113	34.64	97.85
PC_2	3.445	456.25	0.2344	69.52	202.52
PC_3	2.376	590.76	0.4470	124.57	356.82
C_{12+}	1.341	742.58	0.9125	248.30	654.97

[a]Parachor value, unit in dyne$^{0.25}$ cm$^{2.75}$/(g mol), used for the estimation of interfacial tension by Weinaug−Katz correlation.

Table 7.6 Parameters used in example 6.

Components	P_c (MPa)	T_c (K)	ω	M_w (g/mol)	[P]
C_1	4.516	186.12	0.0102	16.54	74.8
C_2	4.978	305.36	0.1028	30.43	107.7
C_3	4.246	369.80	0.1520	44.10	151.9
C_4	3.768	421.60	0.1894	58.12	189.6
C_{5-6}	3.180	486.19	0.2684	78.30	250.2
C_{7-12}	2.505	584.96	0.4291	120.56	350.2
C_{13-21}	1.721	739.87	0.7203	220.72	590.0
C_{22-80}	1.311	1024.54	1.0159	443.52	1216.8

Table 7.7 Compositional parameters of example 7.

	P_c (MPa)	T_c (K)	ω	M_w (g/mol)	[P]
C_1	4.599	190.56	0.0110	16.04	74.05
C_2	4.872	305.33	0.0990	30.07	112.9
C_3	4.248	369.83	0.1520	44.10	154.03
nC_4	3.738	418.71	0.1948	58.12	189.3
CO_2	7.374	304.11	0.2250	44.01	82.0
C_{5-6}	3.377	485.60	0.2398	76.50	247.6
C_{7+}	2.708	606.69	0.3548	122.96	402.3
C_{13+}	1.560	778.31	0.7408	255.28	834.8

Case 2: Binary component thermodynamic equilibrium of C1 & nC5. Conditions:

T in [250, 450];
C in [0, 15000];
zi = [0.489575; 0.510425];
Kij = [0.000 0.041; 0.041 0.000];
Results: Figs. 7.15−7.18

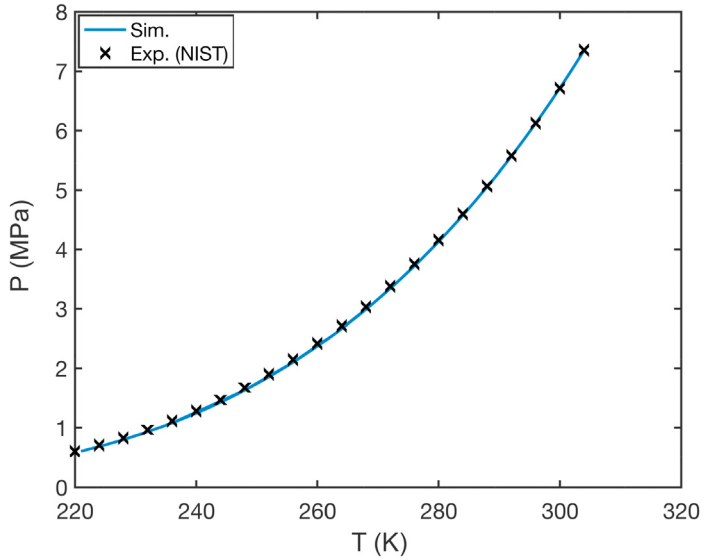

Figure 7.11 Phase envelope of bulk CO_2.

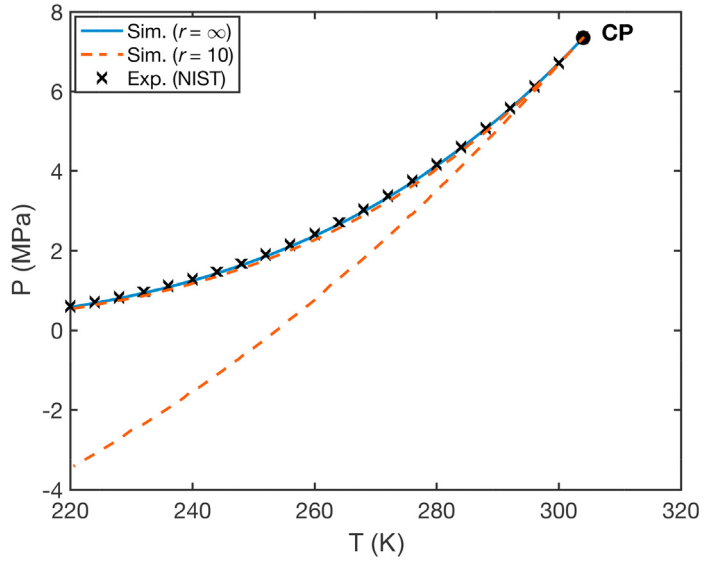

Figure 7.12 Phase envelope of confined CO_2 with $r = 10$.

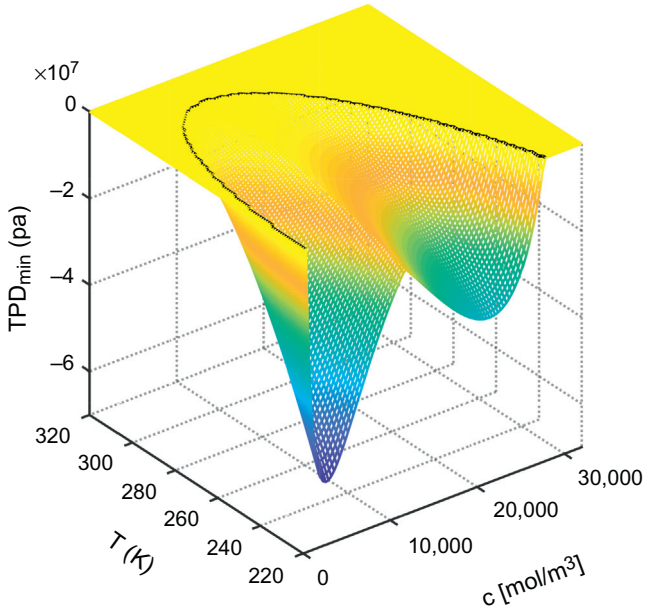

Figure 7.13 TPD of bulk CO_2. *TPD*, Tangent plane distance.

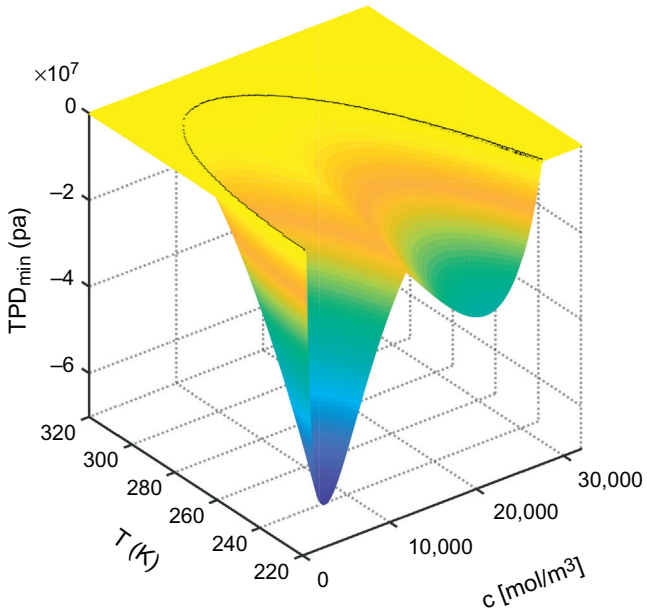

Figure 7.14 TPD of confined CO_2 with $r = 10$. *TPD*, Tangent plane distance.

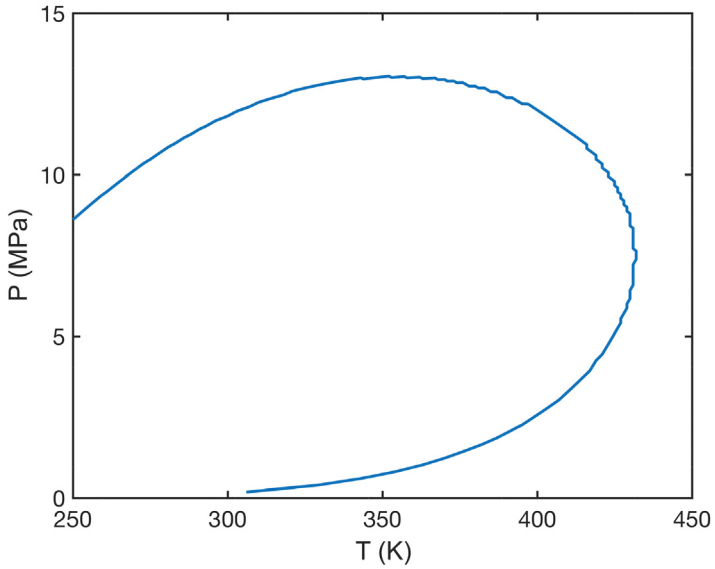

Figure 7.15 Phase envelope of bulk C1 & nC5 mixture.

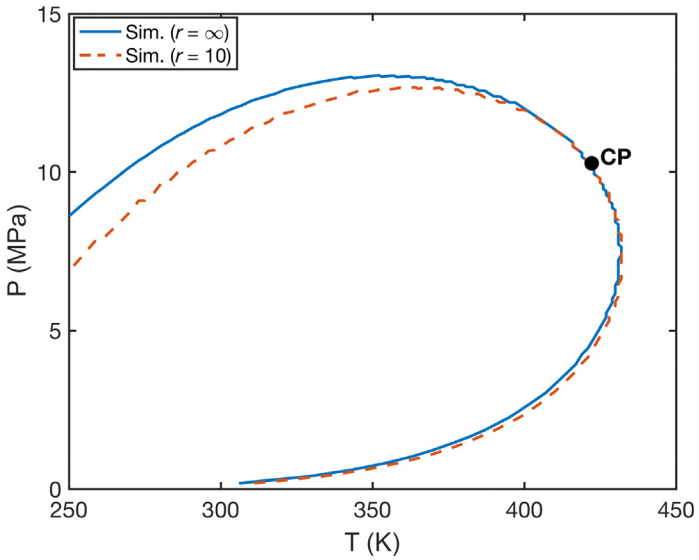

Figure 7.16 Phase envelope of confined C1 & nC5 mixture with $r = 10$.

Case 3: Three component mixture thermodynamic equilibrium of C1 & C6 & nC10. Conditions:

T in [350, 600];

C in [0, 9000];

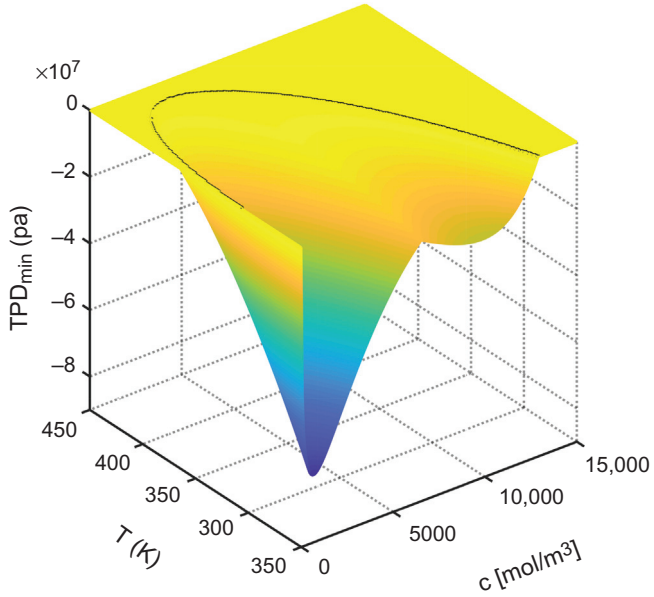

Figure 7.17 TPD of bulk C1 & nC5 mixture. *TPD*, Tangent plane distance.

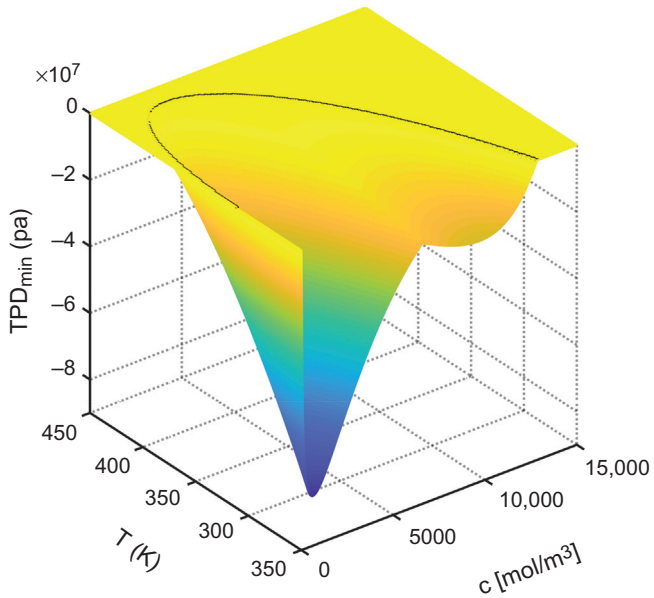

Figure 7.18 TPD of confined C1 & nC5 mixture with $r = 10$. *TPD*, Tangent plane distance.

zi = [0.405946;0.297027;0.297027];
Kij = [0.000 0.043 0.052; 0.043 0.000 0.000; 0.052 0.000 0.000];
Results: Figs. 7.19–7.22

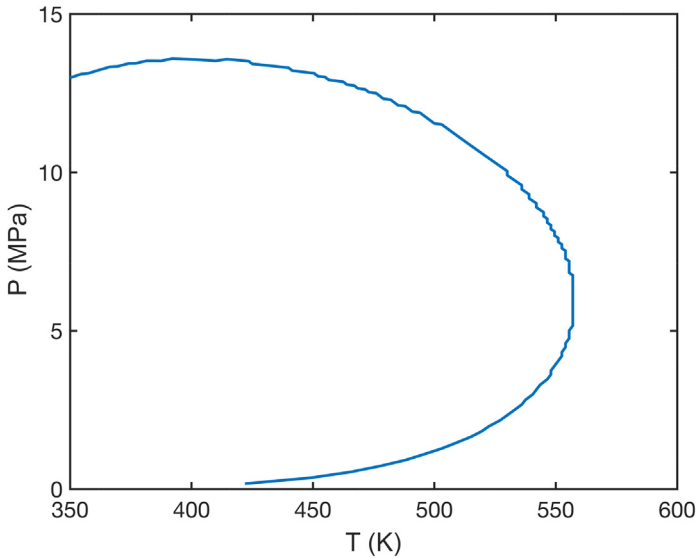

Figure 7.19 Phase envelope of bulk C1 & C6 & nC10 mixture.

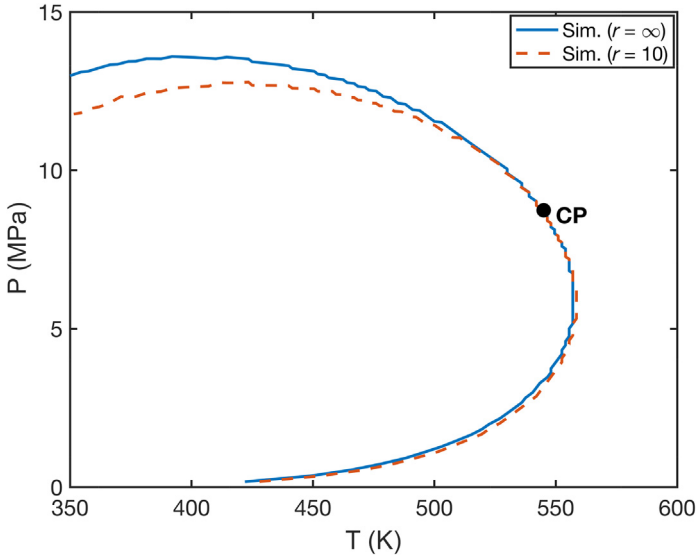

Figure 7.20 Phase envelope of confined C1 & C6 & nC10 mixture with $r = 10$.

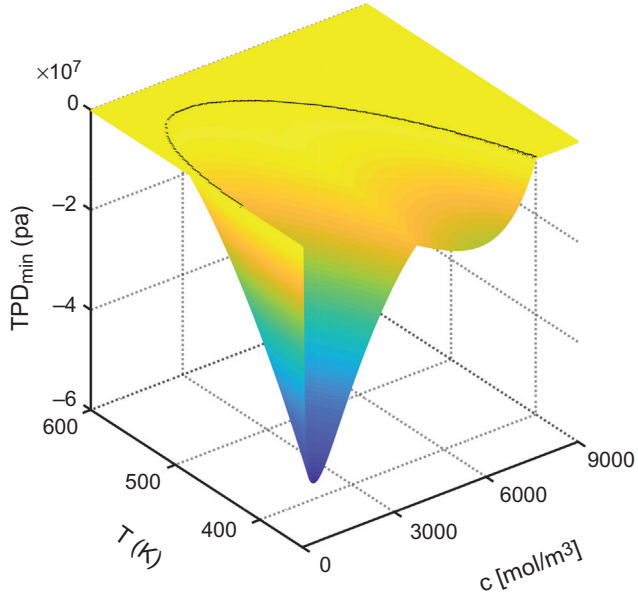

Figure 7.21 TPD of bulk C1 & C6 & nC10 mixture. *TPD*, Tangent plane distance.

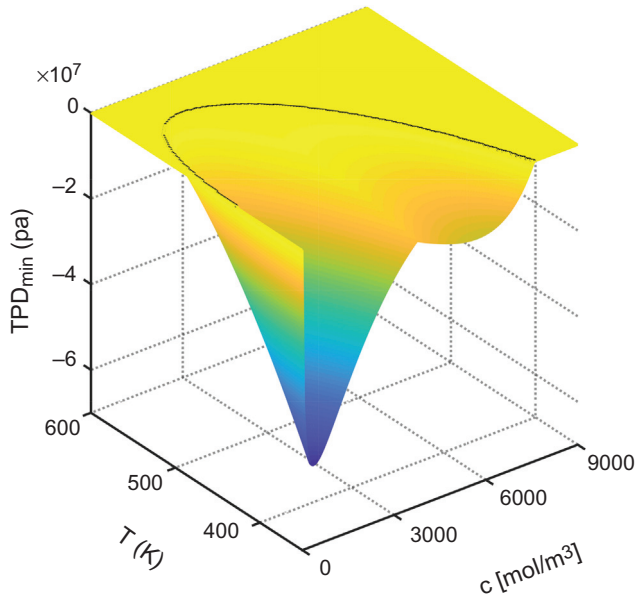

Figure 7.22 TPD of confined C1 & C6 & nC10 mixture with $r = 10$. *TPD*, Tangent plane distance.

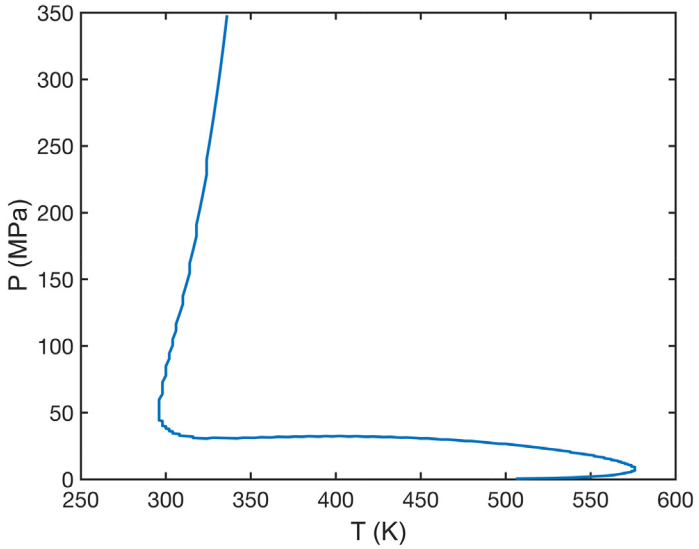

Figure 7.23 Phase envelope of bulk N_2 & CO_2 & C_1 & PC_1 & PC_2 & PC_3 & C_{12+} mixture.

Case 4: Seven component mixture thermodynamic equilibrium of N_2 & CO_2 & C_1 & PC_1 & PC_2 & PC_3 & C_{12+}. Conditions:

T in [250, 650];

C in [0, 20000];

zi = [0.000131; 0.568185; 0.246739; 0.086275; 0.033722; 0.037006; 0.027941];

Kij = [0.000 0.000 0.100 0.100 0.100 0.100 0.100;
0.000 0.000 0.150 0.150 0.150 0.150 0.150;
0.100 0.150 0.000 0.035 0.040 0.049 0.069;
0.100 0.150 0.035 0.000 0.000 0.000 0.000;
0.100 0.150 0.040 0.000 0.000 0.000 0.000;
0.100 0.150 0.049 0.000 0.000 0.000 0.000;
0.100 0.150 0.069 0.000 0.000 0.000 0.000];

Results: Figs. 7.23–7.26

Case 5: Seven component mixture thermodynamic equilibrium of N_2 & CO_2 & C_1 & PC_1 & PC_2 & PC_3 & C_{12+}. Conditions:

T in [250, 650];

C in [0, 20000];

zi = [0.466905; 0.007466; 0.300435; 0.105051; 0.041061; 0.045060; 0.034022];

Kij = [0.000 0.000 0.100 0.100 0.100 0.100 0.100;
0.000 0.000 0.150 0.150 0.150 0.150 0.150;

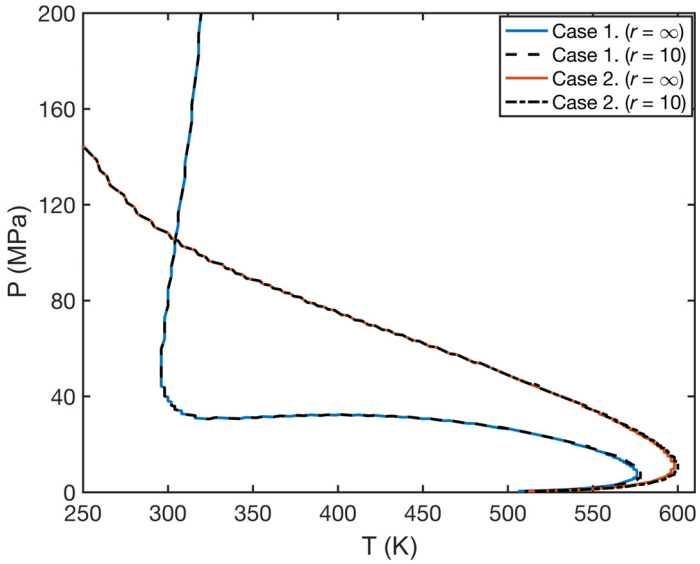

Figure 7.24 Phase envelope of confined N_2 & CO_2 & C_1 & PC_1 & PC_2 & PC_3 & C_{12+} mixture with $r = 10$.

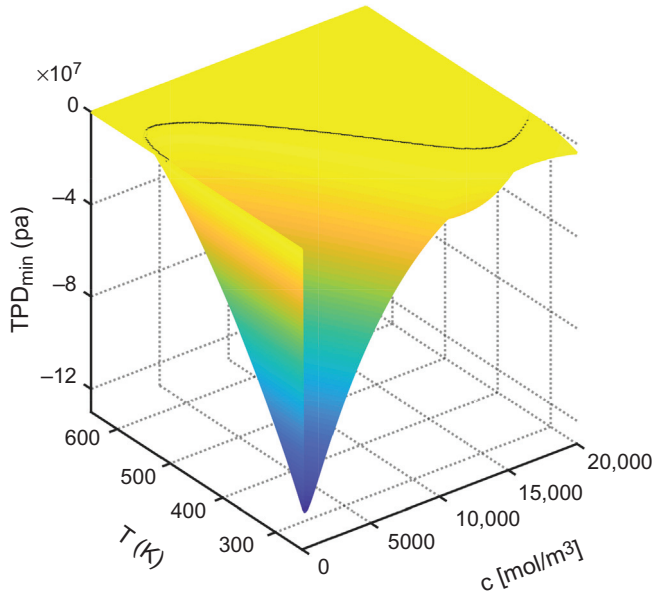

Figure 7.25 TPD of bulk N_2 & CO_2 & C_1 & PC_1 & PC_2 & PC_3 & C_{12+} mixture. *TPD*, Tangent plane distance.

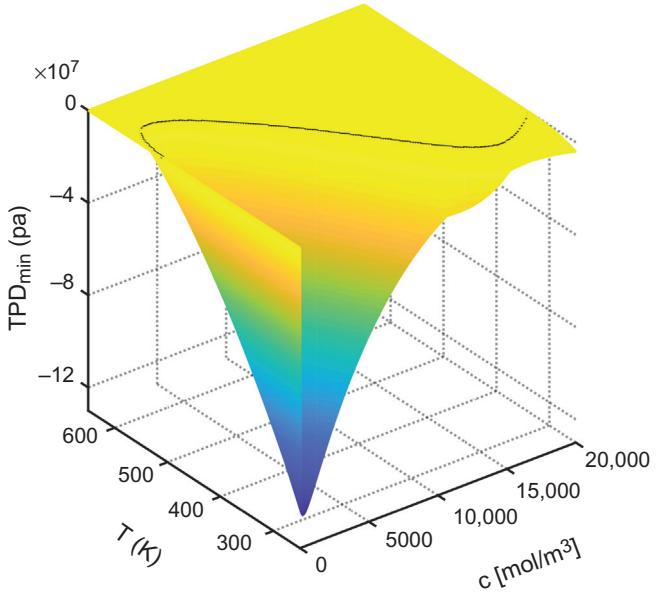

Figure 7.26 TPD of confined N_2 & CO_2 & C_1 & PC_1 & PC_2 & PC_3 & C_{12+} mixture with $r = 10$. *TPD*, Tangent plane distance.

0.100 0.150 0.000 0.035 0.040 0.049 0.069;
0.100 0.150 0.035 0.000 0.000 0.000 0.000;
0.100 0.150 0.040 0.000 0.000 0.000 0.000;
0.100 0.150 0.049 0.000 0.000 0.000 0.000;
0.100 0.150 0.069 0.000 0.000 0.000 0.000];

Results: Figs. 7.27–7.30

Case 6: Bakken Oil. Conditions:

T in [300, 850];
C in [10, 10000];
Molname = {'C1','C2','C3','C4','C5–6','C7–12','C13–21','C22–80'};
zi = [0.36736; 0.14885; 0.09334; 0.05751; 0.06406; 0.15854; 0.0733; 0.03704];
Kij = [0.0000 0.0050 0.0035 0.0035 0.0037 0.0033 0.0033 0.0033;
0.0050 0.0000 0.0031 0.0031 0.0031 0.0026 0.0026 0.0026;
0.0035 0.0031 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000;
0.0035 0.0031 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000;
0.0037 0.0031 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000;
0.0033 0.0026 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000;
0.0033 0.0026 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000;
0.0033 0.0026 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000];

Results: Figs. 7.31–7.34

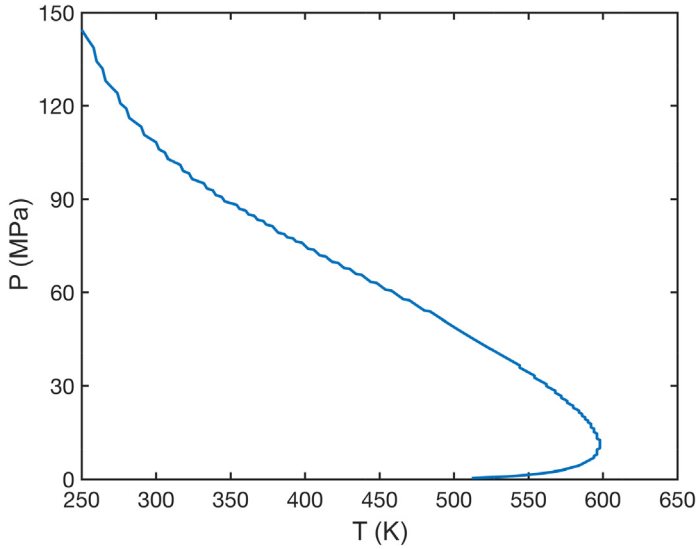

Figure 7.27 Phase envelope of bulk N_2 & CO_2 & C_1 & PC_1 & PC_2 & PC_3 & C_{12+} mixture.

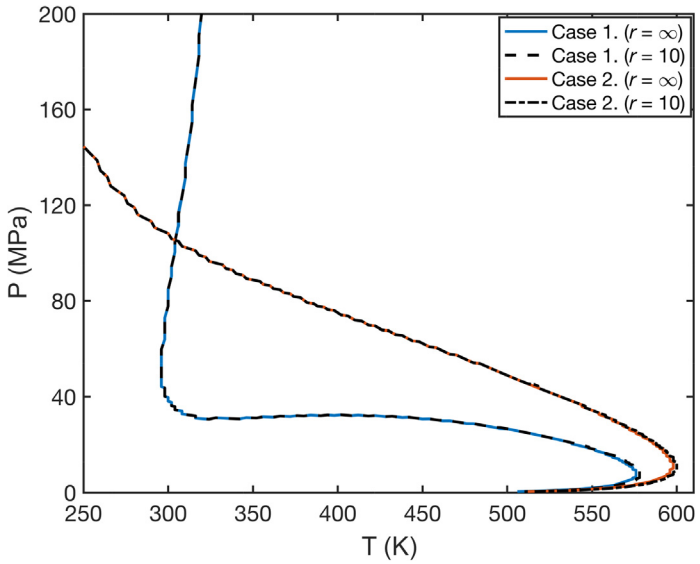

Figure 7.28 Phase envelope of confined N_2 & CO_2 & C_1 & PC_1 & PC_2 & PC_3 & C_{12+} mixture with $r = 10$.

Case 7: EagleFord Oil. Conditions:
T in [260, 700];
C in [10, 12000];

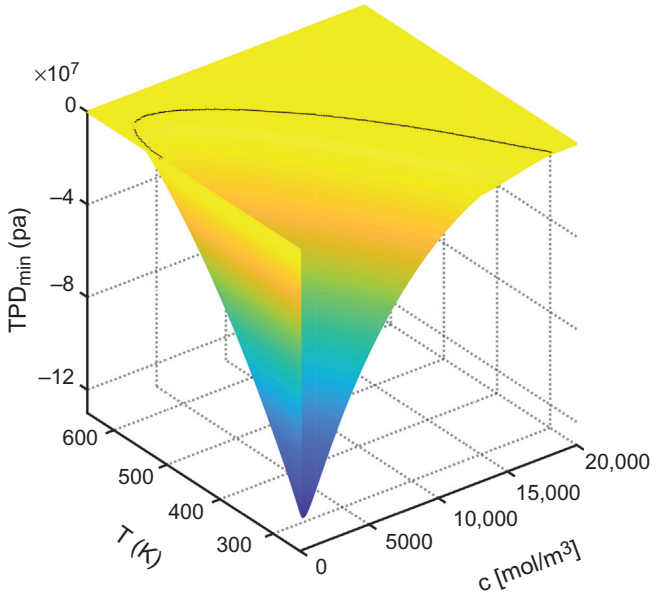

Figure 7.29 TPD of bulk N_2 & CO_2 & C_1 & PC_1 & PC_2 & PC_3 & C_{12+} mixture. *TPD*, Tangent plane distance.

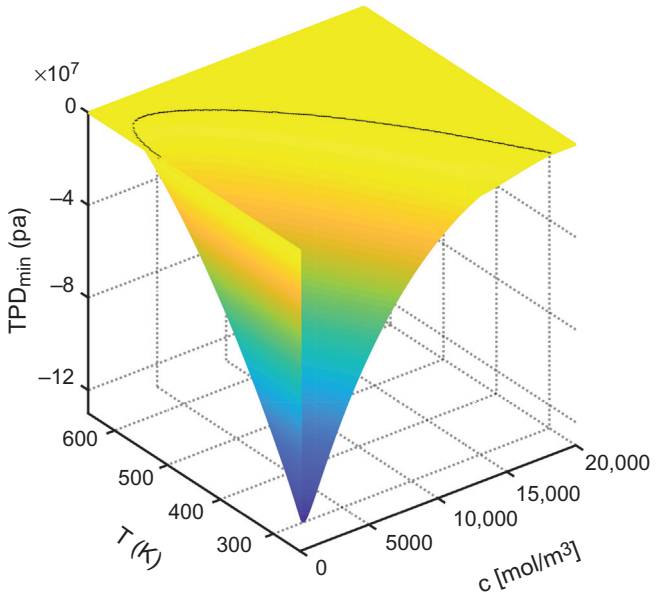

Figure 7.30 TPD of confined N_2 & CO_2 & C_1 & PC_1 & PC_2 & PC_3 & C_{12+} mixture with $r = 10$. *TPD*, Tangent plane distance.

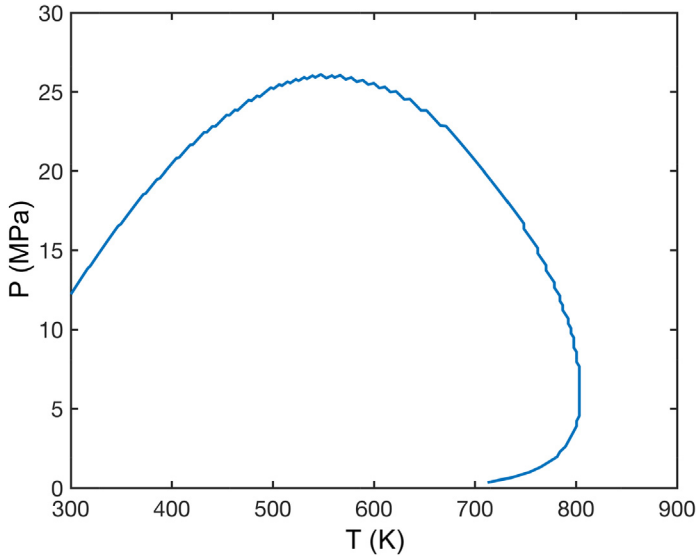

Figure 7.31 Phase envelope of bulk Bakken oil mixture.

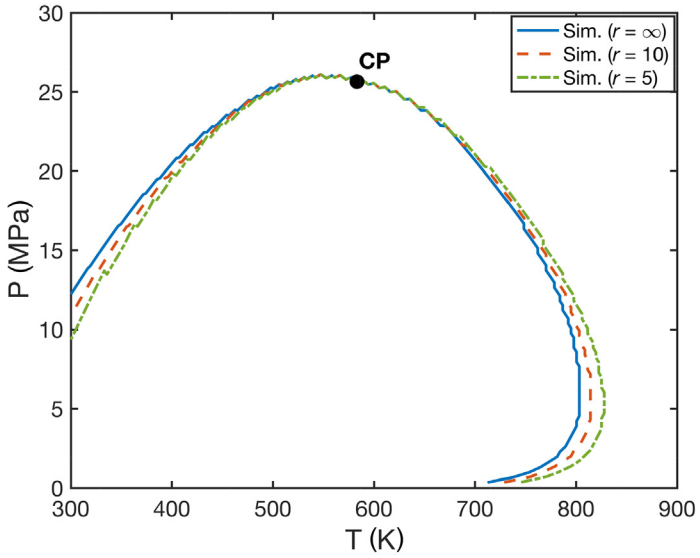

Figure 7.32 Phase envelope of confined Bakken oil mixture with $r = 5$ and 10.

```
Molname = {'C1','C2','C3','nC4','CO2','C5−6','C7 + ','C13 + '};
zi = [0.5816;0.0744;0.0417;0.0259;0.0232;0.0269;0.1321;0.0942];
Kij = [0.0000 0.0000 0.0000 0.0000 0.1200 0.0000 0.0000 0.0000;
```

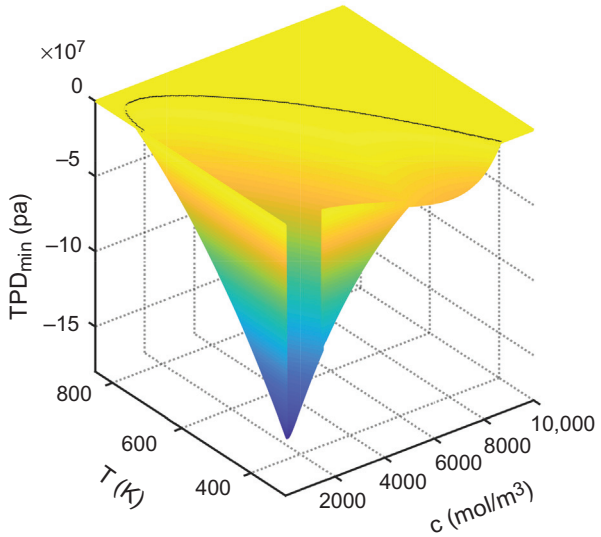

Figure 7.33 TPD of bulk Bakken oil mixture. *TPD*, Tangent plane distance.

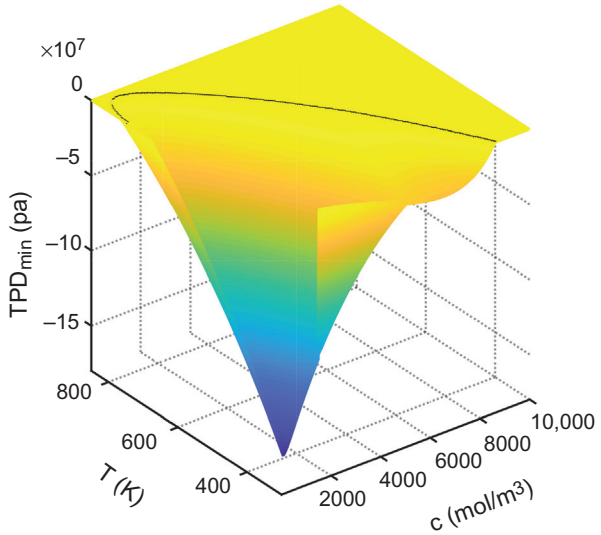

Figure 7.34 TPD of confined Bakken oil mixture with $r = 10$. *TPD*, Tangent plane distance.

0.0000 0.0000 0.0000 0.0000 0.1200 0.0000 0.0000 0.0000;
0.0000 0.0000 0.0000 0.0000 0.1200 0.0000 0.0000 0.0000;
0.0000 0.0000 0.0000 0.0000 0.1200 0.0000 0.0000 0.0000;
0.1200 0.1200 0.1200 0.1200 0.0000 0.1200 0.1000 0.1000;
0.0000 0.0000 0.0000 0.0000 0.1200 0.0000 0.0000 0.0000;

```
0.0000 0.0000 0.0000 0.0000 0.1000 0.0000 0.0000 0.0000;
0.0000 0.0000 0.0000 0.0000 0.1000 0.0000 0.0000 0.0000];
```

Results: Figs. 7.35–7.37

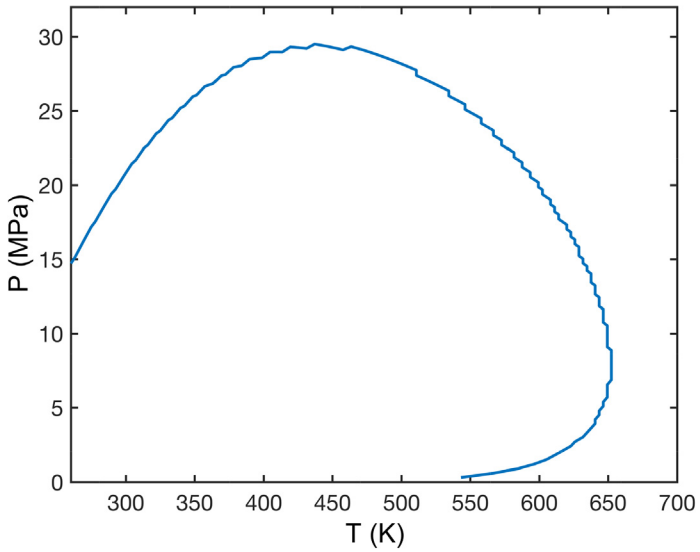

Figure 7.35 Phase envelope of Bulk EagleFord oil mixture.

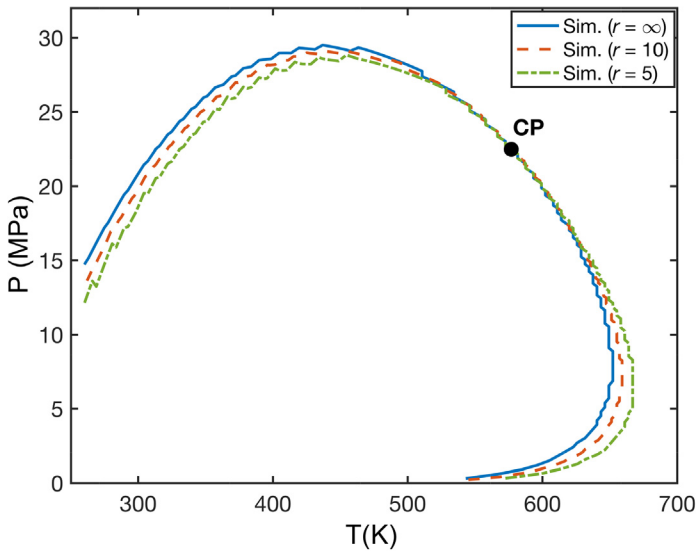

Figure 7.36 Phase envelope of confined EagleFord oil mixture with $r = 5$ and 10.

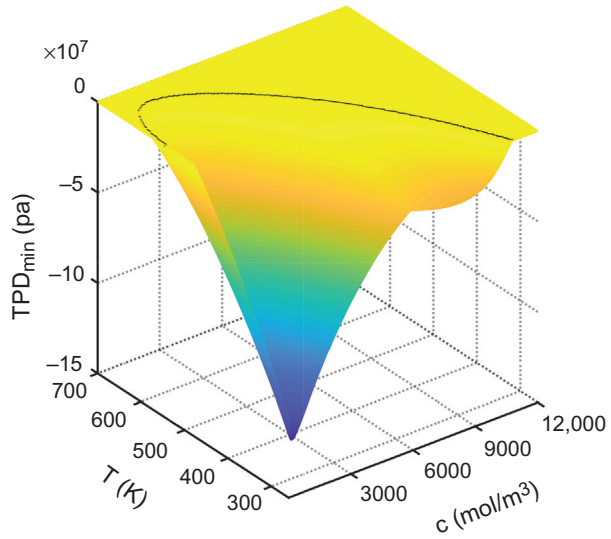

Figure 7.37 TPD of bulk EagleFord oil mixture. *TPD*, Tangent plane distance.

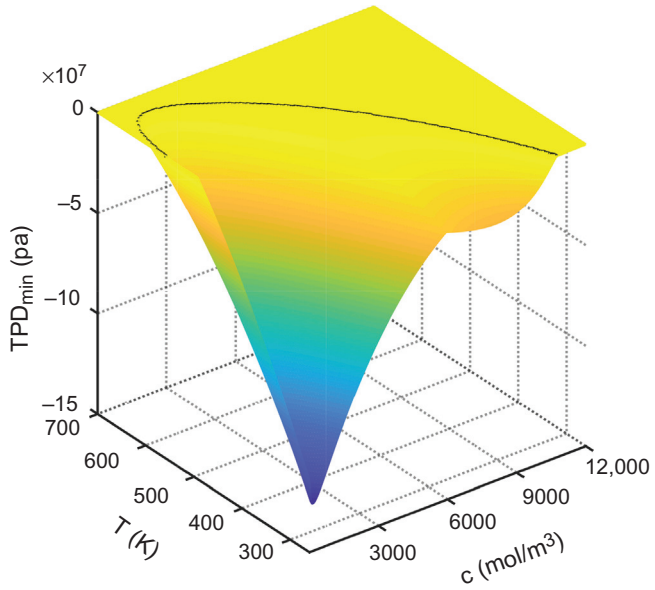

Figure 7.38 TPD of confined EagleFord Oil mixture with $r = 10$. *TPD*, Tangent plane distance.

The effect of capillary pressure on thermodynamic equilibrium conditions with various pore size distributions can be illustrated by the comparisons between Figs. 7.34, 7.38–7.40. It can be referred that the bulk phase envelope is reshaped if

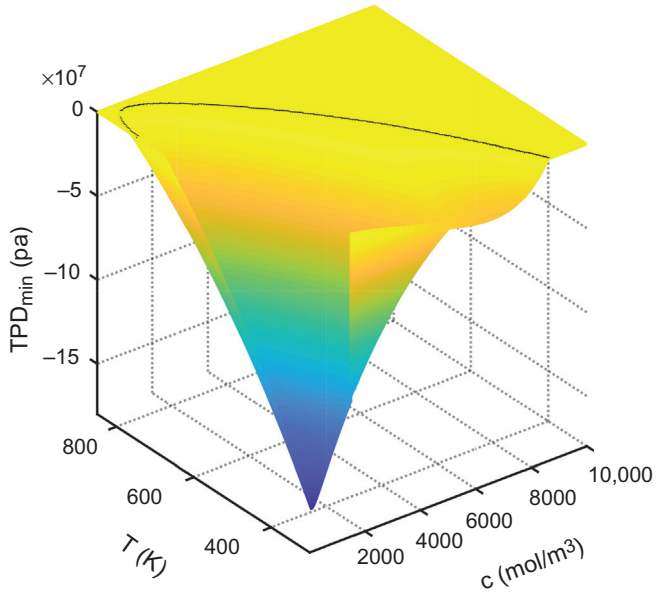

Figure 7.39 TPD of confined Bakken oil mixture with $r = 5$. *TPD*, Tangent plane distance.

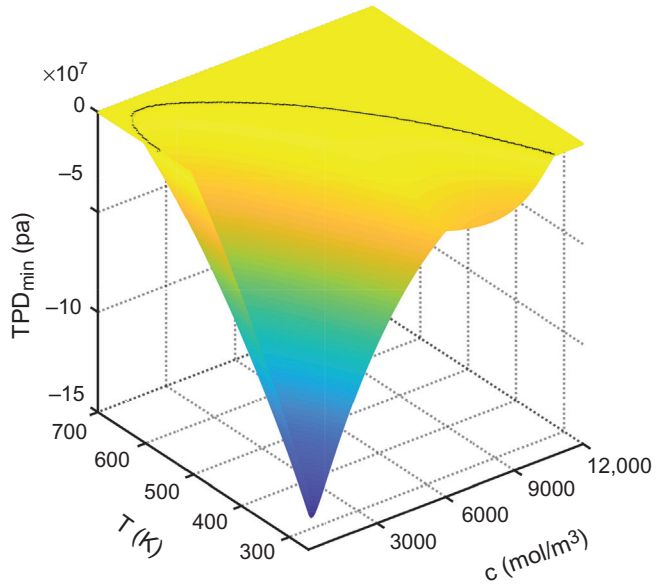

Figure 7.40 TPD of confined EagleFord oil mixture with $r = 5$. *TPD*, Tangent plane distance.

the capillary effect is considered with certain *nano*-scale pore size, and the bubble point curve is suppressed significantly. On the other hand, the dew point curve is expanded outward, which indicates a decreasing dew point pressure, as shown in the lower branch of dew point curve, in the presence of capillary pressure and confinement effect.

References

Baker, L.E., Pierce, A.C., Luks, K.D., 1982. Gibbs energy analysis of phase equilibria. Soc. Pet. Eng. J. 22 (05), 731–742.

Kou, J., Sun, S., 2015. Numerical methods for a multicomponent two-phase interface model with geometric mean influence parameters. SIAM J. Sci. Comput. 37 (4), B543–B569.

Kou, J., Sun, S., 2018a. A stable algorithm for calculating phase equilibria with capillarity at specified moles, volume and temperature using a dynamic model. Fluid Phase Equilibria 456, 7–24.

Kou, J., Sun, S., 2018b. Thermodynamically consistent modeling and simulation of multi-component two-phase flow with partial miscibility. Comput. Methods Appl. Mech. Eng. 331, 623–649.

Li, Z., Firoozabadi, A., 2012. General strategy for stability testing and phase-split calculation in two and three phases. SPE J. 17 (04), 1–096.

Li, Y., Zhang, T., Sun, S., Gao, X., 2019a. Accelerating flash calculation through deep learning methods. J. Comput. Phys. 394, 153–165.

Li, Y., Zhang, T., Sun, S., 2019b. Acceleration of the NVT flash calculation for multicomponent mixtures using deep neural network models. Ind. Eng. Chem. Res. 58 (27), 12312–12322.

Mathias, P.M., 1983. A versatile phase equilibrium equation of state. Ind. Eng. Chem. Process. Des. Dev. 22 (3), 385–391.

Pedersen, K.S., et al., 2006. Phase Behavior of Petroleum Reservoir Fluids. CRC Press.

Shen, J., Xu, J., Yang, J., 2018. The scalar auxiliary variable (SAV) approach for gradient flows. J. Comput. Phys. 353, 407–416.

Sun, D.L., Yu, S., Yu, B., Wang, P., Liu, W.J., 2017. A VOSET method combined with IDEAL algorithm for 3D two-phase flows with large density and viscosity ratio. Int. J. Heat. Mass. Transf. 144, 155–168.

Tao, Z., Li, Y., Sun, S., 2019. Accelerated phase equilibrium predictions for subsurface reservoirs using deep learning methods. International Conference on Computational Science. Springer, Cham.

Zhang, T., Kou, J., Sun, S., 2017. Review on dynamic Van der Waals theory in two-phase flow. Adv. Geo-Energy Res. 1 (2), 124–134.

Zhang, T., Salama, A., Sun, S., et al., 2015. A compact numerical implementation for solving Stokes equations using matrix-vector operations. Procedia Comput. Sci. 51, 1208–1218.

Zhu, G., et al., 2019. Thermodynamically consistent modelling of two-phase flows with moving contact line and soluble surfactants. J. Fluid Mech. 879, 327–359.