MACHINE LEARNING APPROACHES FOR PERMANENT DOWNHOLE GAUGE DATA INTERPRETATION

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ENERGY
RESOURCES ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Chuan Tian

June 2018

This dissertation is online at: http://purl.stanford.edu/wt550nf6227

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Roland Horne, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Jef Caers

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Tapan Mukerji

Approved for the Stanford University Committee on Graduate Studies.

Patricia J. Gumport, Vice Provost for Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.

Abstract

The continuous record of pressure, temperature, and sometimes flow rate by Permanent Downhole Gauges (PDGs) contains rich information about the reservoir. Due to the different conditions where data are collected, PDG data have unique characteristics that require different interpretation approaches than conventional well testing. One of the machine learning approaches developed earlier was the convolution-kernel-based data mining (Liu and Horne, 2013a, 2013b). The method was shown to be promising for pressure-rate deconvolution on noisy data. However, the bottlenecks of computation efficiency and incomplete recovery of reservoir behaviors limit the application of the convolution kernel method to interpret real PDG data.

In this work, two classes of methods were applied for PDG data interpretation. One was the machine learning approach with feature handcrafting based on physics, the other was the deep learning approach by using recurrent neural networks to learn on raw PDG measurements. Both approaches were shown effective to extract the patterns containing the reservoir information from PDG data. The extracted patterns were demonstrated to be useful in three ways: one was deconvolution by predicting the pressure corresponding to a constant rate history; the other two were to predict pressure in the future with given rate control, and vice versa.

Three feature-based machine learning techniques were explored to interpret PDG

data, namely, the linear approach, the kernel method, and kernel ridge regression. First the linear approach was proved to learn with the same features as the previous convolution kernel method, while it required much lower computational cost. The linear approach was used further for pressure-rate deconvolution and it outperformed two conventional deconvolution methods when noise or outliers were contained in the data. Next the kernel method and kernel ridge regression were utilized to adjust the machine learning model capability and to model the pressure behaviors covering early-transient until late-transient periods. Additional machine learning models were developed for the problems of rate reconstruction and multiwell testing, by handcrafting new set of features.

Recurrent neural networks were investigated for PDG data interpretation with a focus on nonlinear autoregressive exogenous models (NARX). It was shown that NARX learned the patterns behind PDG time series data with its uniquer recurrent architecture, by feeding the memory of previous computations back into the model. NARX was also applied for pressure-rate deconvolution and it identified different reservoir models successfully. One key advantage of NARX is that it learns on raw data, although thought and effort are needed to tune the neural network. A case study on temperature transient analysis demonstrated NARX's advantage over feature-based machine learning when feature handcrafting was difficult because of limited knowledge of the physics.

Acknowledgments

First of all, I would like to express my sincere gratitude to my advisor Prof. Roland Horne, for his guidance, support and patience during my study at Stanford. His responsibility, pursuit for perfection and love for research always motivates me over the years, and makes him a true role model for me to follow. It is my great honor to be his PhD student and study under his supervision.

I would also like to thank Prof. Jef Caers, Prof. Tapan Mukerji, and Prof. Daniel Tartakovsky for serving as members of my defense committee, and Prof. Hector Garcia-Molina for chairing my oral exam. Prof. Jef Caers and Prof. Tapan Mukerji were also on my PhD qualifying exam committee, thank you for your precious time in reading this thesis and providing your constructive comments.

I am also grateful to Prof. Lou Durlofsky, who kindly attended all my talks at the Smart Fields meetings and gave me insightful suggestions. Thanks also go to Dr. Yang Liu for sharing his research code with me and answering my questions about his research.

I would also like to thank all the faculty, staff and students in the Energy Resources Engineering Department for their kind assistance in my study, research and personal life. Special thanks go to my colleagues at SUPRI-D and Smart Fields Consortium for their support and discussions.

The generous financial support from Smart Fields Consortium is greatly appreciated. I am deeply grateful to the companies that gave access to field data. My gratitude also goes to Noodle.ai, Chevron, QRI and Halliburton for providing me the internship opportunities.

Parts of this research have been presented previously in SPE papers SPE-174034-MS (Tian and Horne, 2015a), SPE-175059-MS (Tian and Horne, 2015b), SPE-181556-MS (Tian and Horne, 2016), and SPE-187181-MS (Tian and Horne, 2017). Components of those papers that appear here are copyright to Society of Petroleum Engineers (SPE).

Lastly, I want to say thank you to my parents. Your love is the best support for me and always encourages me to be a better person.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

1.1.1 Well Testing and PDG Data Interpretation

Well testing has been widely used in petroleum industry as a key approach for reservoir characterization. During the well testing period, a production well is either shut in or is controlled at a constant flowing rate, and the corresponding well test is referred to as a buildup test or a drawdown test respectively. A well test typically lasts for a period ranging from several hours to a few days. The pressure response during the well test is recorded and analyzed to characterize reservoir parameters such as permeability, reservoir boundary, wellbore storage and skin factor.

Here is an example of a synthetic drawdown test, where an oil well flows at a constant rate for 50 hours (Figure 1.1(a)). Figure 1.1(b) shows the pressure (dashed line) and pressure derivative (solid line) during the test period on log-log scale. The log-log plot can be used by a well testing engineer to identify different flow regimes

and the reservoir model.

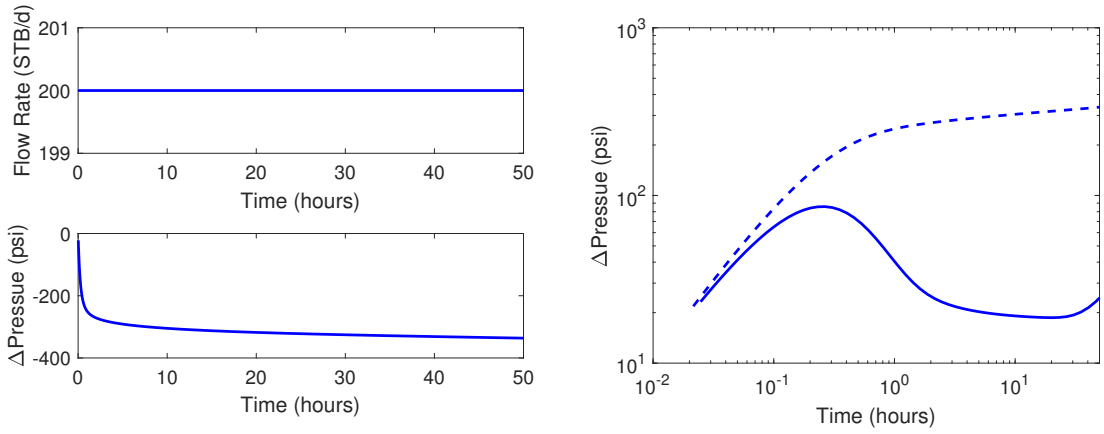


Figure 1.1: A synthetic drawdown test lasting for 50 hours. (a) Flow rate and pressure histories of the drawdown test. (b) Pressure change (dashed line) and pressure derivative (solid line) on log-log scale.

A permanent downhole gauge (PDG) is a device installed permanently in an oil well, to provide a continuous record of pressure, temperature, and sometimes flow rate during production. PDGs were first deployed as back far as 1963 (Nestlerode, 1963), and were used mainly for well monitoring during their early deployments. Later on researchers gradually realized the value of PDG data for reservoir characterization, given the continuous record of reservoir performance in PDGs. PDG data interpretation became a new direction to explore for well testing researchers.

However, PDG data have certain characteristics different from well testing data, as PDG data are recorded under different settings, e.g. continuously variable flow rate, much longer duration. Those characteristics make the interpretation techniques developed for well testing not suitable for analyzing PDG data. Some characteristics of the PDG data and associated challenges for interpretation are summarized as follows.

Continuously variable flow rate: PDG data are recorded subjecting to operational variations in flow rate. Throughout the production history of a well, there may be only few (if any) shut-in periods that can be analyzed by conventional well test interpretation techniques. Thus deconvolution, a process of extracting the constant rate pressure from the multirate pressure signal, is required for analyzing PDG data. A comparison of deconvolution and conventional derivative analysis on synthetic data with two shut-in periods is shown in Figure 1.2. In practice, the conventional derivative analysis is limited to the short shut-in periods because of the poor data quality when the well is open. The log-log plot of the two corresponding pressure buildups (PBUs) in Figure 1.2(b) reveals little information about the reservoir boundary, while the deconvolution on the full multirate data in Figure 1.2(c) shows the existence of a closed boundary clearly. This example illustrates the advantage of deconvolution in extracting long term reservoir behavior from multirate data. The current deconvolution techniques, however, heavily rely on data preprocessing such as denoising and breakpoint detection, which can be error-prone.

Noise in data: Noise is commonly observed in PDG record. As discussed by Horne (2007), the noise comes from the operational variations that occur in the well, and should be treated as an inherent property of PDG data. The noise raises several challenges for analyzing PDG data. For instance, noise can greatly affect the accuracy of some of the deconvolution techniques applied in industry, such as the techniques developed by von Schroeter et al. (2004) and Levitan et al. (2006), which can lead to incorrect interpretation.

Large data volume: Modern PDGs can record data at a frequency as high as once per second. Thus a PDG can easily accumulate millions of data after years of

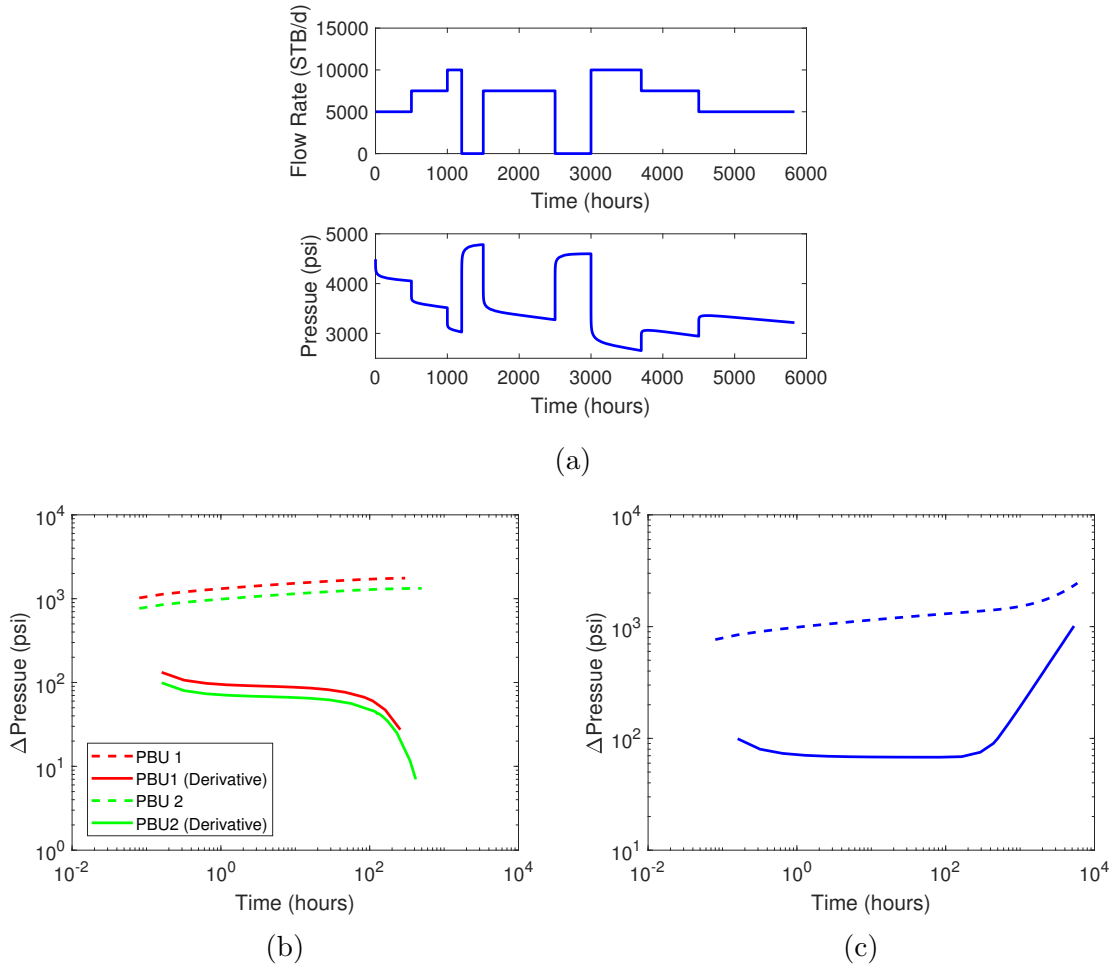


Figure 1.2: (a) Synthetic multirate data with two pressure buildup (PBU) periods when the well is shut in. The first PBU period is from 1200 hour to 1500 hour, and the second PBU period is from 2500 hour to 3000 hour. (b) Conventional derivative analysis on the two PBUs. (c) Theoretical deconvolved pressure based on the full multirate data. Note that the time scale of deconvolution in (c) is much longer than derivative analysis in (b), which leads to a larger radius of investigation.

production. The amount of data for analysis from a PDG record is much larger than that from conventional well testing, which may only last for several days. Hence, we wish to develop a robust method to process and interpret this large volume of data from PDGs in an automated manner.

Temperature measurements: Unlike pressure and rate data, temperature data measured by PDGs have been largely ignored until recently. Research by Duru and Horne (2010) indicates that temperature is very informative, as it is highly correlated with rate and pressure. This demonstrated that temperature data from PDGs could serve as a substitute for flow rate data in reservoir analysis, which is a big advantage as temperature measurements are easier to obtain than flow rate data.

A variety of research has been done to address the difficulties of interpreting PDG data as listed here. The current standard approach for PDG data interpretation is to apply nonlinear regression after data processing, which includes removing outliers, denoising, detecting breakpoints, etc. However, as we will see in Section 1.2.1, each step of data processing requires careful manipulation and can be error-prone. Even small inaccuracy in processing the data may lead to failure in identifying the reservoir model (Nomura, 2006). Besides, nonlinear regression can be very sensitive to the initial guess on parameters, and requires us to identify the reservoir model in advance. Another bottleneck is the scalability of nonlinear regression on large volumes of PDG data, due to the expensive computation required by nonlinear regression. All those limitations make current methodologies inefficient for practical use. Hence PDG data interpretation remains a challenging topic for investigation.

1.1.2 Machine Learning Based PDG Data Interpretation

Machine learning, a subfield of artificial intelligence, refers to the family of techniques that learn the mapping from the input to the output. Machine learning has demonstrated its power in our daily life, from identifying spam emails to recommending products for online shopping. Given machine learning as a powerful tool to extract the patterns behind the data, it is very attractive to apply machine learning on PDG data to extract the correlation between flow rate and pressure (and/or temperature). As pressure is the reservoir response to flow rate, the extracted pattern implicitly contains the reservoir information, which can be recovered in some appropriate manner. Some potential advantages of machine learning based PDG data interpretation are discussed as follows.

Deconvolution: Figure 1.3 illustrates conceptually how to deconvolve PDG pressure signal by using machine learning approaches. Traditionally, we need to identify the periods where flow rate is well controlled (e.g. shut-in) and use the data within those periods for further analysis. However, for machine learning based interpretation, the full pressure-rate data are used to train the machine learning model, and the pressure signal is deconvolved by feeding a constant rate history into the trained model. Because the interpretation is not constrained to those well-controlled periods, neither human intervention nor breakpoint detection is required. The machine learning based interpretation would be even more favorable if we do not have shut-in or constant rate periods in the PDG data at all. In that case, the data set is not interpretable from the conventional well testing perspective.

Denoising: Denoising is not a main focus of machine learning research, but there have been some studies of reducing noise level as a byproduct in the machine learning

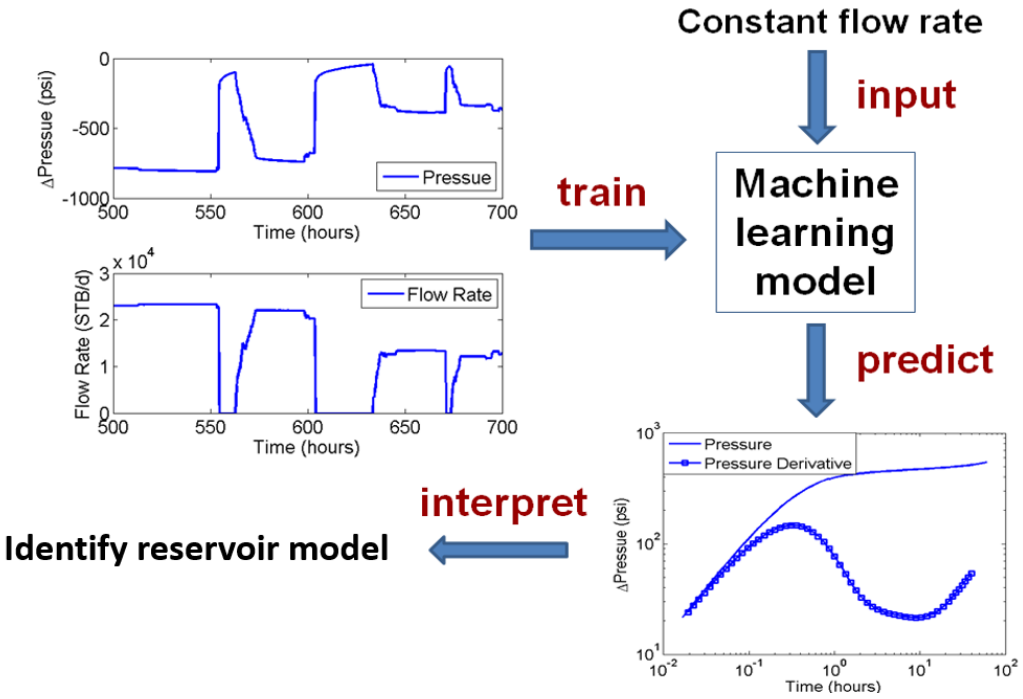


Figure 1.3: Conceptual illustration of deconvolving PDG pressure using machine learning. First the machine learning model is trained on rate-pressure data from PDG to learn the underlying mapping. Next a constant flow rate history is fed into the trained model, and the predicted pressure is plotted on log-log scale together with the pressure derivative. The reservoir model can be identified on the log-log plot with conventional well test interpretation techniques.

process, especially in image denoising. It was also shown by Liu and Horne (2012) that machine learning techniques have a high tolerance of noise in PDG data. The reason for us to apply machine learning for denoising is that noise and the underlying reservoir responses have different patterns, which can be distinguished by machine learning. By putting in a clean constant rate for deconvolution, we are able to remove the noise by only utilizing the reservoir response patterns learned by the machine. In other words, denoising is no longer needed as a separate data processing step, but integrated as part of the deconvolution. That also avoids the tricky threshold selection required for wavelet based denoising (Athichanagorn 1999), and further automates the interpretation process.

Scalability: In traditional well testing, the reservoir model is learned by tuning physical parameters (e.g. permeability, wellbore storage) to match the observed pressure. The tuning is usually implemented by performing nonlinear regression. In machine learning based well testing, those physical parameters are integrated into some mathematical coefficients and the nonlinearity is handled when constructing the features as a function of flow rate and time. Thus reservoir model identification is transformed to the linear regression on those mathematical coefficients, which connect the feature with pressure (see mathematical details in Chapter 2). The computational cost of linear regression on those mathematical coefficients is much cheaper than solving the physical parameters iteratively via nonlinear regression (also see Chapter 2). The inexpensive computation provides the potential for machine learning to work on large volumes of PDG data.

Temperature analysis: As we discussed earlier, temperature data are strongly correlated with pressure and flow rate. Similar to the pressure signal, temperature is

also one kind of reservoir response and contains reservoir information implicitly. We would expect machine learning to help extract the patterns between temperature-rate-pressure and recover the reservoir information. The extracted patterns can be particularly useful given that temperature measurements are easier to obtain compared with rate. It can be an interesting research direction to model the missing pressure or rate history using the available temperature data.

In sum, the inherent characteristics of machine learning make it a very promising tool for PDG data interpretation. There have been earlier successes in applying machine learning techniques to process the PDG data (Liu and Horne, 2012, 2013a, 2013b). However, previous methods have a number of limitations. Particularly, the missing interpretation of early transient behaviors (e.g. wellbore storage, skin) and high computation cost limit the application of such methods on real PDG data. Up to now, the best practice of machine learning based PDG data interpretation is still an open question, and further research is needed in this area.

In this study, we aim to develop the machine learning approaches suitable to extract the patterns behind PDG data, and to explore the engineering applications where the learned patterns can be utilized in a meaningful way. Some examples of the applications are as follows:

- Deconvolution: as discussed earlier, deconvolution extracts the long term drawdown pressure from multirate PDG data, and reveals richer information about the reservoir than conventional well testing analysis on individual shut-in periods. The idea of deconvolution by machine learning is illustrated in Figure 1.3. Detailed case studies of machine learning based deconvolution are discussed in Chapter 5 of this dissertation.

- Flow rate reconstruction: normally flow rate measurements are considered to be less reliable compared to pressure measurements, and it is not uncommon to observe missing data in flow rate measurements. Thus it is useful to train the machine learning algorithms to learn the mapping from pressure to flow rate, then predict the flow rate history based only on the more reliable pressure data. An example of missing flow rate reconstruction on a real set of data is shown in Figure 1.4. Additional case studies are discussed in Section 3.1.2 and Section 6.1.

- Well condition monitoring: the machine learning algorithms can be also trained to learn the patterns behind wellhead pressure, downhole pressure and surface flow rate. Figure 1.5 shows an example of modeling downhole pressure based on wellhead pressure. The learned patterns can be then used as a well alert by comparing the predicted downhole pressure (for instance) with the downhole pressure measured in real time, to identify abnormal well conditions. A field example is discussed in Section 6.2 of this dissertation.

1.2 Literature Review

Historically one focus of well testing research is to develop analytical solutions (mainly for pressure). Those works still continue nowadays, with interests shifted towards to more complex problems, including non-Newtonian fluids (Machado et al., 2018; Kamal et al., 2016), fractured horizontal wells (He et al., 2017; Chen et al., 2016), multiphase flow (Kamal et al., 2015), temperature transient solutions (Palabiyik et al., 2016), etc.

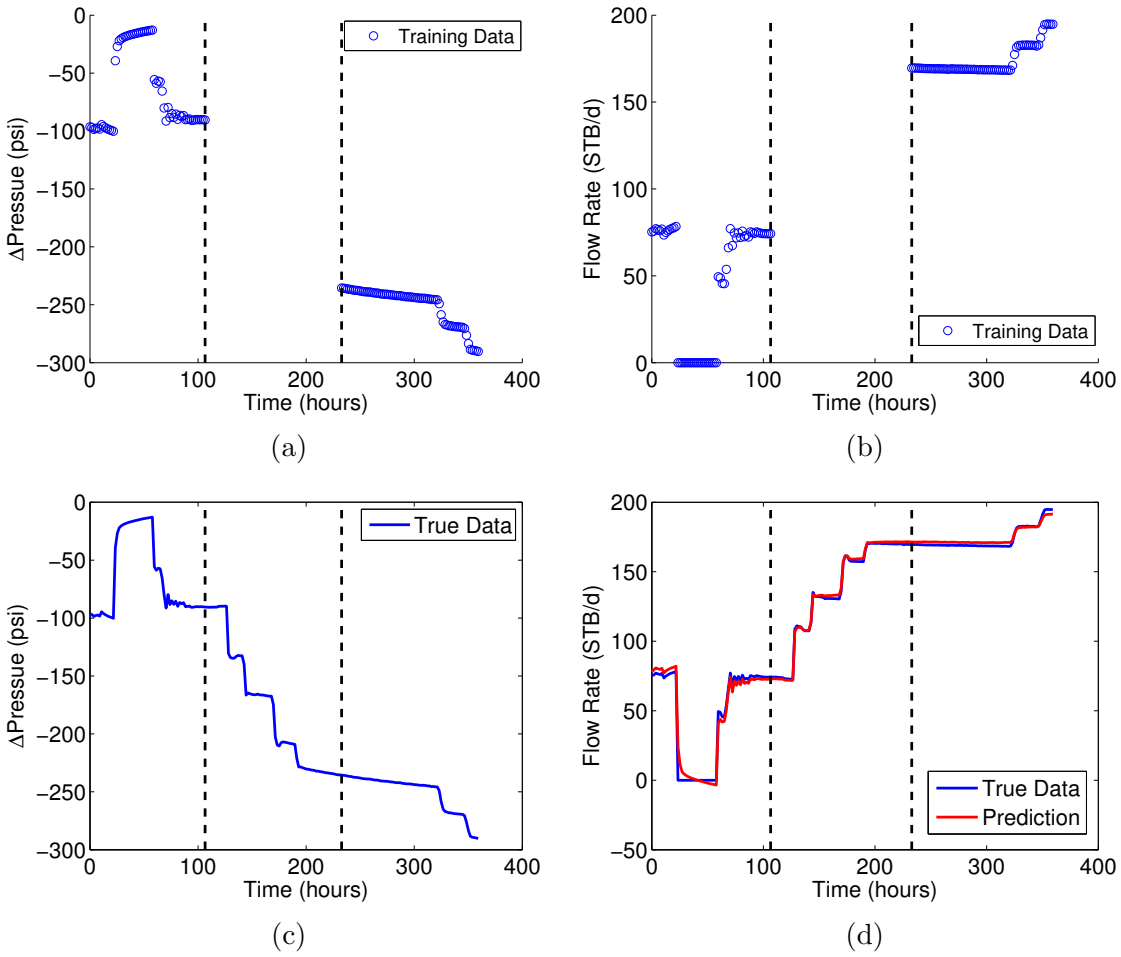


Figure 1.4: Reconstruction of the missing flow rate in between the two dashed lines. As shown in (a) and (b), the information in between the two dashed lines was hidden when the algorithm was trained to learn the mapping from pressure to flow rate. Next the complete pressure history in (c) was fed to the trained algorithm, and the reconstructed flow rate (red line in (d)) showed high agreement with the actual measured flow rate (blue line in (d)).

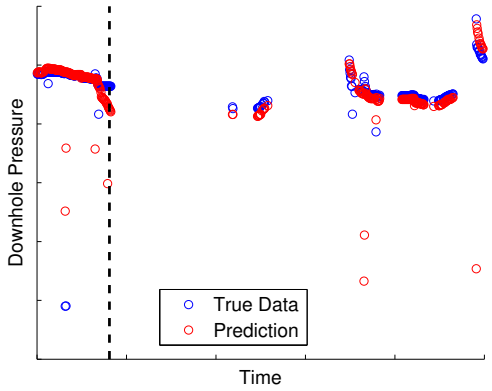


Figure 1.5: Well condition monitoring: the algorithm was trained to learn the mapping from wellhead pressure to downhole pressure on the data in the first $\frac{1}{5}$ of time (left of the dashed line). The algorithm was then able to predict the downhole pressure (red) accurately for the rest of the time.

The deployment of PDGs together with data-driven analytics lead to a new direction of well testing research. To better analyze the PDG data, a variety of approaches have been developed by researchers. In this section, we review those previous work on PDG data interpretation, and the current stage of the applications of machine learning and deep learning for reservoir characterization.

1.2.1 PDG Data Processing

PDG data contain different levels of complexities: outliers, noise, missing measurements, etc. Thus before the actual interpretation, current methodologies need first to preprocess the PDG data to make them easier to handle. Here the studies on two key processing steps, denoising and breakpoint identification, will be reviewed.

Denoising

In general, noise is associated with any type of measurement, especially measurements with high frequencies. That is also true with PDG measurements. As a powerful tool in signal processing, wavelet analysis has been applied by several researchers to denoise the PDG data. Athichanagorn (1999) developed a hybrid thresholding function based on the concepts of hard thresholding and soft thresholding proposed by Donoho and Johnstone (1994). Athichanagorn (1999) showed that the hybrid thresholding preserved the benefit of hard thresholding in preserving signal sharpness and soft thresholding in smoothing noisy features. However, the hybrid threshold cannot be applied directly in most cases and the selection of an appropriate threshold is a trial-and-error process.

To estimate the noise level for better denoising, Kin (2001) suggested a least square error method, which calculated the noise level by fitting a straight line to the pressure trend. This method assumes pressure varies linearly with time over the interval where noise level is estimated. To overcome that linearity assumption, Ouyang and Kikani (2002) proposed a polytope method, which can be applied to nonlinear relationships. It was shown that the polytope method improved the accuracy of noise level estimation, but the estimation is still uncertain.

Breakpoint Identification

One important characteristic differentiating PDG data interpretation from conventional well testing is that PDG flow rate is continuously varying. In conventional well testing, the flow rate is carefully controlled (either constant rate for drawdown test or zero rate for buildup test). The simple flow rate leads to clean formulas of pressure transient solutions, which can be used easily for reservoir diagnostics and

have been applied widely in conventional well test interpretation. In order to transfer our knowledge from conventional well testing to PDG data analysis, we need to break the multitransient PDG data into separate single transients. The process of detecting the points to separate multiple transients is called breakpoint identification.

Wavelet transformation provides a way to identify the breakpoints, as discussed by Athichanagorn (1999), and Ouyang and Kikani (2002). Wavelet analysis indicates breakpoints as the neighborhoods of singularities, which can be detected using a transient threshold in the format of pressure slope. Although the wavelet approach is shown to be successful in many case studies, it relies heavily on tuning the values of the transient threshold, which is strongly case-dependent. A slightly different threshold may lead to a huge difference in breakpoint identification, and a small inaccuracy in breakpoint identification may result in a totally different reservoir model interpretation (Nomura, 2006). To avoid the inaccuracies associated with threshold selection, some researchers proposed nonwavelet based methods, such as the smoothing filter and the segmentation method suggested by Rai (2005). But such methods may still miss or add some breakpoints, although most apparent breakpoints were detected successfully.

1.2.2 Deconvolution

After a series of data processing, we are ready to move on to the actual data interpretation. Deconvolution is one of the key components in PDG data interpretation, and it was also a main focus of this study. Deconvolution is a term to describe the process of extracting the constant rate pressure response from the convolved multirate pressure signal. As shown in Equation 1.1, the pressure of multitransient $\Delta p(t)$ can

be expressed as the convolution of single transient pressure Δp_0:

$$\Delta p(t) = \int_0^t q'(\tau)\Delta p_0(t-\tau)d\tau \tag{1.1}$$

Hence, the target of deconvolution is to extract the constant rate solution Δp_0 from the variable rate measurement $\Delta p(t)$. The difficulty with deconvolution is that it is a "desmoothing" process which is subject to pathological mathematical instability (Horne, 2007). A variety of efforts have been made to address the issues with deconvolution.

von Schroeter et al. (2004) made a series of studies by formulating the deconvolution into a separable nonlinear Total Least Squares problem. Compared with previous research, their method does not require explicit sign constraints, and it also considers uncertainties in both pressure and rate data. Based on the work of von Schroeter et al. (2004), Nomura (2006) developed an effective multivariate smoothing algorithm to address the deconvolution issue. The main difference between methods used by von Schroeter et al. (2004) and Nomura (2006) is in the selection of regularization parameter and smoothing parameter. Based on the cross validation technique, Nomura's parameter selection (2006) is less subjective and shows better performance.

A bottleneck of those deconvolution techniques is the strong assumption that breakpoints are known exactly a *priori*. However, this assumption is often invalid for actual PDG data and may result in the failure of interpretation.

1.2.3 Flow Rate Reconstruction

Incomplete flow rate history is commonly observed in PDG measurements. For those early PDGs where only pressure is measured, it would be very interesting if we could

back-calculate flow rate as a function of pressure, and estimate a rate history to provide us more insights about the operation scheme.

Nomura (2006) treated flow rate as a variable in his smoothing algorithm, which allows uncertainty in both flow rate and pressure measurements. The algorithm was tested on a synthetic data set where the flow rate was 10% variant from the truth. After running the algorithm until convergence, Nomura (2006) obtained a rate profile very close to the true flow rate. Although it looks promising, such a method requires a good initial guess of the rate profile for the algorithm to converge. In other words, the measured flow rate should be similar to the true flow rate. However, if part of the rate history is completely missing, the measured rate (which is zero) can be very different from the true unknown rate. In that case, Nomura's (2006) method would fail to produce an accurate rate estimation.

Ouyang and Sawiris (2003) proposed a theoretical development of calculating in-situ flow rate based on permanent downhole pressure, where they assumed the flow of either single-phase or well-mixed multiphase fluid along the wellbore is steady-state. They performed a field study applying such a calculation to an offshore well, with a short and simple production profile. The estimated rate profile was compared with a PLT survey and showed a high consistency. However the assumptions and simplifications in their derivation limits its application on more complicated multitransient cases.

In a more recent study, Wang and Zheng (2013) applied wavelet transform to estimate the unknown rate history based on downhole pressure data. The Haar wavelet was used to process the pressure data, and the rate was calculated given the change of rate was proportional to the amplitude of pressure transforms. The

approach was shown effective on several case studies, however, the tuning of wavelet parameters was again strongly case-dependent.

The challenge of reconstructing flow rate based on pressure data is that physically flow rate is the cause of pressure response, not the outcome. Estimating flow rate using pressure is not a physical forward modeling but a mathematical reverse modeling. That is why we propose to apply machine learning here, given machine learning as a powerful tool to find the pattern between variables.

1.2.4 Temperature Analysis

Downhole temperature measurements have been available since the initial usage of PDGs. The strong correlation in downhole pressure-rate-temperature, and the abundance of temperature measurements make temperature a valuable source for reservoir analysis.

Duru and Horne (2010) developed a reservoir temperature-transient model as a function of formation parameters, fluid properties, and changes in flow rate and pressure. A semianalytical solution technique known as operator splitting was used to solve the model. They showed that by matching the model to different temperature-transient histories obtained from PDGs, reservoir parameters such as average porosity, near-well permeabilities, saturation, and thermal properties of the fluid and formation could be estimated. In a later study, Duru and Horne (2011) applied Bayesian inversion to jointly interpret all three PDG measurements, namely pressure, temperature and flow rate. The flow rate was first estimated based on the temperature data, and then was used to extract the pressure kernel in the pressure deconvolution problem.

Ribeiro and Horne (2013) investigated temperature analysis with a specific focus

on the hydraulic fracturing problem. A comprehensive numerical model was developed to calculate the temperature in addition to the pressure response during the fracturing. The temperature analysis improved the knowledge about the hydraulic fracturing process by providing detailed descriptions on the flow dynamics in the fractures and the reservoir. In addition, it was shown that temperature analysis can help to obtain a more accurate estimation of fracture length and reservoir permeability.

Although there has been some good progress in temperature analysis, the studies in this area are immature in comparison to the widely applied pressure analysis approach. As a general tool for pattern recognition, we expect an approach developed for machine learning based pressure interpretation could be also extended to the analysis of temperature.

1.2.5 Multiwell Testing

In the previous sections, we have been focused on the studies of a single active well. This is well suited for the analysis of the data from exploration well tests. However, as the field starts being developed, we would expect more than one well to be active and the behaviors of those wells would interact with each other. This is particularly true for PDG measurements, because PDGs are installed permanently and record the well behaviors over the full cycle of reservoir development.

Mathematically, multiwell testing can be considered by superposition. Similar to multirate tests using superposition in time, multiwell tests lead to superposition in space (Earlougher, 1977). The superposition in both time and space makes multiwell test data even more complicated, and makes their interpretation a very challenging research topic.

The simplest multiwell testing is the interference test, where one well (the "active" well) is put on injection or production, and another well (the "observation" well) is shut in to monitor the pressure changes caused by the first. The advantage of interference testing is that a greater area of the reservoir can be tested. Besides, both reservoir transmissivity (kh) and storativity ($\phi c_t h$) can be estimated from the pressure response (Horne, 1995, p. 194). The disadvantage is that pressure drops can be very small over distance and may be hidden by other operational variations. To interpret interference tests, often the data are matched to some form of the exponential integral type curve (Ramey, 1981), which is controlled by tuning the parameters of interest.

Deconvolution is another target of multiwell testing. Similar to single-well testing, the approach to deconvolution of multiwell tests is also to transform the multitransient pressure data into a constant rate drawdown suitable for interpretation. Several researchers, including Levitan (2007) and Cumming et al. (2014), investigated the techniques to extend the existing deconvolution algorithms for single-well testing to multiwell testing.

1.2.6 Machine Learning Based PDG Data Interpretation

Recently machine learning has drawn the attention of the petroleum industry, and there have been successes in using machine learning across various applications. Ahmadi et al. (2014) utilized support vector machine to predict the breakthrough time of water coning in fractured reservoirs. Support vector machine was also applied by Dursun et al. (2014) to estimate downhole pressure while drilling to optimize the drilling programs. Sankaran et al. (2017) used regression methods to capture the pattern behind surface rate and downhole pressure. Then the learned pattern was

fed with a virtual shut-in condition to predict the downhole pressure that was further used to estimate the well productivity index.

Among those works, the research by Liu and Horne (2012, 2013a, 2013b) is highly related to this study. Liu and Horne (2012) used the simple kernel and the convolution kernel based data mining approaches (Liu and Horne, 2013a, 2013b) to interpret flow rate and pressure data from PDGs. Both simple kernel method and convolution kernel method can be categorized as supervised learning in the machine learning domain. Compared with the simple kernel method, the convolution kernel method not only better captures the reservoir model, but also overcomes the limitations in prediction to a multivariable flow rate history (Liu and Horne, 2013a). Thus, the convolution kernel method is reviewed here as a representation of machine learning based interpretation techniques.

The flow rate and time data format used by Liu and Horne (2013a) is shown in Figure 1.6. $q_k^{(a)}$ and $t_k^{(a)}$ respectively represent the flow rate and time at point k according to reference point a. This reflects the idea that pressure at point a is a convolution of all the pressure responses corresponding to previous flow rate change events at points k, where $k = 1, ..., a$.

The feature-based methods described by Liu and Horne (2013a) attempted to define the pattern relating the target variable (which is usually the pressure) in terms of the input variables flow rate and time, in terms of a feature matrix $x_k^{(i)}$, for example:

$$x_k^{(i)} = \begin{bmatrix} q_k^{(i)} \\ q_k^{(i)} \log t_k^{(i)} \\ q_k^{(i)} t_k^{(i)} \\ q_k^{(i)} / t_k^{(i)} \end{bmatrix}, k = 1, ..., i, i = 1, ..., n \tag{1.2}$$

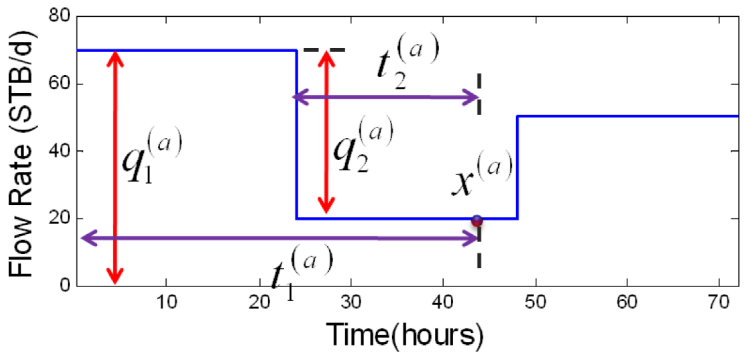


Figure 1.6: Flow rate and time data format in the convolution kernel method (from Liu and Horne, 2013a).

where n is the number of PDG data pairs. The features were developed in this way to describe reservoir behaviors, such as wellbore storage, infinite-acting radial flow and boundary effect. Given two data sets with number of points n_1 and n_2, a linear kernel k was applied onto the features:

$$k(x_k^{(i)}, x_l^{(j)}) =< x_k^{(i)}, x_l^{(j)} >, k = 1, ...i, l = 1, ..., j, i = 1, ...n_1, j = 1, ...n_2 \tag{1.3}$$

Lastly the convolution kernel K summed up all the linear kernels to realize the convolution function:

$$K(x^{(i)}, x^{(j)}) = \sum_{k=1}^{i} \sum_{l=1}^{j} k(x_k^{(i)}, x_l^{(j)}), i = 1, ...n_1, j = 1, ...n_2 \tag{1.4}$$

This typical formulation of features and kernels also frees us from needing to identify the breakpoints, because the formulation treats every data point as a flow rate change event, although it need not be.

Liu and Horne (2013a) denoted pressure data as target y, and used K and y to

train parameter β by solving Equation 1.5 with the conjugate-gradient method:

$$K\beta = y \tag{1.5}$$

where:

$$K = \{K_{ij} = K(x^{(i)}, x^{(j)}), i, j = 1, ..., n\} \tag{1.6}$$

$$\beta = [\beta_1, ..., \beta_n]^T \tag{1.7}$$

$$y = [y^{(1)}, ..., y^{(n)}]^T \tag{1.8}$$

Given a new data set with flow rate and time information, we would be able to construct a new convolution kernel using the new data, and make a prediction of pressure based on the new kernel and trained parameters.

Liu and Horne's method (2013a) was shown to be successful in denoising and deconvolving the pressure signal without explicit breakpoint detection. Figure 1.7 shows a case application of the convolution kernel method, which is denoted by them as Method D in the figure. Figure 1.7(a) shows the training data. Both flow rate and pressure data were added with 3% artificial noise before training. Figure 1.7(b) shows the pressure prediction using flow rate before adding noise as input. The prediction (red) matches the true data (blue) well and the noise was removed successfully. Figure 1.7(c) shows the pressure prediction corresponding to a clean constant rate input. This serves as the deconvolution test. The pressure derivative plot shows that the algorithm is able to capture reservoir behaviors such as radial flow and constant pressure boundary effect. Figure 1.7(d) illustrates the pressure prediction to a different multivariable flow rate history. The pressure prediction is also accurate compared

with the true data.

The work of Liu and Horne (2013a, 2013b) demonstrates the great potential of machine learning in interpreting PDG data. Their approach directly deconvolves the pressure without explicit breakpoint detection, which utilizes the full data set and avoids the interpretation inaccuracy caused by breakpoint identification. In addition, their approach shows a high tolerance of noise. That frees us to design a separate denoising process and further automates the interpretation.

However, the work of Liu and Horne (2013a, 2013b) still has some limitations.

First of all, the convolution kernel algorithm runs very slowly. It was reported that running a case with 600 training data points, which is a modest number, took several hours (Liu, 2013). If we want to apply this method on a one-year data record with millions of points, it would be very difficult. Besides, the long running time also makes it necessary for user manipulation of the data (e.g. subsampling), which could bias the original data (e.g. noise characteristics).

Another issue arises from the data distribution. Liu and Horne (2013a) used hourly distributed data for both training and testing. In their examples, the reservoir went into infinite-acting radial flow at the first hour. The period before the first hour, when the magnitude of pressure change is large, was not included. So Liu and Horne (2013a) were working on "simple" data in terms of data variation. Also we know that the reservoir behavior at the beginning of each transient can matter, and we would expect to see the unit slope and hump shape in the pressure derivative plots in a conventional constant-rate well test. However, this period of information was missing in the examples shown in Liu and Horne′s results. This represents a less stringent test of the method, as some common reservoir behaviors were absent from the example

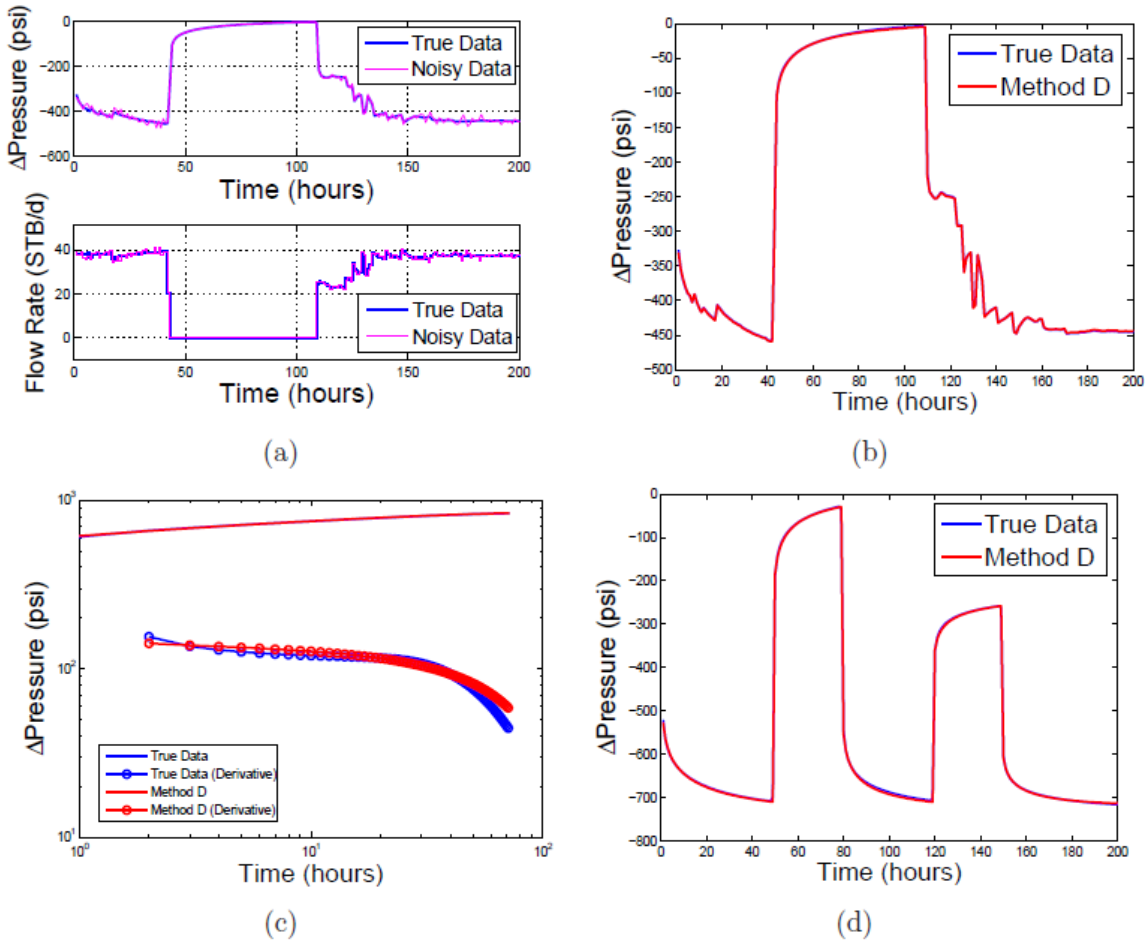


Figure 1.7: Results using the convolution kernel method (from Liu and Horne, 2013a).

data.

1.2.7 Deep Learning for Well Testing

Deep learning, which is also often referred to as deep neural networks, is a particular kind of artificial neural networks (ANNs). ANNs are a computational model based on a collection of connected neurons with simple activation functions, arranged across various layers. ANN has drawn attention from well testing researchers since the 1990s. An early application in well testing was introduced by Al-Kaabi and Lee (1990) to identify the interpretation models from derivative plots. Later on, Ershaghi et al. (1993) proposed an enhanced approach based on the work of Al-Kaabi and Lee (1990), by training multiple ANNs where each neural net was designed to learn the patterns for a specific reservoir model. In 1995, Athichanagorn and Horne combined ANN with sequential predictive probability method to improve the accuracy of interpretation. ANN was used to generate initial parameter estimates of the candidate reservoir models that it identified. The candidate models and initial estimates were then passed to the sequential predictive probability method for final evaluations. A recent work by Spesivtsev et al. (2018) used an ANN with two hidden layers to model the bottomhole pressure based on inputs including wellhead pressure, surface rates and well geometry. The normalized root mean squared error of the predictions was below 5% for most cases, and the ANN model was reported to have certain tolerance on noise.

There are two main limitations of those studies. One comes from the ways that ANN was utilized. Many previous works of ANN-based well test analysis were focused narrowly on learning reservoir models from derivative plots. Some other important

topics of modern well testing, such as PDG data analysis and pattern recognition of flow rate-pressure signals were not explored. Another limitation is the ANN architecture used in those studies, namely multilayer feedforward ANN. A multilayer feedforward ANN is a nonrecurrent neural network, i.e. the response at a given time does not depend on the past computations (no memory). However in well testing, the pressure response clearly depends on the previous rate histories.

Although some of the core concepts in deep learning such as back-propagation date back to the 1980s, it has been only in recent years that deep learning has risen in various research topics and engineering problems. Two popular deep neural networks that have demonstrated their power in our daily life, are recurrent neural networks (RNNs) for natural language processing (e.g. speech recognition) and convolutional neural networks (CNNs) for computer vision (e.g. autonomous driving). One key characteristic of deep learning is that it does not require the laborious feature handcrafting process but learns from the raw data directly. In fact, the information about features are captured by the unique structure of deep neural networks.

Until now, there have been a very limited number of studies of deep learning in reservoir characterization. Aggarwal et al. (2014) used NARX to conduct a simulated well testing, but considered only one synthetic test case with simple model settings. Korjani et al. (2016) studied how deep learning can help estimate petrophysical characteristics by generating synthetic logs of newly drilled wells, and reported that deep learning outperformed kriging for that case. The research by Antonelo et al. (2017) is the only work we are aware of that applies deep learning to model the downhole pressure. Echo State Networks, a particular type of Recurrent Neural Networks, were used to estimate downhole pressure based on topside platform data,

in order to capture the slugging flow behavior. To the best of our knowledge, deep learning for PDG data analysis is so far an almost empty playground for well test researchers.

1.2.8 Limitations of Existing Work

The limitations of the previous work on PDG data analysis are summarized as follows:

- The conventional (not machine learning based) analysis requires intensive human manipulations. To deal with the complexities in data (e.g. noise, breakpoints), the data are usually preprocessed using methods such as wavelet filtering before the actual interpretation. The wavelet thresholds are strongly case-dependent and selecting the thresholds is a trial-and-error process. In addition, the actual interpretation is very sensitive to the quality of PDG data processing. Small inaccuracies in data processing, which are common due to the nontrivial threshold selection, may result in different interpretations. Moreover, most interpretation algorithms require that a reservoir model needs to be defined in advance. That brings even more human intervention into the PDG interpretation process.

- Machine learning based interpretation successfully integrates data processing and interpretation by directly deconvolving the raw data. Although machine learning shows great potential for well test interpretation, the best current methodology (convolution kernel method) has certain bottlenecks that limit its application: the extremely expensive computation and missing interpretation of early transients. To fix those problems, we needed to investigate new

methods to both speed up the computation and to better describe the reservoir responses. Besides, the recent rise of deep learning has demonstrated its power in processing sequential information, yet we could hardly find studies on interpreting PDG data with deep learning techniques.

- As PDGs become more popular in industry, the research interests have expanded from pressure transient analysis to broader areas, including temperature analysis, flow rate reconstruction, multiwell testing, etc. However, research in those areas is still at an early stage. Certain assumptions and simplifications required by the methods are only valid for simple synthetic data. Besides, machine learning has rarely been applied on those topics. As a powerful pattern recognition tool, machine learning is expected to recover additional useful reservoir information from such diverse data sources (e.g. temperature data, multiwell data).

1.3 Research Objectives

From the literature review, we saw several advantages of machine learning compared with the conventional interpretation techniques, as well as the limitations of the best current machine learning method for PDG data analysis (convolution kernel method). Thus the main focus of the this work was to develop fast and reliable machine learning frameworks to analyze the data recorded by PDGs, and to investigate the applications of such frameworks on various engineering problems. The detailed objectives of the proposed work were set to be:

- Investigate machine learning and deep learning techniques that may be suitable

for interpreting flow rate and pressure data from PDGs. Innovative methods are needed to better describe the reservoir behaviors from early-transient until late-transient with efficient computation, while maintaining all the advantages of the convolution kernel method in denoising and deconvolution without breakpoint detection.

- Explore innovative ways of utilizing PDG measurements other than pressure transient analysis. For instance, using pressure data to reconstruct the missing flow rate history, cointerpreting temperature and pressure data when flow rate data are not available, or conducting artificial interference testing by using data collected from multiple wells.

- Study the strength and limitations of the proposed machine learning and deep learning techniques for pressure-rate deconvolution, with a focus on their performance to handle various complexities observed in real data, such as noise and outliers. Compare deconvolution using machine learning and deep learning techniques against the conventional deconvolution methods applied in industry.

1.4 Dissertation Outline

The dissertation is organized as follows:

Chapter 2 introduces the machine learning and deep learning techniques investigated in this study, including linear regression, kernel ridge regression, nonlinear autoregressive exogenous model (NARX), and standard recurrent neural network (RNN). A general comparison on machine learning and deep learning is also discussed in the chapter. Part of the contents in Chapter 2 is from paper SPE-174034-MS (Tian

and Horne, 2015a), SPE-175059-MS (Tian and Horne, 2015b), and SPE-187181-MS (Tian and Horne, 2017).

Chapter 3 shows the applications of the machine learning techniques introduced in Chapter 2 to analyze rate-pressure-temperature data from PDGs. The chapter begins by comparing linear approach machine learning and the convolution kernel method on single-well pressure transient analysis. The chapter also covers the results of multiwell testing, flow rate reconstruction, and temperature data analysis. Part of the contents in this chapter is from paper SPE-174034-MS (Tian and Horne, 2015a) and SPE-175059-MS (Tian and Horne, 2015b).

Chapter 4 illustrates the applications of deep learning techniques introduced in Chapter 2 to analyze the pressure and temperature data of a single well. The results demonstrate the capability of NARX to learn the reservoir models from raw measurements. Hyperparameter tuning for NARX models is also discussed in this chapter. Part of the contents in this chapter is from paper SPE-187181-MS (Tian and Horne, 2017).

Chapter 5 investigates the topic of pressure-rate deconvolution, by comparing the deconvolution results by machine learning and NARX, against two conventional deconvolution methods. The four methods were tested on a series of cases representing practical issues occurred in PDG data deconvolution.

Chapter 6 discusses the applications of machine learning techniques introduced in Chapter 2 on two field examples. The first field example demonstrates the ability of machine learning to reconstruct the missing flow rate from pressure recorded by PDG. The second field example illustrates the application of machine learning on modeling downhole pressure.

Chapter 7 summarizes the findings in this work and proposes some research directions for potential future studies.

Appendix A includes a study of well production history analysis using statistical correlation methodology as a comparison to the machine learning approaches discussed in the main body of the dissertation.

Chapter 2

Methodologies

In this chapter, we introduce the two families of methods utilized in this work, namely, feature-based machine learning and deep learning. Strictly speaking deep learning is a subset of machine learning. However, the great success of recurrent neural networks (RNNs) for natural language processing and convolutional neural networks (CNNs) for computer vision, separates the concept of deep learning from others. In this study, the term "machine learning" refers to the techniques requiring handcrafted features, and the term "deep learning" refers to the techniques that are based on neural networks and can take raw data as inputs, i.e. no feature engineering required. First, three different techniques (linear regression, kernel, model regularization) and feature engineering are introduced in the application of machine learning for single-well pressure transient analysis. The models are subsequently modified to deal with multiwell testing and flow rate reconstruction problems. Next two deep learning techniques (standard RNN, NARX) are discussed in the second section of this chapter.

2.1 Feature-Based Machine Learning

Machine learning arose as a subfield of artificial intelligence in computer science. One particular class of machine learning we will see in this work is supervised learning, where both target y (also called response, outcome) and features x (also called predictors, regressors) are measured, and the goal is to find the pattern behind y and x, then use the trained pattern for further prediction. If the target y is quantitative (e.g. price), the supervised learning problem is further classified as a regression problem; if y is categorical (e.g. spam/legitimate emails), the problem is classified as a classification problem. This study focused on the regression problem.

2.1.1 Single-Well Pressure Transient Analysis

For single-well pressure transient analysis, the data of interest are time, flow rate and pressure recorded by a single PDG. Pressure (more accurately, the pressure difference between the query point and the start point) is labeled as target y, and flow rate and time data are formulated into features x:

$$y^{(i)} = \Delta p^{(i)}, i = 1, ..., n, \tag{2.1}$$

$$x^{(i)} = F(q^{(i)}, t^{(i)}), i = 1, ..., n, \tag{2.2}$$

where $q^{(i)}$ and $t^{(i)}$ are the flow rate and time at the i-th data point, and n is the number of observations from the PDG. The goal of machine learning is to appropriately model the target y using features in x. Thus the formulation of x is important. Here, features

adapted from the work of Liu and Horne (2012, 2013a, 2013b) were applied:

$$x^{(i)} = \begin{bmatrix} 1 \\ \sum_{j=1}^{i}(q^{(j)} - q^{(j-1)}) \\ \sum_{j=1}^{i}(q^{(j)} - q^{(j-1)})\log(t^{(i)} - t^{(j-1)}) \\ \sum_{j=1}^{i}(q^{(j)} - q^{(j-1)})(t^{(i)} - t^{(j-1)}) \\ \sum_{j=1}^{i}(q^{(j)} - q^{(j-1)})/(t^{(i)} - t^{(j-1)}) \end{bmatrix}, i = 1, ..., n. \tag{2.3}$$

This feature formulation is constructed to reflect the possible physical properties of pressure response as a function of flow rate and time. For instance, let us first take a look at $\sum_{j=1}^{i}(q^{(j)} - q^{(j-1)})\log(t^{(i)} - t^{(j-1)})$, the dominant term during infinite-acting radial flow. For a well flowing in infinite-acting radial flow, the pressure change corresponding to a series of i constant step rate changes can be written as (Horne, 1995, p. 52):

$$\frac{\Delta p^{(i)}}{q^{(i)}} = \frac{162.6B\mu}{kh}\left\{\sum_{j=1}^{i}\left[\frac{q^{(j)} - q^{(j-1)}}{q^{(i)}}\log(t^{(i)} - t^{(j-1)})\right] + \log\frac{k}{\phi\mu\ c_t r_w^2} - 3.2275 + 0.8686s\right\}, \tag{2.4}$$

where

B = formation volume factor

c_t = total system compressibility (/psi)

h = thickness (feet)

k = permeability (md)

$q^{(j)}$ = flow step between $t^{(j-1)}$ and $t^{(j)}$ (STB/d)

r_w = wellbore radius (ft)

s = skin factor

$t^{(i)}$ = time at data point number i (hour)

μ = viscosity (cp)

ϕ = porosity

$\Delta p^{(i)}$ = pressure difference between pressure at $t^{(i)}$ and the initial pressure (psi).

$\sum_{j=1}^{i}(q^{(j)} - q^{(j-1)})\log(t^{(i)} - t^{(j-1)})$ is extracted from Equation 2.4 to characterize $\Delta p^{(i)}$ during infinite-acting radial flow. Similarly, $\sum_{j=1}^{i}(q^{(j)} - q^{(j-1)})$ represents pressure signal as a superposition of previous flow rate change events. $\sum_{j=1}^{i}(q^{(j)} - q^{(j-1)})(t^{(i)} - t^{(j-1)})$ is used to describe wellbore storage and reservoir boundary effect. The last term $\sum_{j=1}^{i}(q^{(j)} - q^{(j-1)})/(t^{(i)} - t^{(j-1)})$ does not have a clear physical meaning, but it was reported by Liu and Horne (2013a) that adding this term improves the learning performance. The $\frac{1}{t}$ functionality provides a "fade out" feature to cause early time behaviors to be deemphasized as time goes forward. The first term 1 serves as the intercept.

With these handcrafted features, we investigated three machine learning techniques, namely linear regression, kernel method, and model regularization.

Linear Regression

We first investigated linear regression because of its effectiveness, simplicity and interpretability. In linear regression, the relation between the feature $x^{(i)}$ and the target $y^{(i)}$ is assumed to be linear:

$$y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)}, i = 1, ..., n, \tag{2.5}$$

where $\epsilon^{(i)}$ is the error term, n is the number of measurements from the PDG, and θ is a $(p+1)$-by-1 (p is the number of features excluding intercept) vector with some unknown constants, also known as parameters or coefficients (Hastie et al., 2009, p. 44). The "best" estimation of θ is defined in terms of minimizing the residual sum of squares (Hastie et al., 2009, p. 44):

$$J(\theta) = \sum_{i=1}^{n} (\theta^T x^{(i)} - y^{(i)})^2. \tag{2.6}$$

The objective $J(\theta)$ in Equation 2.6 is a quadratic function of θ, thus the global minimum can be found by setting the first order derivative of $J(\theta)$ with respect to θ to zero, i.e. $X^T X\theta = X^T y$. Then we obtain the optimal value of θ in closed form (Hastie et al., 2009, p. 45)

$$\theta = (X^T X)^{-1} X^T y, \tag{2.7}$$

where X is an n-by-$(p+1)$ matrix ($p = 4$ given the feature formulation in Equation 2.3)

$$X = \begin{bmatrix} (x^{(1)})^T \\ (x^{(2)})^T \\ \cdots \\ (x^{(n)})^T \end{bmatrix}, \tag{2.8}$$

and y is an n-by-1 vector

$$y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \cdots \\ y^{(n)} \end{bmatrix}. \tag{2.9}$$

Here we assume the matrix $X^T X$ in Equation 2.7 is invertible because has full column rank by construction, i.e. the features in Equation 2.3 are linearly independent. It should be also pointed out that the inversion is only performed on the small $(p+1)$-by-$(p+1)$ matrix $X^T X$, even though the number of data points n can be large. Compared with the iterative solution techniques by Liu and Horne (2012, 2013a, 2013b), this closed form solution of θ can speed up the solution process, and give the algorithm the potential to be applied on large volume PDG data.

These discussions outline the mathematical procedures for training. Given a new test data set, the feature matrix X^{pred} corresponding to the test data can be developed using Equation 2.3 and Equation 2.8. Then the pressure prediction is given by:

$$y^{pred} = X^{pred}\theta. \tag{2.10}$$

Kernel Method

The kernel method can be applied to efficiently map the features x to a higher dimensional space without explicit representation of those features (we will explain what that means soon). The kernel on two features x and z, is defined as the inner product of the corresponding expanded features $\phi(x)$ and $\phi(z)$ (Shawe-Taylor and Cristianini, 2004, p. 34). Namely,

$$K(x, z) = \phi(x)^T \phi(z). \tag{2.11}$$

Map $x^{(i)}$ to $\phi(x^{(i)})$ and map θ to ϑ, where ϑ is given by (Shawe-Taylor and Cristianini, 2004, pp. 241-242)

$$\vartheta = \sum_{j=1}^{n} \beta_j \phi(x^{(j)}). \tag{2.12}$$

Thus:

$$\theta^T x^{(i)} = \vartheta^T \phi(x^{(i)}) = \sum_{j=1}^{n} \beta_j (\phi(x^{(j)}))^T \phi(x^{(i)}) = \sum_{j=1}^{n} \beta_j K(x^{(i)}, x^{(j)}), i = 1, ..., n. \quad (2.13)$$

Here, the form of the kernel $K(x^{(i)}, x^{(j)})$ was chosen as the polynomial kernel with power m in this study: $K(x^{(i)}, x^{(j)}) = ((x^{(i)})^T x^{(j)})^m, i, j = 1, ..., n$. Define the kernel matrix K_M (Shawe-Taylor and Cristianini, 2004, p. 52) such that:

$$K_M(i, j) = K(x^{(i)}, x^{(j)}), i, j = 1, ..., n. \quad (2.14)$$

Then we can further write the objective function as:

$$J(\beta) = \sum_{i=1}^{n} (K_M(i, :)\beta - y^{(i)})^2. \quad (2.15)$$

By using the kernel, we change the regression on $\theta = (\theta_0, \theta_1, \theta_2, ..., \theta_p)^T$ to the regression on $\beta = (\beta_1, \beta_2, ..., \beta_n)^T$, where p is the number of original features and n is the number of observations. Similar to linear regression, the optimal solution can be also obtained in closed form $\beta = K_M^{-1} y$, where the invertibility of the n-by-n matrix K_M is guaranteed given X is of full rank. Given the setting $p < n$ in this work, solving the parameters for kernel method takes more time than linear regression as in Equation 8 (inversion of a larger matrix). However, with $n < 10^5$ for all datasets in this study and in the work by Liu and Horne (2012, 2013a, 2013b), such closed form solution still requires less computational efforts compared to minimizing the objective function with an iterative approach, e.g. gradient descent.

Once β is obtained after training, pressure prediction y^{pred} can be calculated given

x^{pred} calculated from the test data:

$$y^{pred} = \sum_{j=1}^{n} \beta_j K(x^{pred}, x^{(j)}). \tag{2.16}$$

From the discussions above, we see that kernel method only requires a user-specific kernel, i.e. the type of kernel, e.g. polynomial kernel (as used in this research), and kernel parameters, e.g. the power for a polynomial kernel. Calculating $\phi(x)$ explicitly is not needed during either training (Equation 2.15) or testing (Equation 2.16). The kernel helps the algorithm to learn in $\phi(x)$ high-dimensional space without explicit representation of $\phi(x)$. This property allows us to add more features into our model with little cost, thus gives the model more flexibility to capture more detailed reservoir behaviors. Besides, the kernel method maintains the computational efficiency with the closed form solution.

Model Regularization

Model regularization is a technique that shrinks the parameter estimates (in our case, θ or β). Model regularization is widely used to address the overfitting issue by reducing the prediction variance. Generally there are two ways of model regularization: ridge regression and lasso (Hastie et al., 2009, p. 61).

Ridge regression minimizes the cost function as

$$J(\theta) = \sum_{i=1}^{n} (\theta^T x^{(i)} - y^{(i)})^2 + \lambda \sum_{j=1}^{p} \theta_j^2, \tag{2.17}$$

where $\lambda \geq 0$ is called the tuning parameter. Similar to linear regression, ridge regression seeks parameters to fit the data by minimizing $\sum_{i=1}^{n} (\theta^T x^{(i)} - y^{(i)})^2$. However,

the penalty on the L_2 norm of parameters, namely $\lambda \sum_{j=1}^{p} \theta_j^2$ forces the parameters towards zero, leading to decreased model flexibility and less danger of overfitting. λ controls the relative weights on a better fit and smaller parameters. Usually a K-fold (e.g. $K = 10$) cross-validation is used to select an appropriate value of λ (Hastie et al., 2009, p. 69). After choosing a value for λ, parameters θ that minimizes the cost function in Equation 2.17 can be calculated (Hastie et al., 2009, p. 64):

$$\theta = (X^T X + \lambda I)^{-1} X^T y. \tag{2.18}$$

Ridge regression can be also applied along with the kernel method, which is called kernel ridge regression. In kernel ridge regression, the cost function is:

$$J(\beta) = \sum_{i=1}^{n} (\sum_{j=1}^{n} \beta_j K(x^{(i)}, x^{(j)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \beta_j^2. \tag{2.19}$$

The quadratic objective function in Equation 2.19 is minimized by setting its first order derivative with respect to β to zero (Shawe-Taylor and Cristianini, 2004, p. 233):

$$\beta = (K_M + \lambda I)^{-1} y. \tag{2.20}$$

Ridge regression uses the penalty to force the parameters towards but not equal to zero. Lasso is a technique that can force some of the parameters exactly to zero, thus excluding redundant features in the model by assigning zero parameters for these features. Lasso changes the penalty term from L_2 norm to L_1 norm of the parameters (Hastie et al., 2009, p. 68). The cost functions of lasso and lasso used along with

kernel are as follows:

$$J(\theta) = \sum_{i=1}^{n}(\theta^T x^{(i)} - y^{(i)})^2 + \lambda \sum_{j=1}^{p} |\theta_j|, \tag{2.21}$$

$$J(\beta) = \sum_{i=1}^{n}(\sum_{j=1}^{n} \beta_j K(x^{(i)}, x^{(j)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} |\beta_j|. \tag{2.22}$$

Unlike ridge regression, lasso does not have a closed form solution for the parameters, but some technical computing programs, such as MATLAB and R, provide useful toolboxes to solve the lasso problem.

Comparison of Linear Regression and the Convolution Kernel Method

Here we have a mathematical comparison of linear regression and the convolution kernel method, to show that the two methods actually learn in the same feature space.

Define the polynomial kernel as:

$$K(x^{(i)}, x^{(j)}) = ((x^{(i)})^T x^{(j)})^m = \phi(x^{(i)})^T \phi(x^{(j)}), i, = 1, ..., n_1, j = 1, ..., n_2, \tag{2.23}$$

where $x^{(i)}$ is given by Equation 2.3, and m is the power of polynomial kernel. It is easy to show that the polynomial kernel with $m = 1$ is equivalent to linear regression:

$$K(x^{(i)}, x^{(j)}) = ((x^{(i)})^T x^{(j)})^1 = \phi(x^{(i)})^T \phi(x^{(j)}), i, = 1, ..., n_1, j = 1, ..., n_2, \tag{2.24}$$

$$\phi(x^{(i)}) = x^{(i)}, \phi(x^{(j)}) = x^{(j)}, i, = 1, ..., n_1, j = 1, ..., n_2. \tag{2.25}$$

Equation 2.24 and Equation 2.25 show that the power one polynomial kernel does

not expand the feature space. The algorithm still learns on feature space given by $x^{(i)}$. So in order to prove linear regression is equivalent to convolution kernel, we will show that power one polynomial kernel is equivalent to convolution kernel.

$x^{(i)}$ in Equation 2.3 (excluding the intercept term) can be also represented as $x^{(i)} = \sum_{k=1}^{i} z_k$, where z_k is the p-by-1 vector defined in Equation 1.2 (Liu and Horne, 2013a). Here the intercept is neglected for the proof because we were not able to find in literature how the intercept was taken into account in the convolution kernel method (Liu and Horne, 2013b). With $x^{(i)} = \sum_{k=1}^{i} z_k$, the matrix of power one polynomial kernel K_M^p and the matrix of convolution kernel K_M^c can be written as follows:

$$K_M^p(i,j) = \Big\langle \sum_{k=1}^{i} z_k, \sum_{l=1}^{j} z_l \Big\rangle, \tag{2.26}$$

$$K_M^c(i,j) = \sum_{k=1}^{i} \sum_{l=1}^{j} \big\langle z_k, z_l \big\rangle. \tag{2.27}$$

For all real vectors $u, v, w \in \mathbb{R}^p$, the inner product has the following properties:

Linearity in the first argument: $\big\langle u+v, w \big\rangle = \big\langle u, w \big\rangle + \big\langle v, w \big\rangle$;

Symmetry: $\big\langle u, v \big\rangle = \big\langle v, u \big\rangle$.

z_k is a real vector because the flow rate and time data are real numbers. By applying the properties above, we obtain:

$$\Big\langle \sum_{k=1}^{i} z_k, \sum_{l=1}^{j} z_l \Big\rangle = \sum_{k=1}^{i} \Big\langle z_k, \sum_{l=1}^{j} z_l \Big\rangle = \sum_{k=1}^{i} \Big\langle \sum_{l=1}^{j} z_l, z_k \Big\rangle = \sum_{k=1}^{i} \sum_{l=1}^{j} \big\langle z_l, z_k \big\rangle = \sum_{k=1}^{i} \sum_{l=1}^{j} \big\langle z_k, z_l \big\rangle, \tag{2.28}$$

i.e.

$$K_M^p(i,j) = K_M^c(i,j). \tag{2.29}$$

Thus we prove that the power one polynomial kernel is indeed equivalent to the

convolution kernel.

If the power of the polynomial kernel and convolution kernel is greater than one, however, the two kernels are not equivalent any more. In that case, it can be shown that the polynomial kernel contains more feature interactions than the convolution kernel, and the feature interactions give the polynomial kernel better performance.

Here, we prove that linear regression is equivalent to the convolution kernel method in terms of learning quality. Given the inexpensive solution techniques, linear regression is preferred when processing PDG data.

2.1.2 Flow Rate Reconstruction

The absence of flow rate measurements is commonly seen in modern PDGs, for at least part (and sometimes all) of the time. It would be very helpful if we could reconstruct the missing rates using the available pressure data, to provide a more complete description of the operation. Because our goal is to estimate the flow rate based on pressure data, flow rate is treated as the target and pressure is used as the feature. The challenge here is that the current approaches imply pressure as a function of flow rate (and time) rather than the reverse. To deal with this, we proposed the idea to apply machine learning to establish a reverse relationship from pressure to flow rate. We imposed the features mapping pressure to flow rate based on the features mapping flow rate to pressure in Equation 2.3. The details of mathematical derivations are not covered here, but the principle idea is that flow rate can be expressed using features

in the form of pressure convolution:

$$x^{(i)} = \begin{bmatrix} 1 \\ \sum_{j=1}^{i}(p^{(j)} - p^{(j-1)}) \\ \sum_{j=1}^{i}(p^{(j)} - p^{(j-1)})\log(t^{(i)} - t^{(j-1)}) \\ \sum_{j=1}^{i}(p^{(j)} - p^{(j-1)})(t^{(i)} - t^{(j-1)}) \end{bmatrix}, i = 1, ..., n. \tag{2.30}$$

After defining the features, we may apply linear regression, kernel or kernel ridge regression following the same solving procedure as shown before.

Notice that the features used are similar to the rate to pressure convolution shown in Equation 2.3. During our investigation, we attempted to include features that were inverse to those in Equation 2.3, however this approach was found to be ineffective. The features in Equation 2.30 worked better.

2.1.3 Multiwell Testing

In this section, we describe a machine learning framework to model pressure using flow rate data for multiwell system. Here we have Well w from 1 to N. The pressure of each Well w is affected by the production (or injection) of itself q_w as well as the production (or injection) of the other wells $q_{\tilde{w}}$, where $\tilde{w} = 1, ..., w-1, w+1, ..., N$. Our goal is to model the pressure response of a given well Δp_w using the flow rate data of all the wells in the system (q_w and $q_{\tilde{w}}$).

The main challenge of mutiwell testing is to model the well interactions caused by pressure superposition in space. To address this issue, we applied feature expansion on the basis features developed for single-well pressure transient analysis (Equation 2.3). The expanded features are written as the combinations of the basis features of

all the wells in the system:

$$x^{(i)} = \begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \\ ... \\ x_N^{(i)} \end{bmatrix}, i = 1, ..., n, \tag{2.31}$$

where $x^{(i)}$ represents the expanded features, and $x_w^{(i)}$ represents the basis features of Well $w = 1, ..., N$ (Equation 2.3). $x_w^{(i)}$ describes the effect of the production (or injection) of Well w on the pressure change at the observation well. Thus, we expect the well interactions to be captured by using those expanded features. For a system of N wells, the dimension of the feature space expands from p (feature dimension for single well) to $N \cdot p$. Accordingly, the dimension of coefficients θ also expands to $N \cdot p$. Because of such explicit feature expansion, model regularization is always recommended for solving multiwell system even without applying kernels. It should also be noted that the expanded features are the same for each well.

A distinct machine learning model is trained for each well, by using the expanded features and the pressure of that well. Namely, a different θ_w is trained using $x^{(i)}$ and $y_w^{(i)}$ to minimize the following cost:

$$J(\theta_w) = \sum_{i=1}^{n} (\theta_w^T x^{(i)} - y_w^{(i)})^2, w = 1, ..., N, i = 1, ..., n, \tag{2.32}$$

where θ_w and $y_w^{(i)}$ are the coefficients and pressure of Well w respectively. The intuition is that $x_w^{(i)}$ should have a small p-value if Well w strongly affects the pressure at the observation well. In other words, the well interaction is learned implicitly as we train the coefficients.

With a new set of flow rate histories, a new feature matrix X^{pred} can be developed using Equation 2.8. Then the pressure prediction y_w^{pred} for each Well w is given by:

$$y_w^{pred} = X^{pred}\theta_w. \tag{2.33}$$

2.2 Deep Learning

Deep learning, which is also often called deep neural networks, refers to the artificial neural networks (ANNs) with large number of layers. One classification of ANNs is to divide them into recurrent neural networks (RNNs) and nonrecurrent neural networks, i.e. feedforward neural networks (Goodfellow et al., 2016, p. 168). RNNs utilize the memory (also called feedback) of previous computations to model the response at a given time, while non-RNNs do not. Such a memory characteristic enables RNNs to learn the information contained in sequential signals, and makes RNNs a suitable choice for natural language processing and time series modeling. PDG data can be also viewed as time series, because each measurement is associated with a timestamp. In this study, we explored how RNNs can be applied to interpret data recorded by PDGs.

Before describing the details of RNNs, we first introduce some key concepts using the example of a feedforward neural network to get familiar with the basics of neural network.

2.2.1 Neural Network Basics

Similar to the machine learning techniques introduced before, a feedforward neural network seeks the best mapping of $x^{(i)}$ to $y^{(i)}, i = 1, ..., n$. Here $y^{(i)}$ is known as

output, which has the same meaning as target in the machine learning context. $x^{(i)}$ is known as input, which can be the same as features in machine learning, but does not have to be. In fact, we can impose the raw data of flow rates and time into $x^{(i)}$ without any handcrafting. In between the input and output, there are hidden layers that collect the information from the previous layer and get transformed by some activation functions before the computation being passed to the next layer. Here is a concrete example of a feedforward neural network with one hidden layer, as shown in Figure 2.1.

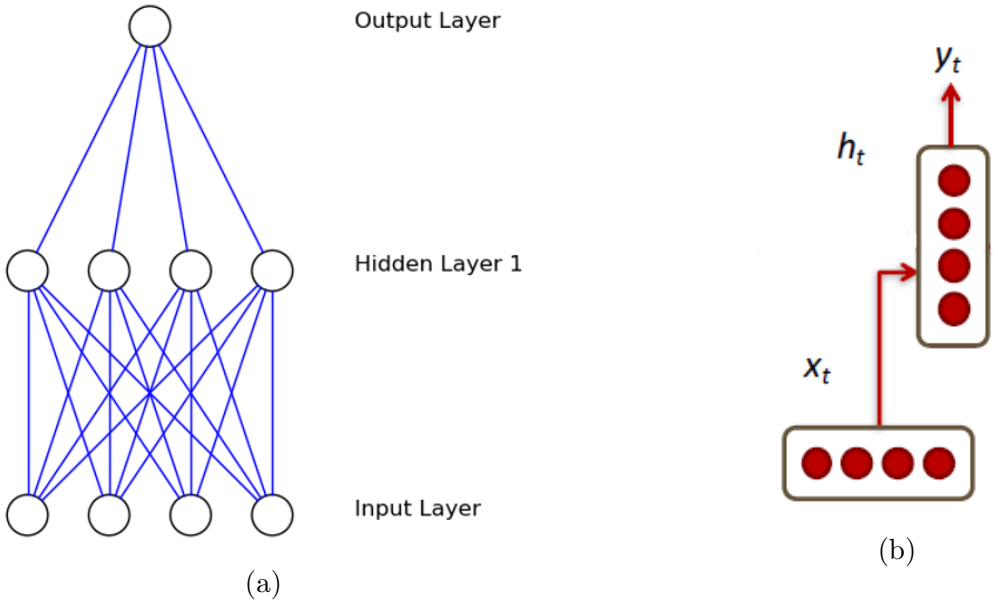


Figure 2.1: A feedforward neural network with one hidden layer. (a) Each circle represents a neuron, and each edge represents an element in the weight matrices. (b) The arrows represent the directions of information flow (adapted from Socher, 2016).

In the left figure, each circle represents a neuron. From bottom to top, the three layers are labeled as input layer $x^{(i)}$, hidden layer $h^{(i)}$ and output layer $\hat{y}^{(i)}$, respectively. Notice the difference between $y^{(i)}$, the target for mapping, and $\hat{y}^{(i)}$, the actual output from the neural network. It is easy to determine the dimension of each layer

from Figure 2.1: $x^{(i)} \in \mathbb{R}^4, h^{(i)} \in \mathbb{R}^4, \hat{y}^{(i)} \in \mathbb{R}$. The mapping from $x^{(i)}$ to $\hat{y}^{(i)}$ can be written as follows:

$$h^{(i)} = f(x^{(i)}W + b_1), \tag{2.34}$$

$$\hat{y}^{(i)} = g(h^{(i)}H + b_2), \tag{2.35}$$

where $W \in \mathbb{R}^{4\times4}, H \in \mathbb{R}^4$ are weight matrices whose elements are represented as the edges in the left figure in Figure 2.1. $b_1 \in \mathbb{R}^4, b_2 \in \mathbb{R}$ are bias terms (same idea as adding intercept to features in machine learning). It is worth mentioning that W, H, b_1, b_2 are the model parameters for the feedforward neural network to learn, and those parameters are shared across all the samples $(x^{(i)}, y^{(i)}), i = 1, .., n$. f, g are activation functions that apply element-wise transforms (mostly nonlinear) to the layers. Some common activation functions include:

sigmoid function: $\sigma(x) = \frac{1}{1+e^{-x}}$;

tanh function: $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$;

ReLU function: $\text{ReLU}(x) = \max(0, x)$.

From Equation 2.34 and Equation 2.35, we see the information flows forward from the input layer to the hidden layer and finally to the output layer. That is why this type of neural network is called feedforward neural network. Another visualization of the neural network defined by Equation 2.34 and Equation 2.35, with explicit representation of information flowing direction by arrows, is shown in Figure 2.1(b). A symbolic expression can be written as:

$$x \to h \to y. \tag{2.36}$$

The objective of the feedforward neural network is to minimize the following cost

function, including the L_2 norm regularization on the weight matrices:

$$J(W, H, b_1, b_2) = \sum_{i=1}^{n} (\hat{y}^{(i)} - y^{(i)})^2 + \lambda \Bigg(\sum_{i',j'} W_{i'j'}^2 + \sum_{i'',j''} H_{i''j''}^2 \Bigg), \tag{2.37}$$

where λ is the tuning parameter as discussed before in the model regularization section. Because of the nonlinearity in the activation functions, the objective function of a neural network is usually nonconvex. Thus a neural network is trained using an iterative optimization algorithm, for instance, gradient descent (Goodfellow et al., 2016, p. 177):

$$\Theta := \Theta - \alpha \nabla J(\Theta), \tag{2.38}$$

where the learning rate α is some hyperparameter for tuning. Θ is a vector concatenating all the model parameters including W, H, b_1, b_2. It is easy to show $\Theta \in \mathbb{R}^{25}$ given the dimension of the four model parameters as $W \in \mathbb{R}^{4\times4}, H \in \mathbb{R}^4, b_1 \in \mathbb{R}^4, b_2 \in \mathbb{R}$. $\nabla J(\Theta)$ is the gradient of the cost function with respect to Θ. One way to obtain the gradient is to take the derivatives of J with respect to each model parameter by applying chain rule repeatedly. An alternative that requires much less effort is by using automatic differentiation implemented in some software libraries, for instance, TensorFlow (Abadi et al., 2016) as used in this study. With TensorFlow, building the forward model as shown in Equation 2.34 and Equation 2.35, leads to automatic gradient calculation by the program.

Gradient descent starts with some initialization of Θ, and performs the update in Equation 2.38 repeatedly until convergence. With nonconvex objective function, the gradient descent algorithm is not guaranteed to converge, thus initialization becomes

very important. One frequently used initialization is called Xavier initialization (Glorot and Bengio, 2010). Given a matrix $W \in \mathbb{R}^{k \times l}$, each element in matrix W is sampled uniformly from $[-\epsilon, \epsilon]$, where:

$$\epsilon = \frac{\sqrt{6}}{\sqrt{k+l}}. \tag{2.39}$$

So far we have introduced the architecture of a feedforward neural network and how it is trained. Next we will move on to recurrent neural networks (RNNs) by examining two RNN models, namely, standard RNN and nonlinear autoregressive exogenous model (NARX).

2.2.2 Recurrent Neural Networks

In a feedforward neural network, the information contained in each input $x^{(i)}$ proceeds through intermediate hidden layers all the way to its corresponding output $y^{(i)}, i = 1, ..., n$. One drawback of this approach is that there is no information flow across different samples i and j, where $i \neq j$ and $i, j = 1, ..., n$. Consider the example of using rates and time as inputs and pressures as outputs, i.e. $x^{(i)} = [q^{(i)}, t^{(i)}], y^{(i)} = \Delta p^{(i)}$. If we were to model the mapping from $x^{(i)}$ to $y^{(i)}$ with a feedforward neural network, the inherent assumption is the pressure change at timestamp $t^{(i)}$ is only dependent on the flow rate at the same timestamp. However, we know from physics that the pressure change at a given time should be modeled by a convolution function of all the previous flow rate changes. Thus a feedforward neural network can easily fail for that problem.

In RNNs, the memory of previous computations is fed back into the model. Such characteristic makes RNNs a suitable choice for modeling sequential signals, such

as rate-pressure data. To illustrate what the feedback in RNNs means, we start by examining a standard RNN model with one hidden layer, as shown in Figure 2.2.

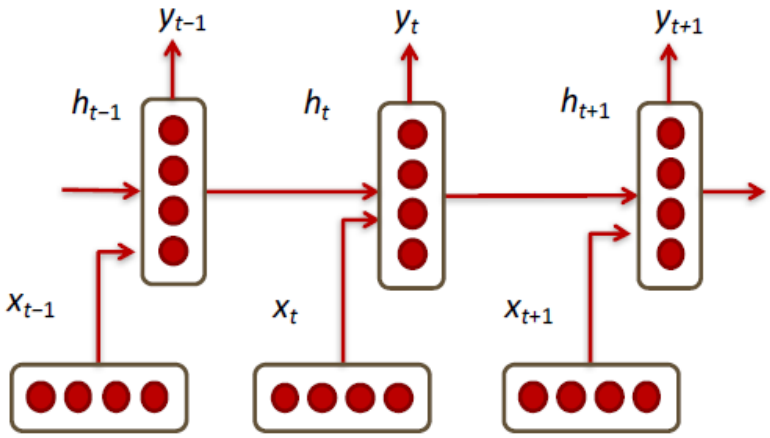


Figure 2.2: A standard RNN model with one hidden layer (adapted from Socher, 2016). h_{t-1} is fed into the hidden layer of the next sample h_t, together with x_t.

Figure 2.2 shows the structure of a standard RNN on three adjacent samples. Similar to the feedforward neural network in Figure 2.1, the RNN also has an input layer with four neurons, a hidden layer with four neurons and an output layer. A significant difference, however, is that the hidden layers of two adjacent samples are connected, which allows the memory of previous computations being fed into the hidden layer together with the current input. By doing so, the standard RNN seeks to capture the relationship between $y^{(i)}$ and all the inputs up to time $t^{(i)}$, i.e. $x^{(j)}, j = 1, ..., i$. That is exactly what we need for modeling pressure based on rate inputs. Here is the mathematical formula describing the RNN shown in Figure 2.2:

$$h^{(i)} = f_1(x^{(i)}W + h^{(i-1)}H_1 + b_1), \tag{2.40}$$

$$\hat{y}^{(i)} = f_2(h^{(i)}H_2 + b_2), \tag{2.41}$$

where W, H_1 and H_2 are weight matrices. The only difference compared to feedforward neural network is the additional term in Equation 2.40 $h^{(i-1)}H_1$, representing the memory from the hidden layer of previous time. The self-pointing arrow on h in the following equation captures such a characteristic:

$$\begin{array}{c} \circlearrowright \\ x \to h \to y. \end{array} \tag{2.42}$$

Similar to feedforward neural network, we define the objective function of the standard RNN shown in Figure 2.2 by combining the residual sum of squares and L_2 norm of the weights:

$$J(W, H_1, H_2, b_1, b_2) = \sum_{i=1}^{n}(\hat{y}^{(i)} - y^{(i)})^2 + \lambda\left(\sum_{i',j'} W_{i'j'}^2 + \sum_{k=1}^{2}\sum_{i'',j''} H_{ki''j''}^2\right). \tag{2.43}$$

Then gradient descent can be applied to minimize the objective function.

One issue that occurs sometimes when training RNNs is the vanishing (or exploding) gradient (Goodfellow et al., 2016, p. 403). The problem is caused by multiplying the same weight matrix over and over when calculating the gradient. For instance, to obtain the gradient of $h^{(n)}$ with respect to weight matrix H_1 in $h^{(2)} = f_1(x^{(2)}W + h^{(1)}H_1 + b_1)$, we first need to calculate:

$$\frac{\partial h^{(n)}}{\partial h^{(2)}} = \frac{\partial h^{(n)}}{\partial h^{(n-1)}}\frac{\partial h^{(n-1)}}{\partial h^{(n-2)}}\cdots\frac{\partial h^{(3)}}{\partial h^{(2)}}. \tag{2.44}$$

By applying the chain rule based on Equation 2.40, it is easy to see that each of the

$(n-2)$ partial derivatives on the right-hand side of Equation 2.49 contains H_1 along with some other terms, i.e. H_1 is multiplied by itself $(n-2)$ times. If the elements in H_1 are close to zeros, the gradient vanishes. In contrast, the gradient explodes if the elements in H_1 are large. One way to deal with the vanishing (or exploding) gradient is to initialize the weights appropriately, for instance, using Xavier initialization. Other solutions include more advanced RNN architectures such as the Long Short-Term Memory (LSTM) and the Gated Recurrent Unit (GRU).

2.2.3 NARX

NARX refers to nonlinear autoregressive exogenous model (Siegelmann et al., 1997). Unlike standard RNN feeding the previous hidden layer $h^{(i-1)}$ into $h^{(i)}$, NARX uses delays from inputs and outputs to model $h^{(i)}$. In a NARX model, the output $y^{(i)}$ at a given time $t^{(i)}$ can be written as a function of the input at that time $x^{(i)}$ plus some delay (memory) terms of previous time:

$$y^{(i)} = F(x^{(i-d_x)}, .., x^{(i-1)}, x^{(i)}, y^{(i-d_y)}, ..., y^{(i-1)}). \tag{2.45}$$

The delays may come from both inputs and outputs, and their periods of delay can be different. In Equation 2.45, the delays of inputs start from $i - d_x$, and the delays of outputs start from $i - d_y$.

When the function F can be approximate by an artificial neural network, the NARX model results in a NARX recurrent neural network, which is also often named as NARX for short. Now let us consider a NARX with one hidden layer,

$$h^{(i)} = f_1(x^{(i-d_x)}W_{d_x} + ... + x^{(i-1)}W_1 + x^{(i)}W_0 + y^{(i-d_y)}U_{d_y} + ... + y^{(i-1)}U_1 + b_1), \tag{2.46}$$

$$y^{(i)} = f_2(h^{(i)}H + b_2), \tag{2.47}$$

where $h^{(i)}$ is the hidden layer neurons at that given time. f_1 and f_2 are activation functions (e.g. tanh). $W_i, i = 0, ..., d_x$, $U_j, j = 1, ..., d_y$ and H are weight matrices. b_1 and b_2 are bias terms. The calculations in Equations 2.46 and 2.47 implement the following: they take linear combinations of input and output delays, add a bias term and apply activation function f_1 to obtain hidden layer neurons $h^{(i)}$. Next the linear combinations of $h^{(i)}$ are added by bias b_2, and then mapped to outputs by function f_2. A high level symbolic expression of those computations is:

$$x \to h \Leftrightarrow y, \tag{2.48}$$

where the one-way arrow represents the mapping from inputs to hidden layers, and the two-way arrow represents the mapping from hidden layers to outputs, as well as feeding the output delays back into the hidden layers. A graphical representation of a NARX model with one hidden layer is shown in Figure 2.3.

The cost function of the NARX model in Equation 2.49 can be optimized using gradient descent, as we did before with the feedforward neural network and standard RNN

$$\begin{aligned} J(W_0, ..., W_{d_x}, U_1, ..., U_{d_y}, H, b_1, b_2) = &\sum_{i=1}^{n} (\hat{y}^{(i)} - y^{(i)})^2 + \\ &\lambda \Bigg(\sum_{k=0}^{d_x} \sum_{i',j'} W^2_{ki'j'} + \sum_{l=1}^{d_y} \sum_{i'',j''} U^2_{li''j''} + \sum_{i''',j'''} H^2_{i'''j'''} \Bigg). \end{aligned} \tag{2.49}$$

Another interesting point we found in this study is that we can actually link NARX

and RNN together to make a comparison between the two under certain assumptions. Let us first assume a NARX and a standard RNN, both with one hidden layer, and further assume the choice of tanh function for f_1 and linear function for f_2 (we will see later that the neural network parameters chosen after tuning in our study satisfy those assumptions). Thus NARX follows:

$$h^{(i)} = \tanh(x^{(i-d_x)}W_{d_x}+...+x^{(i-1)}W_1+x^{(i)}W_0+y^{(i-d_y)}U_{d_y}+...+y^{(i-1)}U_1+b_1), \quad (2.50)$$

$$y^{(i)} = h^{(i)}H + b_2. \quad (2.51)$$

Substituting Equation 2.51 into Equation 2.50, we have:

$$\begin{aligned} h^{(i)} = \tanh[&x^{(i-d_x)}W_{d_x} + ... + x^{(i-1)}W_1 + x^{(i)}W_0+ \\ &(h^{(i-d_y)}H + b_2)U_{d_y} + ... + (h^{(i-1)}H + b_2)U_1 + b_1]. \end{aligned} \quad (2.52)$$

Rearrange the terms to obtain:

$$\begin{aligned} h^{(i)} = \tanh[&x^{(i)}W_0 + h^{(i-1)}HU_1+ \\ &x^{(i-d_x)}W_{d_x} + ... + x^{(i-1)}W_1+ \\ &(h^{(i-d_y)}H + b_2)U_{d_y} + ... + (h^{(i-2)}H + b_2)U_2 + b_2U_1 + b_1], \end{aligned} \quad (2.53)$$

which contains linear combinations of $x^{(i)}$ and $h^{(i-1)}$ as does the standard RNN, plus the inputs and hidden layers with additional delays as well as bias terms. Thus such a NARX model is as least as strong computationally as a standard RNN. A similar argument on more general forms of NARX and RNN can be found in Siegelmann et al. (1997). A detailed comparison of the performance of NARX and RNN is discussed

later a case study in Chapter 3. Because of the stronger computational capability of NARX, it was used as the default deep learning technique for analysis in this study.

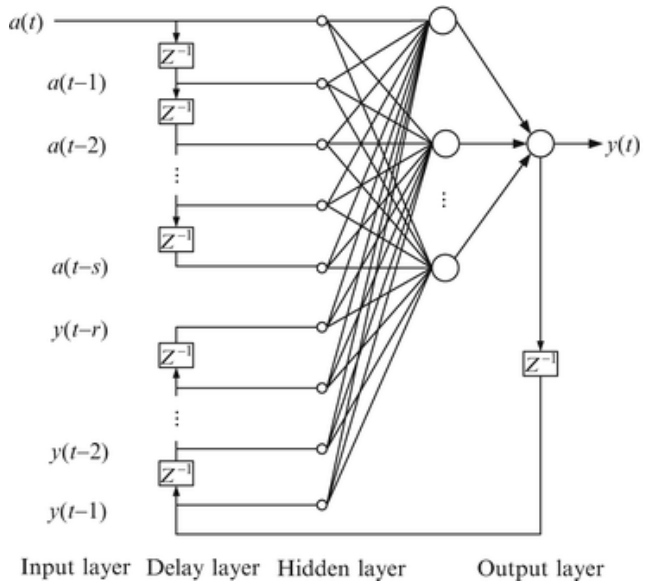


Figure 2.3: A NARX model with one hidden layer (from Li and Lang, 2013). In additional to input a_t, input delays $a_{t-1}, ..., a_{t-s}$ and output delays $y_{t-1}, ..., y_{t-r}$ are fed into the hidden layer h_t, which is further used to obtain the output y_t.

So far, we have discussed how a NARX model (as well as RNN model) learns. Another key step to apply a NARX model is model tuning, which is mostly a trial-and-error process. The details of model tuning are discussed later in Chapter 4. Here we list our choices on various hyperparameters after model tuning, with the example of single-well pressure transient analysis:

$x^{(i)} \in \mathbb{R}^2$, e.g. $x^{(i)} = [q^{(i)}, t^{(i)}]$ (flow rate data as a function of time).

$y^{(i)} \in \mathbb{R}$,, e.g. $y^{(i)} = \Delta p^{(i)}$ (inferred pressure).

$h^{(i)} \in \mathbb{R}^5$ (a hidden layer dimension of three to ten is recommended).

f_1: tanh function.

f_2: linear function.

$d_x = d_y = 2$ (a short delay from one to three is recommended).

There are multiple factors affecting the time required for training of a NARX model, including the number of hidden layers, the number of neurons in each layer, the optimization algorithm, initialization, the amount of training data, etc. Our experiments showed that a NARX with the parameters above can be trained within the order of tens of seconds on datasets with thousands of samples. Such computational efficiency is comparable to the feature-based machine learning techniques discussed previously, and allows NARX to be applied to large volume of PDG data.

To close the chapter, we briefly comment on the comparison of feature-based machine learning and deep learning, with respect to analyzing flow rate, pressure and temperature data recored by PDGs. Both aim at approximating the mapping from input x to output y, but machine learning and deep learning have their own advantages and disadvantages. Feature-based machine learning is extremely reliable when we are able to handcraft the features appropriately (e.g. mapping from rate to pressure), however, that feature handcrafting process can be tedious and sometimes quite challenging (e.g. mapping from temperature to rate). With no requirement for feature handcrafting, deep learning is preferred because its neural network architecture provides additional capabilities to approximate the mapping. A drawback, however, is that the architecture design and hyperparameter tuning for deep learning is mostly trial-and-error. Deep learning also has less interpretability compared to feature-based machine learning. Thus there is no method that always outperforms another, and the applications of feature-based machine learning or deep learning should be carefully

chosen based on the specific problems.

Chapter 3

Feature-Based Machine Learning Results and Analysis

In this chapter, we discuss the application of the feature-based machine learning techniques introduced in Chapter 2 to analyze the PDG data. We begin by discussing the applications of machine learning to interpret the rate-pressure-temperature data from a single well, then we describe the machine learning analysis on multiwell systems.

3.1 Single-Well Data Interpretation

In Section 3.1, we investigate the applications of feature-based machine learning to interpret the data obtained from a single PDG. We first discuss the interpretation of pressure data with a focus on deconvolution, followed by estimating the flow rate based on pressure history. Some potential usages of temperature data are discussed at the end of this section.

3.1.1 Pressure Data Interpretation

Here the discussions of applying feature-based machine learning for single-well pressure data interpretation are focused on comparing the linear approach against the convolution kernel method, to demonstrate its advantage in achieving the same learning quality with much more efficient computations. Machine learning based pressure-rate deconvolution is another key focus of this study, and it will be discussed in more detail later in Chapter 5.

A key contribution of this study is to linearize the machine learning problem of pressure data interpretation, i.e. to replace the convolution kernel method (Liu and Horne, 2013a, 2013b) with linear regression along with the features containing nonlinearity in pressure transient behaviors (Equation 2.3). In Section 2.1.1, it has been shown mathematically that the linear approach and the convolution kernel method learn with the same set of features. Here three case examples were investigated to compare the performance of the two methods from three different angles respectively:

- learning quality on the same type of data used by Liu and Horne (2013a),
- computational efficiency,
- the capability to capture early-transient until late-transient pressure behaviors.

For each case, a workflow suggested by Liu and Horne (2012, 2013a) was implemented as follows:

Training: add 3% uniform noise (uniformly distributed random number in the interval [-0.03, 0.03] multiplied by its original value) to both flow rate and pressure data, and use the noisy flow rate and pressure histories to train the algorithm.

Denoising test: input the original flow rate data (data before adding noise) into

the algorithm and ask for a prediction on pressure, to test whether the algorithm is able to distinguish the noise.

Deconvolution test: input a constant flow rate history (70 STB/d for all synthetic cases here) into the algorithm and ask for a prediction of pressure, to test whether the algorithm is able to deconvolve the pressure signal. The goal of this test is to mimic the actual constant rate control in conventional well testing. The pressure change and pressure derivative on a log-log plot could be used for model identification.

Performance test: input a variable flow rate history into the algorithm and compare the predicted pressure with the one generated by the same reservoir model as training data, to test whether the algorithm works on a new flow rate history that it had never seen before. For all synthetic cases, the rate history is a stepwise function consisting of 60 STB/d for 50 hours, 0 STB/d for 30 hours, 60 STB/d for 40 hours, 20 STB/d for 30 hours and 60 STB/d for 50 hours, consecutively.

Case 1: Comparison of the Linear Approach and the Convolution Kernel Method on Simple Synthetic Data

Both the linear approach and the convolution kernel method were applied on "simple" data starting at one hour, the same as Liu and Horne had in their case examples (2013a). Because a high magnitude of pressure change is shown in early time within each transient, the absence of early time data leads to learning on the pressure with low magnitude of variation, which makes the machine learning task simpler. The "true data" in the four subfigures of Figure 3.1 were generated analytically based on the same reservoir model, which includes wellbore storage, skin effect, infinite-acting radial flow, and closed reservoir boundary. The parameters used to generated the "true data" are summarized in Table 3.1.

Table 3.1: Parameters used to generate the "true data" in Case 1 and Case 3.

Parameter	Value
k (md)	20
S	1
C (STB/psi)	0.001
μ (cp)	2
h (feet)	10
ϕ	0.2
c_t (/psi)	5×10^{-6}
r_w (feet)	0.32
r_e (feet)	600
B	1

The noisy training flow rate and pressure data are shown in Figure 3.1(a). From Figure 3.1(b), we can see that both algorithms have a good reproduction of the pressure history, and they removed the noise successfully with a clean rate history input. Figure 3.1(c) shows the result of the deconvolution test on log-log scale, and the pressure derivative plot indicates radial flow (shown as a flat region), wellbore storage and skin effect (shown as the little hump at the beginning), and closed boundary. The results of the performance test are shown in Figure 3.1(d), where both methods make an accurate prediction on a new rate history. For all the three tests shown in Figure 3.1(a)(b)(c), the results of the linear approach (red) and the convolution kernel method (green) almost overlap on top of each other, validating our conclusion that the two methods have essentially the same learning quality.

Case 2: Interpretation of Simple Real PDG Data

In this case, both flow rate and pressure data for training were obtained from a real PDG. The data include 466587 points recorded during 842 hours of production, with a record frequency around once per 6.5 seconds. Due to the limit of PC memory, the

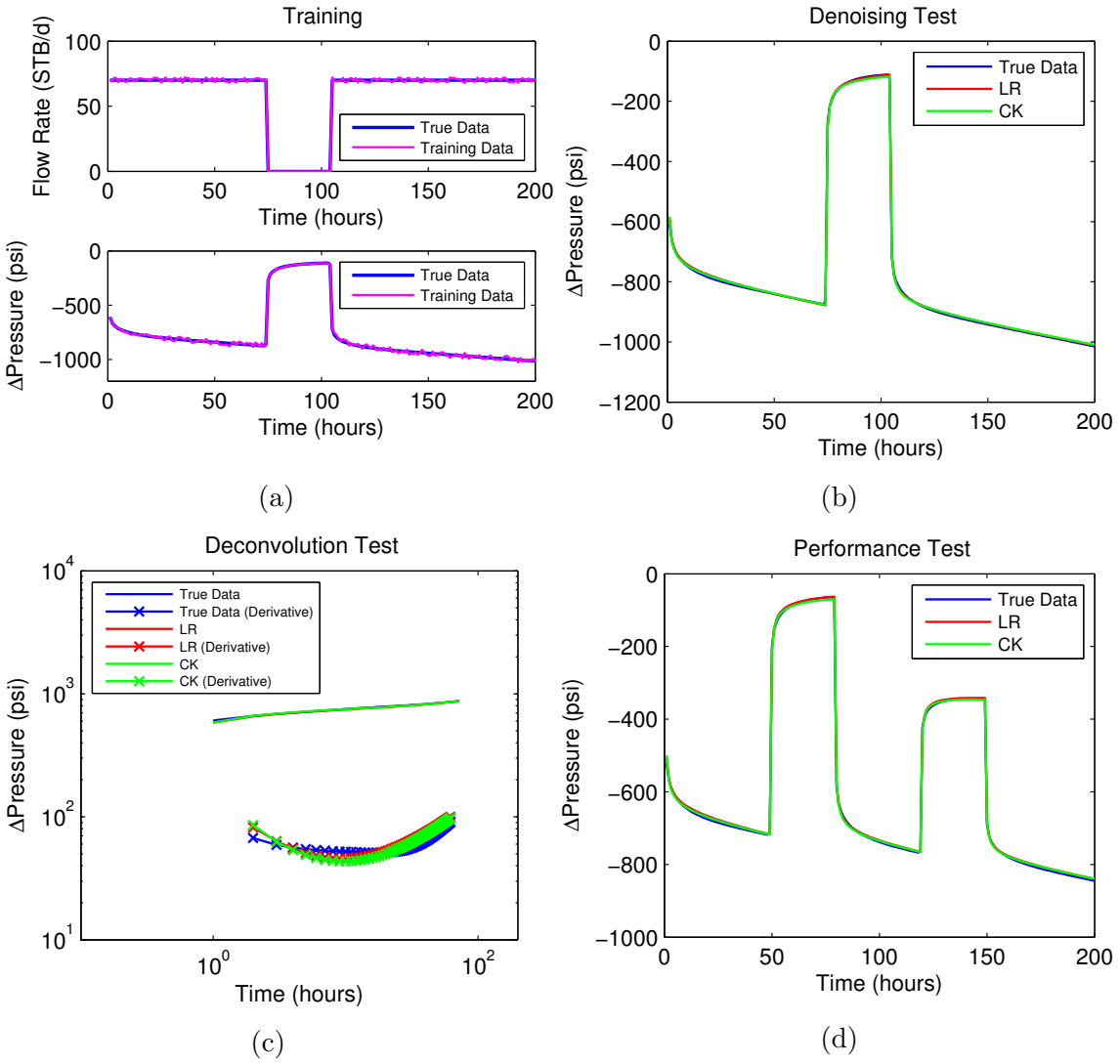


Figure 3.1: Linear approach (LR) and the convolution kernel method (CK) results on Case 1. The overlapping results of the linear approach (red) and the convolution kernel method (green) demonstrate the two methods learn with the same set of features.

data were uniformly sampled down to a size of 4665 as shown in Figure 3.2(a). Unlike the previous case, no artificial noise was added to the data to give the algorithm the original flavor of real PDG measurements.

30% of the data as shown in between the two dashed lines in Figure 3.2(a), were left out during training. That represents the missing data situation sometimes observed in real PDG measurements. The model trained on 70% of the data was then tested using input as that 30% missing rate history. A comparison of the pressure prediction by linear regression (red) against the PDG pressure (blue) is shown in Figure 3.2(b). Although there are some slight deviations from the true data, the linear algorithm is able to describe most of the pressure variations. Root mean squared error (RMSE) defined as $\sqrt{\frac{\sum_{i=1}^{n}(y^{pred(i)}-y^{(i)})^2}{n}}$ for training and testing were 21.67 psi and 23.5 psi, respectively. The pressure predictions to a constant rate input and a multivariate rate input are shown in Figure 3.2(c) and Figure 3.2(d). Because no forward model was used to generate the training pressure data, there is no known true pressure to compare with for both tests. However the reasonable shapes of pressure predictions in Figure 3.2(c) and Figure 3.2(d) still give us confidence on the performance of the linear algorithm. Another thing to point out in this case is the computation time. The training on a data of size 4665 took only 81 seconds on a computer with 2.6GHz CPU and 8GB RAM. Compared with hours of computation needed when using the convolution kernel, the linear approach demonstrated its advantage in terms of computational efficiency in processing large volumes of PDG data.

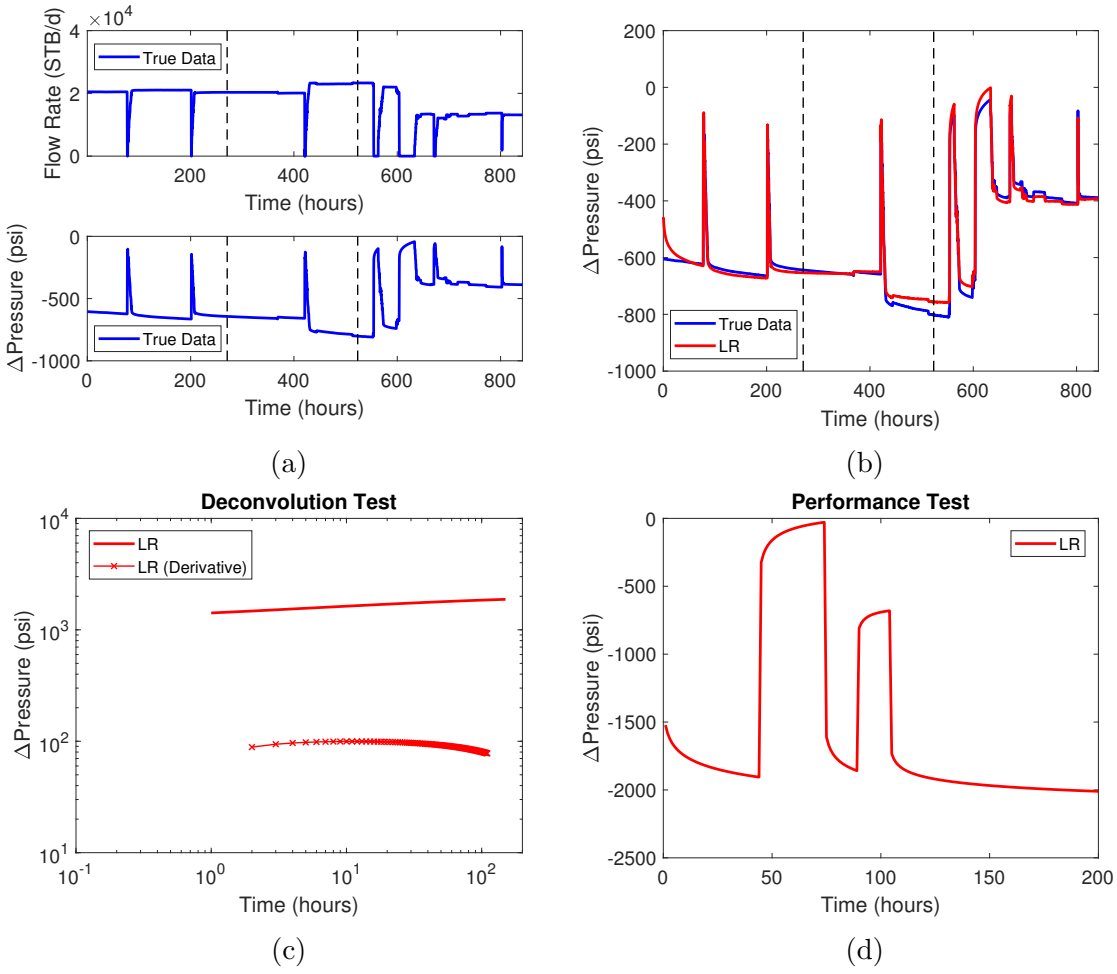


Figure 3.2: Linear approach (LR) results on Case 2. The data in between the two dashed lines in (a) were left out during training, and only used for testing. The computational time for the linear approach (tens of seconds) is much less compared to the convolution kernel method (hours).

Case 3: Comparison of the Linear Approach, the Kernel Method and Kernel Ridge Regression on Strongly Changing Synthetic Data

Here we changed the data distribution in time to cover early-transient until late-transient period in the data, which are referred to as "strongly changing data" in Case 3. $\lambda = 5.43$ was selected after ten-fold cross-validation. The machine learning results using linear regression (red, also represents the convolution kernel method), the kernel method (green) and kernel ridge regression (black) are shown in Figure 3.3.

Figure 3.3(b) shows that ridge regression makes a reproduction of pressure history as accurate as that from the kernel method. This indicates that ridge regression retains the useful expanded features introduced by the kernel method. In the deconvolution test (Figure 3.3(c) on log-log scale), ridge regression shows an almost perfect prediction of the pressure derivative, and captures all the important details in reservoir behaviors. Ridge regression also makes a good prediction in the performance test (shown in Figure 3.3(d)). Table 3.2 provides a quantitative comparison of the performance of the three techniques using the root mean squared error (RMSE) metric. Kernel ridge regression achieves low RMSE across all the tests, validating our previous observations from Figure 3.3.

Table 3.2: RMSE (unit: psi) of Case 3.

	Denoising Test	Deconvolution Test	Performance Test
LR	31.36	37.22	32.17
Kernel	23.87	28.34	118.05
KRR	25.04	25.61	33.72

The results show that kernel ridge regression can be viewed as a balance between linear regression and the kernel method. The linear approach has fewer features,

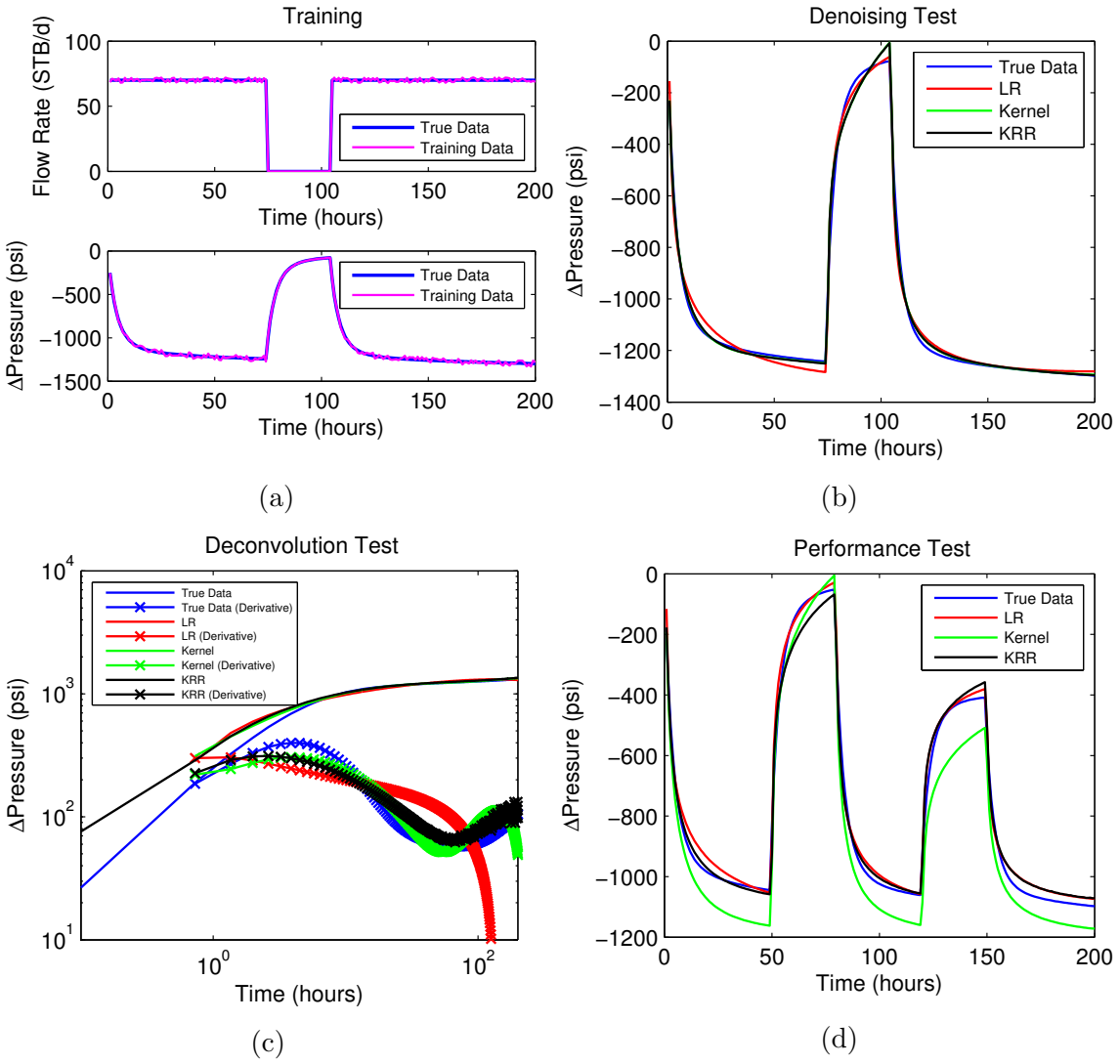


Figure 3.3: Linear approach (LR), the kernel method (Kernel) and kernel ridge regression (KRR) results on Case 3. Kernel ridge regression captures the early-transient until late-transient pressure behaviors more accurately than the linear approach, which is identical to the convolution kernel method.

high bias, and low variance, while the kernel method has more features, low bias, and high variance. Kernel ridge regression trades off the bias in fitting the training data and the variance in predicting the test data by applying feature expansion and model regularization at the same time. Besides, the computation using kernel ridge regression is inexpensive because of the closed form solution shown in Equation 2.20. The tuning parameter is selected using cross-validation and no human intervention is needed for both training and prediction.

In short, the linear approach developed for pressure data interpretation in this study achieves the same learning results as the convolution kernel method, but requires much less computational effort. The linear approach can be also applied along with kernel and model regularization to analyze the pressure data covering early-transient until late-transient period. Because of its effectiveness, the linear approach developed in this study has been adopted by the industry for continuous well surveillance (Sankaran et al., 2017).

3.1.2 Flow Rate Reconstruction

Flow rate reconstruction aims at estimating any missing flow rate history by using available pressure history. This is a very useful capability in practical applications in which individual well rates are not recorded continuously. A set of new features were developed in Section 2.1.2 as functions of pressure to model the flow rate. Coupled with kernel ridge regression, the developed features were tested here on both synthetic and real data sets and demonstrated high prediction accuracy.

Case 4: Flow Rate Reconstruction on Synthetic Data

In this case, we trained the machine learning model using pressure and flow rate

data with artificial noise as shown in Figure 3.4 (a) and (b). It should be noted that the direction of the modeling was different although the training data were still pressure and flow rate. After training, two pressure histories were used as input to generate the flow rate predictions. The comparison of the predictions and the true flow rate that used to generate the input pressure are shown in Figure 3.4 (c) and (d). We observe a high agreement between predictions and true data in both cases, although the requested zig-zag flow rate was very different from the training rate. That demonstrates the ability of our method to generalize well to unknown data.

Case 5: Flow Rate Reconstruction on Real Data

Our machine-learning approach for flow rate reconstruction was tested further on a real PDG data set. In the training step, part of the flow rate data was hidden to mimic the situation of missing measurements (Figure 3.5 (a) and (b)). After training, the complete pressure history was used as input to reconstruct the missing flow rate. Figure 3.5 (d) shows the comparison of reconstructed flow rate (red) and the true measurements (blue). We observe a good agreement between the two. Case 5 illustrates how the developed model can be useful in real life. Usually we have continuous pressure measurements from the PDG, but parts of the flow rate measurement are missing or may be unreliable. In that case, as long as we have at least one period of consistent pressure and flow rate measurements to learn the pattern between the two, we are able to apply the flow rate reconstruction technique to create an estimate of the missing flow rates.

In short, the new set of features developed to model flow rate using pressure measurements, showed promising performance on both synthetic and real data sets. The results also indicate another property of machine learning: the flexibility in the

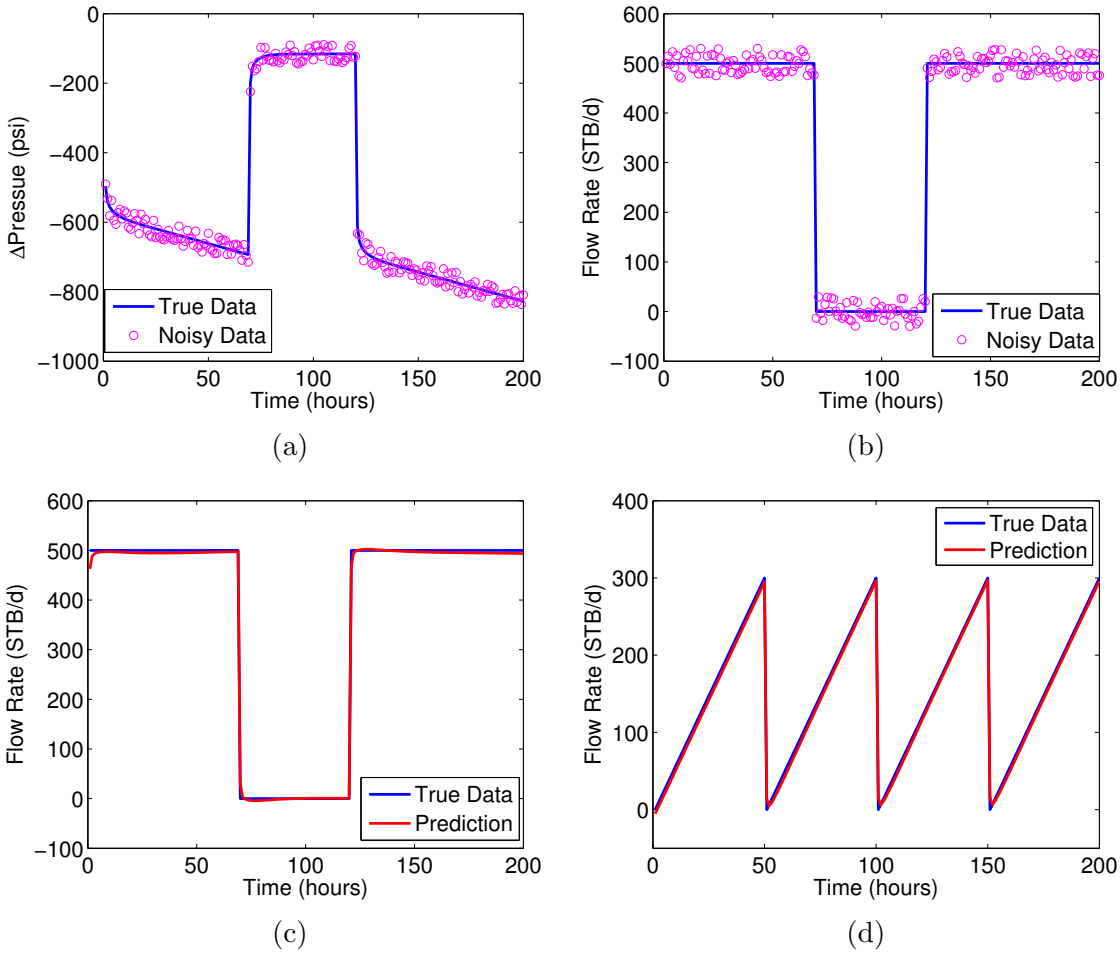


Figure 3.4: Machine learning results on Case 4. (a) and (b) show the noisy training pressure and flow rate data respectively; (c) shows the flow rate reconstruction corresponding to stepwise pressure in (a); (d) shows the flow rate reconstruction corresponding to zigzag pressure.

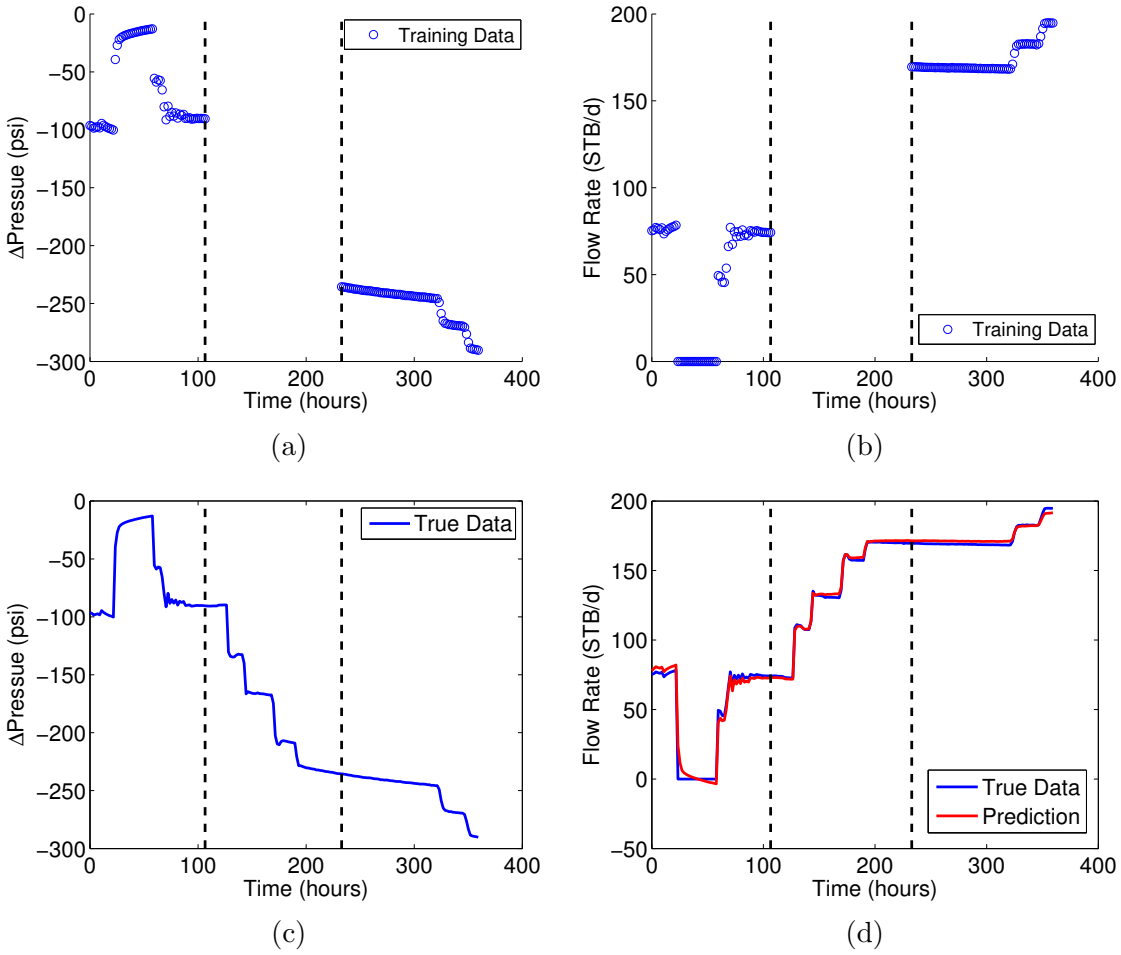


Figure 3.5: Machine learning results on Case 5. (a) and (b) show the training pressure and flow rate with part of history missing; (c) shows the complete pressure history; (d) shows the reconstructed flow rate using pressure input in (c).

direction of modeling by adapting features and targets.

3.1.3 Temperature Data Interpretation

Some potential uses of temperature data from PDGs are also discussed in this study. Machine learning was shown to be able to model temperature and pressure data recorded by PDGs, even if the actual physical model is complex. This originates from the fact that by using features as an approximation of model characteristics, machine learning does not require perfect knowledge of the physical model. The modeling of pressure using temperature data was extended to two promising applications: pressure history reconstruction using temperature data, and the cointerpretation of temperature and pressure data when flow rate data are not available.

Case 6: Pressure History Reconstruction

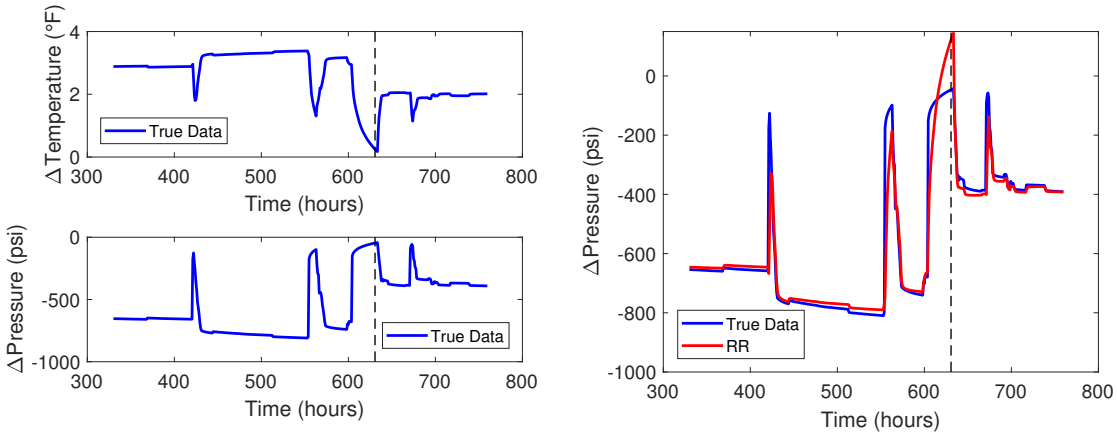
Because machine learning contains the patterns between variables implicitly, the transformation between forward model and inverse model using machine learning is easier compared with conventional ways. To model pressure from temperature data, we use the polynomials of temperature as the features and use pressure as the target:

$$x^{(i)} = \begin{bmatrix} \Delta T^{(i)} \\ {\Delta T^{(i)}}^2 \\ ... \\ {\Delta T^{(i)}}^m \end{bmatrix}, i = 1, ..., n. \tag{3.1}$$

$$y^{(i)} = \Delta p^{(i)}, i = 1, ..., n. \tag{3.2}$$

To test our model, a pair of pressure and temperature data sets were selected from

the same real PDG record used in Case 2. Again, the data was split into training (left of the dashed line) and testing part (right of the dashed line) based on the 70/30 rule as shown in Figure 3.6(a). But this time a temperature history was used as input to obtain a prediction of pressure. The results are shown in Figure 3.6(b). It is clear that the general pressure behaviors are captured by the algorithm. But because temperature and pressure behaviors have their own characteristics, modeling pressure by using purely the polynomials of temperature data has limited accuracy. The prediction of pressure build-up around 620 hours shows a stronger flavor of the temperature variation rather than the pressure variation. The fact that the mismatch around 620 hours falls into the training part contributes to a higher training RMSE of 77.89 psi than testing RMSE of 46.02 psi.



(a) (b)

Figure 3.6: Using ridge regression (RR) to model pressure based on temperature data. Data left of the dashed line were used for training, and the remaining data were used for testing. RMSE for training and testing was 77.89 psi and 46.02 psi, respectively.

This approach of modeling pressure using temperature data may have several applications to reconstruct pressure history. One example is to construct the missing pressure history using a continuous temperature record. Another application is to

model pressure at multiple locations when there are distributed temperature sensing (DTS) devices deployed but only single-point pressure measurements. As long as we have one pair of pressure and temperature data, we can use these data to learn the reservoir model, then ask for predictions of pressure profiles at each location where a temperature measurement is available.

It is worth pointing out here that the actual physical modeling from temperature to pressure is challenging. Because machine learning uses features as an approximation of the complex physical model, we overcome the challenging modeling easily. This shows again the advantage of machine learning in not requiring perfect knowledge of the physical model.

Case 7: Temperature as a Flow Rate Substitute

In this case, we discuss the usage of temperature data as a substitute for flow rate data for pressure-temperature deconvolution. Two data sets were generated using a commercial reservoir simulator. The temperature and pressure data due to a single variate flow rate were augmented with noise and used for training (Figure 3.7(a)). After training, the clean temperature history corresponding to the single variate flow rate (Figure 3.7(a)) and a constant flow rate (Figure 3.7(c)) were used as inputs respectively. Then the pressure predictions were compared with the true pressure generated by the simulator. The comparison results are shown in Figure 3.7(b) and Figure 3.7(d).

In the denoising test, the algorithm removed the noise and gave a modest reproduction of the pressure profile. In the deconvolution test, the algorithm successfully captured the characteristics of pressure derivative plot: wellbore storage, infinite-acting radial flow and closed boundary. The results indicate the potential of using

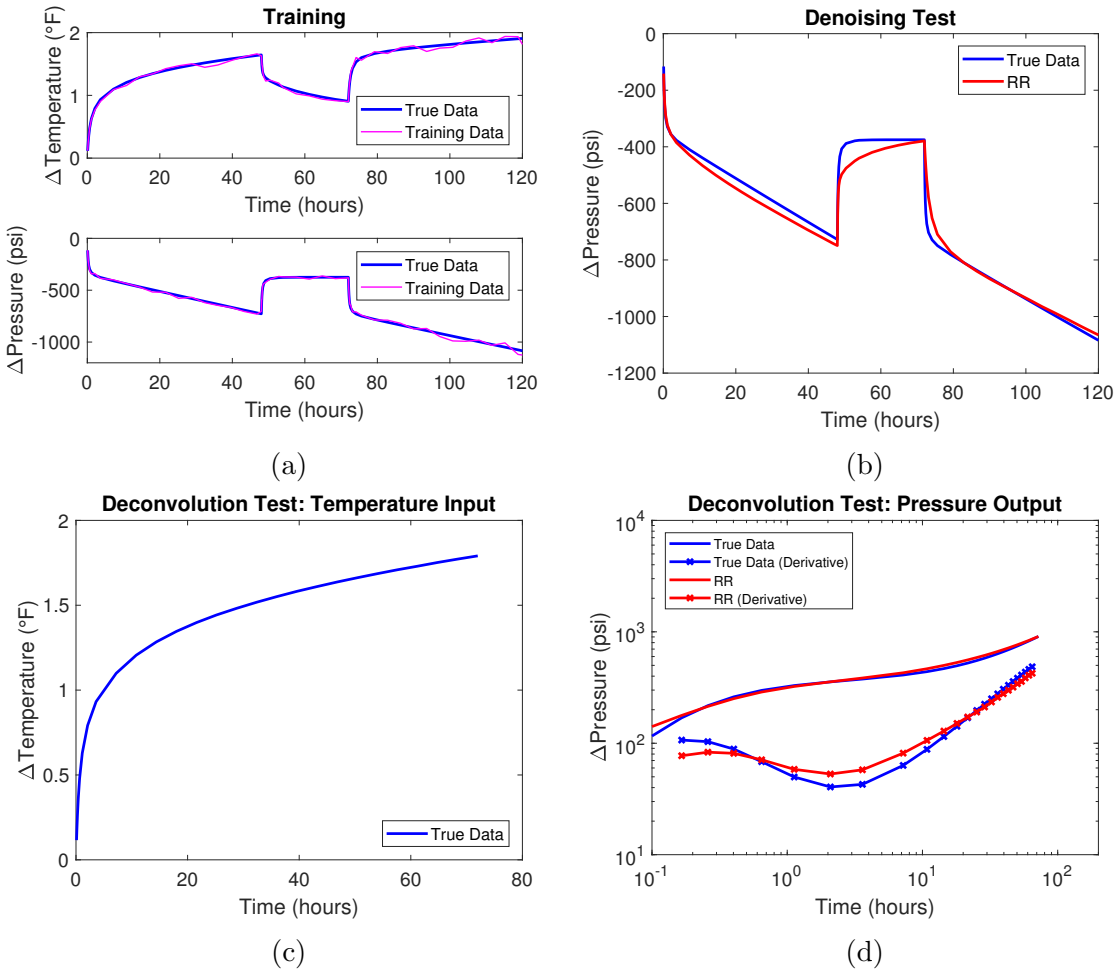


Figure 3.7: Pressure deconvolution using temperature as a flow rate substitute. Ridge regression (RR) was first trained to model pressure based on temperature data. Next a temperature input corresponding to a constant flow rate was fed to the trained model to deconvolve the pressure signal.

temperature data as a substitute for flow rate to be interpreted with pressure. This innovative application is even more promising given that temperature data are usually easier to obtain than flow rate data.

However, there is one limitation: how can we construct the temperature input corresponding to a constant flow rate? To deconvolve the pressure signal, we must use some kind of input containing the constant rate information. This requirement is trivial if the input is flow rate itself, but can be more challenging if the input is temperature. The challenge comes from the fact that constructing temperature response to a constant flow rate requires the knowledge of the reservoir properties, which are unknown before training. Further study would be needed on this topic to realize the full potential of temperaturepressure interpretation.

3.2 Multiwell Testing

In Section 2.1.3, the multiwell testing was formulated into the machine learning algorithm using a feature-coefficient-target model. The features were nonlinear functions of flow rate histories of all the wells. For each well, the features and the pressure target of this well were used to train its coefficients, which implicitly contain the information about the reservoir model and well interactions. The reservoir model can then be revealed by predicting the pressure corresponding to a simple rate history with the trained model. The multiwell machine learning model was demonstrated to be useful in several different ways, including artificial interference testing and reservoir pressure prediction at various well locations based on flow rate data only.

Case 8: Artificial Interference Testing

In this case, we have two wells producing in a homogeneous reservoir. The flow rate and pressure of Well 1 are shown in Figure 3.8 (a) (b), and the flow rate and pressure of Well 2 are shown in Figure 3.8 (c) (d). The pressure data were simulated analytically based on the principle of superposition. The pressure response at each well was designed to be affected by the production of the other well. However, due to the nature of interference, the small magnitude of pressure change caused by the other well is mostly hidden by the dominant pressure change at the well itself. That makes it difficult for the machine-learning algorithm to differentiate the pressure contributions from other wells.

The flow rate and pressure of both wells were augmented with artificial noise before training. After training the model, we created an artificial inference testing by computing the response that would occur if we were to have shut Well 2 while keeping Well 1 producing at constant rate, as shown in Figure 3.8 (e). Namely, Well 1 was used as an active well and Well 2 was used as an observation well. Figure 3.8 (f) shows the comparison of the pressure prediction at Well 2 and the true data. Although the pressure change in the interference test is much smaller in comparison with the training case, the algorithm still captured the trend of this small pressure interference.

The example of interference testing demonstrates the ability of our multiwell machine learning model to learn the well interactions, even though the magnitude of pressure interference is small. The idea of artificial interference testing can also be useful because it does not require additional cost in the field operations. The observation well is shut in the computer program but not in the field. If we have PDG data from multiple wells, an artificial interference test can be implemented easily by

picking one active well and observing the pressure response at other wells.

Case 9: Multiwell Pressure Prediction

Next we added further complexities to the machine-learning task by introducing a greater number of wells and the presence of two-phase flow. Our goal was to predict the pressure given flow rate controls at various well locations. A synthetic homogeneous reservoir model was built with four production wells located at its corners and an injection well in the center. The two producers on the same diagonal shared the same flow rate control, thus there were two different rate profiles for the four producers. Those two rate profiles and the corresponding pressures, together with the flow rate control and pressure response of the injector are shown in Figure 3.9.

As discussed earlier, the key idea in multiwell testing is to expand the feature dimension to include the contributions from multiple wells. Adding more wells only means adding more features. Besides feature expansion, no other treatment was required to account for the additional number of wells. To address the two-phase issue, we replaced the oil rate by total liquid rate when constructing the features. That reflects the physics that the pressure response is caused by the flowing of both phases. A model was trained by using data on the left of the dashed line in Figure 3.9, and it was tested on the remaining part of the data. The results show an accurate match to the training set as well as a good prediction on the test set, although we did observe a deviation in pressure prediction for the injector (Figure 3.9 (f)). Thus we conclude that our machine-learning framework does have the flexibility to work on multiwell systems with two-phase flow.

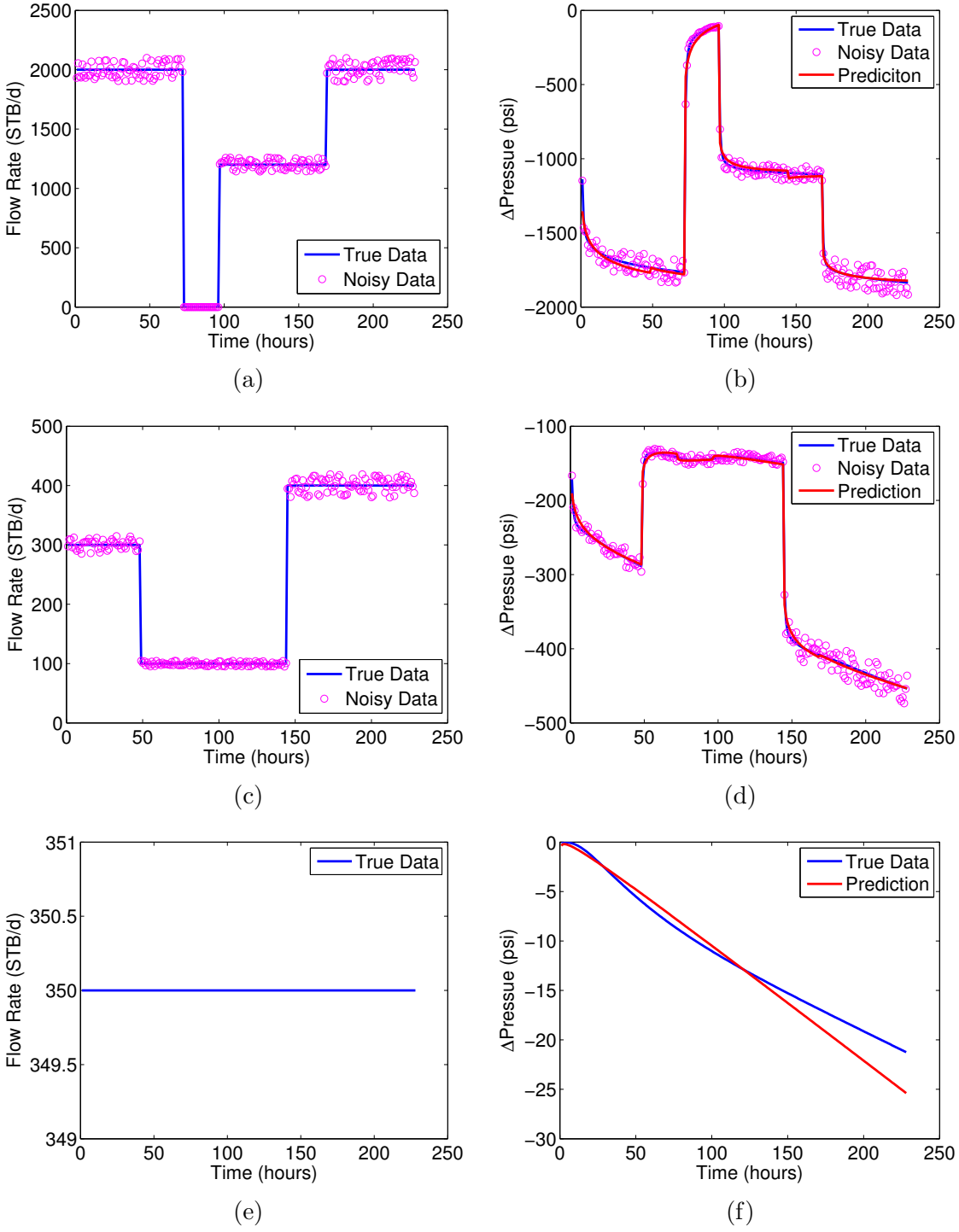


Figure 3.8: Machine learning results on Case 8. (a) and (c) show the true flow rate (blue) and the noisy training rate (circle) of Well 1 and Well 2; (b) and (d) show the true pressure (blue), noisy training pressure (circle) and the pressure prediction (red) corresponding to true flow rate of Well 1 and Well 2; (e) shows the flow rate of Well 1 during the interference test (Well 2 is shut in); (f) shows the comparison of true pressure (blue) and pressure prediction (red) of Well 2 during the interference test.

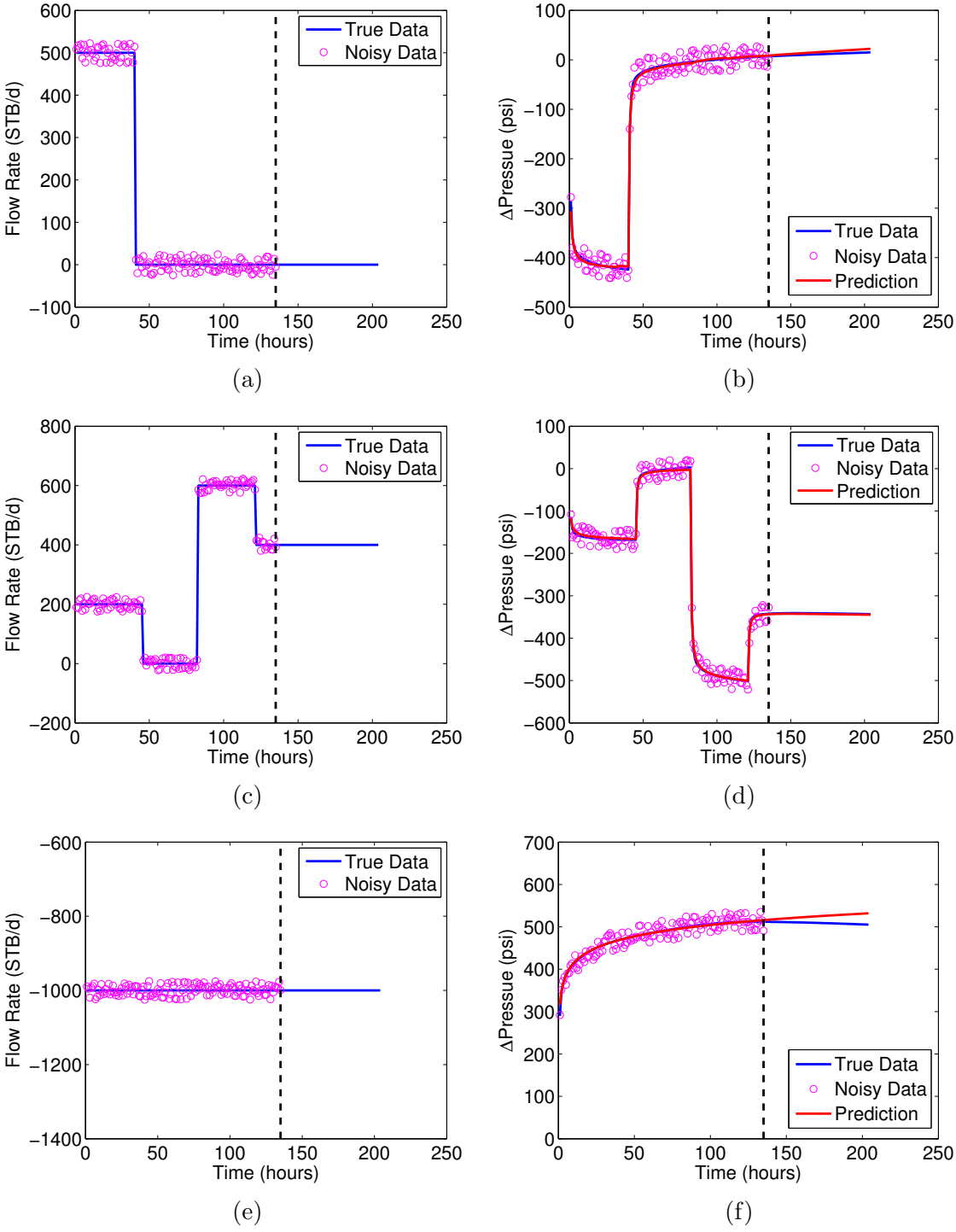


Figure 3.9: Machine learning results on Case 9. (a) and (b) show the flow rate and pressure of Producer 1; (c) and (d) show the flow rate and pressure of Producer 2; (e) and (f) show the flow rate and pressure of the injector. Data to the left of the dashed line were used for training and the data to the left were used for validation.

3.3 Summary

The main conclusions of this chapter are summarized as follows:

- The linear approach machine learning algorithm was shown to have the same learning quality as the convolution kernel method, but with more efficient computations. The linear approach can be also used along with kernel and model regularization to capture early-transient until late-transient pressure behaviors.

- A machine-learning model was developed for flow-rate reconstruction with new definitions of features and targets. The model was tested on both synthetic and real data, and showed promising performance. Machine learning was also utilized to analyze PDG temperature data. It was shown that machine learning can model the pressure signal even with simple polynomial features of temperature data.

- The machine-learning approach for PDG pressure analysis was extended from single-well to multiwell systems. The multiwell model was shown to be able to capture the well interactions accurately, and differentiate the pressure contributions from various wells. The multiwell model was tested in two promising applications: artificial interference test creation without affecting field operation, and pressure prediction for multiwell systems with given rate control.

Chapter 4

Deep Learning Results and Analysis

The recent rise of deep learning was powered by techniques including recurrent neural networks (RNNs), which have been shown useful for processing sequential information. In this chapter, we explored how RNN can be utilized to analyze PDG data for better reservoir characterization and modeling, by examining two specific RNN structures: nonlinear autoregressive exogenous model (NARX) and standard RNN.

RNN is a special type of artificial neural network designed for sequential data processing. Unlike a nonrecurrent neural network, the hidden layers in RNN take memories of previous computations (e.g. hidden layers or outputs computed in previous time) into account. In that way, the information contained in the measurements prior to a certain time can be used to model the response at that time, i.e. the convolutional effects are modeled by the recurrent structure of the RNN. Another favorable property of RNN is that it requires no assumptions on physics in advance.

Both the inputs and outputs come directly from the raw measurements, and no hand-crafted features need to be extracted from the forward physics model. Compared with feature-based machine learning, RNN is more powerful for the modeling where the forward model has not been determined.

In this work, RNN was first tested on a series of synthetic and real flow rate-pressure datasets. The patterns learned by RNN from those data helped correctly identify the reservoir model and forecast the reservoir performance. RNN was also applied on temperature transient data, to demonstrate its advantage over feature-based machine learning with no assumptions on the physics model. The study also showed that RNN has the noise tolerance and computational efficiency that make it a promising candidate to analyze PDG data in practice.

4.1 Hyperparameter Tuning

At the end of the discussions on NARX in Chapter 2, we listed our choices of tuned parameters with the example of single-well pressure transient analysis:

$x^{(i)} \in \mathbb{R}^2$, e.g. $x^{(i)} = [q^{(i)}, t^{(i)}]$ (flow rate data as a function of time).

$y^{(i)} \in \mathbb{R}$,, e.g. $y^{(i)} = \Delta p^{(i)}$ (inferred pressure).

$h^{(i)} \in \mathbb{R}^5$ (a hidden layer dimension of three to ten is recommended).

f_1: tanh function.

f_2: linear function.

$d_x = d_y = 2$ (a short delay from one to three is recommended).

In this section, we discuss in detail about the tuning process of those hyperparameters.

There are multiple hyperparameters that affect the performance of a neural network, for instance, the number of hidden layers, and the choice of activation functions. The goal of hyperparameter tuning is to select the appropriate hyperparameters such that the computational capability of a neural network matches the complexity of the problem that neural network is applied to (Goodfellow et al., 2016, p. 428). In other words, the process of hyperparameter tuning can be also seen as balancing underfitting versus overfitting. If the hyperparameters lead to capability less than the problem complexity, that is underfitting. On the contrary, if the hyperparameters lead to capability more than the problem complexity, that is overfitting. We aim to tune the hyperparameters to prevent either case.

Hyperparameter tuning is a complex optimization problem with various types of variables, for instance, the discrete variable of number of hidden layers, and binary variable of whether to include t in the inputs (Goodfellow et al., 2016, p. 429). It is difficult to formulate an optimization such that all those hyperparameters can be solved at one time. Thus we tuned the hyperparameters manually in this study. Here are the hyperparameters we were focused on:

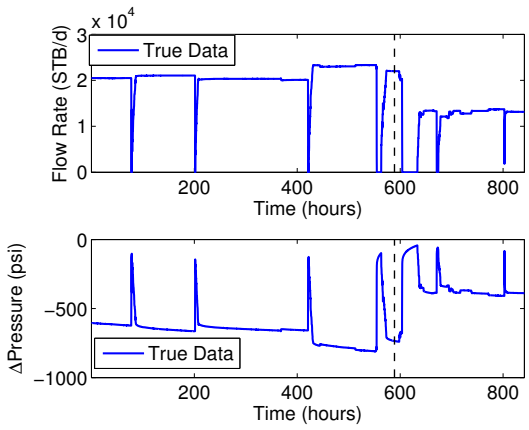
1. Selection of inputs.
2. Number of hidden layers.
3. Number of neurons in each hidden layer.
4. Number of delays.
5. Selection of activation functions.

Inputs affect what data are fed to the neural network. If a key piece of information is missing in the inputs, even the most state-of-the-art neural networks can fail. Hyperparameters two to four essentially affect the dimension of model parameters (weights, bias) of a neural network, thus determine its computational capability. Hyperparameter five, selection of activation functions affects the types of nonlinearity added to the neural network. In this study, we investigated the effects of those five hyperparameters by changing their values one at a time based on the selected optimal values shown here. The results of tuning NARX hyperparameters for single-well pressure data analysis are discussed below.

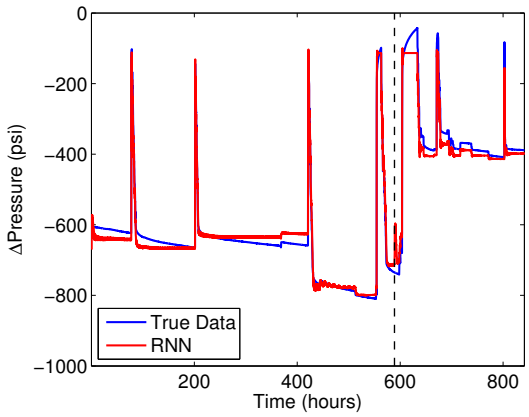
Input

We started by investigating the different choices of inputs $[q]$ and $[q, t]$, namely, whether to feed the time information to the neural network. The data used in this case was the same as the real PDG data in Case 2, Chapter 3. Figure 4.1(a) shows the complete flow rate and pressure histories. The data were divided into two parts as split by the black dashed line. The first 70% of data left of the dashed line were used to train the neural network model and tune the regularization parameter λ, and the remaining 30% of data were used for testing.

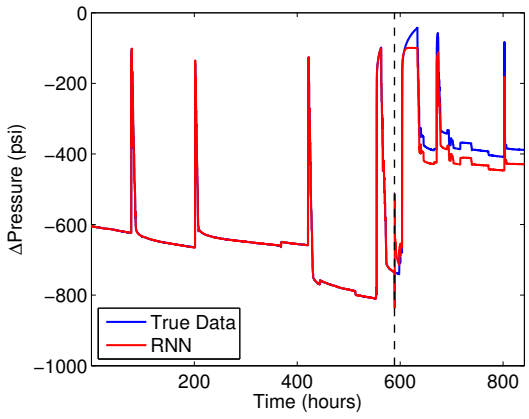
First a standard RNN model with inputs $[q]$ was trained, and its predicted pressure based on the complete rate history is shown in Figure 4.1(b). The pressure prediction has a profile similar to a step function, and it contains a stronger flavor of rate behaviors than pressure behaviors. The RNN model was retrained multiple times, but the step shape remained in the pressure predictions. After t was added to the inputs, however, the pressure characteristics were captured by the RNN (Figure 4.1(c)). That



(a)

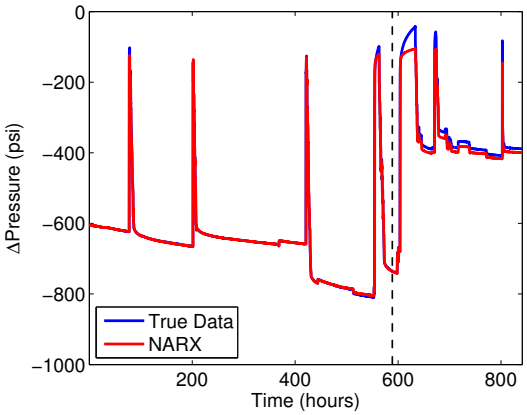


(b)

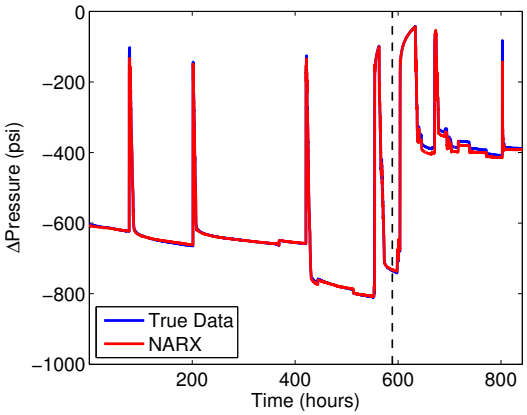


(c)

Figure 4.1: (a) Flow rate and pressure data from a real PDG. (b) Pressure generated by standard RNN using inputs $[q]$. (c) Pressure generated by standard RNN using inputs $[q, t]$. Both RNN models have one hidden layer of five neurons.



(a)



(b)

Figure 4.2: Comparison of pressure generated by NARX models using (a) $[q]$ and (b) $[q, t]$ as inputs. Both models have one hidden layer of five neurons.

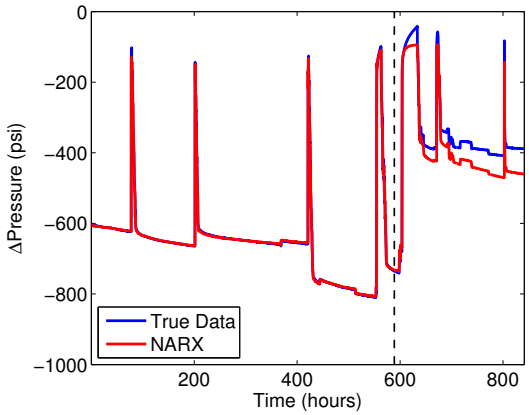
indicates the RNN combined the information from rates and time to approximate the pressure signals, similar to what machine learning did with features as functions of rate and time.

Next $[q]$ and $[q, t]$ were used as inputs for NARX. The same training-testing process was implemented and the results are shown in Figure 4.2. The pressure characteristics were captured well in both tests, which indicates whether time was included in the inputs was less a problem for NARX than standard RNN. That is because NARX utilizes the delays of pressure, which add the pressure transient features into the model. Nevertheless, the use of inputs $[q, t]$ still slightly outperformed inputs $[q]$ for NARX, especially during the testing period. Therefore $[q, t]$ was finally selected as the inputs in this work.

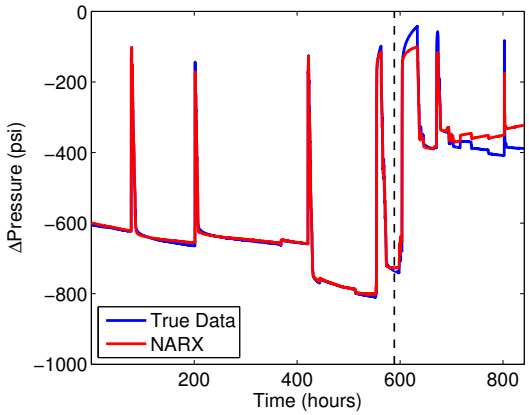
Number of Hidden Layers

The number of hidden layers affect the model capability in two ways. On one hand, an additional hidden layer brings in additional model parameters such as weights and bias associated with that layer, and those additional model parameters strengthen the model capability. On the other hand, an additional hidden layer also brings in additional nonlinearity.

In Figure 4.2(c), we have already seen that one hidden layer with five neurons and tanh activation function worked well. Adding more hidden layers will likely result in overfitting. Here we tested two NARX models, one with two hidden layers (Figure 4.3(a)) and the other with three hidden layers (Figure 4.3(b)). The two hidden layer model had five neurons and ten neurons in each of its hidden layers, and the three hidden layer model has five neurons, five neurons and three neurons



(a)



(b)

Figure 4.3: Comparison of pressure generated by NARX models with (a) two hidden layers and (b) three hidden layers based on flow rate shown in Figure 4.1 (a).

in its hidden layers. The tanh function was applied to all the hidden layers. Both NARX models matched the training data accurately, however, their predictions on the test data obviously deviated from the true data. Moreover, the prediction error increased with time, indicating an even larger discrepancy if the NARX model was used to predict further into the future. That is a typical overfitting regime. Based on those observations, the number of hidden layer was chosen as one in this study.

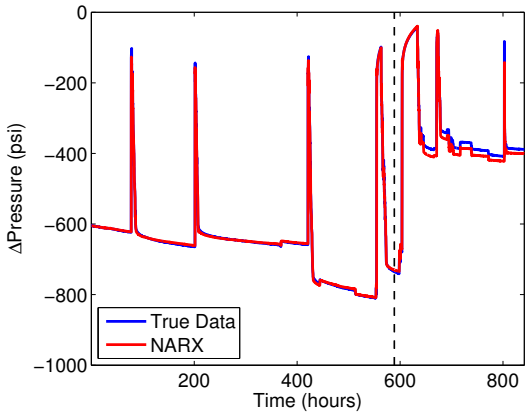
Number of Neurons

The number of neurons in the hidden layers controls the model capability by affecting the dimension of model parameters associated with those hidden layers. Recall the equation representing a NARX hidden layer discussed in Chapter 2:

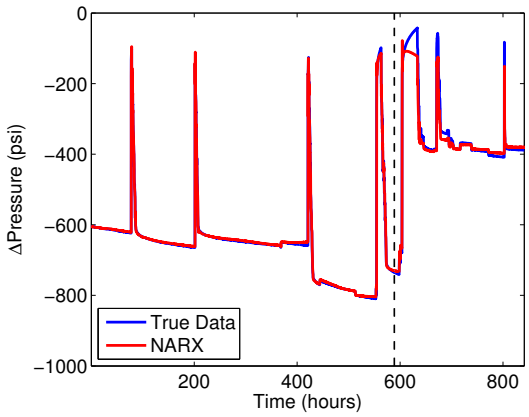
$$h^{(i)} = f_1(x^{(i-d_x)}W_{d_x} + ... + x^{(i-1)}W_1 + x^{(i)}W_0 + y^{(i-d_y)}U_{d_y} + ... + y^{(i-1)}U_1 + b_1), \quad (4.1)$$

where the inputs x have been selected as a vector with length two and the output y has dimension one. For h with l number of neurons, all Ws must have a dimension of $2 \times l$, all Us must have a dimension of $1 \times l$, and b_1 must have length l. Thus, by adjusting the dimension of h, we also change the dimension of all the Ws, Us and b_1.

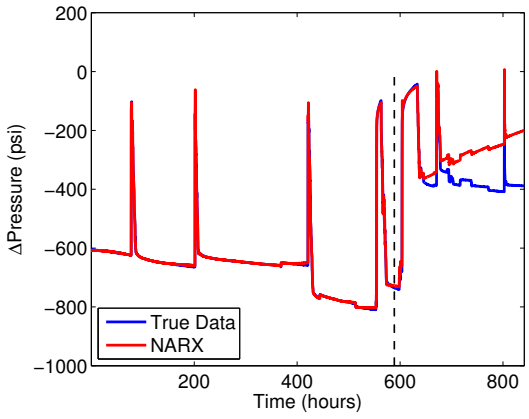
Here three NARX models with one hidden layer of three neurons, ten neurons, and twenty neurons were trained and tested following the same process as in the previous cases. The results are shown in Figure 4.4. The results of hidden layer with 20 neurons show an accurate fit to training data but a prediction with poor quality on test data. Again that indicates overfitting caused by model capability stronger than was actually needed given the problem complexity. As the number of neurons was reduced to three or ten, the NARX model showed good performance on both training



(a)



(b)



(c)

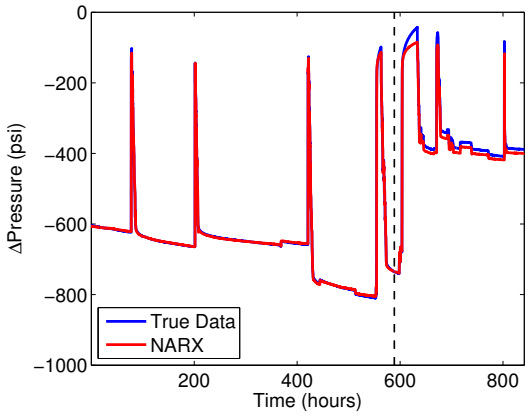
Figure 4.4: Comparison of pressure generated by NARX models with one hidden layer of (a) three neurons, (b) ten neurons, and (c) twenty neurons based on flow rate shown in Figure 4.1 (a).

data and test data. In fact, Figure 4.4 only includes a subset of the tested number of neurons, and our experiments show that a hidden layer with three to ten neurons generally performed better. Finally, a hidden layer with five neurons was selected in this study.

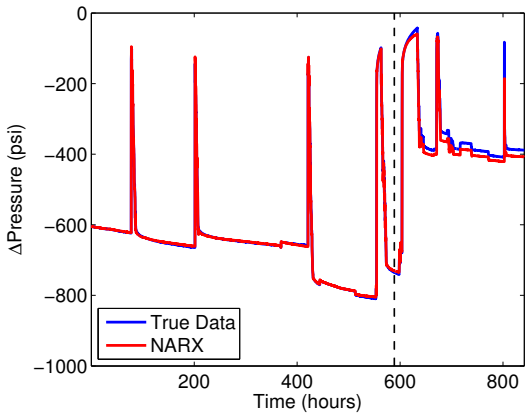
Delay

The delays of a NARX model refer to the input delays $x^{(i-d_x)}, ..., x^{(i-1)}$ and output delays $y^{(i-d_y)}, ..., y^{(i-1)}$ in Equation 4.1. The delays affect the model capability in two ways: on one hand delays control how far the memories of inputs and outputs are fed back to the model; on the other hand, delays affect the number of model parameters, i.e. the Ws and Us associated with the delays.

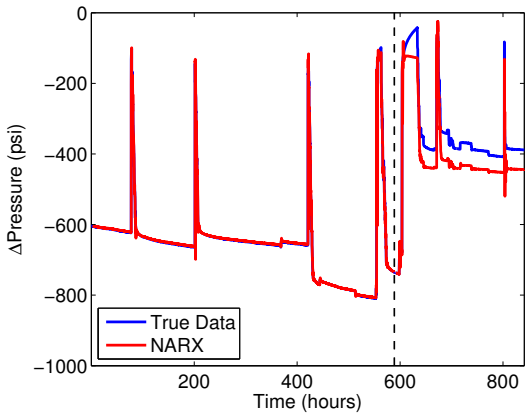
To simplify the problem, the same numbers of delays were used for inputs and outputs in this study. Figure 4.5 shows the pressure generated by NARX models with delays $d_x = d_y = 1$, $d_x = d_y = 3$ and $d_x = d_y = 20$. A short delay of one or three resulted in low training error and test error, while NARX with $d_x = d_y = 20$ overfitted the data with low training error and high test error. The effects of delays can be explained by the principle of superposition in well testing. The pressure can be represented by a convolution of previous rate changes, and the influence of early rate changes fades out as time goes forward. Thus the pressure is most highly related to recent rate changes, i.e. inputs with short delays. Similarly, the pressure is also related to recent pressure data rather than early pressure data. Base on those observations, $d_x = d_y = 2$ was selected in this study.



(a)



(b)



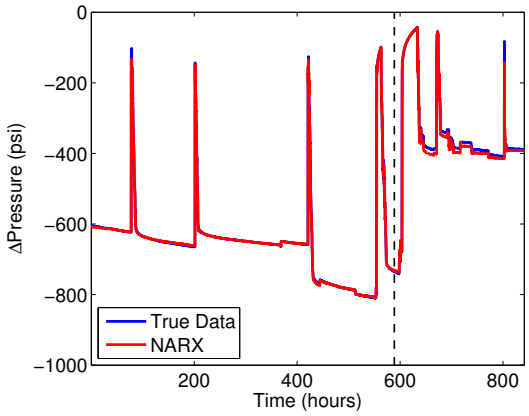
(c)

Figure 4.5: Comparison of pressure generated by NARX models with delay (a) $d_x = d_y = 1$, (b) $d_x = d_y = 3$, and (c) $d_x = d_y = 20$ based on flow rate shown in Figure 4.1 (a).

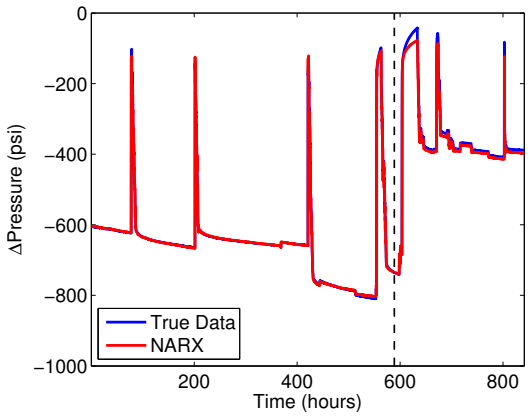
Activation Function

Activation functions are applied on each layer to add nonlinearity into the model before the information is passed to the next layer. Here we compared three common activation functions applied on the only hidden layer of a NARX model. The three activation functions were tanh function, sigmoid function and ReLU function. Shown in Figure 4.6, all three functions resulted in low training error as well as low test error, and the differences among the three functions were negligible. Although ReLU and tanh are often preferred compared with sigmoid in practice (Fei-Fei et al., 2017), we did not observe a significant difference during our hyperparameter tuning. Thus tanh was chosen to be applied on the hidden layer.

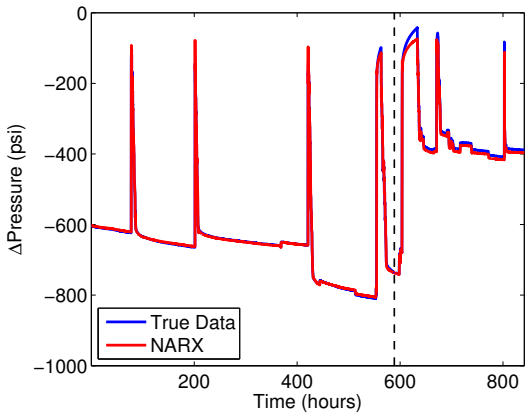
To summarize, a sensitivity analysis was performed on five hyperparameters for a NARX model, namely, the selection of inputs, number of hidden layers, number of neurons in the hidden layer, number of delays, and the selection of activation function. The optimal set of hyperparameters was identified by examining the training error and test error on a real PDG dataset. In fact, because the test data were used for hyperparameter tuning, the data should be really labeled as validation data rather than test data. The performance of the tuned NARX model on test data it never saw during tuning is discussed in the next section. For all the later cases in this chapter, we used the same NARX architecture with those optimized hyperparameters. We expect the same architecture will work on data with different sizes, because the problem complexity, i.e. the underlying physics are not dependent on the size of data.



(a)



(b)



(c)

Figure 4.6: Comparison of pressure generated by NARX models with (a) tanh function, (b) sigmoid function, and (c) ReLU function applied on hidden layer based on flow rate shown in Figure 4.1 (a).

4.2 Single-Well Data Interpretation

Case 1: NARX on Synthetic Flow Rate-Pressure Data

We first applied NARX on a synthetic flow rate-pressure dataset with 1000 data points over 200 hours. The pressure data were generated based on analytical pressure solutions. Raw data were used as inputs and outputs in NARX, i.e. $x^{(i)} = [q^{(i)}, t^{(i)}]$, $y^{(i)} = p^{(i)}$. Given that noise is commonly observed in PDG measurements, the noise tolerance of NARX was tested by adding a 3% Gaussian noise (mean zero, standard deviation 3% multiplied by its original value) to both flow rate and pressure data before training (Fig. 4.7(a)). After training, the same noisy training flow rate was fed to NARX, and the generated pressure (red) together with the training pressure (blue) are plotted in Fig. 4.7(b). The generated pressure is clearly smoother than the training pressure, indicating that NARX learned the patterns behind the noise rather than merely fitting the noisy training data. Such noise tolerance originates from the inherent assumption of a Gaussian error in the deep learning models. Next the clean flow rate before adding noise was fed to NARX, and the generated pressure was compared to the clean pressure (Fig. 4.7(c)). The close match shows that NARX did capture the pattern of flow rate-pressure behind the noisy training data. Lastly, Fig. 4.7(d) shows the results of pressure deconvolution generated based on a constant flow rate input. The derivative plot indicates that NARX identified the reservoir model correctly. A discrepancy before 0.2 hour is expected, given a less frequent data sampling of the training data compared to the deconvolution test data.

To further test the noise tolerance of NARX, the magnitude of Gaussian noise was increased to 15%, i.e. standard deviation as 15% multiplied by the values of original clean data. With a closer look at the pressure data in Fig. 4.8(a), the magnitude of

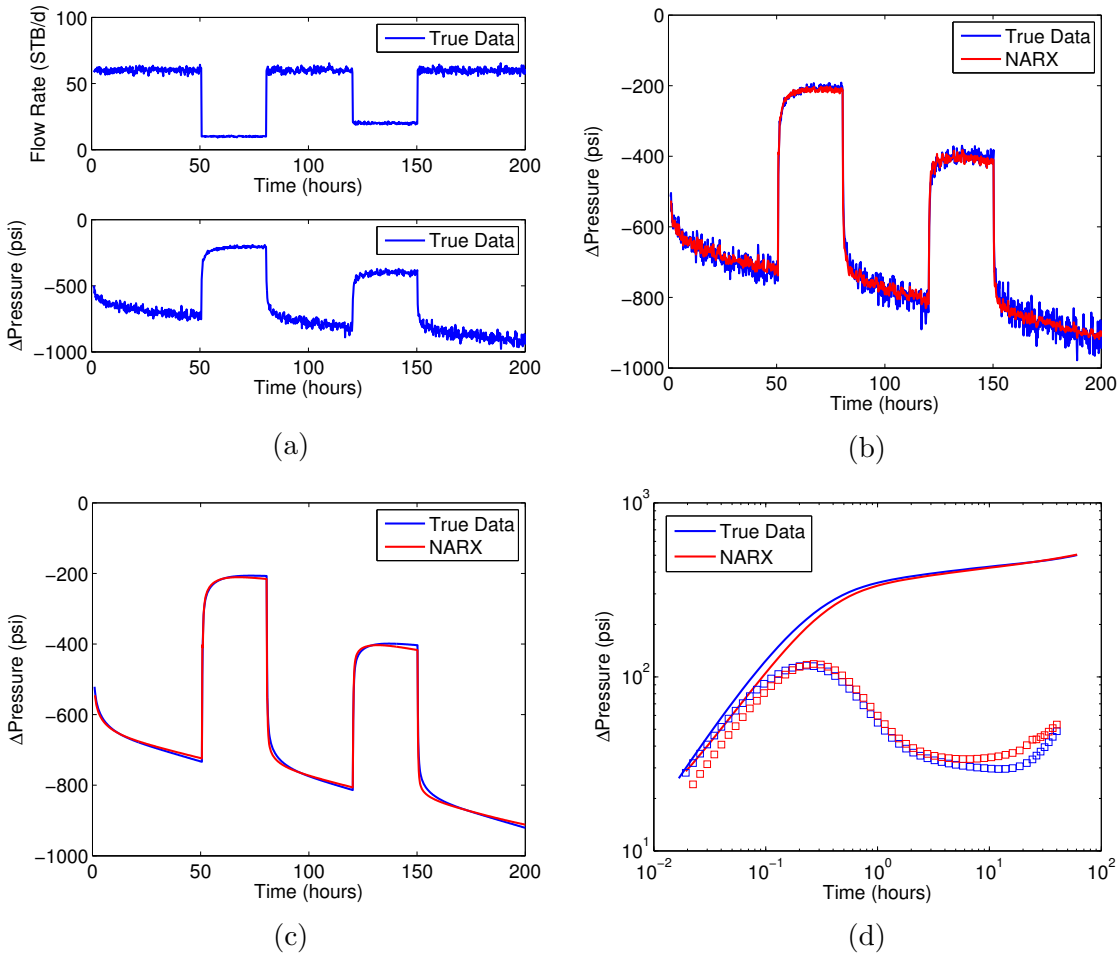


Figure 4.7: (a) Training data with 3% Gaussian noise. (b) Pressure generated by NARX based on noisy flow rate input. (c) Pressure generated by NARX based on clean flow rate input. (d) Pressure generated by NARX based on constant flow rate input (deconvolution).

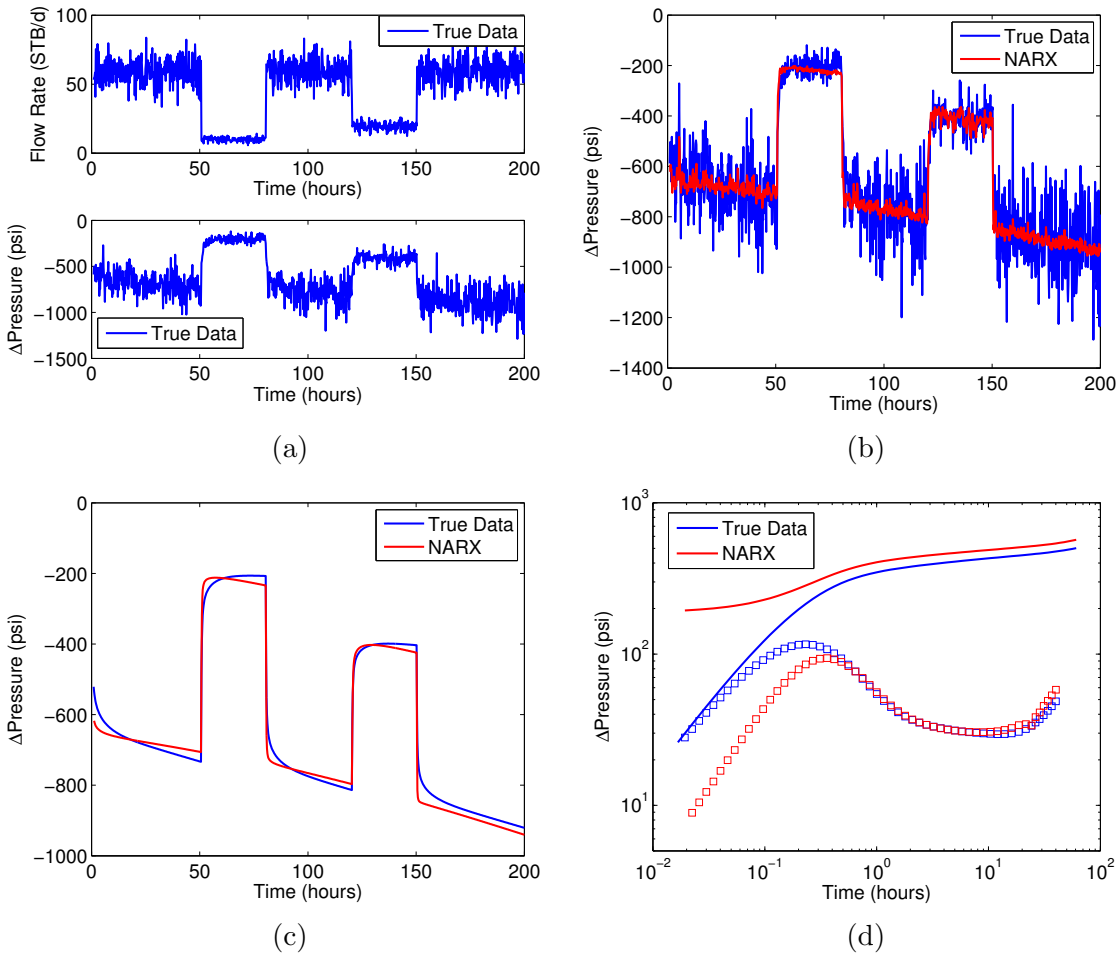


Figure 4.8: (a) Training data with 15% Gaussian noise. (b) Pressure generated by NARX based on noisy flow rate input. (c) Pressure generated by NARX based on clean flow rate input. (d) Pressure generated by NARX based on constant flow rate input (deconvolution).

noise between 150 hours to 200 hours is on the order of the actual pressure transient from 120 hours to 150 hours, which makes it challenging for NARX to differentiate the two. However we still see a significant noise reduction of NARX generated pressure in Fig. 4.8(b) and a credible match in Fig. 4.8(c). Because large noise hid much of the detailed pressure behaviors, the NARX result in Fig. 4.8(c) captured the general pressure decrease but not the early transient. A mismatch of early transient behaviors is observed again in the deconvolution test (Fig. 4.8(d)), while the overall reservoir model is still identified correctly.

Case 1 illustrates how NARX can be applied to identify the reservoir model by learning the patterns of flow rate-pressure data, even in the presence of Gaussian noise up to 15%. Unlike feature-based machine learning, NARX learns on raw flow rate-pressure data without handcrafted features, which frees the assumptions on the physics model.

Case 2: NARX on Semi-Real Flow Rate-Pressure Data

In Case 2, NARX was applied on a dataset with flow rate from a real PDG measurement and synthetic pressure based on that rate, to test NARX on a more realistic setting as well as maintain control of the true model. Fig. 4.9(a) shows the training rate and pressure data, which contain 1000 measurements over around 350 hours. The pressure history generated by NARX overlaps almost exactly with the training pressure in Fig. 4.9(b). The deconvolution result corresponding to a constant flow rate input is shown in Fig. 4.9(c), which indicates the correct reservoir model identification by NARX. Similar to Case 1, the mismatch in early transient can be explained by a higher data frequency in the deconvolution test (we intentionally designed the different data frequencies to test NARX performance under such setting).

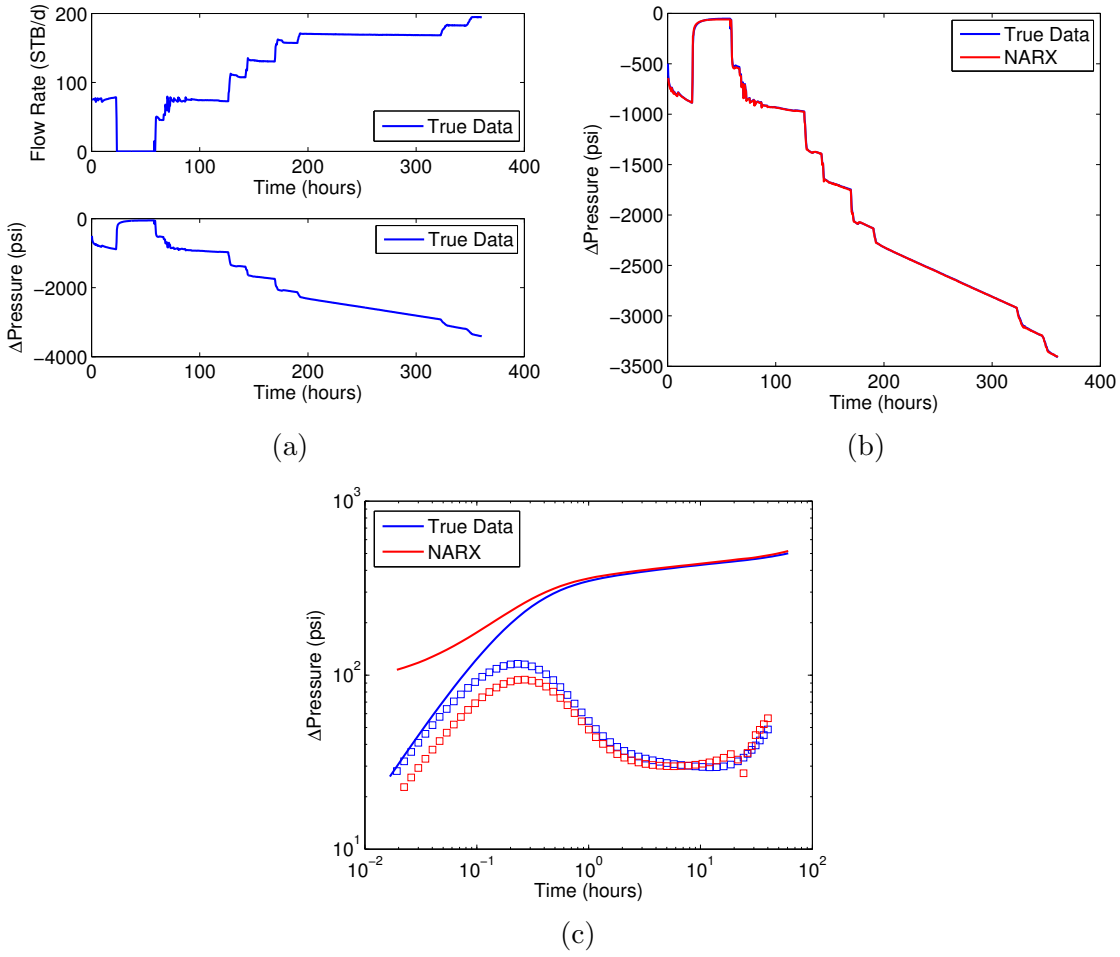


Figure 4.9: (a) Training data of flow rate from a real PDG and synthetic pressure. (b) Pressure generated by NARX based on flow rate input in (a). (c) Pressure generated by NARX based on constant flow rate input (deconvolution).

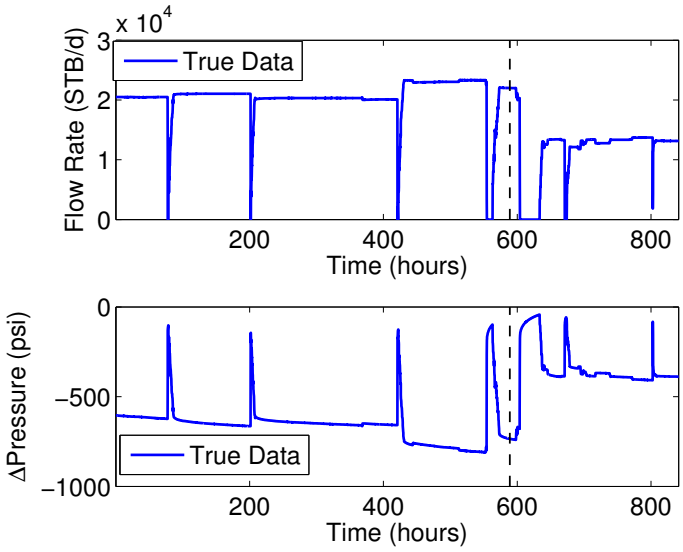
Case 3: Comparison of NARX and RNN on Real PDG Flow Rate-Pressure Data

In this case, we tested both NARX and standard RNN methods on the flow rate-pressure data obtained from a real PDG record. The data have 4665 measurements across about 850 hours (Fig. 4.10(a)). The models were trained on the first 70% of the data (left of the black dashed line), and tested against the true data on the remaining 30% (right of the dashed line). The pressure generated by NARX (red) and standard RNN (green) are shown in Fig. 4.10(b). Although there is little difference between NARX and RNN results on the first 70% part, NARX clearly outperformed RNN on the test data (the rightmost 30%). That validates our observation in Section 2.2.3 that NARX has at least as strong computational capability as RNN, given that the settings of our NARX and RNN models satisfy the assumptions in the statement. Such computational capability of NARX can be useful to forecast the reservoir pressure performance given flow rate controls.

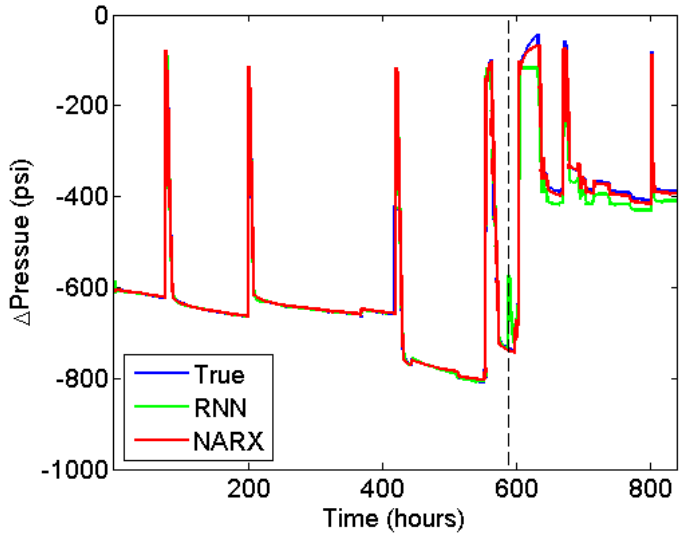
It is also worth pointing out that the computational time of NARX in this case was of the order of tens of seconds, although the exact time it took for each run may vary depending on different initializations. Such computational efficiency is comparable with feature-based machine learning (Chapter 3), and enables NARX to process large volumes of PDG data.

Case 4: Comparison of NARX and Feature-Based Machine Learning on Real PDG Flow Rate-Temperature Data

In this case, we tested NARX on temperature data from the same PDG as in Case 3, to show its advantage over feature-based machine learning when feature handcrafting is difficult.



(a)

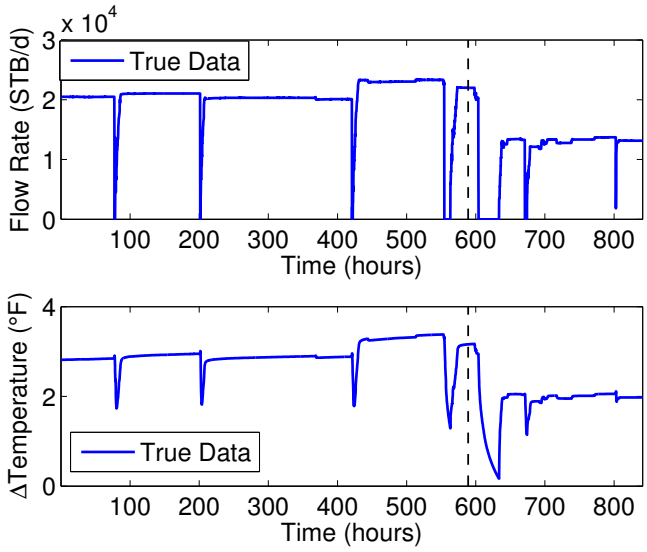


(b)

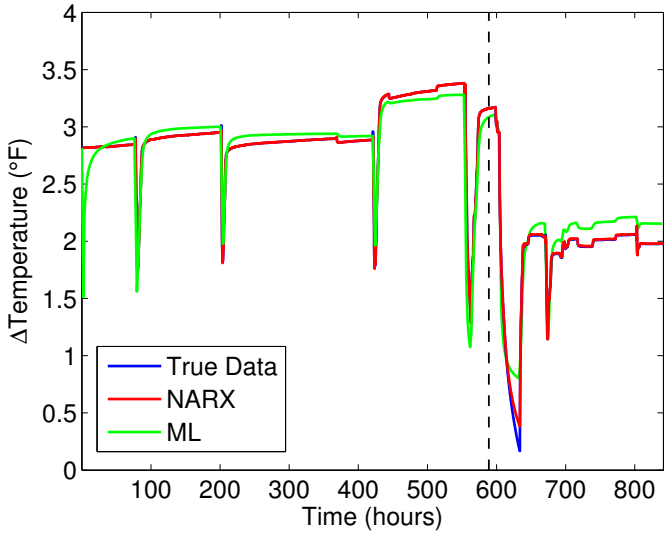
Figure 4.10: (a) Flow rate and pressure data from a real PDG. (b) Pressure generated by NARX and standard RNN based on flow rate input in (a).

The real flow rate and temperature data are shown in Fig. 4.11(a). Here we modeled temperature based on flow rate input, i.e. $x^{(i)} = [q^{(i)}, t^{(i)}]$, $y^{(i)} = T^{(i)}$. Fig. 4.11(b) shows that the temperature generated by NARX had a close match to the training data as well as good generalization of the test data. We also modeled the temperature with feature-based machine learning, using exponential integral function features $\Delta q \cdot Ei(\frac{c}{\Delta t})$ extracted from the analytical temperature transient solution (Palabiyik et al., 2016). Although feature-based machine learning also captured the trend of temperature variations, its modeling accuracy was much worse compared to NARX. Part of the reason comes from the discrepancy between analytical solution and real data, as well as the difficulties in feature design based on the analytical solution. Unfortunately those challenges are not unique to this case, but they are found commonly for many machine learning problems. In contrast, with NARX we can work simply on raw data as inputs and outputs, and skip the whole process of examining the likely analytical solution as an aid to handcrafting the features.

Next NARX was tested further in the situation where no forward model is available: modeling flow rate based on temperature. From Palabiyik et al. (2016), flow rate change is the cause of temperature variations but not vice versa. Thus no forward model has been developed to generate flow rate from temperature, and it would be very challenging to handcraft features without a forward model. Here NARX was applied on the same temperature-flow rate data as in Case 3, and the generated flow rate is shown in Fig. 4.12. The credible match to the true flow rate indicates that NARX learned the patterns mapping from temperature to flow rate, even though there is no well-developed mathematical formula for that mapping. From an engineering point of view, the mapping patterns learned by NARX can be useful in constructing flow



(a)



(b)

Figure 4.11: (a) Flow rate and temperature data from a real PDG. (b) Temperature generated by NARX and feature-based machine learning (ML) based on flow rate input in (a).

rate profiles given the measurements from numerous temperature sensors deployed in the field. Effectively, the temperature record becomes a flow-rate record.

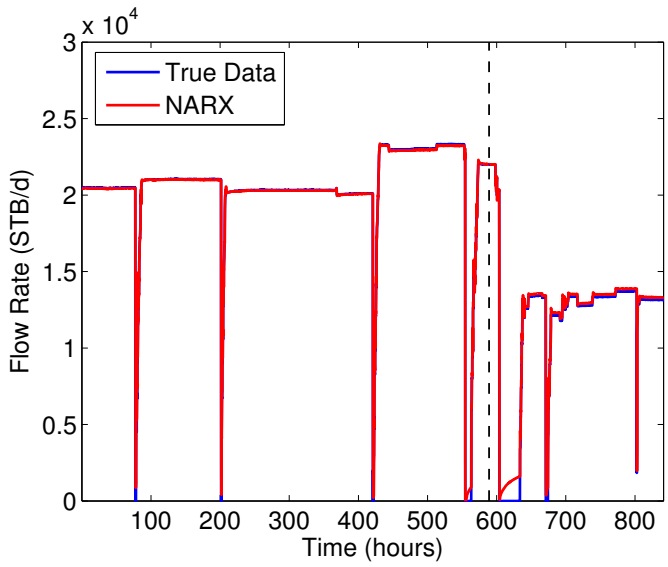


Figure 4.12: Flow rate generated by NARX based on temperature input in Fig. 4.11(a).

4.3 Summary

The main findings of this chapter are summarized here:

- In the application of recurrent neural networks (RNNs) to PDG data analysis, the study showed that RNNs, specifically NARX, provide a feature-free alternative to traditional machine learning methods to learn the useful patterns from PDG data. Such a characteristic makes NARX a preferred approach when feature handcrafting is difficult, e.g. limited knowledge of the physics.

- The work demonstrates how the patterns learned by NARX can be utilized

for reservoir model identification and reservoir performance forecasting, with various case studies.

- The work shows that NARX has the noise tolerance and computational efficiency that allow it to analyze PDG data in practice.
- With mathematical proof and a case example, we showed that the computational capability of a NARX is at least strong as a standard RNN, given they both have only one hidden layer of tanh function and an output layer with linear function.

Chapter 5

Pressure-Rate Deconvolution

Pressure-rate deconvolution refers to the process of extracting a drawdown pressure response corresponding to constant rate from a multirate pressure-rate history. Because the multirate data cover longer period than a single transient, it is possible to extract information for a larger radius of investigation by deconvolution. It has been shown in Chapter 3 and Chapter 4 that both feature-based machine learning and deep learning can be used for deconvolution by feeding a constant rate history into the trained models. This chapter discusses our rigorous investigation of deconvolution with machine learning and deep learning approaches on a series of case studies, and the comparison of the performance with two conventional deconvolution methods developed by von Schroeter et al. (2004) and Levitan et al. (2006).

5.1 Methodologies

The multirate data used for deconvolution testing were generated analytically based on the parameters shown in Table 5.1. The same parameters were used to generate

the reference of true deconvolution response, i.e. the pressure corresponding to the constant rate history with the same duration as the multirate data. A closed reservoir boundary model was used in Case 1 to Case 3, and a constant pressure boundary model was used in Case 4.

Table 5.1: Parameters used to generate the data for deconvolution in Case 1 to Case 4.

Parameter	Value
k (md)	33
S	2
C (STB/psi)	0.01
μ (cp)	0.65
h (feet)	200
ϕ	0.25
c_t (/psi)	2.355×10^{-5}
r_w (feet)	0.354
r_e (feet)	1800
B	1.3
p_i (psia)	5000

The deconvolution processes using the feature-based machine learning approach and deep learning approach are identical, thus we use the feature-based machine learning approach as an example for discussion. The deconvolution process using machine learning approach is the same as the procedure of the deconvolution tests in the earlier chapters: first train the algorithm on a multirate pressure-rate data, then feed the trained model with constant rate history to obtain the deconvolved pressure. The only difference is that the original pressure p was used instead of the pressure change Δp, in order to have the same deconvolution data for machine learning and the two conventional deconvolution methods in Saphir (Saphir requires p for deconvolution). The machine learning approach adds the intercept obtained from training data to the pressure prediction on test data, thus an inherent assumption

of the machine learning approach is that training data and test data share the same initial pressure p_i. Therefore $p_i = 5000$ psia was used along with the predicted pressure by machine learning to obtain Δp to be plotted on log-log scale.

The two deconvolution methods developed by von Schroeter et al. (2004) and Levitan et al. (2006), were tested based on their implementations in the commercial software Saphir. In this study, we refer to the two methods as the "von Schroeter method" and the "Levitan method" for simplicity. The von Schroeter method was applied on the full multirate data, and it generated one deconvolved pressure with the same duration as the multirate data. We used the full dataset to make it a fair comparison between the von Schroeter method and the machine learning approach, given machine learning was trained on the full dataset. The Levitan method can be also applied on the full multirate data, however, it generates a separate deconvolved pressure signal for each transient period. The length of each deconvolved pressure equals the end time of the transient period that deconvolution is applied to. We selected the pressure deconvolution on the last pressure buildup, because of the common practice of interpreting PDG buildup data (Levitan et al., 2006) and the rich information contained in data of the late periods. The same as the machine learning approach, p_i was forced to be 5000 psia for both the von Schroeter method and Levitan method. That prevents the possible failure of the von Schroeter method and Levitan method caused by inaccurate estimate of p_i, and it also ensures the impartial comparison of the three methods with the same p_i input.

The von Schroeter deconvolution method and the Levitan deconvolution method are reported to perform well when certain conditions are met, for instance, p_i estimate is relatively accurate, and the data for deconvolution are consistent, i.e. reservoir

parameters remain the same. Those two conditions were satisfied in our analysis, given the true $p_i = 5000$ psia was fed to the deconvolution algorithms and the data were generated using constant reservoir parameters (as listed in Table 5.1). In fact, both methods performed quite well on a simple test case with clean multirate data. The estimated permeabilities by the von Schroeter method and the Levitan method were 33.2 md and 33.1 md respectively, almost identical to the true permeability of 33 md. However, there are still a number of other practical issues to be considered when analyzing real PDG data, for instance: (a) data with noise, (b) data with outliers, and (c) data only containing short transients in which the effect of a reservoir boundary is not seen.

A series of cases were designed to represent those complexities. The four deconvolution methods, namely the feature-based machine learning method, the deep learning method, the von Schroeter method and the Levitan method were tested on those cases. We first discuss the comparison of deconvolution by feature-based machine learning against the two conventional methods, followed by comparing the deconvolution results by deep learning against feature-based machine learning.

5.2 Deconvolution by Feature-Based Machine Learning

In this section, deconvolution by feature-based machine learning is compared against the von Schroeter method and the Levitan method on four cases, which represent various complexities associated with PDG data deconvolution in practice.

Case 1: Deconvolution on Data With Noise in Pressure

We first tested the three methods on a dataset with 1% Gaussian noise (mean zero, standard deviation 1% multiplied by its original value) in pressure. Here Gaussian noise was added instead of the uniform noise we saw in Case 1 and Case 3 in Chapter 2, to test the robustness of the machine learning approach in interpreting data with different types of noise. Shown in Figure 5.1(a), the noisy pressure (magenta) was used along with the clean flow rate (blue) for deconvolution. In Figure 5.1(b), the pressure prediction by machine learning corresponding to the clean rate in Figure 5.1(a) was compared with the true pressure before noise was added. Here the linear regression approach (LR) was selected as the machine learning technique, given its reliable performance on data containing mostly late-transient behaviors. The comparison of the deconvolution results by the three methods is summarized in Figure 5.1(c). The pressure derivative of the true data (blue) indicates the presence of infinite-acting radial flow and pseudosteady state starting at around 200 hours. All three methods captured the radial flow and pseudosteady state characteristics, although the linear approach deviated from true data during the transition period between radial flow and pseudosteady state, while the von Schroeter method and the Levitan method led to derivative curves with oscillations. Another point to pay attention to on the log-log plot is that the pressure derivatives by both the von Schroeter method and the Levitan method deviated highly from the true derivative in the early-transient period, in particular when $t < 1$ hour. That deviation was also observed in later cases from Case 2 to Case 4, whose deconvolution data all started with a time sequence $t = 0.08, 0.16, 0.32, 0.64, 1.28$ hour. Thus the deviation should be interpreted as the performance of the two methods on the particular early time intervals, rather than their performance on noisy data.

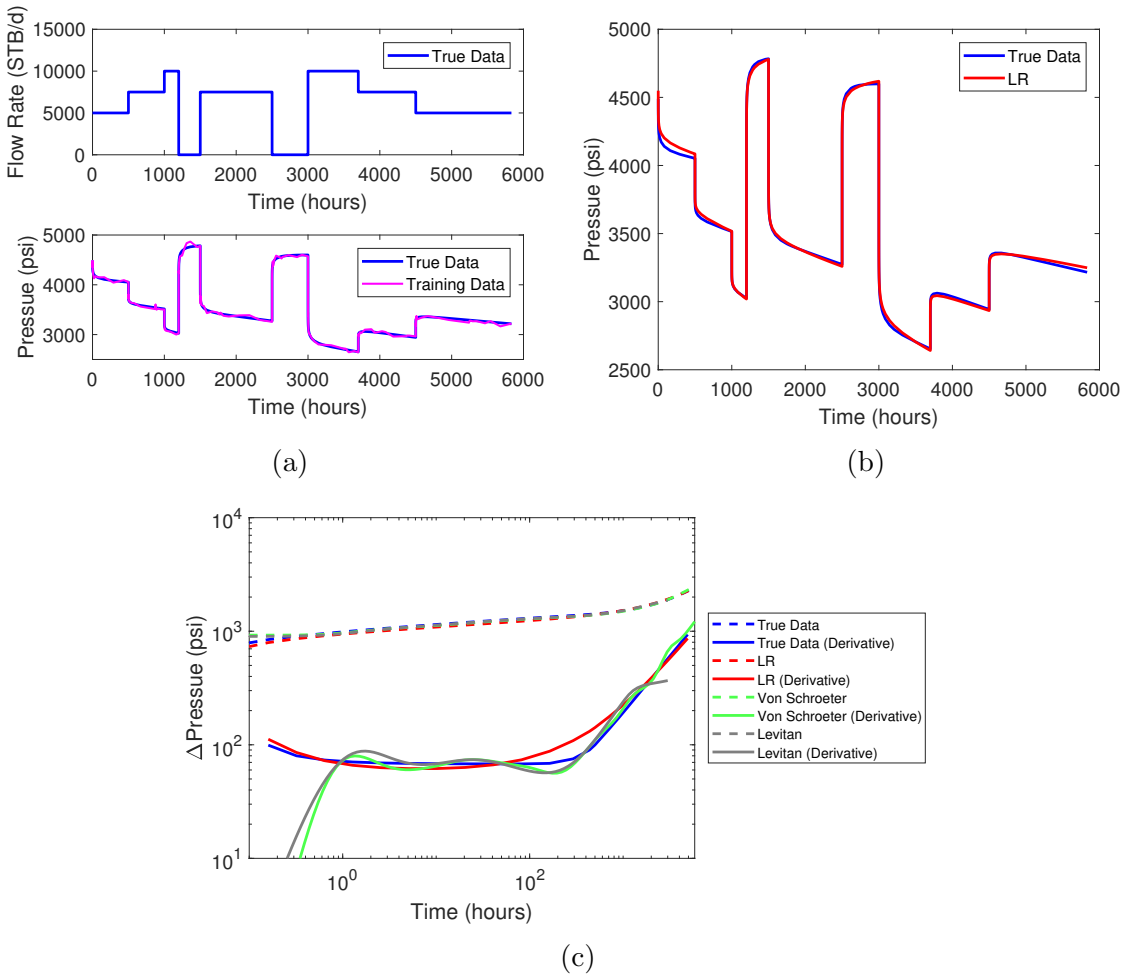


Figure 5.1: Deconvolution on data with 1% Gaussian noise in pressure.

The three methods were tested further on data with 5% Gaussian noise in pressure (Figure 5.2(a)). In Figure 5.2(b), the pressure predicted by the linear approach was less accurate compared to the case with 1% noise in pressure. Some obvious discrepancies can be observed at $t = 3000$ hour and $t = 5000$ hour. On the log-log plot, the machine learning approach still captured the characteristics of radial flow and closed boundary, while the pressure derivative by the Levitan method barely indicated any reservoir information. The upward trend towards the end of deconvolved pressure derivative by the von Schroeter method seemed to indicate the presence of a closed boundary, however, the poor quality of deconvolution made the interpretation very difficult. The test shows that the linear approach for deconvolution is much less sensitive to noise compared to the von Schroeter method or the Levitan method. One of the reasons is that the machine learning approach generates pressure profiles using features in Equation 2.3, which are based on physics of pressure transients. It is not misled by the noise because noise has different characteristics that can not be modeled by those features.

Case 2: Deconvolution on Data With Outliers in Pressure

In this case, the three methods were tested in the presence of outliers in pressure data. Outliers were added to the pressure by randomly selecting 1/3 of the data, and moving those data points 200 psi up or down with a 50/50 probability (Figure 5.3(a)). We can see from the log-log plot that the machine learning approach was again able to identify the reservoir model correctly, although there were discrepancies at around $t = 200$ hour. Such discrepancies at the transition from radial flow to pseudosteady state were observed in all deconvolution results by machine learning, because there were no features designed to describe the transition behaviors. In Figure 5.3(c), both

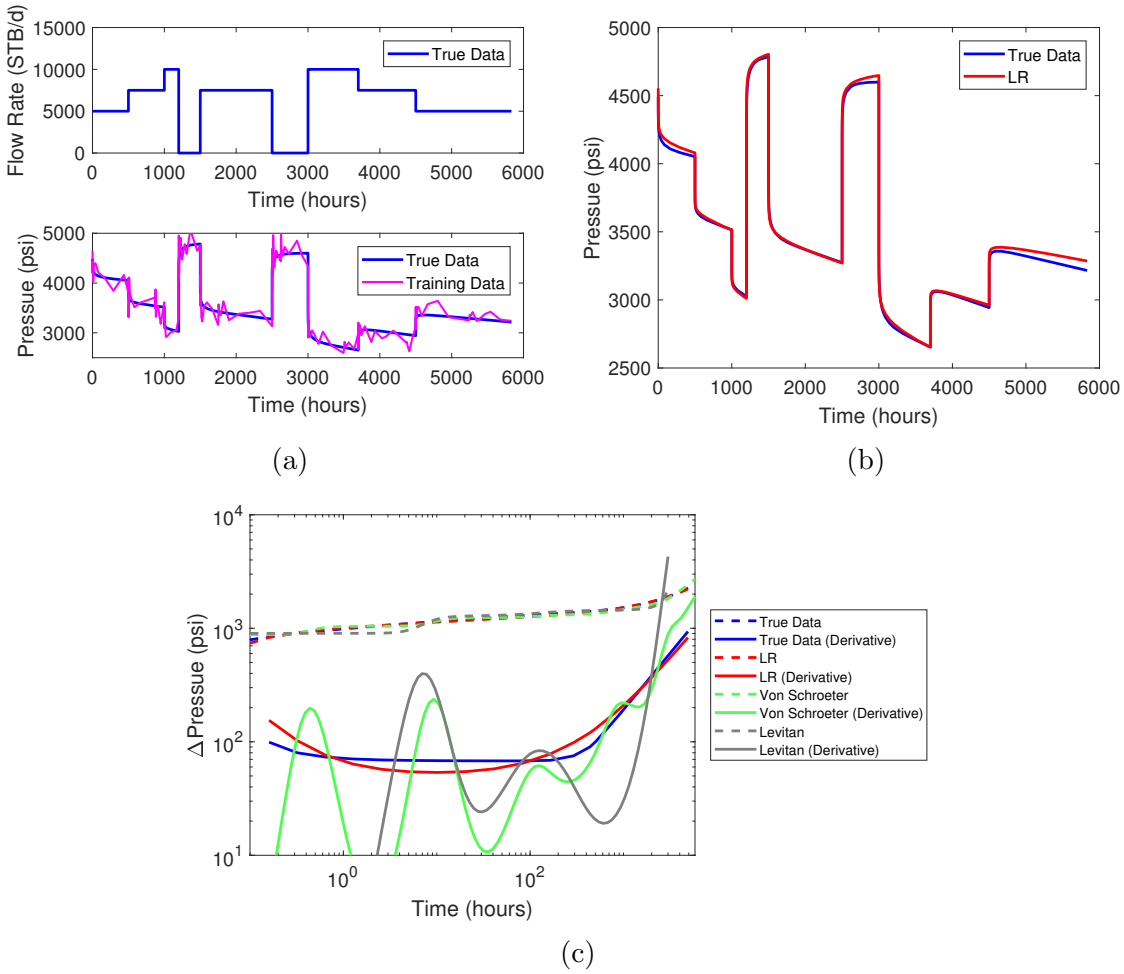


Figure 5.2: Deconvolution on data with 5% Gaussian noise in pressure. Deconvolution by the linear approach (LR in red) shows higher tolerance to noise compared to the von Schroeter method (green) and the Levitan method (gray).

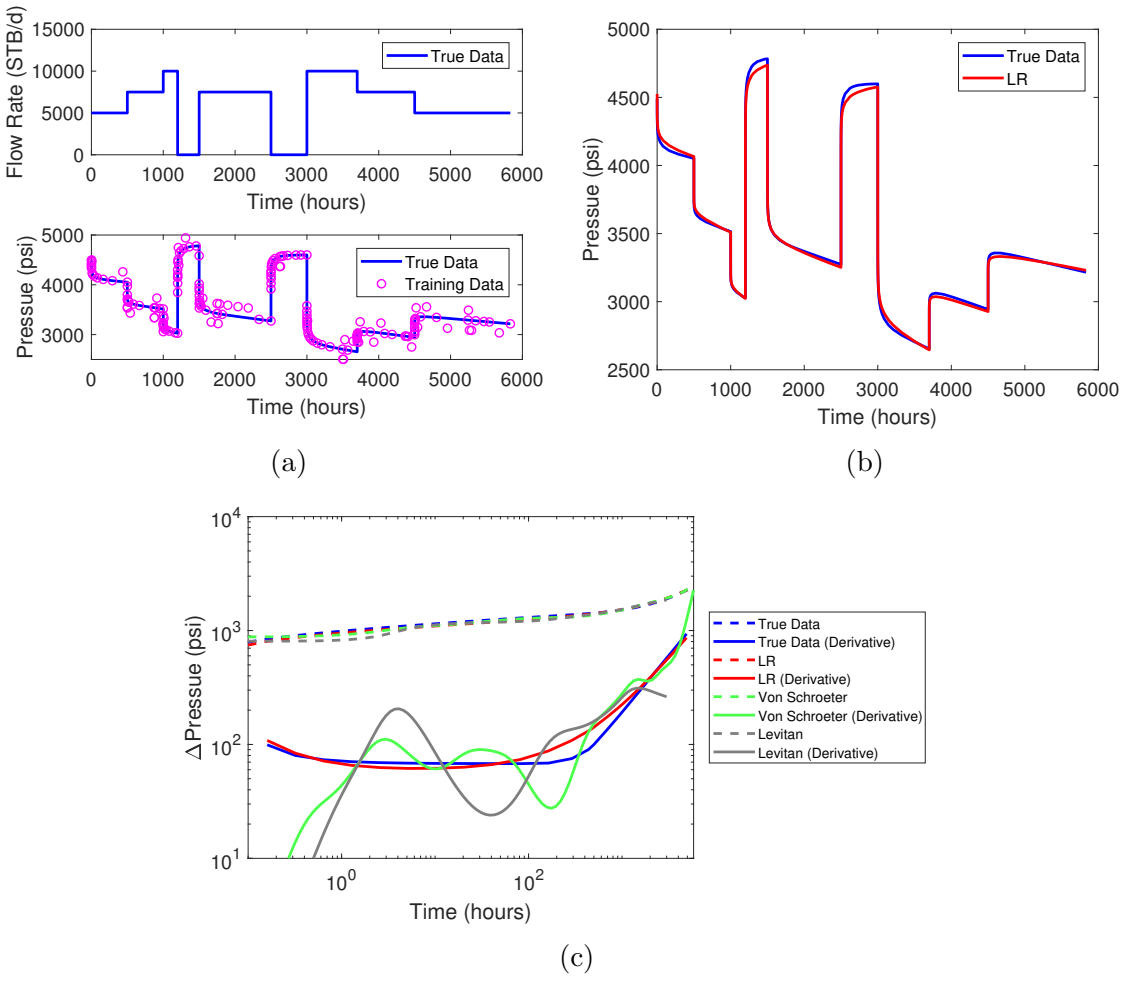


Figure 5.3: Deconvolution on data with outliers in pressure.

the von Schroeter method and Levitan method seemed to capture the pseudosteady state behavior, but the strong oscillations made radial flow hardly detectable.

Case 3: Deconvolution on Data With Short Transients

This case was designed to test the capability of the deconvolution algorithms in revealing the reservoir information for a larger radius of investigation from a multirate history than a single-rate transient. We see from the log-log plot that pseudosteady state started at around 200 hours. An interesting question is, can the deconvolution algorithms capture the pseudosteady state if all the transients in the multirate data lasted less than 200 hours?

For the deconvolution data shown in Figure 5.4(a), all the step rates lasted shorter than 100 hours, thus no information on reservoir boundary can be revealed by applying conventional well test interpretation on any single one of those transients. Next the deconvolved pressure was generated with input of a constant rate that lasted for 2000 hours, the same duration as the multirate data. The deconvolution results of the three methods are shown in Figure 5.4(c). The von Schroeter method captured the closed boundary with a distinct unit slope on the pressure derivative starting at $t = 500$ hour. The pressure derivative of machine learning approach showed a upward trend starting at 100 hours, but still did not reach unit slope at the end of the test. The Levitan approach led to a wrong interpretation of the boundary. This case demonstrated the use of the machine learning approach deconvolution method in learning reservoir information for a larger radius of investigation from multiple short transients. That approach can be useful when the operators can not afford to run long buildup tests.

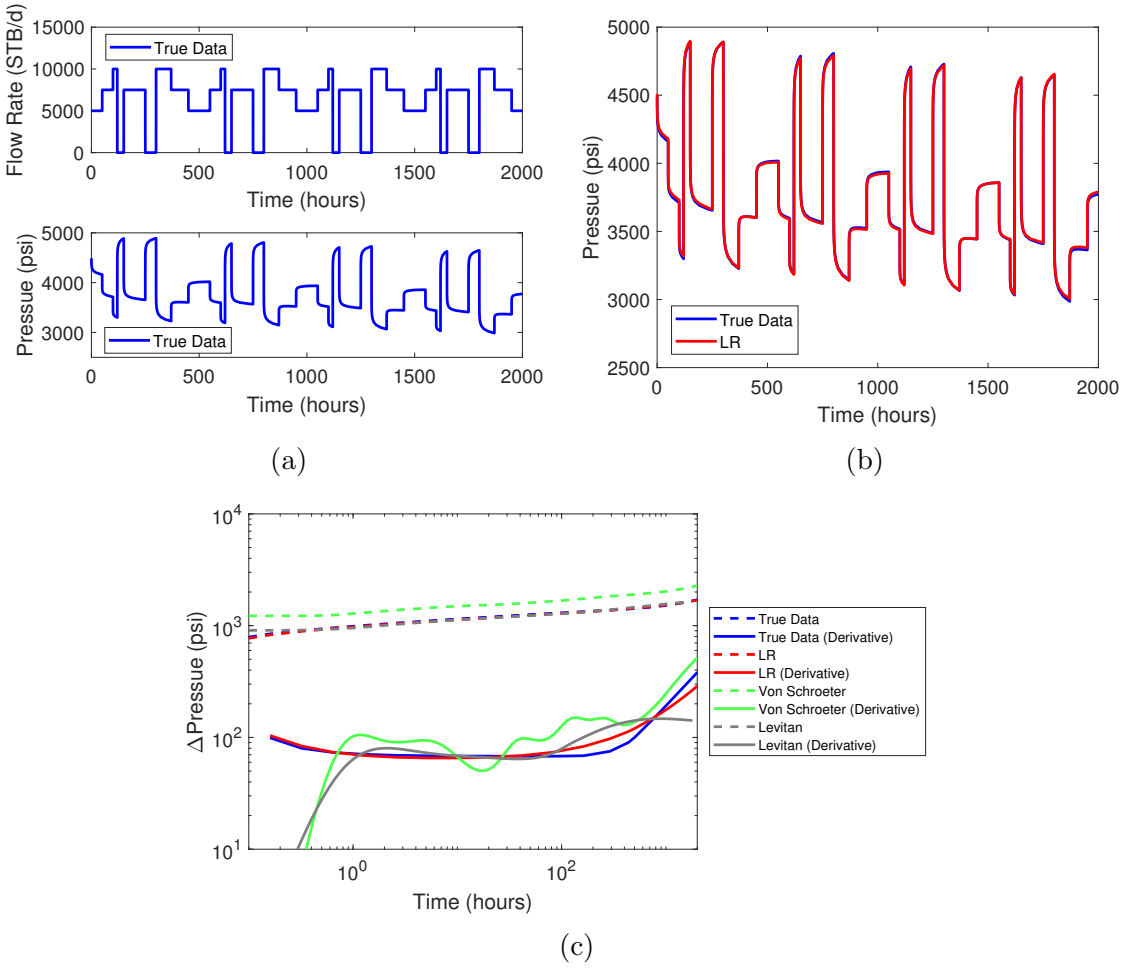


Figure 5.4: Deconvolution on data with short transients. Pseudosteady state cannot be observed from any of the single transient in (a).

Case 4:Deconvolution on Data With Constant Pressure Boundary

There are few discussions on deconvolution on data with constant pressure boundary behaviors in the work by von Schroeter et al. (2004) and Levitan et al. (2006), therefore we designed a case with constant pressure boundary, to test the robustness of the deconvolution methods on data with different boundary types. The multirate data generated based on the constant pressure boundary model and the parameters in Table 5.1 are shown in Figure 5.5(a). In Figure 5.5(c), the constant pressure boundary is represented by a pressure derivative moving downwards following the flat region. All the three methods identified the boundary type correctly, although the Levitan method had the closest match with the true data. The pressure derivatives of the late-transient period by the machine learning approach and the von Schroeter method almost overlapped on top of each other.

5.3 Deconvolution by NARX

Next, NARX was applied to deconvolution on the same data sets used in Case 1 to Case 4. Deconvolution results by NARX were compared with feature-based machine learning, shown in Figure 5.6 to Figure 5.9. Some key findings are summarized as follows.

First of all, in all the cases, NARX had a credible match to the training data, i.e. data for deconvolution. NARX also identified the reservoir models correctly as indicated by the type curves of infinite-acting radial flow, closed boundary and constant pressure boundary on the log-log plot. That demonstrates the ability of NARX to extract the underlying reservoir information from multirate data.

Secondly, NARX has certain tolerance to noise, although its noise tolerance is less

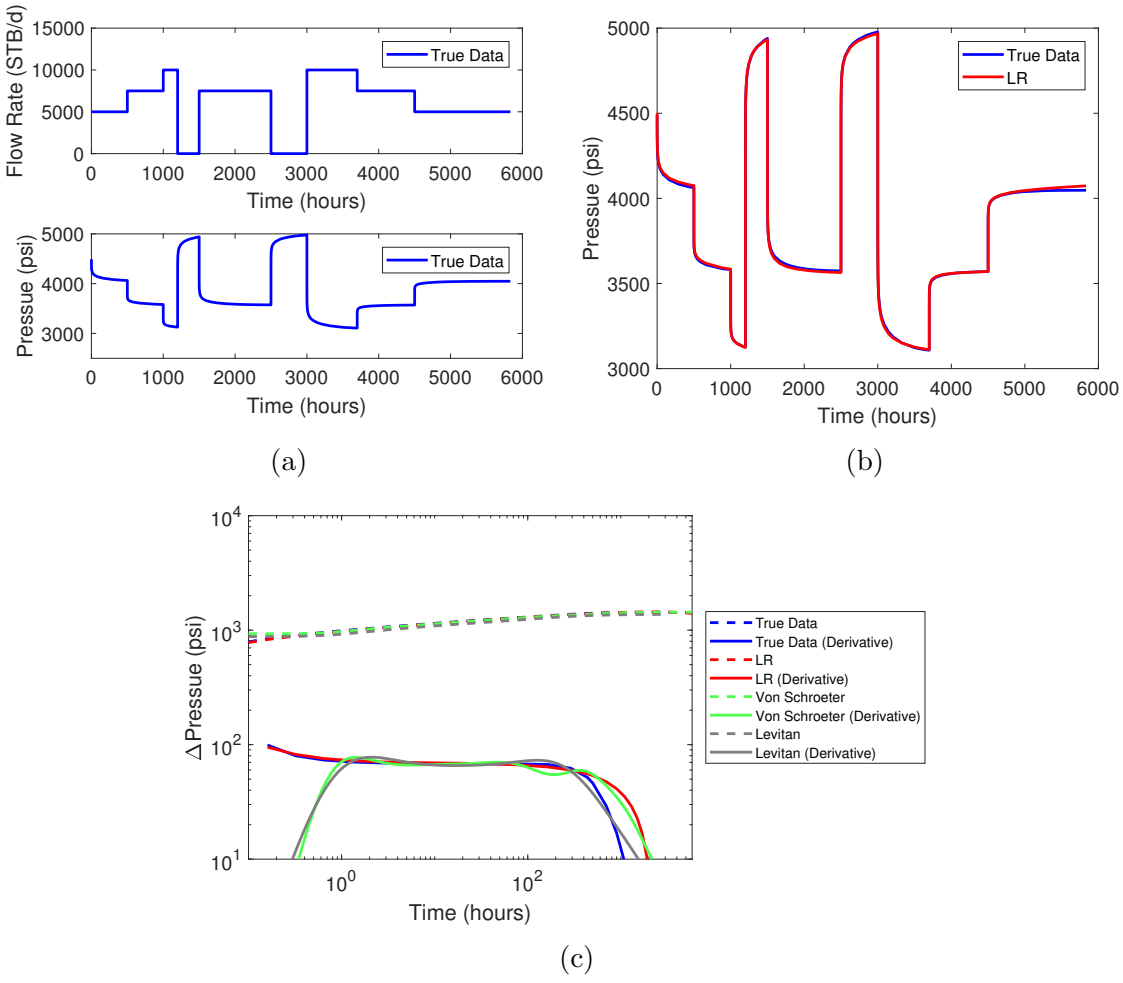


Figure 5.5: Deconvolution on data with constant pressure boundary.

compared with feature-based machine learning. Shown in Figure 5.6(b) and Figure 5.7(b), NARX generated smooth pressure predictions even if it was trained using noisy pressure, which suggested that NARX did not merely match the noisy pressure during training. However, pressure derivative by NARX in Figure 5.7(c) did show more wiggles compared with Figure 5.6(c), when the noise level increased from 1% to 5%.

Thirdly, pressure derivatives generated by NARX contain characteristics that cannot be explained by physics, for instance, the wiggles around 500 hours. Unlike machine learning using features based on physics, there were no constraints to force the NARX model to obey the physics. In fact, NARX tried to learn those physics with its unique architecture. The feature-based machine learning approach showed its advantage in terms of generating pressure and pressure derivative profiles that honored the physics.

To sum up, deconvolution by NARX was slightly less accurate compared to feature-based machine learning, especially when the data contained noise. Nevertheless, NARX correctly identified the reservoir models for all the test cases, and demonstrated a higher tolerance to noise than the conventional deconvolution methods such as the von Schroeter method and the Levitan method.

5.4 Summary

To summarize, although the von Schroeter method and the Levitan method work well to deconvolve the data in certain situations (e.g. clean data, accurate p_i estimate), those methods can fail in practice when the data are noisy or contain outliers. Both the feature-based machine learning approach and NARX show high tolerance to noise

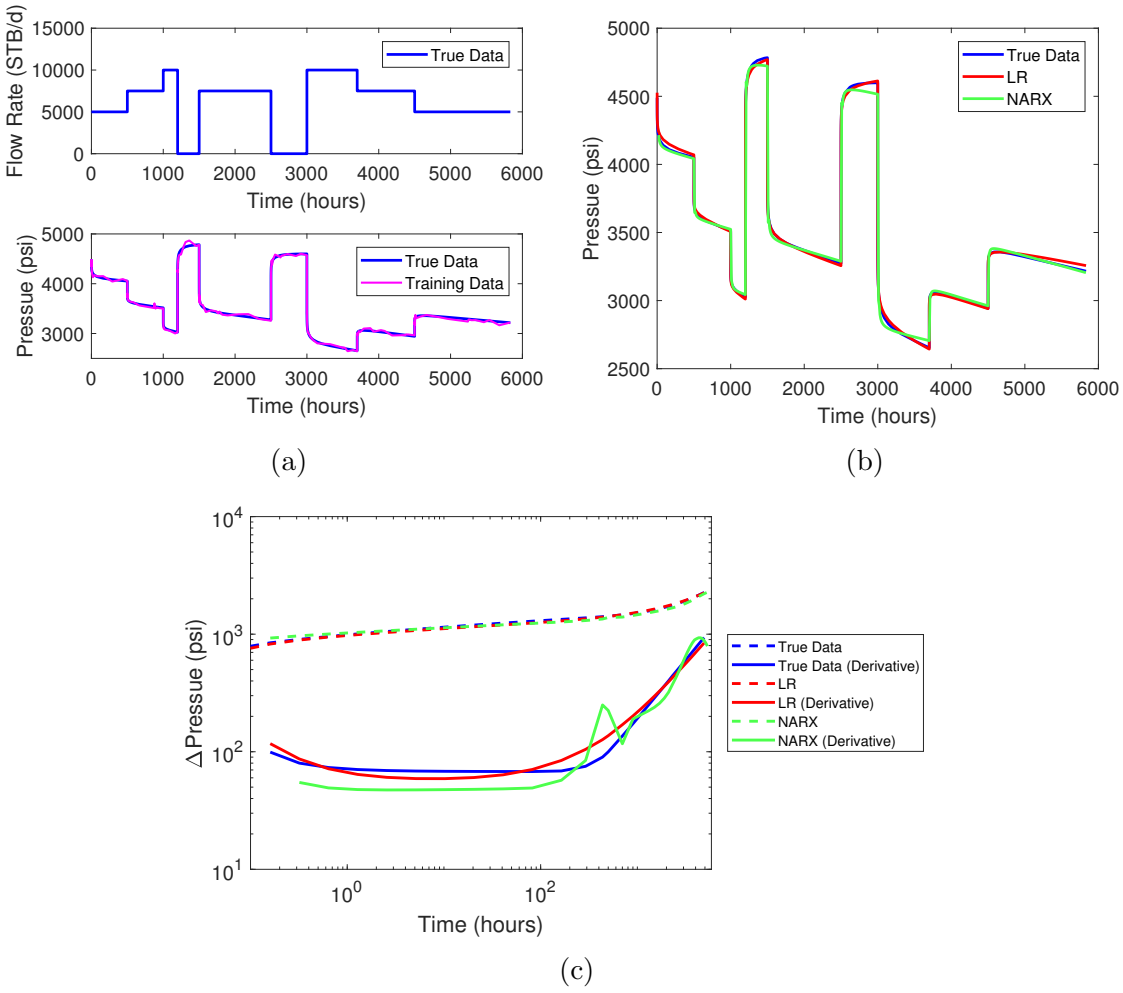


Figure 5.6: Deconvolution on data with 1% Gaussian noise in pressure.

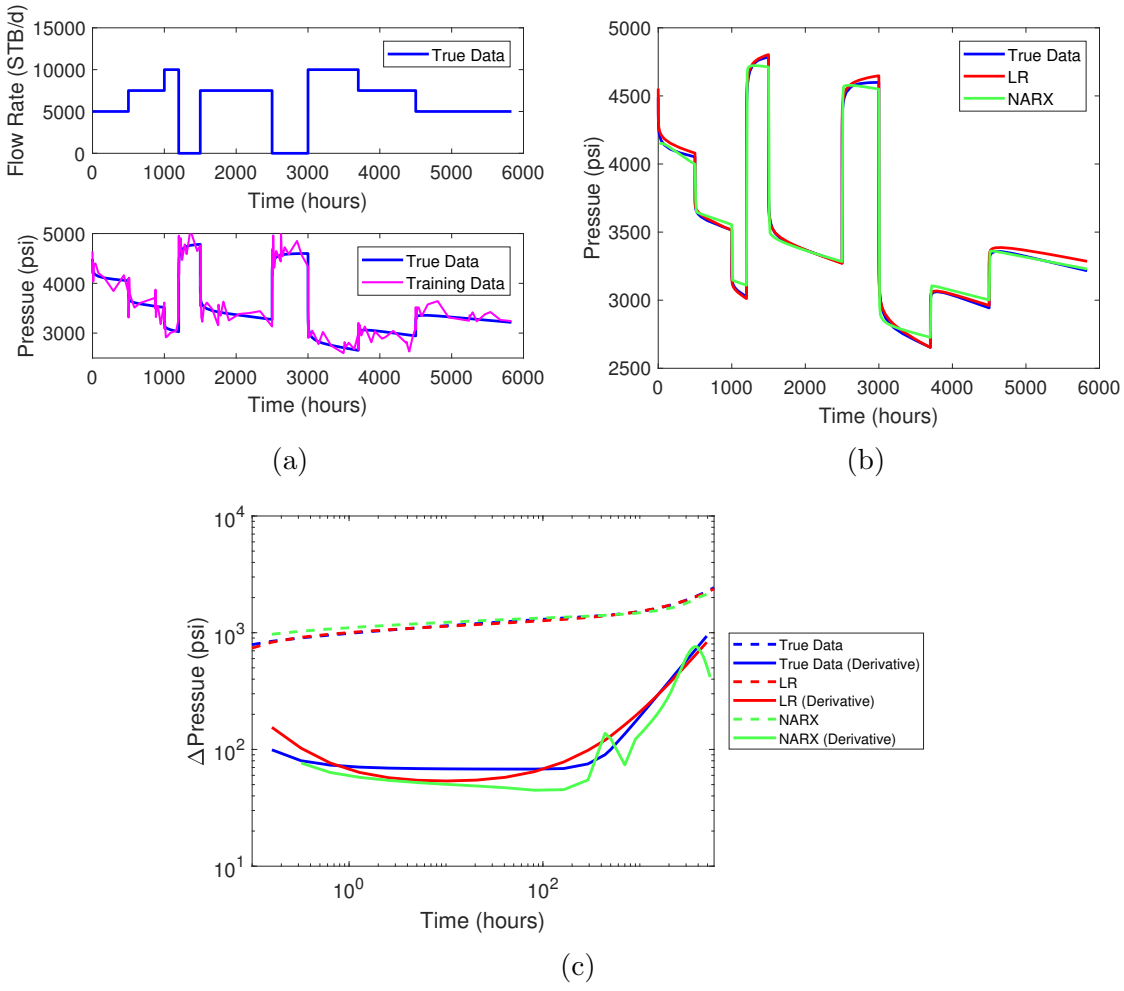


Figure 5.7: Deconvolution on data with 5% Gaussian noise in pressure.

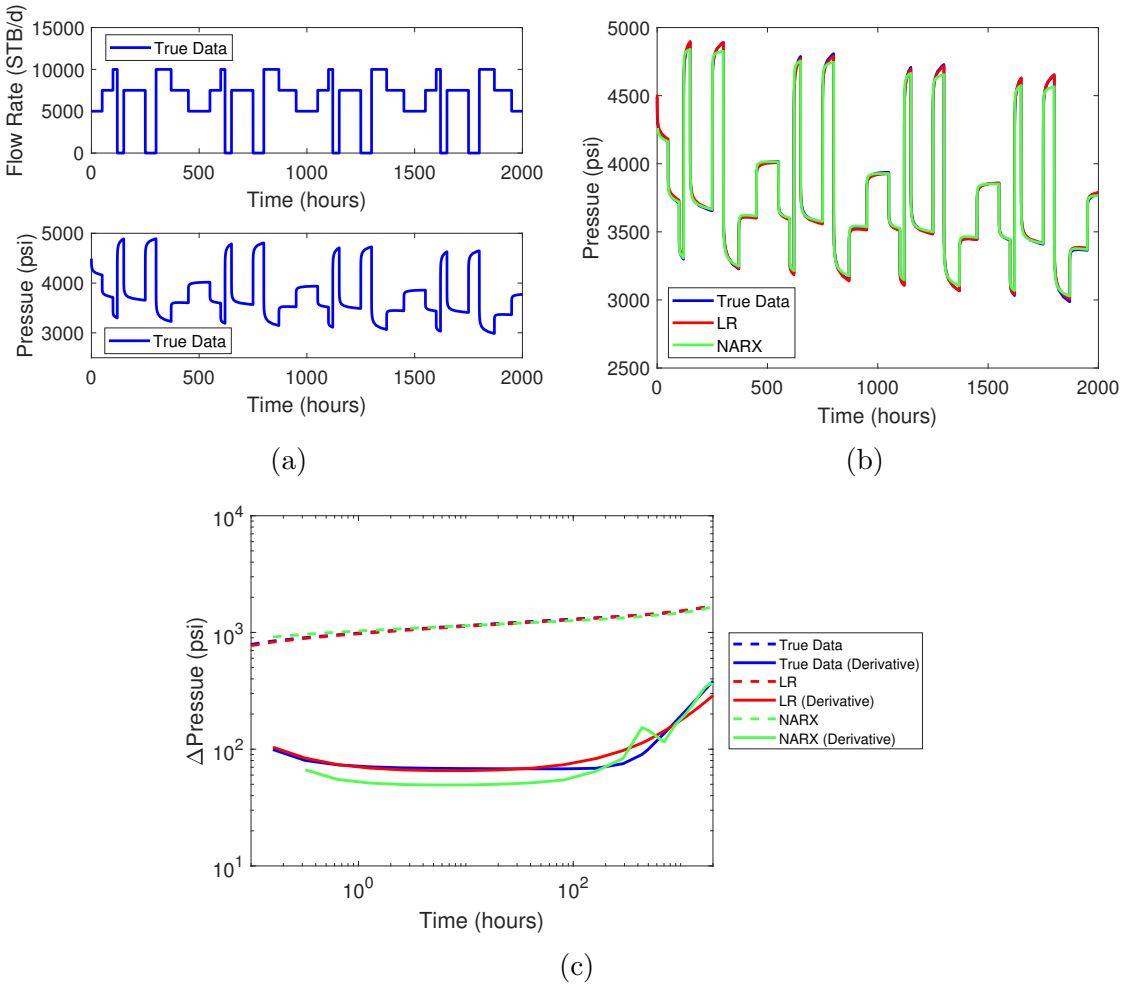


Figure 5.8: Deconvolution on data with short transients. Pseudosteady state cannot be observed from any of the single transient in (a).

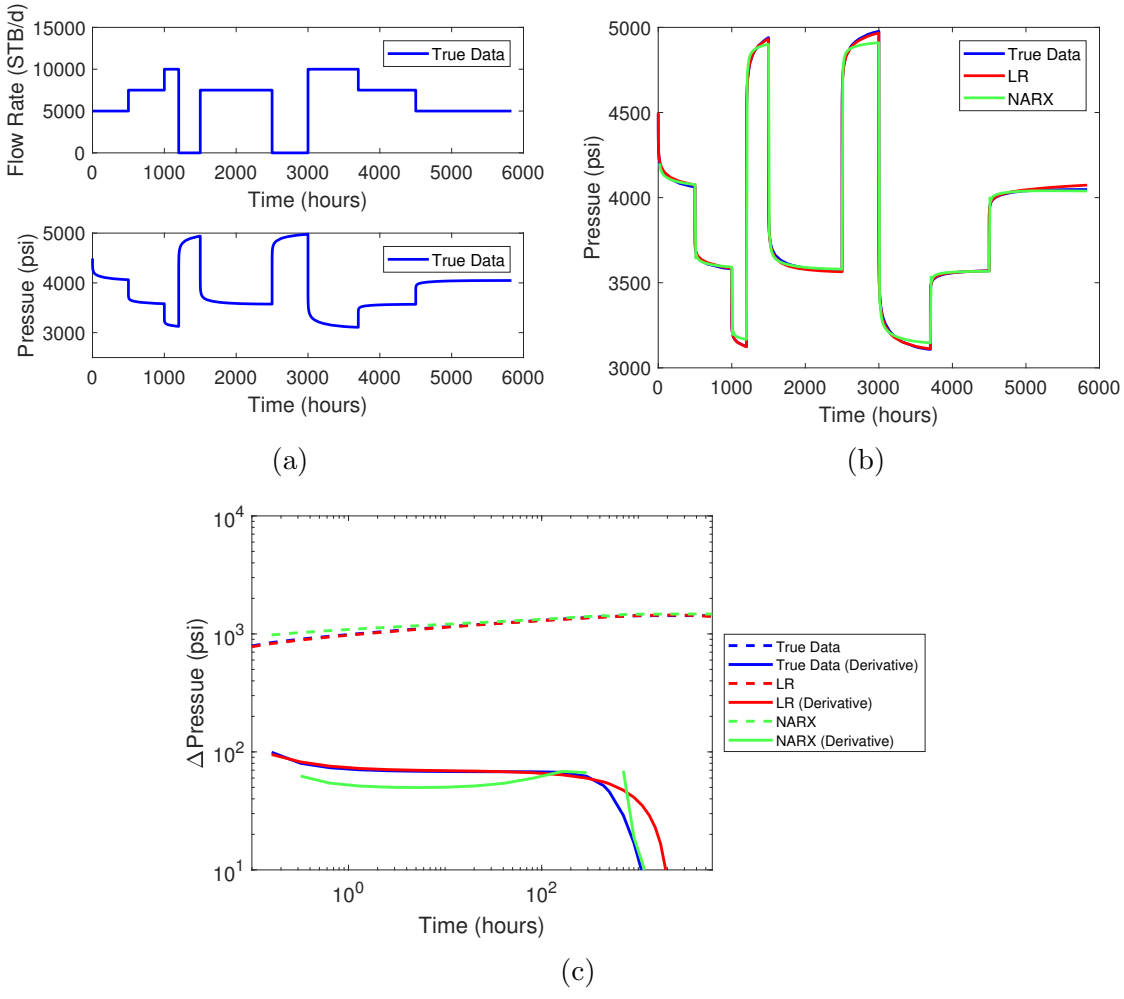


Figure 5.9: Deconvolution on data with constant pressure boundary.

and outliers, which are commonly observed in PDG measurements. They were also demonstrated to work with different reservoir boundary types and data with short transients. Thus the feature-based machine learning approach and NARX provide alternatives for deconvolution besides the conventional methods developed by von Schroeter et al. and Levitan et al. The feature-based machine learning based deconvolution looks even more promising given it does not require breakpoint detection in data preprocessing, which is a necessary but error-prone step for both the von Schroeter method and the Levitan method.

Chapter 6

Field Examples

6.1 Field Example 1: Flow Rate Reconstruction

In the first field example, we were interested in estimating the flow rate of a single well. The field data include pressure and temperature measured by a PDG, and the liquid rate measured by a test separator. There are 11359 measurements in the dataset. Because the test separator was used to test multiple wells, it was not always aligned to the well of interest. Thus there are some missing periods in the flow rate measurements. The objective of this study was to estimate the missing flow rate using reliable pressure data that were measured continuously.

The original data of the well are plotted in Figure 6.1. The values on the y-axis are covered, and the time was normalized (the actual data duration was of order of years). The opened-closed status is a binary measurement indicating whether the well was aligned to the separator: a high value means the well was aligned to the separator, and a low value means not aligned. It is easy to see that the measured flow rate data contain many zeros, but pressure buildup is not observed at every zero flow

rate period. In fact, the pressure profile indicates the well was flowing for most of the time. Therefore there is ambiguity in the zero flow rate: it can be either caused by an actual shut-in or by the misalignment between the well and the separator. That is why we aim to reconstruct the flow rate profile by using the pressure history, which is considered to be more reliable.

6.1.1 Data Preprocessing and Exploration

From the original data, it is easy to observe outliers, e.g. temperature around $t = 0.8$, and aberrant segments, e.g. the segment of $0.2 < t < 0.4$ where both pressure and temperature data are constant. Basic data preprocessing was conducted to remove those aberrant segments and outliers. The preprocessed data are shown in Figure 6.2. The goal of this field example study was to reconstruct the flow rate in 6.2(a) using PDG pressure in 6.2(c), based on the flow rate reconstruction methodology introduced in Chapter 2.

After obtaining the test data, it is important to select the right data for training. The training data should contain the physics to be learned by the algorithm. In this field example, one easy approach is to select the data conditioned to well being aligned to the separator. A section of data satisfying such condition is shown in Figure 6.3. The rate and pressure data in the selected period are consistent, for instance, the rate decrease at $t = 0.033$ led to a pressure jump at the same time. Thus the pressure-rate data shown in Figure 6.3 were used as training data for the rate reconstruction problem.

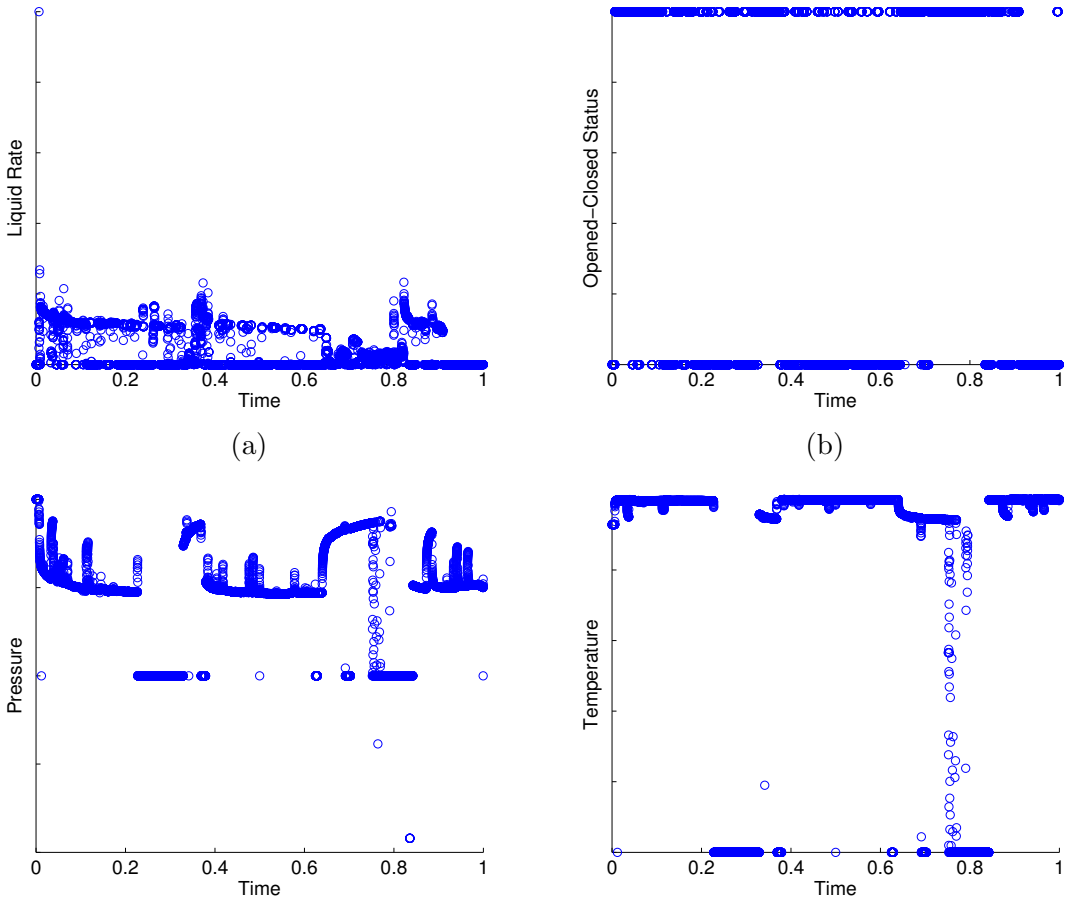


Figure 6.1: Original data of liquid rate, pressure, temperature and opened-closed status.

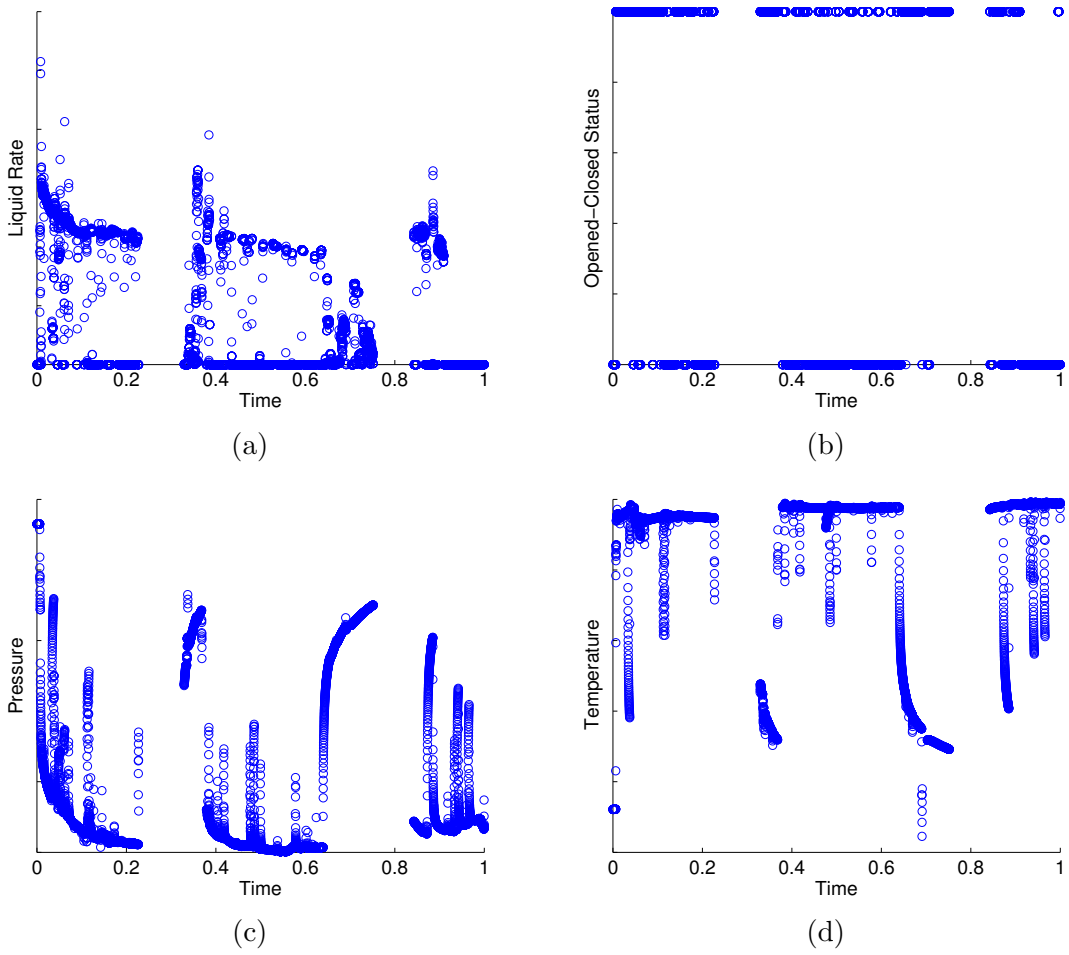


Figure 6.2: Liquid rate, pressure, temperature and opened-closed status after data preprocessing.

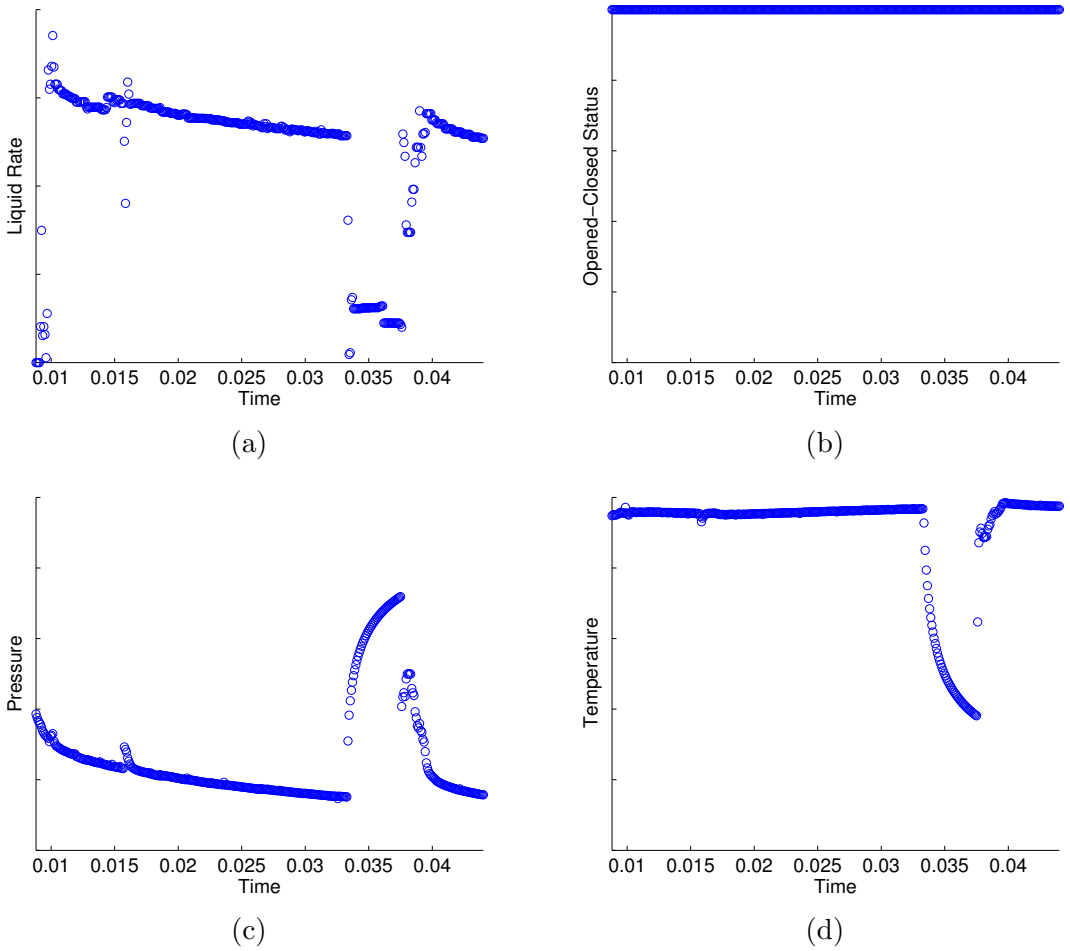


Figure 6.3: Zoom in on data in time period 0.009-0.044. This period of data were selected as training data for flow rate reconstruction.

6.1.2 Results and Analysis

In the training step, features were constructed following Equation 2.30, then the linear approach machine learning was applied to learn the mapping from pressure data to rate data shown in Figure 6.4(a) and (b). After training, the full pressure history in Figure 6.4(c) was used as input to generate the flow rate prediction. That was a challenging test of the method because the time scale of the test data was about 30 times that of the training data. As shown in Figure 6.4(d), the algorithm rejected most of the zero rates as shut-ins except for the period around $t = 0.7$. Importantly, the reconstructed flow rate matched well to the nonzero rate measurements, which improved our confidence in the prediction. Compared with the ambiguous original rate data containing many zeros, the reconstructed flow rate provides a better description of the operation scheme.

6.2 Field Example 2: Wellbore Modeling and Multilwell Analysis

6.2.1 Wellbore Modeling

Here the feature-based machine learning approach was applied on the data of Well A. The well was producing mainly in single phase, with negligible amount of water. The data contain 309 averaged measurements. Figure 6.5 shows the three data attributes of interest: wellhead pressure (WHP), downhole pressure (DHP) and total surface flow rate (Q). The values on the axes are covered, the time scale was multiple years. From the shapes of WHP and DHP profiles, we can easily observe that a correlation

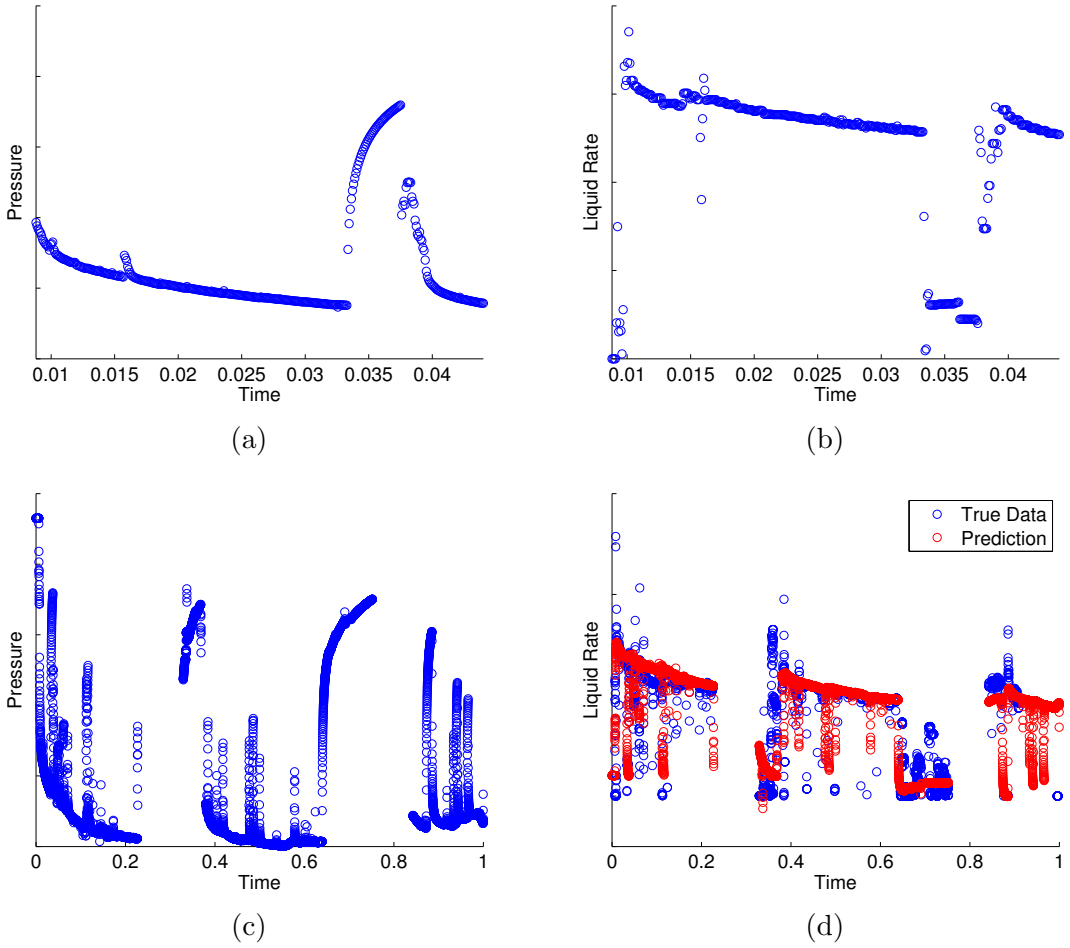
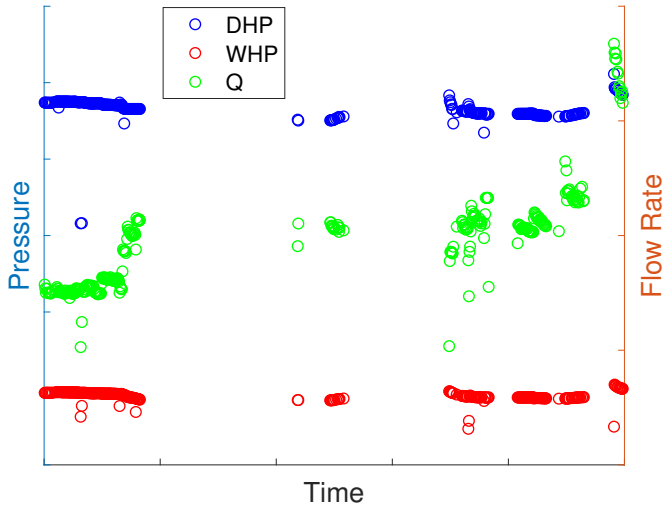


Figure 6.4: Train the machine learning model on pressure-rate data of the short period shown in (a) and (b), then reconstruct the rate profile based on pressure input in (c). The predicted rate captures the trend of non-zero rate measurements. The rate prediction helps to distinguish the zero rate caused by shut-in (around $t = 0.7$) and the zero rate caused by well not aligned to the separator (most of the zeros).

exists.



(a)

Figure 6.5: Downhole pressure (DHP), wellhead pressure (WHP) and total surface flow rate (Q) of Well A.

We know from physics that wellhead pressure and downhole pressure are correlated by wellbore pressure loss. Thus an intuition would be to model downhole pressure using polynomials of wellhead pressure (and vice versa). In other words, $y = DHP$, $x = [WHP, WHP^2, ..., WHP^m]$. Lasso method introduced in Chapter 2 was applied to learn the coefficients θ by minimizing $\|X\theta - y\|_2^2 + \lambda\|\theta\|_1$.

The data were split into two halves. The first half was used for training to learn θ. The second half was used to test the learning performance, by feeding the algorithm with new x and comparing the predicted $\hat{y}$ against the true y. The result is shown in Figure 6.6(a), where we observe a fairly good prediction but a discrepancy in the training part. We further added flow rate features (Tian and Horne, 2015a) into x to

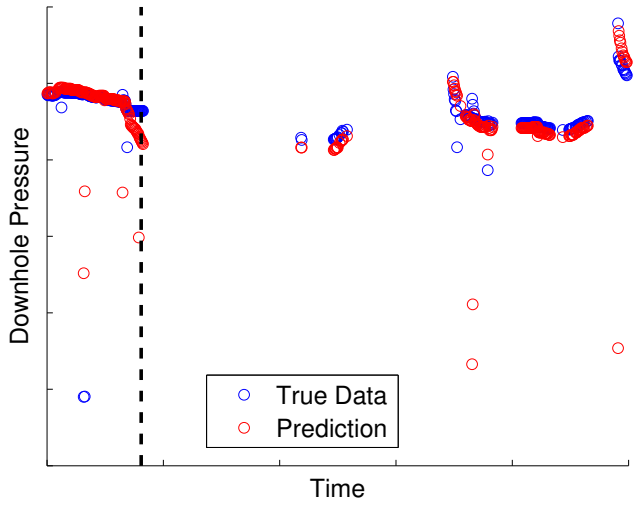
test the leaning performance by utilizing both wellhead pressure and rate information:

$$x = [WHP, WHP^2, ..., WHP^m, \sum \Delta q, \sum \Delta q \log \Delta t, \sum \Delta q \Delta t, \sum \frac{\Delta q}{\Delta t}]. \quad (6.1)$$

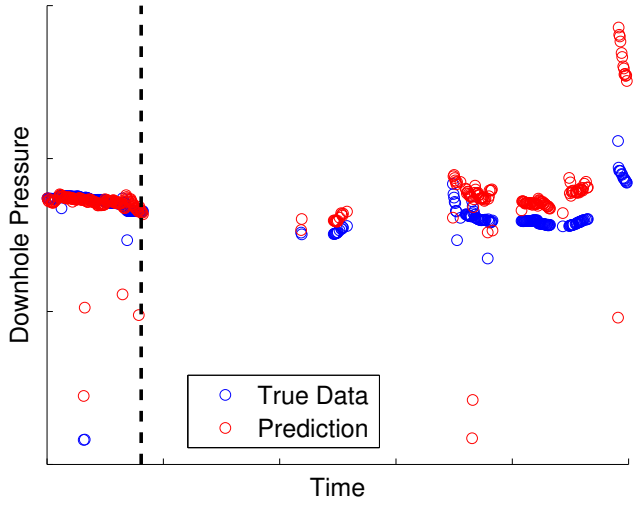
Figure 6.6(b) shows the result based on the expanded features. By adding flow rate features, we obtained a better match to the training data, but a prediction that obviously deviated from the true data. This validates an important rule in machine learning: higher model complexity always results in better fit on training data, but may hurt the prediction accuracy of the model. This is a clear example of overfitting.

Next more data were added into the training set to see how that could impact the learning performance. 60% of the data were used for training and the remaining data were used for testing. As shown in Figure 6.7, the prediction accuracy was improved for both Case (a) using wellhead pressure feature only and Case (b) using wellhead pressure plus flow rate features. The improvements can be explained by the fact that this time the algorithm had more training data to learn the true model from. The improvement of Case (b) was even greater compared to Case (a), but an obvious discrepancy still existed at the end of the data.

Overall, lasso was shown to be promising to model downhole pressure using other types of measurements, especially wellhead pressure. Therefore it would be interesting to apply lasso to figure out the normal relationship between downhole and wellhead measurements, and use it as a well alert to identify abnormal behaviors as real time data are fed in.

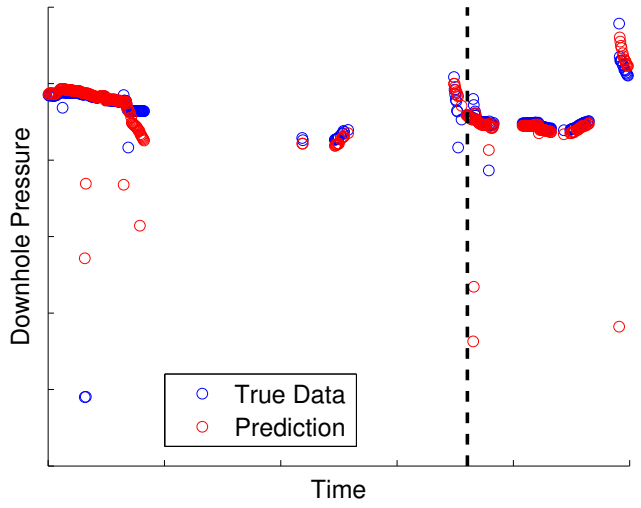


(a)

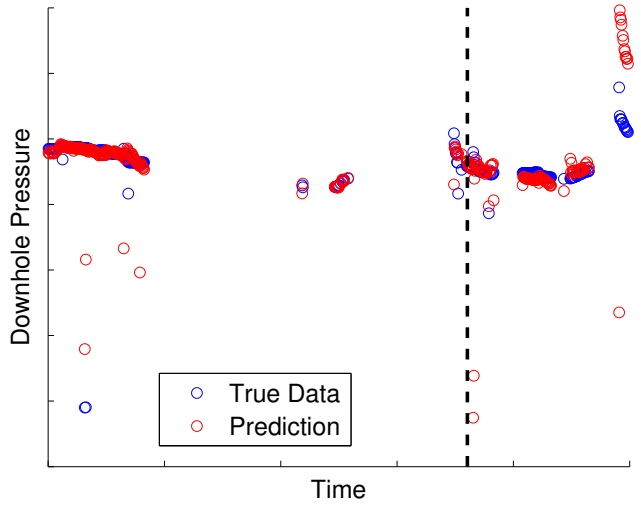


(b)

Figure 6.6: Modeling downhole pressure based on (a) wellhead pressure only (b) both wellhead pressure and surface flow rate. The dashed line splits the data into training set (left) and test set (right). (b) shows a better match on training set, but less accurate prediction on the test set.



(a)



(b)

Figure 6.7: Modeling downhole pressure based on (a) wellhead pressure only (b) both wellhead pressure and surface flow rate. Compared to Figure 6.6, more data were used for training, both (a) and (b) show more accurate predictions.

6.2.2 Multiwell Analysis

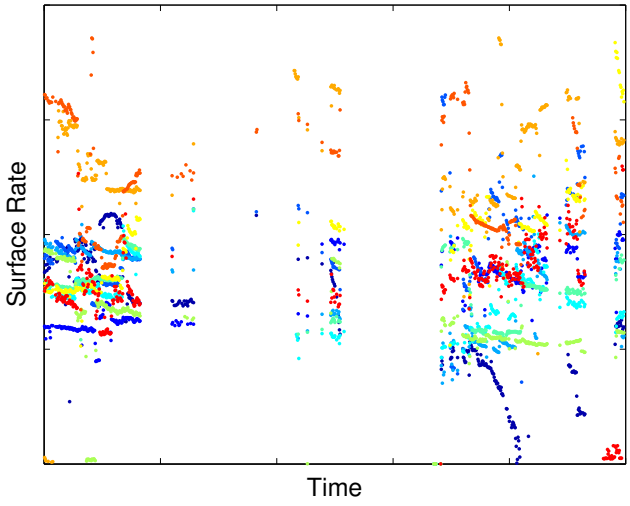
Next we moved on to apply machine learning for the similar analysis, but considering the measurements of multiple wells in the same field. The dataset contains the production history of 31 wells, most of which were producing with low water cut. Here the downhole pressure was modeled based on liquid rate only, because the problem became trivial once wellhead pressure was used for the modeling, as we have seen in the previous section.

The data attributes of interest were downhole pressure (DHP) and total surface flow rate (Q). The number of data points for each well varies, for instance, one well has 385 measured data points, while another well has only one measured data point. Thus we first selected 11 wells that have more than 300 data points. The flow rate and downhole pressure of those 11 wells are shown in Figure 6.8.

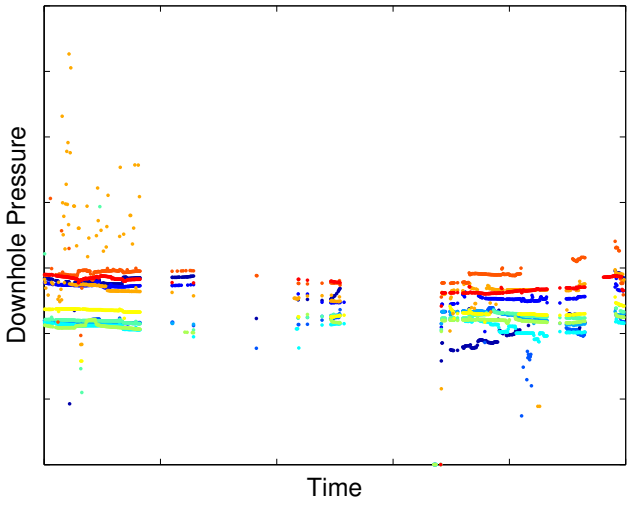
Noise and gaps are observed in the data. To clean the data, we manually removed the data of Well D, whose downhole pressures are shown as the noisy orange dots in Figure 6.8(b). The data of the remaining ten wells are plotted in Figure 6.10.

We further manipulated the data by identifying 196 common measurements at the same timestamps across the ten selected wells, and removing all other measurements, as suggested by Tian and Horne (2015b). After that, the data were ready for machine learning analysis (Figure 6.10).

We started by applying machine learning to model the downhole pressure of a well based on the surface rate of that well itself. Again lasso method was applied to avoid overfitting. 68% of the data were used for training and the remaining data were used for testing. Due to the skewed data distribution, the training data all lie in the first 1/5 of time, which makes the learning task challenging. The results of Well B and

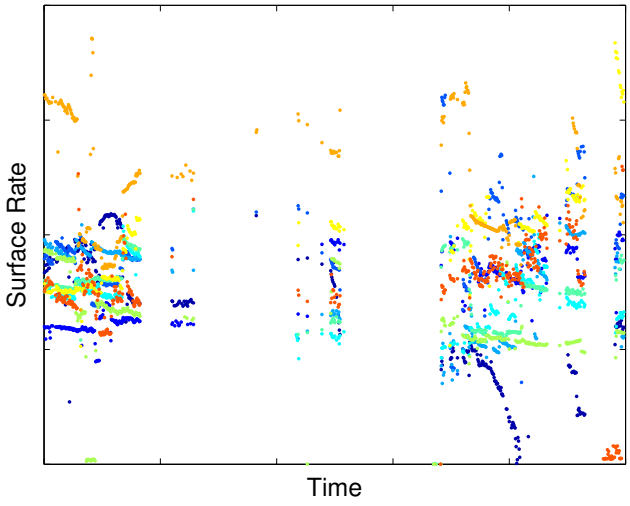


(a)

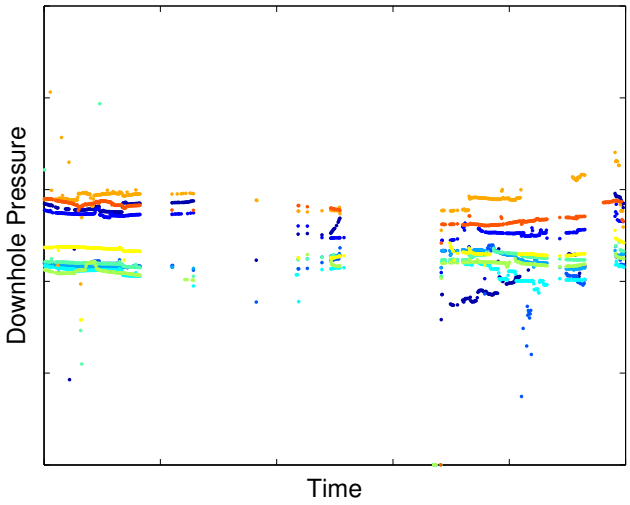


(b)

Figure 6.8: (a) Flow rate and (b) downhole pressure of the 11 wells that have more than 300 data points. Colors represent different wells. The noisy orange dots in (b) are downhole pressure data of Well D.

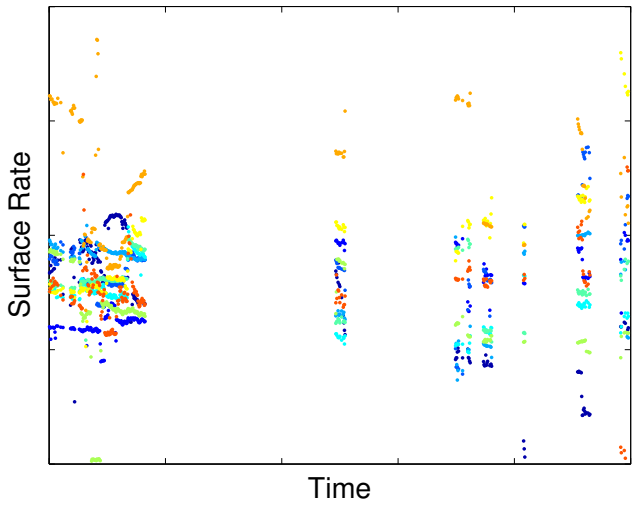


(a)

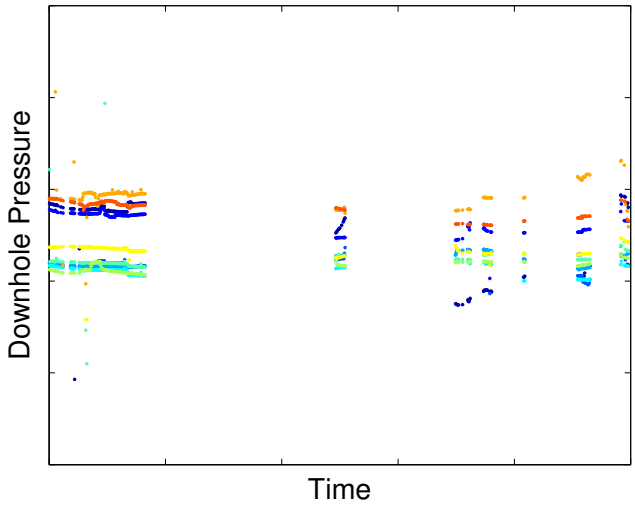


(b)

Figure 6.9: (a) Flow rate and (b) downhole pressure of the 10 wells that have more than 300 data points, after removing Well D.



(a)



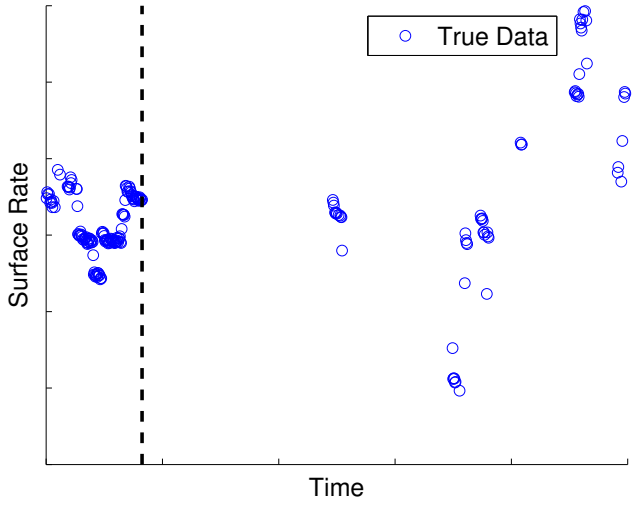
(b)

Figure 6.10: (a) Flow rate and (b) downhole pressure data after data preprocessing. The data contain the measurements on the same 196 timestamps across the ten selected wells.

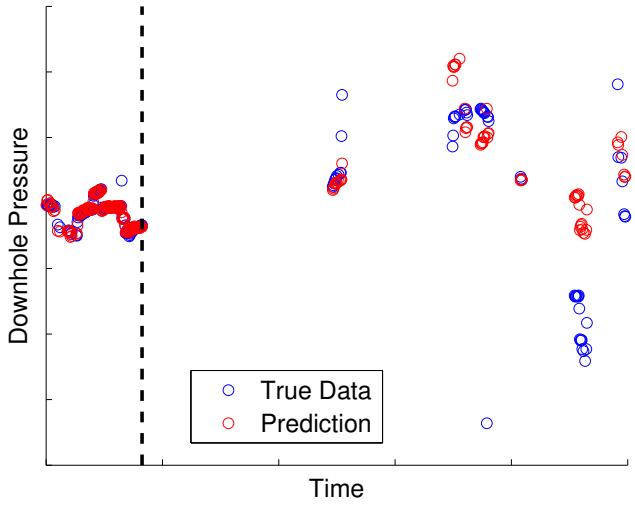
Well C are shown in Figure 6.11 and Figure 6.12 respectively.

Both figures show that machine learning had a modest fit to the training downhole pressure, and captured the trend of pressure in the testing period, despite the noise and gaps in the data. Some of the details in the mismatch are quite interesting. For instance, the downhole pressure of Well C (6.12(b)) shows an obvious mismatch between the prediction and true data around 3/4 of time. The true data contain a sharp pressure drop, while the flow rate almost remains constant during that period. Thus the lasso prediction based on flow rate was not able to capture that pressure drop.

One possible explanation of the pressure drop is the influence of flow rate changes from other wells nearby. That led us to apply multiwell machine learning (Tian and Horne, 2015b) for further investigation. Multiwell machine learning expands the feature space to include the flow rate features of all the ten wells, and model regularization (e.g. lasso method) is always required because overfitting can easily occur with a feature space that is ten times larger. The pressure predictions using multiwell machine learning on Well B and Well C are shown in Figure 6.13. For Well B, the predicted pressure was almost the same as in Figure 6.11(b), which indicated that Well B were not influenced significantly by other wells. For Well C, the fitting on training data was improved compared to Figure 6.12(b), but the prediction further deviated from the true pressure. However, we did observe a pressure drop in the predicted pressure at 3/4 of time, which can be an evidence of multiwell interactions. Further analysis is needed to have deeper understanding about the interactions among the wells in this dataset.

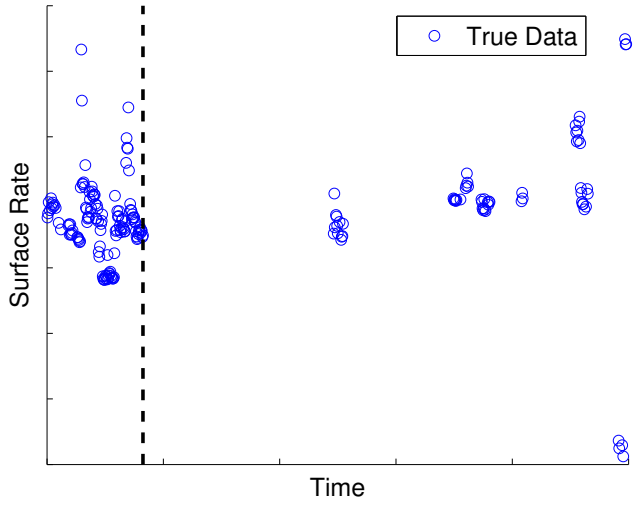


(a)

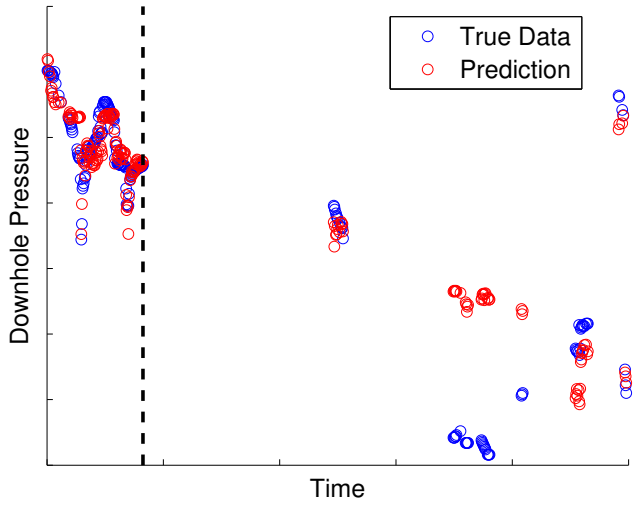


(b)

Figure 6.11: Well B: modeling downhole pressure based on surface rate of itself.

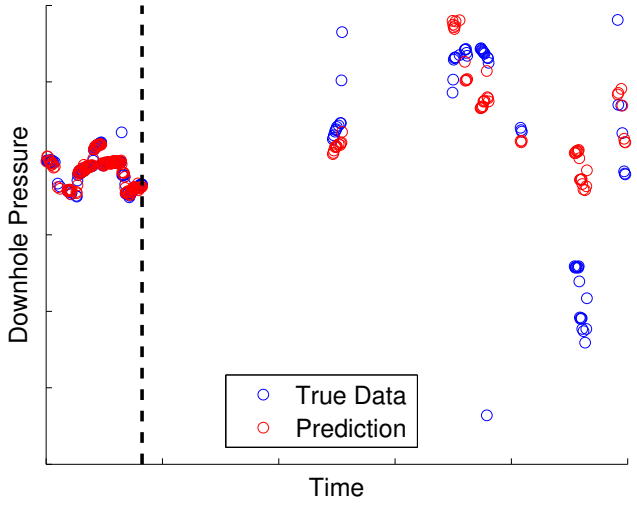


(a)

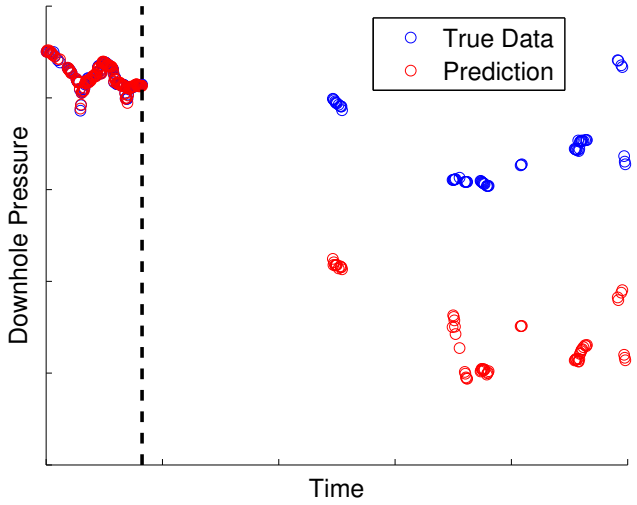


(b)

Figure 6.12: Well C: modeling downhole pressure based on surface rate of itself.



(a)



(b)

Figure 6.13: Modeling downhole pressure of (a) Well B and (b) Well C based on surface rate of tens wells shown in Figure 6.10(a).

6.3 Summary

In this chapter, we discussed the usages of feature-based machine learning in flow rate reconstruction and downhole pressure modeling with two field examples. Because of its effectiveness, the feature-based leaning approach developed in this study has been adopted by the industry to help analyzing pressure and flow rate measured in the field. For instance, Sankaran et al. (2017) utilized our approach as part of their production surveillance solutions. In their work, the patterns extracted from pressure-rate data were used to predict the bottomhole pressure for a virtual shut-in test, the predicted pressure was then used to calculate the productivity index (PI). Figure 6.14 shows a good match between the machine learning based PI (red) and the PI obtained from actual shut-ins (blue), although the PI calculated by machine learning was somewhat noisy. In their application, the methodology developed in this study helped offset the need for shut-ins.

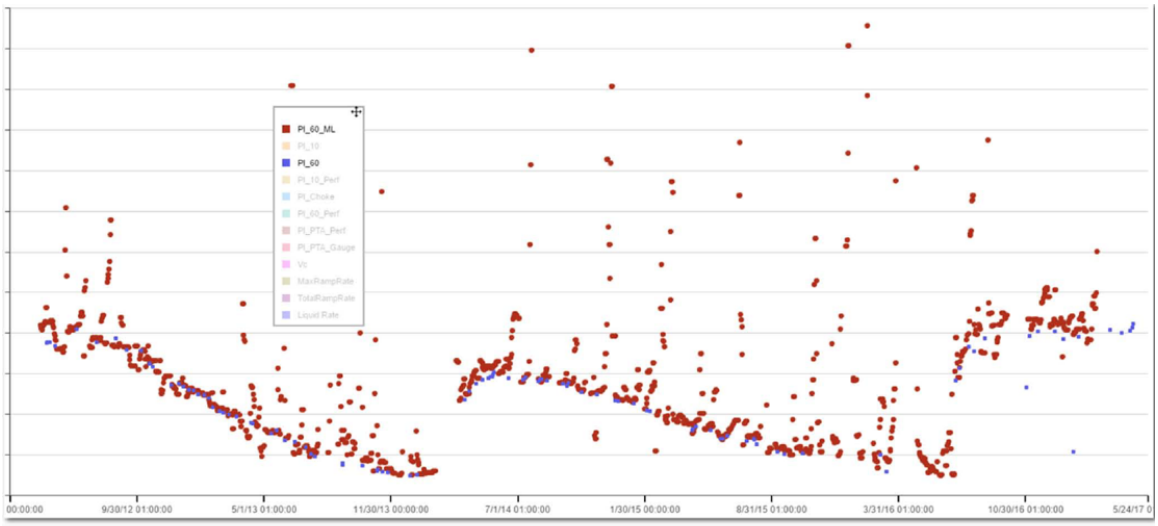


Figure 6.14: Machine learning based productivity index (PI) calculation offsets need for shut-ins (from Sankaran et al., 2017). PI calculated by machine learning (red) captures well performance trends quite well compared to actual shut-ins (blue).

Chapter 7

Conclusions and Future Work

Conclusions

In this study, we applied several feature-based machine learning and deep learning techniques to interpret flow rate, pressure and temperature data from PDGs. The results demonstrate that feature-based machine learning and deep learning are promising for PDG data interpretation. The main conclusions of this study are summarized as follows:

1. Feature-based machine learning and deep learning were shown effective to extract the patterns containing the reservoir information from PDG data. The extracted patterns were demonstrated to be useful in three ways: one was deconvolution by predicting the pressure corresponding to a constant rate history; the other two were to predict pressure in the near future with given rate control, and vice versa.

2. Three feature-based machine learning techniques, namely, linear approach, the kernel method and kernel ridge regression were explored in this study. First linear approach was proved to learn with the same features as the convolution kernel method

(Liu and Horne, 2013a, 2013b), while it required much lower computational cost. The linear approach was used for pressure-rate deconvolution and it outperformed two existing deconvolution methods when noise or outliers were contained in the data. Next the kernel method and kernel ridge regression were utilized to adjust the machine learning model capability and to model the pressure behaviors covering early-transient until late-transient periods. Additional machine learning models were developed for the problems of rate reconstruction and multiwell testing.

3. Deep learning techniques were explored in this study with a focus on NARX models. It was shown that NARX provided a feature-free alternative to traditional machine learning methods to learn the useful patterns from PDG data. NARX also identified the reservoir models correctly by pressure-rate deconvolution, although it was more sensitive to noise compared with feature-based machine learning. The ability to learn on raw data makes NARX a preferred approach when feature handcrafting is difficult, e.g. limited knowledge of the physics.

4. Feature-based machine learning and deep learning have their own advantages and disadvantages, and accordingly should be selected based on the problem of interest. If we have enough knowledge to properly design the features, feature-based machine learning leads to very reliable and interpretable results. Examples include pressure transient analysis and rate reconstruction. On the other hand, when our knowledge is limited for feature handcrafting, deep learning provides an alternate solution, although thought and effort are needed for the neural network design.

Future Work

Some possible directions for future research are as follows:

1. Full-physics reservoir models: The reservoir models used in this study followed the common assumptions of typical PDG data interpretation: single-phase radial flow in a homogeneous reservoir. Moving from single-well data analysis to multiwell testing, requires more complex reservoir models to be considered. In Case 13 Multiwell Pressure Prediction, the machine learning model was applied on modeling the pressure with the presence of two phase flow, by replacing oil rate with total liquid rate in the features. The results look promising. Future research is needed to extend the machine learning model to work on data contain complex physics such as reservoir heterogeneity and multiphase flow.

2. Temperature and rate transient analysis using NARX: Currently the main focus of NARX applications is still pressure-rate deconvolution. In Chapter 4, NARX was demonstrated to have the potential of modeling temperature based on flow rate and vise versa. Given NARX's advantage in not requiring feature handcrafting, it is very promising to apply NARX for temperature and rate transient analysis where extracting features from their analytical solutions (if any) is difficult.

3. Unconventional data analysis: This study is focused on interpreting the data obtained from conventional wells. Unconventional data contain different characteristics than conventional data, and some physics drive the pressure and rate transient behaviors during unconventional production remain unknown. Because deep learning techniques do not require those physics for feature engineering, it would be very interesting to apply deep learning techniques such as NARX to analyze unconventional data by asking the neural network to learn the physics behind the data. Future study is needed to appropriately design the neural network architecture in order to analyze unconventional data.

4. Integrating other sources of information: The data analyzed in the current study were time series data, more specifically, rate-pressure-temperature measurements along with time. Other sources of information such as spatial information can be also useful, and it is possible to integrate those information into the machine learning and deep learning models. However, integrating different types of data is not a trivial task. For instance, RNNs are only suitable for modeling sequential signals, and the feedback mechanisms applied on spatial information does not make any sense. Further study is needed in this direction.

Appendix A

Inferring Interwell Connectivity Using Production Data

This chapter summarizes our work on the investigation into inferring interwell connectivity based on production data. It is discussed as an appendix because the topic is a comparative alternative to the machine learning approaches described in Chapter 2. Instead, this part of the study investigated the use of more traditional correlation approaches. Part of the contents in this appendix is from paper SPE-181556-MS (Tian and Horne, 2016).

A.1 Introduction

Production and injection rates are the most accessible data in many projects. Interwell connectivity analysis using production and injection rates can reveal how the production is affected by the injection, and can be used to improve the injection strategy.

In this work, a modified Pearsons correlation coefficient (MPCC) was proposed as a new connectivity definition based on the correlation between injection and production histories. The proposed methodologies were applied on both synthetic and real data sets. The connectivity was shown to be able to capture the injectors influence on producers, and was verified by comparing with the true reservoir model and results from tracer tests. The target injectors with high influence on production were identified easily as the ones connected to multiple producers, information that can be used to guide better field management.

A.2 Literature Review

One approach for connectivity analysis uses Spearmans rank correlation coefficient (Corder and Foreman, 2014) to analyze the correlation between the production and injection rates of each producer/injector pair. Spearmans rank correlation coefficient measures the degree of monotonic dependence between two variables. Intuitively, production increases with higher injection rates, therefore a high Spearmans rank correlation coefficient indicates a strong connection between the producer/injector pair. Heffer et al. (1997) studied the interwell connectivity using Spearmans rank correlation coefficient and related the interwell connectivity to geomechanics. Soeriawinata and Kelkar (1999) extended the connectivity analysis based on Spearmans rank correlation coefficient to account for the superposition effects from multiple injectors. As a model-free method, although Spearmans rank correlation coefficient can give an initial estimate of interwell connectivity only based on production and injection rates, the estimate may violate the reservoir geology, as we will see in the examples Case 1 and Case 2 later in this appendix. The artificial tuning required in

those works (e.g. selection of time lag) is also not preferred for automatic analysis.

There is another class of model-based interwell connectivity analysis. Albertoni and Lake (2003) utilized a linear model to correlate the production rate of a producer and the injection rates of all injectors. The connectivity was then estimated by multivariate linear regression or balanced multivariate linear regression. Yousef et al. (2006) developed a more complete model to account for compressibility and transmissibility effects, where both connectivity and degree of fluid storage can be calculated. Although promising, those model-based methods require the assumption of a model in advance, which is actually what we want to be learned from the data.

In this work, following the Spearmans rank correlation coefficient approach, we developed a model-free interwell connectivity estimation based on modified Pearsons correlation coefficient (MPCC). MPCC was applied on synthetic and real field cases, and validated by comparing with reservoir geology, tracer test, etc. The estimated connectivity can be further used to build a connectivity network as an abstract reservoir representation on which a series of network analysis can be performed.

A.3 Methodologies

Pearsons correlation coefficient (PCC) characterizes the degree of linear dependence between two variables (Corder and Foreman, 2014). For production rate $\{q_1^P, ..., q_n^P\}$ and injection rate $\{q_1^I, ..., q_n^I\}$ each containing n records, PCC is given by:

$$r = \frac{\sum_{i=1}^{n}(q_i^P - \overline{q^P})(q_i^I - \overline{q^I})}{\sqrt{\sum_{i=1}^{n}(q_i^P - \overline{q^P})^2}\sqrt{\sum_{i=1}^{n}(q_i^I - \overline{q^I})^2}}, \tag{A.1}$$

where $\overline{q^P} = \frac{1}{n}\sum_{i=1}^{n} q_i^P$, $\overline{q^I} = \frac{1}{n}\sum_{i=1}^{n} q_i^I$.

However, the production and injection rates of a producer/injector pair are not likely to be linearly correlated, thus PCC may fail for reservoir connectivity analysis. However the formula in Equation A.1 still provides us some insights. In Equation A.1, the numerator on the right-hand-side measures the degree to which the production and injection rates deviate concurrently from their own mean values, and the denominator is used for normalization. Ideally, a strong connectivity should also make the production and injection rates vary concurrently, but not necessarily from the mean values. That idea motivates us to replace the mean values in Equation A.1 with reference state rates to obtain the modified Pearsons correlation coefficient (MPCC):

$$r' = -\frac{\sum_{i=1}^{n}(q_i^P - q^{Pref})(q_i^I - q^{Iref})}{\sqrt{\sum_{i=1}^{n}(q_i^P - q^{Pref})^2}\sqrt{\sum_{i=1}^{n}(q_i^I - q^{Iref})^2}}, \tag{A.2}$$

where q^{Pref} and q^{Iref} are the reference state rates of the producer/injector pair. An intuitive choice of the reference state is the state with no injection, where q^{Pref} is given by decline curve and $q^{Iref} = 0$. In that case, the numerator on the right-hand-side of Equation A.2 measures the concurrent changes in production and injection rates caused by injection, which is what we need to characterize the interwell connectivity.

The negative sign in Equation A.2 represents the opposite sign of production/injection rate change, for instance, production rate increases from 1000 STB/d to 2000 STB/d, while injection rate increases from -1000 STB/d to -2000 STB/d. $r' = 1$ is achieved between a totally connected producer/injector pair under mass balance condition, and $r' = 0$ indicates that production is not affected by injection at all.

A.4 Results

Case 1: Synthetic Homogeneous Field With Fault

We first tested MPCC on a synthetic homogeneous field with two producers, two injectors and an impermeable fault (Figure A.1). Without the fault, connectivity is only determined by distance, therefore higher connectivity is expected between P1 and the two injectors compared to P2. Adding the fault should reduce the connectivity between P1/I1 and P2/I2 pairs.

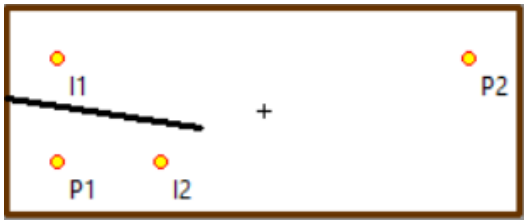


Figure A.1: Homogeneous field with two producers, two injectors and an impermeable fault. The fault reduces the connectivity between P1/I1 and P2/I2 pairs.

The simulation period was five years, with two injectors injecting at 1000 STB/d in two separate periods. The two producers operated at fixed bottom-hole pressures (BHPs). Figure A.2 shows the production/injection rates of each well in the simulation (blue), as well as their own reference states rates corresponding to the situation without injection in the field (red). Here the reference state rates were obtained by running another simulation with two injectors shut-in, but they can be also obtained using decline curve or other approximations based on rate profiles with injection.

By taking the difference between the blue and red curves in Equation A.2, MPCC essentially measures the magnitude of production rate changes caused by injection. For instance, if we focus on P1 production rate in Figure A.2, the rate increase corresponding to I1 injection (1-2 year) is lower compared to the one corresponding

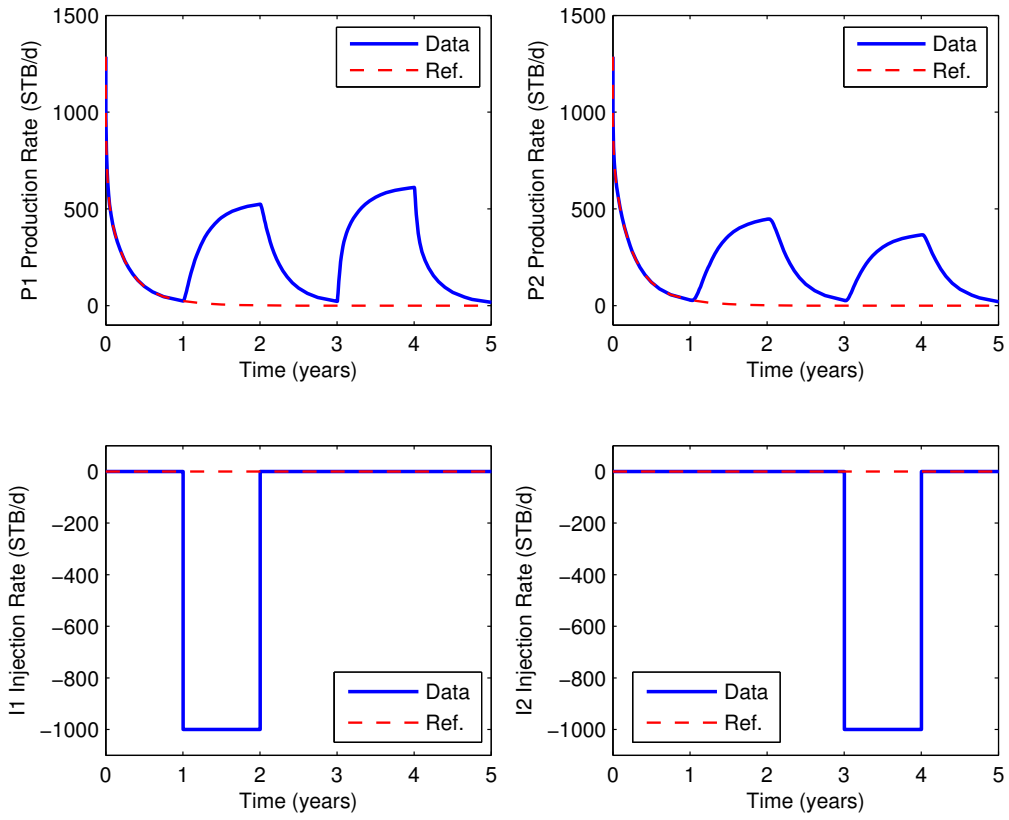


Figure A.2: Rates of each producer/injector with injection (blue solid line) and without injection (red dash line).

to I2 injection (3-4 year). That indicates a lower connectivity (Table A.1) between P1/I1 (0.2092) than P1/I2 (0.3015) and honors the reservoir geology with a fault in between P1/I1. Similarly, P2/I1 has a connectivity of 0.1605, which is higher than the connectivity of 0.1319 for P2/I2. From Table A.1, we also observe that P1 has higher connectivity to both injectors than P2, which can be explained by the shorter distance between P1 and two injectors. However none of those features were honored by Spearmans rank correlation coefficient, as shown in the values in brackets in Table A.1.

Table A.1: Summary of connectivity of Case 1 by MPCC and Spearmans correlation (in brackets).

	P1	P2
I1	0.2092 (0.3106)	0.1605 (0.2940)
I2	0.3015 (0.0984)	0.1319 (0.3963)

Case 2: Synthetic Heterogeneous Field

Next MPCC was tested on a synthetic heterogeneous field with permeability generated using the sequential Gaussian simulation algorithm (Figure A.3). An injector sits in the center of the reservoir and four producers are located at the corners. Figure A.3 shows a high permeability zone across P3, I1 and P2. Thus we anticipate P2 and P3 to have higher connectivity with I1 compared to P1 and P4.

The simulation period was one and a half years, with I1 injecting at 3000 STB/d from half year to one year. The production rate of each producer (blue) and the corresponding reference state rate (red) are shown in Figure A.4. Again the reference state rates were obtained by running another simulation with zero injection rate. We can clearly observe a higher rate increase caused by injection (0.5-1 year) for P2 and P3 compared to P1 and P4, which leads to higher connectivity between P2/I1 and

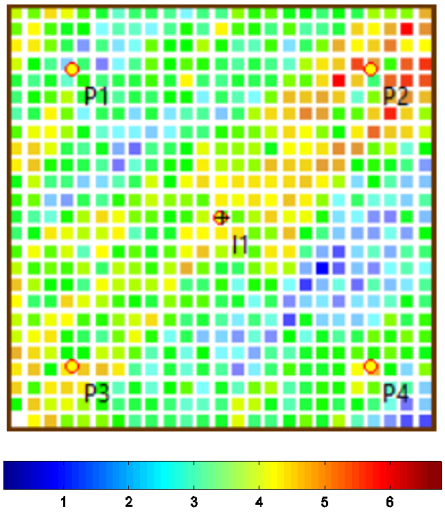


Figure A.3: Log permeability map (md) for synthetic field with four producers and one injector. P2 and P3 are expected to have higher connectivity with I1 compared to P1 and P4.

P3/I1 than P1/I1 and P4/I1 (Table A.2). The connectivity estimates based on MPCC honor the permeability heterogeneity shown in Figure A.3, while the estimates based on Spearmans correlation do not.

Table A.2: Summary of connectivity of Case 2 by MPCC and Spearmans correlation (in brackets).

	P1	P2	P3	P4
I1	0.1468 (0.3631)	0.1619 (0.3135)	0.1561 (0.3105)	0.1491 (0.3158)

Case 3: Palinpinon-I Geothermal Field

In 2007, Horne and Szucs used the alternating conditional expectation (ACE) method to relate the chloride productions and injection rates to estimate connection indices for different producer/injector pairs in the Palinpinon-I Geothermal Field. In this work we applied MPCC on the same data set used by Horne and Szucs (2007) for connectivity analysis.

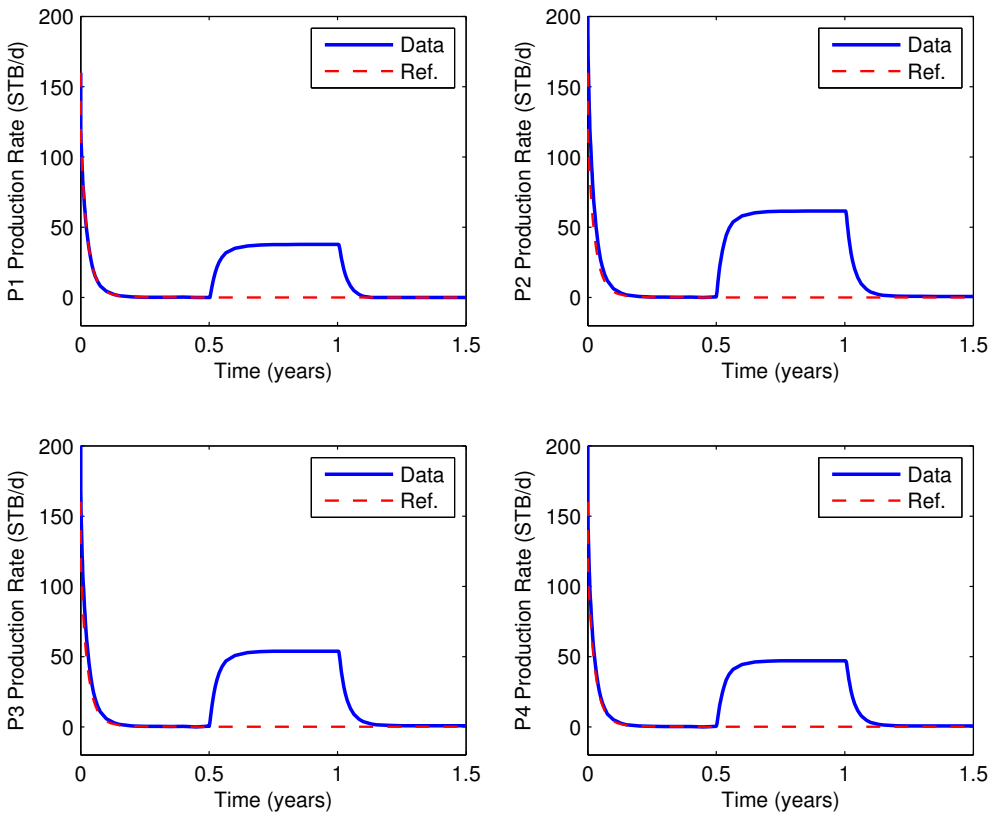
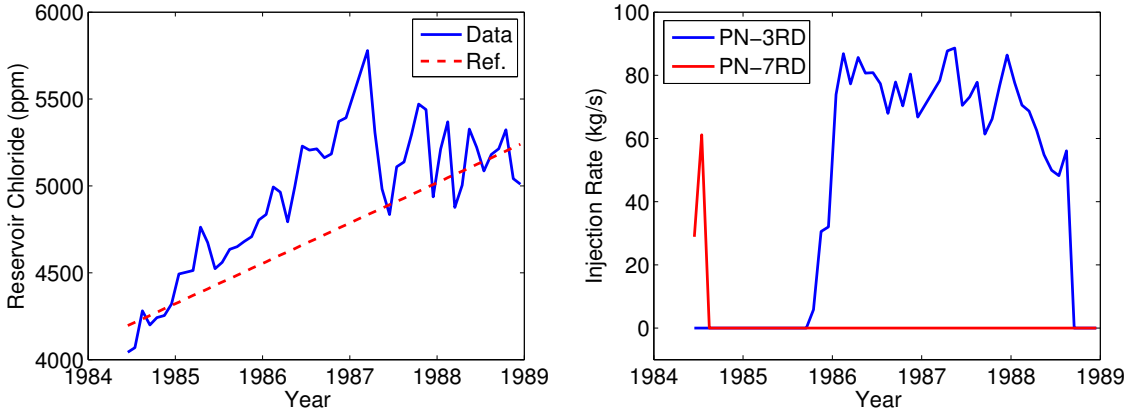


Figure A.4: Rates of each producer with injection (blue solid line) and without injection (red dash line).

To account for the chloride production, a slight modification is required in Equation A.2 by replacing production rate with chloride production rate. Figure A.5(a) shows the chloride production rate of well PN30D during a period over four years (blue). A linear trend line (red) was used to represent the reference state chloride production with no injection. Interestingly, here the reference state rate in MPCC is quite similar to the idea of extracting a time-dependence element, as implemented by Horne and Szucs (2007). After specifying the reference state rate, we applied MPCC to calculate the connectivity between well PN30D and nine injectors, namely, PN-1RD, PN-2RD, ..., PN-9RD. The injector with the highest and lowest connectivity with PN30D were PN-3RD and PN-7RD respectively. Figure A.5(b) shows the injection profiles of those two injectors, which can be used to explain their different connectivity values with PN30D. PN-7RD was shut in for most of the time, which makes it unlikely to cause the fluctuations in PN30D chloride production, while the varying pattern of PN-3RD injection shows similarity to the PN30D chloride production.

The connectivity values between PN30D and all nine producers by MPCC (blue) and ACE (red) are shown in Figure A.6. Both methods identified the same injectors with high connectivity (e.g. PN-3RD, PN-5RD) and the ones with low connectivity (e.g. PN-7RD, PN-9RD). The connectivity rankings among nine injectors are almost the same by MPCC and ACE, although the absolute connectivity values are different (MPCC subtracts linear trend by definition, while ACE assigns high connection indices for the time-dependence element). Horne and Szucs (2007) showed in their work that the connectivity analysis by ACE was consistent with the result of tracer tests. Therefore the consistent results from MPCC and ACE give us confidence in



(a) (b)

Figure A.5: (a) Chloride production (blue solid line) and reference state (red dash line) of production well PN30D. (b) Injection rates of the injector with highest connectivity (PN-3RD, blue) and lowest connectivity (PN-7RD, red) with PN30D. Injection profile of PN-3RD shows more similarity with chloride production compared to PN-7RD.

the correctness of the MPCC method.

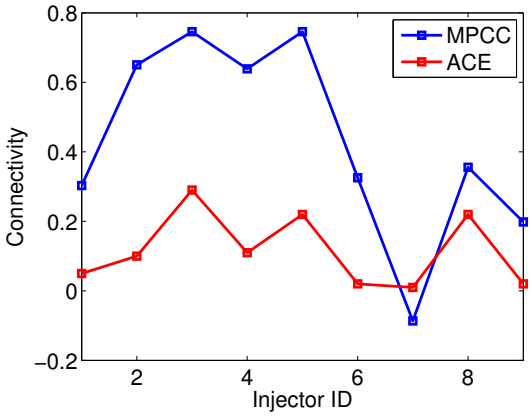


Figure A.6: Summary of connectivity between production well PN30D and nine injectors estimated by MPCC (blue) and ACE (red). The horizontal axis represents the well ID (PN-iRD, i=1,..,9) of nine injectors. Similar ranking among the nine injectors is observed between MPCC and ACE, although the absolute connectivity values are different.

Case 4: Stanford VI Reservoir

The Stanford VI reservoir is a standard model with exhaustive data for the general purpose of testing reservoir modeling algorithms (Lee and Mukerji, 2012). The reservoir is a three-layer prograding fluvial channel system, with 31 oil producers (eight of which converted to injectors after reaching 0.5 water cut) and 15 water injectors operating during 30 years. The location maps of the producers and injectors are shown in Figure A.7.

Because the producers operated at fixed total liquid rate, we modified q^P in Equation A.2 to be water production rate (WPR) to reflect the reservoir response to injection. Figure A.8 shows the water injection rate (WIR) of I32 and WPR of two producers P11 and P31, which respectively has the highest and lowest connectivity with I32 by MPCC. Water breakthrough was found in P11 immediately after injection began in I32, and water production increased as the injection continued. This indicates a strong connectivity between I32 and P11. On the other hand for P31, its WPR was hardly influenced by WIR in I32, which indicates a lack of connection between those two wells. The connectivity results by MPCC were compared with tracer simulation, where tracer was injected with 1.0 concentration in I32. By the end of 30-year simulation, we observed a tracer production rate of 1142 STB/d at P11 and 0 STB/d at P31. The connectivity between other producers and I32 estimated by MPCC were also consistent with the tracer tests. One possible reason of the difference in connectivity comes from the well locations (Figure A.7). I32 is much closer to P11 than P31, which will likely make the responses at I32 and P11 more correlated.

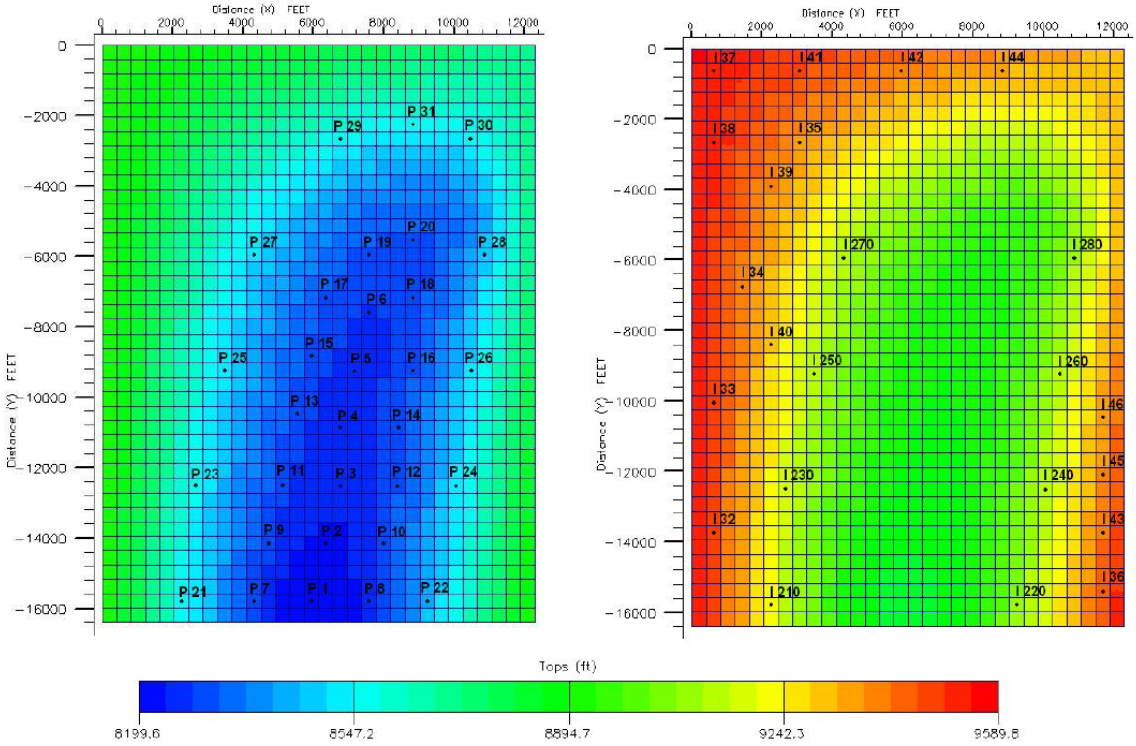


Figure A.7: Location maps of producers and injectors. The color represents horizon top depth (from Lee and Mukerji, 2012).

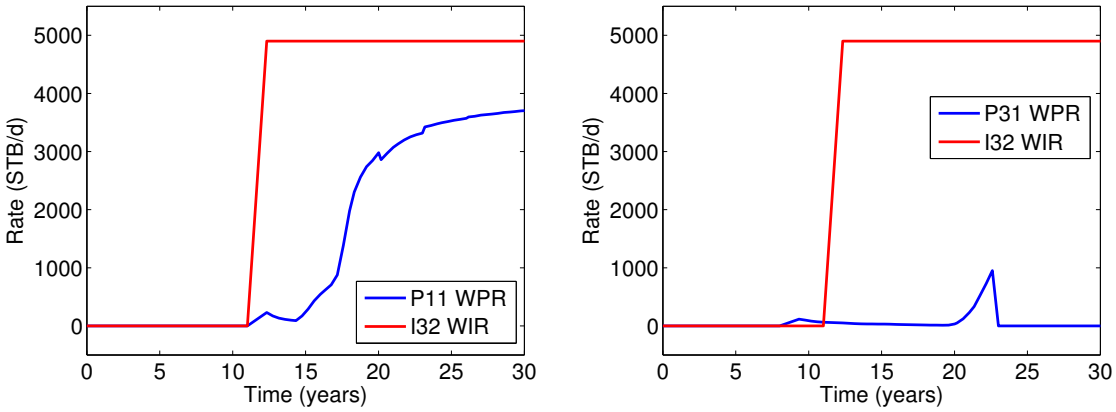


Figure A.8: Water injection rate (WIR) of I32, water production rate of (WPR) of P11 and P31. P11 and P31 are the producers with the highest and lowest connectivity with I32 respectively. The WPR profile of P11 shows more similarity with I32 WIR compared to P31.

A.5 Summary

- An interwell connectivity formula was developed based on the modified Pearsons correlation coefficient. The new approach was tested on several synthetic and real field cases. The estimated connectivity showed consistency with the reservoir geology and tracer test.

- Given the connectivity calculated by the modified Pearsons correlation coefficient, a connectivity based network can be built as an abstract reservoir representation. A series of network analyses can be performed on the established network, e.g. reservoir compartmentalization by detecting the strongly connected clusters in the network. Further efforts are needed to perform network analysis for better reservoir characterization and waterflooding operation.

Nomenclature

α	learning rate
β	parameters (coefficients) for features $\phi(x)$
Δp	pressure change
Δp_0	pressure change caused by constant rate production
ΔT	temperature change
$\hat{y}$	actual output generated by neural network
λ	tuning parameter
$\overline{q^I}$	mean of injection rate
$\overline{q^P}$	mean of production rate
$\phi(x)$	feature mapping on features x
τ	time
Θ	vector containing all the model parameters
θ	parameters (coefficients) for features x

b	bias term
c	constant inside the exponential integral function
d_x	index of the earliest delay of input
d_y	index of the earliest delay of output
f, g	activation function
H	weights of hidden layer
h	hidden layer
J	objective function for optimization
$K(x, z)$	kernel on two vectors x and z
K_M	kernel matrix
K_M^c	convolution kernel matrix
K_M^p	polynomial kernel matrix
m	power of polynomials
N	number of wells
n	number of observations
p	dimension of features excluding intercept
p	pressure
q	flow rate

q^I	injection rate
q^P	production rate
q^{Iref}	reference state injection rate
q^{Pref}	reference state production rate
q_w	flow rate of well w
$q_{\tilde{w}}$	flow rate of well $\tilde{w} \neq w$
r	Pearsons correlation coefficient
r'	modified Pearsons correlation coefficient
T	temperature
t	time
U	weights of output
u, v, w	real vector
W	weights of input
w	well index
X	feature matrix
x	features in machine learning, inputs in deep learning
X^{pred}	feature matrix for prediction
y	target in machine learning, output in deep learning

y^{pred} target for prediction

ACE alternating conditional expectation

MPCC modified Pearsons correlation coefficient

NARX nonlinear autoregressive exogenous model

PDG Permanent Downhole Gauge

RMSE root mean squared error

RNN Recurrent Neural Network

WIR water injection rate

WPR water production rate

Bibliography

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, and M. Kudlur. Tensorflow: A system for large-scale machine learning. In *OSDI*, Savannah, Georgia, USA, 2016.

[2] A. Aggarwal and S. Agarwal. Ann powered virtual well testing (OTC paper 24981). In *Offshore Technology Conference-Asia*, Kuala Lumpur, Malaysia, 2014.

[3] M. A. Ahmadi, M. Ebadi, and S. M. Hosseini. Prediction breakthrough time of water coning in the fractured reservoirs by implementing low parameter support vector machine approach. *Fuel*, 117:579–589, 2014.

[4] A. Albertoni and L. W. Lake. Inferring interwell connectivity only from well-rate fluctuations in waterfloods. *SPE Reservoir Evaluation & Engineering*, 6(1):6–16, 2003.

[5] A. U. Al-Kaabi and W. J. Lee. Using artificial neural networks to identify the well test interpretation model (includes associated papers 28151 and 28165). *SPE Formation Evaluation*, 8(3):233–240, 1993.

[6] E. A. Antonelo, E. Camponogara, and B. Foss. Echo state networks for data-driven downhole pressure estimation in gas-lift oil wells. *Neural Networks*, 85:106–117, 2017.

[7] S. Athichanagorn. *Development of an Interpretation Methodology for Long-Term Pressure Data from Permanent Downhole Gauges*. PhD thesis, Stanford University, 1999.

[8] S. Athichanagorn and R. N. Horne. Automatic parameter estimation from well test data using artificial neural network (SPE paper 30556). In *SPE Annual Technical Conference and Exhibition*, Dallas, Texas, USA, 1995.

[9] Z. Chen, X. Liao, X. Zhao, S. Lv, and L. Zhu. A semianalytical approach for obtaining type curves of multiple-fractured horizontal wells with secondary-fracture networks. *SPE Journal*, 21(2):538–549, 2016.

[10] D. M. Chorneyko. Real-time reservoir surveillance utilizing permanent downhole pressuresan operators experience (SPE paper 103213). In *SPE Annual Technical Conference and Exhibition*, San Antonio, Texas, USA, 2006.

[11] G. W. Corder and D. I. Foreman. *Nonparametric Statistics: A Step-by-Step Approach (second edition)*. John Wiley & Sons, 2014.

[12] J. Cumming, D. Wooff, , T. Whittle, and A. C. Gringarten. Multiwell deconvolution. *SPE Reservoir Evaluation & Engineering*, 17(4):457–465, 2014.

[13] D. L. Donoho and I. M. Johnstone. Adapting to unknown smoothness via wavelet shrinkage. *Journal of the American Statistical Association*, 90(432):1200–1224, 1994.

[14] S. Dursun, A. Kumar, and R. Samuel. Using data-driven predictive analytics to estimate downhole temperatures while drilling (SPE paper 170982). In *SPE Annual Technical Conference and Exhibition*, Amsterdam, The Netherlands, 2014.

[15] O. Duru and R. N. Horne. Modeling reservoir temperature transients and reservoir-parameter estimation constrained to the model. *SPE Reservoir Evaluation & Engineering*, 13(06):873–883, 2010.

[16] O. Duru and R. N. Horne. Simultaneous interpretation of pressure, temperature and flow-rate data using bayesian inversion methods. *SPE Reservoir Evaluation & Engineering*, 14(02):225–238, 2011.

[17] Jr. R. C. Earlougher. *Advances in Well Test Analysis*. Society of Petroleum Engineers Monograph, 1977.

[18] I. Ershaghi, X. Li, M. Hassibi, and Y. Shikari. A robust neural network model for pattern recognition of pressure transient test data (SPE paper 26427). In *SPE Annual Technical Conference and Exhibition*, Houston, Texas, USA, 1993.

[19] L. Fei-Fei, J. Johnson, and S. Yeung. *Convolutional Neural Networks for Visual Recognition lecture notes*. Stanford University, 2017.

[20] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Thirteenth International Conference on Artificial Intelligence and Statistics*, Sardinia, Italy, 2010.

[21] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[22] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning (second edition)*. Springer, 2009.

[23] Y. He, S. Cheng, S. Li, Y. Huang, J. Qin, L. Hu, and H. Yu. A semianalytical methodology to diagnose the locations of underperforming hydraulic fractures through pressure-transient analysis in tight gas reservoir. *SPE Journal*, 22(3):924–939, 2017.

[24] K. J. Heffer, R. J. Fox, C. A. McGill, and N. C. Koutsabeloulis. Novel techniques show links between reservoir flow directionality, earth stress, fault structure and geomechanical changes in mature waterfloods. *SPE Journal*, 2(2):91–98, 1997.

[25] R. N. Horne. *Modern Well Test Analysis (second edition)*. Petroway, 1995.

[26] R. N. Horne. Listening to the reservoir interpreting data from permanent downhole gauges. *JPT*, 59(12):78–86, 2007.

[27] R. N. Horne and P. Szucs. Inferring well-to-well connectivity using nonparametric regression on well histories. In *Thirty-Second Workshop on Geothermal Reservoir Engineering, Stanford University*, Stanford, California, USA, 2007.

[28] M. M. Kamal, S. S. Morsy, F. Suleen, Y. Pan, A. Dastan, M. R. Stuart, E. C. Mire, and Z. Zakariya. Determination of insitu reservoir absolute permeability under multiphase flow conditions using transient well testing (SPE paper 175012). In *SPE Annual Technical Conference and Exhibition*, Houston, Texas, USA, 2015.

[29] M. Kamal, C. Tian, and S. Suleen. Use of transient tests to monitor progress of flooding in IOR/EOR operations (SPE paper 181473). In *SPE Annual Technical Conference and Exhibition*, Dubai, UAE, 2016.

[30] K. C. Kin. Permanent downhole gauges data interpretation. Master's thesis, Stanford University, 2001.

[31] M. Korjani, A. Popa, E. Grijalva, S. Cassidy, and I. Ershaghi. A new approach to reservoir characterization using deep learning neural networks (SPE paper 180359). In *SPE Western Regional Meeting*, Anchorage, Alaska, USA, 2016.

[32] J. Lee and T. Mukerji. The Stanford VI-E reservoir: A synthetic data set for joint seismic-EM time-lapse monitoring algorithms. In *25th Annual Report, Stanford Center for Reservoir Forecasting, Stanford University*, Stanford, California, USA, 2012.

[33] M. M. Levitan, G. E. Crawford, and A. Hardwick. Practical considerations for pressure-rate deconvolution of well test data. *SPE Journal*, 11(1):35–47, 2006.

[34] M. M. Levitan. Deconvolution of multiwell test data. *SPE Journal*, 12(4):420–428, 2007.

[35] X. Li and M. Lang. Combination prediction of railway freight volume based on support vector machine and NARX neural network. In *LISS*. Springer, Berlin, Heidelberg.

[36] Y. Liu. *Interpreting Pressure and Flow Rate Data from Permanent Downhole Gauges Using Data Mining Approaches*. PhD thesis, Stanford University, 2013.

[37] Y. Liu and R. N. Horne. Interpreting pressure and flow-rate data from permanent downhole gauges by use of data-mining approaches. *SPE Journal*, 18(1):69–82, 2012.

[38] Y. Liu and R. N. Horne. Interpreting pressure and flow rate data from permanent downhole gauges using convolution-kernel-based data mining approaches (SPE paper 165346). In *SPE Western Regional and AAPG Pacific Section Meeting 2013 Joint Technical Conference*, Monterey, California, USA, 2013.

[39] Y. Liu and R. N. Horne. Interpreting pressure and flow rate data from permanent downhole gauges with convolution-kernel-based data mining approaches (SPE paper 166440). In *SPE Annual Technical Conference and Exhibition*, New Orleans, Louisiana, USA, 2013.

[40] C. G. Machado, M. M. Firoozabad, and A. C. Reynolds. Carbon dioxide effects on wellbore-pressure response during injection/falloff test. *SPE Journal*, 2018.

[41] W. A. Nestlerode. The use of pressure data from permanently installed bottomhole pressure gauges (SPE paper 590). In *SPE Rocky Mountain Joint Regional Meeting*, Denver, Colorado, USA, 1963.

[42] M. Nomura. *Processing and Interpretation of Pressure Transient Data from Permanent Downhole Gauges*. PhD thesis, Stanford University, 2006.

[43] L. B. Ouyang and J. Kikani. Improving permanent downhole gauge (PDG) data processing via wavelet analysis (SPE paper 78290). In *European Petroleum Conference*, Aberdeen, Scotland, 2002.

[44] L. B. Ouyang and R. Sawiris. Production and injection profiling: A novel application of permanent downhole gauges (SPE paper 84399). In *SPE Annual Technical Conference and Exhibition*, Denver, Colorado, USA, 2003.

[45] Y. Palabiyik, M. Onur, O. I. Tureyen, and M. Cinar. Transient temperature behavior and analysis of single-phase liquid-water geothermal reservoirs during drawdown and buildup tests: Part i. theory, new analytical and approximate solutions. *Journal of Petroleum Science and Engineering*, 146:637–656, 2016.

[46] H. Rai. Analyzing rate data from permanent downhole gauges. Master's thesis, Stanford University, 2005.

[47] P. M. Ribeiro and R. N. Horne. Pressure and temperature transient analysis: Hydraulic fractured well application (SPE paper 166222). In *SPE Annual Technical Conference and Exhibition*, New Orleans, Louisiana, USA, 2013.

[48] P. M. Ribeiro and R. N. Horne. Detecting fracture growth out of zone by use of temperature analysis. *SPE Journal*, 21(4):1263–1278, 2016.

[49] S. Sankaran, D. Wright, H. Gamblin, and D. Kumar. Creating value by implementing an integrated production surveillance and optimization systeman operator's perspective (SPE paper 187222). In *SPE Annual Technical Conference and Exhibition*, San Antonio, Texas, USA, 2017.

[50] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[51] H. T. Siegelmann, B. G. Horne, and C. L. Giles. Computational capabilities of recurrent NARX neural networks. *IEEE Transactions of Systems, Man, and Cybernetics, Part B (Cybernetics)*, 27(2):208–215, 1997.

[52] R. Socher. *Deep Learning for Natural Language Processing lecture notes*. Stanford University, 2016.

[53] T. Soeriawinata and M. Kelkar. Reservoir management using production data (SPE paper 52224). In *SPE Mid-Continent Operations Symposium*, Oklahoma City, Oklahoma, USA, 1999.

[54] P. Spesivtsev, K. Sinkov, I. Sofronov, A. Zimina, A. Umnov, R. Yarullin, and D. Vetrov. Predictive model for bottomhole pressure based on machine learning. *Journal of Petroleum Science and Engineering*, 166:825–841, 2018.

[55] C. Tian and R. N. Horne. Applying machine learning techniques to interpret flow rate, pressure and temperature data from permanent downhole gauges (SPE paper 174034). In *SPE Western Regional Meeting*, Garden Grove, California, USA, 2015.

[56] C. Tian and R. N. Horne. Machine learning applied to multiwell test analysis and flow rate reconstruction (SPE paper 175059). In *SPE Annual Technical Conference and Exhibition*, Houston, Texas, USA, 2015.

[57] C. Tian and R. N. Horne. Inferring interwell connectivity using production data (SPE paper 181556). In *SPE Annual Technical Conference and Exhibition*, Dubai, UAE, 2016.

[58] C. Tian and R. N. Horne. Recurrent neural networks for permanent downhole gauge data analysis (SPE paper 187181). In *SPE Annual Technical Conference and Exhibition*, San Antonio, Texas, USA, 2017.

[59] T. von Schroeter, F. Hollaender, and A. C. Gringarten. Deconvolution of well-test data as a nonlinear total least-squares problem. *SPE Journal*, 9(4):375–390, 2004.

[60] F. Wang and S. Zheng. Unknown rate history calculation from down-hole transient pressure data using wavelet transform. *Transport in porous media*, 96(3):547–566, 2013.

[61] A. A. Yousef, P. H. Gentil, J. L. Jensen, and L. W. Lake. A capacitance model to infer interwell connectivity from production and injection rate fluctuations. *SPE Reservoir Evaluation & Engineering*, 9(6):630–646, 2006.