USE OF BAYESIAN PROBABILISTIC GRAPHICAL MODELS FOR SIMULTANEOUS VARIABLE RATE / VARIABLE PRESSURE-DROP DECONVOLUTION OF SINGLE WELL AND MULTI-WELL CASES

A Thesis

by

DAVID STEVEN FULFORD

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Thomas A. Blasingame
Committee Members,	Peter P. Valkó
	Maria A. Barrufet
Head of Department,	Jeffrey B. Spath

May 2020

Major Subject: Petroleum Engineering

ABSTRACT

Deconvolution is a useful tool for removing of the effects of non-constant production rate on the observed flowing pressures, or the effects of non-constant flowing pressure on the observed producing rates. The caveat of deconvolution is that it is non-unique and numerically unstable. To maximize the chance of success, an operator can run a controlled experiment in order to minimize these properties of the deconvolution. Examples of such experiments are a stepped production rate test (constant rate steps) or stepped flowing pressure test (constant flowing pressure steps) where the value of a step is constant over a specified time interval. These experiments are costly and therefore frequently not performed, thereby limiting the use of deconvolution as a practical tool for reservoir performance analysis.

In this work we deconvolve both cases simultaneously—constant rate and constant flowing pressure—by correlation of the deconvolved responses. A probabilistic graphic model (PGM) captures the relationships of influence (or probability) between the observed time, rate, and pressure values, and the unobserved latent constant-rate pressure function and constant-pressure rate function. A stochastic algorithm using Hamiltonian Monte Carlo simulation iteratively solves and directly samples the solution space to illustrate the inherent non-uniqueness of the solution. Acceptance or rejection of a specific iteration's proposed solution after evaluation of all functions ensures that acceptance only occurs if

all variables and functions are probabilistically likely given the posterior joint distribution of the entire PGM. Numerical stability is achieved by solving the forward convolution problem–as opposed to the inverse deconvolution problem–as well as utilization of cost functions that are less biased by noise than the squared error cost function.

The structure of the PGM is not altered no matter whether we choose to evaluate just the variable rate case or the variable pressure-drop case, both independently, both together (enforcing correlation), whether we impute missing data, correct data in error, or whether we are analyzing a single well or a multi-well case. We verify our new method with analytical solutions and application to field cases.

DEDICATION

This thesis is dedicated to:

My wife, Robin, for unconditional support and encouragement to pursue my passions;

My children, Mason and Piper, for giving everything purpose;

My family and friends, for always expecting great things;

My advisor, Dr. Thomas A. Blasingame, for helping recognize when I've wandered from the path.

The actual science of logic is conversant at present only with things either certain, impossible, or entirely doubtful ...Therefore the true logic for this world is the calculus of probabilities, which takes account of the magnitude of the probability which is, or ought to be, in a reasonable man's mind.

— *James Clerk Maxwell*

ACKNOWLEDGEMENTS

I would like to thank my committee chair, Dr. Thomas A. Blasingame, for his support and encouragement to pursue this research.

Thanks also go to Dr. Dilhan Ilk, for acting as a source of inspiration of this work and contributing several key ideas to the solutions presented here.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This research was supervised by a thesis committee consisting of Dr. Thomas A. Blasingame, Dr. Peter P Valkó, and Dr. Maria A. Barrufet of the Harold Vance Department of Petroleum Engineering.

The data for the field cases analyzed in this work were provided by Dr. Thomas A. Blasingame.

Portions of this research were conducted with high performance research computing resources provided by Texas A&M University (https://hprc.tamu.edu).

All other work conducted for this research was completed by the student independently.

Funding Sources

There are no outside funding contributions to acknowledge related to the research and compilation of this document.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

1. INTRODUCTION

1.1 Motivation

Recently, considerable advancements have been made in the computer science domain to solve problems involving operations on multi-dimensional arrays. While these efforts have been primarily directed toward improving data-driven modeling, some of the algorithmic machinery has general application to scientific and engineering problems. Specifically, research interest in convolutional neural networks has led to the development of highly-optimized multi-dimensional convolution algorithms. Additionally, to optimize or "train" neural networks, it is necessary to compute the gradient of the model's cost function with respect to each input. This has led to development of frameworks that implement automatic differentiation algorithms that can efficiently compute the gradient of a convolution operation, such as TensorFlow[1], PyTorch[2], and Theano (Bergstra et al. 2010).

Taken together, these algorithms enable construction of computational graphs of complex, high-dimensional problems involving hundreds or thousands of unknown variables and multiple convolution operations. Even with such models, an exact calculation of the gradient can be performed, allowing optimization of these models without the instability or machine precision issues that stem from computing gradients numerically (such as by perturbation, or over iteration steps).

[1] Abadi et al., TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.

[2] Paszke et al., Automatic Differentiation in PyTorch, 2017.

Other researchers primarily interested in solving Bayesian statistical problems, such as hierarchical models that are intractable with analytic methods, have developed open source libraries built upon the computational graph frameworks. Some examples are Edward[3], Stan[4], PyMC3 (Salvatier et al. 2016), and TensorFlow Probability. These libraries abstract much of the mechanics of constructing and manipulating multi-dimensional arrays, allowing a user to write simple statements that may look like trivial variable assignments, but actually direct complex actions of sampling random variables from probability distributions, relating those variables, and computing the probabilistic model likelihood given all variates.

Lastly, no matter the underlying framework and abstraction of a programming library, high-dimensional problems are difficult to optimize for the simple reason that as the number of dimensions increases, the probability of stepping all variables in the correct direction decreases exponentially. The traditional methods of optimization of inverse problems involves Markov Chain Monte Carlo (MCMC), typically in combination with the Metropolis algorithm for acceptance of proposal steps. Two breakthroughs have recently improved MCMC methods by making use of gradient information. First, Hamiltonian Monte Carlo (HMC) has improved the ability to step in the correct direction by exploiting an analogy between the sampled variate and gradient of log likelihood, and the potential and kinetic energy relationship described by Hamiltonian dynamics. This minimizes the random walk behavior and sensitivity to correlated parameters of MCMC,

[3] Tran et al., Edward: A library for Probabilistic Modeling, Inference, and Criticism, 2016.

[4] Carpenter et al., Stan: A Probabilistic Programming Language, 2017.

at the cost of great sensitivity to user choices of step size and length. Second, an algorithm that performs automatic tuning of step size, the No-U-Turn Sampler, has been developed that eliminates the issues of step size and length for HMC. By exploiting the fact that the Hamiltonian is invariant to time, and using an initial random momentum as a seed, forward-time and backward-time processes of parameter steps can be simulated until a reversal in the momentum vector occurs. In practical terms, the model parameters are stepped as far as possible, thereby gaining back sampling efficiency and reducing convergence time.

These algorithmic tools lead us to a re-inspection of deconvolution with an optimization approach. While recent methods have focused on analytic solutions via solving systems of linear equations, those methods require an assumption of a least-squares error model that is not appropriate for time-variant data series, and hand-filtering of data to avoid significant bias by even small outliers. Instead, by optimizing the convolution problem, we immediately obtain a much more stable mathematical operation (vs. deconvolution) and may arbitrarily compose our model within the computational graph framework. This enables us to employ robust cost functions that do not require any data filtering, and perform numerical operations during optimization time, such as to perform regularization of a proposal model evaluated from stepped parameters.

Furthermore, deconvolution is an inherently uncertain process. We are attempting to generation a function that would be the solution of the diffusivity equation for a well geometry corresponding to the data under analysis. Working the forward problem, we can observe that multiple reservoir geometries may generate the same function; or nearly enough such that analysts cannot distinguish between them. Bayesian methods are ideal

for this type of problem and are the de facto standard for solving inverse problems in another industry domain: seismic inversion. Just as geophysicists may bound their solution space by a physical model that relates time and distance, we too can bound our solution space by enforcing positivity of rates and pressures and their time-variant characteristics such as monotonicity and smoothness. This provides the basis for an *a priori* distribution of the constant-rate pressure and constant-pressure rate functions, which we will utilize in conjunction with well data to obtain an *a posteriori* distribution of possible functions. While prior work has generated error ranges around a mean response, our search of the literature shows that generation of a full distribution of possible deconvolved functions to be a novel approach, and therefore a worthwhile problem to explore.

1.2 Statement of the Problem

For decades, deconvolution has been a useful tool in the analysis of well performance. Removal of the effects of non-constant production rate on the observed flowing pressures, or the effects of non-constant flowing pressure on the observed producing rates, allows comparison of producing rates and pressures directly against those predicted by analytic derivation of rate and pressure solutions. The caveat of deconvolution is that, typically, one must choose the variable rate case (**Eq. 1.1**) *or* the variable pressure-drop (**Eq. 1.2**) case; that is, whether the data are treated as a superposition of constant producing rates or as a superposition of constant flowing pressures.

$$\Delta p(t) = \frac{1}{q_r}\int_0^t q'(\tau)\,\Delta p_{cr}(t-\tau)\;d\tau \quad \text{..... Variable Rate Case, 1.1}$$

$$q(t) = \frac{1}{p_i - p_r} \int_o^t \Delta p'(\tau)\, q_{cp}(t_D - \tau)\, d\tau$$ ..... Variable Pressure-Drop Case, 1.2

Whether the assumption of variable rate or variable pressure-drop is valid is determined by the deviation of the operating conditions of the well from the expected case. For example, analysis of the variable rate case is hampered by any period of wide-open flow where the predominant boundary condition at the sand face is no longer the production rate, but instead the flowing pressure. Likewise, analysis of the variable pressure-drop case is complicated by any period of shut-in, where the producing rate is constant at a value of zero. To put it simply, we can state that the success of either approach depends on the assumed flowing condition matching the actual flowing conditions.

We may derive a solution for either case from the same initial state, indicating that "the reservoir doesn't know the difference." In doing so, we might expect each to be accurate to the degree of agreement of the operating conditions with assumed by each case. Further improvement may come from enforcing relationship or correlation between the two. However, given that we know that error may result from deviance from constant rate or pressure operation conditions, the correlation must also allow, or even expect, a degree of error.

We use a probabilistic graphical model (PGM) (**Fig. 1.1**) to represent the variable rate case, the variable pressure-drop case, and their correlation to one another. The PGM captures the conditional dependencies that exist in the relationships of our variables. Latent (hidden) variables on the left represent the beta-derivatives (log-log derivatives) of the constant-rate pressure and constant-pressure rate functions (kernel functions). Convolution

of the kernel functions with observed rates and pressures occurs in each grouped box. The variables on the right represent the outputs of the convolution, with Duhamel's principle applied to correlate the two kernel functions. Observed variables are indicated with double line borders.

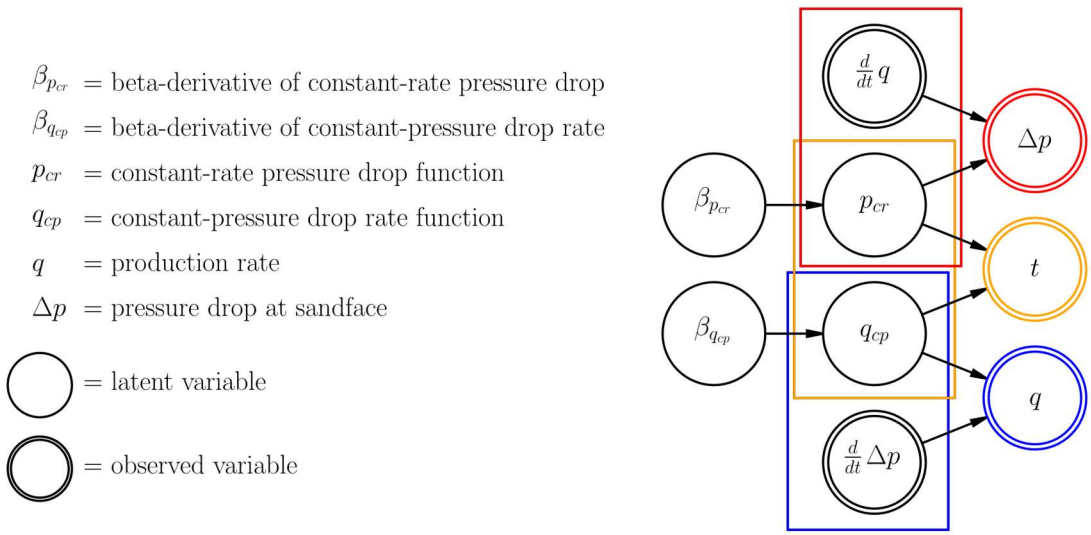

Figure 1.1 — Probabilistic graphic model (PGM) structure.

Exploitation of the graph structure allows us to infer the value of a variable given observation of another. For example, we may reconstruct the constant flowing pressure type curve from observation of the variable rate production data and use of a correlating function. Further, we may provide interpretation of the constant flowing pressure type curve to exert influence on the constant rate type curve.

The use of the PGM provides a natural framework for a Bayesian approach to a situation in which multiple variables give evidence for a single variable. This is an ideal property for the ill-conditioned problem of deconvolution, where even a known function (convolution), known output (observed rate profile), and known input (observed pressure profile) is not adequate to uniquely resolve the remaining input (constant-rate pressure profile). As opposed to a frequentist statistical framework in which the "true" parameters are fixed and the data is uncertain, the Bayesian framework treats the data as fixed and the parameters uncertain. This aligns well with our real-world perspective in which we are attempting to determine the parameters of an unknown function given our observation of a variable-rate or variable-pressure profile. We have observed the data; it does not change. It is the function that must be determined.

Given that we explicitly treat every variable's "true" parameters as uncertain, we may compile information from among all observed variables that contain information about another specific unobserved variable, respective to the inverse of variance of each variable we have observed. There are several advantages to this; primarily, the data may provide conflicting evidence which the Bayesian approach automatically addresses:

- First, we do not have to give preference for one variable while discarding others.
- Second, the method should be highly tolerant of errors. The numerical optimization scheme allows utilization of robust, non-analytic cost functions.
- Third, the posterior distribution of deconvolved functions is directly sampled, and may be viewed for better understanding of uncertainty.

- Fourth, if any part of the data are missing, we may impute (i.e. reconstruct) the missing data from evidence among all variables. For example, we can impute:
  - all or part of the constant-pressure rate profile when solving for the constant-rate pressure profile.
  - all or part of the variable-pressure profile when solving for the constant-rate pressure profile.
  - all or parts of both the variable-rate and variable-pressure profiles, even if the missing parts fall on different intervals.
  - the initial reservoir pressure, or correct an incorrect initial guess.
  - other possibilities exist, including combinations of the above.

The PGM captures the conditional dependencies that exist between variables. Evaluation of all variables, and all variable relationships, in the PGM occurs simultaneously. In the multi-well case, we treat each well as an additional observation of a point in the convolved variable rate or variable pressure-drop function. This representation maintains the exact same model no matter whether we wish to evaluate just the variable rate case (or vice versa), both independently, both together (enforcing correlation), or whether we are analyzing a single well or a multi-well case.

A regression of the solution occurs with a stochastic algorithm, Markov Chain Monte Carlo simulation (MCMC). MCMC methods are well suited for non-unique inverse model problems. By random sampling the latent variables of a model and comparing the model output against observations, the posterior distribution of the model may be directly sampled. Then, any latent variables in the model are explicitly known. We also avoid the

complexity of inverting a convolution function and solving the resulting deconvolution function; however, we introduce the high-dimensional and non-linear problem of determining how to construct a proposal for the constant-rate pressure drop or constant-pressure rate functions.

Recent advances in machine learning techniques for direct sampling of posterior distributions of complex non-linear models merits an investigation into application of these techniques as a potential avenue for solving our problem. Hoffman and Gelman (2014) have introduced the No-U-Turn Sampler (NUTS) which improves upon prior Hamiltonian Monte Carlo (HMC) methods. HMC (Duane et al. 1987) is itself an improvement on MCMC whereby the total "energy" of a variable is conserved by consideration of the variable's position (the current value) and momentum (the negative log probability density function (PDF)). While HMC suppresses the random walk nature of MCMC, the cost of implementation is that the gradient of the log-PDF must be known, and success depends upon hand-tuned step-size. NUTS, then, improves upon HMC by introducing an adaptive form that requires no hand-tuning.

The convolution algorithms that we make use of were developed for other machine learning methods such as deep learning (neural networks). The algorithms may operate in the real domain or the spectral domain by automatic determination of which provides better computing performance. Generally, as the length of the arrays increases, convolution performed by use of the Finite Fourier Transform begins to show better computing performance. However, the inputs and outputs of the convolution algorithms are always in the real time domain.

We generate the constant-pressure rate and constant-rate pressure functions prior to acceptance of a given iteration of the MC algorithm, allowing us to state that they are generated simultaneously. The functions may also be correlated by use of Duhamel's principle. This allows us to apply our deconvolution algorithm no matter whether a subject well is in a state of constant rate or constant pressure-drop. Observed variables may be "masked" over intervals where the assumption of constant-rate or constant-pressure not are valid.

Due to the probabilistic nature of the approach, we may construct arbitrary cost functions to judge goodness of fit. The advantage of doing so is that our method becomes less sensitive to outliers in the data. Additionally, we may provide regularization to any metric of interest without requiring a reformulation of our cost function in contrast to what would occur in a minimization of least squares approach. It follows that extension to the case of multi-well deconvolution is treated as repeated observations of the rate and/or pressure variables and is no more difficult to perform than the single well case. The multi-well case is a simple hierarchical model in which we assume every well must have the same dimensionless rate or pressure function.

Flow regime diagnosis is expressed as *a priori* knowledge (a "prior") for the beta-derivative, $\beta(t)$ (**Eq. 1.3**), over specified time intervals.

$$\beta(t) = \frac{d}{dt}\frac{\ln f(t)}{\ln t} = \frac{t}{f(t)} f'(t) \quad \text{..... 1.3}$$

Eq. 1.3 can be applied to the constant-pressure rate function or the constant-rate pressure function – $\beta_p(t)$ and $\beta_q(t)$. When the constant-rate pressure function and constant-pressure

rate function are correlated, a prior on $\beta_q(t)$ will also affect the behavior of $\beta_p(t)$ as influence flows left-to-right in **Fig. 1.1** from to $\beta_q(t)$ to t, and back right-to-left to $\beta_p(t)$. The model's graph, e.g. **Fig. 1.1**, allows visualization of the probability flows.

The beta-derivative for each case is constructed with a Gaussian Process (GP). A GP is non-parametric function that acts as a prior over functions (as opposed to a prior over scalar values). Rather than expressing $\beta(t)$ as a series of distributions at each observed time step, the GP expresses all the possible $\beta(t)$ functions that yield the observed rates and pressures. The advantage of a non-parametric method is that absolutely no prior assumption is made regarding the form of the deconvolved response. Transformations of the GP values enforce the constraints that $\beta(t)$ be strictly positive and monotonic.

The utility of the PGM allows an interpretation of the constant-rate pressure function, such as would occur in pressure transient analysis (PTA), to be applied generally to any well history. This extends to any other variable in the model, including variables typically in the domain of rate transient analysis (RTA). By simultaneous evaluation, we take a first step towards merging RTA & PTA.

1.3 Objectives

The main objectives of the work are:

- To *provide* a probabilistic graphical model for simultaneous variable rate / variable pressure-drop deconvolution.
- To *provide* a means of inferring the posterior distribution of the probabilistic graphical model.

- To *validate* the probabilistic graphic model across a variety of single and multi-well cases.
- To *evaluate* variable-rate / variable-pressure field performance data using the model to perform simultaneous deconvolution of both cases.

2. LITERATURE REVIEW

Deconvolution has been discussed in the petroleum literature for the past 70 years dating back to the seminal work of van Everdingen and Hurst (1949). They presented the development of the relationship between the constant-rate pressure solution and the constant-pressure rate solution in Eqs. 1.1 & 1.2 in both the real time domain (**Eq. 2.1**), and in the Laplace domain (**Eq. 2.2**).

$$t_D = \int_0^{t_D} q_D(\tau)\, p_D(t_D - \tau)\, d\tau \quad \text{.....2.1}$$

$$\frac{1}{s^2} = \bar{q}_D(s)\bar{p}_D(s) \quad \text{.....2.2}$$

Most methods introduced into the literature invert and solve a discretized form of Eq. 2.1. Less frequently, due to the physical challenges involved with setting a fixed pressure boundary condition in a well test, as opposed to a fixed flow rate boundary condition, do methods in the literature solve a discretized form Eq. 2.2. **Eq. 2.3** and **Eq. 2.4** provide the discretized forms.

$$\Delta p(t) = \frac{1}{q_r}\sum_{i=1}^{n}\left(q_i - q_{i-1}\right)\Delta p_{cr}(t - t_{i-1}) \quad \text{.....2.3}$$

$$q(t) = \frac{1}{p_i - p_r}\sum_{i=1}^{n}\left(\Delta p_i - \Delta p_{i-1}\right) q_{cp}(t - t_{i-1}) \quad \text{.....2.4}$$

Recent efforts have formulated the deconvolution problem as a system of linear equations solved by minimization of least squares, and have added additional constraints in efforts to increase stability of this approach. Von Schroeter, Hollaender, and Gringarten (2001,

2002, 2004), and Ilk, Valkó, and Blasingame (2005, 2006) both introduced real time domain methods that solve a weighted least squares problem with additional regularization terms for the "non-smoothness" of the solution. The author's approaches differ in how constraints are enforced. Von Schroeter, Hollaender, and Gringarten (2001, 2002, 2004) enforce monotonicity of the pressure function by means of the "encoding of the solution." We refer to this as a variable transformation and note that an exponential transform ensures the positivity of the von Schroeter solution. The Ilk method does not ensure monotonicity, however, their regularization scheme makes monotonicity a likely (but not certain) outcome.

Levitan (2003, 2005) and Levitan, Hardwick, and Crawford (2004, 2006) presented an improvement to von Schroeter's method wherein the sandface rate is added to the cost function as a model parameter. In a sense, the Levitan algorithm constructs a model of pressures and flow rates that are adjusted during optimization of the cost function.

Çinar et al. (2006) performed a comparative study of the Von Schroeter, Hollaender, and Gringarten (2002, 2004), Ilk, Valkó, and Blasingame (2005, 2006), Levitan (2005), and Levitan, Hardwick, and Crawford (2006) methods and found each to be robust and accurate.

Ilk, Valkó, and Blasingame (2007) introduced an alteration of their original method that uses cumulative production to achieve a variable-pressure drop deconvolution. The authors then proposed use of *both* variable-rate and variable-pressure drop deconvolution as a theoretically consistent diagnostic method.

Pimonov et al. (2009, 2010) introduced an adjustment to the methods of von Schroeter and Levitan that added weights to each rate and pressure value similar to the Ilk, Valkó, and Blasingame (2005, 2006) method. The effect of the weights creates an algorithm that is more robust to noise by adjusting the variance on specific data points when the assumption of gaussian error is not valid.

Onur and Kuchuk (2010, 2012) introduced an adjustment to the prior methods of von Schroeter and Levitan. The author's based their algorithm upon the pressure derivative, which eliminates the dependency on initial reservoir pressure.

Ahmadi, Sartipizadeh, and Ozkan (2017) introduced a two-step process in which the first step involves a least squares solution in the real time domain, and then an optimization technique that minimizes the second log-derivative of the pressure solution. The optimization step seeks to find a smooth log-derivative that is close to the exact solution.

Cheng, Lee, and McVay (2003, 2005) developed a spectral domain method that utilizes the Finite Fourier Transform (FFT). After transformation, Eq. 1 becomes a simple arithmetic function, $\overline{p}_{wD} = \overline{q}_D \overline{p}_D$, where the overbar refers to the transform function to the spectral domain – in this case, the Fourier transform. Cheng et al. developed a novel method of de-noising data in the spectral domain, however, the author's efforts were focused on removal of wellbore storage effects, and not a general algorithm for deconvolution.

Various researchers have also developed variations on the von Schroeter, Hollaender, and Gringarten (2001, 2002, 2004) algorithm for multi-well scenarios. Levitan (2006, 2007) presented a version that considers interference effects as a superposition in space.

Cumming et al. (2013, 2014) applied the Levitan multiwell concept to field data to validate the method as a practical tool.

Liu and Horne (2011) introduced a machine learning method for deconvolution. The author's method does not attempt to directly solve for the constant-rate pressure function, but rather proposes a matrix $\mathbf{X}$ of values $\mathbf{x}(t)$ composed of many possible time-rate functional forms, and the constant-rate pressure function is predicted by a simple linear model $y(t) = \theta \mathbf{x}(t)$. The authors then make use of a kernel function to project the linear model into a high-dimensional phase-space where non-linear features in the low-dimensional space may be transformed into linear features in the phase-space.

Tian and Horne (2015, 2019) extend the method of Liu and Horne (2013) by adding L1 and L2 regularization terms to simplified version of the kernel regression model. The authors found an improvement with a lower-dimensional (fewer parameter) kernel and regularization over a more complex kernel and no regularization due to the tendency of regularization to prevent overfitting of training data.

Tian and Horne (2017) have also applied recurrent neural networks (RNN) to the problem of deconvolution. The use RNN does not require any hand-tuning of functional time-rate forms, providing a non-parametric optimization method. The authors found that the use of the RNN improved results over the prior use of the linear regression model.

The majority of current research has been focused on improving the algorithm laid out by von Schroeter, Hollaender, and Gringarten (2001, 2002, 2004). Our approach is fundamentally different than any we are aware of in the petroleum literature. Indeed, recent

developments by machine learning researchers in non-linear optimization have only just enabled our method. We do not solve any least squares problem. Therefore, we do not suffer catastrophic divergence due to minor deviations from the assumption of Gaussian residuals. We also do not require an analytic representation of the solution, which allows us to perform complex numerical operations during solve time such as convolutions, evaluation of first- and second-order derivatives, and solving systems of linear equations. Taken together, these allow us to use alternative non-convex cost functions that may not have a unique analytic solution but are more robust to noise.

3. BAYESIAN DECONVOLUTION MODEL DEVELOPMENT

We represent both the unknown constant-rate pressure function and constant-pressure-drop rate function as a random multidimensional normal distribution, i.e. a Gaussian process, of the beta-derivatives evaluated over logarithmically distributed fixed knots. Interpolation of the knots is performed with B-splines, which requires solving a pseudo-inverse of a non-square matrix in every iteration of our solver (although we implement a special case for degree 1 B-splines that eliminates this calculation). Regularization of our solution comes in three forms: 1) the number of knots chosen over each log cycle, 2) the covariance matrix that defines the Gaussian processes, and 3) the measurement of curvature of the log of the unknown function with respect to the log of time.

We perform optimization by use of Hamiltonian Monte Carlo. By making use of modern software libraries that us to express our model symbolically and automatically differentiate with respect to any variable, we supply the gradient information of the model to the Hamiltonian Monte Carlo algorithm. The algorithm can then solve non-linear, non-convex problems.

Our method is enabled by expression of the deconvolution problem within a Bayesian context. Our advantage is a complete separation of the mechanisms of the model—that is, our understanding of the physics of the problem—and how we reason about the model, as opposed to a domain-specific formulation. Any parameter may be treated as uncertain and solved for during optimization.

We may never obtain the "true" maximum likelihood (or equivalently, minimum error) with this approach, only many possibilities that lie close to the maximum likelihood. This is the tradeoff we make. Our method is inherently stochastic and taking advantage of this fact precludes a discrete solution. In practice, the gain in understanding from illustration of the uncertainty of the solution space dominates any loss in [likely false] precision of a discrete approach.

3.1 Power-law Basis for Flow Rate and Pressure Drop Functions

We are primarily interested in diagnostics for commonly observed flow regimes in onshore well such as those occurring from fractured vertical wells, or multi-fractured horizontal wells (MFHW). In these reservoirs, transient radial flow and/or transient linear flow will dominate at early-times, and boundary-dominated flow and/or pseudosteady state flow will dominate at late times. These flow regimes yield distinct rate-time and pressure-time signatures that we may use to diagnose reservoir parameters such as permeability or reservoir geometry. A summary of these relationships is given in **Table 3.1**.

Each of these has a unique signature of the beta-derivative (Hosseinpour-Zonoozi, Ilk, and Blasingame, 2006). We generalize the pressure-time function (and equivalently, the rate-time function) as a power-law function with stepwise values of the beta-derivative (Eq. 3.1), and then derive a diagnostic from the beta-derivative that yields a constant for each flow regime. These transforms are given in **Table 3.2**. Details of the derivations are given in **Appendix A**.

$$\Delta p_{cr}(t_n) = \Delta p_1 \prod_{i=2}^{n} t_i^{\beta_{p,i}} \, t_{i-1}^{-\beta_{p,i}} \quad \text{.....3.1}$$

Eq. 2.6 may be discretized for numerical evaluation:

$$\Delta p_{cr}(t_n) = \Delta p_1 \exp\left[\sum_{i=2}^{n} \beta_{p,i} \ln \frac{t_i}{t_{i-1}} \right] \quad \text{.....3.2}$$

Defining:

$$\tau_i = \ln \frac{t_i}{t_{i-1}} \quad \text{.....3.3}$$

We have after substitution:

$$\Delta p_{cr}(t_n) = \Delta p_1 \exp \sum_{i=2}^{n} \beta_{p,i} \tau_i \quad \text{.....3.4}$$

We can write a similar function for $q(t)$ where the beta-derivative would be negative for a monotonically decreasing function:

$$q_{cp}(t) = q_1 \exp \sum_{i=2}^{n} -\beta_{q,i} \tau_i \quad \text{.....3.5}$$

Equivalently, taking the reciprocal:

$$\frac{1}{q_{cp}(t)} = \frac{1}{q_1} \exp \sum_{i=2}^{n} \beta_{q,i} \tau_i \quad \text{.....3.6}$$

Eq. 3.4 and Eq. 3.5 are the functions used to calculate the pressure and rate functions, respectively. We note that Δp_1 and q_1 are scalars and, therefore, will *fall out* as bias (intercept) terms to minimize the residuals.

Table 3.1 — Mathematical forms of flow regimes of interest

Flow Regime	Rate-Time	Pressure-Time	Functional Form
Radial	$q(t) \propto 1/\ln t$	$\Delta p(t) \propto \ln t$	Logarithmic
Linear, Bi-linear, etc.	$q(t) \propto 1/\sqrt{t}$	$\Delta p(t) \propto \sqrt{t}$	Power-law
Boundary-Dominated	$q(t) \propto 1/e^{t}$	n/a	Exponential
Pseudosteady State	n/a	$\Delta p(t) \propto t$	Linear

Once we have the pressure function, we compute the pressure derivative as a function of the beta derivative. In doing so, we can correlate the beta derivative to standard slope interpretations of flow regimes in rate-transient analysis and pressure-transient analysis workflows:

$$t\Delta p'(t) = \beta(t)\Delta p(t) \quad \text{.....} \quad 3.7$$

Table 3.2 — Transforms of beta-derivative to diagnose flow regimes of interest

Flow Regime	Beta-derivative Diagnostic	Well Testing Diagnostic
Radial	$\beta \ln t = 1$	$\frac{t}{\beta\Delta p}\frac{d\beta\Delta p}{dt} = 0$
Linear	$\beta = \frac{1}{2}$	$\frac{t}{\beta\Delta p}\frac{d\beta\Delta p}{dt} = \frac{1}{2}$
Boundary-Dominated	$\frac{\beta_q}{t} = \text{constant}$	n/a
Pseudosteady State	$\beta_p = 1$	$\frac{t}{\beta\Delta p}\frac{d\beta\Delta p}{dt} = 1$

We note that Δp and q_1 are scalars and, therefore, will *fall out* as a terms to minimize the residuals.

We desire that the pressure and rate functions should be monotonic. This translates to enforcing the beta-derivative to be strictly positive. Exponentiation of the reconstruction function is a common practice in the literature but it presents an non-linear gradient which may be difficult to optimize. For our pressure function, we have chosen to use the softplus function (Eq. 3.7) as it combines the aspects of an exponential transform for negative values, and no transform for positive values. Importantly, it is monotonic, it is a strictly positive transform, it has an analytic solution for its derivative, and it reduces the non-linearity of the exponential transform. For the rate function, where we expect exponential behavior during boundary-dominated flow, we do employ an exponential transform. **Fig. 3.1** shows a comparison of the softplus function with the exponential function and the linear rectifier function, i.e. $f(x) = \max(x, 0)$. We do not use the linear rectifier function as it has an undefined gradient for values less than zero.

$$\text{softplus}(x) = \ln\left[1 + e^{x}\right] \quad \text{.....} \quad 3.7$$

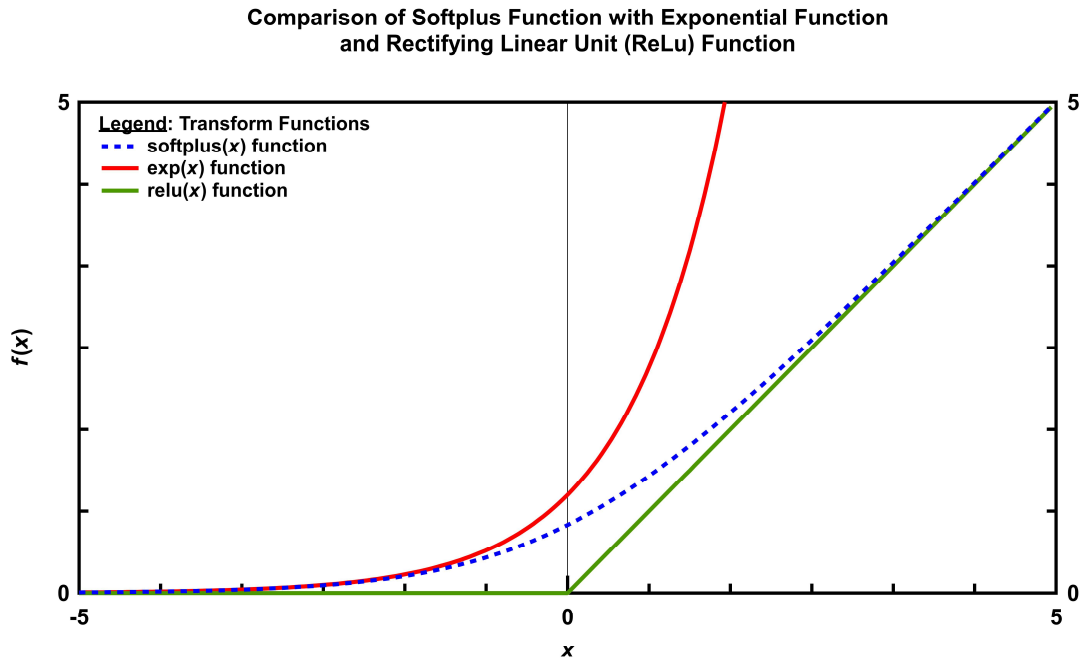

Figure 3.1 — Comparison of softplus, exponential, linear rectifier functions.

The derivative of the softplus function is the logistic sigmoid function, which itself is a hyperbolic tangent function with non-zero location and scale. Using the identity for the derivative a logarithm and applying the chain rule:

$$\frac{d}{dx}\text{softplus}(x) = \frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}} = \frac{1}{2} + \frac{1}{2}\tanh\frac{x}{2} \quad \text{.....3.8}$$

It is important that an analytic derivative exist for every operation performed in our algorithm as it is critical for parameter stepping of the optimization scheme.

3.2 Use of Gaussian Processes to Generate Beta-Derivative Functions

The $\beta_p(t)$ and $\beta_q(t)$ functions remain to be determined. To continue with our non-parametric approach, we introduce the use of Gaussian processes (GP) to construct the

beta-derivatives. A GP is an *n*-dimensional Gaussian distribution fit over *n* data points. An appealing feature of a multidimensional Gaussian distribution is that the conditional distribution given any other number of Gaussians distributions is also a Gaussian distribution. Starting with a two-dimensional Gaussian distribution with means u and covariance matrix Σ:

$$\begin{bmatrix} f(x_1) \\ f(x_2) \end{bmatrix} = \mathcal{N}\left(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix} \right) \quad \text{.....3.9}$$

The parameters of the conditional distribution of $f(x_{2|1}) \sim \mathcal{N}(\mu_{2|1}, \sigma_{2|1})$ are:

$$\mu_{2|1} = \mu_2 + \Sigma_{21}\Sigma_{11}^{-1}\left(f(x_1) - \mu_1\right) \quad \text{.....3.10}$$

$$\sigma_{2|1} = \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12} \quad \text{.....3.11}$$

We can see that the GP is defined completely by its mean and covariance. As an example, if we construct our GP such that the mean is zero, i.e. every point is a normal distribution with mean of zero, then the GP is defined completely by its covariance matrix. The covariance matrix controls the behavior of the GP, while the mean is the value the GP will regress to if the covariance between the current point and the nearest inducing points is very near zero.

Extending the concept to apply to an arbitrary number of x_n inducing points, the prediction at a specific point x^* is the conditional distribution:

$$f\left(x^* | f(x_1), f(x_2), \ \ldots \ , f(x_{n-1}), f(x_n)\right) \quad \text{.....3.12}$$

A multidimensional Gaussian distribution represents the distribution at a specific point:

$$f(x^*) = \frac{1}{z}\exp\left[\left(x^* - \mu^*\right)\Sigma^{-1}\left(x^* - \mu^*\right)\right] \quad \text{.....3.13}$$

Where z is some normalizing constant, and Σ is of the form:

$$\Sigma = \begin{bmatrix} \kappa(x_1,x_1) & \kappa(x_1,x_2) & \cdots & \kappa(x_1,x_{n-1}) & \kappa(x_1,x_n) \\ \kappa(x_2,x_1) & \kappa(x_2,x_2) & \cdots & \kappa(x_2,x_{n-1}) & \kappa(x_2,x_n) \\ \vdots & & \ddots & & \vdots \\ \kappa(x_{n-1},x_1) & \kappa(x_{n-1},x_2) & \cdots & \kappa(x_{n-1},x_{n-1}) & \kappa(x_{n-1},x_1) \\ \kappa(x_n,x_1) & \kappa(x_{n1},x_2) & \cdots & \kappa(x_n,x_{n-1}) & \kappa(x_n,x_n) \end{bmatrix} \quad \text{.....3.14}$$

By analogy to Eq. 3.10 and 3.11, we may write:

$$\mathbf{E}\left[f(x^*)\middle|f(x_1),f(x_2),\ \ldots\ ,f(x_{n-1}),f(x_n)\right] = \mu^* + \Sigma_{x^*\mathbf{x}}\Sigma_{\mathbf{xx}}^{-1}\left(f(x^*) - \mu^*\right) \quad \text{.....3.15}$$

$$\mathbf{cov}\left[f(x^*)\middle|f(x_1),f(x_2),\ \ldots\ ,f(x_{n-1}),f(x_n)\right] = \Sigma_{x^*x^*} - \Sigma_{x^*\mathbf{x}}\Sigma_{\mathbf{xx}}^{-1}\Sigma_{x^*\mathbf{x}}^{\mathbf{T}} \quad \text{.....3.16}$$

And by analogy to Eq. 3.9, this is given by a matrix of some kernel function, $\kappa(x,x)$:

$$\begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{n-1}) \\ f(x_n) \\ f(x^*) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_{n-1} \\ \mu_n \\ \mu^* \end{bmatrix}, \begin{bmatrix} \kappa(\mathbf{x},\mathbf{x}) & \kappa(x^*,\mathbf{x})^{\mathbf{T}} \\ \kappa(x^*,\mathbf{x}) & \kappa(x^*,x^*) \end{bmatrix}\right) \quad \text{.....3.17}$$

Where:

$$\kappa(x^*,\mathbf{x}) = \begin{bmatrix} \kappa(x^*,x_1) \\ \kappa(x^*,x_2) \\ \vdots \\ \kappa(x^*,x_{n-1}) \\ \kappa(x^*,x_n) \end{bmatrix} \quad \text{.....3.18}$$

We summarize the GP nomenclature as:

$$f(x^*) = \mathrm{GP}(\mu, \kappa) \quad \text{..... 3.19}$$

The choice of kernel function introduces the correlation between close samples in order to enforce some level of smoothness on the function. One of the most common kernel functions is the Squared Exponential Kernel. Its form is similar to that of a Gaussian distribution:

$$\kappa(x^*, \mathbf{x}) = \exp\left[\frac{-(x^* - \mathbf{x})^2}{2\ell^2}\right] \quad \text{..... 3.20}$$

Where ℓ is a characteristic length (i.e. length-scale). Illustrations of the covariance matrix for various values of the length-scale, ℓ, are given in **Fig. 3.2**:

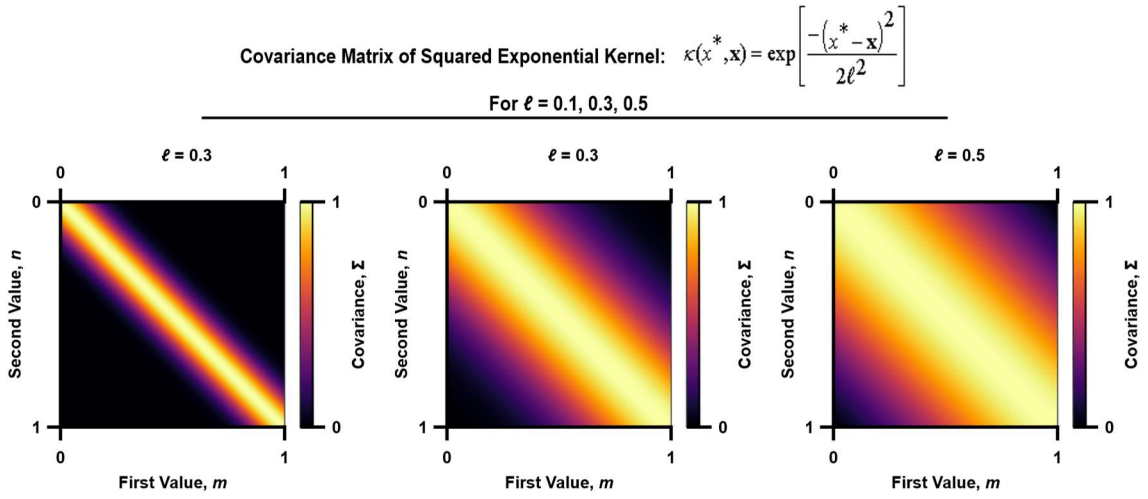

Figure 3.2 — Evaluation of squared exponential kernel for $\ell = 0.1, 0.3, 0.5$.

Any function that evaluates to a valid covariance matrix can be a kernel function. Other examples include:

White Noise:

$$\kappa(x^*, \mathbf{x}) = \sigma^2 \mathbf{I} \quad \text{.....} 3.21$$

Constant:

$$\kappa(x^*, \mathbf{x}) = \sigma^2 \quad \text{.....} 3.22$$

Laplace Kernel:

$$\kappa(x^*, \mathbf{x}) = \exp\left(\frac{-\left|x^* - \mathbf{x}\right|}{\ell}\right) \quad \text{.....} 3.23$$

Linear Kernel:

$$\kappa(x^*, \mathbf{x}) = x^* \mathbf{x} \quad \text{.....} 3.24$$

Kernel functions may also be constructed as an additive process from other kernel functions, or by multiplication of other kernel functions. One appealing choice of kernel functions is the Gibbs kernel, with which we make the length-scale dependent on the feature value. The Gibbs kernel is:

$$\kappa(x^* \mathbf{x}) = \prod_{i=1}^{n}\left[\left(\frac{2\ell(x^*)\ell(\mathbf{x})}{\left[\ell(x^*)\right]^2 + \left[\ell(\mathbf{x})\right]^2}\right)\exp\left(\frac{-\left(x^* - \mathbf{x}\right)^2}{\left[\ell(x^*)\right]^2 + \left[\ell(\mathbf{x})\right]}\right)\right] \quad \text{.....} 3.25$$

The length-scale function is given by any arbitrary function. We have chosen a logarithmic function:

$$\ell(\mathbf{x}, b, m) = b + m \ln \mathbf{x} \quad \text{.....} \quad 3.26$$

Where b is an intercept term, and m is a slope term.

We have evaluated $\ell(b = 0.04, m = 0.25)$ for the covariance matrix in **Fig. 3.3**:

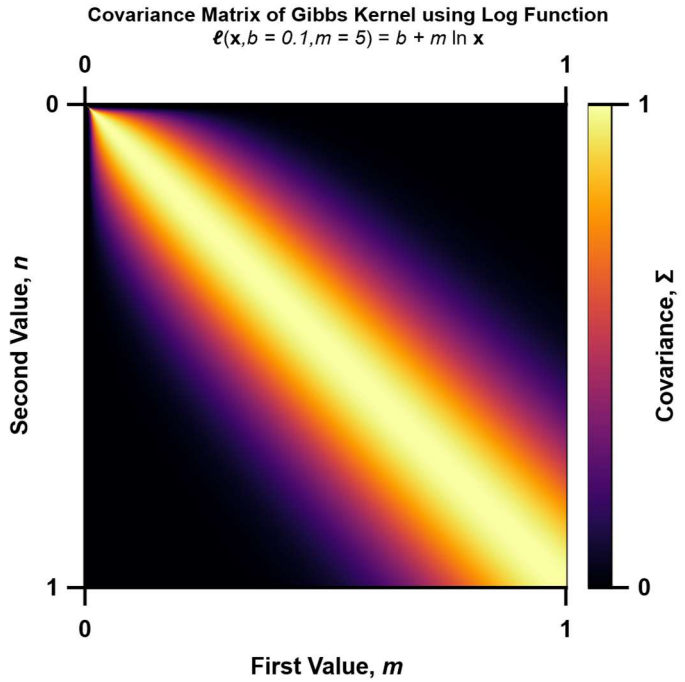

Figure 3.3 — Covariance matrix for Gibbs kernel using log function..

We justify this choice by our desire to apply smoothing over log cycles to a linear time scale. That is, we wish greater smoothing on large values of time than on smaller values of time such that the level of smoothing is somewhat even when measured across log cycles.

Recalling the similar forms of Eq. 3.9 and Eq. 3.11, we expect the same characteristic behavior from both the dimensionless rate and dimensionless pressure functions. Therefore, they both share the same GP latent function. The posterior distribution of each function is computed from the latent function and the observed data. Using the softplus and exponential transforms to achieve strictly positive values of what is otherwise an unbounded Gaussian distribution:

$$\beta_p(t)\,\big|\,q'(t),\, p_{wf}(t) \sim \text{softplus}\big[\text{GP}(\mu,\kappa)\big] \quad \text{.....3.27}$$

$$\beta_q(t)\,\big|\,p'(t),\, q(t) \sim \exp\big[\text{GP}(\mu,\kappa)\big] \quad \text{.....3.28}$$

Where the value of μ is typically set to zero. The effect of this assumption is that the GP will converge to zero where there are no inducing-points and the covariance diminishes to zero. In practice, this only occurs before or after our time period of interest and does not affect our results.

In practice we have obtained better results with a White Noise kernel function and our regularization method. This kernel function reduces to a Gaussian random walk (GRW). This is primarily due to the complexity of computing a GP , which is $\mathcal{O}(n^2)$ in the storage demands, and $\mathcal{O}(n^3)$ in computational complexity due to inversion of an $n \times n$ matrix (Rasmussen and Williams 2006). With 10,000 data points, equivalent to roughly 300 hours of data at 120 second polling intervals, holding just the covariance matrix of a GP in memory, not including any decompositions or inversions, requires roughly three-quarters

of a gigabyte of memory. A GRW, on the other hand, has no correlation between points, and therefore the covariance matrix need not be stored.

While we may "control" the GRW by adjusting the standard deviation of the step size, the lack of any covariance matrix means that there is no constraint on the smoothness of the GRW, only the magnitude of sampled values (hence white noise). The loss in smoothing achieved with the covariance function is addressed by alternative regularization schemes. A GRW step is summarized as:

$$f(x_i) = f(x_{i-1}) + \mathcal{N}\left(0, \sigma^2\right) \Leftrightarrow \mathcal{N}\left(f(x_{i-1}), \sigma^2\right) \quad \text{.....} 3.29$$

From Eq. 3.29 it is obvious that each step is independent of every other step, thereby significantly reducing complexity of calculation.

3.3 Contrast of Maximum Likelihood with Minimization of Error

We use Markov Chain Monte Carlo to step the model parameters at each iteration. While we obtain a posterior distribution of functions, we hope to provide an analogy for understanding the Bayesian maximum likelihood estimate (MLE) from a least-squares perspective.

Assume that we had a model parameter that is normally distributed in accordance with an ordinary least squares model. Regression proceeds by minimizing the sum of square of residuals between model and data. Assume *m* and *b* to be some arbitrary model parameters that are also function of time. We write a model for the prediction *y* of *m* and *b* as:

$$y_i = m_i t_i + b_i + e \quad \text{.....} 3.30$$

And the cost function to minimize the residuals would be:

$$J(m,b) = \frac{1}{2n}\sum_{i=1}^{n}\left(y_i - \left[m_i x_i + b_i\right]\right)^2$$

The equivalent Bayesian representation of this is a series product Gaussian likelihoods:

$$\prod_{i=1}^{n}\mathcal{N}\left(y_i\middle|\left[t_i, m_i, b_i\right], \sigma^2\right) \quad \text{.....3.31}$$

We wish to impose regularization on m_i. We will show the equivalence of a Gaussian distribution on m with L2 regularization. Starting with a Gaussian prior:

$$\prod_{i=1}^{n}\mathcal{N}\left(m_i\middle|t_i, \lambda^{-1}\right) \quad \text{.....3.32}$$

We state Bayes' theorem to emphasize the relationship between ***priors*** and ***likelihoods***. Here, Eq. 3.31 is a likelihood, and Eq. 3.32 is a prior. Bayes theorem indicates that we simply take the product of all priors and likelihoods. We pause here to note that this indicates that the difference in treatment of priors and likelihoods is only one of semantics. Priors are model parameters for which we express a constraint as a matter of knowledge, or equivalently, belief, while likelihoods are model parameters or outputs over which we express a constraint as a matter of observed data. Priors allow us to encode data and information that may not be explicitly quantifiable to a model; we emphasize that this is analogous equivalent to a constraint in a frequentist framework. Proceeding with our demonstration, we have Bayes theorem:

$$\pi(\theta \mid y) = \frac{f(y\middle|\theta)\pi(\theta)}{\int f(y\middle|\theta)\pi(\theta)d\theta} \propto f(y\middle|\theta)\pi(\theta) \quad \text{.....3.33}$$

where $f(y|\theta)$ is the likelihood of observation given prior probability of model parameters, and $\pi(\theta)$ is the prior probability of model parameters. These correspond to Eq. 3.31 and Eq. 3.32, respectively. Combining our equations as components of the numerator:

$$f(y|\theta)\pi(\theta) = \prod_{i=1}^{n} \mathcal{N}\left(y_i|[t_i, m_i, b_i], \sigma^2\right)\mathcal{N}\left(m_i|t_i, \lambda^{-1}\right)$$

Or, writing out the Gaussian distributions:

$$f(y|\theta)\pi(\theta) = \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} \exp\left[\frac{-\left(y_i - [t_i m_i + b_i]\right)^2}{2\sigma^2}\right] \frac{1}{\sqrt{\lambda^{-1}}\sqrt{2\pi}} \exp\left[\frac{-1}{2}\frac{m_i^2}{\lambda^{-1}}\right]$$

Taking the negative logarithm, which we will refer to as "logp" here and throughout this document:

$$-\ln f(y|\theta)\pi(\theta) = \text{logp}\left[f(y|\theta)\pi(\theta)\right] =$$

$$= \sum_{i=1}^{n} -\ln\frac{1}{\sigma\sqrt{2\pi}} + \frac{\left(y_i - [t_i m_i + b_i]\right)^2}{2\sigma^2} - \ln\frac{1}{\sqrt{\lambda^{-1}}\sqrt{2\pi}} + \frac{1}{2}\frac{m_i^2}{\lambda^{-1}}$$

The first and third terms are simply constants which we group together. We also invert the lambda term, giving:

$$\text{logp}\left[f(y|\theta)\pi(\theta)\right] = \sum_{i=1}^{n} \frac{\left(y_i - [t_i m_i + b_i]\right)^2}{2\sigma^2} + \frac{\lambda}{2} m_i^2 + \text{constant}$$

Last, we can move the ½ on each term to the constant:

$$\text{logp}\left[f(y|\theta)\pi(\theta)\right] = \sum_{i=1}^{n} \frac{\left(y_i - [t_i m_i + b_i]\right)^2}{\sigma^2} + \lambda m_i^2 + \text{constant} \quad \text{..... 3.34}$$

Minimization of Eq. 3.34 is equivalent to minimization of squared error, i.e. least squares regression, with L2 regularization on *m*. Clarifying, we may impose a Gaussian prior on any parameter of the Bayesian model to perform L2 regularization on that parameter *without* needing to reconstruct a matrix to solve a system of linear equations. In addition, we may impose a prior distribution of any form just as easily as a Gaussian distribution. Significantly, this enables the use of L1 regression as opposed to L2. The advantage L1 regression is that L1 fit is less affected by outliers than the L2 fit. This makes our deconvolution algorithm able to handle a greater degree of noise as compared with other models in the petroleum literature. Errors as high as 40% – 60% have still yielded reasonable results on test cases.

In the Bayesian framework, an L1 cost function corresponds to a Laplace distribution:

$$\mathcal{L}(\mu,b)=\frac{1}{2b}\exp\left[\frac{-|x-\mu|}{b}\right] \quad \text{.....3.35}$$

And the negative log probability is:

$$\text{logp}\left[\mathcal{L}(\mu,b)\right]=\frac{|x-\mu|}{b}+\text{constant} \quad \text{.....3.36}$$

Which is the absolute deviance. Hence, the Laplace prior is equivalent to minimization of the sum of absolute deviance of residuals as well as an L1 regularization:

$$\text{logp}\left[\mathcal{L}\left(\mu,\lambda^{-1}\right)\right]=\lambda|x-\mu|+\text{constant} \quad \text{.....3.37}$$

Other probability distributions of interest include the Half Normal:

$$\mathcal{HN}(\sigma^2) = \frac{\sqrt{2}}{\sigma\sqrt{\pi}} \exp\left[\frac{-\mu^2}{2\sigma^2}\right] \quad \text{for } x > 0 \qquad \text{.....3.38}$$

And the Exponential:

$$\mathcal{E}(\lambda) = \frac{1}{\lambda^{-1}} \exp\left[\frac{-x}{\lambda^{-1}}\right] \quad \text{for } x > 0 \qquad \text{.....3.39}$$

Which we have written to make it obvious these are equivalent to one-half of a Normal and Laplace distribution, respectively. In practice, we may use any distribution for which the logarithm yields a function that has an existent derivative. Then, for any proposed parameter step, the change in total model probability is simply the change in the probability of the distributions that are a function of that parameter. Given that the total model probability is proportional to the sum of all probabilities, we may find the partial derivative of the total model probability with respect to any model parameter. The recognition of this property is powerful and provides us with a means to quickly regress a model on high-dimension, non-linear problems.

3.4 Hamiltonian Monte Carlo

Markov Chain Monte Carlo simulation is a numerical technique used to solve inverse model problems. The method performs random walks over model parameters at each iteration, computes the model as a proposal, and then accepts or rejects the proposal. Each iteration improves the approximate distribution used for each parameter. Given enough iterations, the method will converge to the target distribution. Formally, we draw proposal values of θ^* from approximate distributions θ, compute the likelihood of the model $P(y|\theta)$

based upon the observations, and then correct θ^* to better approximate the target posterior distribution $P(\theta|y)$.

In the simplest case, the updates of each parameter at each iteration are done randomly. For simple problems composed of one parameter, this may work well. However, several challenges present themselves—even when the number of parameters is increased to only two—if the correlation of the parameters cannot be neglected. Or, for some greater number of parameters, even if the parameters may be assumed to be independent. The primary difficulty is that, when a proposal is rejected, there is no information on which parameter(s) may have caused the rejection. Thus, a single parameter may lead to rejection in a proposal, and the likelihood of this behavior occurring increases quadratically as the parameter count increases.

The properties of an efficient sampling algorithm are (Gelman et al. 2014):

- For any θ, it is easy to sample from $J(\theta^*|\theta)$.
- It is easy to compute the acceptance ratio of an iteration.
- Each parameter jump goes a reasonable distance in the parameter space (otherwise the random walk moves too slowly).
- The jumps are not rejected too frequently (otherwise the random walk wastes too much time standing still).

Our problem is, by its nature, high-dimensional and non-linear. Therefore, a purely random walk sampling algorithm will not perform well. Hamiltonian Monte Carlo (HMC) suppresses the random walk behavior, allowing a great increase in sampling efficiency, by

use of an analogy to Hamiltonian dynamics. A 'momentum' variable, p, is added to the 'position' of each parameter, q. Both q and p are updated together, and the jump for q is a function of p. Each jump can move rapidly throughout the parameter space while preserving the 'energy' (kinetic plus potential) of the trajectory. The combination of the two gives a joint distribution:

$$\pi(q, p|y) = \pi(p)\pi(q|y) \quad \text{.....3.40}$$

Or, defining $\pi = -\log p(P)$:

$$\pi(q, p|y) = \pi(p) + \pi(q|y) \quad \text{.....3.41}$$

The sampling is performed over the joint distribution, and the result used to determine q at each step in the simulation. The momentum variable p is given a multivariate normal distribution with mean 0 and covariance of a 'mass matrix' M. If the mass matrix is diagonal, then the components p_j are independent, i.e. $p_j \sim \mathcal{N}(0, M_{jj})$. Convention is to use $M_{jj} = 1$. The Hamiltonian equations then determine how p and q change as we 'step' in time:

$$\frac{dq}{dt} = \frac{\partial H}{\partial p} \quad \text{.....3.42}$$

$$\frac{dp}{dt} = -\frac{\partial H}{\partial q} \quad \text{.....3.43}$$

We combine these into functions written as:

$$H(q, p) = U(q) + K(p) \quad \text{.....3.44}$$

Written this way, we have the potential energy, $U(q)$, and the kinetic energy, $K(p)$. We define them as:

$$U(q) = \pi(q|y) \quad \text{.....} 3.45$$

$$K(p) = \frac{1}{2} p^T M^{-1} p \quad \text{.....} 3.46$$

Substituting Eqs. 3.45 & 3.46 into the preceding equations, we can re-write Eqs. 3.42 & 3.43 as:

$$\frac{dq}{dt} = M^{-1} p \quad \text{.....} 3.47$$

$$\frac{dp}{dt} = \frac{d}{dq} \pi(q|y) \quad \text{.....} 3.48$$

The gradient of K can be imagined as the velocity vector and is used to determine the distance of some number of 'leapfrog' steps. The gradient is often simple to compute analytically for most distributions; numerical differentiation is computationally inefficient. This adds a restriction that the log probabilities must be differentiable; in practice this is rarely a constraint. For example, the negative log probability of a Gaussian distribution is:

$$\pi(q|y) = \log p\left[P(q|y) = \mathcal{N}(\mu, \sigma^2) \right] = \frac{\left(q|y - \mu\right)^2}{2\sigma^2} - \ln \frac{1}{\sigma\sqrt{2\pi}} \quad \text{.....} 3.49$$

And its gradient is:

$$\frac{d}{dq} \pi(q|y) = \frac{q|y - \mu}{\sigma^2} \quad \text{.....} 3.50$$

The Hamiltonian has two important properties. First, it is reversible by negating the time derivative. Thus, use of the Hamiltonian maintains the Markov property, which is that the conditional probability distribution of futures states depends only upon the present state, and not the sequence that preceded it.

Second, the Hamiltonian is invariant (conserved). As the parameter changes value during sampling, we can imagine exchanging momentum (kinetic energy) with position (potential energy) as shown in Eq. 3.44. Taking the derivative of Eq. 3.44 with respect to time and applying the chain rule:

$$\frac{\partial}{\partial t} H(q,p) = \frac{\partial H}{\partial q}\frac{dq}{dt} + \frac{\partial H}{\partial p}\frac{dp}{dt} \quad \text{.....} 3.51$$

Substituting Eqs. 3.42 & 3.43 shows the result to be zero:

$$\frac{\partial}{\partial t} H(q,p) = \frac{dq}{dt}\frac{dp}{dt} - \frac{dp}{dt}\frac{dq}{dt} = 0 \quad \text{.....} 3.52$$

In terms of our parameter draws θ^*, this equates to the magnitude of change θ^* over the step (the derivative of the kinetic energy) and the probability of θ^* given θ (the potential energy). As θ^* approaches a state space of lower probability, the rate of change will slow, and it will eventually reverse direction. As θ^* approaches a state space of higher probability, the rate of change will increase. The effect of this behavior is that we can achieve quick and efficient sampling of the posterior distribution $P(\theta|y)$ while avoiding state space of low probability.

To solve Eqs. 3.47 & 3.48 it is typical to employ Euler's method. Discretizing time into a series of steps of size δ, we update p and q by:

$$p(t+\delta) = p(t) + \delta \frac{d}{dt} p(t) = p(t) + \delta \frac{d}{dq} \pi(q|y) \quad \text{.....3.53}$$

$$q(t+\delta) = q(t) + \delta \frac{d}{dt} q(t) = q(t) + \delta p(t) M^{-1} \quad \text{.....3.54}$$

However, Euler's method is unstable for practical step sizes and will diverge. Improvement can be made with the leapfrog method as follows:

$$p\left(t + \frac{\delta}{2}\right) = p(t) + \frac{\delta}{2} \frac{d}{dq} \pi(q|y) \quad \text{.....3.55}$$

$$q(t+\delta) = q(t) + \delta p\left(t + \frac{\delta}{2}\right) M^{-1} \quad \text{.....3.56}$$

$$p(t+\delta) = p\left(t + \frac{\delta}{2}\right) + \frac{\delta}{2} \frac{d}{dq} \pi(q|y) \quad \text{.....3.57}$$

Each iteration in HMC has three parts:

1. Initialize p with a draw from its posterior distribution, which by specification is the same as its prior distribution:

   $$p \sim \mathcal{N}(0, M)$$

2. Leapfrog steps of q and p, each scaled by a step-size factor δ, repeated L times:

   a. Use velocity $\frac{d}{dq} \pi(q|y)$ to make a half-step, $\frac{\delta}{2}$, of momentum p:

   $$p \leftarrow \phi + \frac{\delta}{2} \frac{d}{dq} \pi(q|y)$$

   b. Use the momentum p to update the position q:

   $$q \leftarrow q + \delta M^{-1} p$$

c. Use velocity $\frac{d}{dq}\pi(q|y)$ to make a final half-step, $\frac{\delta}{2}$, of momentum p:

$$p \leftarrow \phi + \frac{\delta}{2}\frac{d}{dq}\pi(q|y)$$

3. Except for the first and last steps, c. and a. are performed together in a full-step.

The pseudocode for the algorithm is:

```
function tuple(float, float) leapfrog(float q, float p,
   function dUdq, int length, float step):

   p -= step / 2.0 * dUdq(q) # first half-step

   for i in length - 1:      # begin whole steps
      q += step * p
      p -= step * dUdq(q)

   q += step * p             # last whole-step
   p -= step / 2.0 * dUdq(q) # last half-step

   return q, -p
```

An example of the leapfrog algorithm, with the positions drawn as lines and the momentum shown as arrows, is displayed in **Fig. 3.4** for a Gaussian mixture model with the following components:

$$\mathcal{N}_1\left(\begin{bmatrix}1.5\\2\end{bmatrix},\begin{bmatrix}1 & 0.8\\0.8 & 1\end{bmatrix}\right)$$
$$\mathcal{N}_2\left(\begin{bmatrix}0\\-1\end{bmatrix},\begin{bmatrix}1 & -0.6\\-0.6 & 1\end{bmatrix}\right)$$
$$\mathcal{N}_3\left(\begin{bmatrix}-1\\2\end{bmatrix},\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right) \quad \text{.....} 3.58$$

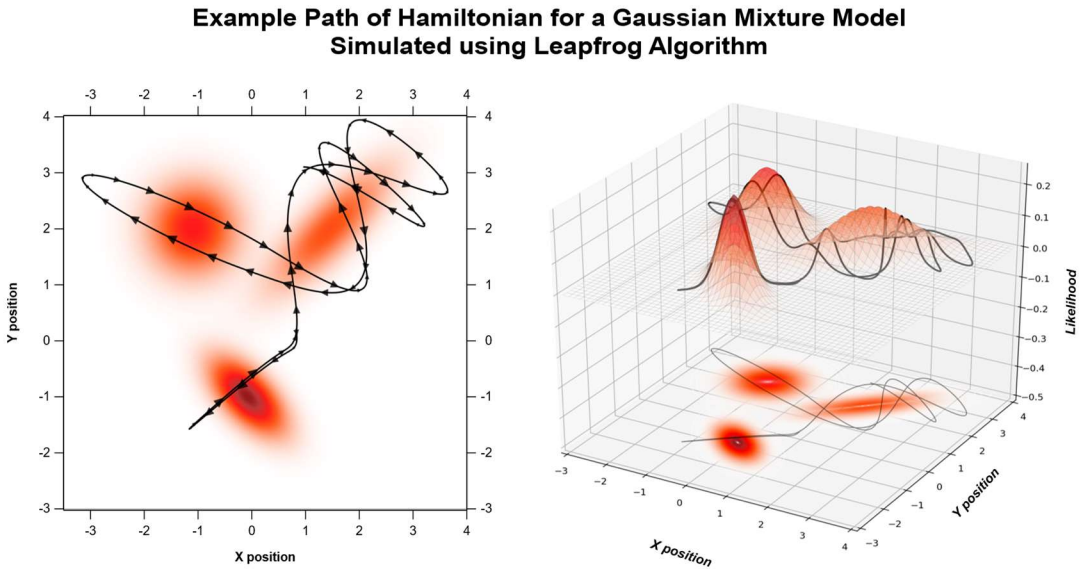

Figure 3.4 — Numerical simulation of Hamiltonian equations for a Gaussian mixture model.

The leapfrog algorithm will continue to conserve the Hamiltonian for as long as it is run, so we must set the length of the path, and then evaluate the returned position. The last step for HMC is to accept or reject a drawn sample.

The acceptance ratio is computed as:

$$r = \frac{P(q^*|y)P(p^*)}{P(q^{t-1}|y)P(p^{t-1})} \quad \text{.....3.59}$$

Or:

$$\log r = \left[\pi(q^*|y) + \pi(p^*)\right] - \left[\pi(q^{t-1}|y) + \pi(p^{t-1})\right] \quad \text{.....3.60}$$

Last, we set the value of q^t based on the acceptance ratio r:

$$q^t \leftarrow \begin{cases} q^* & \text{with probability } \min(r,1) \\ q^{t-1} & \text{otherwise} \end{cases} \quad \text{.....3.61}$$

The pseudocode for the algorithm is written as:

```
function float hmc(float q0, float logp, int length,
   float step, int n_steps):

   q = q0
   dUdq = gradient(logp)           # auto-grad function

   for i in n_steps:
      p0 = std.norm.rand()         # intialize momentum
      q, p = leapfrog(q, p0, dUdq, lenth, step)

      prior_logp = -logp(q0) - logp(p0)
      sample_logp = -logp(q) - logp(p)
      r = sample_logp - prior_logp # acceptance ratio

      if log(unif.rand(0, 1)) < r:
         append(accepted, True)
      else:
         append(accepted, False)
         q = q0

   return q
```

Fig. 3.5 illustrates HMC on the same data as in **Fig. 3.4**, but without rejection of steps. The blue markers indicate the end of a step, after which a new momentum is sampled and the velocity changes.

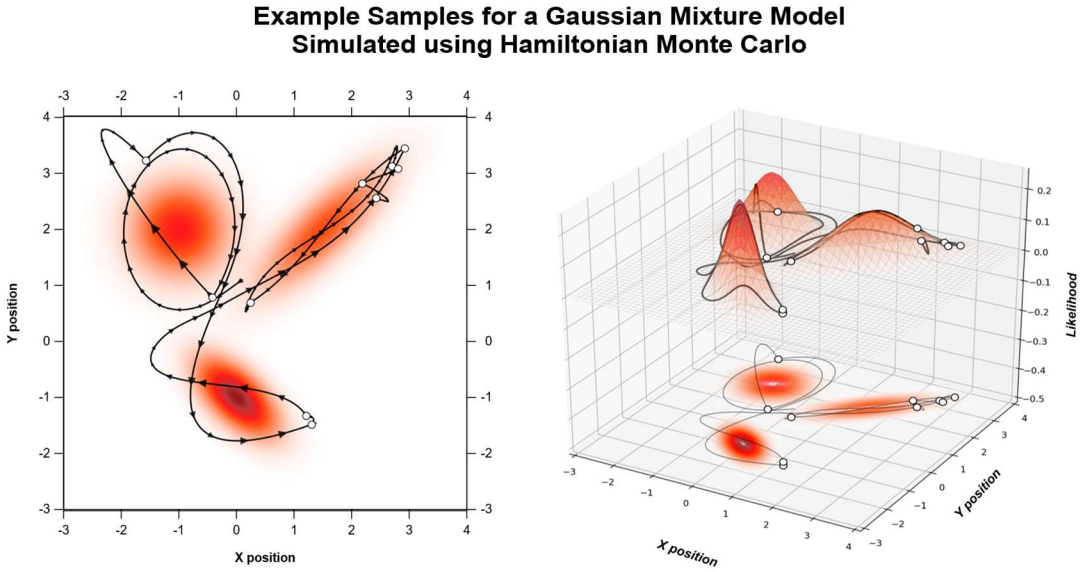

Figure 3.5 — Numerical simulation of Hamiltonian Monte Carlo for a Gaussian mixture model.

We apply HMC to sample the kernel functions from the posterior distribution. We apply priors to each model parameter, and likelihoods to observed variables such as the residuals of the computed flow rates and sandface flowing pressures, as well as the residuals from the convolution of each kernel function and Duhamel's principle. We note that the sandface rates and pressures are both model parameters and used as observed data. By building a model-based representation, we can impute any missing. We can do the same for initial reservoir pressure; it is expressed as a bounded random variable. No discrete initial guess of initial pressure is required as the model will impute or correct the value unless it is specifically treated as a constant. These concepts extend to imputing the constant-pressure-drop rate function when performing variable-rate deconvolution to solve for the constant-rate pressure drop function, or vice versa. We also may perform both

deconvolutions simultaneously as an improvement upon the "double deconvolution" method introduced by Ilk, Valkó, and Blasingame (2007).

HMC is implemented by PyMC3 (Salvatier, Wiecki, and Fonnesbeck 2016). PyMC3 is a Python library that implements the probabilistic programming paradigm. The library provides a framework for representation of variables as probability distributions or results of computations performed on other variables, and whether each variable is observed or not. Primarily, it allows flexible specification of the model as clear expression of symbolic variable relationships.

3.5 Computational Graphs

For HMC to operate, the gradient of the target distribution $\pi(\theta|y)$ must be known. The sequence of operations that occur to compute the probability must be considered, starting with the Gaussian processes, continuing with the convolution operations, and ending with the cost function that compares each predicted value with an observed value. A computational graph represents the mathematical operations performed within the model. Storing the operator along with the variables that are operated upon enables automatic differentiation of any variable z in the model with respect to any other variable x that has been used to compute z.

PyMC3 implements Theano (Bergstra et al. 2010) to perform these calculations. Theano is a Python library that focuses on efficient evaluation of n-dimensional arrays, referred to as tensors. Symbolic representations of a model are constructed in Python, and then Theano may be invoked to compile the model directly to C code for evaluation on a CPU

or GPU. Theano performs various checks and simplifications of the model before compilation, such as combining duplicate expressions, improving the numerical stability of computations, replacing operations with faster implementations that may be more specialized, before finally compiling C code. The result is highly efficient numerical computation constructed through user-exposed high-level symbolic representation.

An example of a Theano computational graph is shown by **Fig. 3.6**. The green boxes are inputs, and red ellipse are operators to be applied, and the blue boxes are outputs. The numbers by each arrow represent the order of inputs to each operator. We can understand the left side of **Fig. 3.6** as an element-wise addition of two zero-dimensional vectors, i.e. scalars. The right side outputs the gradient of the addition operator. The gradient computation always starts with the inputs and operator, and performs additional operations to return the gradient. In this case, the operator `Elemwise{second,no_inplace}` takes the second input (index one) and drops the first (index zero). The output will be a scalar value equal to one.

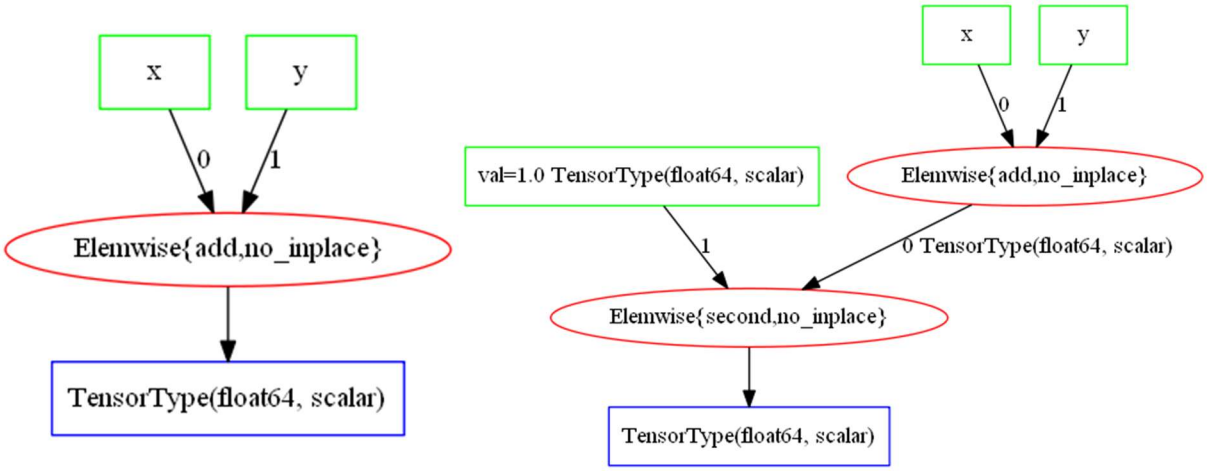

Figure 3.6 — Example of computational graph for addition operator and its gradient.

A computational graph for a power operator is given in **Fig. 3.7**. This graph highlights the repeated application of the chain rule. Tracking the computations starting with the input x, we see the original operation repeated. The power operator and inputs (x, 2) are then dropped and replaced by a scalar value of one, and then this value is multiplied by the original exponent of two, repeating the steps involved in application of the Power rule. We would write this as:

node 1	=[	x,	2,	power	]≡	x^2
node 2	=[	node 1,	1,	second item	]≡	1
node 3	=[	node 2,	1,	multiply	]≡	1×2

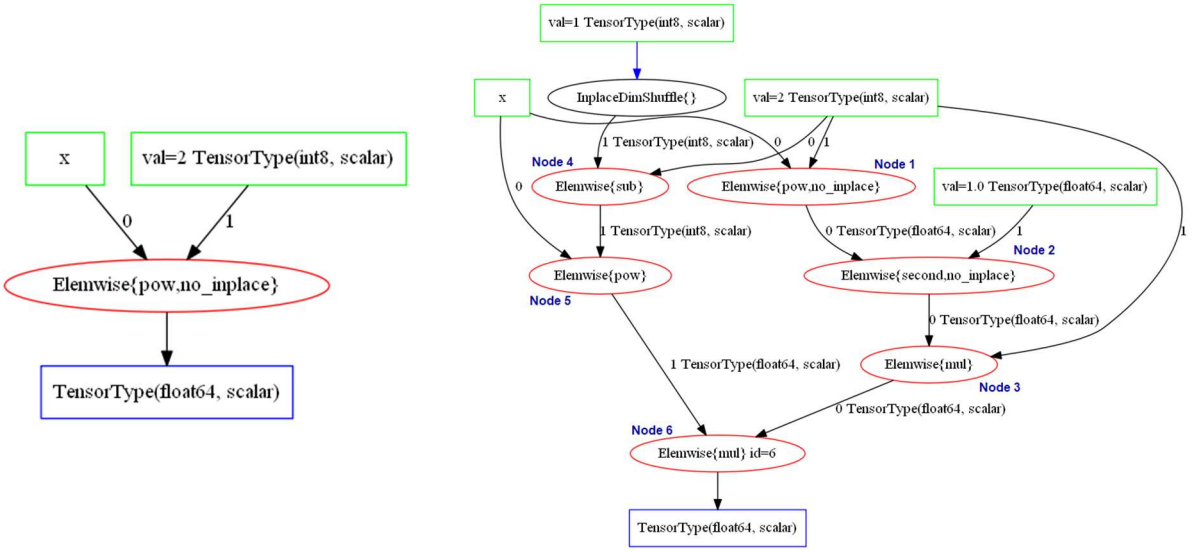

Figure 3.7 — Example of computational graph for power operator and its gradient.

On the left side the first operation subtracts one from the exponent, then applies a power function with the new value to x. The result is multiplied by the last node on the right side. Writing this out, we have:

node 4	=[	2,	1,	subtract	] ≡	$2-1$
node 5	=[	x,	node 4,	power	] ≡	x^1
node 6	=[	node 3,	node 5,	multiply	] ≡	$2x$

And node 6 returns the correct value. Following this approach, a gradient is defined for every operation. Building a computational graph of the gradient becomes a set of lookups applied to each operator. Given that the computational graph can be walked backward from the final output back to the initial inputs, and that the initial inputs will always be given as constants, the gradient of model is known for any input. For our use case, the significance is that we always know the change in the probability of the entire model when

determining how to step any individual parameter. Thus, our optimization scheme does not rely upon numerically computed gradients, learning rates such as in a gradient descent algorithm, or precision thresholds to judge convergence. Instead, we can use the gradient information to directly sample from the posterior distribution.

3.6 Probabilistic Graphical Models

Although many domain-specific algorithms have been written for the problem of deconvolution, in practice they are quite rigid. Each solution is formulated for a specific subset or class of deconvolution problem and tends to require significant changes from one problem class to the next. In contrast, probabilistic graphical models are a general framework that allow a computer to reason, or infer parameter values, based upon a model of the system that encodes our knowledge of how the system works. "The key property of the model is the separation of knowledge and reasoning. The representation has its own clear semantics, separate from the algorithms that one can apply to it." (Koller and Friedman 2009).

For many problems and their corresponding models, it is often the case that an observation of a variable is fixed, and the relationship between variables is probabilistic. For our problem and many science and engineering problems, the exact opposite is true. The relationships between our variables is fixed to the extent of the assumptions used to derive the relationship, while the data we observe is probabilistic in the sense of noise, error, or incompatibility with a model that is over- or under-specified, thus giving rise to some

hyperdistribution of residuals. Still, the same principles of PGM apply to both cases. The relationship of two variables in the model is given by the conditional probability:

$$P(\mathrm{B} \mid \mathrm{A}) = \frac{P(\mathrm{A} \bigcap \mathrm{B})}{P(\mathrm{A})} \quad \text{.....} 3.62$$

Therefore, if only one possible value of A is possible, then B must follow. However, given that A itself is uncertain, then B must also be uncertain, even if it exactly follows from A. We formally write this as the chain rule of conditional probabilities:

$$P(\mathrm{A} \bigcap \mathrm{B}) = P(\mathrm{A})P(\mathrm{B} \mid \mathrm{A}) \quad \text{.....} 3.63$$

And we may expand this out:

$$P(\alpha_1 \bigcap \ldots \bigcap \alpha_n) = P(\alpha_1)P(\alpha_2 \mid \alpha_1)P(\alpha_3 \mid \alpha_2 \bigcap \alpha_1)\ldots P(\alpha_n \mid \alpha_{n-1} \bigcap \alpha_1) \quad \text{.....} 3.64$$

The result being that the probability of any value in the model can be put in terms of the probabilities of all other terms, with the other terms occurring in any order. We also find Bayes' rule follows from the conditional probability:

$$P(\mathrm{A} \mid \mathrm{B}) = \frac{P(\mathrm{B} \mid \mathrm{A})P(\mathrm{A})}{P(\mathrm{B})} \quad \text{.....} 3.65$$

Which states that we may determine the conditional probability of A from the expression of the conditional probability of B given A. In our case, we may determine the conditional probability of the unknown constant-rate pressure or constant-pressure-drop rate function given the observation of the sandface rates and pressures. Going further, we may determine the conditional probability of missing sandface rates and pressures, and initial reservoir pressure, given our observed variables. We can also make a direct statement about the

probability of our unknown functions, such as a diagnosis of a specific flow regime at a certain time, and it will influence all other variables in the model. The model structure given in **Fig. 3.8** is general for any deconvolution problem. It encodes our knowledge of the convolution integral as derived within the literature. In every step of our HMC algorithm, every variable is computed based upon this knowledge. Then, the model *as an entirety* is accepted or rejected given the observed data for each variable. If we choose to neglect a specific part of the model, such as to focus solely on the variable-rate problem, then we may simplify our model as in **Fig. 3.9**.

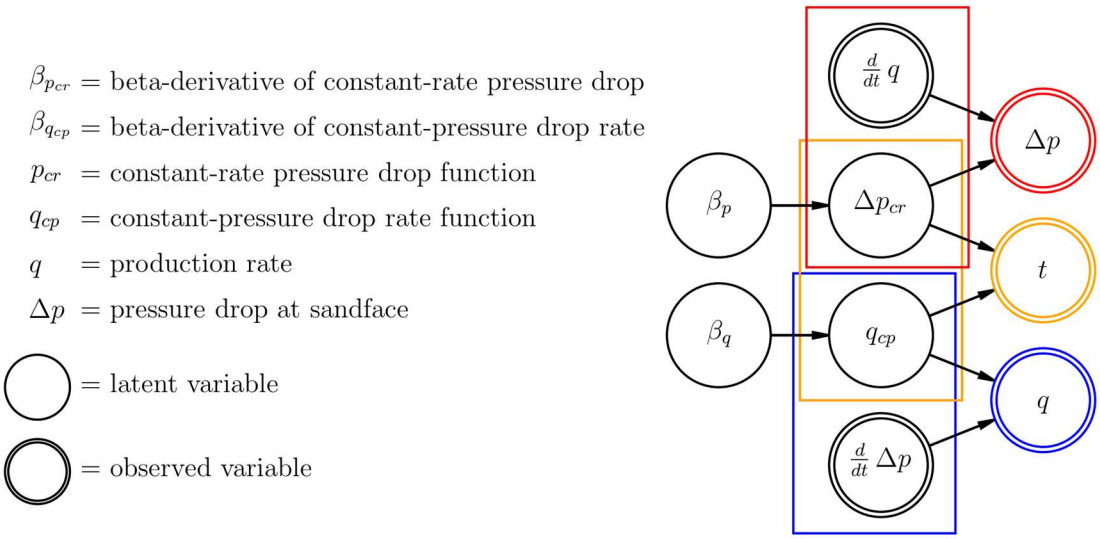

Figure 3.8 — The probabilistic graphic model (PGM) encodes the variable relationships from our knowledge of the system.

The reduced models for these cases clearly show a direct linear path to invert for the unknown pressure function derivative $\beta_p(t)$ as in previous direct solution methods. We

may also simultaneously yet independently solve the variable-rate and variable-pressure drop cases by not treating the time variable t as unobserved as shown in **Fig. 3.10**. Enforcing the correlation of Δp_{cr} and q_{cp} by use of Duhamel's principle leads us back to **Fig. 3.8**.

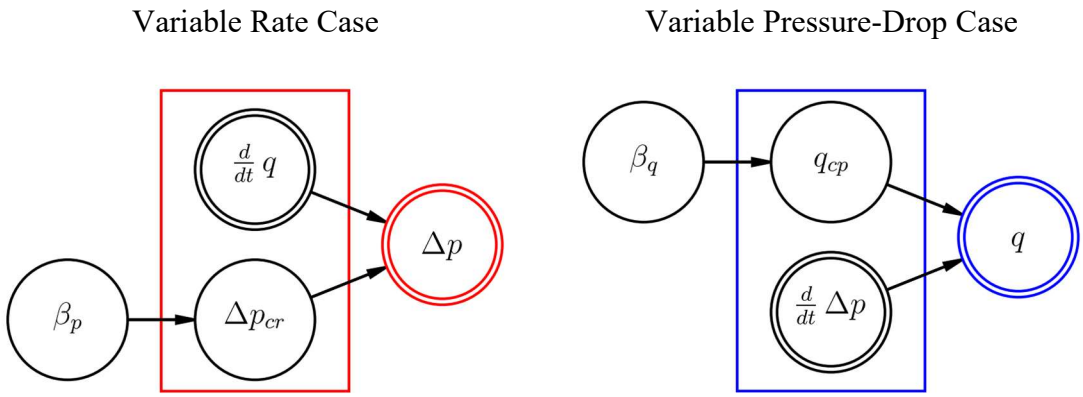

Figure 3.9 — Variable rate and variable pressure-drop cases as reductions of the base model.

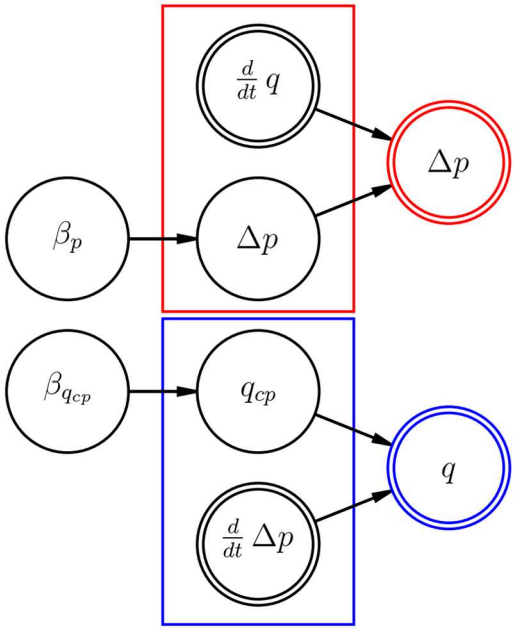

Figure 3.10 — Variable rate and variable pressure-drop cases may be generated simultaneously and independent of one another.

We may extend the concept of Bayes' Rule to the multiwell case (**Fig. 3.11**) as a hierarchical model. Rather than a multiwell case in which the wells are interfering with one another within a single reservoir, we consider a case in which the wells are non-interfering. If we assume that each well shares the same kernel function, then we may treat each well as a repeated observation of the convolution of the kernel function with the observed rates or pressures. We illustrate this concept in **Fig. 3.12**, where the kernel function (green) is simultaneously convolved with each well (blue).

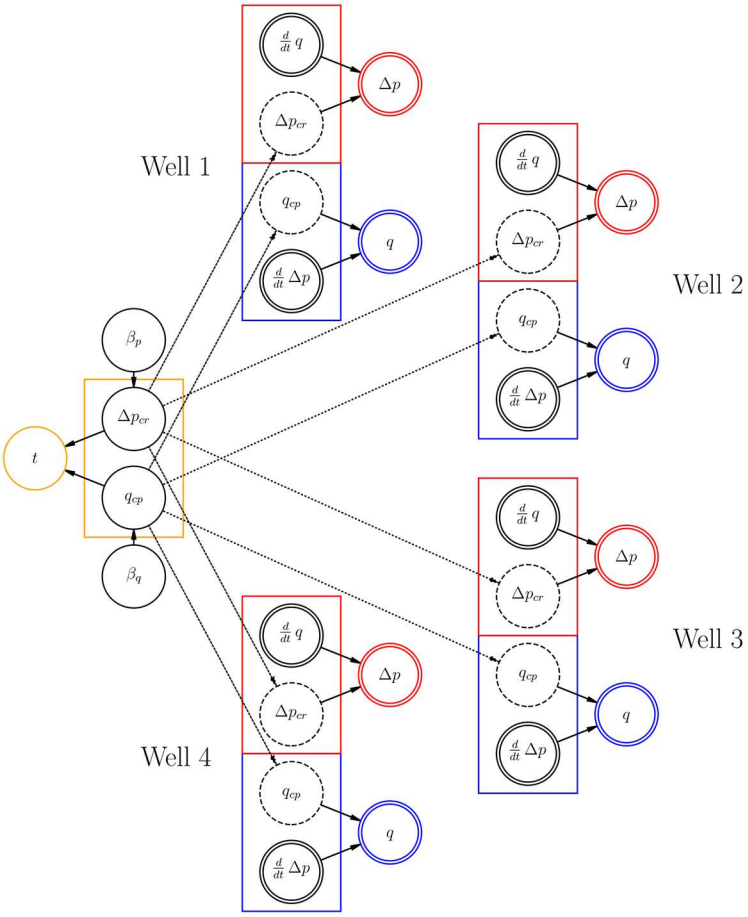

Figure 3.11 — Multi-well deconvolution as a hierarchical model with the kernel function convolved with rates or pressures from each well .

The validity of this assumption remains to be seen, but it is similar in concept to that of rate-transient analysis workflows in which an evaluator builds a reservoir model based upon a limited set of high-quality data, and then applies that model to wells where they have less data on hand. In both cases, there is an assumption that a same or similar reservoir model applies. In our algorithm, differences in the model are accounted for by the

probabilistic representation; that is, it need not be exactly true that the same kernel function applies to all observed data, only approximately true.

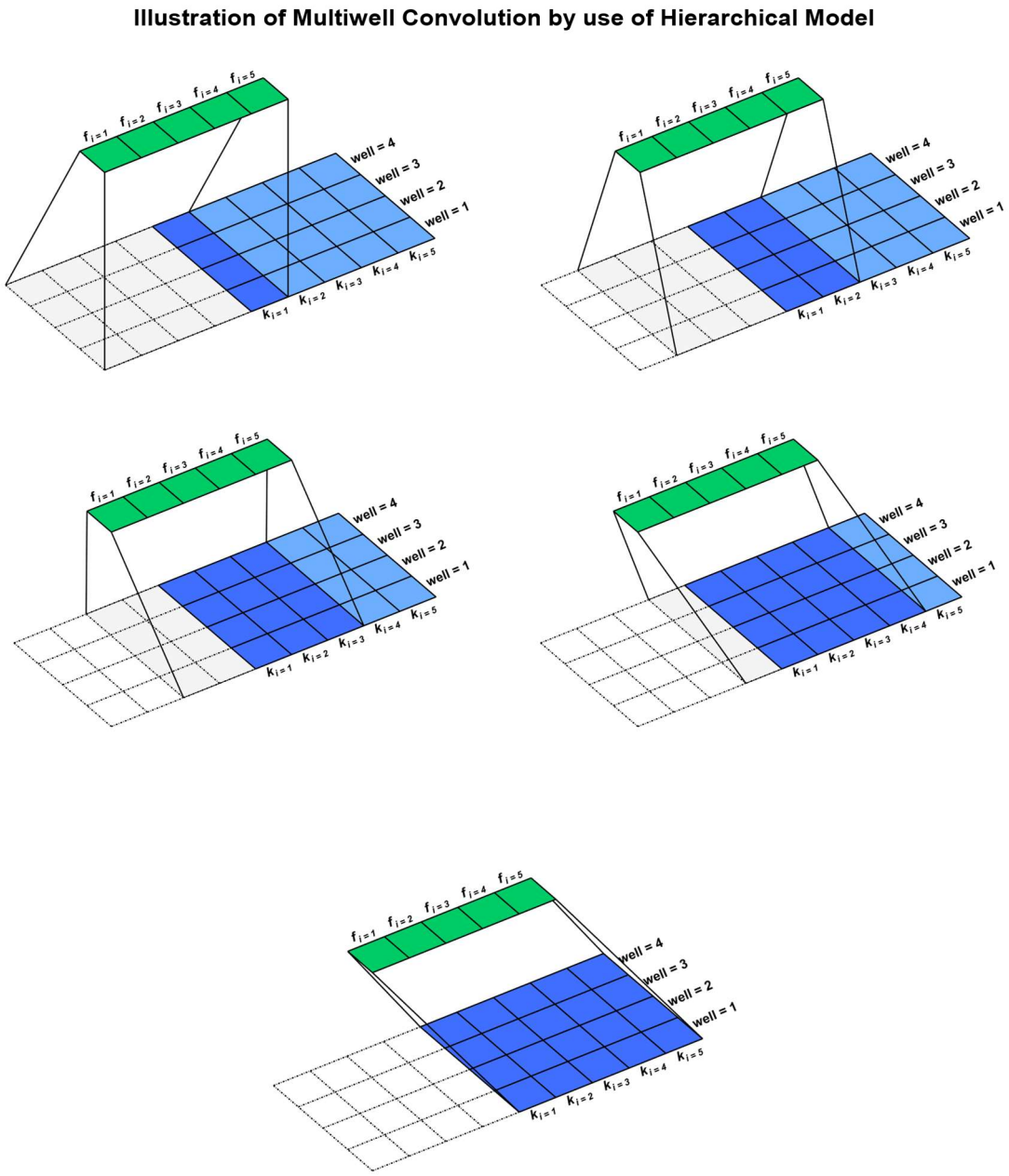

Figure 3.12 — Illustration of multi-well deconvolution.

3.7 B-splines

We have mapped the relationship between our parameters with the probabilistic graphical model, but some method of computation remains to be implemented. For the convolution operations, the implementation is obvious as a discrete convolution sum. However, we have some choice in implementation of a numerical integration scheme to compute the pressure function from $\beta_p(t)$.

B-splines give several advantages; namely, they are defined on the discrete intervals of the knots while still being continuous between the knots. B-splines also have continuous derivatives and integrals. Practically, this enables us to simultaneously integrate and interpolate $\beta_p(t)$ as a degree k spline defined over the knots, to a degree $k+1$ spline defined over the time intervals of the observed data. The i-th degree 0 B-spline for data points x and knots t^* is defined by the de Boor algorithm as:

$$B_i^0(x,t^*) = \begin{cases} 1 & t^*_i \le x < t^*_{i+1} \\ 0 & \text{otherwise} \end{cases} \quad \text{.....3.66}$$

And any higher-degree B-spline is defined recursively:

$$B_i^k(x,t^*) = \frac{x - t^*_i}{t^*_{i+k} - t^*_i} B_i^{k-1}(x,t^*) + \frac{t^*_{i+k+1} - x}{t^*_{i+k+1} - t^*_{i+1}} B_{i+1}^{k-1}(x,t^*) \quad \text{.....3.67}$$

Fig. 3.13 illustrates B-splines of various degrees. Importantly, we note that any set of B-splines forms a matrix, and the spline function S is a linear combination of B-splines numbering the length of the interpolating grid, with each B-spline of length $k+n+1$, where n is defined as the number of knots t^*:

$$S(x,t^*)=\sum_{i=1}^{n} c_i B_i^k\left(x,t^*\right)=\mathbf{B}^k\mathbf{c} \quad \text{.....3.68}$$

or

$$\mathbf{S}(x,t^*)=\mathbf{B}^k\mathbf{c} \quad \text{.....3.69}$$

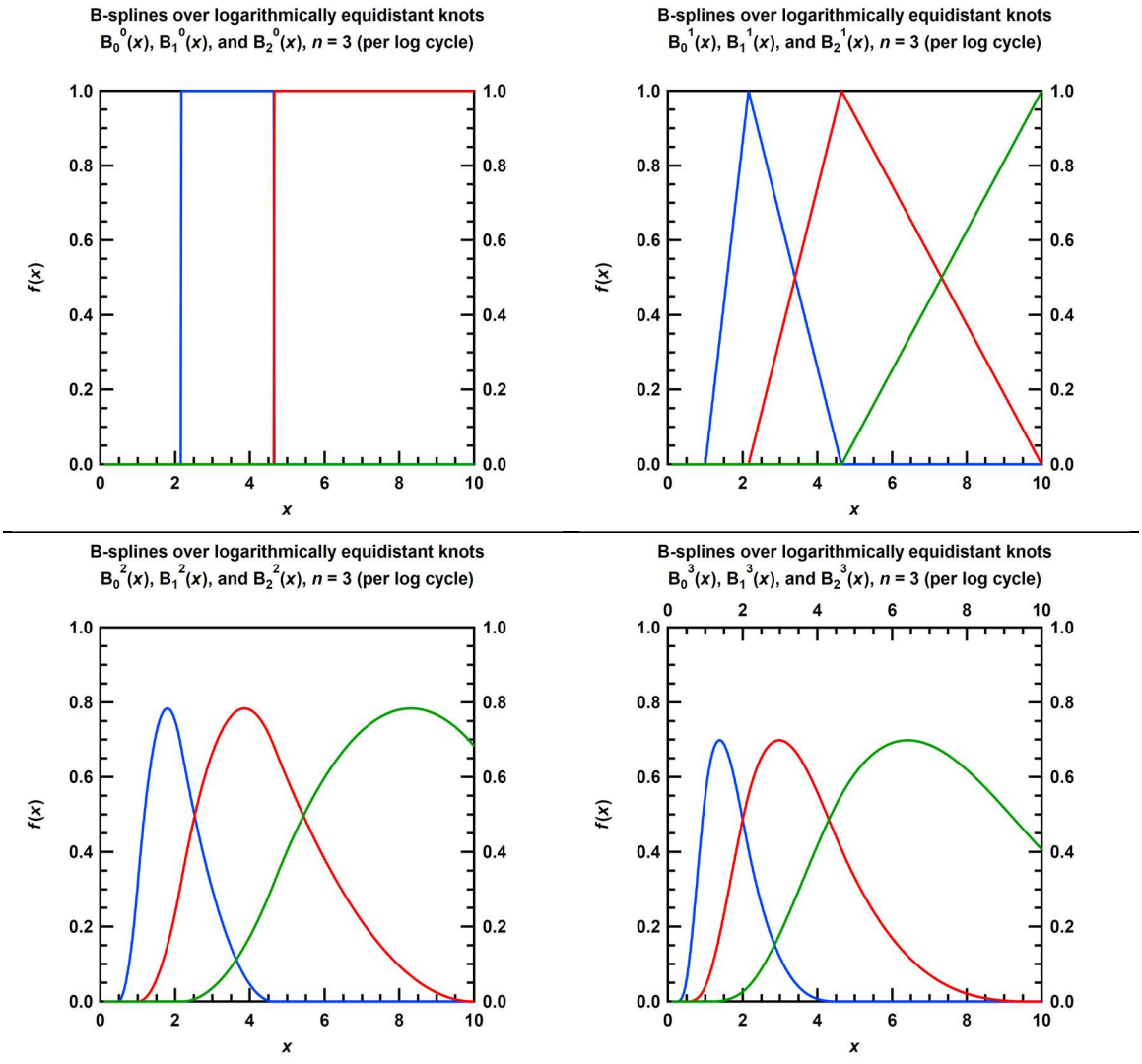

Figure 3.13 — B-splines of various degrees for n = 3 knots placed over the interval (0, 10).

While this formulation differs from prior work, it allows us to generate any number of B-splines prior to simulation and avoid the computational penalty of doing so during the simulation. Thus, we generate the following sets of B-splines to 1) interpolate over the knots t^*, 2) interpolate over the data time values t, and 3) integrate *and* interpolate over the data time values:

$$S1_i = B_i^k\left(t^*, t^*\right) \quad \text{3.70}$$

$$S2_i = B_i^k\left(t, t^*\right) \quad \text{3.71}$$

$$S3_i = B_i^{k+1}\left(t, t^*\right) \quad \text{3.72}$$

The set of splines *S1* are used in Eq. 3.69 to solve for the coefficients. Special cases of solution for the coefficients exists for degree 0 and degree 1 B-splines. From Eq. 3.66 we have for degree 0 B-splines:

$$B_i^0(t^*_j, t^*_j) = \delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \quad \text{3.73}$$

And the solution for the coefficients would just be the values of the function for which we are fitting the B-spline $\beta_p(t)$:

$$\mathbf{S}(t^*, t^*) = \boldsymbol{\beta}_\mathbf{p} = \mathbf{B}^0\mathbf{c} \quad \text{3.74}$$

$$\mathbf{c} = \boldsymbol{\beta}_\mathbf{p} \quad \text{3.75}$$

For degree 1 B-splines using Eq. 3.67, we have:

$$B_{i-1}^1(t^*_j, t^*_j) = \delta_{ij} \quad \text{3.76}$$

The subscript i - 1 comes from the fact that we must have k knots preceding the both first value and succeeding the last value of the data to which we are fitting the spline function. The coefficients are then the same as in the degree 0 case:

$$\mathbf{S}(t^*, t^*) = \boldsymbol{\beta}_\mathbf{p} = \mathbf{B}^1_{i-1}\mathbf{c} \quad \text{.....3.77}$$

$$\mathbf{c} = \boldsymbol{\beta}_\mathbf{p} \quad \text{.....3.78}$$

Continuing using Eq. 3.67 recursively, we can no longer reduce the solution for the coefficients but must solve the system of linear equations. Setting $\mathbf{S} \equiv \boldsymbol{\beta}_\mathbf{p}$, and dropping the superscript k for simplicity, we re-arrange Eq. 3.69 to solve for $\mathbf{c}$:

$$\mathbf{c} = \mathbf{B}^{-1}\boldsymbol{\beta}_\mathbf{p} \quad \text{.....3.79}$$

We desire a matrix $\mathbf{A}$ such that:

$$\mathbf{c} = \mathbf{A}\boldsymbol{\beta}_\mathbf{p} \quad \text{.....3.80}$$

and

$$\mathbf{BAB} = \mathbf{B} \quad \text{.....3.81}$$

If $\mathbf{B}^k$ were square and non-singular then $\mathbf{B}^{-1}$ would satisfy these conditions. However, for certain degrees k, $\mathbf{B}^k$ may not be square. For a general solution, we must find a generalized inverse matrix, or pseudoinverse, notated by $\mathbf{B}^+$. To compute the pseudoinverse, we may use singular value decomposition:

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\mathbf{T} \quad \text{.....3.82}$$

The $(m \times n)$ $\mathbf{A}$ decomposes into $(m \times m)$ $\mathbf{U}$, $(m \times n)$ $\boldsymbol{\Sigma}$, and $(n \times n)$ $\mathbf{V}^\mathbf{T}$. The pseudoinverse $\mathbf{A}^+$ of $\mathbf{A}$ is then:

$$\mathbf{A}^{+} = \mathbf{V}^{T}\mathbf{\Sigma}^{+}\mathbf{U} \quad \text{3.83}$$

where the pseudoinverse $\mathbf{\Sigma}^{+}$ is given by taking the reciprocal of the non-zero elements of the diagonal matrix $\mathbf{\Sigma}$, and then transposing the matrix.

To eliminate out-of-range errors in our arrays, our implementation pads the knots and data values with k+1 additional knots. This adds additional coefficients of value zero on both ends of each B-spline, except for the case of $k=0$ in which there is only a preceding value of zero. As a result, the size of our B-spline matrix is always $(m \times m+k+1)$. The pseudocode to generate the B-splines is:

```
function float[] add_edges(float[] x, int k):
# extend with k+1 knots
   for i in k + 1:
      x = x.prepend(min(x) * 0.99)
      x = x.append(max(x) * 1.01)
   return x

function float[] B(float x, int j, float[] t, int k):
   # compute value of Bspline recursively
   if k == 0:
      if t[j] <= x < t[j + 1]:
         return 1.
      else:
         return 0.

   c1 = (x - t[i]) / (t[i + k] - t[i]) * \
        B(x, k - 1, i, t)
   c2 = (t[i + k + 1] - x) / (t[i + k + 1] - t[i + 1] * \
        B(x, k - 1, i + 1, t)

   return c1 + c2
```

```
function float[] Bsplines(float[] x, float[] t, int k):
   # build Bspline matrix
   x = add_edges(x, k)
   m = len(x)
   n = m - k - 1
   b_splines = Array.zeros(m, n)

   for i in m:
      for j in n:
         b_splines[i, j] = B(x[i], j, t, k)

   return b_splines
```

For our application, given that we expect the pressure function to be a stepwise power-law function and a constant flow-regime behavior is represented by a constant value of $\beta_p(t)$, we represent $\beta_p(t)$ as a degree 1 B-spline, which is integrated with respect to the log of time. Recalling Eq. 3.4:

$$\Delta p_{cr}(t_n) = \Delta p_1 \exp \sum_{i=2}^{n} \beta_{p,i} \tau_i \quad \text{.....} \quad 3.4$$

Substituting in our representation of $\beta_p(t)$ as a B-spline over knots τ:

$$\Delta p_{cr}(t) = \Delta p_1 \exp \sum_{i=2}^{n} \left(\mathbf{B}^k \mathbf{c}\right)_i \quad \text{.....} \quad 3.84$$

The relations for the integral of a B-spline are given by Cheney and Kincaid (2003):

$$\int_{-\infty}^{x_m} B_i^k(x,t^*)\,dt^* = \frac{x_{i+k+1} - x_i}{k+1} \sum_{j=i}^{m} B_j^{k+1}(x,t^*) \quad \text{.....3.85}$$

Where m is the index of the upper limit of integration, and both sides are zero for any $x < t^*_i$. The integral of the spline function S, applying the relation in Eq. 3.85, is:

$$\int_{\infty}^{x_m} S(x,t^*)\,dt^* = \int_{-\infty}^{x_m} \sum_{i=m-k}^{m} c_i B_i^k(x,t^*)\,dt^* = \sum_{i=m-k-1}^{m} e_i B_i^{k+1}(x,t^*) \quad \text{.....3.86}$$

where

$$e_1 = 0, \quad e_i = \frac{1}{k+1} \sum_{i=-\infty}^{i} c_i \left(x_{i+k+1} - x_i\right) \quad \text{.....3.87}$$

As the starting index of i is reduced by one, the corresponding value of x must be less than the first knot, meaning the first coefficient e_{m-k-1} must be zero.

Using the identity in Eq. 3.78, $\mathbf{c} = \boldsymbol{\beta}_\mathbf{p}$, and the identity in Eq. 3.86 and applying to Eq. 3.84 gives us:

$$\Delta p_{cr}(t) = \Delta p_1 \exp\left[\mathbf{B}^{k+1}\mathbf{e}\right] \quad \text{.....3.88}$$

Which is equivalent to evaluation of the definite integral over the log of time:

$$\Delta p_{cr}(t) = \Delta p_1 \exp \int_{\ln t_i}^{\ln t} \mathbf{B}^k \boldsymbol{\beta}_\mathbf{p} \quad \text{.....3.89}$$

The pseudocode to compute Eq. 3.89 using Eq. 3.86 and Eq. 3.87 is:

```
function float[] integrate_coef(float[] x, float[] c,
  int k):
  # compute coefficient of k + 1 degree splines

  n = len(c)
  x = add_edges(x, k)      # x now has length of n + k + 1
  e = Array.zeros(n + 1)

  for i in n:
    e[i + 1] = e[i] + c[i] * (x[i + k + 1] - x[i])
    e /= k + 1

  return e

function float[] integrate_spline(float[] b_splines_i,
  float[] c, float[] x, int k):
  # integrate a Bspline over x using k + 1 degree spline

  e = integrate_coef(x, c, k)
  spline_i = dot_product(b_splines_i, e)
  spline_i -= spline_i[0]

  return spline_i
```

Again, we stress that the integration is done over the log of the knots. Even with large step changes in the values of $\beta_p(t)$, the resulting Δp_{cr} tends be reasonably smooth. **Fig. 3.14** shows an example evaluation of $\beta_p(t)$ by Eq. 3.89 with values of one-half, zero, and one, with each constant over a log cycle. This would correspond to linear flow, radial flow, and pseudosteady-state flow. The extra knot and extrapolation of $\beta_p(t)$ as a constant stabilizes the trend at the edges. For illustrative purposes, Δp_1 is set equal to one.

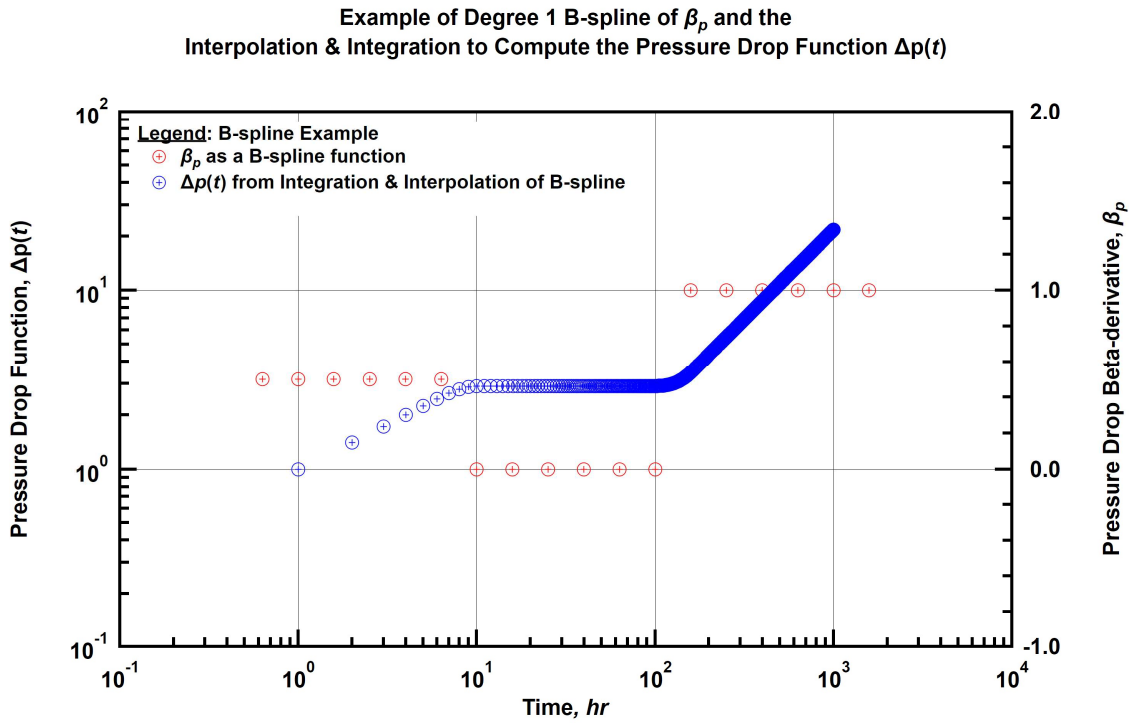

Figure 3.14 — Example of degree 1 B-spline integration and interpolation to compute pressure drop function.

The B-spline method for integration we have described is general and may be applied to any type of variable transformation to *x*- and *y*-values, not just a power-law function as we have shown. In particular, **Section 4** provides applications to computation of the well-testing derivative, and production data *q*-*D*-*b* plots. We observe that the resulting derivatives are smoother than traditional approaches, as well we have a posterior distribution of non-unique solutions that highlight periods of greater or lesser uncertainty.

As a final thought before discussing implementation, it is of interest to contrast our method with that of Ilk, Valkó, and Blasingame (2005, 2006). While both our method and their method use B-splines, there are several key differences. Primarily, we generate the beta-

derivative, $\beta_p(t)$, and use B-splines to integrate to obtain the pressure function, while their method directly solves for the coefficients of the linear derivative of the pressure function (not the beta-derivative). The advantage of using the beta-derivative is that it is invariant to scale of the pressure function, enabling implementation of a regularization scheme based on curvature of the beta-derivative that inherits the invariance to scale. The level of regularization can be determined a priori, thereby avoiding iterative computation. **Appendix E** provides an implementation of the Ilk, Valkó, and Blasingame method using our Bayesian approach, and a comparison of the results.

3.8 Huber Density Function

Prior methods presented over the past 20 years have framed the solution for deconvolution as a least squares problem using an L2 norm for error in order to invert the convolution integral and solve the resulting system of linear equations. In many applications in science, the L2 norm is chosen for computational convenience, i.e. a least squares solution has a unique, analytic solution. However, if the true distribution of residuals deviates slightly from a Gaussian distribution, then the assumption that the mean is the best representation may give significant error (Huber 1964). The L2 norm is highly sensitive to minor errors, and in practice its use requires strong regularization techniques that must be tuned uniquely for each data set under analysis. Huber proposed a piecewise cost function (aka loss function, error function, residual function, etc.) that corresponds to an L2 norm for small values of residuals, and an L1 norm for large values of residuals. The convenience of Huber's proposed cost function is that, while it required iterative calculation to solve, it did have a unique solution as it is a convex function.

The advantage of the L1 norm over the L2 is greatly reduced sensitivity to outliers and deviation from the assumed Gaussian distribution of errors, which results in much better tolerance to noise in the data (as we will show in the next section).

The Huber cost function adds a third parameter, *t*, which is the standard deviation at which the switch is made from the L2 form to the L1 form. The Huber probability density function is given by:

$$f_H\left(x \mid \mu,\sigma,t\right) = \frac{1}{2\left[\Phi^{L1} + \Phi^{L2}\right]} \exp\left[-J_H(x,t)\right] \quad \text{.....3.90}$$

where:

$$J_H(x,t) = \begin{cases} \dfrac{(x-\mu)^2}{2\sigma^2} & \dfrac{|x-\mu|}{\sigma} \le t \\ t\dfrac{|x-\mu|}{\sigma} - \dfrac{t^2}{2} & \dfrac{|x-\mu|}{\sigma} > t \end{cases} \quad \text{.....3.91}$$

$$\Phi^{L1} = \frac{\sigma}{t}\exp\left[-\frac{t^2}{2}\right] \quad \text{.....3.92}$$

$$\Phi^{L2} = \sqrt{\frac{\pi}{2}}\sigma\,\mathrm{erf}\left[\frac{t}{\sqrt{2}}\right] \quad \text{.....3.93}$$

Use of the Huber function eliminates non-unique solutions for low-error or exact cases, while maintaining the robustness of the L1 for high-error or noisy cases. We have used a value of $t = 0.1$ so that the L1 form dominates. The details for the derivation of the Huber density function are presented in **Appendix D**.

3.9 Implementation Details

We have laid out the major variables that we must compute in the prior sections, but the listing is not exhaustive of all intermediate steps. In this section, we outline the details necessary to build the model. It is important to stress that we are performing deconvolution by Bayesian *inference*, not by inversion of the convolution integral. Thus, we are not building a deconvolution model, but a convolution model. The kernel functions are inferred by a comparison of the model output (the convolved functions) with observed data. Significantly, we have an *exact* solution for the kernel functions as defined by the convolution integral. It is the addition of a model for error measurement, or in the Bayesian sense the uncertainty model, that allows inference to take place.

3.9.1 Construction of Kernel Functions

Next, we recall the equations necessary to compute the model, ordered by sequence of computation. For simplicity we will isolate the steps to the variable-rate case, noting that all steps are mirrored for the variable-pressure drop case. First, we have the computation of $\beta_p(t)$, which we simplify by assuming the GP has a mean of zero:

$$\beta_p(t) = \text{softplus}\left[\text{GP}(0, \kappa)\right] \quad \text{.....} \quad 3.94$$

The choice of kernel function completely defines $\beta_p(t)$, and the softplus function enforces positive values of $\beta_p(t)$. Each kernel function creates a different characteristic behavior of the function by encoding the covariance of data points to one another, serving as a form of regularization. We will discuss regularization later, but for now we note that the choice of kernel function greatly effects the behavior of $\beta_p(t)$.

Although a GP is an integrable function, in practice this is dependent upon the specific kernel function chosen and does not apply generally. If an analytic form of the integral does not exist, numerical methods require evaluation of the integral of the covariance matrix, which would be computationally restrictive. Instead, we take the sampled values of the GP as discrete values of the beta-derivative. While we could use the GP to directly place a prior upon the pressure function itself as opposed to the beta-derivative, doing so would not enforce monotonicity of the pressure function, nor would it provide us with the ability to easily obtain the pressure derivative by inference. The derivative of a GP is another GP, but it is defined by the partial derivative of the GP with respect to each inducing point, which once again leads us to reiterate that an analytic solution does not exist generally for all kernel functions. For these reasons, we model the beta-derivative with the GP.

Recalling our pressure function:

$$\Delta p_{cr}(t_n) = \Delta p_1 \exp \sum_{i=2}^{n} \beta_{p,i} \tau_i \quad \text{..... 3.4}$$

It is of interest to note that Eq. 3.4 is analogous to Geometric Brownian Motion if we use a white noise kernel. In this case, the GP reduces to a random walk which, following our analogy, is a Wiener process. Use of a random walk has compelling numerical advantages; namely, the computational complexity is reduced from $\mathcal{O}(n^3)$ to $\mathcal{O}(n)$. To justify the increased computational demand from using another kernel, there would need to be a great benefit from enforcing the correlation of values at distant points. This advantage does not

appear to exist, as we can enforce smoothness by other means of regularization, such as curvature. We've found that the algorithm performance is unaffected by reduction to the random walk behavior of the white noise kernel, while gaining a polynomial reduction in runtime.

We perform a numerical integration of $\beta_p(t)$ with B-splines to compute the constant rate pressure, which is the kernel function for the convolution. As shown in Section 3.7, this will give a smooth kernel function ever for large stepwise changes in $\beta_p(t)$. We take advantage of the fact that the beta-derivative is typically constant for a given flow regime, and changes smoothly and monotonically during flow regime transitions. We extrapolate the value of the $\beta_p(t)$ to the k additional knots that have been added for the degree of the B-spline, using $k = 1$. For the degree 1 case, the values of the spline function are equivalent to the values of the coefficients, so we do not require any solution to the set of linear equations that describe the B-splines. Numerical evaluation of the integral identity of the coefficients is followed by computing the dot product of the coefficients and the degree $k + 1$ B-spline matrix to both integrate and interpolate $\beta_p(t)$.

Determination of Δp_1 in Eq. 3.4 remains. We do not fix the value of Δp_1 or make any strong assumption about the flow behavior prior to the first data point value as other authors have discussed. Instead, we generate Δp_1 directly by *a priori* distribution. The choice of distribution may vary; in practice, we may decide to fix it at a specific value, enforce a range of likelihood around a specific value, or express no likelihood for any (except for, of course, prohibiting negative values). For our general implementation, we use a HalfFlat distribution which expresses no preference for any value.

3.9.2 Construction of Knots

Various researchers have addressed the value of the initial inducing point, or knot, with a variety of thoughts on the matter. Çinar et al. (2006) and Onur et al. (2008) have provided a comprehensive discussion of the subject. To summarize the matter, the first knot should be equal to or less than the first data point. While the choice of the point at which the first not is placed may result in differences for other methods, we stress that it should make little difference to our approach. This is due to the fact that we remove any effect of the first knot by setting its value equal to the first value of $\beta_p(t)$, and after integration we remove all points preceding the first data point as a constant of integration. The value of the kernel function at the first data point, Δp_1, is represented by a random variable that is independent of the $\beta_p(t)$ function. With this formulation, we are not concerned with the choice of placement of the first knot; we consciously choose to exert prior belief, including no prior belief, for the value of the first knot by use of the prior distribution placed upon it.

The number of knots is another choice we must make. While some discussion of the *total* number of knots is given in recent literature, Ilk, Valkó, and Blasingame (2005, 2006) give a recommendation in terms of the number of knots *per log cycle* with a value of 2–6 in order to capture characteristic reservoir behavior. They also discuss a reduction in the number of knots for data that is low quality, such as data containing large errors, or noise. We have chosen to use a constant value of 5 knots per log cycle and address low quality data with our robust cost function as well as our regularization scheme based on curvature.

The set of logarithmically spaced knots are placed over the time interval of the observed data with additional logarithmically spaced knots beyond first and last values of the data

to account for the extrapolation of $\beta_p(t)$. We compute the number of log cycles that occur between the first and last data points:

$$z = \text{ceiling}\left(\log 10\left[\frac{x_n}{x_1} \right] \right) \quad \text{.....3.95}$$

We then generate the knots:

$$\tau_i = \tau_1 + \frac{i}{n-1}(\tau_n - \tau_1) \quad \text{for} \quad -k \le n \le k + 5z + 1 \quad \text{.....3.96}$$

Where k is the degree of the B-spline, and $5z$ indicates five knots per log cycle. Eq. 3.95 will automatically add the $k + 1$ knots required to evaluate the spline function, and over which $\beta_p(t)$ is extrapolated.

3.9.3 Initial Value of β_p(t) and Kernel Function

To encourage quick regression to the optimization scheme, it is helpful to provide starting values. Invalid starting values will result in singularities within the model, as the model requires the ability to compute the likelihood. Examples of invalid values would be a negative value supplied for a HalfNormal distribution, or something otherwise outside the imposed bounds such as an initial pressure that is less than observed pressures. "Correct" initial values are not strictly required; only values that fit within the defined probability density functions. Valid yet divergent initial values will lead to longer optimization times before convergence.

The initial value of $\beta_p(t)$ should be based upon prior knowledge of the reservoir geometry and the suggested flow regime as provided in **Table 3.2**. For example, if we are processing

a fractured well, we would expect the transient regime to display linear flow, so the chosen value of $\beta_p(t)$ would be 0.5. This must be transformed by the inverse of the softplus function, given by:

$$\text{softplus}^{-1}(x) = \ln\left[e^x - 1\right] \quad \text{.....} \quad 3.97$$

Better still, we may determine the initial value from the slope of a regression of a power-law function to the initial values of rate-normalized pressure (or pressure-normalized rate). We have found the first thirty values to yield good results.

The initial value of Δp_1 in Eq. 3.4 is assigned by the following equation based upon the observed data:

$$p_{1,\text{iv}} \leftarrow \frac{\Delta p_1}{q_1} \quad \text{.....} \quad 3.98$$

Where the subscript iv refers to initial value. Similarly, the initial value of q_1 should be the scaled reciprocal of the same equation. The scaling value is the difference between p_D and $1/q_D$ for transient linear and radial regimes, i.e. $\pi/2$:

$$q_{1,\text{iv}} \leftarrow \frac{\pi}{2}\frac{q_1}{\Delta p_1} \quad \text{.....} \quad 3.99$$

As for the prior probability distributions, we have several choices. We can choose a flat distribution which expresses that the value is a random variable but has no likelihood. A uniform distribution is slightly more informative, providing a bound for upper and lower values of the variable, but constant likelihood that is marginalized out of the model. If we do not have reliable evidence of a specific value, either of these would suffice. Otherwise,

a Gaussian distribution may be preferable. In the base case, we use a HalfFlat, but note that this choice can be changed arbitrarily from case-to-case:

$$p_1 \sim \mathcal{HF} \quad \text{.....3.100}$$

Likewise, we have for q_1:

$$q_1 \sim \mathcal{HF} \quad \text{.....3.101}$$

3.9.4 Construction of Flow Rate and Pressure Drop Functions

Recalling the convolution equation:

$$\Delta p(t) = \frac{1}{q_r} \sum_{i=1}^{n} \left(q_i - q_{i-1} \right) \Delta p_{cr} (t - t_{i-1}) \quad \text{.....Eq. 2.3}$$

We must next compute the derivative of the rate values. Again, we have a few choices. If we consider the rate values to be stepwise constant values, we can simply take the first difference of the observed data, where $q_0 = 0$. However, this does not give us the ability to "correct" any data quality issues. Prior authors have discussed this issue at length, and a summary of such issues may be found in Onur et al. (2008). Instead of stepwise values, the application of a PGM allows us to treat the rate data as an unknown function to be determined.

The choice of how to represent this presents some challenges. In contrast to the kernel functions we proposed to place a GP prior upon, we may not assume that the rate function is smooth nor monotonic. In fact, if we are correcting data of poor quality, then it is likely the rate function is quite noisy. We do know that for a depletion case, we must constrain

the rates to non-negative values, and there is a realistic upper bound (but not a general upper bound that applies to all cases. This suggests that we should represent the rates by a stepwise half flat distribution. That is, a distribution of non-negative values in the interval $[0,\infty)$ that expresses no likelihood for what each stepwise value may be. We perform the fit of these stepwise values by a second distribution. Recalling that Laplace and Gaussian distributions are equivalent to L1 and L2 regularization, respectively, we term the second distribution as a regularization on the half flat distribution of stepwise rates. Formally:

$$f_q(t) \sim \mathcal{HF}(t) \quad \text{.....} 3.102$$

$$f_q(t) \sim \mathcal{N}(q(t), \sigma_{q(t)}) \quad \text{.....} 3.103$$

The standard deviation of Eq. 3.103 states that distribution around the observed rate value we are willing to accept. In practice, we have found a value of 1 BPD to work well (or 1 psi in the case of pressure drop), but this may scale with the observed data. While it may appear odd to state that the unknown rate function is both distributed semi-randomly and not at the same time, one can think of Eq. 3.102 as a generating function, and Eq. 3.103 as a constraining function.

It may also be ideal to treat the *total* error in the model, rather than the error in rates and pressures independently. In this case, we would have an alternative to Eq. 3.103:

$$f_{t.e.,q}(t) \sim \mathcal{D}(\ln q(t), \sigma_{t.e.}) \quad \text{.....} 3.104$$

$$f_{t.e.,p}(t) \sim \mathcal{D}(\ln \Delta p(t), \sigma_{t.e.}) \quad \text{.....} 3.105$$

where the use of the log would evaluate relative error as opposed to absolute error, and $\mathcal{D}$ is some distribution to be determined.

We may also use an ensemble method to generate the rate function, such as combining sequences of observed data, sequences of rate-correction, and/or sequences of generated rates with no observation. We may not observe rates in all cases, meaning part of the rate history could be missing. The same can, of course, apply to the observed pressures. In these cases, we may impute any/all missing values, including both rates and pressure simultaneously.

3.9.5 Initial Reservoir Pressure

The details unknown function for initial pressure follows those of the initial value of Δp_1. Some rules around the initial pressure are that it must be greater than the maximum observed flowing pressure (or the maximum flowing pressure must be constrained below the initial reservoir pressure), and that it should be less than the maximum possible value given the formation depth and pressure gradient. For test cases of little-to-no noise we have not found any dependence on the inferred value of initial pressure with respect to the initial guess, so the initial value is set equal to the maximum observed flowing pressure unless data indicates otherwise.

3.9.6 Cost Function

Next, we perform the convolutions in Eq. 2.3. The kernel function of constant-rate pressure is convolved with either the data or the fit rate function. The result is compared with the observed pressure data with a Huber prior.

$$\ln \Delta p_{conv} \sim \mathcal{H}(\ln \Delta p, \sigma_{conv}, t = 0.1) \quad \text{.....3.106}$$

$$\sigma_{conv} \sim \mathcal{HN}(1) \quad \text{.....3.107}$$

The choice of Huber distribution improves the robustness of our method, but the value of σ_{conv} must still be calibrated for each data set. Here we also place a HalfNormal prior upon the σ_{conv} value of the Huber distribution, in which the most likely value is zero. The standard deviation equal to unity avoids numerical errors while initialization and tuning of the Markov chain occurs during the first few iterations. After completion of the simulation, the fit value of σ_{conv} may be as small as 10^{-3} or 10^{-4}. The HMC algorithm will infer the value(s) of σ_{conv} that fall within the posterior distribution of the entire model, thus removing our need to manually adjust the cost function to address noise or other types of large variances in the data. The optimization will determine the value automatically.

The variable-pressure drop case also employs a Huber prior upon residuals of the convolved constant-pressure rate kernel function and the observed pressure drop:

$$\ln q_{conv} \sim \mathcal{H}(\ln q, \sigma_{conv}) \quad \text{.....3.108}$$

The value of σ_{conv} is the same as in the variable-rate case.

3.9.7 Correlation of Kernel Functions

We enforce the relationship between the kernel functions by utilization of Duhamel's principle. We must convolve both kernel functions together and expect that the output be equal to our array of time against which the kernel functions are evaluated.

$$t_D = \int_0^{t_D} q_D(\tau)\, p_D(t_D - \tau)\, d\tau \quad \text{.....Eq. 2.1}$$

In this case, we *do* wish to place a high cost upon any outliers, so we place a Gaussian prior on the output of Eq. 2.1:

$$\ln t_{D,conv} \sim \mathcal{L}(\ln t_D, \sigma_{Duhamel}) \quad \text{.....3.109}$$

Because this relationship is exact, while the kernel functions are unknown and are to be determined, we keep the value of $\sigma_{Duhamel}$ small. We use a value corresponding to a standard deviation of 2% relative error:

$$\sigma_{Duhamel} = \ln 1.02 \quad \text{.....3.110}$$

While this value could be decreasing even further, there is a tradeoff in the time it takes the algorithm to run, and potential numerical issues when the error is too large. The output of Eq. 2.1 is the result of two unknown functions, so values of $\sigma_{Duhamel}$ much smaller than we have proposed may cause the HMC algorithm to struggle to converge (find the space of high likelihood in the posterior distribution).

3.9.8 Regularization for Smoothness of Kernel Functions

We have reviewed all of the parameters fundamental to the deconvolution PGM. What remains are additional constraints we wish to add, or must add, to achieve reliable results

from the deconvolution. The first of these is regularization of the kernel functions. Prior authors have found success in two means of ensuring the kernel functions are smooth. First, restricting the number of points, or knots, in the time interval of over which the function is evaluated. Second, by penalizing "non-smoothness" of the function. We employ both.

The number of knots per cycle influences the amount by which the kernel function can change per log cycle, and the amount of potential non-smoothness in the kernel function. Values of the kernel function are found by interpolating the gridded knots. We have evaluated both linear interpolation and B-spline based interpolation and found that while performance is comparable for developing the kernel functions, B-splines give superior results for the computation of derivatives but incur a computational penalty. This can be offset by reducing the number of knots used for B-splines, and by solving for the coefficients before optimization time, which reduces the interpolation using B-splines to a dot product during optimization time. Linear interpolation is implemented based upon a pre-computed slice of the time grid and can be applied during optimization as a single vector operation. We recommend 5 knots per cycle for the B-splines, and 40 or 50 knots per cycle for the linear interpolation. The high number of knots is a requirement to achieve smooth derivatives for use in regularization. Due to their superior performance for this purpose, we would recommend the B-splines over linear interpolation.

We employ regularization on curvature based upon the ideas of von Schroeter, Hollaender, and Gringarten (2002, 2004), but in a more general form. This is enabled by the fact that computing finite difference equations is a trivial matter for our model. We may do so in

either a vectorized form across arrays, or with the finite difference weights as a convolution kernel. In either case, we have implemented a general solution utilizing the radius of curvature given by Eq. 3.111. As we are evaluating the curvature of the log of the kernel function with respect to the log of time, the approximation used by von Schroeter is not valid. Their implementation has not accounted for the fact that the second derivative is *not* taken *of the log* of the kernel function. The second order derivative term in Eq. 3.111 will not return the correct result if evaluated on the log of the kernel function. They have also neglected the denominator in Eq. 3.111, with the result that the method is not invariant to step size, and the curvature in periods where the first derivative is not negligible will be incorrect. Full details of the derivation are presented in **Appendix B**.

$$K = \frac{\frac{t^2}{y}\frac{d^2y}{dt^2} + \frac{t}{y}\frac{dy}{dt} - \frac{t^2}{y^2}\left[\frac{dy}{dt}\right]^2}{\left[1+\left[\frac{t}{y}\frac{dy}{dt}\right]^2\right]^{3/2}} \quad \text{.....3.111}$$

Some important details regarding Eq. 3.111 are as follows: We use a central difference for all derivatives, giving in total we have $n - 2$ points for $d\tau$ over which we evaluate the curvature. Recall that we add additional $k + 1$ knots to our B-splines, so if we use degree 1 or greater B-splines, we suffer no reduction in resolution from the discretization.

The regularization placed upon K enforces smoothness of the kernel functions. A Gaussian prior is used as we desire strong penalties for any period of sharpness.

$$K \sim \mathcal{N}(0, \sigma_{curv})$$

We note that we place an independent and identically distributed distribution on every K_i term. This differs from the von Schroeter method, which placed a penalty on the sum of K_i terms, and an initial estimate of the proper amount of regularization to use comes from computing an average over the total number of rows of their solution matrix. We do not require any adjustment, as our formulation already accounts for differences in the length of the data arrays we may evaluate.

3.9.9 Priors Placed upon Flow Regime Behavior

Inconsistencies, noise, limited sample size, or other issues present in the rate and/or pressure data may not allow for the correct deconvolution of the data. In these cases, we may apply our algorithm iteratively, expressing a prior for specific behavior of the kernel function during specific intervals. Utilizing the relation of the beta-derivative to the well-testing derivative:

$$tp'(t) = \beta(t)\,p(t) \quad \text{.....} \quad 3.112$$

We may place a prior upon the derivative of the identity of the type given in **Table 3.2**. As examples, Eq. 3.113 corresponds to radial flow, and Eq. 3.114 corresponds to boundary-dominated flow. Eq. 3.113 may apply to either kernel function as both yield the same signature for radial flow, while Eq. 3.114 must apply only to the constant-pressure rate kernel function. Full details of the derivations are provided in **Appendix A**.

$$\frac{t}{\beta p}\frac{d\beta p}{dt} = 0 \quad \text{.....} \quad 3.113$$

$$\frac{t^2}{\beta_q}\frac{d}{dt}\left[\frac{\beta_q}{t}\right]=0 \quad \text{.....3.114}$$

One can imagine using these priors as an additional form of regularization. Analogous to expressing a belief of monotonicity or smoothness, we also express belief for additional behavior, such as periods where the beta-derivative must take specific values in addition. Priors allow us to overcome noise or information limits in the data.

3.9.10 Summary

Development of a workflow necessitates recognition of the flexibility of the PGM. The PGM separates knowledge of the system, and reasoning about the system. We never need to change the basic structure of the PGM. Instead, if we wish to change how the model behaves, we are only required to alter the distributions placed upon the variables or mask the data such that a variable is no longer treated as observed. Ideally, the PGM is a general solution for any deconvolution problem.

A workflow for application of the PGM evolves from understanding the desired reasoning, and the necessary variables in the PGM to perform that reasoning. For example, if we only wish to perform variable-rate deconvolution, then we do not need to apply Duhamel's principle to correlate the constant-pressure rate kernel function, nor do we even need the constant-pressure rate kernel function.

The general outline of our algorithm is as follows:

- Determine which "case" of deconvolution to perform:
  - variable-rate,

- variable-pressure,
- both simultaneously uncorrelated, or
- both simultaneously correlated.

- Set state for the solution to determine what variables are observed, and what variables must be generated. "Variable" in this context refers to a single time-step observation of rates or pressures, and initial reservoir pressure. None, all, or any combination of these may be missing and be imputed. (Of course, the quality and reasonableness of the result from the deconvolution is dependent on the data quality, as will be shown).
- Initialize the beta derivative(s) and its intercept(s). This is performed by computing the beta derivative of the pressure-normalized rate, or the rate-normalized pressure, over the first 30 time-steps of the data.
- Generate the dimensionless pressure and dimensionless rate functions, depending on the chosen deconvolution case. Transform these to the constant-rate pressure and constant-pressure rate functions respective to the reservoir properties. If these are not known, then a unit value may be used, and the dimensionless function and its dimensional form are equivalent.
- Impute initial reservoir pressure, if necessary, using a chosen generating function.
- Impute any missing rate or pressure values, if necessary, using a chosen generating function such as a series of Gaussian random walks over the contiguous missing time steps.
- Perform the convolutions respective to the chosen deconvolution case.

- Perform a convolution of the dimensionless rate and dimensionless pressure functions, if correlating the two in the chosen deconvolution case.
- Evaluate the likelihood of each time step value from the result of the deconvolution(s) using a prior distribution with mean equal to the respective observed value. To clarify, a prior distribution, such as a Gaussian, Laplace, or Huber, is placed upon each and every value along the array of rates or pressures. The mean of this distribution is the observed value, and the variance is a hyperparameter with an Exponential prior distribution that is inferred by the optimization.
- Evaluate the likelihood of the result of the convolution of the dimensionless functions, if necessary, for the chosen deconvolution case, using a Gaussian prior.

3.10 Validation and Application

We now validate our algorithm by comparison with the response from a known reservoir model. We obtain the dimensionless pressure and rate solutions by inversion from the Laplace domain using the GWR algorithm (Valkó and Abate 2002, 2004) in the Mathematica software (Wolfram Research 2018). For any reservoir model, we create a synthetic variable-rate and variable-pressure drop profile. These profiles are then convolved with the reservoir model solutions in the real domain. While we could perform the convolution in the Laplace domain, in practice the discontinuities we have encoded into our synthetic profiles present significant challenges for numerical inversion as piecewise functions are not defined for the inverse Laplace transform. If, however, smoothly changing profiles are utilized instead of step-wise profiles, then the convolution should be performed in the Laplace domain.

We perform several validations with each data set. First, we deconvolve the unaltered data for the variable-rate case. Then, we add random noise and repeat the deconvolution to test our algorithm for error tolerance. Next, we mask some intervals in the data and impute the missing rates and pressures. Following this, we assume the initial reservoir pressure is unknown, and impute the value in addition to the missing rates and pressures.

We repeat the sequence for the variable-pressure drop case. Then, we consider a multi-well case consisting of several wells with a variety of variable-pressure drop profiles and impute the pressure history of entire wells. Last, we demonstrate the simultaneous deconvolution approach on field data.

We run the HMC simulation for 1000 burn-in iterations in which the algorithm converges to the posterior distribution. We then discarded these and sample 50 iterations from the posterior. While this is a small number, we are not seeking to represent the full statistics of the posterior. Rather, we explore the posterior for an understanding of the uncertainty of the deconvolution and identify the iteration that contains a maximum likelihood estimate (MLE). We plot the MLE as a thicker opaque line whereas we plot other posterior samples as thinner, transparent lines. The goal is to provide a visualization of the uncertainty of the deconvolution with a visual anchor around the MLE.

The cases we run range from simple to complex, and the optimization times vary by two orders of magnitude from the simplest to the most complex. At the low-end of run time are the field cases, which may take between five to fifteen minutes. The field cases presented consist of only tens or hundreds of data points, and only ten or twenty parameters. At the high-end of run time are the imputation cases, which consist of ten thousand data

points, and several thousand parameters. These cases were run on the Texas A&M High Performance Research Computing group's Terra cluster, and generally ran for about twenty-four hours.

3.10.1 Generation of Synthetic Data

We have chosen as a base case a fractured vertical well in a circular reservoir (**Fig. 3.15**) due to the characteristic behavior of both linear and radial transient regimes. Given a constant rate inner boundary and a closed outer boundary Ozkan and Raghavan (1991a, 1991b) give the solution in the Laplace domain:

$$\overline{p}_D = \overline{p}_{D,inf} + \overline{p}_{D,nf} \quad \text{.....3.115.}$$

That is, the sum of the infinite acting solution and the "no-flow" outer boundary solution:

$$\overline{p}_{D,inf} = \frac{1}{2s\sqrt{s}}\left[\int_0^{\sqrt{s}(1-x_D)} K_0(z)dz + \int_0^{\sqrt{s}(1+x_D)} K_0(z)dz\right]$$

.....(infinite -acting) 3.116

$$\overline{p}_{D,nf} = \frac{1}{2s\sqrt{s}}\frac{K_0(\sqrt{s}r_{eD})}{I_1(\sqrt{s}r_{eD})}\left[\int_0^{\sqrt{s}(1-x_D)} I_0(z)dz + \int_0^{\sqrt{s}(1+x_D)} I_0(z)dz\right]$$

.....(no flow) 3.117

Eqs. 4.1 to 4.3 are for a uniform flux fracture. We obtain the infinite-conductivity case by using the value $x_D = 0.732$ (Gringarten, Ramey, and Raghavan 1974). A list of reservoir

properties is given in **Table 3.3.** The dimensionless pressure, dimensionless rate, and dimensionless pressure derivative functions for these properties are plotted in **Fig. 3.16**.

We require a synthetic rate and pressure profiles to demonstrate or deconvolution. We have chosen the following rate and pressure profiles:

$$q(t) = \begin{array}{llll} 40 & 0 & < t \leq & 100 \\ 0 & 100 & < t \leq & 163 \\ 70 & 163 & < t \leq & 260 \\ 0 & 260 & < t \leq & 353 \\ 50 & 353 & < t \leq & 510 \\ 30 & 510 & < t \leq & 620 \\ 0 & 620 & < t \leq & 733 \\ 60 & 733 & < t \leq & 856 \\ 0 & 856 & < t \leq & 903 \\ 40 & 903 & < t & \end{array} \quad \text{.....} 3.118$$

$$p_{wf}(t) = \begin{array}{llll} 3000 & 0 & < t \leq & 250 \\ 2000 & 250 & < t \leq & 500 \\ 1250 & 500 & < t \leq & 800 \\ 0 & 800 & < t & \end{array} \quad \text{.....} 3.119$$

We have chosen these profiles to highlight the utility of our deconvolution algorithm's ability to impute rate and/or pressure history. Data quality and consistency is a concern for any deconvolution method. The number of switches that occur in the synthetic histories provide several instances over which we can evaluate the performance of the imputation.

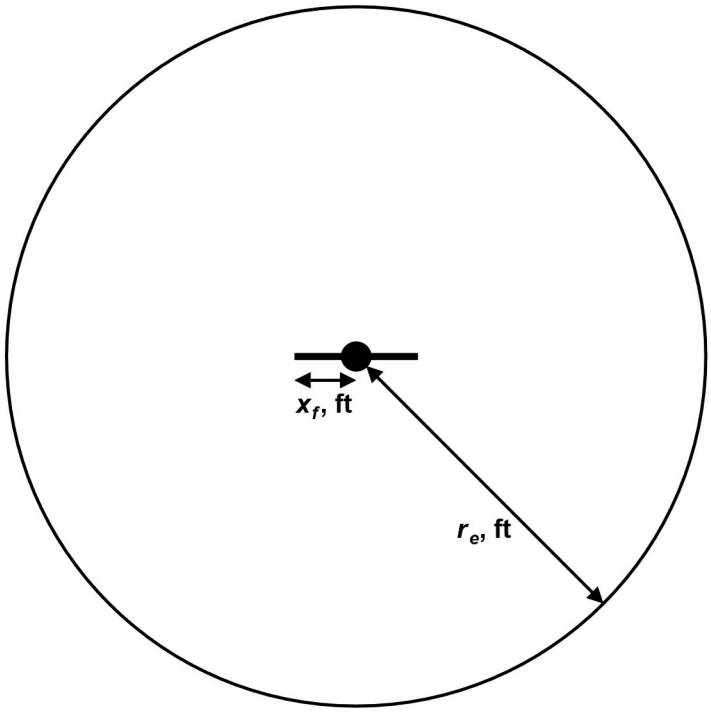

Figure 3.15 — Schematic of fractured vertical well in a circular reservoir.

Table 3.3 — Reservoir and fluid properties for Validation Cases

Reservoir Properties		
Reservoir Radius, x_e	=	283 ft
Wellbore Radius, r_w	=	0.354 ft
Fracture Half-length, x_f	=	50 ft
Reservoir Thickness, h	=	20 ft
Porosity, ϕ	=	0.1 (fraction)
Permeability, k	=	1.0 md
Total Compressibility, c_t	=	$1x10^{-5}$ psi^{-1}
Fluid Properties		
Fluid viscosity, μ	=	1 cp
Formation Volume Factor, B	=	1 RB/STB

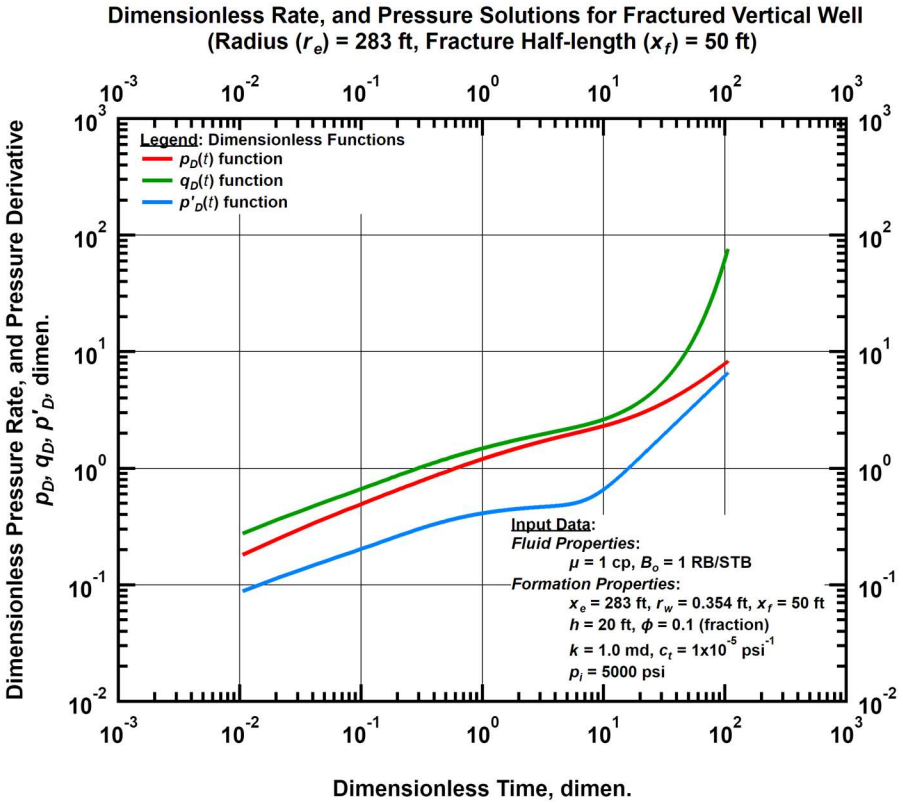

Figure 3.16 — Dimensionless pressure, dimensionless, and dimensionless pressure derivative solutions for fractured vertical well case.

3.10.2 Deconvolution of Variable-Rate Cases

Using the given synthetic rate profile, we generate several validation cases:

- Validation Case 1 is an exact result from the convolution of the rate profile with the dimensionless pressure function. **Fig. 3.17** shows the generated data for the deconvolution.
- Validation Case 2 adds errors to the rate and pressure histories in the form of random error. Each rate-time and pressure-time pair is randomly adjusted within $\pm$5% of its original value by multiplication with i.i.d. uniform distributions $\mathcal{U}(0.95, 1.05)$.
- Validation Case 3 increases the errors to within $\pm$20% of its original value by multiplication with i.i.d. uniform distributions $\mathcal{U}(0.8, 1.2)$.
- Validation Case 4 masks the rate history from 200 to 750 hours and from 800 to 1,000 hours. The "missing" rate values are then imputed during by the deconvolution algorithm.
- Validation Case 5 uses the input data from Case 2 with the 5% random noise, and masks the rate history from 200 to 750 hours and from 800 to 1,000 hours as in Case 3.
- Validation Case 6 removes the known initial reservoir pressure from Case 2 with the 5% random noise. The value of initial reservoir pressure is then imputed.

Each validation case first presents the input data, with any noise and/or masked periods as compared with the true values. The results of the deconvolution are shown in dimensional and non-dimensional form for 50 samples from the posterior distribution. We show the

constant-rate pressure function, the reconstructed-by-convolution variable-rate pressure function, the constant-pressure-drop rate function, and the reconstructed-by-convolution variable-pressure drop rate function. These are referred to as the "trained kernel" functions, as they are wholly dependent on the accuracy of the deconvolution to obtain the proper dimensionless function.

We also show the dimensionless rate, dimensionless pressure, dimensionless pressure derivative, and time functions. The dimensionless rate and dimensionless pressure functions are generated by use of Eq. 3.6 and Eq. 3.4, respectively. The dimensionless pressure derivative is computed by use of Eq. 3.112. The result of convolution of the dimensionless rate and dimensionless pressure functions, as defined by Duhamel's principle in Eq. 2.1, is shown as a comparison the time step. As this will always be a unit slope, we have scaled it on the y-axis for plotting convenience.

In Validation Case 1 (**Fig. 3.18**) we see that our algorithm performs an exact match of the $p_D(t)$ and $p'_D(t)$ functions. Although we have no way of determining very first value of $p'_D(t)$ as is typical for discrete computations of derivatives, extrapolation of the beta derivative beyond the range of data provides an assumption of a constant derivative at the edges, stabilizes the calculation (in plain terms, the expected value of the next step is the current value), and provides a good match in this case.

We note that the generation of the $q_D(t)$ function is inadequate at early time. This is a limitation of discrete computations with data. Duhamel's principle only holds if we have

continuous functions, or the entire history of the $p_D(t)$ and $q_D(t)$ functions with arbitrary time resolution.

Validation Case 2, with 5% random error on both the input flow rates and pressures, is presented in **Fig. 3.19** and **Fig. 3.20**. No change is made to the hyperparameters of the deconvolution algorithm for regularization – the variance parameter of the distribution placed upon the deconvolved function values is automatically determined by the algorithm as it converges to the posterior distribution. For the most part, the result is the same as in the exact case, with the primary difference being an increase in variance of the posterior distribution, and slight deviation at the beginning and end points of the $p_D(t)$ function. We can now see the distribution of the first value of the $p'_D(t)$ function as well. The reconstructed variable-rate pressure function values are quite smooth despite the noise in the input rate data. However, the reconstructed variable-pressure drop rate values now contain a great deal of noise, but "on average" appear correct.

Validation Case 3 in **Fig. 3.21** and **Fig. 3.22** increases the random error to 20%. Once again, no change to the hyperparameters is made. The reconstructed variable-rate pressure function is close to the true value even with the significant level of noise. The reconstructed variable-pressure drop rate values are now uninterpretable, but this is due to the noise in the input data, not the lack of a smooth constant-pressure rate function.

The input data are masked in Validation Case 4, presented in **Fig. 3.23** and **Fig. 3.24**, and most of the rate values are imputed. No noise is added to the input data. We observe a

greater amount of oscillation in the dimensionless rate function. However, we see that the significant features of the reservoir model are still captured by all functions.

Fig. 3.25 shows the distribution of imputed rates as compared with the true values, observed (non-masked) values, and the reconstructed rate values. There is good agreement between all four, with a greater variance in the imputed values than those constrained by observation.

Validation Case 5 adds the mask to the 5% random error of Validation Case 2. In **Fig. 3.26** and **Fig. 3.27**, once again we see a considerable amount of the feature on both the dimensionless rate and dimensionless pressure functions, despite the now significant level of noise in the reconstructed variable-rate pressure function. **Fig. 3.28** also shows an immense spread of variance on the imputed rate values. We emphasize the tolerance of the algorithm for this level of uncertainty regarding the true rate values.

Validation Case 6 presented in **Fig. 3.29** and **Fig. 3.30** impute the value of initial reservoir pressure given the 5% random error from Validation Case 2. While any prior can be used for the initial reservoir pressure, we have chosen a uniform non-informative prior of $p_i \sim \mathcal{U}(\max(p_{wf}), 6000)$, where the max observed flowing pressure is 4948.1 psia. **Fig. 3.31** shows the histogram of p_i in the posterior distribution. We infer an initial reservoir pressure in the range of 5002–5006 psi compared with the true value of 5000 psi.

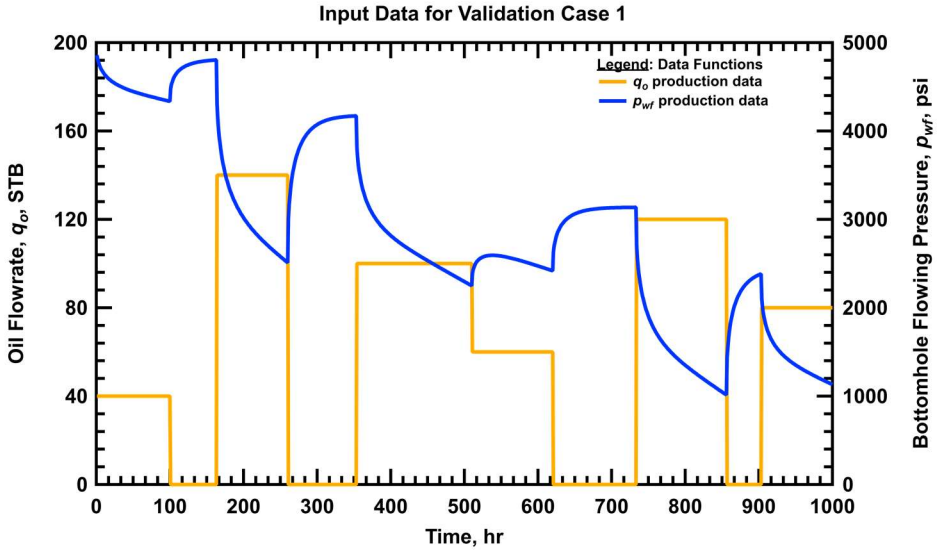

Figure 3.17 — Input data for Validation Case 1 (no error).

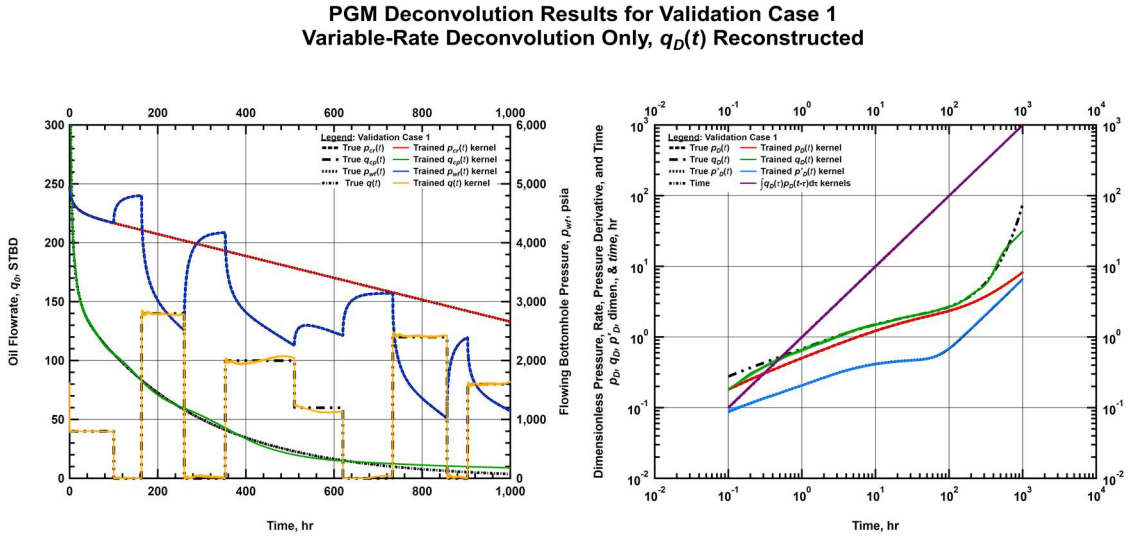

Figure 3.18 — Deconvolution results for Validation Case 1 (no error).

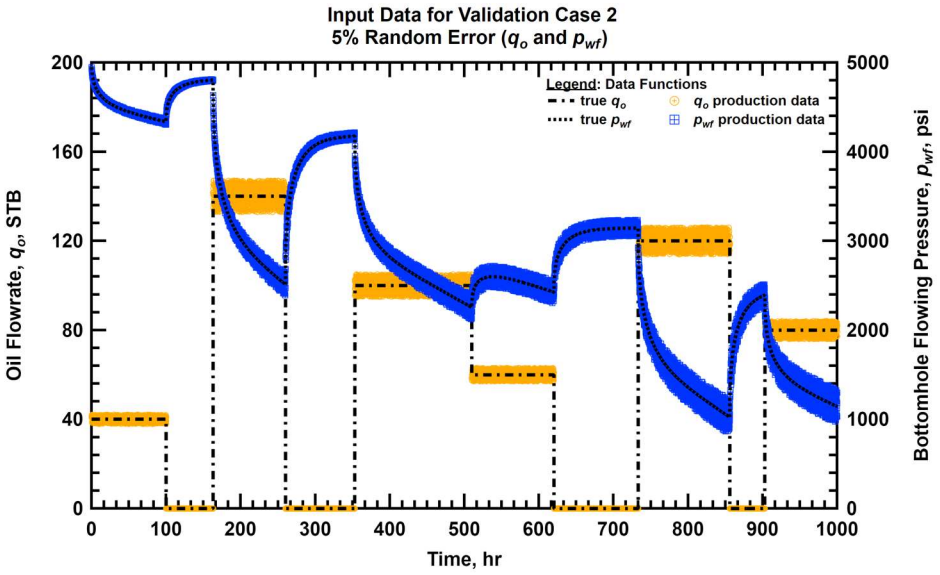

Figure 3.19 — Input data for Validation Case 2 (5% random error).

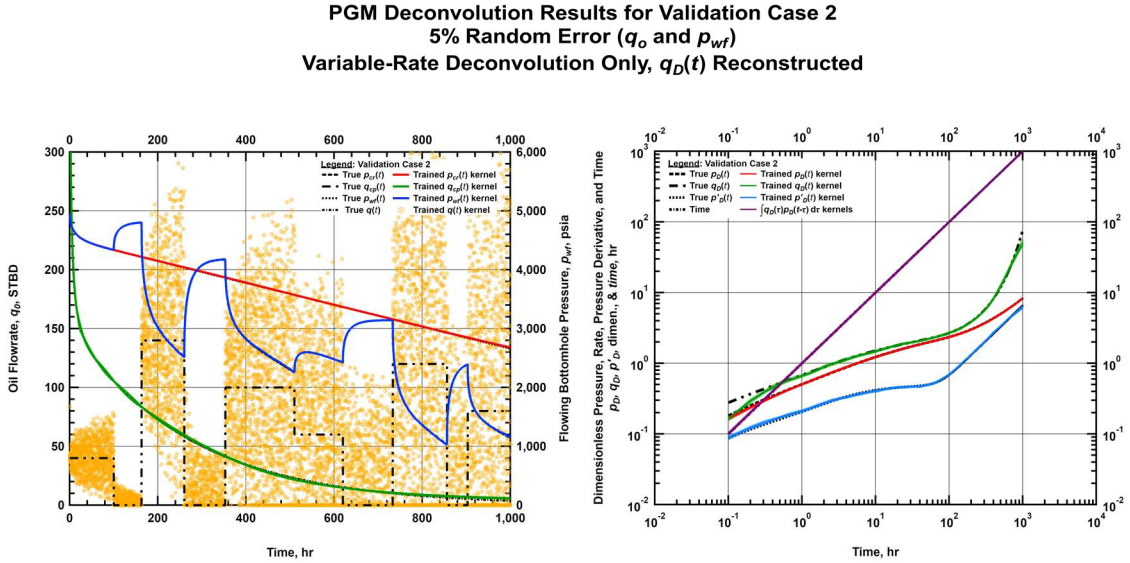

Figure 3.20 — Deconvolution results for Validation Case 2 (5% random error).

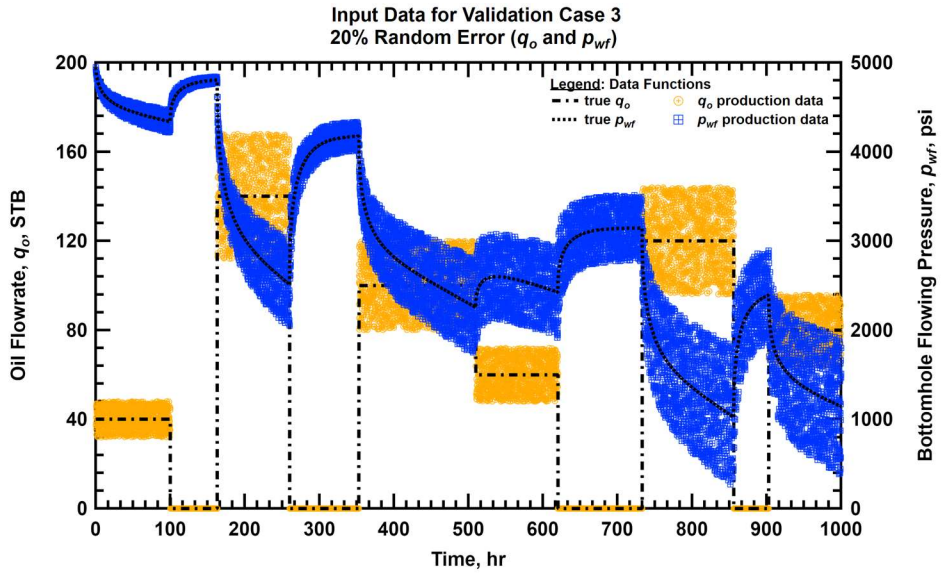

Figure 3.21 — Input data for Validation Case 3 (20% random error).

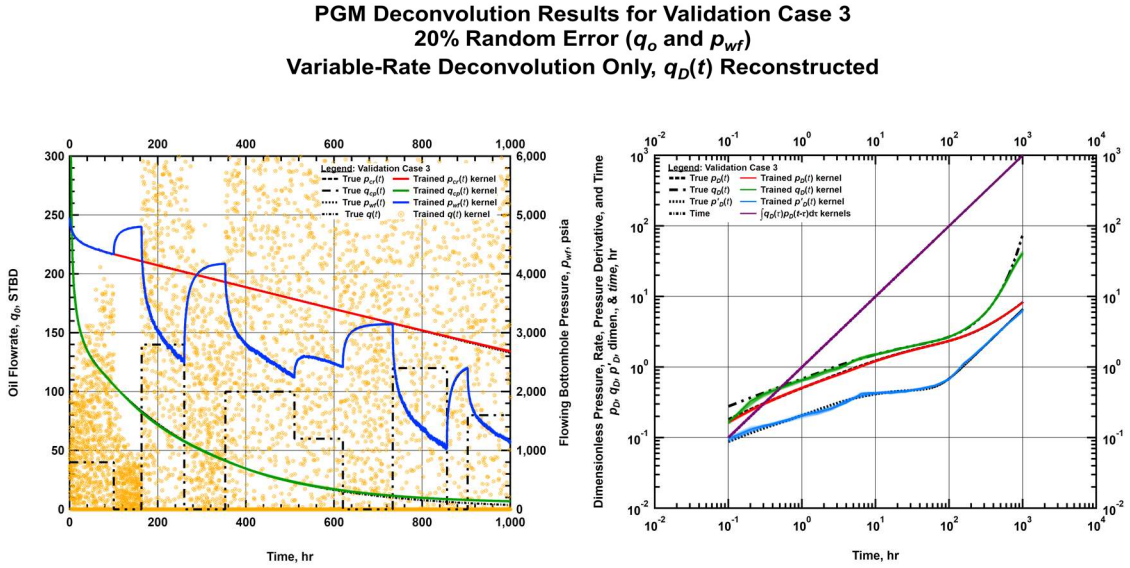

Figure 3.22 — Deconvolution results for Validation Case 3 (20% random error).

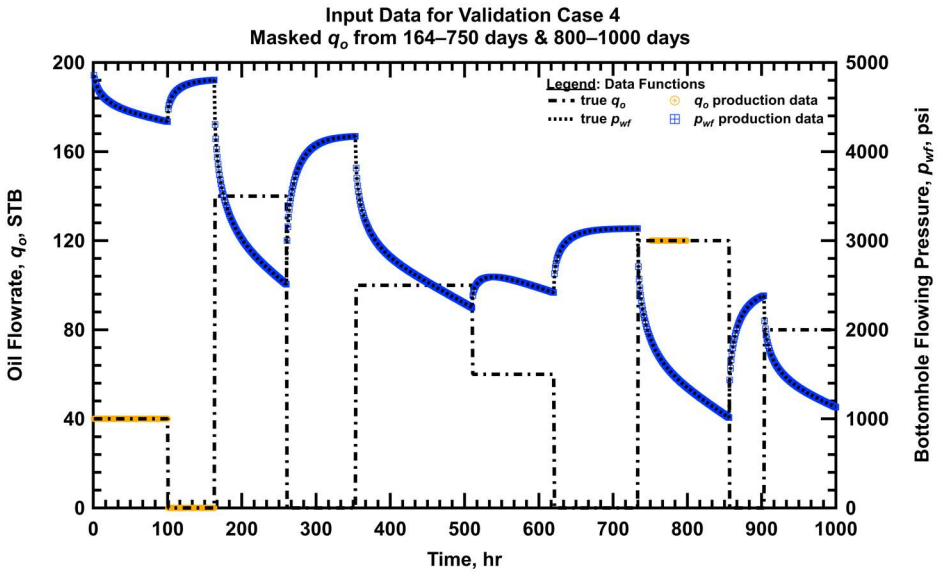

Figure 3.23 — Input data for Validation Case 4 (imputed missing rates, no error).

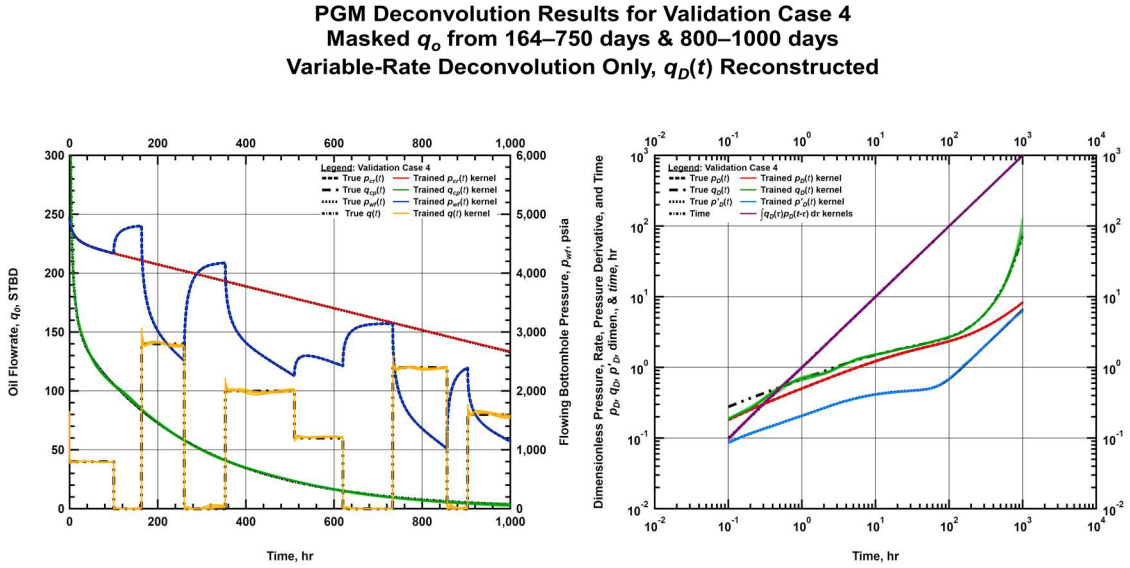

Figure 3.24 — Deconvolution results for Validation Case 4 (imputed missing rates, no error).

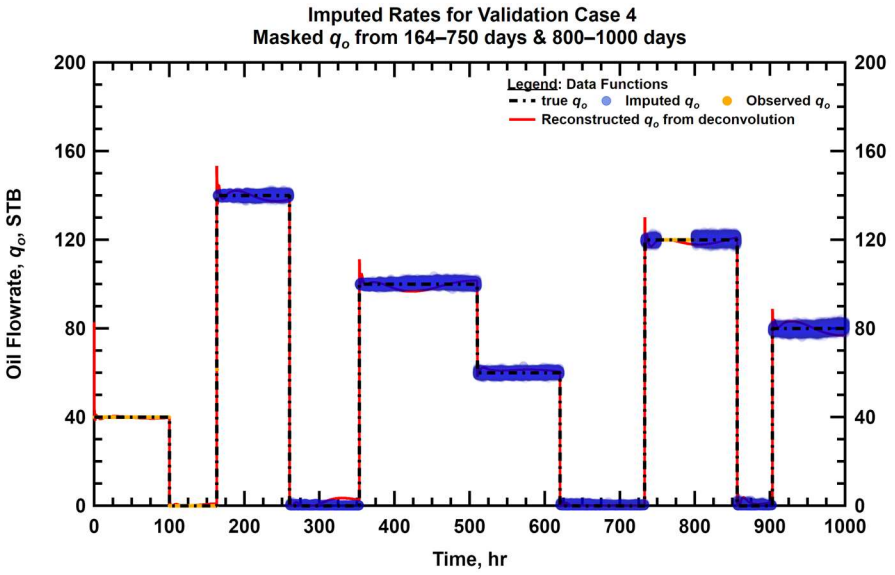

Figure 3.25 — Imputed Oil Rate for Validation Case 4 (imputed missing rates, no error).

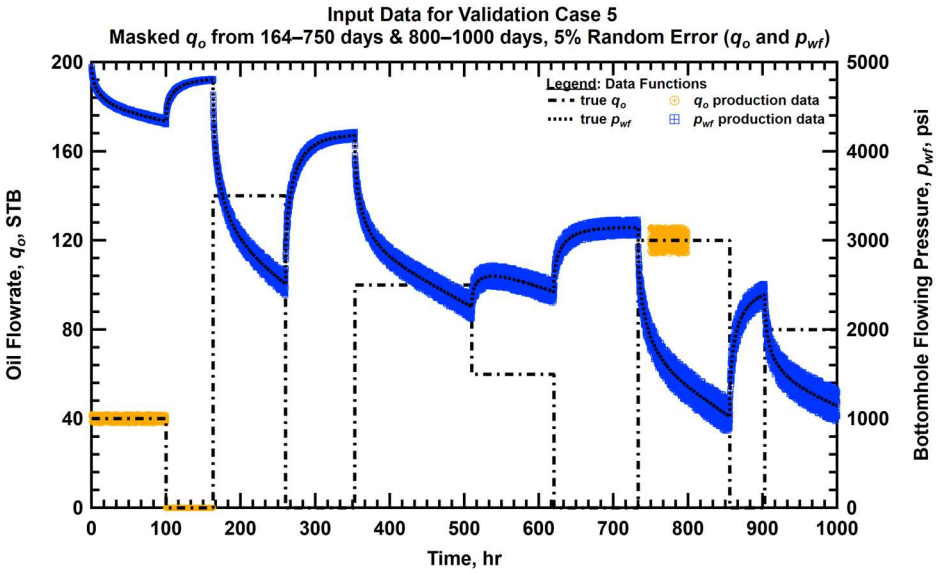

Figure 3.26 — Input data for Validation Case 5 (imputed missing rates 5%, random error).

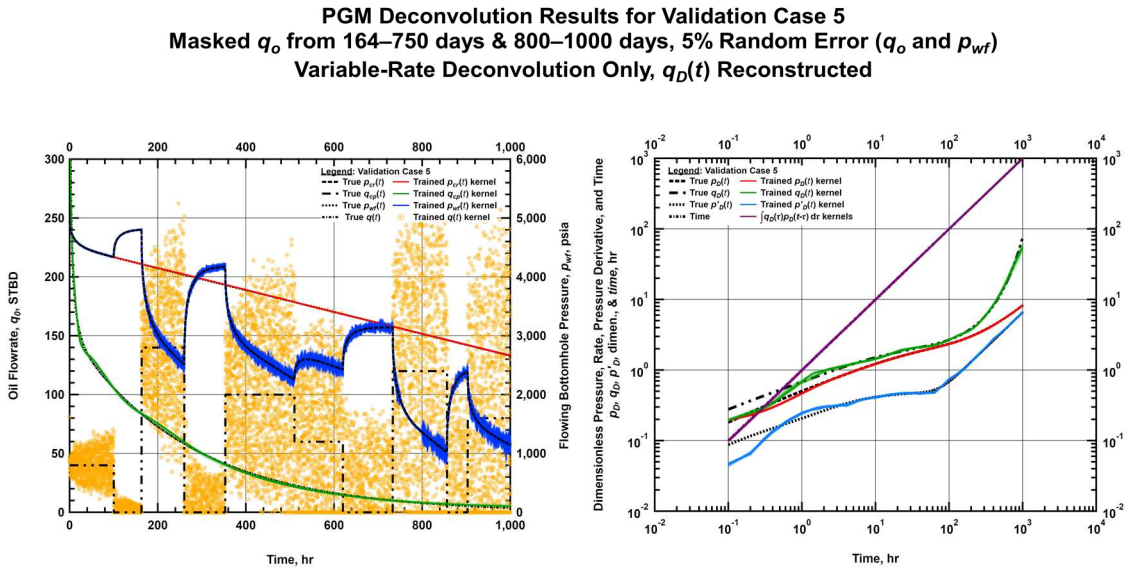

Figure 3.27 — Deconvolution results for Validation Case 5 (imputed missing rates, 5% random error).

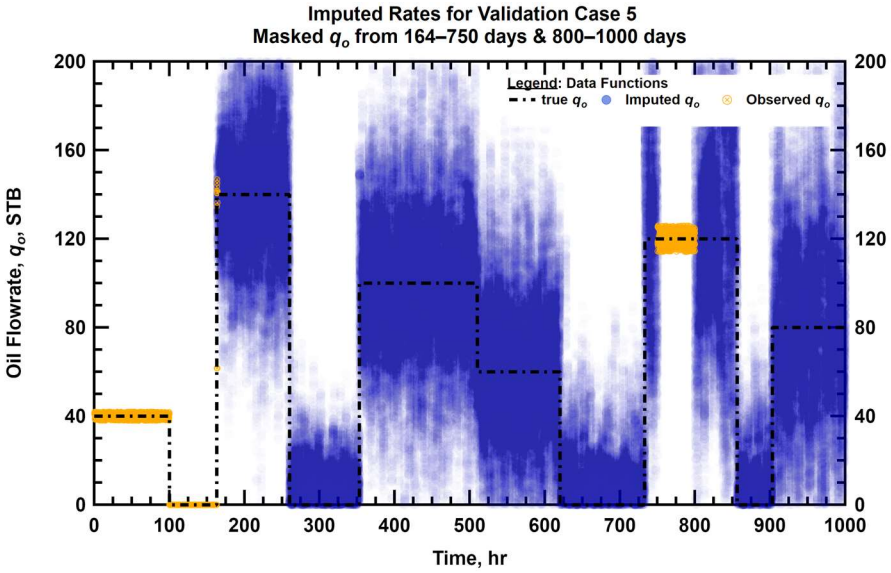

Figure 3.28 — Imputed Oil Rate for Validation Case 5 (imputed missing rates, 5% random error).

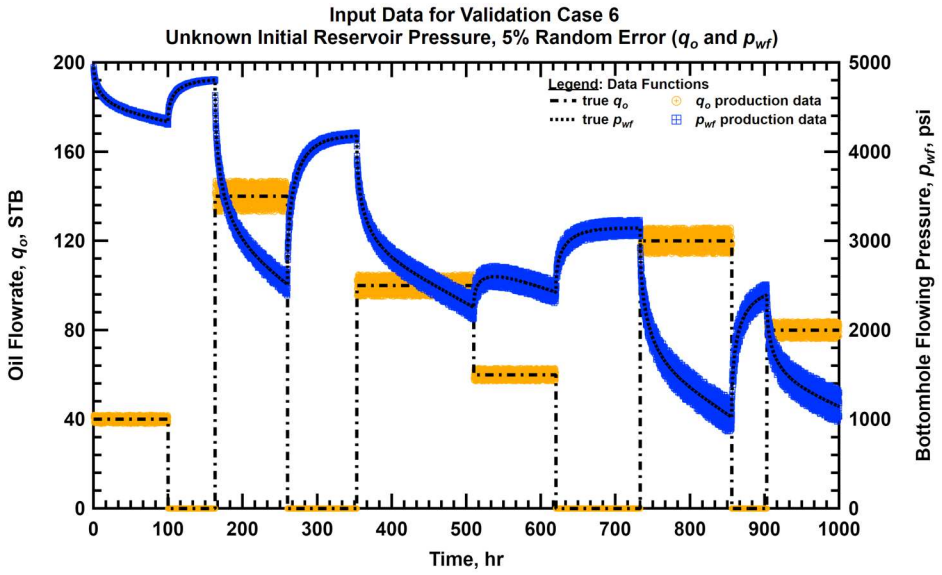

Figure 3.29 — Input data for Validation Case 6 (imputed initial reservoir pressure).

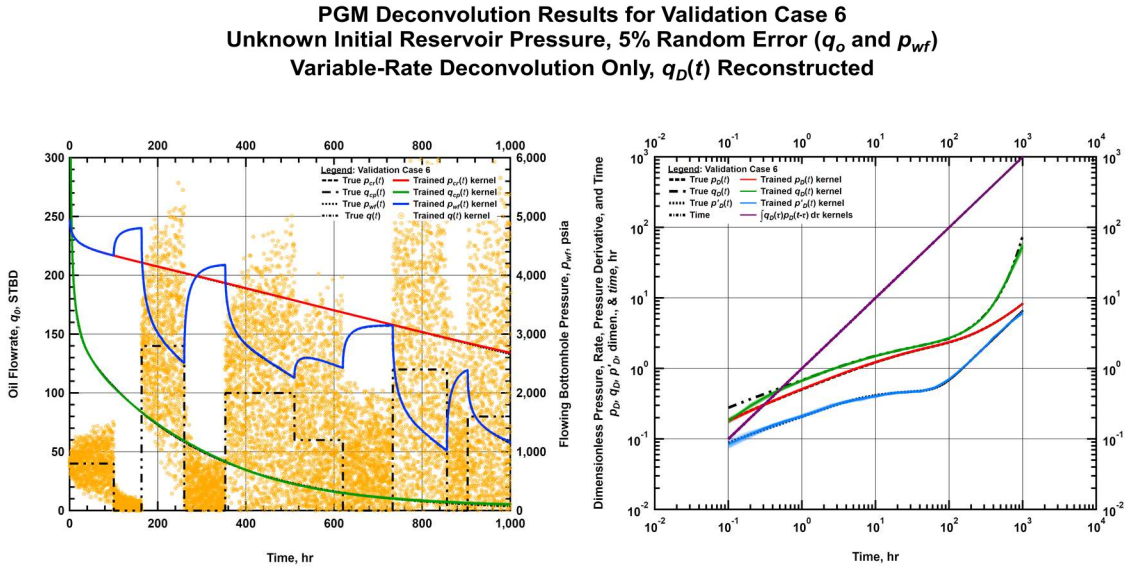

Figure 3.30 — Deconvolution results for Validation Case 6 (imputed initial reservoir pressure).

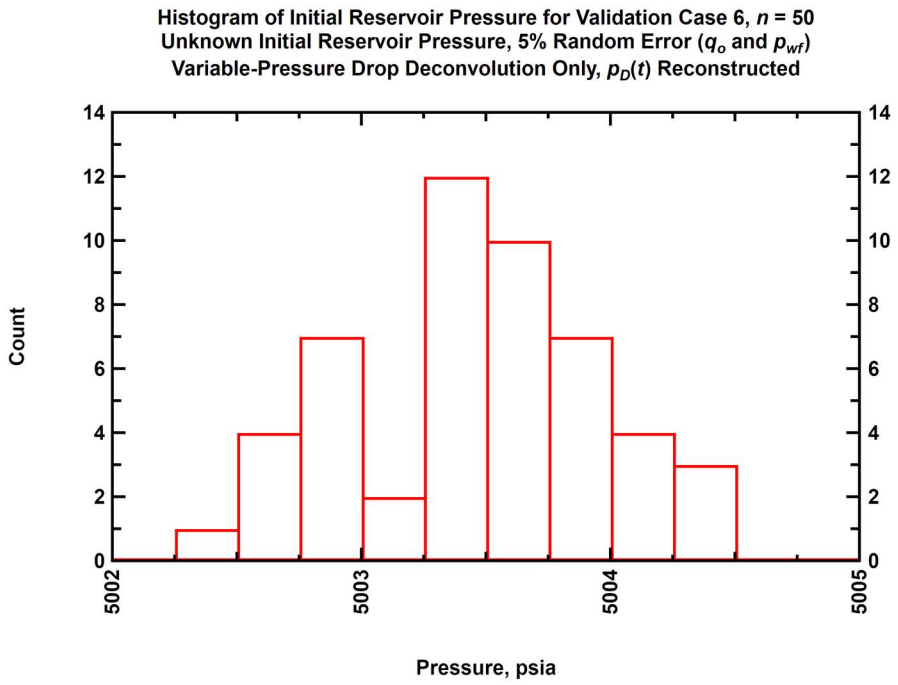

Figure 3.31 — Histogram of Imputed Initial Reservoir Pressure for Validation Case 6 (imputed initial reservoir pressure).

3.10.3 Deconvolution of Variable-Pressure Drop Cases

Next, we repeat similar validation cases with a different profile for the variable-pressure drop case:

- Validation Case 7, like Validation Case 1, is an exact result from the convolution of the rate profile with the dimensionless pressure function.
- Validation Case 8 mirrors Validation Case 2 and adds errors to the rate and pressure histories in the form of random error. Each rate-time and pressure-time pair is randomly adjusted within $\pm 5\%$ of its original value by multiplication with i.i.d. uniform distributions $\mathcal{U}(0.95, 1.05)$.

- Validation Case 9 increases the errors to within $\pm$20% of its original value by multiplication with i.i.d. uniform distributions $\mathcal{U}(0.8, 1.2)$.
- Validation Case 10 masks the input pressure history from 300 to 600 hours. The "missing" pressure values are then imputed during by the deconvolution algorithm.
- Validation Case 11 uses the input data from Case 2 with the 5% random noise and masks the rate history from 300 to 600 hours as in Case 10.

In Validation Case 7 (**Fig. 3.32** and **Fig. 3.33**) we see that our algorithm performs an exact match of the dimensionless rate function, performs well with the dimensionless pressure function, but mismatches most of the transient period of the dimensionless pressure derivative. As in the variable-rate case where we saw the dimensionless rate function begin overlying the dimensionless pressure function, the reverse is true here. Smaller timesteps would alleviate the issue.

Aside from the early-time mismatch, the reconstructed rate and pressure functions are nearly an exact match. The algorithm struggles to fit variable-rate pressure values of zero at late-time but reconstructs the function correctly elsewhere.

Validation Case 8 shown in **Fig. 3.34** and **Fig. 3.35** demonstrates the deconvolution on 5% random error. The smaller number of changes in the input data, as compared to Validation Case 2, result in greater uncertainty of the deconvolved result, and the results are not nearly as good. However, most of the rate and pressure function characteristics are captured except for the early-time behavior. The deconvolved response has not captured the high initial rate values as in the exact case.

For Validation Case 9, this is not the case, and the significant level of noise, especially the truncated pressure values at late time, lead to a poor result on all accounts (**Fig. 3.36** and **Fig. 3.37**), and the kernel functions cannot be recovered. The stability of the behavior of the model in failure is appealing. We do not observe oscillation or divergence. Instead, we see model failing to converge from the prior to the posterior, and the prior is returned.

Validation Case 10 follows Validation Case 4 and masks a portion of the input pressure data from 300 to 600 hours. **Fig. 3.38** and **Fig. 3.39** present the results. The significant reservoir features are captured by all functions, and exceptionally well by the dimensionless rate function after the first 20 or 30 data points. **Fig. 3.40** shows the imputed pressure values. The difficulty in imputing this profile is that the derivative of the profile is a constant of zero except for a single non-zero value at 500 hours. Therefore, any set of constant values is a valid solution for all but three values (the first at 300 hours, the point at 500 hours, and the last at 600 hours), but this information regarding the relationship of the masked values is unavailable. The minimal error in the imputed values, with only significant artifact seen around 600 hours at the end of the masked interval, is impressive given the non-uniqueness of this problem.

Validation Case 11, as illustrated by **Fig. 3.41** and **Fig. 3.42**, masks the 5% random noise from Validation Case 8 from 300 to 600 hours. The resulting dimensionless rate function appears similar to the unmasked case – that is, it fails to capture the early-time behavior. **Fig. 3.43** shows a greater degree of error in the imputed values, but less than in the observed data.

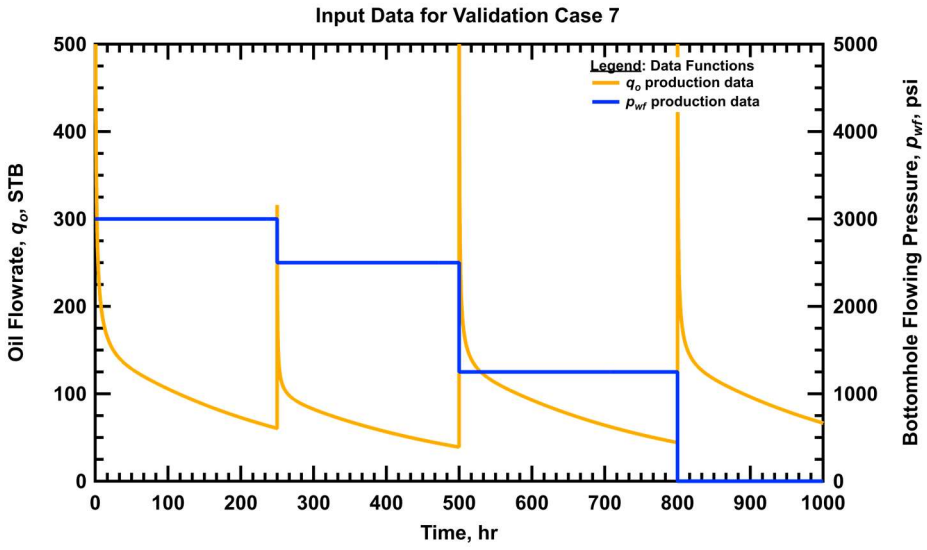

Figure 3.32 — Input data for Validation Case 7 (no error).

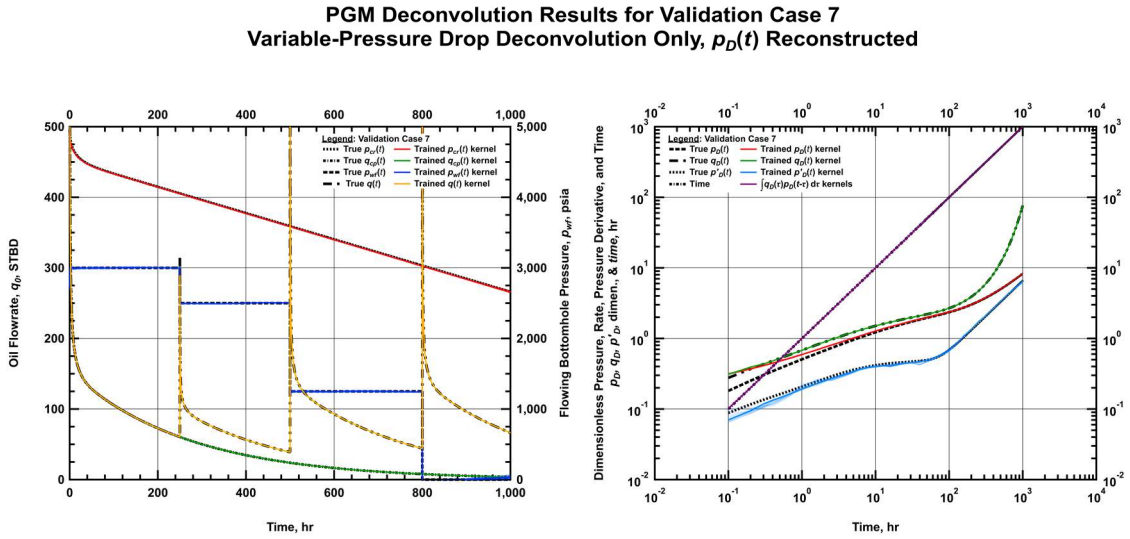

Figure 3.33 — Deconvolution results for Validation Case 7 (no error).

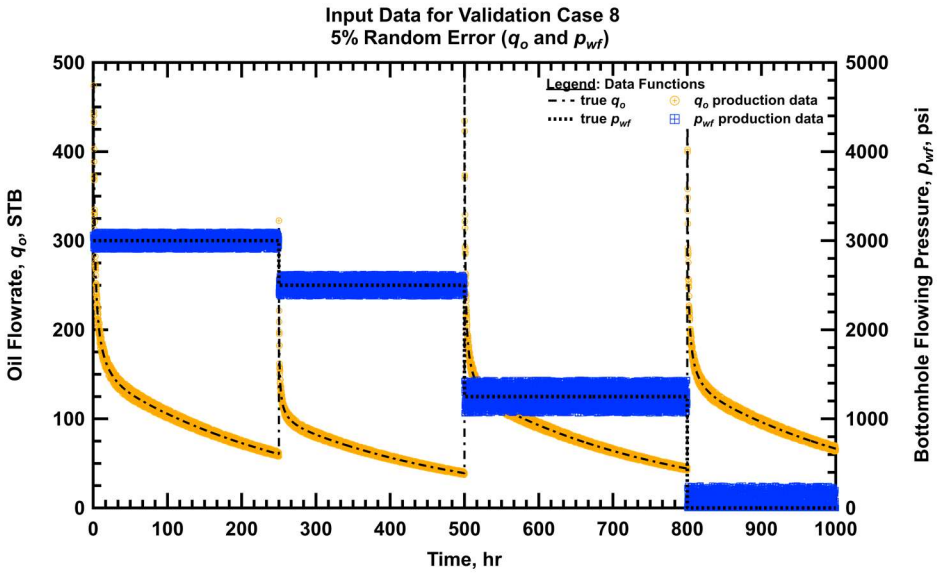

Figure 3.34 — Input data for Validation Case 8 (5% random error).

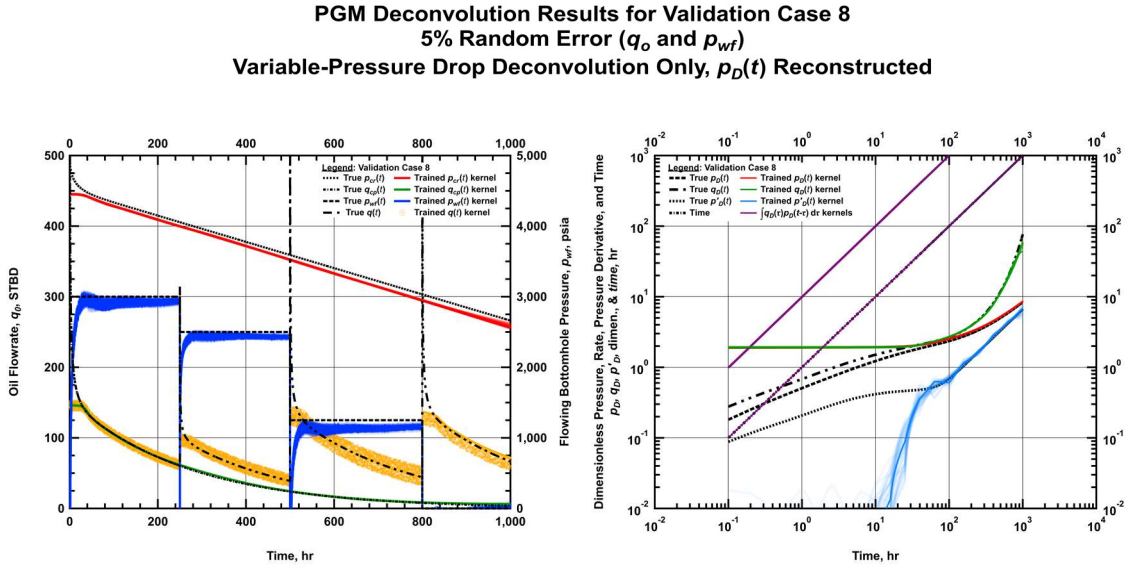

Figure 3.35 — Deconvolution results for Validation Case 8 (5% random error).

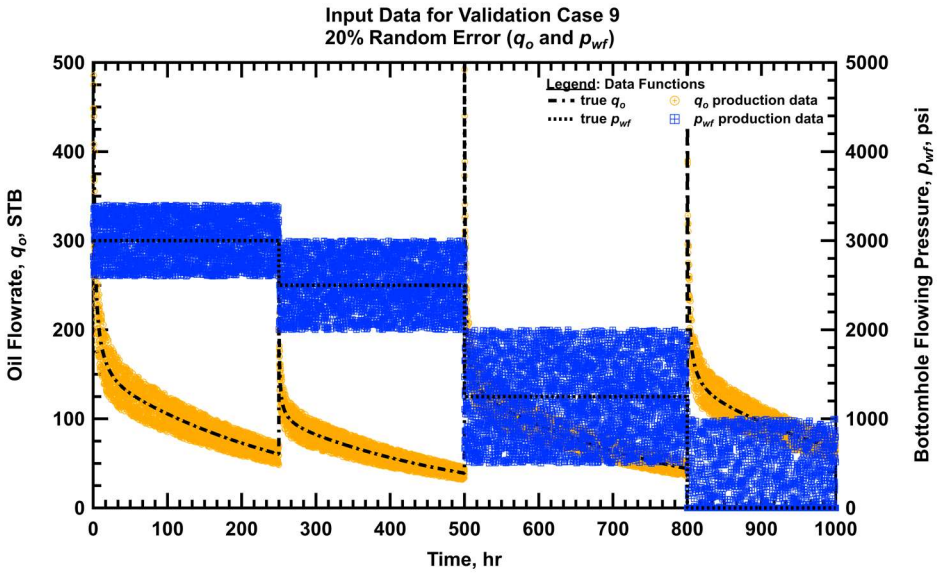

Figure 3.36 — Input data for Validation Case 9 (20% random error).

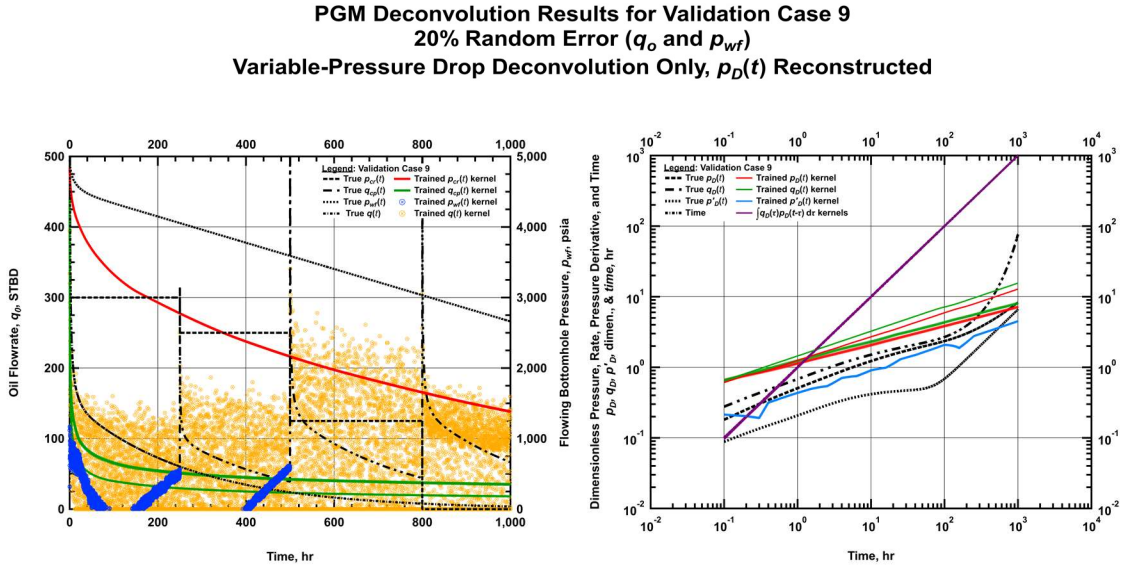

Figure 3.37 — Deconvolution results for Validation Case 9 (5% random error).

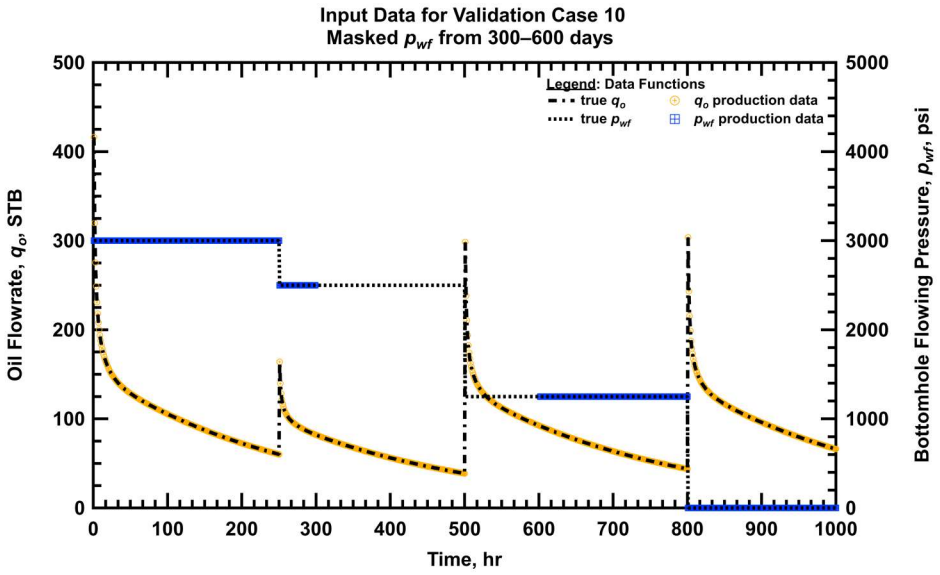

Figure 3.38 — Input data for Validation Case 10 (imputed missing pressures, no error).

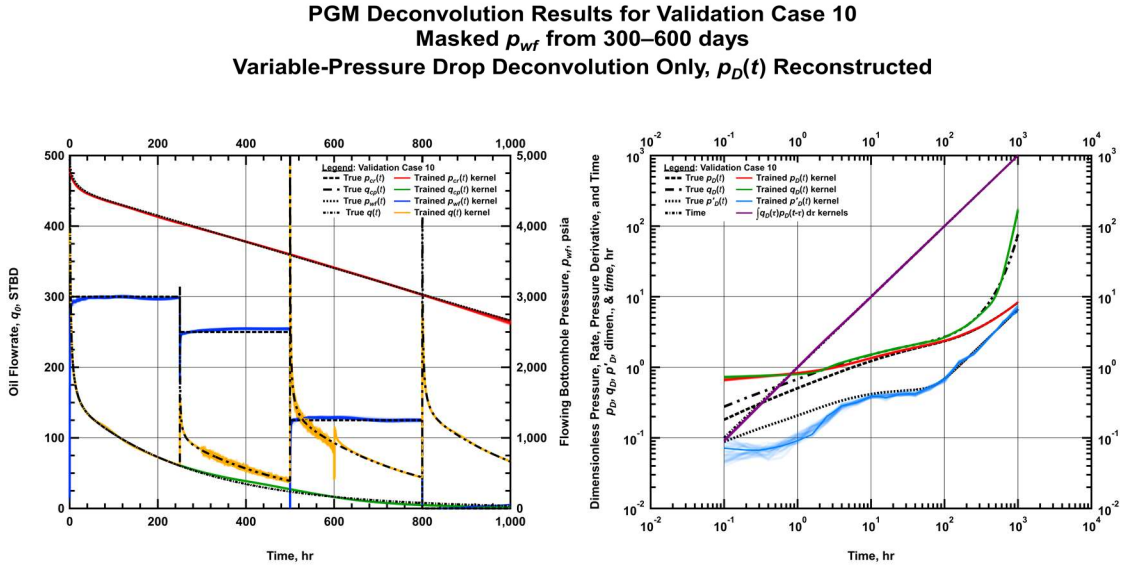

Figure 3.39 — Deconvolution results for Validation Case 10 (imputed missing pressures, no error).

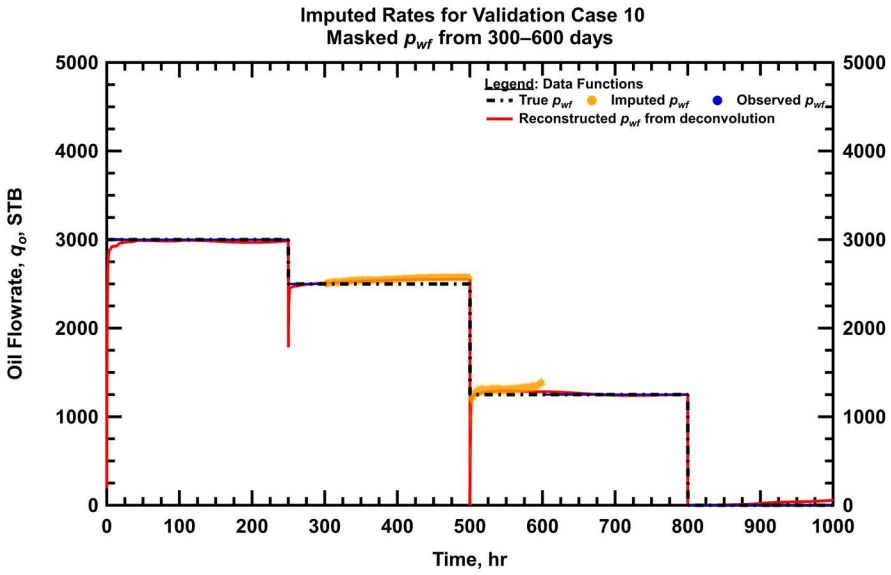

Figure 3.40 — Imputed bottomhole flowing pressure for Validation Case 10 (imputed missing pressures, no error).

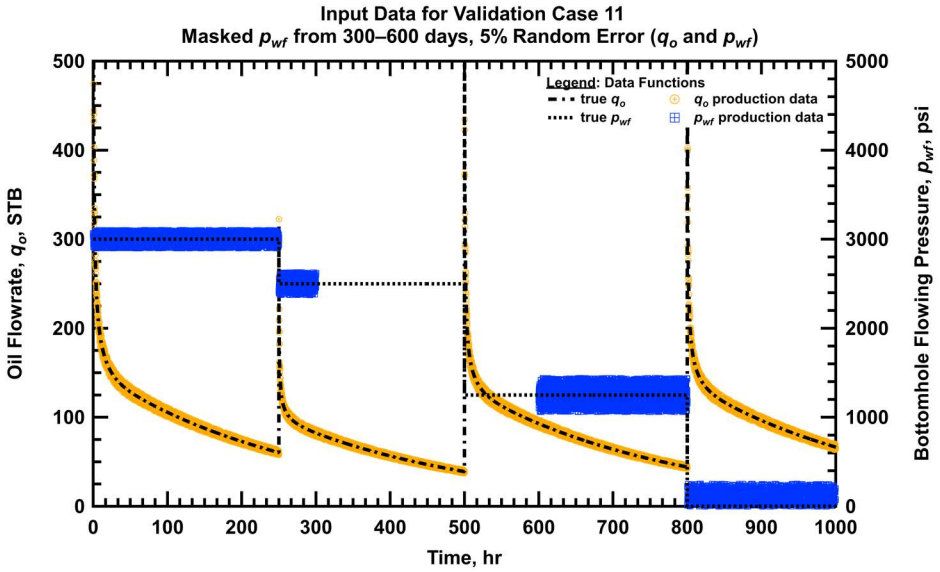

Figure 3.41 — Input data for Validation Case 11 (imputed missing pressures, 5% error).

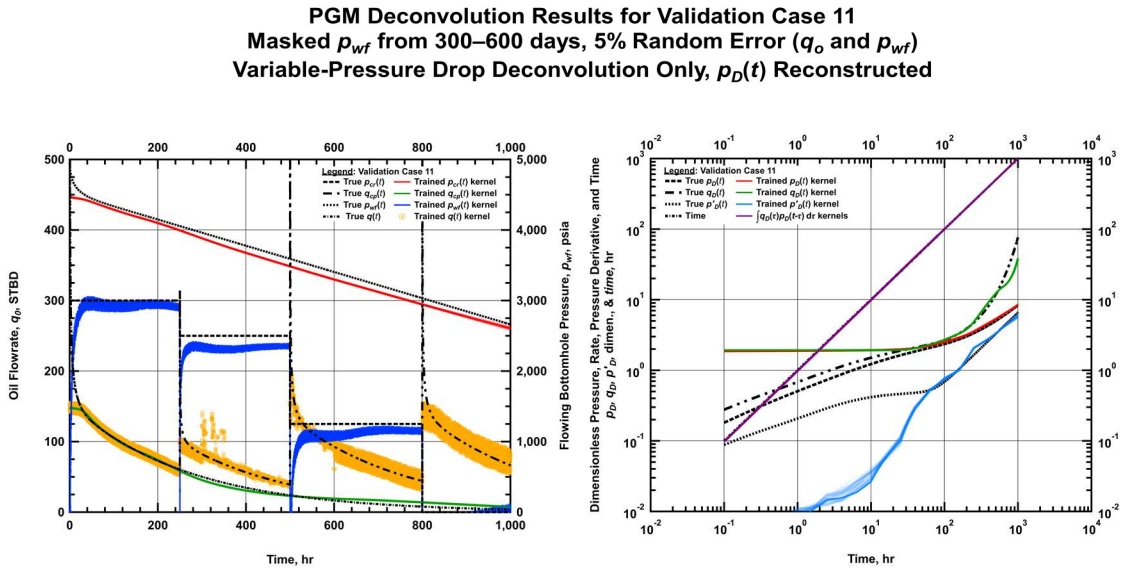

Figure 3.42 — Deconvolution results for Validation Case 11 (imputed missing pressures, 5% error).

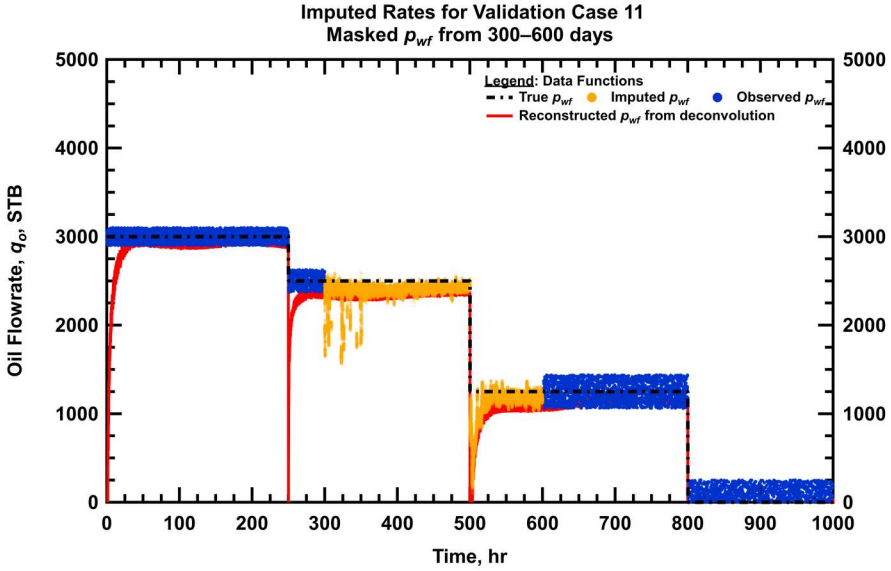

Figure 3.43 — Imputed bottomhole flowing pressure for Validation Case 11 (imputed missing pressures, 5% error).

3.10.4 Deconvolution of Variable-Rate Cases with Random Rate Profile

We next test our algorithm with a completely random rate profile and severe levels of noise. This test demonstrates the numerical stability of our method:

- Validation Case 12 is an exact result from the convolution of the random rate profile with the dimensionless pressure function.
- Validation Case 13 adds random error to Validation Case 12. Each rate-time and pressure-time pair is randomly adjusted within $\pm$5% of its original value by multiplication with i.i.d. uniform distributions $\mathcal{U}(0.08, 1.05)$.
- Validation Case 14 increases the errors to within $\pm$20% of its original value by multiplication with i.i.d. uniform distributions $\mathcal{U}(0.8, 1.2)$.
- Validation Case 15 increases the errors to within $\pm$40% of its original value by multiplication with i.i.d. uniform distributions $\mathcal{U}(0.6, 1.4)$.
- Validation Case 6 increases the errors to within $\pm$60% of its original value by multiplication with i.i.d. uniform distributions $\mathcal{U}(0.4, 1.6)$.

In Validation Case 12 (**Fig. 3.44** and **Fig. 3.45**) we see that our algorithm performs almost identically to Validation Case 1. The random rate profile presents no issues for the deconvolution. We observe an exact match of the dimensionless pressure function, and reconstruction of the dimensionless rate function performs well except for the tail of the array. The mismatch of the tail seems to occur only for the exact cases with no noise.

Validation Case 13 shown in **Fig. 3.46** and **Fig. 3.47** demonstrates the deconvolution on 5% random error. There is a minor degradation of the recovered pressure derivative, but the dimensionless rate function is improved as compared with Validation Case 12.

For Validation Case 14, we begin to see oscillation in the pressure derivative and dimensionless rate profile in **Fig. 3.48** and **Fig. 3.49** due to the 20% random noise.

Validation Case 15 in **Fig. 3.50** and **Fig. 3.51** adds 40% random noise and begins to present a challenge to recover the first few data points of the dimensionless pressure profile. While the general trend of the function is recovered, the oscillation in the pressure derivative begins to create challenges for interpretation.

Validation Case 16 adds severe error of 60% random noise as illustrated by **Fig. 3.52** and **Fig. 3.53**. While the level of noise of significant, the true functions are still identifiable in the deconvolved response. The early-time behavior is not recovered, but on the other hand, the results are not far from the true functions.

Overall, we see in these cases that the deconvolved response is stable despite unrealistically large noise coupled with a completely random rate profile. There is no divergence of the recovered functions to infinity, and any oscillation is well controlled and centered around the true function.

Interestingly, we see that, despite yielding a full posterior distribution, we do not observe the range of uncertainty significantly increase such that it would cover the true function. Any increase is rather small. We interpret this to indicate that despite the high levels of noise, use of 10,000 data points should give enough confidence to converge upon the true

answer. However, the true answer is not contained by the posterior distribution. If we assume that our method is yielding the true posterior given the observed data, we can conclude that the true answers are recoverable from data that contains such high error.

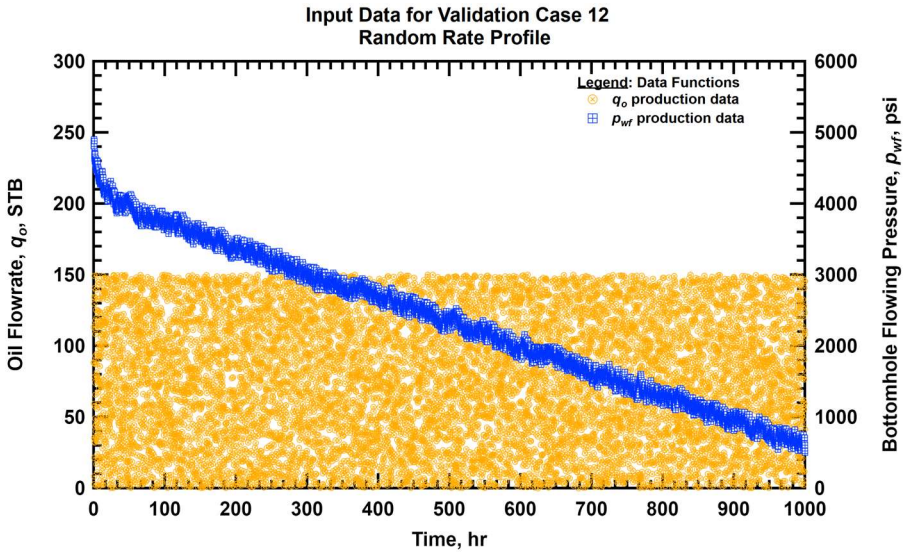

Figure 3.44 — Input data for Validation Case 12 (random rate, no error).

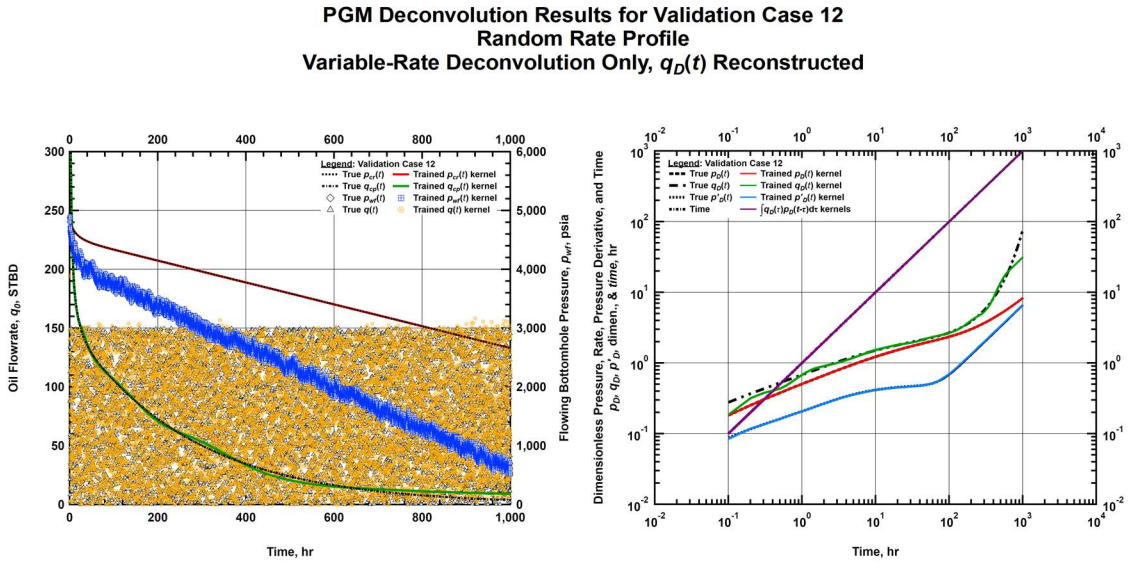

Figure 3.45 — Deconvolution results for Validation Case 12 (random rate, no error).

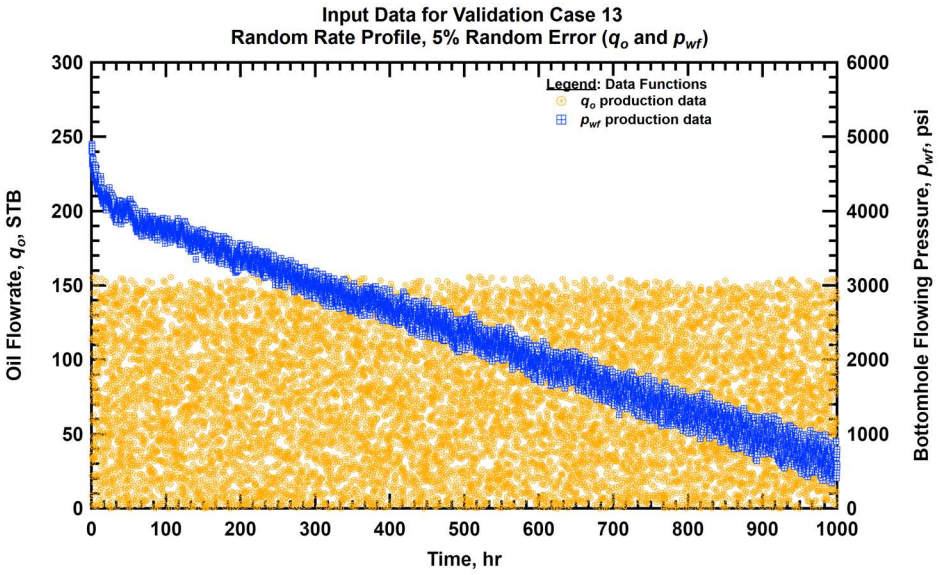

Figure 3.46 — Input data for Validation Case 13 (random rate, 5% random error).

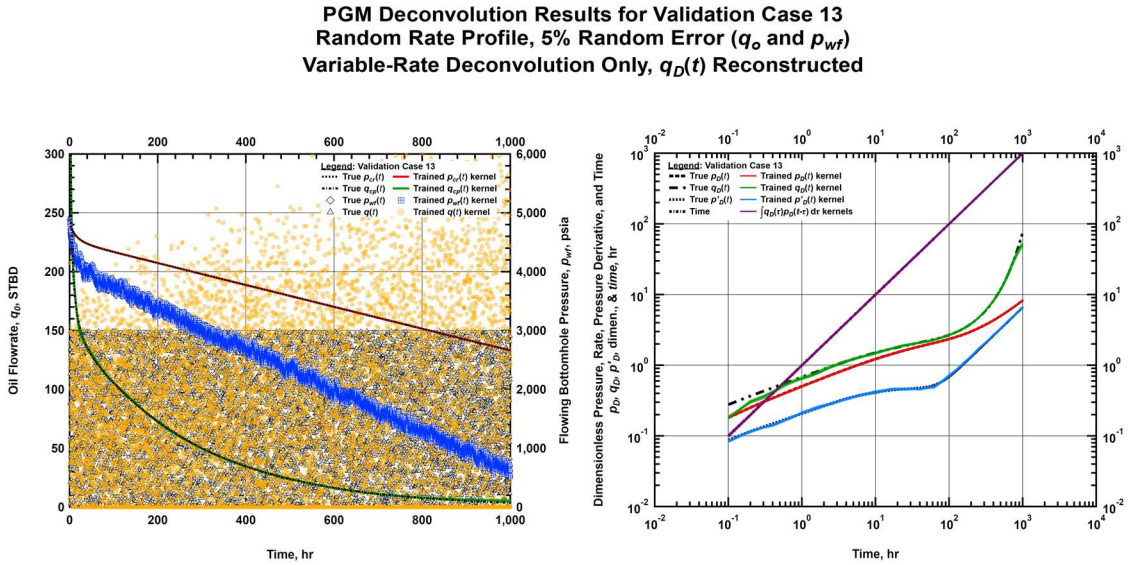

Figure 3.47 — Deconvolution results for Validation Case 13 (random rate, 5% random error).

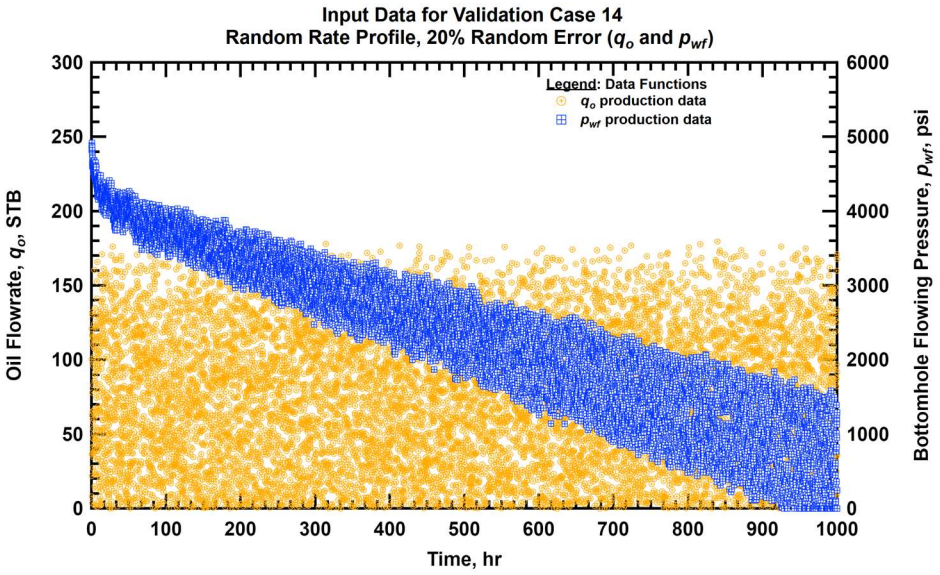

Figure 3.48 — Input data for Validation Case 14 (random rate, 20% random error).

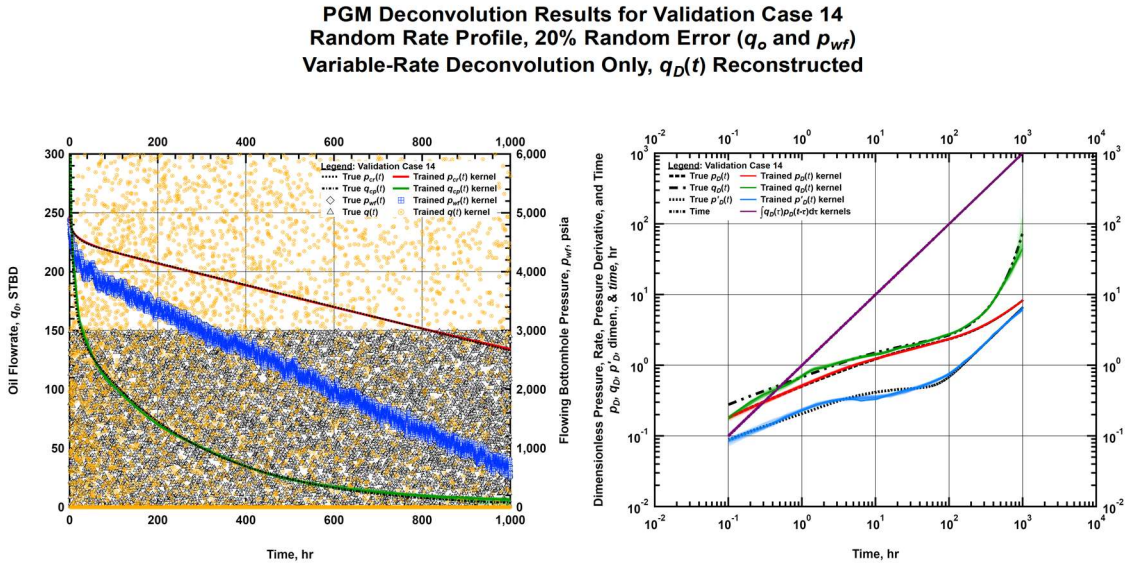

Figure 3.49 — Deconvolution results for Validation Case 14 (random rate, 20% random error).

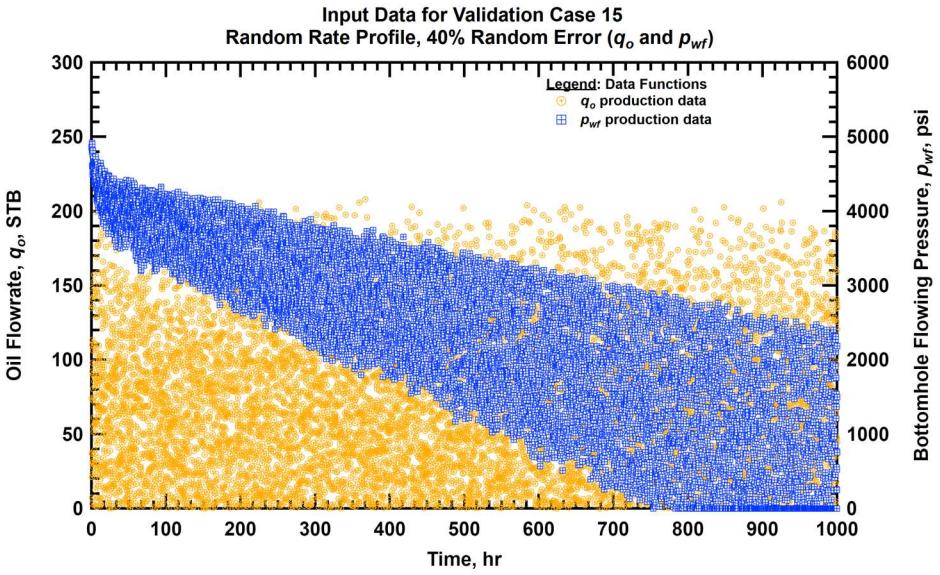

Figure 3.50 — Input data for Validation Case 15 (random rate, 40% random error).

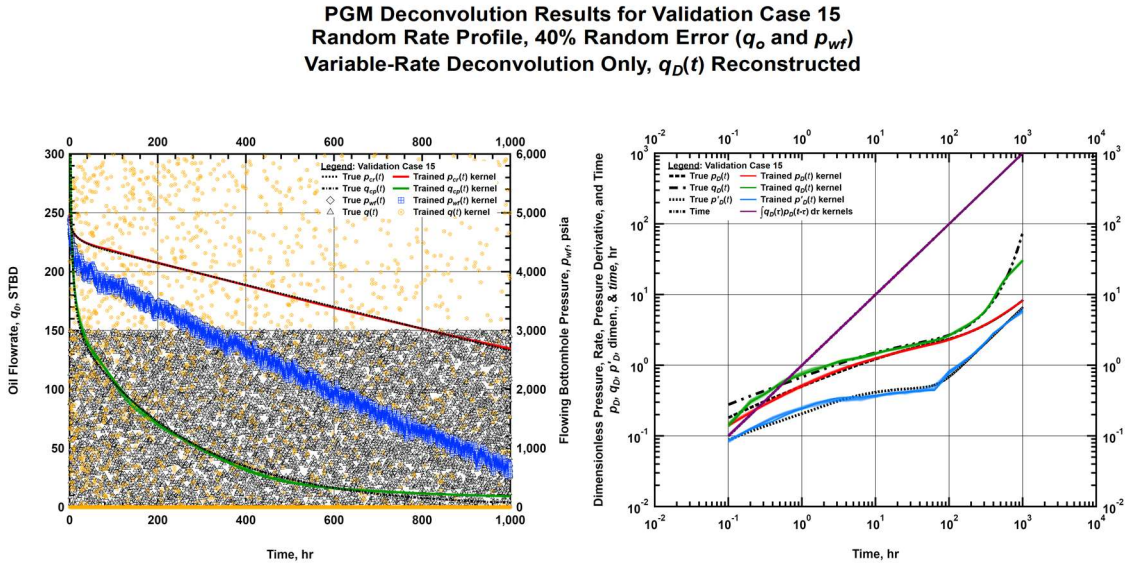

Figure 3.51 — Deconvolution results for Validation Case 15 (random rate, 40% random error).

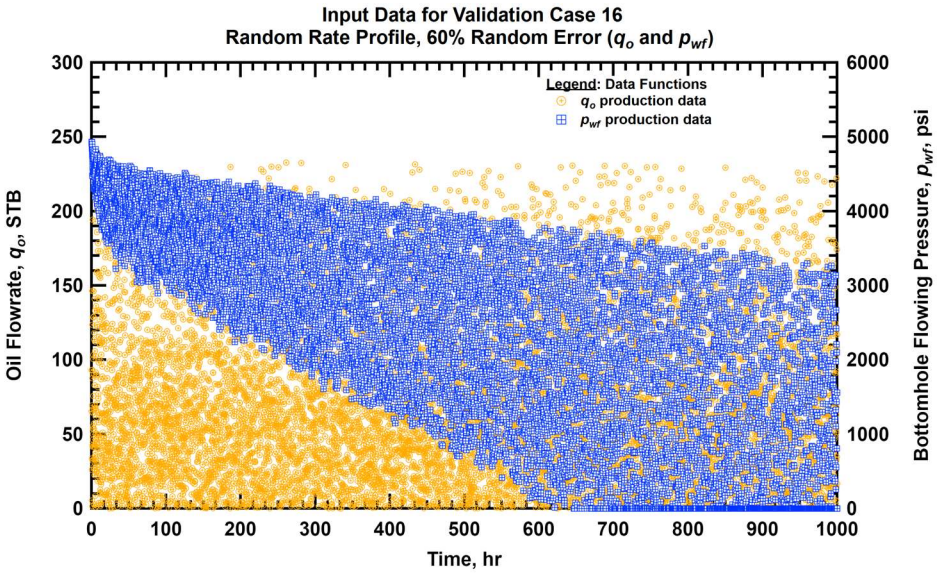

Figure 3.52 — Input data for Validation Case 16 (random rate, 60% random error).

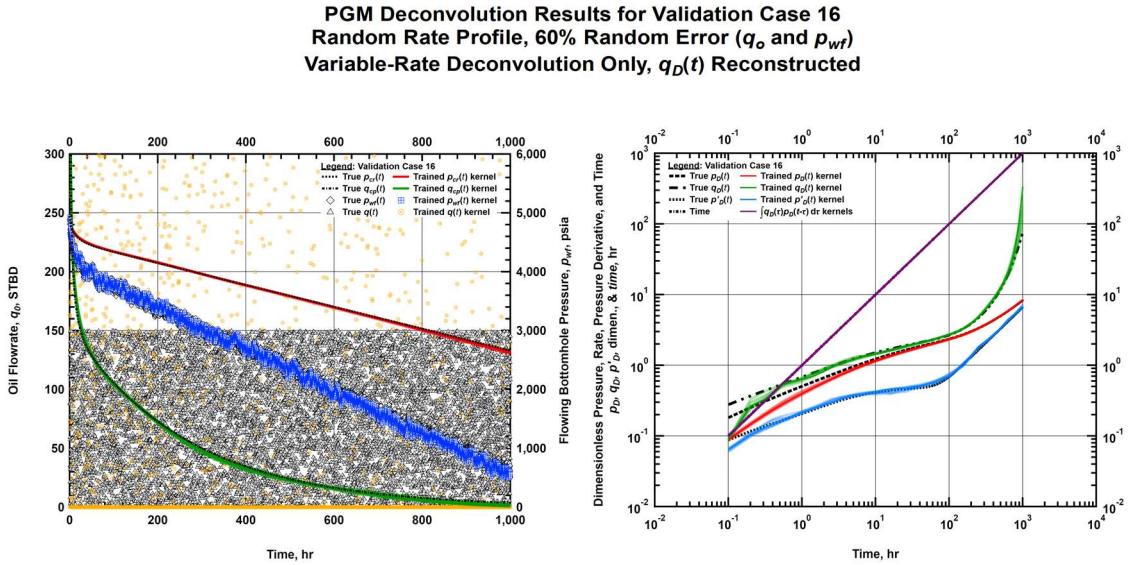

Figure 3.53 — Deconvolution results for Validation Case 16 (random rate, 60% random error).

3.10.5 Deconvolution of Multi-well Variable-Rate Cases

In this section we apply the graphical model of **Fig. 3.11** to scale our method to a multi-well case. Due to our representation of the deconvolution problem as a probabilistic graphical model, extension to the multi-well case requires only minor modification to our method. Recalling **Fig. 3.12**, we imagine a rate or pressure profile for a single well as a 2-dimensional array. The first dimension is time, and the second dimension is the assignment of a well number, e.g. for a single well, the well number is 1.

As we have already shown the capability to deconvolve exact cases, the primary goal of this section is to demonstrate the capability to impute missing rate profiles from multiple wells, for both partial and completely missing rate profiles.

- Validation Case 17 is an exact result from the convolution of the various rate profiles, unique for each well, with the dimensionless pressure function. However, the input data for each well is missing 25% of the total duration. Each well is missing a different period of history.
- Validation Case 18 adds random error to Validation Case 17. Each rate-time and pressure-time pair is randomly adjusted within $\pm$5% of its original value by multiplication with i.i.d. uniform distributions $\mathcal{U}(0.08, 1.05)$.
- Validation Case 19 increases the errors to within $\pm$20% of its original value by multiplication with i.i.d. uniform distributions $\mathcal{U}(0.8, 1.2)$.
- Validation Case 20 changes the missing rate profiles for each well. Well 1 has the entire rate history, while the history is entirely missing for the other wells.

- Validation Case 21 adds $\pm 5\%$ to Validation Case 17 by multiplication with i.i.d. uniform distributions $\mathcal{U}(0.95, 1.05)$.
- Validation Case 22 increases the errors to within $\pm 20\%$ of its original value by multiplication with i.i.d. uniform distributions $\mathcal{U}(0.8, 1.2)$.

The Validation Case 17 is shown in **Fig. 3.54**, **Fig. 3.55**, and **Fig. 3.56**. Similar to Validation Case 4, we are able to impute the missing rate values. However, the addition of multiple wells gives us three distinct observations of the constant-rate pressure function at each time step. We observe a corresponding decrease in the uncertainty of the imputed rate profiles, and they are nearly exact.

Validation Case 18 in **Fig. 3.57**, **Fig. 3.58**, and **Fig. 3.59** also performs very well. The difference with the results in Validation Case 5 is stark; the greater amount of information available across the well set improves the result as compared with that case. We do observe a wider range at the later-time values as compared with the earlier-time values. This is due to the nature of the convolution operator, as earlier-time values influence every time value afterwards.

Validation Case 19, shown in **Fig. 3.60**, **Fig. 3.61**, and **Fig. 3.62**, contains clear oscillation in the pressure derivative in addition to increase uncertainty in the imputed rate values. Additionally, the pressure function is not correctly recovered at early times.

Validation Case 20 is illustrated by **Fig. 3.63**, **Fig. 3.64**, and **Fig. 3.65**. As in Validation Case 17, we are able to obtain near-exact reconstruction of each rate, even with a more difficult problem of entirely missing rate profiles.

Validation Case 21 shows a loss of the early-time values of the pressure function and pressure derivative (**Fig. 3.66**, **Fig. 3.67**, and **Fig. 3.68**. The performance is worse than that of Validation Case 19, which had a greater level of noise. This is due to the reduction in observed data of the completely missing rate profiles.

Validation Case 22, shown in **Fig. 3.69**, **Fig. 3.70**, and **Fig. 3.71**, is the final validation case for this section and demonstrates the difficult in recovering the entire rate profiles with a high level of noise. As before, we note that even though the correct function is not recovered, the algorithm does not exhibit instability or divergence. Instead, it reverts back to a power-law function with a slope based upon its initialization value, as no strong indication of a change from the prior is provided by the data.

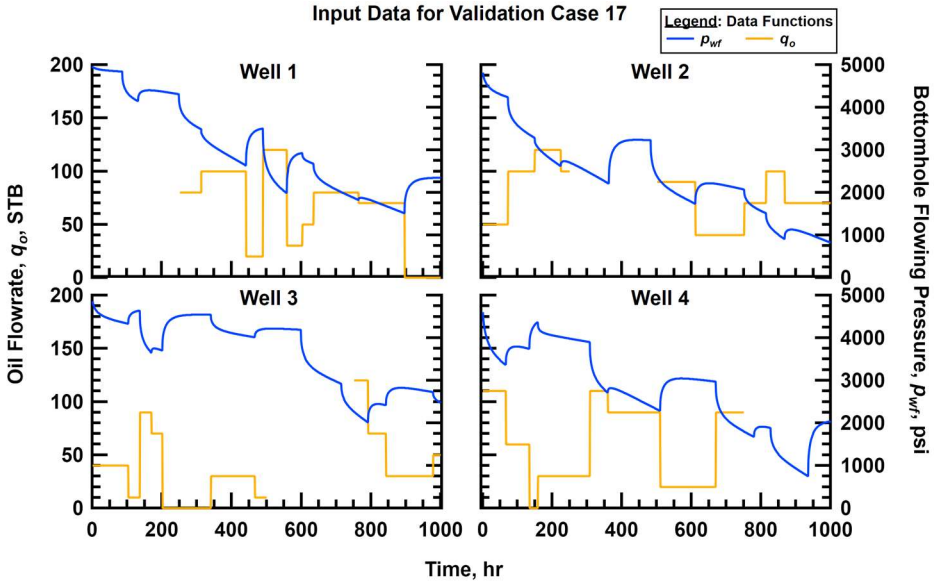

Figure 3.54 — Input data for Validation Case 17 (multiwell, partial missing rates, no error).

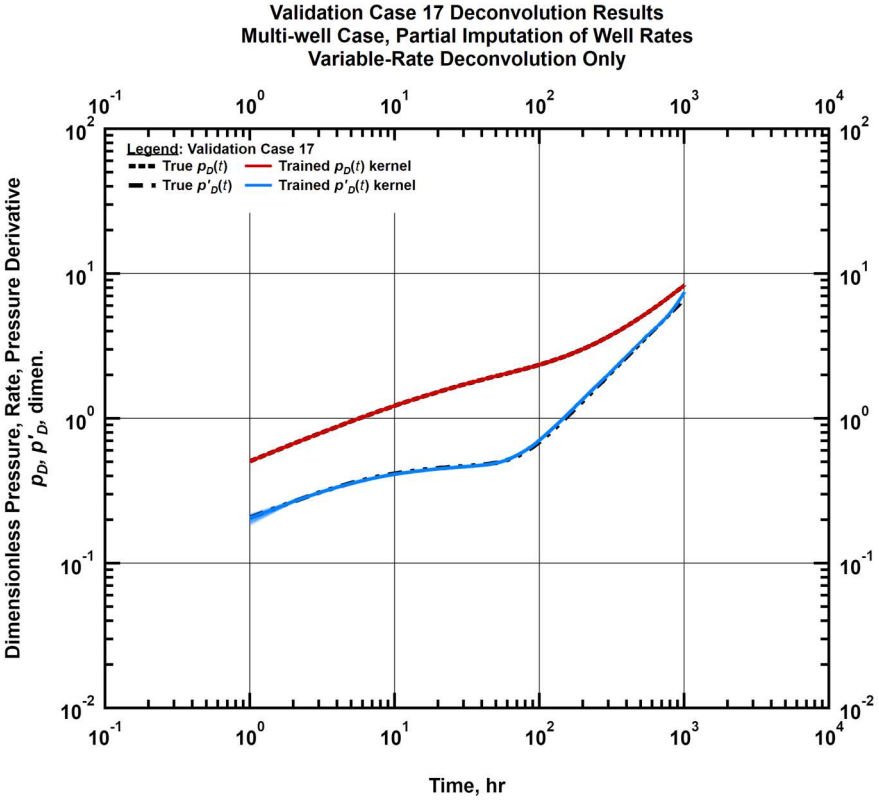

Figure 3.55 — Deconvolution results for Validation Case 17 (multiwell, no error) – Dimensionless Functions.

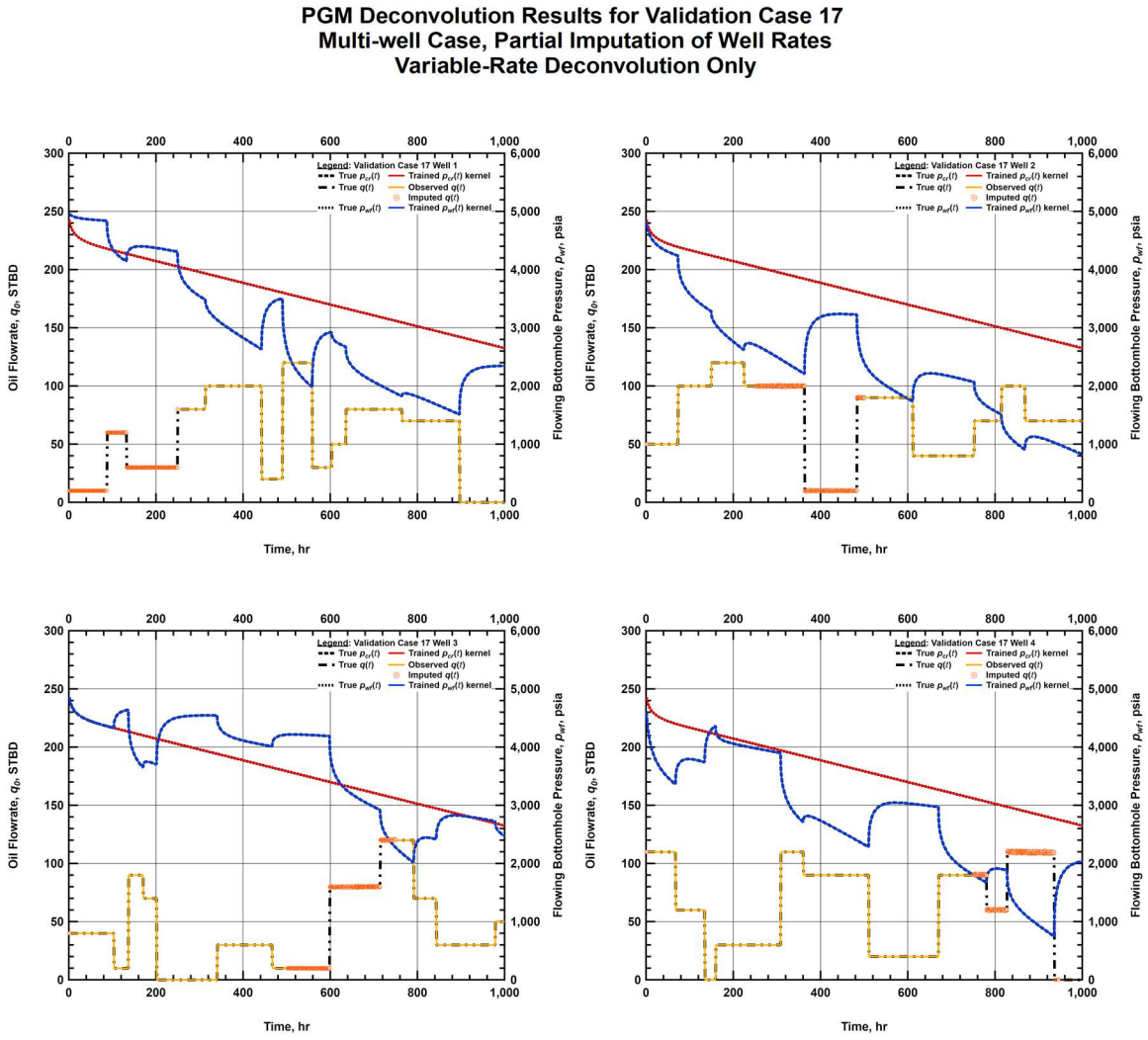

Figure 3.56 — Deconvolution results for Validation Case 17 (multiwell, partial missing rates, no error) – Individual Well Results.

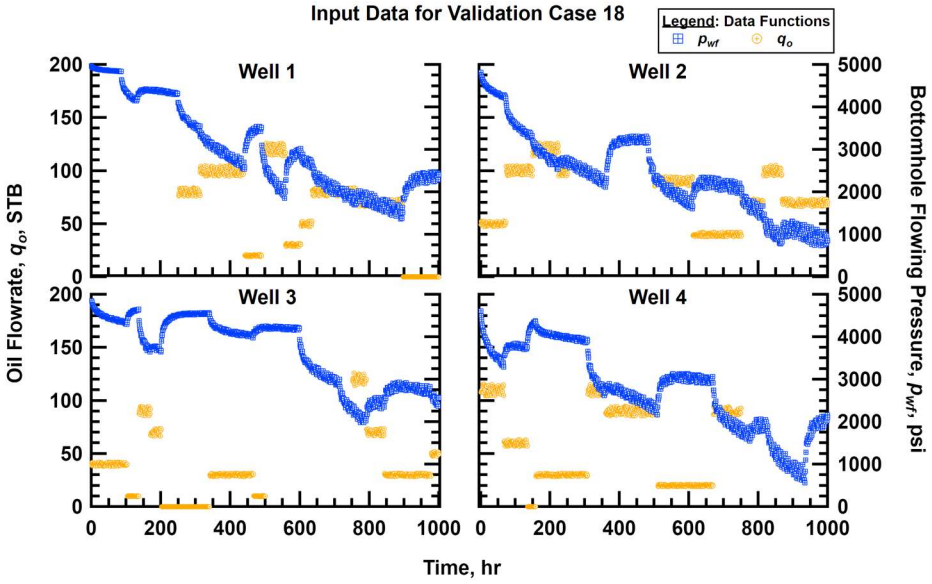

Figure 3.57 — Input data for Validation Case 18 (multiwell, partial missing rates, 5% random error).

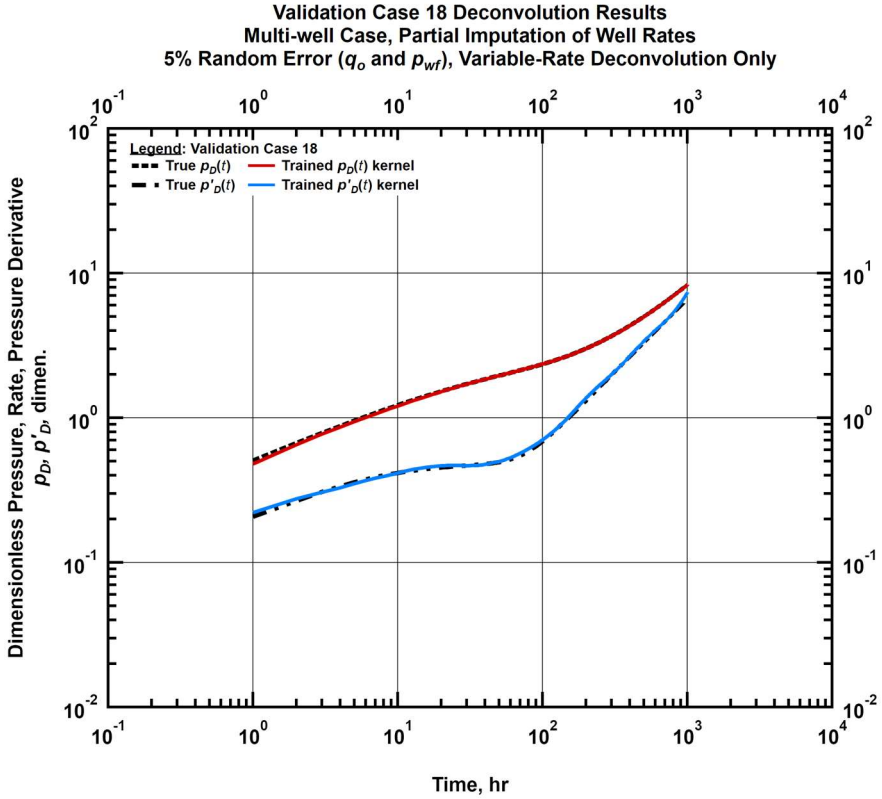

Figure 3.58 — Deconvolution results for Validation Case 18 (multiwell, partial missing rates, 5% random error) – Dimensionless Functions.

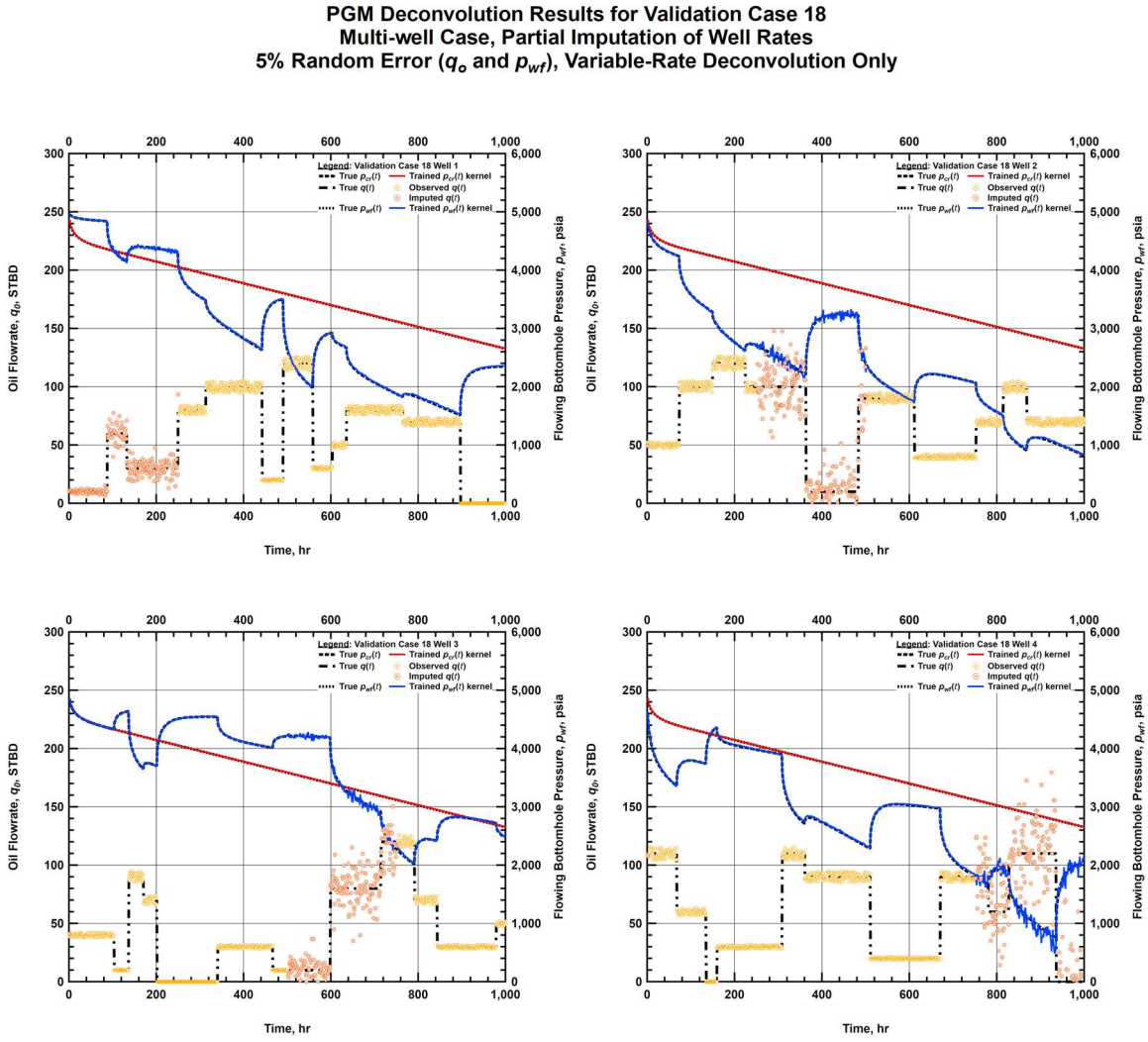

Figure 3.59 — Deconvolution results for Validation Case 18 (multiwell, partial missing rates, 5% random error) – Individual Well Results.

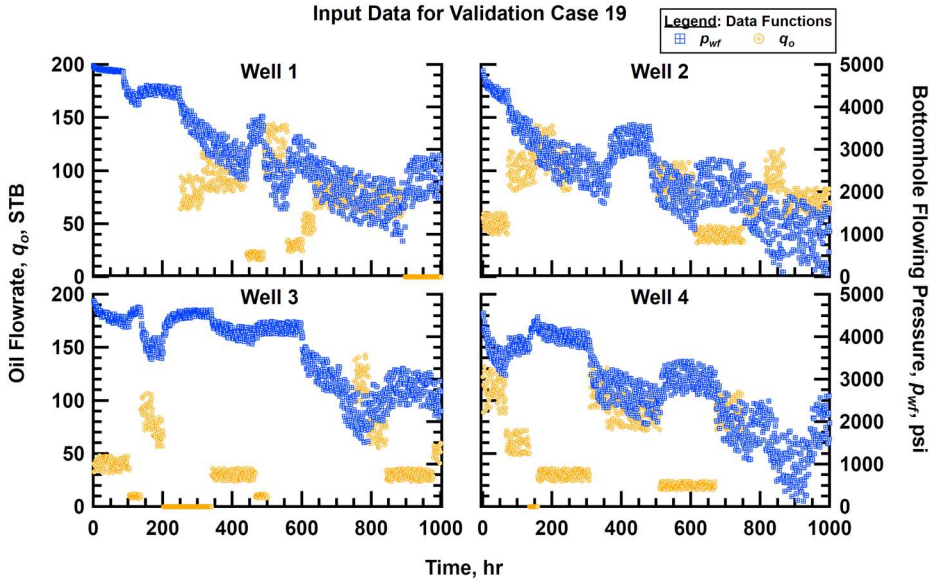

Figure 3.60 — Input data for Validation Case 19 (multiwell, partial missing rates, 20% random error).

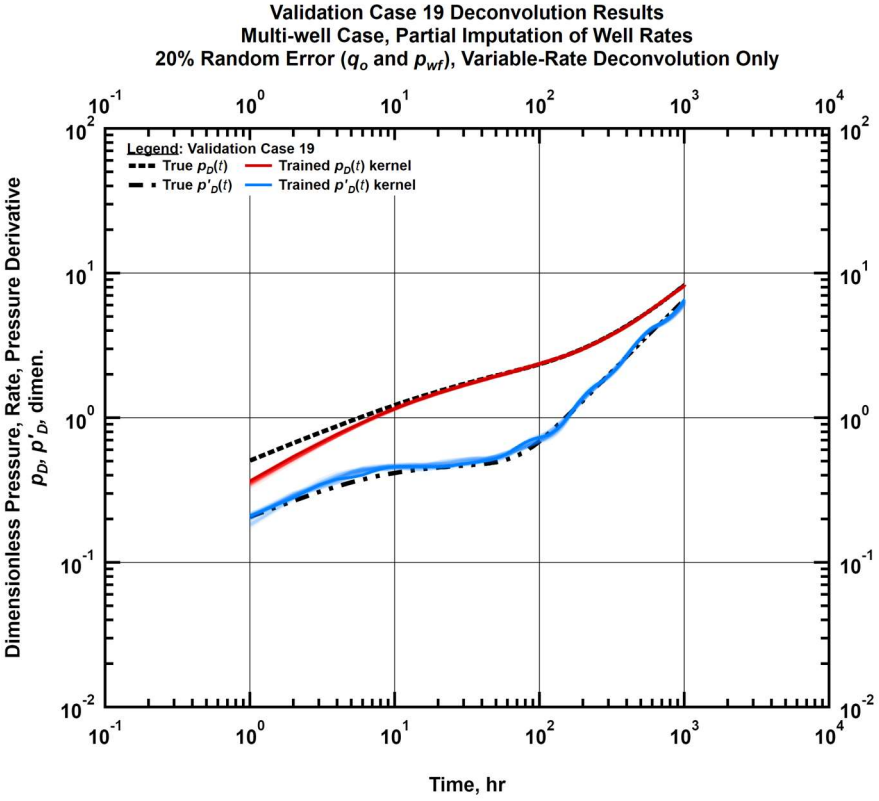

Figure 3.61 — Deconvolution results for Validation Case 19 (multiwell, partial missing rates, 20% random error) – Dimensionless Functions.

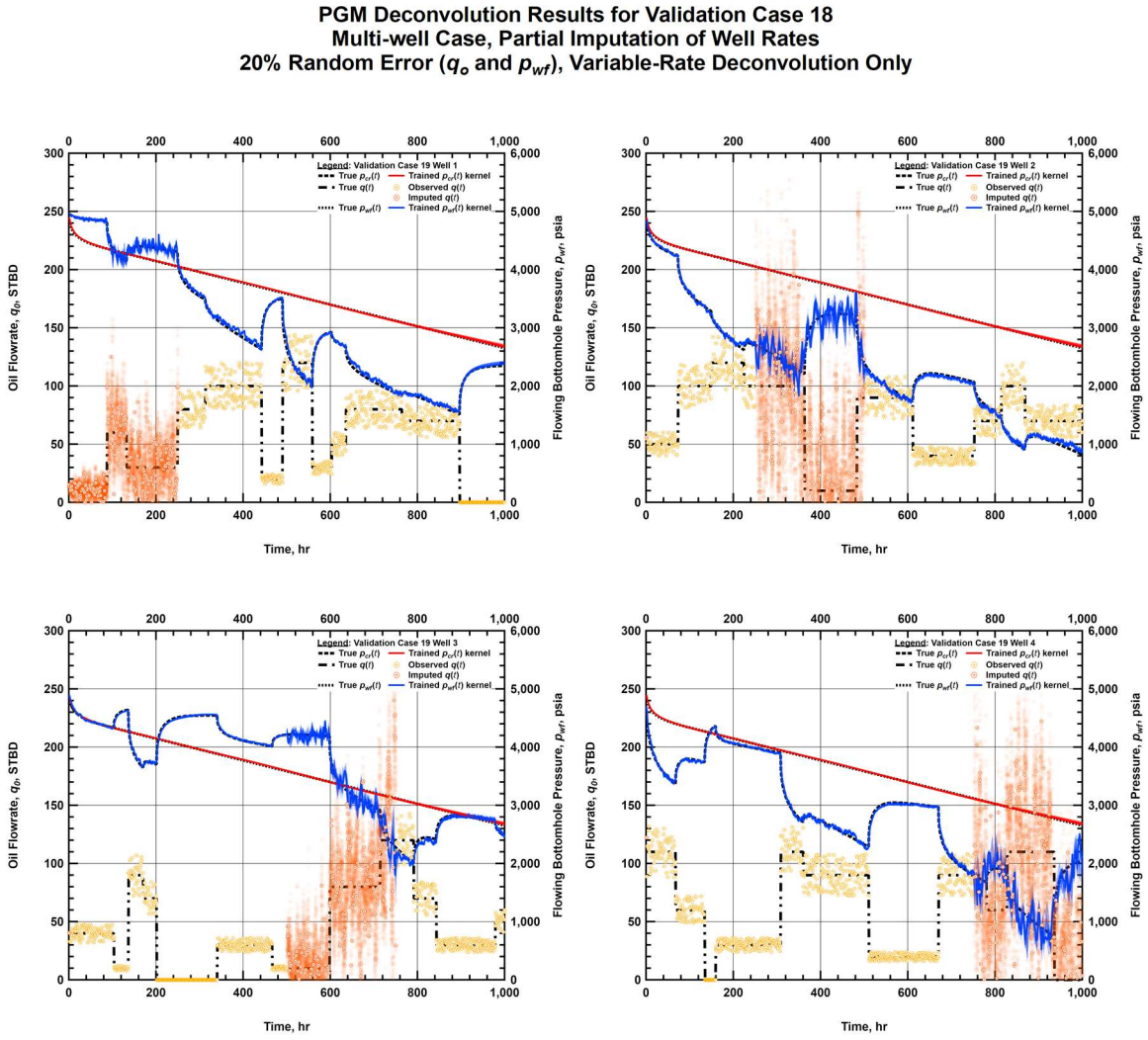

Figure 3.62 — Deconvolution results for Validation Case 19 (multiwell, partial missing rates, 20% random error) – Individual Well Results.

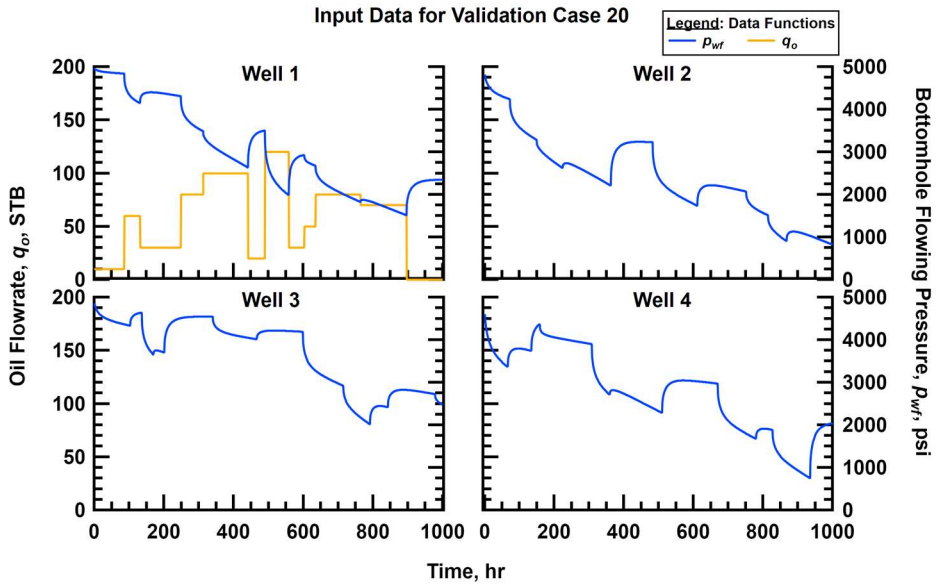

Figure 3.63 — Input data for Validation Case 20 (multiwell, completely missing rates, no error).

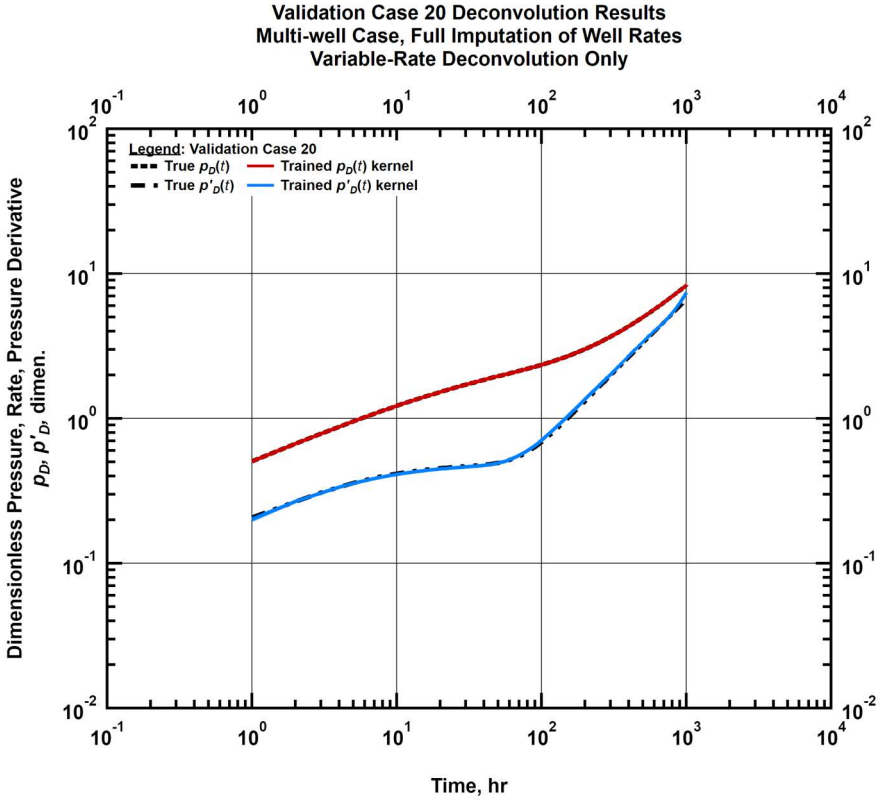

Figure 3.64 — Deconvolution results for Validation Case 20 (multiwell, completely missing rates, no error) – Dimensionless Functions.

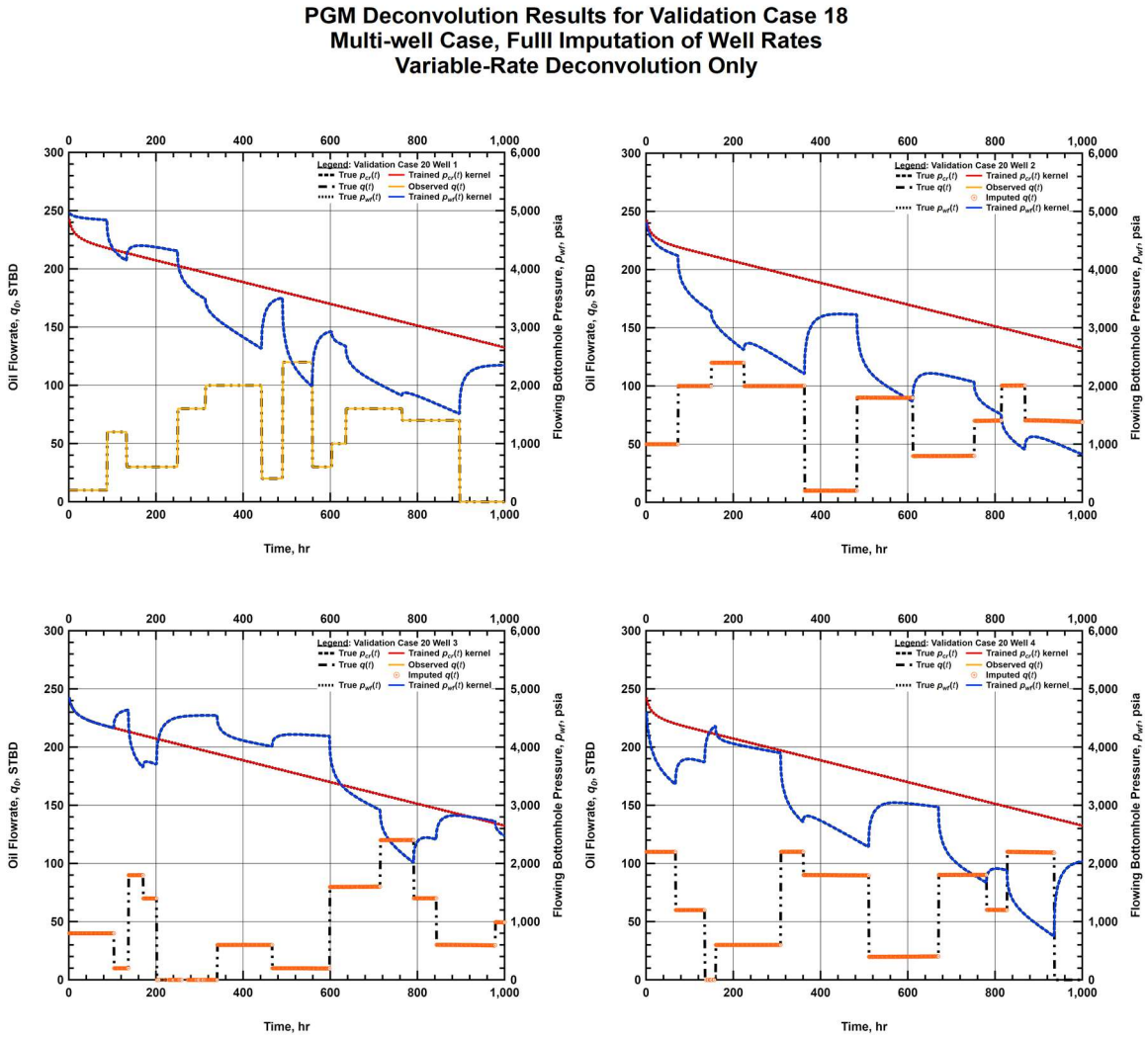

Figure 3.65 — Deconvolution results for Validation Case 20 (multiwell, completely missing rates, no error) – Individual Well Results.

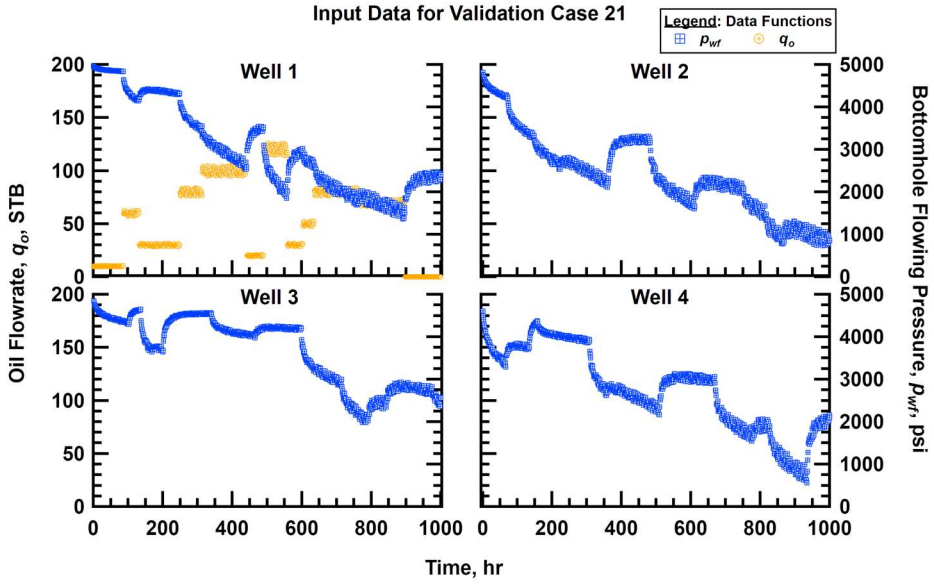

Figure 3.66 — Input data for Validation Case 21 (multiwell, completely missing rates, 5% random error).

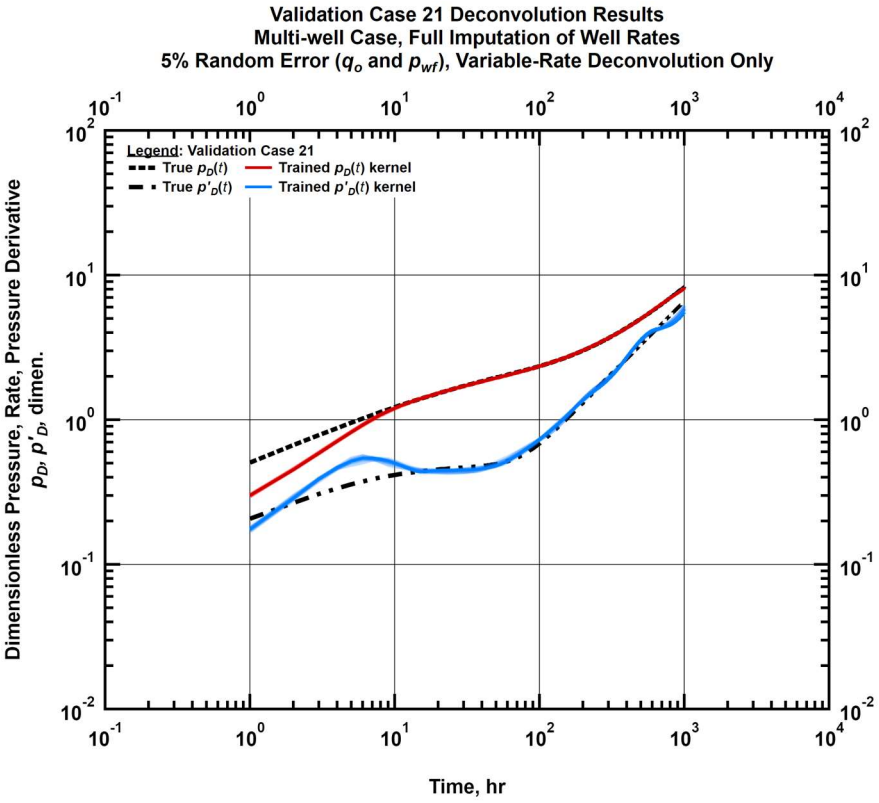

Figure 3.67 — Deconvolution results for Validation Case 21 (multiwell, completely missing rates, 5% random error) – Dimensionless Functions.

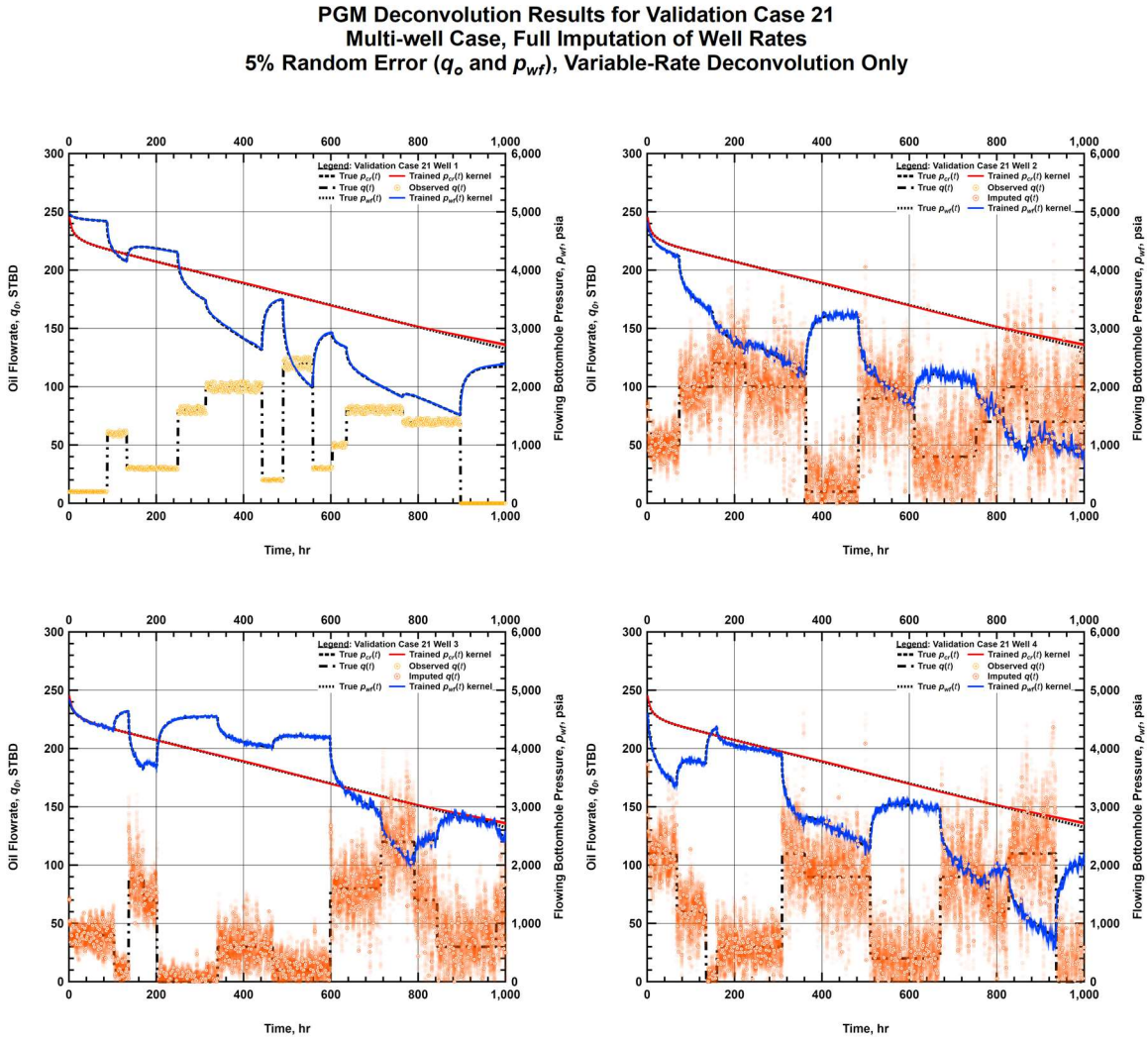

Figure 3.68 — Deconvolution results for Validation Case 21 (multiwell, completely missing rates, 5% random error) – Individual Well Results.

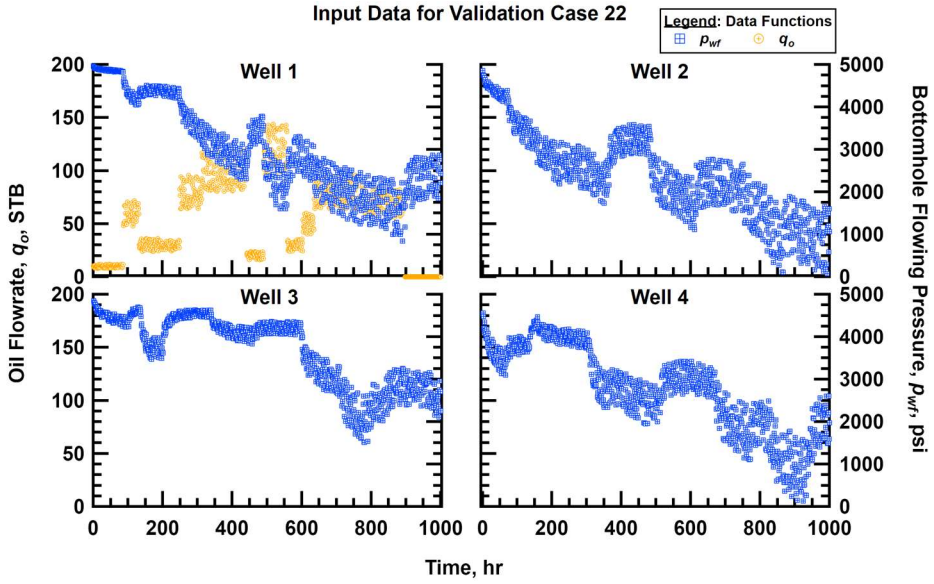

Figure 3.69 — Input data for Validation Case 22 (multiwell, completely missing rates, 20% random error).

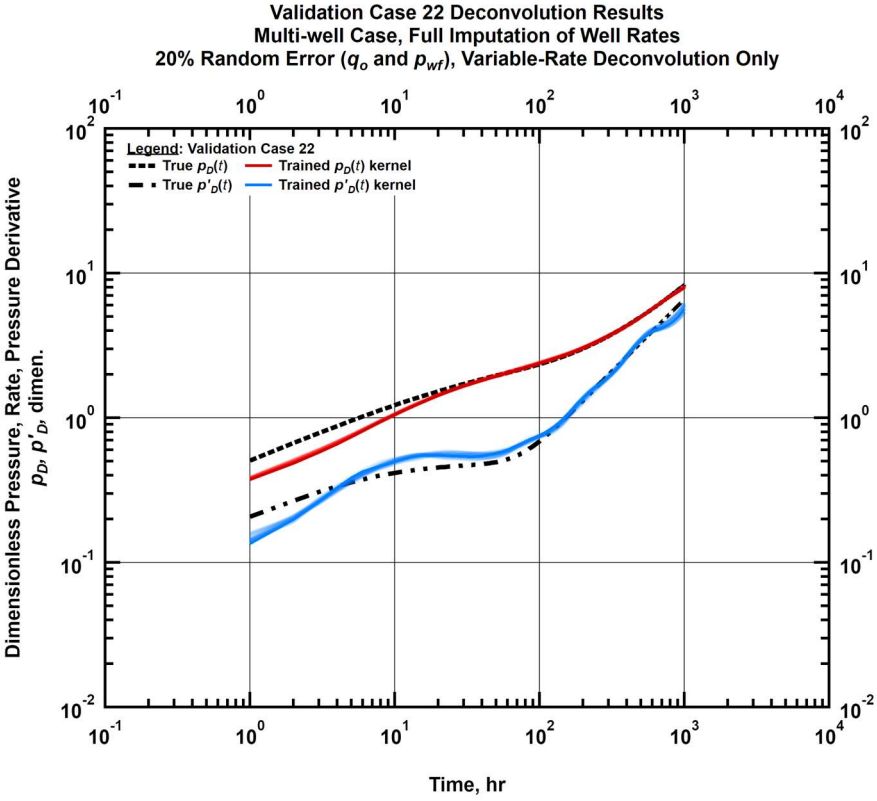

Figure 3.70 — Deconvolution results for Validation Case 22 (multiwell, completely missing rates, 20% random error) – Dimensionless Functions.

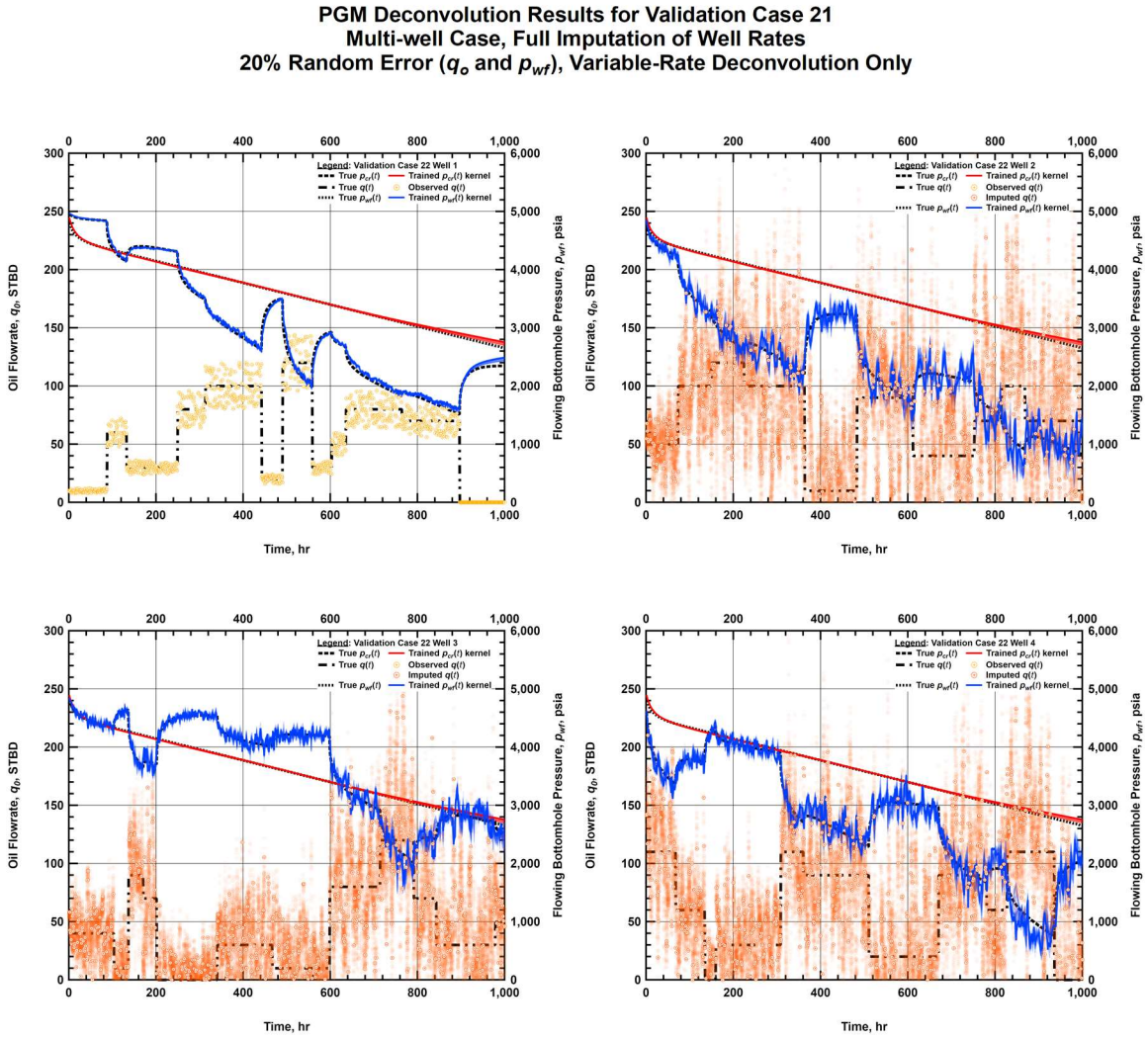

Figure 3.71 — Deconvolution results for Validation Case 22 (multiwell, completely missing rates, 20% random error) – Individual Well Results.

3.11 Application to Field Cases

With validation of the technique and confirmation of the generality of the PGM to any deconvolution problem and tolerance to severe levels of noise, we apply our deconvolution method to two field cases. Given that our method builds upon that of Ilk, Valkó, and

Blasingame (2006), we can state with confidence that it will perform at least as well in the cases Ilk et al. tested. (See Appendix E for a discussion of the differences, and modifications to our method to re-create the Ilk et al. method within our Bayesian framework).

Deconvolution requires that the linearity of the relationship between rate and pressure is maintained. In such cases where this does not hold, we would not expect that our method would be able to invert the correct deconvolved function. However, given that we can enforce the linearity of the rate and pressure solutions with our method by performing a simultaneous correlated deconvolution, we are interested to observe the result in the case where the assumption of linearity is not valid. To perform the simultaneous deconvolution, we require a data set in which both rates and pressures are changing. That is, a situation in which the assumption that the inner boundary condition is completely controlled by the pressure or the flow rate, not both, is invalid. Otherwise, an independent variable-pressure or variable-rate case, respectively, would be more correct to apply.

3.11.1 Attaka Oil Well

We analyze an oil well given by Blasingame (2009) which displays a clear non-linearity at late time (**Fig. 3.72**). We observe a sudden decrease in the flow rate corresponding to a sudden increase in the decline rate, with no observable reaction from the pressure data. We first perform a traditional variable-pressure deconvolution, interpolating values at missing time-steps but otherwise performing no data correction, to obtain the constant-pressure rate function (**Fig. 3.73**).

We first note that, relative to our validation cases, the much smaller size of the data set corresponds to a wider posterior distribution of response. We can see that the uncertainty of the deconvolved constant-pressure rate increases toward the tail of the data, corresponding to the deviation of the reconstructed rate from the data.

Next, we repeat the variable-pressure case, but allow our method to impute missing data values instead of generating them by deterministic interpolation. The result is shown in **Fig. 3.74**. Immediately, we note that the late-time match has been corrected, but the early-time reconstruction is in large error, and the imputed values have an extremely large distribution. Additionally, the uncertainty has increased compared to the former case. It is interesting that the allowance of the model to impute values did not decreases the uncertainty, as this allows it to better match the observed flow rates.

Next, we look at the variable-rate deconvolution case in **Fig. 3.75**. Interestingly, we do not observe nearly as wide an uncertainty range as in the variable-pressure case, nor as wide a range on the imputed values. The cause of this difference is not clear.

Fig. 3.76 is the result of a simultaneous deconvolution with the two functions correlated by use of Duhamel's principle. While the constant-rate pressure function is only minorly affected, we observe that the constant-pressure rate function reverts back to the form it took in the first case with the interpolated data, but with a greater range of uncertainty for the entire history, especially at the tail.

Although it is clear that the linearity of rates and pressures is not holding, our method clearly communicates this result by displaying an increased range of uncertainty where the

data are inconsistent. Compared with the first variable-pressure case, it is clear that without consideration of both cases, the result is an overconfident model in a case that it clearly does not correctly apply. While the validity of the imputed rates and pressure needs more investigation, and we do not claim that it is the correct method for analysis of this data set, it is encouraging that the method communicates the uncertainty of its results. Ideally, for a given well, each case would be run as we have done, and utilized to construct a more complete interpretation of the validity of deconvolution as a valid analysis method for the data set.

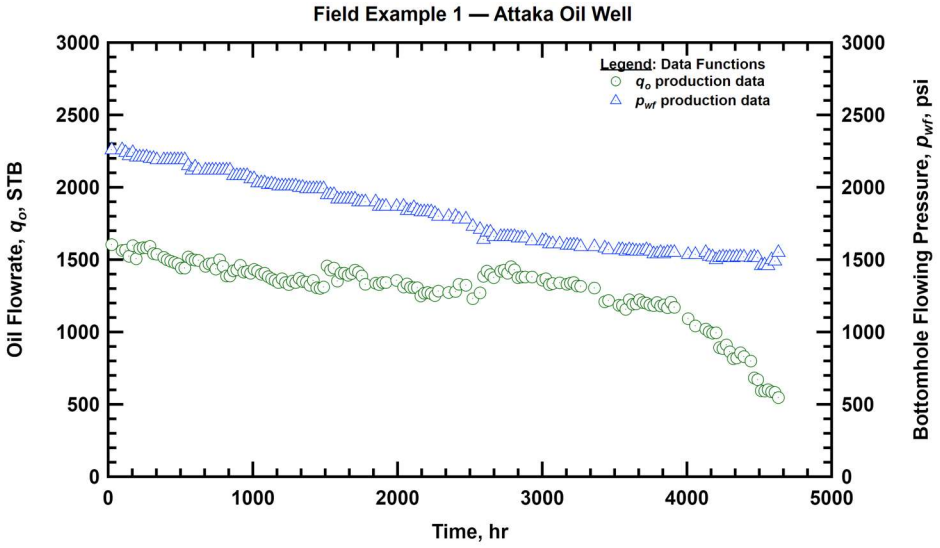

Figure 3.72 — Flowrate and pressure data for the Attaka oil well.

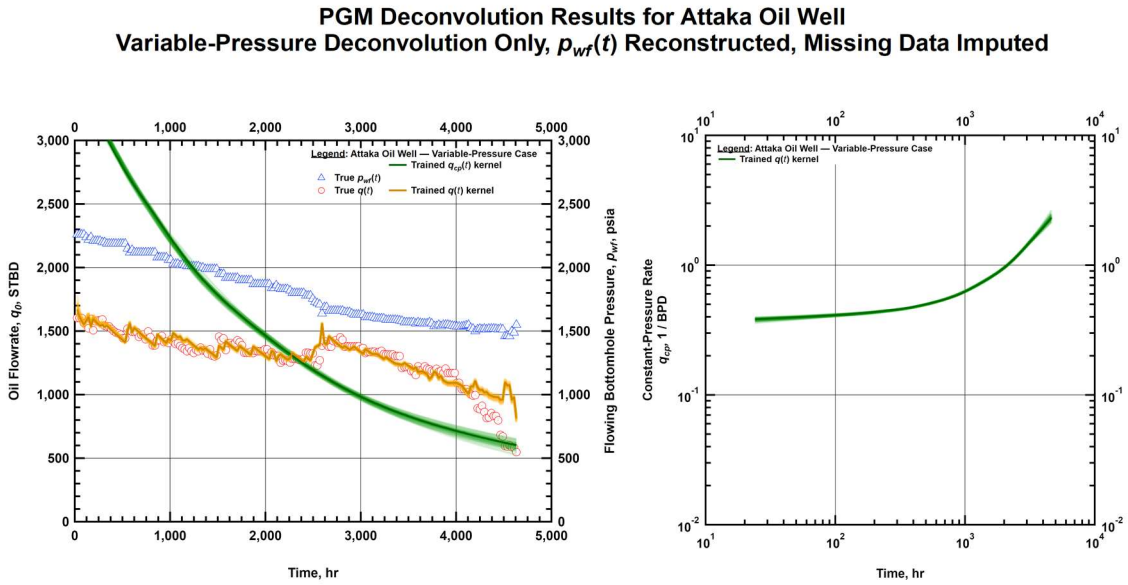

Figure 3.73 — Deconvolution results for Attaka oil well (variable-pressure).

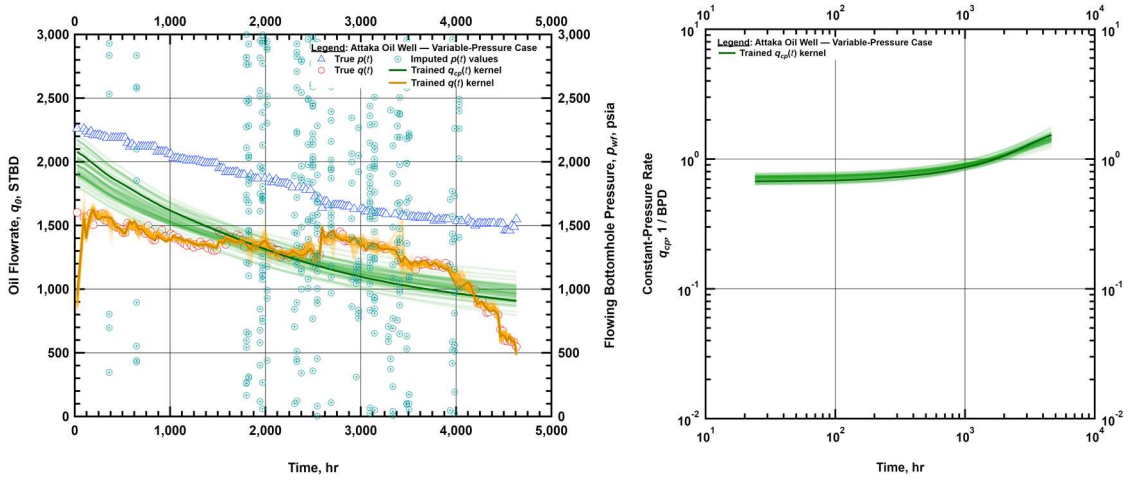

Figure 3.74 — Deconvolution results for Attaka oil well (variable-pressure, imputed values).

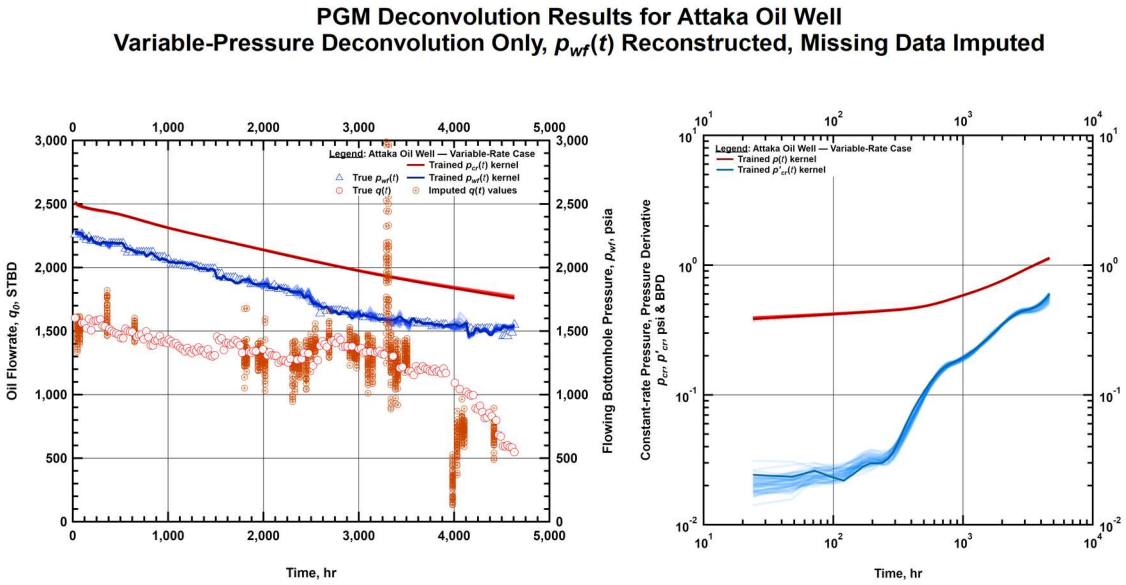

Figure 3.75 — Deconvolution results for Attaka oil well (variable-rate, imputed values).

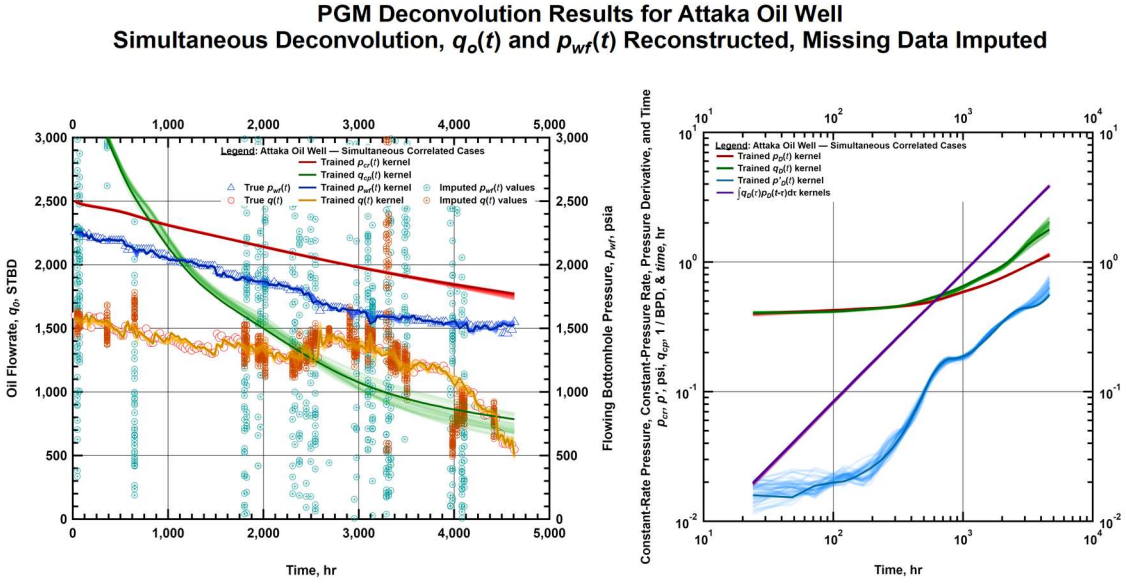

Figure 3.76 — Deconvolution results for Attaka oil well (simultaneous case).

3.11.2 East Texas Gas Well

Our second field case is a re-presentation of a well analyzed by Pratikno, Rushing, and Blasingame (2003), as well as Ilk, Valkó, and Blasingame (2006). The East Texas Gas Well (**Fig. 3.77**) is a tight gas well that has had daily monitoring of production data since it was put on production. The data set has slightly less than a year of production history, and contains multiple periods of flow and shut-ins. While the data quality is good, we do note inconsistencies in the data, such as around 1500 – 2000 hours where a pressure increase corresponds to a rate increase. As it is a gas well, we use pseudopressure to linearize the rate-pressure relationship. Using pseudotime would of course be ideal for analyzing a gas case, but our current method does not include the capability for changing the timesteps during the Hamiltonian Monte Carlo simulation; we reserve this as a subject for future research.

We first analyze the variable-rate deconvolution case (**Fig. 3.78**) and see that, while the maximum likelihood estimate is a good match of the data, the limited resolution of low-frequency daily measurements, along with the inconsistencies in the data, lead to considerable uncertainty in the posterior distribution of the constant-pressure rate function, and correspondingly, the reconstruction variable-pressure rate function. We observe that the late-time response appears to be behaving opposite of what we would expect to see for a transition to boundary-dominated flow.

Analyzing the variable-pressure deconvolution (**Fig. 3.79**) shows that there is considerable more certainty in the pressure responses, particularly during the shut-in build up periods.

Anecdotally, this is consistent the generally higher regard for pressure build-up data as being of "higher-quality" than production data.

Fig. 3.80 is the result of the simultaneous deconvolution. We see good correlation between the rate and pressure functions, a considerable decrease in uncertainty of the rate functions, as well as a minor decrease in the pressure functions. In effect, we see that the confidence gained from the higher-quality pressure build-up data is "transferring" from the variable-pressure case to the variable-rate case, and vice versa. The constant-pressure rate function is behaving more like we would expect, in that the late-time transition is toward boundary-dominated flow as opposed to away from it. In summary, we see that the simultaneous deconvolution improves upon both cases.

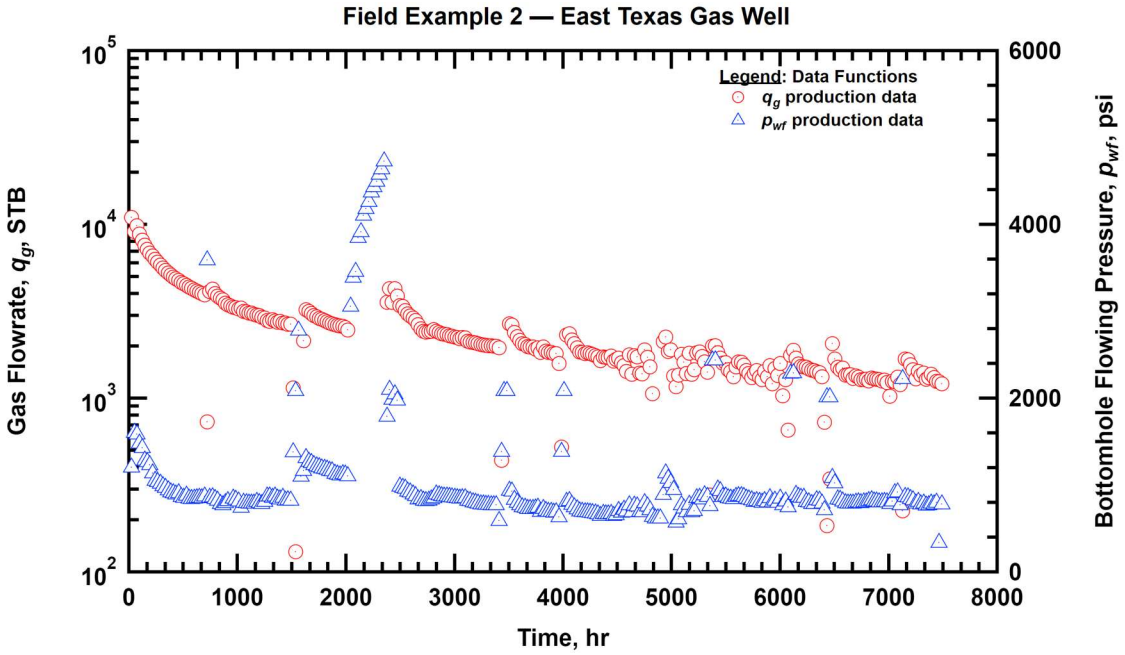

Figure 3.77 — Flowrate and pressure data for East Texas gas well.

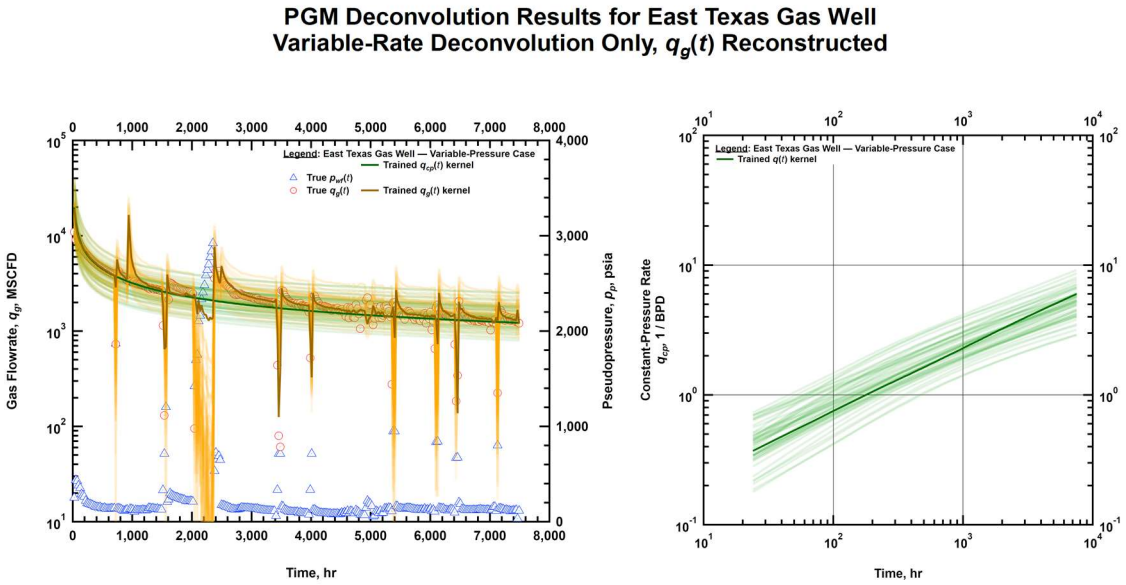

Figure 3.78 — Deconvolution results for East Texas gas well (variable-pressure).

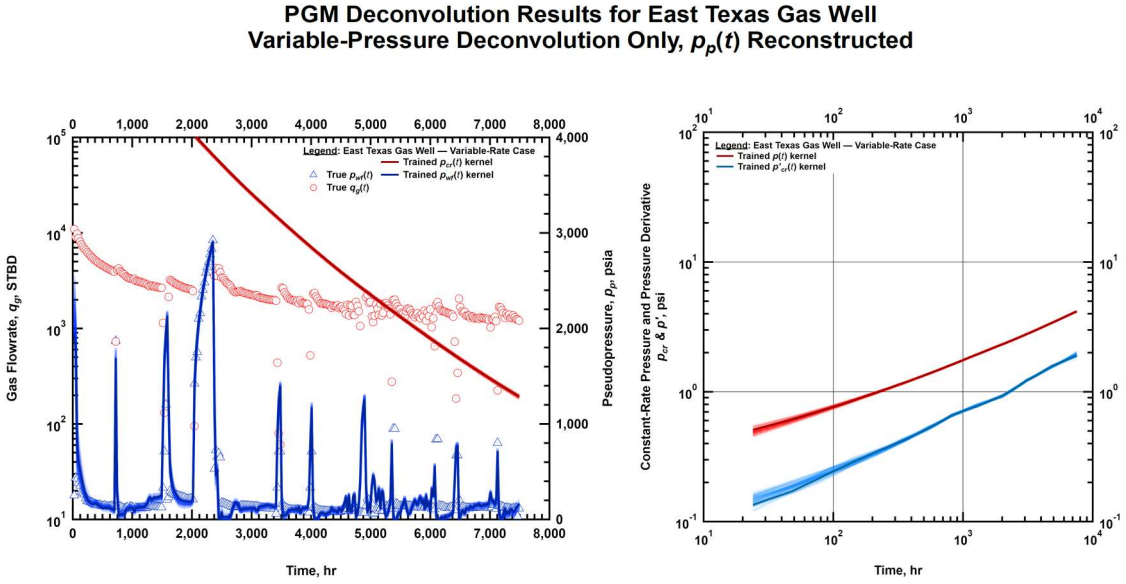

Figure 3.79 — Deconvolution results for East Texas gas well (variable-rate).

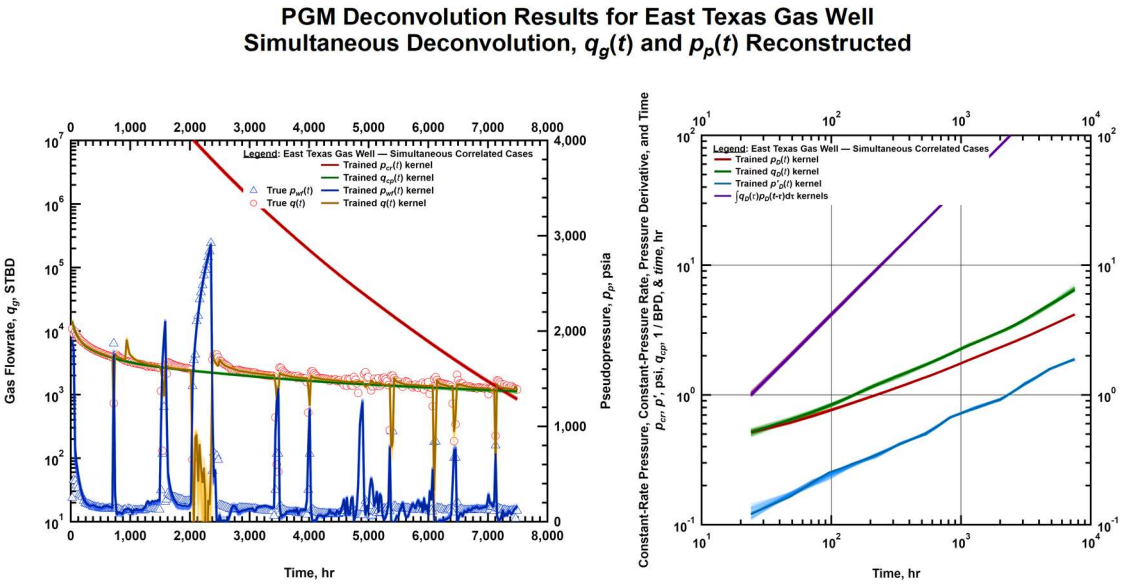

Figure 3.80 — Deconvolution results for East Texas gas well (simultaneous, correlated).

4. NEW METHOD FOR BAYESIAN DIFFERENTIATION

4.1 Introduction

This chapter presents a new methodology for computing derivative functions of rate and pressure data. We show that the developed methods for non-parametric representation of a function using B-splines, the integration of that function, and a robust cost function to regress the spline function, can be applied generally to compute ***any*** derivative where it is valid to approximate the derivative function with linear B-splines. Practically, this applies to essentially all rate and/or pressure histories.

The primary goal is to provide a robust method for computing derivatives for "high-frequency" permanent downhole gauge and "long-term" production data. The approach yields a derivative as a smooth, continuous function using a priori information that is available to a practitioner before they have ever observed a specific data set. This is in contrast to the widely used approach of Bourdet et al. (1989), which constructs a derivative as a series of discrete functions that use a priori information in the form of "smoothness over log cycles".

The work by Bourdet et al. (1989) is concerned with well-testing applications of pressure derivatives. In contrast, while we note the applications of our method to well-testing, we are providing a methodology for computing derivatives of any kind and are less concerned with the specific application. Furthermore, validation of the approach for computing derivatives also provides some level of validation for the deconvolution method, as the

deconvolution method implements the derivative method as the algorithm by which it obtains the unknown kernel function prior to the convolution step.

Our expectation is that the result of our new method can achieve greater levels of smoothing with no loss in signal due to a) modeling the derivative function as a continuous function, and b) optimizing with a robust cost function. Furthermore, sections of the data that would appear to be "high-noise" with the current approach will be interpreted as sections of "high-uncertainty" in the Bayesian interpretation of the data. Finally, the nature of our B-spline approach enables us to easily process multiples of thousands of data points as we do not require interpolation before integration; we may wait until we have computed the final integral.

4.2 Development of the Method

In the construction of a graphical model to represent a derivative, the derivative is a latent variable, and the data we wish to find the derivative for is observed. This model is shown in **Fig. 4.1**.

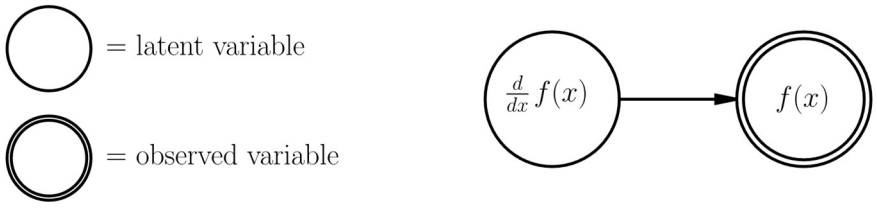

Figure 4.1 — Probabilistic graphical model for derivative of a function.

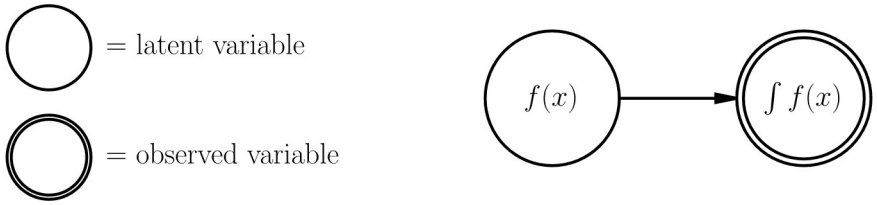

Figure 4.2 — Alternative probabilistic graphical model for derivative of a function.

If we instead consider the problem not as differentiation of observed data, but the unknown derivative as a function $f(x)$, and the observed data is the integral of the latent function, we arrive at the model shown in **Fig. 4.2**.

As the function $f(x)$ is a non-parametric function in a Bayesian model, we require some information to place priors upon it. The assumptions regarding $f(x)$ follow from the primary focus of this work.

- First, we have constrained $f(x)$ to be strictly positive.
- Second, we assume that $f(x)$ can be represented by some number of knots per interval, with a Gaussian distribution for the step distance at each knot.
- Third, we assume a smoothness for $f(x)$ and place a prior upon its curvature.

It stands to reason that these assumptions must be reviewed for each specific application. In doing so, we recognize that each has a translation to a specific step within our algorithm or a hyperparameter. The implementation of the assumptions are:

- Transformation of $f(x)$ using $\exp[f(x)]$ or $\text{softplus}[f(x)]$ enforces positivity.
- The number of knots, n, is adjustable. Recalling Eq. 3.95 and Eq. 3.96:

$$z = \text{ceiling}\left(\log 10\left[\frac{x_n}{x_1} \right] \right) \quad \text{.....3.95}$$

$$\tau_i = \tau_1 + \frac{i}{n-1}(\tau_n - \tau_1) \quad \text{for} \quad -k \le n \le k + 5z + 1 \quad \text{.....3.96}$$

And changing the range of evaluation:

$$\tau_i = \tau_1 + \frac{i}{n-1}(\tau_n - \tau_1) \quad \text{for} \quad -k \le n \le k + rz + 1 \quad \text{.....4.1}$$

We place *r* evenly spaced knots every *z* interval, where *z* is using Eq. 3.95 or some other arbitrary method.

The curvature of $f(x)$ is given by:

$$K = \frac{\frac{d^2 f}{dx^2}}{\left[1 + \left[\frac{df}{dx}\right]^2\right]^{3/2}} \quad \text{.....4.2}$$

Where *K* is evaluated over Cartesian scale. If smoothness is enforced with respect to the log of *x*, then Eq. 4.3 must be used where $d\tau = d\ln t = \frac{1}{t}dt$:

$$K = \frac{\frac{d^2 f}{d\tau^2} + \frac{df}{d\tau} - \left[\frac{df}{d\tau}\right]^2}{\left[1 + \left[\frac{df}{d\tau}\right]^2\right]^{3/2}} \quad \text{.....4.3}$$

If smoothness is enforced with respect to the log of $f(x)$, then

$$df \to d\ln f(x) = \frac{1}{f(x)} df(x)$$

can be substituted in Eq. 4.2 or Eq. 4.3. Derivations for Eq. 4.2 and Eq. 4.3 are given in **Appendix B**.

4.3 Implementation Details

We have presented the solution for the coefficients of B-splines of any degree, but it is appealing and convenient to use B-splines of degree 1 as it obviates the expensive computation of a pseudo-inverse matrix. Alternatively, we can transform the integral from a linear function, i.e. $y = mx+b$, using any type of pure function that acts on x or y; i.e. a function where the output is only determined by the input without observable side effects. Practically, this means we can compute derivatives for linear, power-law, exponential, and logarithmic functions. **Table 4.1** lists the x and y transforms required for these functional forms. We may extend this to other concepts as well, such as polynomial or trigonometric functions, etc., although these are not of interest to this work and are not discussed here.

Table 4.1 — Summary of variable transformations for B-spline integration

Function	x Transform	y Transform
Linear	$f(x) = x$	$f(y) = y + b$
Power-law	$f(x) = \ln x$	$f(y) = be^y$
Logarithmic	$f(x) = \ln x$	$f(x) = y + b$
Exponential	$f(x) = x$	$f(y) = be^y$

The pseudocode implementation of **Fig. 4.2** for an arbitrary order of derivative is:

```
with model:
   # construct B-spline integral model for inference of
   # derivative function

   j = 1   # order of derivative to computed

   beta = GaussianRandomWalk()     # slope term
   beta = add_edges(beta, k) # add k+1 knots at ends
   beta = softplus(beta)           # enforce positivity

   while j - i ≥ 0; i=1, i++:
      alpha[i] = HalfFlat()        # intercept term

      coef = solve(beta)           # compute coefficients

      if j - i == 0:
         # integrate and interpolate with f(x)
         y = int_interp(coef, f_x(x, i))
      else:
         # integrate with f(X)
         y = integrate(coef, f_x(x, i))

      y -= y[0]   # remove constant of integration
      y = f_y(y, alpha[i], i)# transform with f(y)

   # Compute model likelihood with cost function
   σ = Exponential()               # determine model sigma
   likelihood = Huber(μ=log(data), σ=σ, t=.1,
      observed=log(y))

   # Reguarlization
   curvature = curvature((f_X(x, k), beta)
   λ = 1 / Exponential() # determine regulariztion sigma
   likelihood *= Normal(μ=0., σ=1 / λ,
      observed=curvature)

   model.sample()
```

And for comparison, the algorithm for the Bourdet et al. (1989) derivative method is:

```
function float[] get_end_L(float[] dx, float L):
   # find leftmost point of smoothed range
   idx = argwhere(dx ≤ L & dx ≥ 0.0)    # non-zero elements
   if len(idx) > 0:
      return min(len(dx) - 1, idx[-1] + 1)

   return len(dx) - 1

function float[] get_end_R(float[] dx, float L):
   # find rightmost point of smoothed range
   idx = argwhere(dx ≤ L & dx ≥ 0.0)    # non-zero elements
   if len(idx) > 0:
      return max(0, idx[0] - 1)

   return 0

function float[] get_L(float[] dx, float L):
   # get change at left edge
   idx = argwhere(dx ≤ L & dx ≥ 0.0)    # non-zero elements
   if len(idx) > 0:
      return min(0, idx[0] - 1)

   return -1
```

```
function float[] get_R(float[] dx, float L):
   # get change at right edge

   idx = argwhere(dx ≤ L & dx ≥ 0.0)    # non-zero elements
   if len(idx) > 0:
      return min(len(dx) - 1, idx[-1] + 1)

   return 0

function float[] bourdet(float[] x, float[] y, int L,
   bool xlog = True, bool ylog = False):
   # compute Bourdet method smoothed derivative

   logx = log10(x) # must smooth over log base 10 cycles
   L *= log(10)    # transform from base 10 to base e

   if xlog:
      x = log(x)
   if ylog:
      y = log(y)

   # generate arrays of zeros
   n = len(x)
   x_L = zeros(n)
   x_R = zeros(n)
   y_L = zeros(n)
   y_R = zeros(n)

   # get points for forward and backward derivatives
   k1 = get_end_L(logx - logx[0], L)
   k2 = get_end_R(logx[-1] - logx, L)

   # compute bourdet derivative
   for i in range(k1, k2):
      dx = logx[i] - logx[:i]
      idx = get_L(dx, L)
      x_L[i] = dx[idx]
      y_L[i] = y[i] - y[:i][idx]
```

```
39
40          dx = logx[i:] - logx[i]
41          idx = get_R(x, L)
42          x_R[i] = dx[idx]
43          y_R[i] = y[i:][idx] - y[i]
44
45      der = (y_L / x_L * x_R + y_R / x_R * x_L) / (x_L + x_R)
46      if not xlog:
47          der /= x
48
49      # compute forward difference at left edge
50      for i in range(0, k1):
51          idx = get_end_L(logx - logx[i], L)
52          dx = x[idx] - x[0]
53          dy = y[idx] - y[0]
54
55          der[i] = dy / dx
56
57      # compute backward difference at right edge
58      for i in range(k2, n):
59          idx = get_end_R(logx[i] - logx, L)
60          dx = x[-1] - x[idx]
61          dy = y[-1] - y[idx]
62
63          der[i] = dy / dx
64
65      return der
```

4.4 Validation

Our first validation case is the case presented by Bourdet et al. (1989) of a well in a circular reservoir with wellbore storage. Although the authors present a calculation for $L = 0.1$ in their work, we have performed our own implementation and calculations for this work so we may also compare the methods additional cases. For our spline function, we have chosen 10 knots per log cycle, and let the algorithm automatically determine the correct

standard deviation of the cost function, σ. We also assume a power-law form of the integral. **Fig. 4.3** presents our results in which we have also allowed the algorithm to automatically determine the level of regularization upon the curvature of the derivative function, λ. Recall from Eq. 3.34 that a larger value of λ equates to a greater level of regularization.

In comparing the result of the approaches, we note that our method generates an overall smoother function with no loss in resolution of the reservoir signal. We also observe that, rather than noise in the output of our algorithm, we instead obtain a wider posterior distribution. The posterior is particularly wide on the right-hand edge where the Bourdet et al. method using $L = 0.1$ has six data points that incur edge effects. We note that we do not require any special treatment of edge effects.

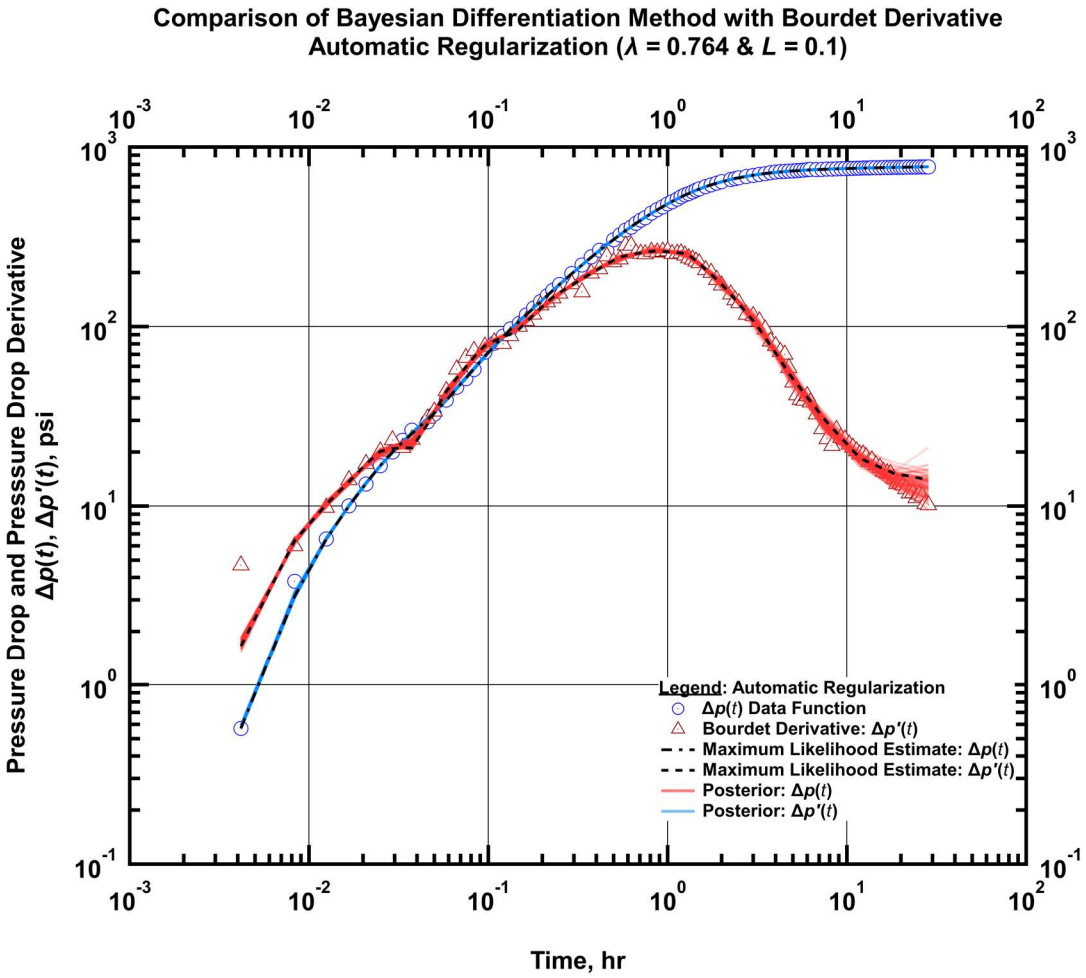

Figure 4.3 — Comparison of Bayesian derivative method with Bourdet et al. (1998) case using automatic regularization, λ = 0.764, vs. $L = 0.1$.

For comparison, **Fig. 4.4** presents the same case where no regularization is used for either method. For our approach, this means we do not add the prior placed upon to the model rather than setting $\lambda = 0$, which would result in a division by zero. The Bourdet et al. approach typically communicates this with the statement $L = 0.0$. This case highlights the strength of our method; because the spline function is a continuous function, it is not

possible to achieve the same level of noise as seen in the Bourdet et al. method (owed to their finite difference approach).

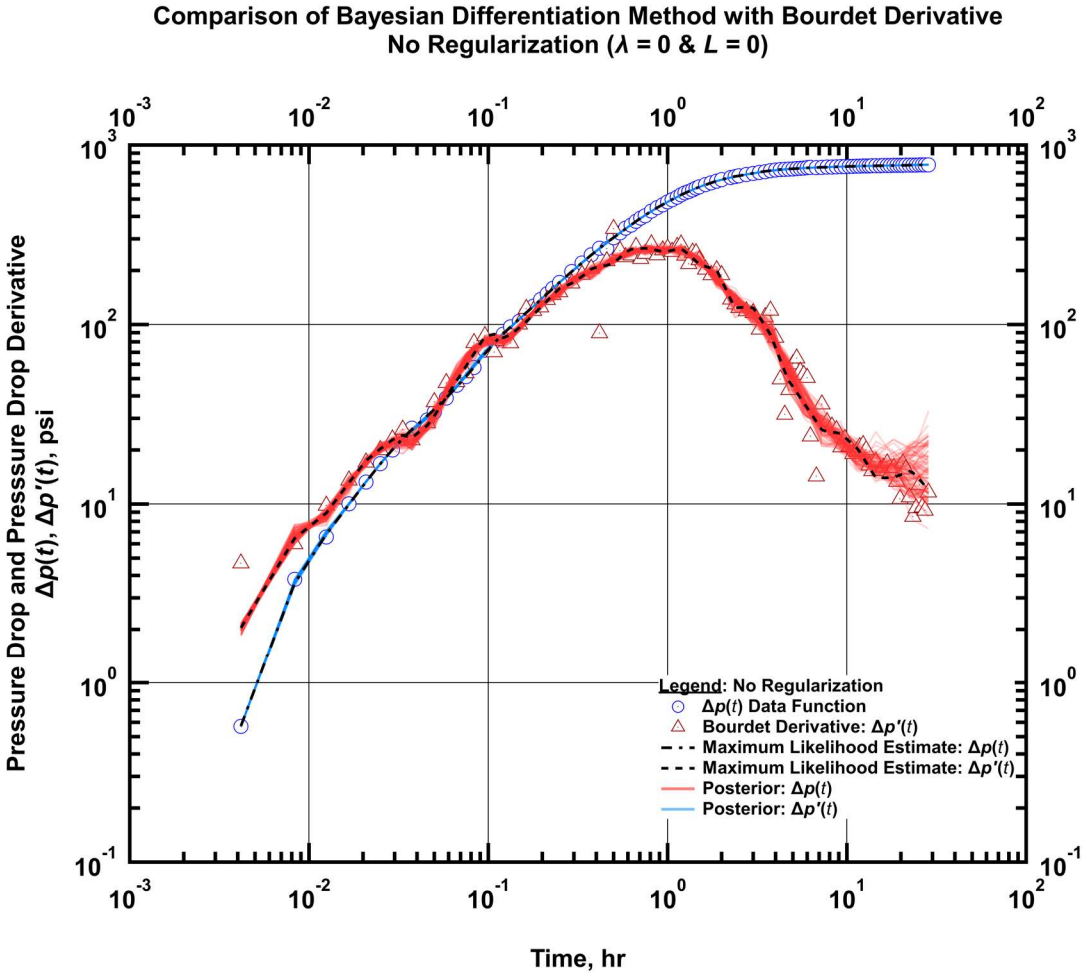

Figure 4.4 — Comparison of Bayesian derivative method with Bourdet et al. (1998) case using no regularization vs. $L = 0.0$.

Fig. 4.5 presents a case of over regularization using our method. Interestingly, there is a sharp kink around 1.05 hr, even more so than the other cases, while the rest of the noise is

smoothed. The prior placed upon curvature ensures a high cost for any such kink, but the overall model must result in a more likely answer when such a kink is included.

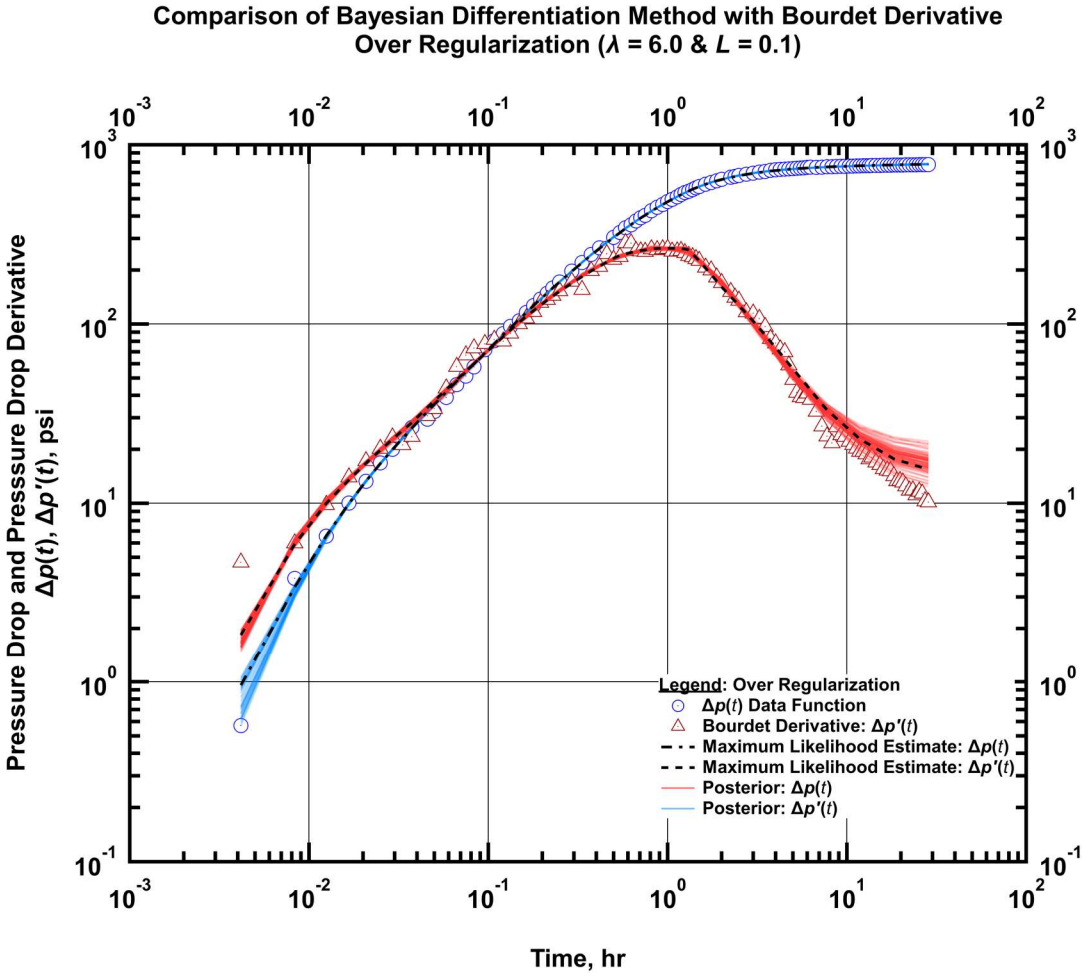

Figure 4.5 — Comparison of Bayesian derivative method with Bourdet et al. (1998) case using too much regularization, λ = 6.0, vs. L = 0.1.

Lastly, **Fig. 4.6** shows an application to a field case (ETX Gas Well). For the Bourdet method, we have performed a considerable amount of filtering to achieve a reasonable

response. All data points that lie greater than 1 standard deviation from the MLE estimate has been excluded from the calculation. Although the comparison does present a somewhat circular conclusion, the noise is otherwise unmanageable without some arbitrary method of filtering the data.

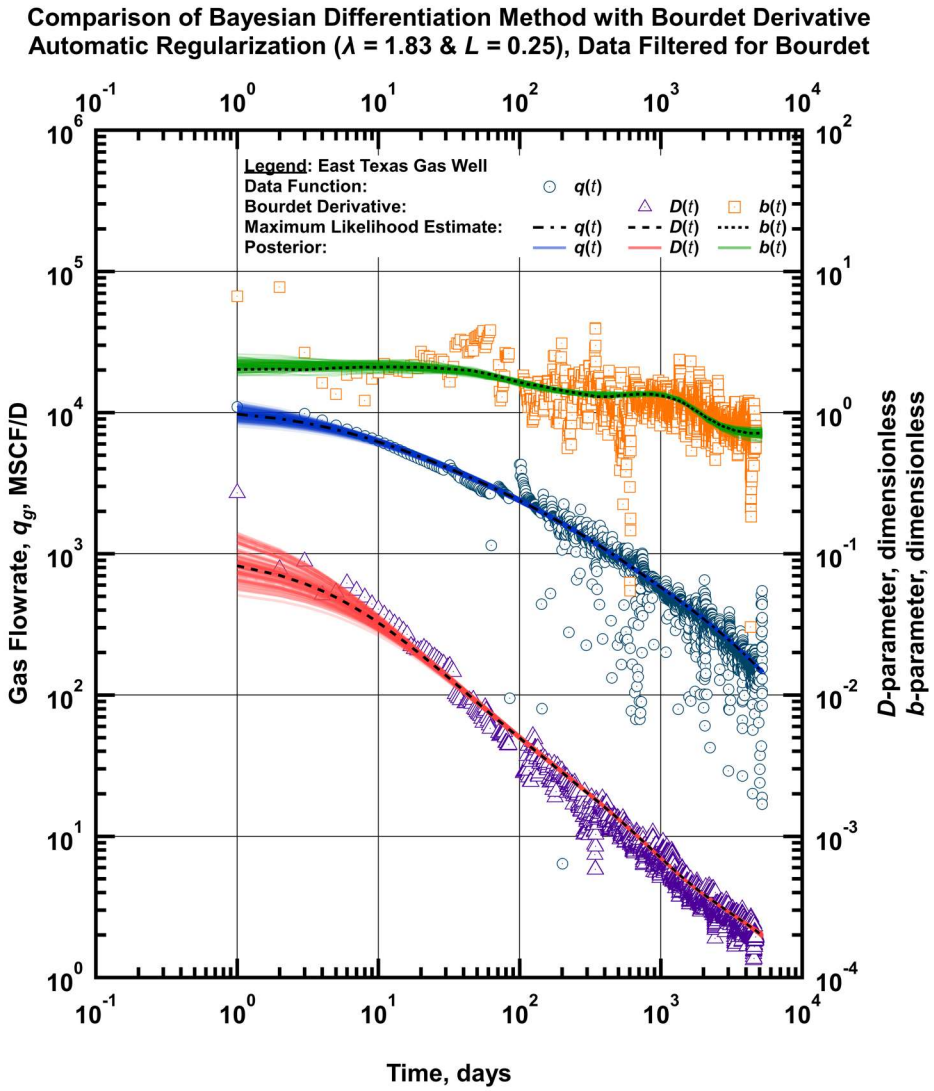

Figure 4.6 — Comparison of Bayesian derivative method with Bourdet et al. (1998) for ETX Gas Well using automatic regularization, λ = 1.83, vs. $L = 0.1$.

5. USE OF PARAMETRIC KERNEL FUNCTIONS FOR DECONVOLUTION

5.1 Introduction

This chapter presents an implementation of deconvolution using parametric kernel functions. We show that any function may be used to represent the unknown constant-pressure rate function, given it conforms to the requirements as laid out in the primary development of the non-parametric kernel functions. That is, it must be monotonic and decays to zero.

The primary goal is to provide a robust method for evaluation of time-rate-pressure data in cases where 1) geologic, reservoir, and petroyphysical properties are unavailable, and 2) data measurement errors may be large, possibly systemically biased, and contain considerable noise. We build upon the work of Ilk and Blasingame (2013) and Collins, Ilk, and Blasingame (2014) and mirror their motivation to apply rigorous convolution/superposition theory to empirical rate decline relations to typical time-rate-pressure field data sets. Our addition to their prior efforts is the application of our Bayesian method to solve the inverse problem of determination of rate decline function parameters.

5.2 Development of the Method

The method shares many similarities with the primary method of this thesis that utilizes Gaussian processes (GP) as the non-parametric function. The probabilistic graphical

model is shown in **Fig. 5.1**. Rather than a prior over functions as in the case of the GP, we place a prior of the parameters of the empirical rate decline function that is chosen to represent $f(\theta_i,t)$. Once the constant-pressure-drop rate function is generated, the convolution with the derivative of pressure-drop occurs as before.

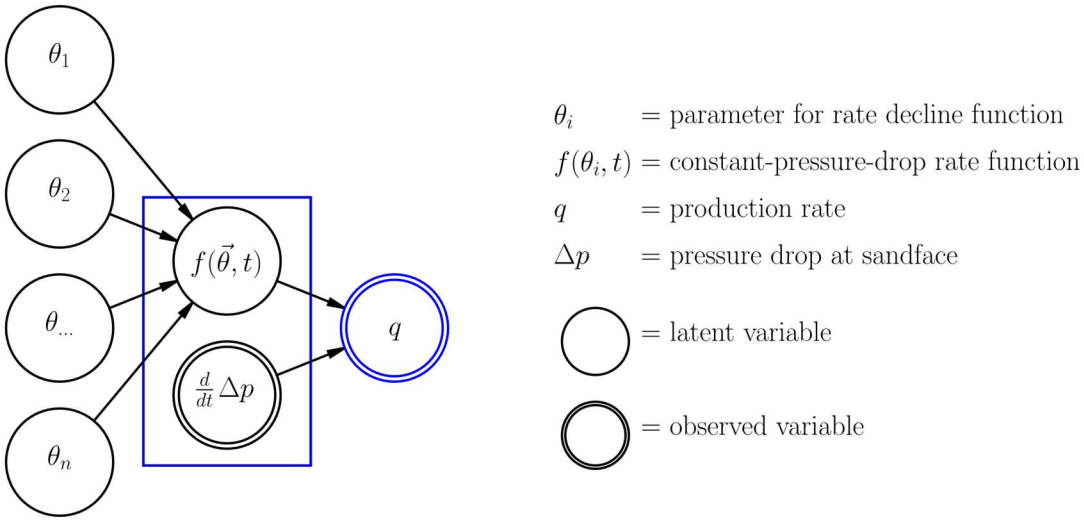

Figure 5.1 — Probabilistic graphical model for deconvolution of time-rate-pressure data using parametric kernel function.

Aside from the choice of kernel function, there are no other differences. We note that this stems from the generality of our approach, and Bayesian methods in general. Once the physical relationship of all variables is laid out, the manner in which each variable is generated is independent of all other parameters. The influence of variables flows along the edges of the graph to other variables during optimization, in which all variables are

evaluated, and each iteration of the model accepted (or not) from the evaluation of all parameters simultaneously.

Once setup, the switch from one rate decline function to another only involves 1) a function to generate the rate values given the input parameters and a value of time at which to evaluate, and 2) an assignment of a random variable to each parameter. While the popularity of the Arps (Arps 1945) relations persist, numerous time-rate relations exist for the evaluation of unconventional resources. These developments evaluated in this work include:

- Modified Hyperbolic Model (Robertson 1988)

$$q(t)=\begin{cases}\dfrac{q_i}{\left(1+D_i b t\right)^{1/b}} & t<t_{exp}\\ q_{exp}\exp\left[-D_{exp}\left(t-t_{exp}\right)\right] & t>=t_{exp}\end{cases} \quad \text{5.1}$$

where:

$$t_{exp}=\frac{1/D_{exp}-1/D_i}{b} \quad \text{5.2}$$

$$q_{exp}=\frac{q_i}{\left(1+D_i b t_{exp}\right)^{1/b}} \quad \text{5.3}$$

- Stretched Exponential Model (Valkó 2009)

$$q(t)=\hat{q}_i\exp\left[-\left(\frac{t}{\tau}\right)^n\right] \quad \text{5.4}$$

- Power-Law Exponential Model (Ilk et al. 2008, 2009)

$$q(t)=\hat{q}_i\exp\left[-\hat{D}_i t^n-D_\infty t\right] \quad \text{5.5}$$

● Duong Model (Duong 2011)

$$q(t) = q_1 t^{-m} \exp\left[\frac{a}{1-m}\left[t^{(1-m)} - 1\right]\right] \quad \text{.....5.6}$$

● Transient Hyperbolic Model (Fulford and Blasingame 2013)

$$q(t) = q \exp\left[-\int_0^t \frac{1}{\int_0^t b\, dt} dt\right] \quad \text{.....5.7}$$

where:

$$b(t) = b_i + (b_i - b_f) \exp\left[-\exp\left[-c\left(t - t_{elf}\right) + e^{\gamma}\right]\right] \quad \text{.....5.8}$$

$$c = \frac{e^{\gamma}}{1.5 t_{elf}} \quad \text{.....5.9}$$

Alternatively, Fulford (2018) gave the discretized approximation for a series of piecewise hyperbolic or exponential segments:

$$q_n = \frac{q_{n-1}}{\left[1 + D_{n-1} b_{n-1}\left(t - t_{n-1}\right)\right]^{1/b_{n-1}}} \quad \text{.....5.10}$$

$$D_n = \frac{1}{1/D_{n-1} + b_{n-1}\left(t - t_{n-1}\right)} \quad \text{.....5.11}$$

where:

Table 5.1 — Transient Hyperbolic discretization parameters

Segment	*b*-parameter	Start Time
1	b_i	0
2	$b_i - \frac{b_i - b_f}{b_i}$	$t_{elf}(e-1)$
3	b_f	$t_{elf}(e+1)$

And the appropriate Arps hyperbolic or exponential function is applied for each segment. An implementation of an arbitrary *n* segment hyperbolic model is created for both the Modified Hyperbolic Model and Transient Hyperbolic Model.

Additionally, we define the loss ratio and derivative of the loss ratio as:

$$D(t) \equiv -\frac{1}{q(t)}\frac{d}{dt}\left[q(t)\right] \quad \text{.....5.12}$$

$$b(t) \equiv \frac{d}{dt}\left[\frac{1}{D(t)}\right] \quad \text{.....5.13}$$

From which the hyperbolic relations are derived.

5.3 Implementation Details

With a known model, we have significantly reduced the degrees of freedom of the problem; from several tens of parameters, to somewhere between three to five for the given decline curve models. Therefore, we can simply represent each parameter as a uniform distribution with the respective bounds for the model, listed in **Table 5.2**.

Alternatively, we can specify other distributions of any type that might best represent our knowledge of each parameter. For example, if we feel strongly that the correct *b*-parameter is 1.0, we can use a Normal distribution with a small standard deviation to shift the posterior distribution of models to this belief. Similar logic may be applied to any parameter.

5.4 Validation

5.4.1 Model Hyperparameters

We present the results of the five models based upon the same variable pressure-drop as the unknown kernel function case described in **Fig. 3.12** and **Fig. 3.13**. For consistency and comparability, we present the reciprocal of the dimensionless rate function. **Fig. 5.2 – Fig. 5.6** present the results for each model with no noise. **Fig. 5.7 – Fig. 5.11** present the results for 5% random uniform noise. **Fig. 5.12 – Fig. 5.16** present the results for 20% random uniform noise.

Table 5.2 — Uniform distribution bounds for decline curve models

Modified Hyperbolic Model

Parameter	Lower Bound	Upper Bound
q_i	1e-2	1e6
D_i, *secant effective*	0.1	0.99
b	0.0	2.0
D_{exp}	0.0	0.2

Transient Hyperbolic Model

Parameter	Lower Bound	Upper Bound
q_i	1e-2	1e6
D_i, *secant effect*	0.1	0.99
b_i	*constant* @ 2.0	
b_f	0.2	2.0
t_{elf}	0.1	100.0

Stretched Exponential Model

Parameter	Lower Bound	Upper Bound
q_i	1e-2	1e6
τ	0.01	10.0
n	0.01	0.99

Power-Law Exponential Model

Parameter	Lower Bound	Upper Bound
q_i	1e-2	1e6
D_i	0.1	.099
n	0.01	0.99
D_∞	0.001	0.1

Duong Model

Parameter	Lower Bound	Upper Bound
q_i	1e-2	1e6
a	0.01	10.0
m	0.01	0.99

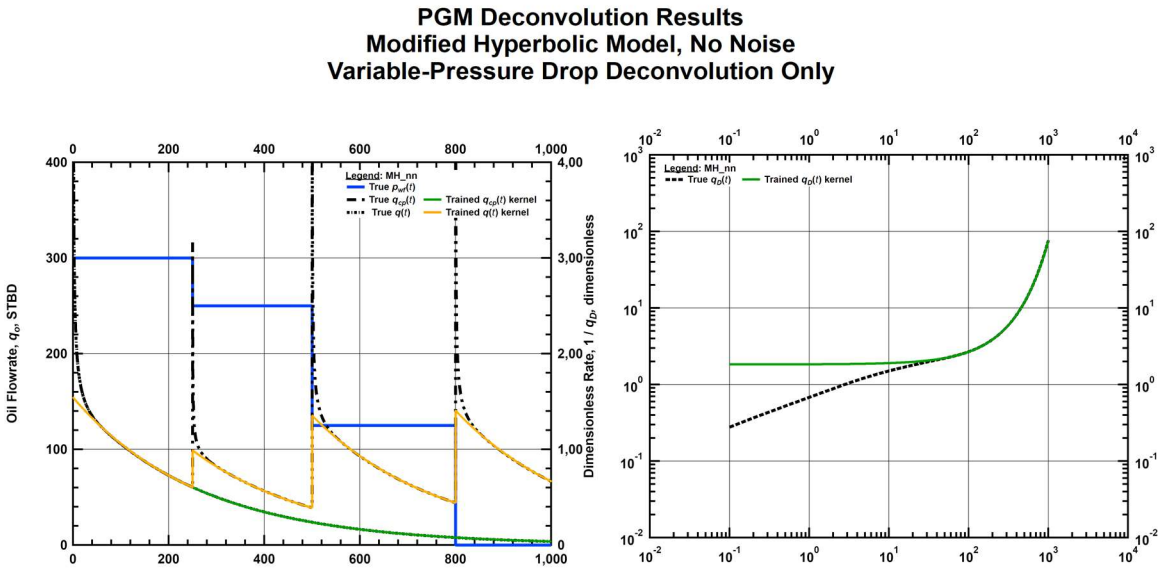

Figure 5.2 — Deconvolution of variable pressure-drop case utilizing the Modified Hyperbolic Model (no random error).

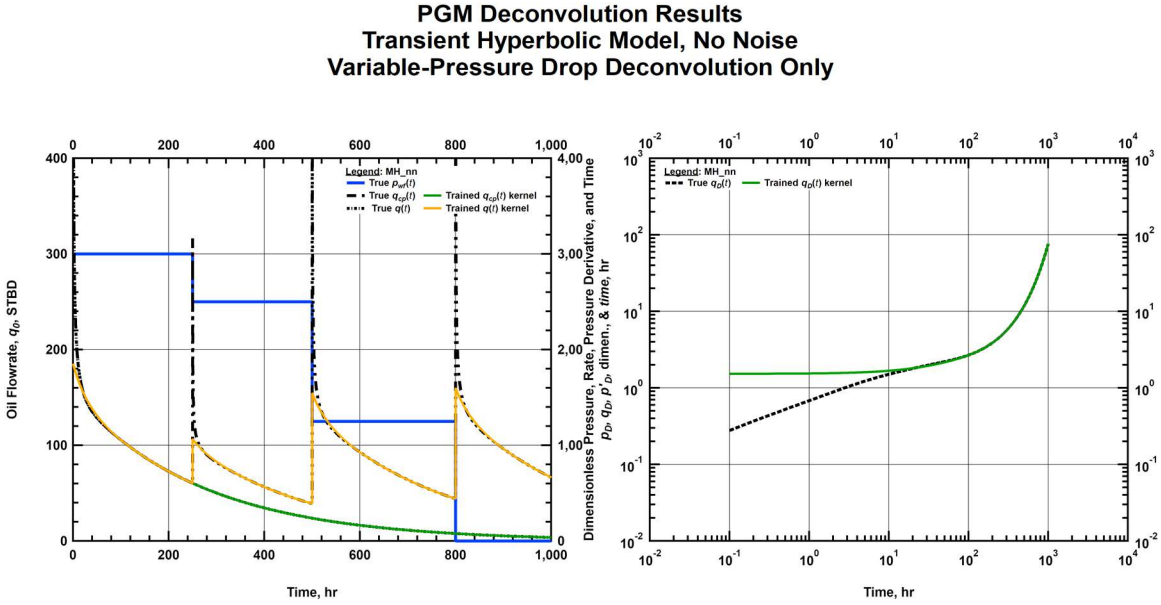

Figure 5.3 — Deconvolution of variable pressure-drop case utilizing the Transient Hyperbolic Model (no random error).

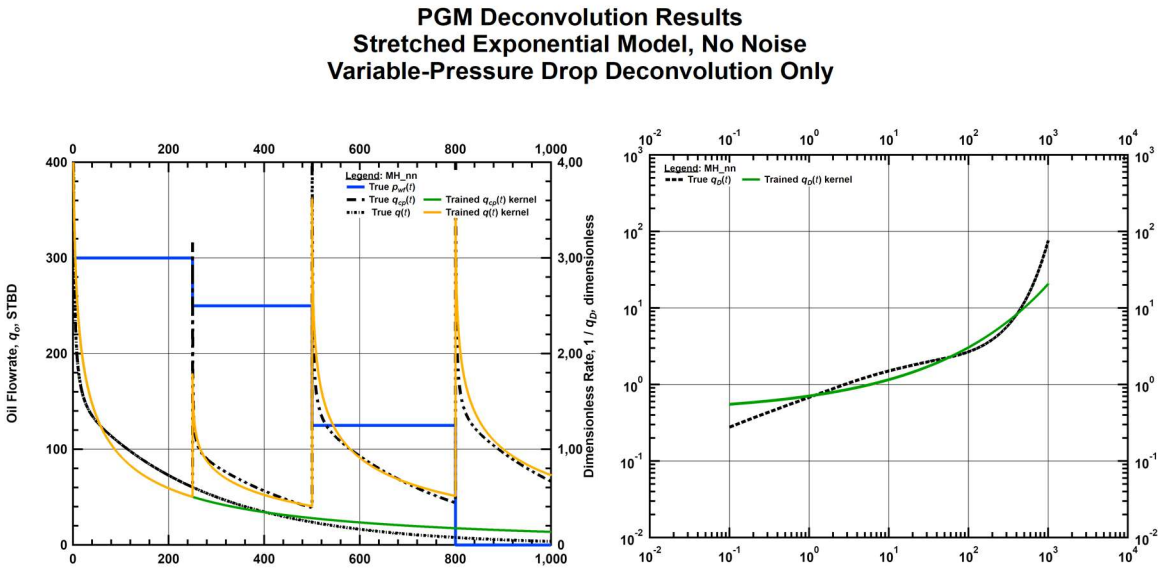

Figure 5.4 — Deconvolution of variable pressure-drop case utilizing the Stretched Model (no random error).

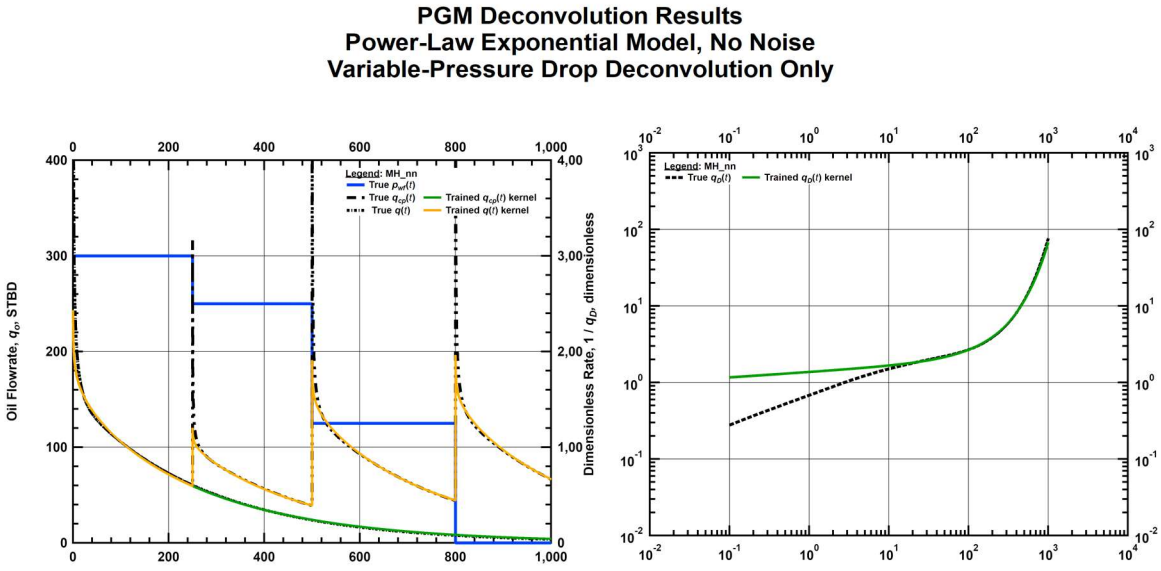

Figure 5.5 — Deconvolution of variable pressure-drop case utilizing the Power-Law Exponential Model (no random error).

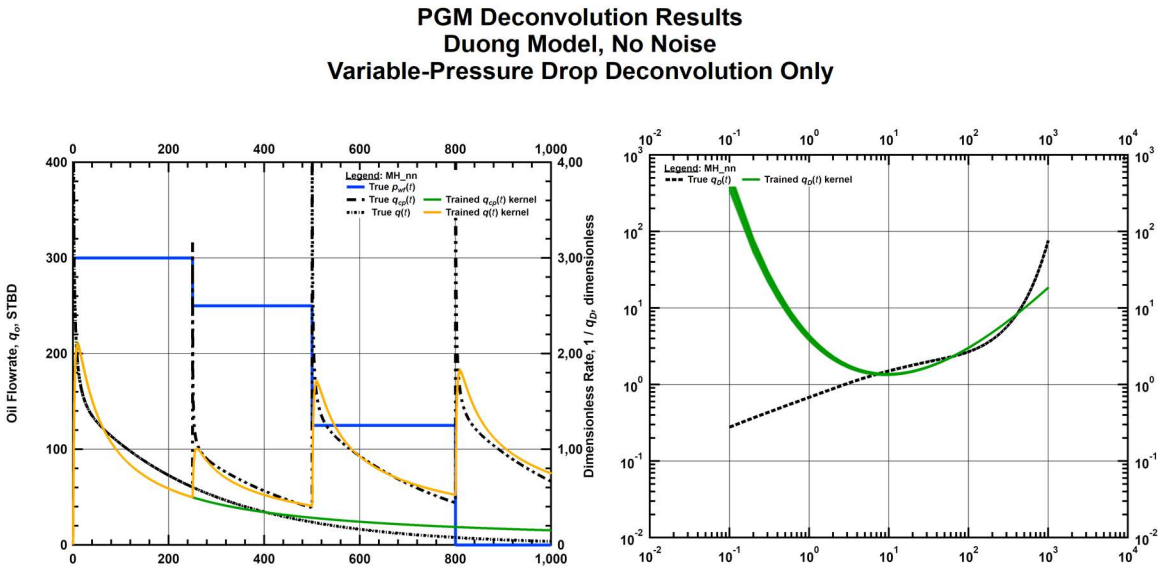

Figure 5.6 — Deconvolution of variable pressure-drop case utilizing the Duong Model (no random error).

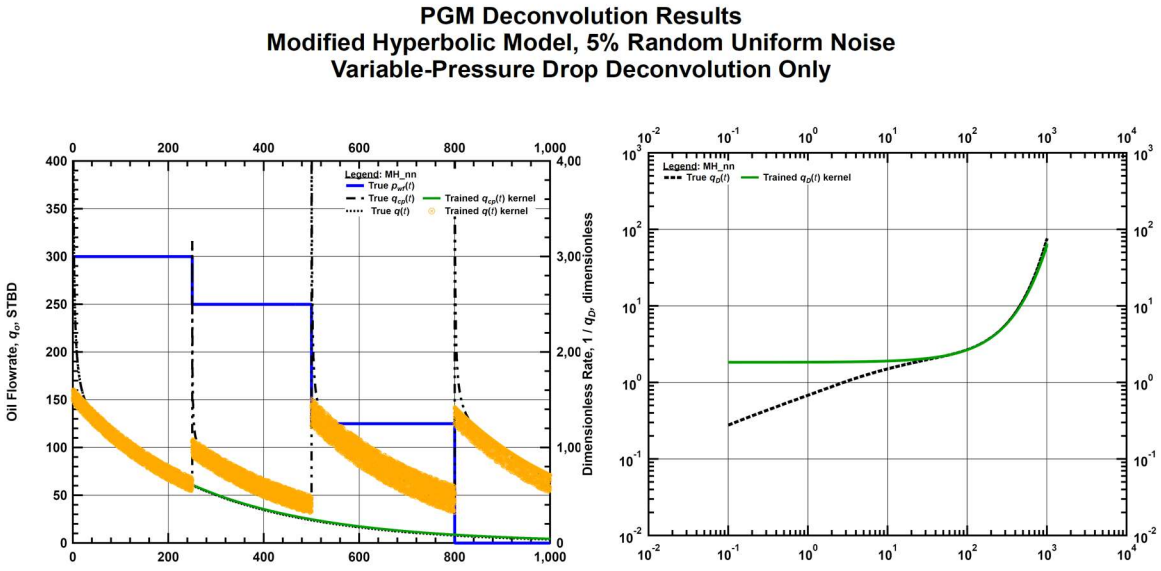

Figure 5.7 — Deconvolution of variable pressure-drop case utilizing the Modified Hyperbolic Model (5% random error).

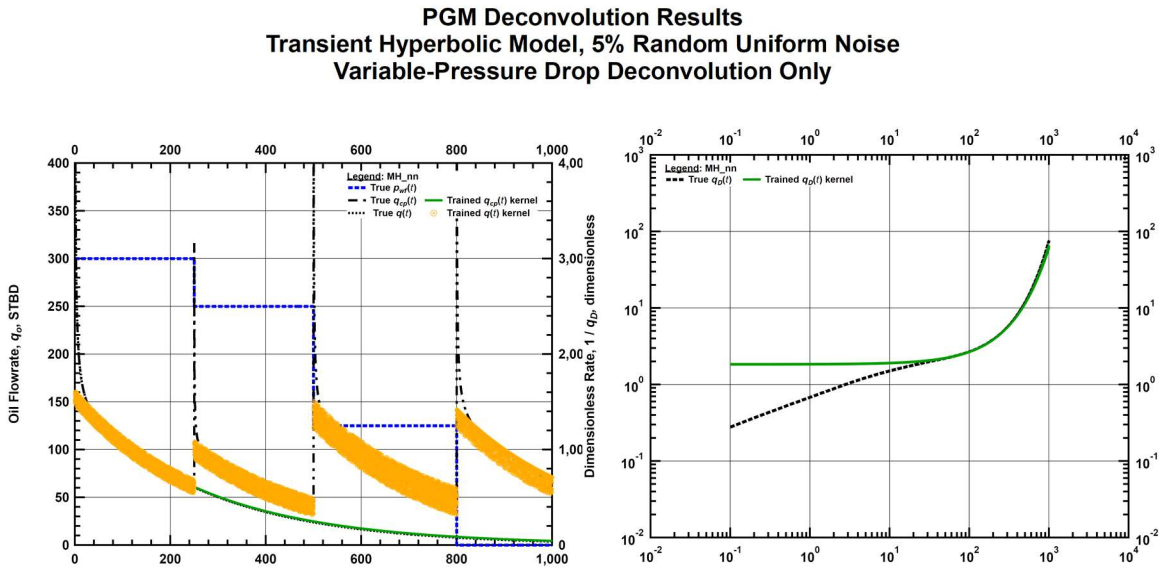

Figure 5.8 — Deconvolution of variable pressure-drop case utilizing the Transient Hyperbolic Model (5% random error).

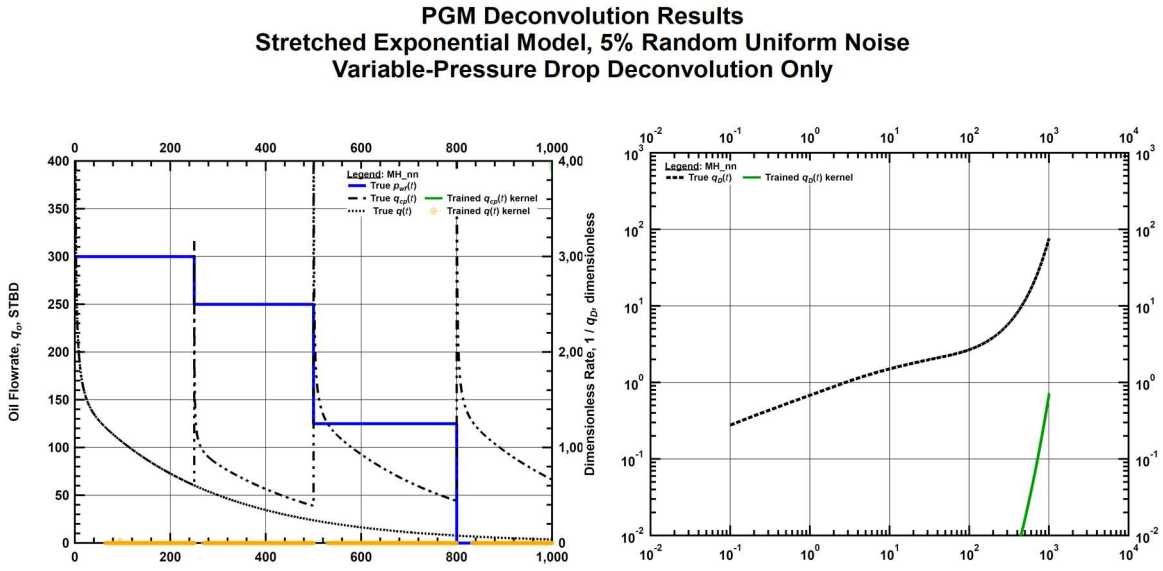

Figure 5.9 — Deconvolution of variable pressure-drop case utilizing the Stretched Model (5% random error).

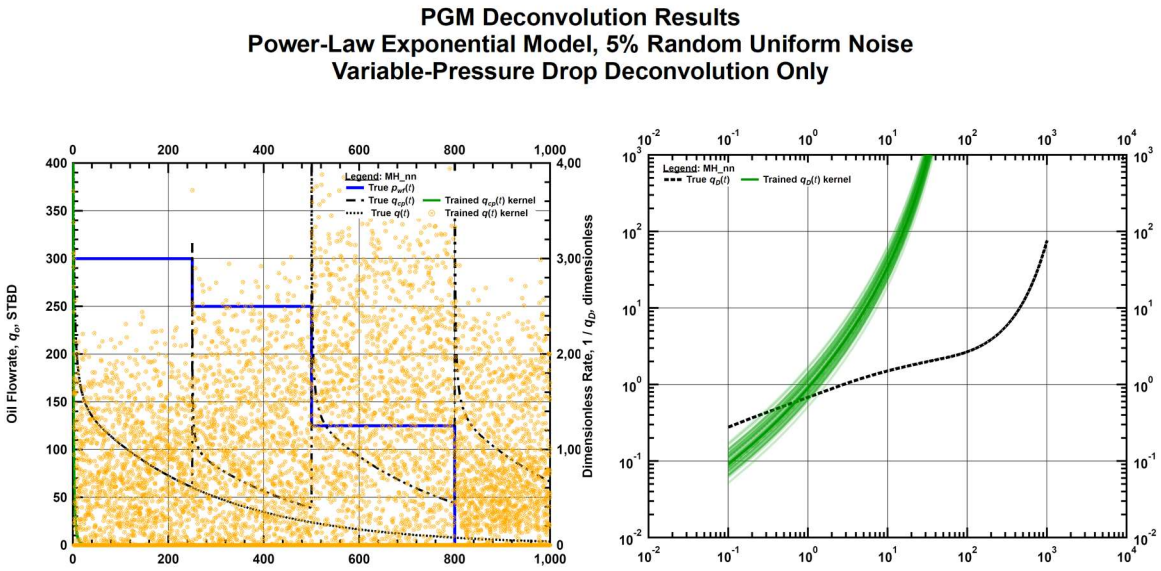

Figure 5.10 — Deconvolution of variable pressure-drop case utilizing the Power-Law Exponential Model (5% random error).

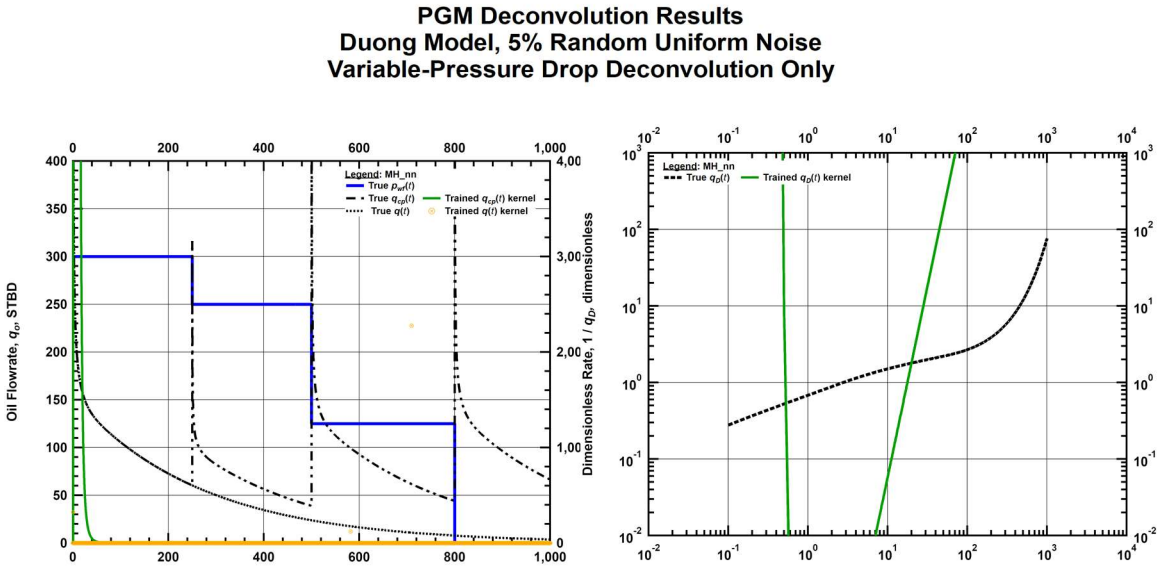

Figure 5.11 — Deconvolution of variable pressure-drop case utilizing the Duong Model (5% random error).

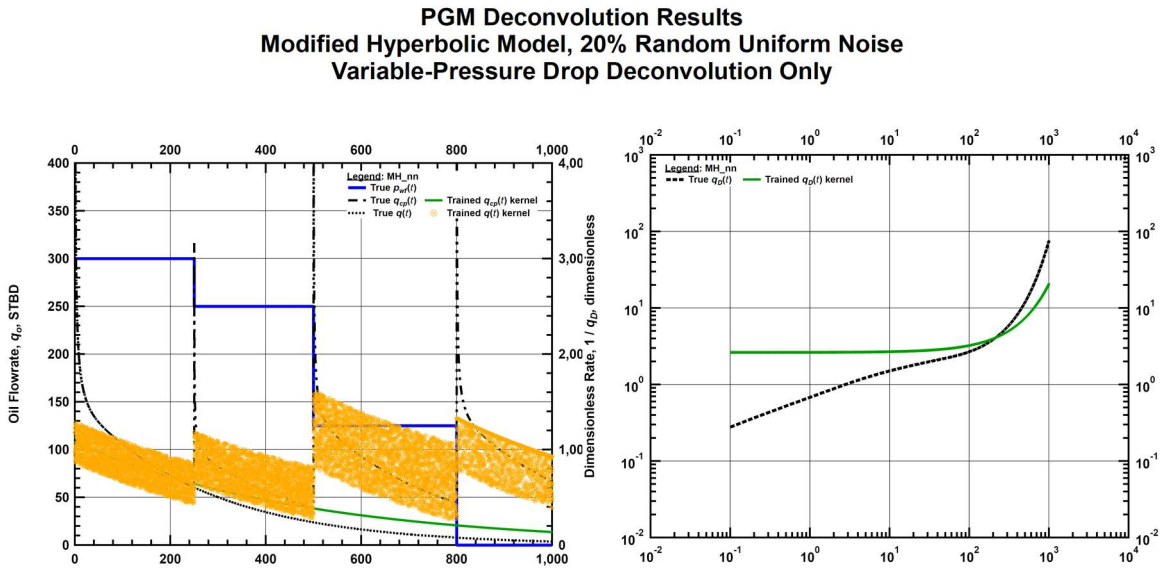

Figure 5.12 — Deconvolution of variable pressure-drop case utilizing the Modified Hyperbolic Model (5% random error).

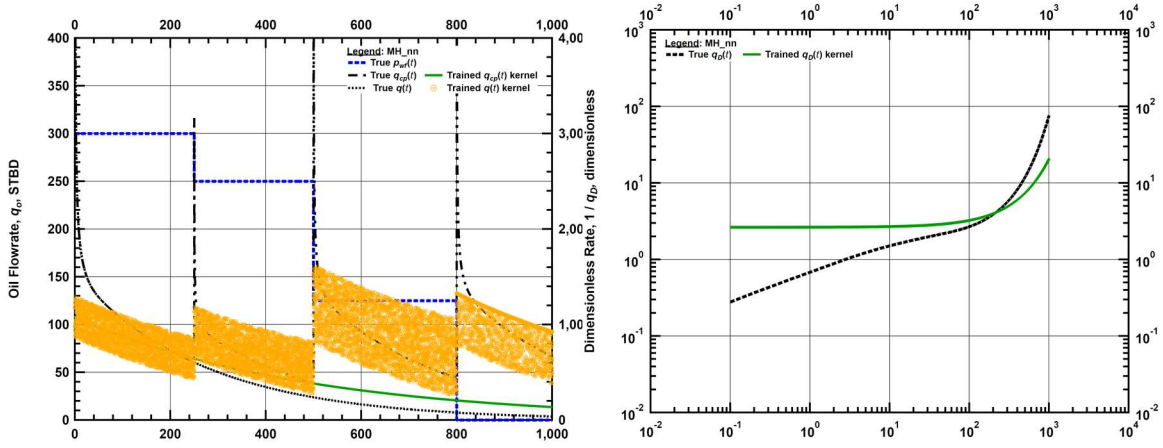

Figure 5.13 — Deconvolution of variable pressure-drop case utilizing the Transient Hyperbolic Model (5% random error).

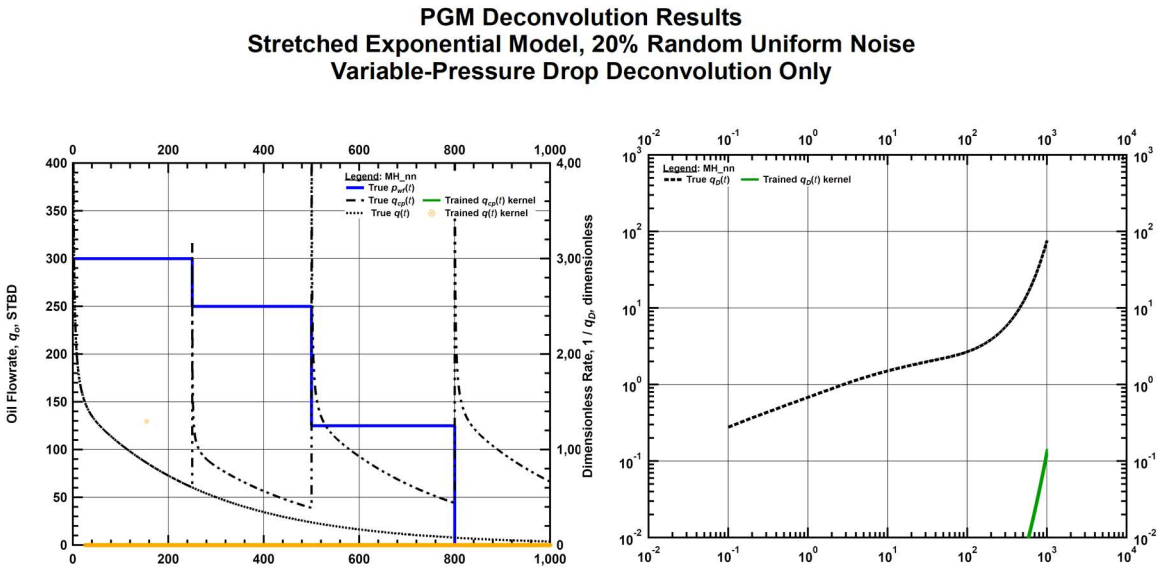

Figure 5.14 — Deconvolution of variable pressure-drop case utilizing the Stretched Model (5% random error).

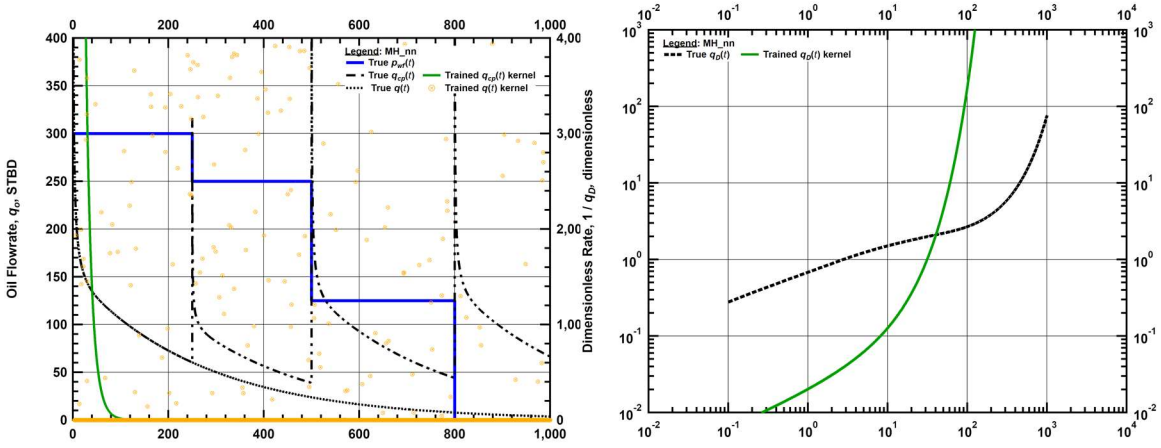

Figure 5.15 — Deconvolution of variable pressure-drop case utilizing the Power-Law Exponential Model (5% random error).

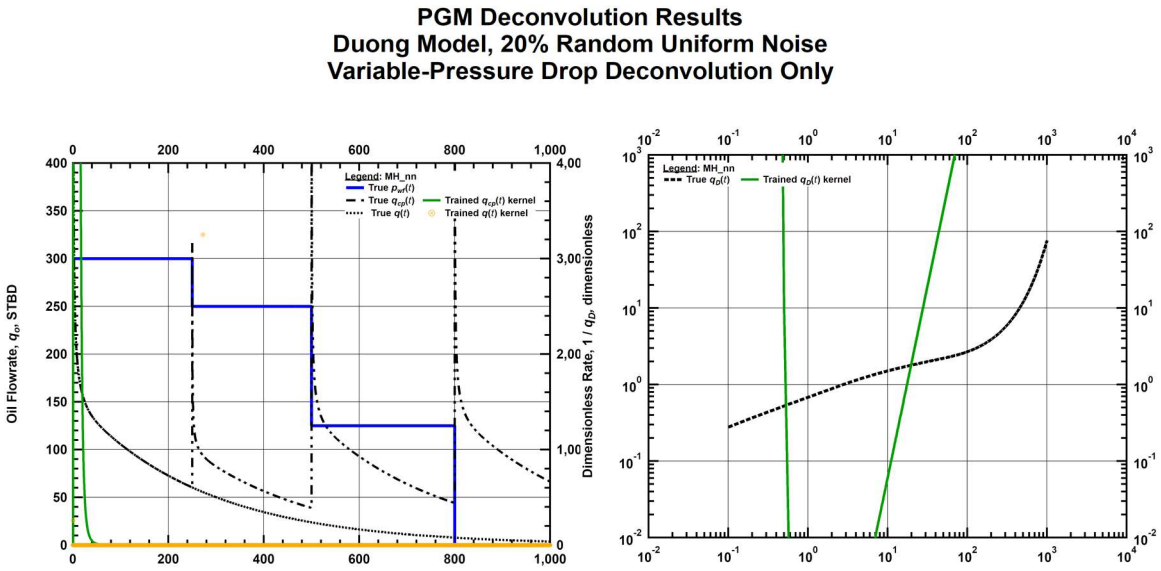

Figure 5.16 — Deconvolution of variable pressure-drop case utilizing the Duong Model (5% random error).

5.4.2 Discussion of Results

We note that we have successfully deconvolved the given case with each decline curve model. While each model performs differently, the differences are due to the inherent behavior of each model and not any limitation of the implementation. To generalize, we observe that the models with the largest number of parameters provide a better fit to the know dimensionless rate function, while those with fewer parameters are (naturally) less able to achieve the known solution.

As an interesting point-of-fact, we observe that the results of the models in the cases of 5% and 20% random noise more clearly explain the results of the non-parametric method in the presence of noise. For the chosen profile of flowing pressure for these cases, the algorithm cannot distinguish between the true large values of rate, and the large values as

a result of the added noise. Instead, the deconvolved rate profile for both the GP and B-spline based nonparametric method and the decline curve based parametric method have a lower value of initial (early-time) rate, and a lower rate of decline.

We also observe that the three models that are able to functionally reduce to exponential decline at late time—the Modified Hyperbolic, Transient Hyperbolic, and Power-Law Exponential—are able to nearly exactly reproduce the late-time behavior. These three are can be used to represent distinct flow regimes for the analyzed case, while the other two—the Stretched Exponential and Duong, which each have three parameters best described as an intercept, initial slope, and curvature—attempt to represent the rate decline behavior as a single regime that "smooths" or "averages" the behavior of the true dimensionless rate function.

The cases with 5% random noise added reveal that the hyperbolic models tend toward greater stability than the other models. The biggest difference is with the Power-Law Exponential model, which provides an excellent answer in the case with no noise but diverges with only 5% noise. Both the Power-Law Exponential and Stretched Exponential models have an initial rate parameter which may range over a few orders of magnitude, this wide range likely leads to difficulty in finding the optimal range of solution.

We conclude that while a model with more parameters and therefore able to match more distinct flow regime behavior should be preferred for our deconvolution method, stability of the model's range of parameter values is also an important aspect to be considered to handle noise.

5.5 Application to Field Case

We apply the Transient Hyperbolic Model to the East Texas Gas Well. As before, we stress that we do not filter any data and rely upon our cost function to handle noise and outlier data. It is obvious in **Fig. 5.17** that the decline curve cannot exactly reproduce the observed data, especially the early-time transient portion. Like the nonparametric method, we observe a large degree of uncertainty for the rate functions.

This case serves as a confirmation that our deconvolution method should be compatible with any type of rate function, not just our primary method that employs B-splines. We have observed varying levels of stability for different parametric functions. The success of the deconvolution then depends on the stability of the chosen rate function more so than the inference/optimization process.

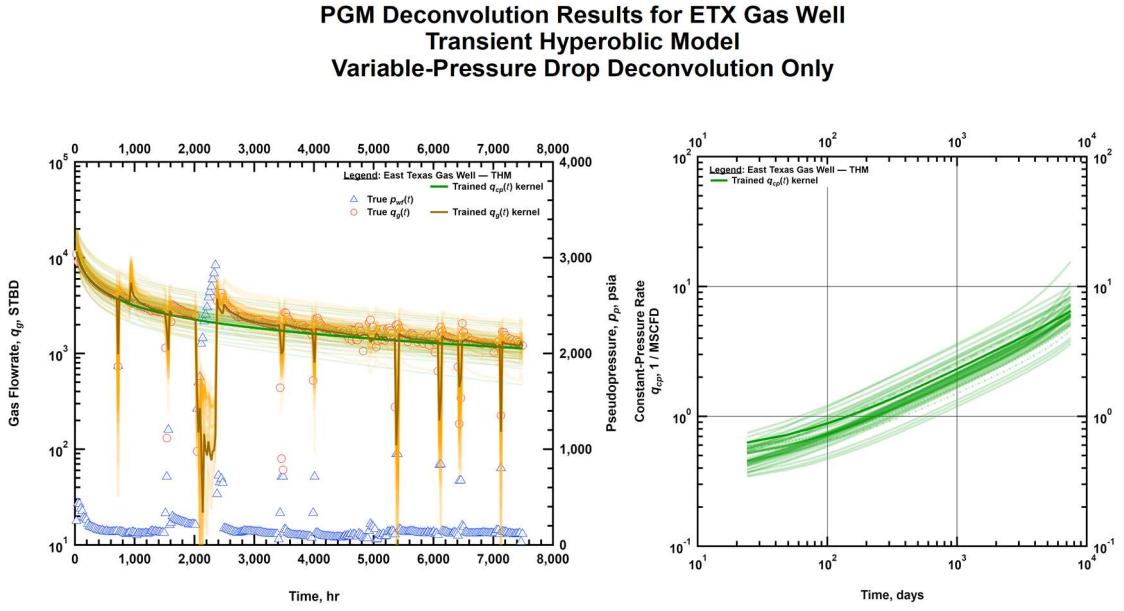

Figure 5.17 — Deconvolution of East Texas Gas Well (Transient Hyperbolic Model).

6. SUMMARY AND CONCLUSIONS

6.1 Summary

We have provided a novel method of deconvolution that is unique within the existing petroleum literature. The new method is based upon a probabilistic graphical model which describes the functional relationships of all model parameters/functions. Given observation of any variable in the graph, an inference is made for every other variable in the graph. The observations are the sandface rates and pressures—partial or complete histories—and the inference is usually—but not limited to—the unknown constant-rate pressure function and constant-pressure-drop rate function. We validate the method by analysis of synthetic data with and without error. We demonstrate the utility of the model to infer other parameters in the graph, such as missing rates and/or pressures, and unknown initial reservoir pressure.

6.2 Conclusions

- We have developed a probabilistic graphical model sampled by a nonlinear stochastic optimization algorithm with a non-parametric function as a generalization of the time-based deconvolution problem. The model is used to perform simultaneous deconvolution to achieve constant-rate pressure functions and constant-pressure-drop rate functions correlated by Duhamel's principle.
- We have eliminated the need for tuning of the regularization scheme as an adjustment solely for the presence of noise, instead addressing noise by utilization of robust cost

functions other than squared error. We have implemented a rigorous regularization scheme based upon curvature of the beta-derivative.

- We have validated our technique against the exact dimensionless rate and dimensionless pressure functions, their derivatives, and by comparison of the reconstructed-by-convolution variable-rate pressure function and reconstructed-by-convolution variable-pressure drop rate function. Additional validations of the method are shown by deconvolving the correct response in the presence of random errors, partial missing histories, and unknown initial reservoir pressure.
- We have shown that our method is not limited to our implementation of the beta-derivative as the base parameterization by implementation of parametric functions, i.e. decline curves, as the generative functions for the constant-pressure rate function. We have also shown an alterative parameterization of a non-parametric function using the well-testing derivative.
- We have developed a new method for Bayesian differentiation for analysis of production data. Our new method applies to pressure data, such as to compute a well-testing derivative, as well as production data to compute the D-parameter and b-parameter. We have shown that our new method illustrates non-uniqueness of the inverse problem of differentiation and provides smoother responses than current methods.

6.3 Recommendations for Future Work

1. Investigate alternative methods for generating the unknown kernel functions, including simultaneously generation such as from correlated Gaussian processes. As the signature of the rate and pressure functions is known for any single flow regime, it may be possible to generate correlated steps of each function.
2. Investigate functional relationships of known flow-regimes to correlate the kernel functions instead of utilizing Duhamel's principle, or investigate enforcing correlation in the Laplace domain. This may improve recovery of the constant-pressure rate kernel function in the variable rate case, and vice versa, for example by better capturing the exponential behavior of the constant-pressure rate function during pseudosteady-state constant-rate decline.
3. Investigate back-extrapolation of the rate and pressure functions to reduce the influence of numerical start-up effects of the correlation.
4. Evaluate methods for generation of sandface rate functions to deconvolve wellbore storage effects.
5. Evaluate alternative parameterizations of data imputation, e.g. by regularization of imputed values, to reduce unrealistically large variance.
6. Investigate inclusion of effects that break the rate-pressure linearity assumption, such as pseudopressure and pseudotime, as part of the calculation chain within the computational graph.

NOMENCLATURE

Variables:

B_j^k	=	The j-th B-spline of degree k
$f(t)$	=	dummy function
K	=	Curvature of function
$p(t)$	=	Pressure, psi
$p_{cr}(t)$	=	Constant-rate pressure solution, psi
$p_D(t)$	=	Dimensionless pressure, dimen.
p_i	=	Initial reservoir pressure, psi
p_r	=	Reference pressure, psi
p_{wf}	=	Flowing bottomhole pressure, psi
p_{wD}	=	Dimensionless flowing bottomhole pressure, dimen.
$\Delta p(t)$	=	Pressure drop, psi
$\Delta p_{cr}(t)$	=	Constant-rate pressure drop, psi
$q(t)$	=	Rate, STB/D or MSCF/D
$q_{cp}(t)$	=	Constant-pressure rate, STB/D or MSCF/D
$q_D(t)$	=	Dimensionless rate, dimen.
$q_r(t)$	=	Reference rate, STB/D or MSCF/D
n	=	Total number of steps for discrete step-wise functions
s	=	Laplace transform variable
t	=	Time, hr or day

Greek Symbols:

$\beta(t)$ = beta-derivative

$\beta_p(t)$ = beta-derivative of constant-rate pressure drop

$\beta_q(t)$ = beta-derivative of constant-pressure-drop rate

κ = Kernel function

λ = Regularization scalar, proportional to reciprocal of scale

μ = Location of probability distribution

σ = Scale of probability distribution

Σ = Covariance matrix of probability distribution

τ = Dummy variable or natural log of time

θ = random variable representing a model parameter

Bold symbols:

$\mathbf{B}^k$ = B-spline or degree k

$\mathbf{c}$ = B-spline coefficients

$\mathbf{e}$ = Integral of B-spline coefficients

$\mathbf{S}$ = Spline function, i.e. evaluation of the B-spline and coefficients

$\boldsymbol{\beta}_\mathbf{p}$ = beta-derivative of constant-rate pressure drop, vector form

Script:

$\mathcal{HN}$ = HalfNormal distribution

$\mathcal{N}$ = Normal distribution

$\mathcal{L}$ = Laplace distribution

$\mathcal{U}$ = Uniform distribution

ℓ = Length scale for kernel function

Subscript:

i = Index for discrete sum or product functions

iv = Initial value of model parameter

Superscript:

$\bar{}$ = Laplace transform

REFERENCES

1. M. Abramowitz and I. Stegun. 1972. *Handbook of Mathematical Functions*. New York: Dover Publications, Inc.

2. Ahmadi, M., Sartipizadeh, H., and Ozkan, E. 2017. A new pressure-rate deconvolution algorithm based on Laplace transformation and its application to measured well responses. *J Pet Sci and Eng* **157** (1): 68–80. https://doi.org/10.1016/j.petrol.2017.06.060.

3. Blasingame, T. A. 2009. Petroleum Engineering 324 Course Notes.

4. Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., and Bengio, Y. 2010. Theano: A CPU and GPU Math Compiler in Python. Proceedings of the 9th Python in Science Conference (SciPy 2010) in Austin, Texas, USA, 30 June – 3 July. https://arxiv.org/abs/1605.02688.

5. Bourdet, D., Ayoub, J. A., and Pirard, Y. M. 1989. Use of Pressure Derivative in Well-Test Interpretation. *SPE Form Eval* **4** (2): 293–302. SPE-12777-PA. https://doi.org/10.2118/12777-PA.

6. Cheng, Y., Lee, W. J., and McVay, D. A. 2003. A Deconvolution Technique Using Fast-Fourier Transforms. Presented at SPE Annual Technical Conference and Exhibition in Denver, Colorado, USA, 5–8 October. SPE-84471-MS. https://doi.org/10.2118/84471-MS.

7. Cheng, Y., Lee, W. J., and McVay, D. A. 2005. Fast-Fourier-Transform-Based Deconvolution for Interpretation of Pressure-Transient-Test Data Dominated by Wellbore Storage. *SPE Res Eval & Eng* **8** (3): 224–239. SPE-84471-PA. https://doi.org/10.2118/84471-PA.

8. Cheney, E. W. and Kincaid, D. R. 2003. *Numerical Mathematics and Computing*. Belmont: Thompson Brooks/Cole.

9. Çinar, M., Ilk, D., Onur, M., Valkó, P. P., and Blasingame, T. A. 2006. A Comparative Study of Recent Robust Deconvolution Algorithms for Well-Test and Production-Data Analysis. Presented at SPE Annual Technical Conference and Exhibition in San Antonio, Texas, USA, 24–27 September. SPE-102575-MS. https://doi.org/10.2118/102575-MS.

10. Collins, P. W., Ilk, D., and Blasingame, T. A. 2014. Practical Considerations for Forecasting Production Data in Unconventional Reservoirs — Variable Pressure Drop Case. Presented at SPE Annual Technical Conference and Exhibition in Amsterdam, The Netherlands, 27-29 October. SPE-170945-MS. https://doi.org/10.2118/170945-MS.

11. Cumming, J. A., Wooff, D. A., Whittle, T., and Gringarten, A. C. 2013. Multiple Well Deconvolution. Presented at SPE Annual Technical Conference and Exhibition in New Orleans, Louisiana, USA, 30 September – 2 October. SPE-166458-MS. https://doi.org/10.2118/166458-MS.

12. Cumming, J. A., Wooff, D. A. Whittle, T., and Gringarten, A. C. 2013. *SPE Res Eval & Eng* **17** (4): 457–465. SPE-166458-PA. https://doi.org/10.2118/166458-PA.

13. Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. 1987. Hybrid Monte Carlo. *Physics Letters B* **195** (2): 216–222. http://doi.org/10.1016/0370-2693(87)91197-X.

14. Duong, A. N. 2001. Rate-Decline Analysis for Fracture-Dominated Shale Reservoirs. *SPE Res Eval & Eng* **14** (3): 377–387. SPE-137748-PA. https://doi.org/10.2118/137748-PA.

15. Fulford, D. S., and Blasingame, T. A. 2013. Evaluation of Time-Rate Performance of Shale Wells using the Transient Hyperbolic Time-Rate Relation. Presented at SPE Unconventional Resources Conference – Canada in Calgary, Alberta, Canda, 5–7 November. SPE-167242-MS. https://doi.org/10.2118/167242-MS.

16. Fulford, D.S. 2018. A Model-Based Diagnostic Workflow for Time-Rate Performance of Unconventional Wells. Presented at Unconventional Resources Conference in Houston, Texas, USA, 23–25 July. URTeC-2903036. https://doi.org/10.15530/urtec-2018-2903036.

17. Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. 2014. *Bayesian Data Analysis*. Boca Raton: Chapman and Hall/CRC.

18. Gringarten, A. C., Ramey Jr., H. J., and Raghavan, R. 1974. Unsteady-State Pressure Distributions Created by a Well With a Single Infinite-Conductivity Vertical Fracture. *SPE J* **14** (4): 347–360. SPE-4051-PA. https://doi.org/10.2118/4051-PA.

19. Hoffman, M. D., and Gelman, A. 2014. The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research* **15** (1): 1593–1693.

20. Hosseinpour-Zonoozi, N., Ilk, D., and Blasingame, T. A. 2006. The Pressure Derivative Revisited—Improved Formulations and Applications. Presented at SPE Annual Technical Conference and Exhibition in San Antonio, Texas, USA, 24–27 September. SPE-103204-MS. https://doi.org/10.2118/103204-MS.

21. Huber, P. J. 1964. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics* **35** (1): 73–101. https://doi.org/10.1214/aoms/1177703732.

22. Ilk, D., Valkó, P. P., and Blasingame, T. A. 2005. Deconvolution of Variable-Rate Reservoir Performance Data Using B-Splines. Presented at SPE Annual Technical Conference and Exhibition in Dallas, Texas, USA, 9–12 October. SPE-95571-MS. https://doi.org/10.2118/95571-MS.

23. Ilk, D., Valkó, P. P., and Blasingame, T. A. 2006. Deconvolution of Variable-Rate Reservoir Performance Data Using B-Splines. *SPE Res Eval & Eng* **9** (5): 582–595. SPE-95571-PA. https://doi.org/10.2118/95571-PA.

24. Ilk, D., Valkó, P. P., and Blasingame, T. A. 2007. A Deconvolution Method Based on Cumulative Production for Continuously Measured Flowrate and Pressure Data. Presented at SPE Eastern Regional Meeting in Lexington, Kentucky, USA, 17–19 October. SPE-111269-MS. https://doi.org/10.2118/111269-MS.

25. Ilk, D., Perego, A. D., Rushing, J. A., and Blasingame, T. A. 2008. Exponential vs. Hyperbolic Decline in Tight Gas Sands – Understanding the Origin and Implications for Reserve Estimates Using Arps Decline Curves. Presented at SPE Annual Technical Conference and Exhibition in Denver, Colorado, USA, 21–24 September. SPE-116731-MS. https://doi.org/10.2118/116731-MS.

26. Ilk, D., Rushing, J. A., and Blasingame, T. A. 2009. Decline Curve Analysis for HP/HT Gas Wells: Theory and Applications. Presented at SPE Annual Technical Conference and Exhibition in New Orleands, Louisiana, USA, 4–7 October. SPE-125031-MS. https://doi.org/10.2118/125031-MS.

27. Ilk, D., and Blasingame, T. A. 2013. Decline Curve Analysis for Unconventional Reservoir Systems – Variable Pressure Drop Case. Presented at SPE Unconventional Resources Conference in Calgary, Alberta, Canada, 5–7 November. SPE-167253-MS. https://doi.org/10.2118/167253-MS.

28. Koller, D., and Friedman, N. 2009. *Probabilistic Graphical Models: Principles and Techniques*. Cambridge: The MIT Press. https://doi.org/10.1007/978-1-4471-6699-3.

29. Levitan, M. M. 2003. Practical Application of Pressure-Rate Deconvolution to Analysis of Real Well Tests. Presented at SPE Annual Technical Conference and

Exhibition in Denver, Colorado, USA, 5–8 October. SPE-84290-MS. https://doi.org/10.2118/84290-MS.

30. Levitan, M. M. 2005. Practical Application of Pressure/Rate Deconvolution to Analysis of Real Well Tests. *SPE Res Eval & Eng* **8** (2): 113–121. SPE-84290-PA. https://doi.org/10.2118/84290-PA.

31. Levitan, M. M., Crawford, and G. E., Hardwick, A. 2004. Practical Considerations for Pressure-Rate Deconvolution of Well Test Data. Presented at SPE Annual Technical Conference and Exhibition in Houston, Texas, USA, 26–29 September. SPE-90680-MS. https://doi.org/10.2118/90680-MS.

32. Levitan, M. M., Crawford, and G. E., Hardwick, A. 2006. Practical Considerations for Pressure-Rate Deconvolution of Well-Test Data. *SPE J* **11** (1): 35–47. SPE-90680-PA. https://doi.org/10.2118/90680-PA.

33. Levitan, M. M. 2006. Deconvolution of Multiwell Test Data. Presented at SPE Annual Technical Conference and Exhibition in San Antonio, Texas, USA, 25–27 September. SPE-102484-MS. https://doi.org/10.2118/102484-MS.

34. Levitan, M. M. 2007. Deconvolution of Multiwell Test Data. *SPE J* **12** (4): 420–428. SPE-102484-PA. https://doi.org/10.2118/102484-PA.

35. Liu, Y. and Horne, R. N. 2011. Interpreting Pressure and Flow Rate Data from Permanent Downhole Gauges Using Data Mining Approaches. Presented at SPE Annual Technical Conference and Exhibition in Denver, Colorado, USA, 30 October – 2 November. SPE-147298-MS. https://doi.org/10.2118/147298-MS.

36. Liu, Y., and Horne, R. N. 2013. Interpreting Pressure and Flow-Rate Data From Permanent Downhole Gauges by Use of Data-Mining Approaches. *SPE J* **18** (1): 69–82. SPE-147298-PA. https://doi.org/10.2118/147298-PA.

37. Wolfram Research, Inc. 2018. Mathematica, Version 11.3. https://www.wolfram.com/mathematica.

38. Onur, M, Çinar, M., Ilk, D., Valkó, P. P., and Blasingame, T. A. 2008. An Investigation of Recent Deconvolution Methods for Well-Test Data Analysis. *SPE J* **13** (2): 226–247. SPE-102575-PA. https://doi.org/10.2118/102575-PA.

39. Onur, M., and Kuchuk, F. 2010. A New Pressure-Rate Deconvolution Technique Based on Pressure Derivatives for Pressure Transient Test Interpretation. Presented at SPE Annual Technical Conference and Exhibition in Florence, Italy, 19–22 September. SPE-134315-MS. https://doi.org/10.2118/134315-MS.

40. Onur, M., and Kuchuk, F. 2012. A New Deconvolution Technique Based on Pressure-Derivative Data for Pressure Transient Test Interpretation. *SPE J* **17** (1): 307–320. SPE-134315-PA. https://doi.org/10.2118/134315-PA.

41. Ozkan, E., and Raghavan, R. 1991. New Solutions for Well-Test-Analysis Problems: Part 1 – Analytical Considerations. *SPE Formation Eval* **6** (3): 359–368. SPE-18615-PA. https://doi.org/10.2118/18615-PA.

42. Ozkan, E., and Raghavan, R. 1991. New Solutions for Well-Test-Analysis Problems: Part 2 – Computational Considerations and Applications. *SPE Formation Eval* **6** (3): 369–378. SPE-18616-PA. https://doi.org/10.2118/18616-PA.

43. Pratikno, H., Rushing, J. A., and Blasingame, T. A. 2003. Decline Curve Analysis Using Type Curves – Fractured Wells. Presented at SPE Annual Technical Conference and Exhibition in Denver, Colorado, USA, 5–8 October. SPE-84287-MS. https://doi.org/10.2118/84287-MS.

44. Pimonov, E. A., Ayan, C., Onur, M., and Kuchuk, F. J. 2009. A New Pressure Rate Deconvolution Algorithm To Analyze Wireline Formation Tester and Well-Test data. Presented at SPE Annual Technical Conference and Exhibition in New Orleans, Louisiana, USA, 4–7 October. SPE-123982-MS. https://doi.org/10.2118/123982-MS.

45. Pimonov, E., Ayan, C., Onur, M., and Kuchuk, F. 2010. A New Pressure/Rate-Deconvolution Algorithm to Analyze Wireline-Formation-Tester and Well-Test Data. *SPE Res Eval & Eng.* **13** (4): 603–613. SPE-123982-PA. https://doi.org/10.2118/123982-PA.

46. Rasmussen, C. E., and Williams, C. K. I. 2006. *Gaussian Processes for Machine Learning*. Cambridge: The MIT Press. https://doi.org/10.1007/978-3-540-28650-9_4.

47. Robertson, S. 1988. Generalized Hyperbolic Equation. Available from SPE, Richardson, Texas, USA. SPE-18731-MS.

48. Salvatier, J., Wiecki, T.V., and Fonnesbeck, C. 2016. Probabilistic Programming in Python using PyMC3. *PeerJ Computer Science* **2** (55). https://doi.org/10.7717/peerj-cs.55.

49. Tian, C., and Horne, R. N. 2015. Applying Machine-Learning Techniques to Interpret Flow-Rate, Pressure, and Temperature Data From Permanent Downhole Gauges. Presented at SPE Western Regional Meeting in Garden Grove, California, USA, 27–30 April. SPE-174034-MS. https://doi.org/10.2118/174034-MS.

50. Tian, C., and Horne, R. N. 2019. Applying Machine-Learning Techniques to Interpret Flow-Rate, Pressure, and Temperature Data From Permanent Downhole Gauges. *SPE Res Eval & Eng.* Preprint. SPE-174034-PA. https://doi.org/10.2118/174034-PA.

51. Tian, C., and Horn, R. N. 2017. Recurrent Neural Networks for Permanent Downhole Gauge Data Analysis. Presented at SPE Annual Technical Conference and Exhibition in San Antonio, Texas, USA, 9–11 October. SPE-187181-MS. https://doi.org/10.2118/187181-MS.

52. Valkó, P. P., and Abate, J. 2004. Comparison of Sequence Accelerators for the Gaver Method of Numerical Laplace Transform Inversion. *Computers and Mathematics with Application* **48** (3): 629–636. https://doi.org/10.1016/j.camwa.2002.10.017.

53. Valkó, P., and Abate, J. 2002. Numerical Laplace Inversion. *Wolfram Library Archive*. http:// http://library.wolfram.com/infocenter/MathSource/4738.

54. Valkó, P. P. Assigning Value to Stimulation in the Barnett Shale: A Simultaneous Analysis of 7000 Plus Production Histories and Well Completion Records. 2009. Presented at SPE Hydraulic Fracturing Technology Conference in College Station, Texas, USA, 19–21 January. SPE-119369-MS. https://doi.org/10.2118/119369-MS.

55. van Everdingen, A. F., and Hurst, W. 1949. The Application of the Laplace Transformation to Flow Problems in Reservoirs. *Pet. Trans. AIME* **1** (12): 305–324. https://doi.org /10.2118/949305-G.

56. von Schroeter, T., Hollaender, F., and Gringarten, C. 2001. Deconvolution of Well Test Data as a Nonlinear Total Least Squares Problem. Presented at SPE Annual Technical Conference and Exhibition in New Orleans, Louisiana, USA, 30 September – 3 October. SPE-71754-MS. https://doi.org/10.2118/71754-MS.

57. von Schroeter, T., Hallaender, F., and Gringarten, A. C. 2002. Analysis of Well Test Data from Permanent Downhole Gauges by Deconvolution. Presented at SPE Annual Technical Conference and Exhibition in San Antonio, Texas, USA, 29 September – 2 October. SPE-77688-MS. https://doi.org/10.2118/77688-MS.

58. von Schroeter, T., Hallaender, F., and Gringarten, A. C. 2004. Deconvolution of Well Test Data as a Nonlinear Total Least-Squares Problem. *SPE J* **9** (4): 375–390. SPE-77688-PA. https://doi.org/10.2118/77688-PA.

APPENDIX A

DERIVATION OF BETA-DERIVATIVE SIGNATURES FOR FLOW REGIMES OF INTEREST

We propose the pressure drop function as power-law function with stepwise slope:

$$\ln \Delta p(t_n) = \ln \Delta p_1 + \sum_{i=2}^{n} \beta_i (\ln t_i - \ln t_{i-1}) \quad \text{.....A-1}$$

Or exponentiating both sides and writing in continuous form:

$$\Delta p(t_n) = \Delta p_1 \prod_{i=2}^{n} \left(\frac{t_i}{t_{i-1}} \right)^{\beta_i} = \Delta p_1 \prod_{i=2}^{n} t_i^{\beta_i} \, t_{i-1}^{-\beta_i} \quad \text{.....A-2}$$

Consider when $n = 2$:

$$\Delta p(t_{n=2}) = \Delta p_1 t_n^{\beta_n} \, t_{n-1}^{-\beta_n} \quad \text{.....A-3}$$

And when $n = 3$:

$$\Delta p(t_{n=3}) = \Delta p_1 t_{n-1}^{\beta_{n-1}} \, t_{n-2}^{-\beta_{n-1}} t_n^{\beta_n} \, t_{n-1}^{-\beta_n} \quad \text{.....A-4}$$

Changing the indices of Eq. A-3 in terms of $n = 3$:

$$\Delta p(t_{n=2}) \,|_{n=3} = \Delta p_1 t_{n-1}^{\beta_{n-1}} \, t_{n-2}^{-\beta_{n-1}} \quad \text{.....A-5}$$

Substituting Eq. A-5 into Eq. A-4 we have:

$$\Delta p(t_{n=3}) = \Delta p(t_{n=2}) t_n^{\beta_n} \, t_{n-1}^{-\beta_n} \quad \text{.....A-6}$$

Consider Eq. A-2 when $n = 4$:

$$\Delta p(t_{n=4}) = \Delta p_1 t_{n-2}^{\beta_{n-2}}\ t_{n-3}^{-\beta_{n-2}} t_{n-1}^{\beta_{n-1}}\ t_{n-2}^{-\beta_{n-1}} t_n^{\beta_n}\ t_{n-1}^{-\beta_n} \quad \text{A-7}$$

Writing Eq. A-3 in terms of $n = 4$:

$$\Delta p(t_{n=2})\,|_{n=4} = \Delta p_1 t_{n-2}^{\beta_{n-2}}\ t_{n-3}^{-\beta_{n-2}} \quad \text{A-8}$$

And Eq. A-4 in terms of $n = 4$:

$$\Delta p(t_{n=3})\,|_{n=4} = \Delta p_1 t_{n-1}^{\beta_{n-1}}\ t_{n-2}^{-\beta_{n-1}} \quad \text{A-9}$$

Following from before, we can substitute Eq. A-8 into Eq. A-7:

$$\Delta p(t_{n=4}) = \Delta p(t_{n=2}) t_{n-1}^{\beta_{n-1}}\ t_{n-2}^{-\beta_{n-1}} t_n^{\beta_n}\ t_{n-1}^{-\beta_n} \quad \text{A-10}$$

And then substitute Eq. A-9 into Eq. A-10:

$$\Delta p(t_{n=4}) = \Delta p(t_{n=3}) t_n^{\beta_n}\ t_{n-1}^{-\beta_n} \quad \text{A-11}$$

Generally, we can write $\Delta p(t_n)$ in Eq. A-2 as:

$$\Delta p(t_n) = \Delta p_1 \prod_{i=2}^{n} t_i^{\beta_i}\ t_{i-1}^{-\beta_i} = \Delta p(t_{n-1}) t_n^{\beta_n}\ t_{n-1}^{-\beta_n} \quad \text{A-12}$$

The utility of Eq. A-12 is that we can calculate $\Delta p(t_n)$ with just knowledge of $\Delta p(t_{n\text{-}1})$ and the current timestep's value of β_n. Because we are dealing with discrete timesteps, the value of β_n is a constant at each timestep, thus simplifying the computation of the derivative of Eq. A-35. We also note that although we have used the pressure function nomenclature, the result of Eq. A-12 is general and may also be used for the rate function, such as:

$$q(t_n) = q_1 \prod_{i=2}^{n} t_i^{\beta_{q,i}}\ t_{i-1}^{-\beta_{q,i}} = q(t_{n-1}) t_n^{\beta_{q,n}}\ t_{n-1}^{-\beta_{q,n}} \quad \text{A-13}$$

Derivative of Step Change Power-Law Function

To compute the derivative of Eq. 4, we will repeat the process of evaluating the first three discrete timesteps as a verification of the independence of pressure function only upon the prior timestep and a known β_i for the current timestep. Taking the derivative of Eq. A-2:

$$\frac{d}{dt}\Delta p(t_n) = \Delta p'(t_n) = \frac{d}{dt}\Delta p_1 \left[\prod_{i=2}^{n} t_i^{\beta_i}\, t_{i-1}^{-\beta_i} \right] \quad \text{.....A-14}$$

Consider the case where $n = 2$:

$$\Delta p'(t_{n=2}) = \Delta p_1 \frac{d}{dt}\left[\prod_{i=2}^{n=2} t_i^{\beta_i}\, t_{i-1}^{-\beta_i} \right] = \Delta p_1 \frac{d}{dt}\left[t_n^{\beta_n}\, t_{n-1}^{-\beta_n} \right] \quad \text{.....A-15}$$

Given the current timestep, any values of prior timesteps are constant as they are fixed in the past. Then, $t_{n-1}{}^{-\beta_i}$ is no longer a function of time but is constant. Factoring it out, we have:

$$\Delta p'(t_{n=2}) = \Delta p_1 t_{n-1}^{-\beta_n} \frac{d}{dt}\left[t_n^{\beta_n} \right] = \Delta p_1 t_{n-1}^{-\beta_n} \beta_n t_n^{\beta_n - 1} \quad \text{.....A-16}$$

Multiplying by $\frac{t_n}{t_n}$:

$$\Delta p'(t_{n=2}) = \Delta p_1 t_{n-1}^{-\beta_n} \beta_n t_n^{\beta_n - 1} \frac{t_n}{t_n} = \Delta p_1 t_n^{\beta_n} t_{n-1}^{-\beta_n} \frac{\beta_n}{t_n} \quad \text{.....A-17}$$

We immediately note we can substitute Eq. A-3 into Eq. A-17:

$$\Delta p'(t_{n=2}) = \Delta p(t_{n=2}) \frac{\beta_n}{t_n} \quad \text{.....A-18}$$

Similarly, for $n = 3$:

$$\Delta p'(t_{n=3}) = \Delta p_1 \frac{d}{dt}\left[\prod_{i=2}^{n=3} t_i^{\beta_i}\, t_{i-1}^{-\beta_i}\right] = \Delta p_1 \frac{d}{dt}\left[t_{n-1}^{\beta_{n-1}} t_{n-2}^{-\beta_{n-1}} t_n^{\beta_n} t_{n-1}^{-\beta_n}\right] \quad \text{A-19}$$

Factoring out constants as only $t_n^{\beta_n}$ is a function of time:

$$\Delta p'(t_{n=3}) = \Delta p_1 t_{n-1}^{\beta_{n-1}} t_{n-2}^{-\beta_{n-1}} t_{n-1}^{-\beta_n} \frac{d}{dt}\left[t_n^{\beta_n}\right] \quad \text{A-20}$$

We immediately note that we can substitute Eq. A-5 into Eq. A-20:

$$\Delta p'(t_{n=3}) = \Delta p(t_{n=2}) t_{n-1}^{-\beta_n} \frac{d}{dt}\left[t_n^{\beta_n}\right] \quad \text{A-21}$$

Evaluating the derivative:

$$\Delta p'(t_{n=3}) = \Delta p(t_{n=2}) t_{n-1}^{-\beta_n}\left[\beta_n t_n^{\beta_n - 1}\right] \quad \text{A-22}$$

Multiplying by $\frac{t_n}{t_n}$:

$$\Delta p'(t_{n=3}) = \Delta p(t_{n=2}) t_{n-1}^{-\beta_n} \beta_n t_n^{\beta_n - 1} \frac{t_n}{t_n} = \Delta p(t_{n=2}) t_n^{\beta_n} t_{n-1}^{-\beta_n} \frac{\beta_n}{t_n} \quad \text{A-23}$$

And substituting Eq. A-6 into Eq. A-23:

$$\Delta p'(t_{n=3}) = \Delta p(t_{n=3}) \frac{\beta_n}{t_n} \quad \text{A-24}$$

Lastly, for $n = 4$:

$$\Delta p'(t_{n=4}) = \Delta p_1 \frac{d}{dt}\left[\prod_{i=2}^{n=4} t_i^{\beta_i}\, t_{i-1}^{-\beta_i}\right] = \Delta p_1 \frac{d}{dt}\left[t_{n-2}^{\beta_{n-2}} t_{n-3}^{-\beta_{n-2}} t_{n-1}^{\beta_{n-1}} t_{n-2}^{-\beta_{n-1}} t_n^{\beta_n} t_{n-1}^{-\beta_n}\right] \quad \text{A-25}$$

Factoring out constants:

$$\Delta p'(t_{n=4}) = \Delta p_1 t_{n-2}^{\beta_{n-2}} t_{n-3}^{-\beta_{n-2}} t_{n-1}^{\beta_{n-1}} t_{n-2}^{-\beta_{n-1}} t_{n-1}^{-\beta_n} \frac{d}{dt}\left[t_n^{\beta_n} \right] \quad \text{A-26}$$

Substituting Eq. A-8 into Eq. A-26:

$$\Delta p'(t_{n=4}) = \Delta p(t_{n=2}) t_{n-1}^{\beta_{n-1}} t_{n-2}^{-\beta_{n-1}} t_{n-1}^{-\beta_n} \frac{d}{dt}\left[t_n^{\beta_n} \right] \quad \text{A-27}$$

Substituting Eq. A-6 into Eq. A-27:

$$\Delta p'(t_{n=4}) = \Delta p(t_{n=3}) t_{n-1}^{-\beta_n} \frac{d}{dt}\left[t_n^{\beta_n} \right] \quad \text{A-28}$$

Evaluating the derivative:

$$\Delta p'(t_{n=4}) = \Delta p(t_{n=3}) t_{n-1}^{-\beta_n} \left[\beta_n t_n^{\beta_n - 1} \right] \quad \text{A-29}$$

Multiplying by $\frac{t_n}{t_n}$:

$$\Delta p'(t_{n=4}) = \Delta p(t_{n=3}) t_n^{\beta_n - 1} t_{n-1}^{-\beta_n} \beta_n \frac{t_n}{t_n} = \Delta p(t_{n=3}) t_n^{\beta_n} t_{n-1}^{-\beta_n} \frac{\beta_n}{t_n} \quad \text{A-30}$$

Finally, substituting Eq. A-11:

$$\Delta p'(t_{n=4}) = \Delta p(t_{n=4}) \frac{\beta_n}{t_n} \quad \text{A-31}$$

As with the pressure function, we have verified that the pressure derivative function is dependent only upon the immediately prior timestep. Recalling Eq. A-2:

$$\Delta p(t_n) = \Delta p_1 \prod_{i=2}^{n} t_i^{\beta_i} \, t_{i-1}^{-\beta_i} \quad \text{A-2}$$

And taking the derivative:

$$\frac{d}{dt}\Delta p(t_n) = \Delta p'(t_n) = \frac{d}{dt}\Delta p_1 \left[\prod_{i=2}^{n} t_i^{\beta_i}\, t_{i-1}^{-\beta_i} \right] \quad \text{.....A-32}$$

Factoring out Δp_1 and $t_{i-1}^{-\beta_i}$ as a constants and dropping the subscript n as it is no longer necessary:

$$\Delta p'(t) = \Delta p_1 \prod_{i=2}^{n} t_{i-1}^{-\beta_i} \frac{d}{dt}\left[t_i^{\beta_i} \right] \quad \text{.....A-33}$$

Evaluating the derivative in Eq. A-33:

$$\Delta p'(t) = \Delta p_1 \prod_{i=2}^{n} t_{i-1}^{-\beta_i} \left[\beta_i t_i^{\beta_i - 1} \right] \quad \text{.....A-34}$$

Multiplying by $\frac{t_i}{t_i}$:

$$\Delta p'(t) = \Delta p_1 \prod_{i=2}^{n} \beta_i t_i^{\beta_i - 1}\, t_{i-1}^{-\beta_i} \frac{t_i}{t_i} = \Delta p_1 \prod_{i=2}^{n} t_i^{\beta_i}\, t_{i-1}^{-\beta_i} \frac{\beta_i}{t_i} \quad \text{.....A-35}$$

which matches our results for the discrete timestep derivative. Substituting Eq. A-2 into Eq. A-35, we can compute the derivative in timestep-to-timestep as:

$$\Delta p'(t) = \Delta p(t) \frac{\beta(t)}{t} \quad \text{.....A-36}$$

Re-arranging yields the well-testing derivative on the LHS:

$$t\Delta p'(t) = \beta(t)\Delta p(t) \quad \text{.....A-37}$$

And solving for β, we verify that it matches the definition of the beta-derivative:

$$\beta(t) \equiv \frac{d \ln \Delta p(t)}{d \ln t} = \frac{t}{\Delta p(t)} \Delta p'(t) \quad \text{.....A-38}$$

Power-law Flow Diagnostic from Beta-Derivative

The beta-derivative is defined as the log-log derivative, as seen in Eq. A-39. Therefore, any power-law flow regime will be transformed into a linear equation, for which the derivative is a constant. Starting with:

$$\Delta p(t) = c_2 + c_1 t^n \quad \text{A-39}$$

Where n is an exponent corresponding to some relation of time, such as n = ½ for linear flow, or n = ¼ for bi-linear flow, etc. Taking the log-log derivative (beta-derivative), we have:

$$\beta(t) = \frac{d\left[\ln\left(c_2 + c_1 t^n\right)\right]}{d\ln t} \quad \text{A-40}$$

Applying the log-log derivative identity, this becomes:

$$\beta(t) = \frac{t}{c_2 + c_1 t^n}\frac{d\left[c_2 + c_1 t^n\right]}{dt} \quad \text{A-41}$$

Evaluating the derivative, we have:

$$\beta(t) = \frac{t}{c_2 + c_1 t^n} c_1 \alpha t^{n-1} \quad \text{A-42}$$

Simplifying:

$$\beta(t) = \frac{c_1 \alpha t^n}{c_2 + c_1 t^n} \quad \text{A-43}$$

If we assume sufficient time has passed to minimize the influence of the intercept term, i.e.:

$$c_2 \ll c_1 t$$

Then we have:

$$\beta(t) = \frac{c_1 \alpha t^n}{c_1 t^n} = n \quad \text{A-44}$$

Indicating that the beta-derivative will converge to a constant as a diagnostic of a power-law flow regime.

Radial Flow Diagnostic from Beta-Derivative:

For radial flow, we have the "log approximation" from van Everdingen and Hurst (1949):

$$\Delta p(t) \propto \ln t = c_2 + c_1 \ln t \quad \text{A-45}$$

Taking the derivative of Eq. A-45 gives us:

$$\Delta p'(t) = \frac{c_1}{t} \quad \text{A-46}$$

Multiplying by *t* yields the "well-testing" derivative:

$$t\Delta p'(t) = c_1 \quad \text{A-47}$$

That is, Eq. A-47 demonstrates that the well-testing derivative will yield a constant in radial flow with a slope of zero. Substituting Eq. A-37 into Eq. A-47:

$$\beta(t)\Delta p(t) = c_1 \quad \text{A-48}$$

Substituting Eq. A-48 into Eq. A-45:

$$\Delta p(t) = c_2 + \beta(t)\Delta p(t) \ln t \quad \text{A-49}$$

Dividing by $p(t)$ and solving for $\beta(t)\ln t$ yields:

$$\beta(t)\ln t = 1 - \frac{c_2}{\Delta p(t)} \quad \text{.....A-50}$$

Given by the fact that in Eq.-49 we have $c_2 \rightarrow 0$ as $t \rightarrow 0$, we assume that:

$$\frac{c_2}{\Delta p(t)} \ll 1$$

and we can simplify to:

$$\beta(t)\ln t = 1 \quad \text{.....A-51}$$

Or in other words, a value of $\beta(t)\ln t = 1$ is diagnostic of radial flow.

Pseudosteady-state Flow Diagnostic from Beta-Derivative:

For pseudosteady-state flow, we have a linear change in rate or pressure with respect to time. Assuming the distance to reservoir boundary is much larger than the wellbore diameter:

$$r_e \gg r_w$$

The solution is given by van Everdingen and Hurst (1949):

$$p_D\left(r_D, t_D\right)\Big|_{r_D = r_{wD}} = \ln r_{eD} - \frac{3}{4} + \frac{2t_D}{r_{eD}^2} \quad \text{.....A-52}$$

Which we may re-write as:

$$\Delta p(t) = c_2 + c_1 t \quad \text{.....A-53}$$

Taking the logarithm Eq. A-53:

$$\ln \Delta p(t) = \ln\left[c_2 + c_1 t\right] \quad \text{A-54}$$

Taking the derivative with respect to the logarithm of time:

$$\frac{d \ln \Delta p(t)}{d \ln t} = \frac{d\left[c_2 + c_1 t\right]}{d \ln t} \quad \text{A-55}$$

We immediately note that the LHS is the definition of the beta-derivative in Eq. A-38:

$$\beta(t) \equiv \frac{d \ln \Delta p(t)}{d \ln t} = \frac{t}{\Delta p(t)} \Delta p'(t) \quad \text{A-38}$$

Substituting Eq. A-38 into Eq. A-55:

$$\beta(t) = \frac{d\left[c_2 + c_1 t\right]}{d \ln t} \quad \text{A-56}$$

Evaluating the derivative on the RHS:

$$\beta(t) = \frac{t}{c_2 + c_1 t} c_1 = \frac{c_1 t}{c_2 + c_1 t} \quad \text{A-57}$$

If we assume t is large as pseudosteady state flow occurs at late-time, i.e.:

$$c_2 \ll c_1 t$$

Then we have:

$$\beta(t) \approx \frac{c_1 t}{c_1 t} = 1 \quad \text{A-58}$$

Indicating that a value of $\beta(t) = 1$ is diagnostic of pseudosteady-state flow.

Boundary Dominated Flow Diagnostic from Beta-Derivative

With the same method used to derive Eq. A-38, we can formulate the rate function derivative as:

$$q'(t) = q(t)\frac{\beta_q(t)}{t} \quad \text{.....A-59}$$

Re-arranging yields the Arps *D*-parameter:

$$\frac{1}{q(t)}q'(t) = \frac{\beta_q(t)}{t} \quad \text{.....A-60}$$

where:

$$D(t) \equiv \frac{d}{dt}\ln q(t) \equiv \frac{-1}{q(t)}q'(t) \quad \text{.....A-61}$$

And substituting the identity yields:

$$\frac{\beta_q(t)}{t} = -D(t) \quad \text{.....A-62}$$

For boundary dominated flow, we have from the combination of the liquids relations for material balance and pseudosteady-state flow that rate is proportional to an exponential of time:

$$q(t) \propto e^{-Dt} = c_2 + c_1 e^{-Dt} \quad \text{.....A-63}$$

Because rate must eventually decline to zero at late-time, we can say that:

$$c_2 = 0$$

Taking the logarithm of Eq. A-63:

$$\ln q(t) = \ln c_1 - Dt \quad \text{A-64}$$

Taking the derivative of Eq. A-64:

$$\frac{d}{dt}\ln q(t) = \frac{d}{dt}\ln c_1 - \frac{d}{dt}Dt \quad \text{A-65}$$

Evaluating the derivative:

$$\frac{1}{q(t)}q'(t) = -D \quad \text{A-66}$$

yields the Arps *D*-parameter as a constant. Therefore, a constant value of $\beta_q(t)/t$ is diagnostic of boundary dominated flow.

Elimination of intercept term

Each of the derived identities for pressure function diagnostics is not exact due to the existence of an intercept term we have neglected. Alternatively, we can eliminate the c_2 slope term by transforming $\beta(t)$ to the well-testing derivative as in Eq. A-37:

$$t\Delta p'(t) = \beta(t)\Delta p(t) \quad \text{A-37}$$

If:

$$\Delta p(t) = c_2 + c_1\tau \quad \text{A-67}$$

Where τ is a function of time respective to each flow regime, such as:

$\tau = t$ ..... for pseudosteady-state flow

$\tau = \ln t$ ..... for radial flow

$\tau = t^n$ ..... for power-law flow

And n = ½ for linear flow, n = ¼ for bi-linear flow, etc.

Taking the derivative of the Eq A-67 with respect to the log of time:

$$\frac{d\Delta p(t)}{d\ln t} = \frac{d\left[c_2 + c_1\tau\right]}{d\ln t} \quad \text{A-68}$$

Substituting the log derivative identity:

$$t\frac{d\Delta p(t)}{dt} = t\frac{d\left[c_2 + c_1\tau\right]}{d\ln t} \quad \text{A-69}$$

Evaluating the derivative for each case of τ:

$$t\Delta p'(t) = t\,c_1\Big|_{\tau=t} \quad \text{A-70}$$

$$t\Delta p'(t) = t\frac{c_1}{t}\Big|_{\tau=\ln t} = c_1 \quad \text{A-71}$$

$$t\Delta p'(t) = tc_1 nt^{n-1}\Big|_{\tau=t^n} = c_2 nt^n \quad \text{A-72}$$

And taking the derivative of the log of each case with respect to the log of time:

$$\frac{d\ln\left[t\Delta p'(t)\right]}{d\ln t} = \frac{d\left[\ln tc_1\right]}{d\ln t}\Big|_{\tau=t} \quad \text{A-73}$$

$$\frac{d\ln\left[t\Delta p'(t)\right]}{d\ln t} = \frac{d\ln c_1}{d\ln t}\Big|_{\tau=\ln t} \quad \text{A-74}$$

$$\frac{d\ln\left[t\Delta p'(t)\right]}{d\ln t} = \frac{d\left[\ln c_2 nt^n\right]}{d\ln t}\Big|_{\tau=t^n} \quad \text{A-75}$$

Substituting the log-log derivative identity:

$$\frac{t}{t\Delta p'(t)}\frac{dt\Delta p'(t)}{dt}=\frac{t}{tc_1}\frac{dtc_1}{dt}\bigg|_{\tau=t} \quad \text{A-76}$$

$$\frac{t}{t\Delta p'(t)}\frac{dt\Delta p'(t)}{dt}=\frac{t}{c_1}\frac{dc_1}{dt}\bigg|_{\tau=\ln t} \quad \text{A-77}$$

$$\frac{t}{t\Delta p'(t)}\frac{dt\Delta p'(t)}{dt}=\frac{t}{c_2 nt^{\alpha}}\frac{dc_2 nt^{n}}{dt}\bigg|_{\tau=t^n} \quad \text{A-78}$$

Evaluating each derivative:

$$\frac{t}{t\Delta p'(t)}\frac{dt\Delta p'(t)}{dt}=\frac{t}{tc_1}c_1\bigg|_{\tau=t}=1 \quad \text{A-79}$$

$$\frac{t}{t\Delta p'(t)}\frac{dt\Delta p'(t)}{dt}=\frac{t}{c_1}0\bigg|_{\tau=\ln t}=0 \quad \text{A-80}$$

$$\frac{t}{t\Delta p'(t)}\frac{dt\Delta p'(t)}{dt}=\frac{t}{c_2 nt^{c_3}}c_2 n^2 t^{n-1}\bigg|_{\tau=t^n}=n \quad \text{A-81}$$

And substituting Eq. A-38 into each:

$$\frac{t}{\beta(t)\Delta p(t)}\frac{d\beta(t)\Delta p(t)}{dt}=1\big|_{\tau=t} \quad \text{A-82}$$

$$\frac{t}{\beta(t)\Delta p(t)}\frac{d\beta(t)\Delta p(t)}{dt}=0\big|_{\tau=\ln t} \quad \text{A-83}$$

$$\frac{t}{\beta(t)\Delta p(t)}\frac{d\beta(t)\Delta p(t)}{dt}=n\big|_{\tau=t^n} \quad \text{A-84}$$

Which reproduces the classic well-testing diagnostic based upon a known beta-derivative. For boundary-dominated flow, we eliminate the intercept term in the course of the

derivation of the beta-derivative identity, so no additional derivation is needed. We would set the slope of the log-log plot of the *D*-parameter to zero, resulting in Eq. A-86.

$$\frac{t^2}{\beta_q(t)}\frac{d}{dt}\left[\frac{\beta_q(t)}{t}\right] = n\Big|_{\tau=t^n} \quad \text{.....A-86}$$

Summarizing

The power-law function can reproduce exponential, logarithmic, or power-law behavior:

Table A-1 — Summary of beta-derivative diagnostics

Flow Regime	Beta-derivative Diagnostic	Well Testing Diagnostic
Radial	$\beta \ln t = 1$	$\frac{t}{\beta\Delta p}\frac{d\beta\Delta p}{dt} = 0$
Linear	$\beta = \frac{1}{2}$	$\frac{t}{\beta\Delta p}\frac{d\beta\Delta p}{dt} = \frac{1}{2}$
Boundary-Dominated	$\frac{\beta_q}{t} = \text{constant}$	n/a
Pseudosteady State	$\beta_p = 1$	$\frac{t}{\beta\Delta p}\frac{d\beta\Delta p}{dt} = 1$

APPENDIX B

DERIVATATION OF RADIUS OF CURVATURE

In this appendix we derive the radius of curvature measurement utilized as a regularization on smoothness of the constant-rate pressure and constant-pressure-drop rate kernel functions. Radius of curvature, R, is common in other field of engineering, for example in beam deflection. It represents the radius of a circle having the same curvature as a portion of the function of interest. The arc length s is related to R by the angle of the arc θ as shown in **Fig. B-1** and in Eq. B-1. T is a tangent line at a point where we desire to measure the curvature.

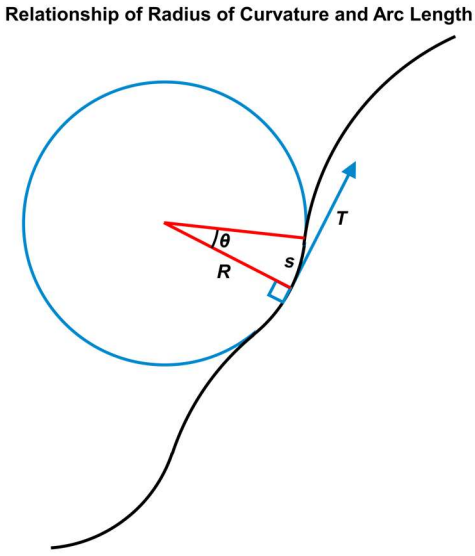

Figure B-1 — Relationship of arc length to radius of curvature and angle of the arc.

$$R\theta = s \quad \text{B-1}$$

And if we take the derivative of both sides of Eq. B-1 with respect to the change in time, we have:

$$R\frac{d\theta}{dt} = \frac{ds}{dt} \quad \text{B-2}$$

Left-hand Side

We will first determine the $d\theta/dt$ term on the LHS. Given $y = f(t)$, we may write:

$$\tan\theta = \frac{dy}{dt} \quad \text{B-3}$$

And solving for θ we have:

$$\theta = \tan^{-1}\frac{dy}{dt} \quad \text{B-4}$$

Taking the derivative with respect to t:

$$\frac{d\theta}{dt} = \frac{d}{dt}\left[\tan^{-1}\frac{dy}{dt}\right] \quad \text{B-5}$$

Where the derivative of $\tan^{-1}$ is given by Abramowitz and Stegun (1964):

$$\frac{d}{dx}\tan^{-1} z = \frac{1}{1+z^2} \quad \text{B-6}$$

Defining:

$$u = \frac{dy}{dt} \quad \text{B-7}$$

Then:

$$\frac{du}{dt} = \frac{d^2 y}{dt^2} \quad \text{B-8}$$

Recalling the chain rule:

$$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx}$$

Substituting in to Eq. B-5 and applying the chain rule, we have:

$$\frac{d\theta}{dt} = \frac{d}{du}\left[\tan^{-1} u\right]\frac{du}{dt} \quad \text{B-9}$$

Substituting in Eqs. B-6, B-7, and B-8 we have:

$$\frac{d\theta}{dt} = \frac{\frac{d^2 y}{dt^2}}{1+\left[\frac{dy}{dt}\right]^2} \quad \text{B-10}$$

Right-hand Side

Now we determine the *ds/dt* term on the RHS. Starting with the change in arc length given a change in Cartesian position:

$$ds^2 = dy^2 + dt^2 \quad \text{B-11}$$

Dividing both sides by dt^2:

$$\frac{ds^2}{dt^2} = \frac{dy^2}{dt^2} + 1 \quad \text{B-12}$$

Taking the square root of Eq. B-12:

$$\frac{ds}{dt} = \sqrt{1 + \left[\frac{dy}{dt}\right]^2} \quad \text{B-13}$$

Substituting Eqs. B-10 and B-13 into Eq. B-2:

$$R\frac{\frac{d^2y}{dt^2}}{1 + \left[\frac{dy}{dt}\right]^2} = \sqrt{1 + \left[\frac{dy}{dt}\right]^2} \quad \text{B-14}$$

And solving for K we obtain:

$$R = \frac{\left[1 + \left[\frac{dy}{dt}\right]^2\right]^{3/2}}{\frac{d^2y}{dt^2}} \quad \text{B-15}$$

In Eq. B-15, R will be large when y is smooth, and small when y is sharp. We wish to regularize a parameter such that a model is less likely, or penalty is high, when a parameter is large. Therefore, we take the reciprocal of Eq. B-15 to obtain curvature K:

$$K = \frac{1}{R} = \frac{\frac{d^2y}{dt^2}}{\left[1 + \left[\frac{dy}{dt}\right]^2\right]^{3/2}} \quad \text{B-16}$$

Eq. B-16 is physically interpreted as the curvature being equal to the second-order derivative of the function of interest, with respect to the local slope (first-order derivative) of the function.

Transform for log values

The curvature is applied to the first-order derivative of the log of the kernel function with respect to the log of time, but the second-order derivative is not with respect to the log of the kernel function but *is* with respect to the log of time. If we substitute in the log-log derivative identities,

$$\frac{d}{d\ln x}\left[\frac{d\ln y}{d\ln x}\right] = x\frac{d}{dx}\left[\frac{x}{y}\frac{dy}{dx}\right]$$

We have:

$$K = \frac{t\frac{d}{dt}\left[\frac{t}{y}\frac{dy}{dt}\right]}{\left[1+\left[\frac{t}{y}\frac{dy}{dt}\right]^2\right]^{3/2}} \quad \text{B-17}$$

Which requires the quotient rule to evaluate. Defining:

$$u = t\frac{dy}{dt} \quad \text{B-18}$$

$$v = y \quad \text{B-19}$$

$$\frac{du}{dt} = t\frac{d^2y}{dt^2} + \frac{dy}{dt} \quad \text{B-20}$$

$$\frac{dv}{dt} = \frac{dy}{dt} \quad \text{B-21}$$

Recalling the quotient rule:

$$\frac{d}{dt}\frac{u}{v} = \frac{v\frac{du}{dt} - u\frac{dv}{dt}}{v^2} \quad \text{B-22}$$

Substituting in Eqs. B-18 – B-23 into the numerator of Eq. B-17 we have:

$$t\frac{d}{dt}\left[\frac{t}{y}\frac{dy}{dt}\right]=t\left[\frac{y\left[t\frac{d^2y}{dt^2}+\frac{dy}{dt}\right]-t\frac{dy}{dt}\frac{dy}{dt}}{y^2}\right] \quad \text{B-23}$$

Which simplifies to:

$$t\frac{d}{dt}\left[\frac{t}{y}\frac{dy}{dt}\right]=\frac{t^2}{y}\frac{d^2y}{dt^2}+\frac{t}{y}\frac{dy}{dt}-\frac{t^2}{y^2}\left[\frac{dy}{dt}\right]^2 \quad \text{B-24}$$

And substituting Eq. B-24 into Eq. B-17 results:

$$K=\frac{\frac{t^2}{y}\frac{d^2y}{dt^2}+\frac{t}{y}\frac{dy}{dt}-\frac{t^2}{y^2}\left[\frac{dy}{dt}\right]^2}{\left[1+\left[\frac{t}{y}\frac{dy}{dt}\right]^2\right]^{3/2}} \quad \text{B-25}$$

If we substitute τ into Eq. B-25, where τ is defined as:

$$\tau = \ln t \quad \text{B-26}$$

$$d\tau=\frac{1}{t}dt \quad \text{B-27}$$

And, similarly:

$$d\upsilon=\frac{1}{y}dy \quad \text{B-28}$$

We obtain a simplified form for numerical evaluation using finite difference approximations:

$$K = \frac{y\dfrac{d^2\upsilon}{d\tau^2} + \dfrac{d\upsilon}{d\tau} - \left[\dfrac{d\upsilon}{d\tau}\right]^2}{\left[1 + \left[\dfrac{d\upsilon}{d\tau}\right]^2\right]^{3/2}} \quad \text{B-29}$$

We also note that, when $dy/dt \ll 1$, then Eq. B-16 reduces to a form similar but not equal to that derived by von Schroeter, Hollaender, and Gringarten (2002, 2004).

$$K \approx \frac{d^2y}{dt^2} \quad \text{B-30}$$

Whereas their function in finite difference form is not a second derivative, but:

$$K \approx \frac{d^2y}{dt} = \frac{y_{i+1} - y_i}{t_{i+1} - t_i} - \frac{y_i - y_{i-1}}{t_i - t_{i-1}} \quad \text{B-31}$$

And with some manipulation:

$$K \approx \frac{y_{i+1} - y_i}{t_{i+1} - t_i}\left[\frac{t_i - t_{i-1}}{t_i - t_{i-1}}\right] - \frac{y_i - y_{i-1}}{t_i - t_{i-1}}\left[\frac{t_{i+1} - t_i}{t_{i+1} - t_i}\right]$$

$$K \approx \frac{y_{i+1}}{t_{i+1} - t_i} - \frac{y_i}{t_{i+1} - t_i}\left[\frac{t_i - t_{i-1}}{t_i - t_{i-1}}\right] - \frac{y_i}{t_i - t_{i-1}}\left[\frac{t_{i+1} - t_i}{t_{i+1} - t_i}\right] + \frac{y_{i-1}}{t_i - t_{i-1}}$$

$$K \approx \frac{y_{i+1}}{t_{i+1} - t_i} - \frac{y_i\left(t_i - t_{i-1}\right)}{\left(t_{i+1} - t_i\right)\left(t_i - t_{i-1}\right)} - \frac{y_i\left(t_{i+1} - t_i\right)}{\left(t_i - t_{i-1}\right)\left(t_{i+1} - t_i\right)} + \frac{y_{i-1}}{t_i - t_{i-1}}$$

$$K \approx \frac{y_{i+1}}{t_{i+1} - t_i} - \frac{y_i\left[\left(t_i - t_{i-1}\right) + \left(t_{i+1} - t_i\right)\right]}{\left(t_{i+1} - t_i\right)\left(t_i - t_{i-1}\right)} + \frac{y_{i-1}}{t_i - t_{i-1}}$$

We obtain:

$$K \approx \frac{y_{i+1}}{t_{i+1} - t_i} - \frac{y_i\left(t_{i+1} - t_{i-1}\right)}{\left(t_{i+1} - t_i\right)\left(t_i - t_{i-1}\right)} + \frac{y_{i-1}}{t_i - t_{i-1}} \quad \text{B-32}$$

Which is identical to Eq. 28 in von Schroeter, Hollaender, and Gringarten (2002). Our result suggests Equation B-32 can be improved by substitution into Eq. B-30, yielding:

$$K \approx \frac{1}{dt}\left[\frac{y_{i+1}}{t_{i+1}-t_i} - \frac{y_i\left(t_{i+1}-t_{i-1}\right)}{\left(t_{i+1}-t_i\right)\left(t_i-t_{i-1}\right)} + \frac{y_{i-1}}{t_i-t_{i-1}}\right]$$

$$K \approx \frac{1}{t_{i+1}-t_i}\left[\frac{y_{i+1}}{t_{i+1}-t_i} - \frac{y_i\left(t_{i+1}-t_{i-1}\right)}{\left(t_{i+1}-t_i\right)\left(t_i-t_{i-1}\right)} + \frac{y_{i-1}}{t_i-t_{i-1}}\right]$$

$$K \approx \frac{y_{i+1}}{\left(t_{i+1}-t_i\right)^2} - \frac{y_i\left(t_{i+1}-t_{i-1}\right)}{\left(t_{i+1}-t_i\right)^2\left(t_i-t_{i-1}\right)} + \frac{y_{i-1}}{\left(t_{i+1}-t_i\right)\left(t_i-t_{i-1}\right)} \quad \text{..... B-33}$$

Which is, effectively, just the second-order derivative. **Fig. B-2** compares the von Schroeter et al. curvature function (Eq. B-32), our "correction" (Eq. B-33), and the curvature function derived in this work (Eq. B-29). All functions are evaluated with respect to the log values of β_p and time. (If we used untransformed values, the curvature at late times would be approximately zero.) We observe in **Fig. B-2** that the measurement of curvature is analogous between the von Schroeter curvature function and the other two functions at early-time. However, at late-times the von Schroeter function underestimates the curvature. The correction we have applied alleviates this somewhat, but another issue presents itself in that the sign of the functions is incorrect. At 100 hours, the curvature of the beta-derivative is convex, indicating positive curvature, and continues to be so for the next few knots. The von Schroeter function and the correction both evaluate to negative values over these knots.

The pseudocode for our curvature function is:

```
function float[] deriv_1(float[] a):
 # 1st order central difference derivative
 return (a[2:] - a[:-2]) / 2

function float[] deriv_2(float[] a):
 # 2nd order central difference derivative
 return a[2:] - a[1:-1] + a[:-2]

function float[] curvature(float[] x, float[] y,
 bool logx, bool logy):
 # compute curvature
 if logx:
    x = log(x)
 if logy:
    y0 = y
    y = log(y)

 dx = deriv_1(x)
 dy = deriv_1(y)
 d2y = deriv_2(y)
 if logy:
    d2y *= y0
 d2ydx2 = d2y / dx**2

 if logx:
    d2ydx2 += dy / dx - (dy / dx)**2

 return d2ydx2 / (1 + (dy / dx)**2)**(3 / 2)
```

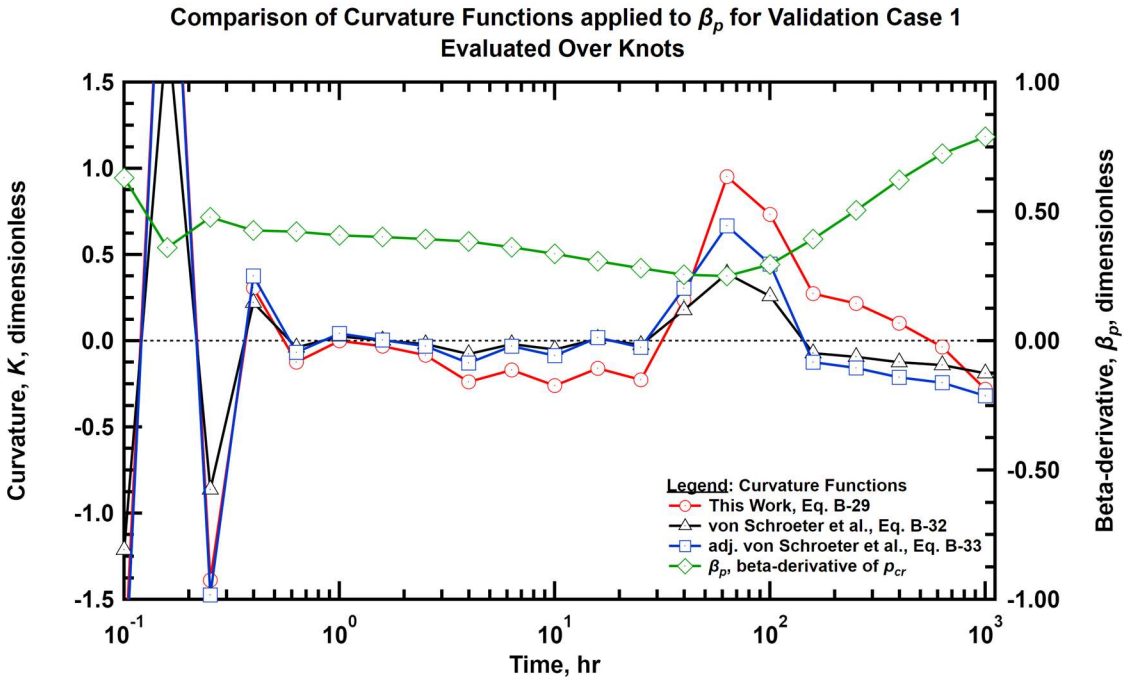

Figure B-2 — Comparison of curvature functions for Validation Case 1.

To investigate the issue further, we evaluate each curvature function against a sin function with logarithmically spaced knots. In **Fig. B-3**, we observe that the von Schroeter curvature function is does not yield consistent values of curvature over each period of the function, and would in fact grow unbounded to infinity as the knots become ever closer together. While our correction in Eq. B-33 does resolve this behavior, the curvature as the sin function passes through zero grows too rapidly because the correction does not normalize with respect to the slope of the function. We conclude that our derived function for curvature is the optimal one for regularization.

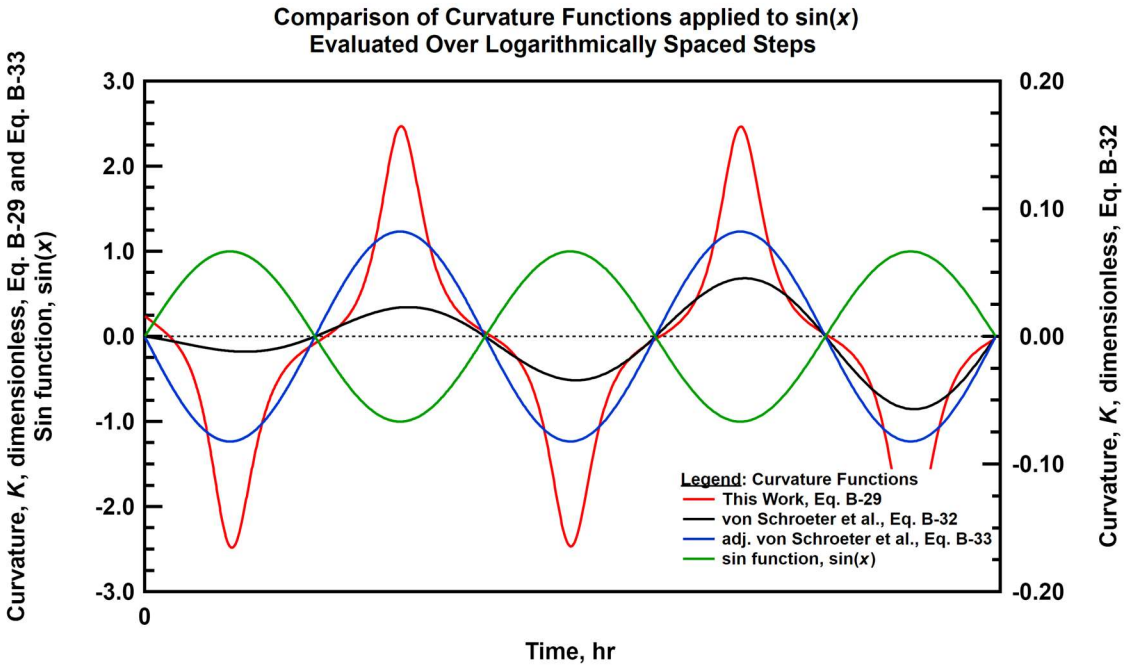

Figure B-3 — Comparison of curvature functions for evaluated on a sin function.

APPENDIX C

DERIVATION OF B-SPLINE DERIVATIVE AND INTEGRAL

In this appendix we derive the integration identity for B-splines. Although B-splines are defined piecewise, their derivatives and integrals are continuous over the knots that define the B-spline. This allows us to quickly and easily interpolate a degree k+1continuous spline function after numerical integration of a degree k spline function.

Beginning with the derivative of the spline function:

$$\frac{d}{dx}\sum_{i=\infty}^{\infty} c_i B_i^k\left(x,t^*\right) = \sum_{i=\infty}^{\infty} c_i \frac{d}{dx} B_i^k\left(x,t^*\right) \quad \text{..... C-1}$$

We need to compute the derivative of the B-splines. Recalling the recursive definition of a B-spline for $k > 1$:

$$B_i^k\left(x,t^*\right) = \frac{x - t^*_i}{t^*_{i+k} - t^*_i} B_i^{k-1}\left(x,t^*\right) + \frac{t^*_{i+k+1} - x}{t^*_{i+k+1} - t^*_{i+1}} B_{i+1}^{k-1}\left(x,t^*\right) \quad \text{..... C-2}$$

And taking the derivative:

$$\frac{d}{dx} B_i^k\left(x,t^*\right) = \frac{1}{t^*_{i+k} - t^*_i} \frac{d}{dx} B_i^{k-1}\left(x,t^*\right) - \frac{1}{t^*_{i+k+1} - t^*_{i+1}} \frac{d}{dx} B_{i+1}^{k-1}\left(x,t^*\right) \quad \text{..... C-3}$$

Which would have k recursions such that:

$$\frac{d}{dx} B_i^k\left(x,t^*\right) = \frac{k}{t^*_{i+k} - t^*_i} B_i^{k-1}\left(x,t^*\right) - \frac{k}{t^*_{i+k+1} - t^*_{i+1}} B_{i+1}^{k-1}\left(x,t^*\right) \quad \text{..... C-4}$$

Substituting into Eq. C-1:

$$\frac{d}{dx}\sum_{i=-\infty}^{\infty} c_i B_i^k\left(x,t^*\right)=\sum_{i=-\infty}^{\infty} c_i\left[\frac{k}{t^*_{i+k}-t^*_i}B_i^{k-1}\left(x,t^*\right)-\frac{k}{t^*_{i+k+1}-t^*_{i+1}}B_{i+1}^{k-1}\left(x,t^*\right)\right] \quad \text{C-5}$$

Distributing the coefficient c_i:

$$\frac{d}{dx}\sum_{i=-\infty}^{\infty} c_i B_i^k\left(x,t^*\right)=\sum_{i=-\infty}^{\infty}\frac{c_i k}{t^*_{i+k}-t^*_i}B_i^{k-1}\left(x,t^*\right)-\sum_{i=-\infty}^{\infty}\frac{c_i k}{t^*_{i+k+1}-t^*_{i+1}}B_{i+1}^{k-1}\left(x,t^*\right) \quad \text{C-6}$$

Changing the indices on the second term on the RHS:

$$\frac{d}{dx}\sum_{i=-\infty}^{\infty} c_i B_i^k\left(x,t^*\right)=\sum_{i=-\infty}^{\infty}\frac{c_i k}{t^*_{i+k}-t^*_i}B_i^{k-1}\left(x,t^*\right)-\sum_{i=-\infty}^{\infty}\frac{c_{i-1} k}{t^*_{i+k}-t^*_i}B_i^{k-1}\left(x,t^*\right) \quad \text{C-7}$$

Factoring out the $B_i^{k-1}\left(x,t^*\right)$ term and combing the sum terms:

$$\frac{d}{dx}\sum_{i=-\infty}^{\infty} c_i B_i^k\left(x,t^*\right)=\sum_{i=-\infty}^{\infty}\left[\frac{c_i k}{t^*_{i+k}-t^*_i}-\frac{c_{i-1} k}{t^*_{i+k}-t^*_i}\right]B_i^{k-1}\left(x,t^*\right) \quad \text{C-8}$$

Simplifying:

$$\frac{d}{dx}\sum_{i=-\infty}^{\infty} c_i B_i^k\left(x,t^*\right)=\sum_{i=-\infty}^{\infty} k\frac{c_i-c_{i-1}}{t^*_{i+k}-t^*_i}B_i^{k-1}\left(x,t^*\right) \quad \text{C-9}$$

We can define:

$$d_i=k\frac{c_i-c_{i-1}}{t^*_{i+k}-t^*_i} \quad \text{C-10}$$

And substitute into Eq. C9:

$$\frac{d}{dx}\sum_{i=-\infty}^{\infty} c_i B_i^k\left(x,t^*\right)=\sum_{i=-\infty}^{\infty} d_i B_i^{k-1}\left(x,t^*\right) \quad \text{C-11}$$

Which gives us the derivative identity.

To derive the integral identity, we first separate the variables:

$$d\sum_{i=-\infty}^{\infty} c_i B_i^k\left(x,t^*\right)=\sum_{i=-\infty}^{\infty} d_i B_i^{k-1}\left(x,t^*\right)dx \quad \text{C-12}$$

And then set up a definite integral:

$$\sum_{i=-\infty}^{\infty} c_i B_i^k\left(x,t^*\right)=\int_{-\infty}^{x}\sum_{i=-\infty}^{\infty} d_i B_i^{k-1}\left(x,t^*\right)dx \quad \text{C-13}$$

Adjusting the indices:

$$\sum_{i=-\infty}^{\infty} c_i B_i^{k+1}\left(x,t^*\right)=\int_{-\infty}^{x}\sum_{i=-\infty}^{\infty} d_i B_i^{k}\left(x,t^*\right)dx \quad \text{C-14}$$

Using Eq. C-10 we adjust the index and solve for c_i:

$$c_i = d_i\frac{t^*_{i+k+1}-t^*_i}{k+1}+c_{i-1}=\frac{1}{k+1}\sum_{j=-\infty}^{i} d_i\left(t^*_{j+k+1}-t^*_j\right) \quad \text{C-15}$$

For clarity to make the integral coefficient distinct from the derivative coefficient, we re-write the coefficients using c_i to describe the known coefficients and e_i to describe the integral coefficient (that is, $c_i \rightarrow e_i$ and $d_i \rightarrow c_i$):

$$e_i=\frac{1}{k+1}\sum_{j=-\infty}^{i} c_i\left(t^*_{j+k+1}-t^*_j\right) \quad \text{C-16}$$

And we note that:

$$e_1=0 \quad \text{C-17}$$

Substituting Eq. C-16 into Eq. C-14 and switching the LHS and RHS:

$$\int_{-\infty}^{x} \sum_{i=-\infty}^{\infty} c_i B_i^k \left(x, t^*\right) dx = \sum_{i=-\infty}^{\infty} e_i B_i^{k+1} \left(x, t^*\right) = \quad \text{..... C-18}$$

Which is the integral identity.

APPENDIX D

DERIVATION OF HUBER PROBABILITY DENSITY FUNCTION

A probability density function (PDF) represents an unknown variate, or random variable. For a density function $f(x)$, the cumulative distribution function (CDF) is given by:

$$F(x) = \int_{-\infty}^{x} f(z)\,dz \qquad \text{D-1}$$

If the upper limit of the integral in Eq. D-1 is infinity, then by definition $F(x) = 1$.

$$F(x) = \int_{-\infty}^{\infty} f(x)\,dx \equiv 1 \qquad \text{D-2}$$

D-2 must hold for any density function to be a valid density function. Given the piecewise Huber loss function, illustrated in **Fig. D-1**:

$$J_H\left(x,\mu,\sigma,t\right) = \begin{cases} \dfrac{\left(x-\mu\right)^2}{2\sigma^2} & \dfrac{|x-\mu|}{\sigma} \le t \\ t\dfrac{|x-\mu|}{\sigma} - \dfrac{t^2}{2} & \dfrac{|x-\mu|}{\sigma} > t \end{cases} \qquad \text{D-3}$$

If we exponentiate Eq. D-3 we obtain a function that *looks* like a density function; that is, the maximum value lies at μ and the tails approach zero. However, we require that:

$$\frac{1}{g}\int_{-\infty}^{\infty} \exp\left[-J_H(x,\mu,\sigma,t)\right]dx \equiv 1 \qquad \text{D-4}$$

This may not be true without some constant, g, to normalize the integral to unity. Our goal in this appendix is to seek the normalizing constant and yield a valid density function. Setting up a definite integral for Eq. D-3 with an unknown normalizing constant:

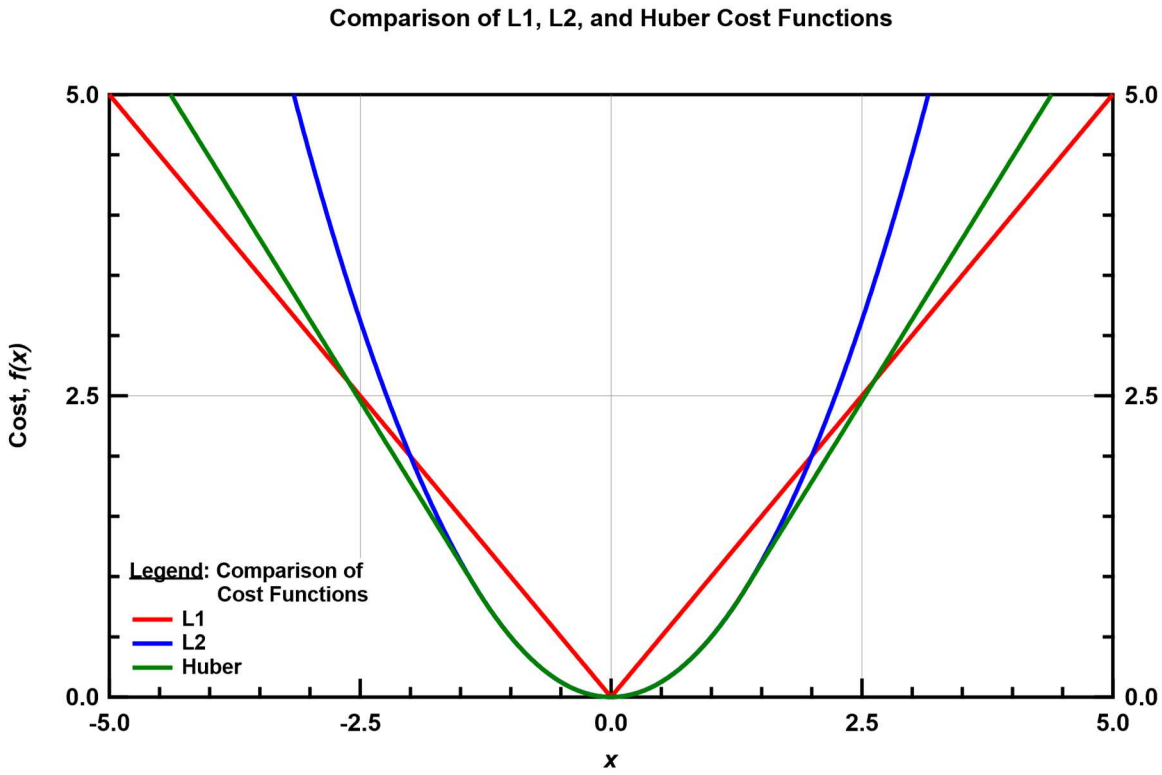

Figure D-1 — Comparison of L1, L2, and Huber Cost Functions.

$$F_H(x) = \frac{1}{b}\int_0^x \exp\left[J_H(x,\mu,\sigma,t)\right]dx = \frac{1}{g}\begin{cases} \int_{-\infty}^{\infty} \exp\left[-\dfrac{(x-\mu)^2}{2\sigma^2}\right]dx & \dfrac{|x-\mu|}{\sigma} \le t \\ \int_{-\infty}^{\infty} \exp\left[-t\dfrac{|x-\mu|}{\sigma} + \dfrac{t^2}{2}\right]dx & \dfrac{|x-\mu|}{\sigma} > t \end{cases} \quad \text{.....D-5}$$

We evaluate the first term, FT. This term is defined in the middle, and is symmetrical, so we simplify the evaluation with the integral limits of zero and *x*:

$$\text{FT} = \int_0^x \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]dx \qquad \frac{|x-\mu|}{\sigma} \le t \quad \text{.....D-6}$$

Defining a variable of substitution:

$$u = \frac{x-\mu}{\sigma\sqrt{2}} \quad \text{.....D-7}$$

And the derivative would be:

$$du = \frac{1}{\sigma\sqrt{2}}dx \quad \text{.....D-8}$$

Or:

$$dx = \sigma\sqrt{2}du \quad \text{.....D-9}$$

And the limits of the integral become:

$$x \to 0 \qquad u \to 0$$

$$x \to x \qquad u \to \frac{x-\mu}{\sigma\sqrt{2}}$$

Substituting Eq. D-7 and Eq. D-9 into Eq. D-6 and changing the limits of the integral:

$$\mathrm{FT} = \sigma\sqrt{2}\int_0^{\frac{x-\mu}{\sigma\sqrt{2}}} \exp\left[-u^2\right]du \quad \text{.....D-10}$$

And with some manipulation we achieve:

$$\mathrm{FT} = \frac{\sqrt{2\pi}}{\sqrt{2\pi}} \times \sigma\sqrt{2}\int_0^{\frac{x-\mu}{\sigma\sqrt{2}}} \exp\left[-u^2\right]du$$

$$\mathrm{FT} = \sqrt{\frac{\pi}{2}}\sigma \times \frac{2}{\sqrt{\pi}}\int_0^{\frac{x-\mu}{\sigma\sqrt{2}}} \exp\left[-u^2\right]du \quad \text{.....D-11}$$

And we immediately note that Eq. D-11 contains the error function:

$$\mathrm{erf}(u) = \frac{2}{\sqrt{\pi}}\int_0^{x} \exp\left[-u^2\right]du \quad \text{.....D-12}$$

And by definition the second term in Eq. D-12 equals one. Substituting Eq. D-12 and Eq. D-7 into Eq. D-11 provides:

$$\mathrm{FT} = \sqrt{\frac{\pi}{2}}\sigma \,\mathrm{erf}\left[\frac{x-\mu}{\sigma\sqrt{2}}\right] \qquad \frac{|x-\mu|}{\sigma} > t \qquad \text{.....D-13}$$

Next, we evaluate the second term:

$$\mathrm{ST} = \int_{-\infty}^{0} \exp\left[-t\frac{|x-\mu|}{\sigma} + \frac{t^2}{2}\right] dx \qquad \text{.....D-14}$$

Evaluating the integral:

$$\mathrm{ST} = \exp\left[t\frac{|x-\mu|}{\sigma} + \frac{t^2}{2}\right]\frac{\sigma}{t}\Bigg|_{-\infty}^{0} \qquad \text{.....D-15}$$

$$\mathrm{ST} = -\exp\left[-t\frac{|x-\mu|}{\sigma} + \frac{t^2}{2}\right]\frac{\sigma}{t}\Bigg|_{0}^{\infty} \qquad \text{.....D-16}$$

Combining Eq. D-13, Eq. D-15, and Eq. D-16 into a piecewise function where the constant at the start of each piece is the evaluation of the prior piece's integral:

$$\frac{1}{g}\int_0^x \exp\left[-J_H(x,\mu,\sigma,t)\right]dx = \frac{1}{g}\begin{cases} \dfrac{\sigma}{t}\exp\left[t\dfrac{|x-\mu|}{\sigma} + \dfrac{t^2}{2}\right] & \dfrac{x-\mu}{\sigma} < -t \\ \sqrt{\dfrac{\pi}{2}}\sigma\,\mathrm{erf}\left[\dfrac{x-\mu}{\sigma\sqrt{2}}\right] + c_1 & -t < \dfrac{x-\mu}{\sigma} < t \\ -\dfrac{\sigma}{t}\exp\left[-t\dfrac{|x-\mu|}{\sigma} + \dfrac{t^2}{2}\right] + c_2 & \dfrac{x-\mu}{\sigma} < t \end{cases} \qquad \text{.....D-17}$$

Recalling that we require the integral to be equal to one, we next must determine *g*. Given that the integral is symmetrical, we need only evaluate from negative infinity to zero:

$$\int_{-\infty}^{\infty} f(x)\,dx = 2\int_{-\infty}^{0} f(x)\,dx$$

Thus, solving for g in Eq. D-17 for $x = 0$, we have:

$$g = 2\left[\sqrt{\frac{\pi}{2}}\sigma\left(\ \mathrm{erf}\left[\frac{-\mu}{\sigma\sqrt{2}}\right] - \mathrm{erf}\left[\frac{-t-\mu}{\sigma\sqrt{2}}\right]\right) + \frac{\sigma}{t}\exp\left[t\frac{|\mu|}{\sigma} + \frac{t^2}{2}\right]\right]$$

Or noting that:

$$\mathrm{erf}\left[c\right] = \mathrm{erf}\left[0\right] - \mathrm{erf}\left[-c\right]$$

We have:

$$g = 2\left[\sqrt{\frac{\pi}{2}}\,\sigma\mathrm{erf}\left[\frac{t-\mu}{\sigma\sqrt{2}}\right] + \frac{\sigma}{t}\exp\left[t\frac{|\mu|}{\sigma} + \frac{t^2}{2}\right]\right] \quad \text{.....D-18}$$

If we define:

$$z = \frac{|x-\mu|}{\sigma} \quad \text{.....D-19}$$

Then we can simplify Eq. D-17 to:

$$\frac{1}{g}\int_0^x \exp\left[-J_H(x,t)\right]dx = \frac{1}{g}\begin{cases} \dfrac{\sigma}{t}\exp\left[t\,z + \dfrac{t^2}{2}\right] & z < -t \\ \sqrt{\dfrac{\pi}{2}}\sigma\ \mathrm{erf}\left[\dfrac{z}{\sqrt{2}}\right] + c_1 & -t < z < t \\ -\dfrac{\sigma}{t}\exp\left[-t\,z + \dfrac{t^2}{2}\right] + c_2 & z < t \end{cases} \quad \text{.....D-21}$$

And Eq. D-18 to:

$$g = 2\left[\sqrt{\frac{\pi}{2}}\sigma\ \mathrm{erf}\left[\frac{t}{\sqrt{2}}\right] + \frac{\sigma}{t}\exp\left[-\frac{t^2}{2}\right]\ \right] \quad \text{.....D-22}$$

We define:

$$\Phi^{L1} = \frac{\sigma}{t}\exp\left[-\frac{t^2}{2}\right] \quad \text{D-23}$$

$$\Phi^{L2} = \sqrt{\frac{\pi}{2}}\sigma\,\mathrm{erf}\left[\frac{t}{\sqrt{2}}\right] \quad \text{D-24}$$

And rewrite Eq. D-22 as:

$$g = 2\left[\Phi^{L1} + \Phi^{L2}\right] \quad \text{D-25}$$

Finally, we substitute Eq. D-25 into Eq. D-4:

$$\frac{1}{2\left[\Phi^{L1} + \Phi^{L2}\right]}\int_{-\infty}^{\infty}\exp\left[-J_H(x,\mu,\sigma,t)\right]dx \equiv 1 \quad \text{D-26}$$

Summarizing our results, the Huber density function is:

$$f_H(x \mid \mu,\sigma,t) = \frac{1}{2\left[\Phi^{L1} + \Phi^{L2}\right]}\exp\left[-J_H(x,\mu,\sigma,t)\right] \quad \text{D-27}$$

As the Huber density function is a mix of the L1 and L2 functions, it can re-create the behavior of either depending on the value of the t parameter (**Fig. D-2**). It is worth noting that when $t \ll 1$, i.e. the L1 behavior dominates and the value of σ is scaled by t. This results from t only appearing in the piecewise term when $z < |t|$.

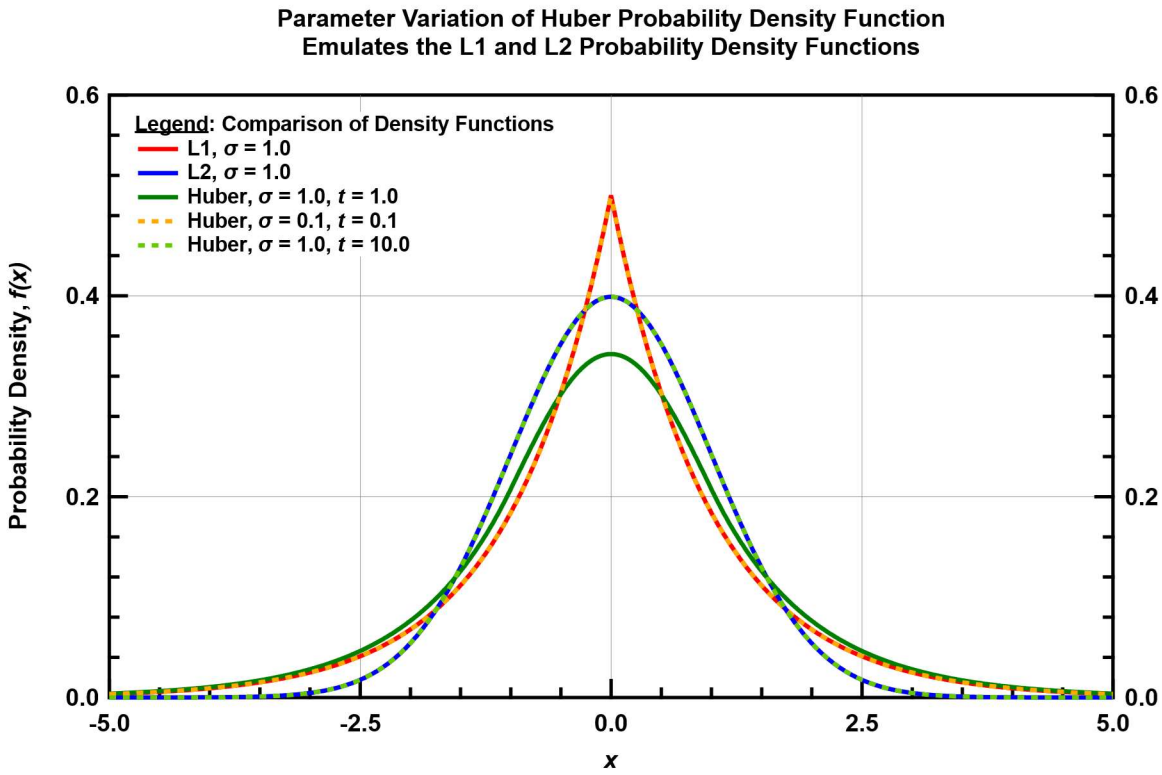

Figure D-2 — Comparison of L1, L2, and Huber Density Functions.

APPENDIX E

IMPLEMENTION OF ILK, VALKÓ, AND BLASINGAME DECONVOLUTION METHOD USING BAYESIAN PROBABILISTIC GRAPHICAL MODEL

In this appendix we implement the Ilk, Valkó, and Blasingame (2005, 2006) deconvolution scheme using our Bayesian PGM. Although quite different in implementation, conceptually the two methods share a lot in common. The key differences are the following:

- Our method solves the inverse problem of deconvolution by a stochastic optimization scheme of the forward problem. We utilize a generating function for the values of the beta derivative of the constant-pressure rate and constant-rate pressure functions.
- The Ilk et al. method utilizes a direct solution to the inverse problem by setting up and solving a system of linear equations. The solution is the column vector of coefficients for the spline function that evaluates to the derivative of the pressure function.

Put another way, both methods represent a pressure derivative with the B-spline function given in Eq. 3.68:

$$\mathbf{S}(x, t^*) = \mathbf{B}^k \mathbf{c} \quad \text{.....} \quad 3.68$$

The evaluation of the spline function, $S(x, t^*)$, is a pressure derivative. As we use the beta-derivative, we define $S(x, t^*)$ as the beta-derivative of the pressure function:

$$\beta(t) \equiv \frac{d \ln \Delta p(t)}{d \ln t} = \frac{t}{\Delta p(t)} \Delta p'(t) \quad \text{.....A-38}$$

substitution of Eq. 3.68 into Eq. A-38 yields Eq. 3.89:

$$\Delta p_{cr}(t) = \Delta p_1 \exp \int_{t_i}^{t} \mathbf{B}^k \boldsymbol{\beta}_\mathbf{p} \quad \text{.....3.89}$$

Where $\mathbf{c} = \boldsymbol{\beta}_\mathbf{p}$ for a degree 1 B-spline. For clarity, $\mathbf{B}^k$ is a B-spline that interpolates from our evenly log-spaced knots to our evenly-cartesian spaced data time series. Re-arranging and integrating both sides of Eq. A-38:

$$\Delta p(t) = \exp \int_0^{\ln t} \beta(t)\, d\tau \quad \text{..... E-1}$$

We observe that we must integrate with respect to the log of time, and exponentiate the result, as shown in Eq. 3.89. Recalling that we can simultaneously interpolate and integrate with our B-spline using the identity shown in **Appendix C**, we may re-write Eq. E-1 as:

$$\Delta p(t) = \exp\left[\mathbf{B}^{k+1}\mathbf{e}\right] \quad \text{..... E-2}$$

Our method directly generates the values of $S(x, t^*)$ by random sampling via a Gaussian process, which are also the values of the beta derivative and the values of the coefficients, and integrates the result to obtain the pressure function.

The Ilk et al. method differs solves the Eq. 3.68 for $\boldsymbol{c}$, where $S(x, t^*)$ is defined as the derivative of the pressure function. This is given by Eq. 7 in Ilk et al. as:

$$\Delta p'(t) = S\left(\mathbf{x}, t^*\right) = \mathbf{B}^2\mathbf{c} \quad \text{.....Ilk, Valkó, and Blasingame (2006), E-3}$$

Once the coefficients have been found by solution of the system of linear equations, the pressure function is evaluated by use of the B-spline integral identity.

To modify our algorithm to implement the approach of Ilk et al, we perform two simple changes:

- We generate the coefficients of the spline function as opposed to the values to which the spline function evaluates.
- We integrate the B-spline with respect to time and do not exponentiate the result.

A comparison of our method, and the implementation of the Ilk et al. method using our Bayesian scheme and regularization by curvature, follows below. Examples from the validation cases in section four were selected for comparison. As Ilk et al. did not propose any scheme for generation of the rate function during variable-rate deconvolution, we do not generate a rate function for our comparison. For comparison, we use the variable-rate cases with 0%, 20%, and 60% random error, and the random rate cases with 0%, 20%, and 60% random error.

In general, we observe from comparison of our results with those of Ilk et al. that we achieve significantly greater stability of our pressure derivatives. This is due in part to the inherent stability of the convolution operator as opposed to the instability of deconvolution.

Furthermore, we observe that our use of the beta-derivative yields more stable results than the Ilk et al. use of the coefficients of the pressure derivative. We attribute this to two factors. First, the beta-derivative is invariant to scale. This means that our solution is not sensitive to any initial value of rate or pressure. Second, the beta-derivative is constrained

to a small range of values, ideally in the domain of (0, 1) for a pressure function. A value of zero indicates radial flow, while a value of unity indicates pseudosteady-state. A large number of commonly observed flow regimes fall within this range. This means that our problem is almost always of order 1, which reduces the difficulty of the problem for the optimization scheme.

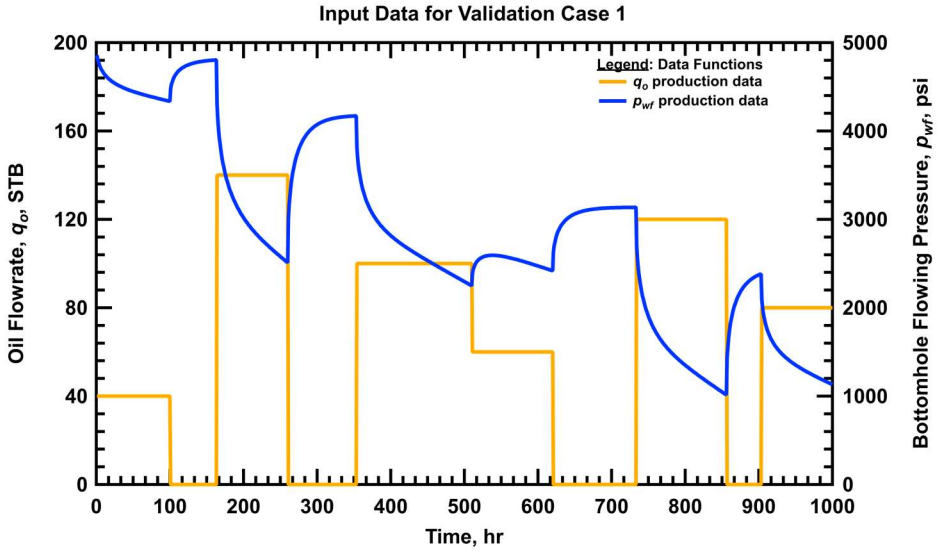

Figure E-1 — Input data for Validation Case E-1 (variable rate, no error).

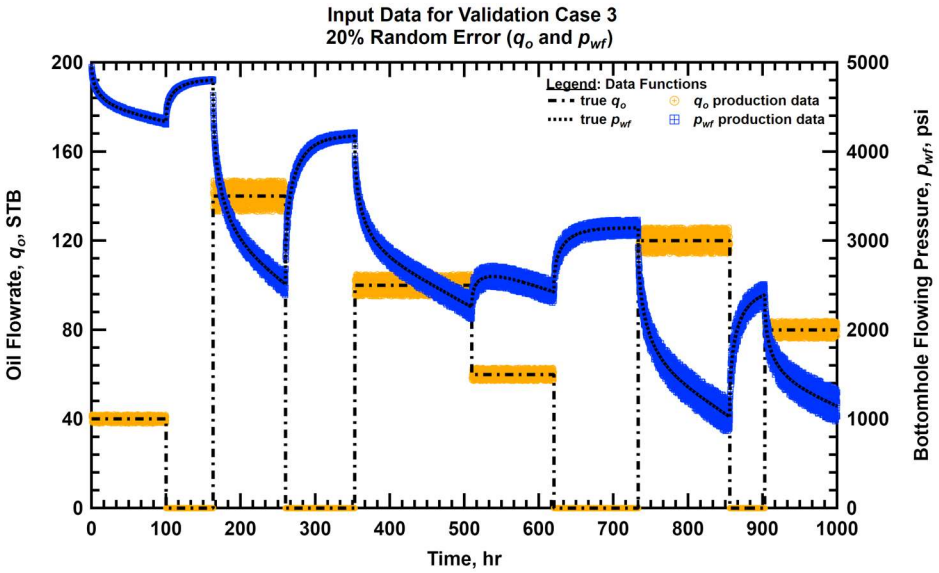

Figure E-2 — Input data for Validation Case E-2 (variable rate, 20% random error).

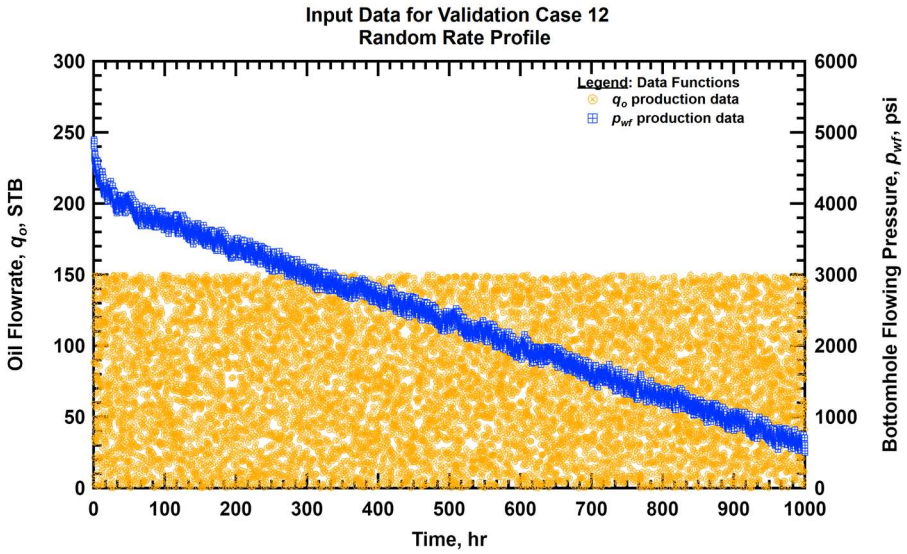

Figure E-3 — Input data for Validation Case E-3 (random rate, no error).

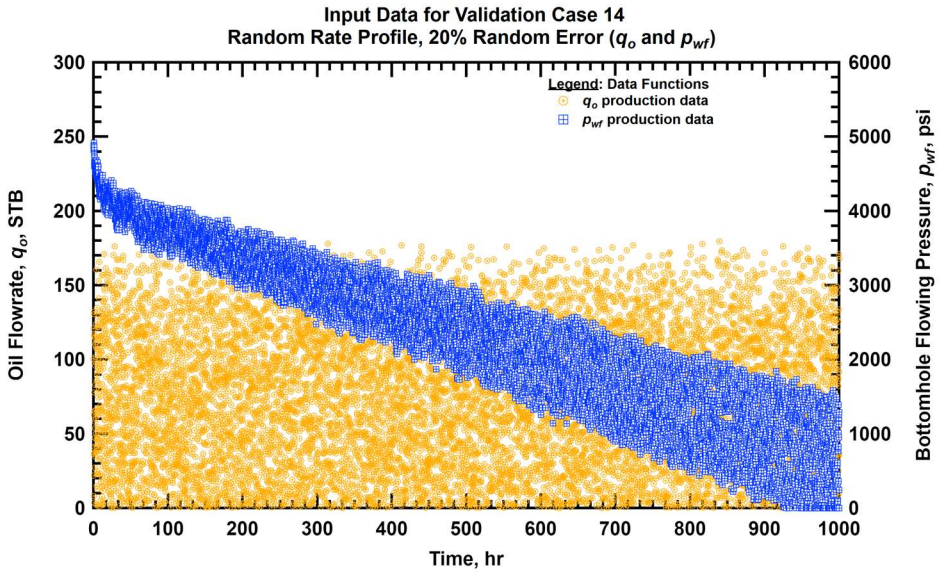

Figure E-4 — Input data for Validation Case E-4 (random rate, 20% random error).

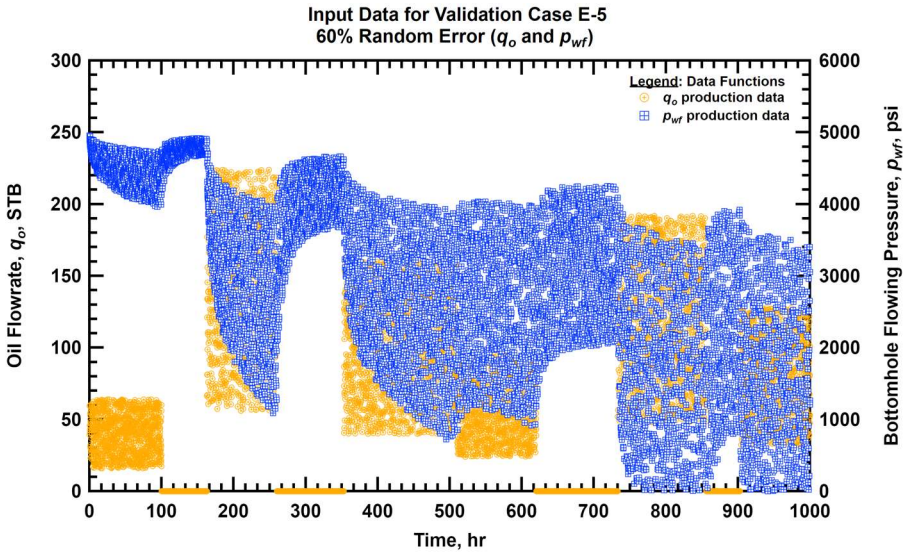

Figure E-5 — Input data for Validation Case E-5 (variable rate, 60% random error).

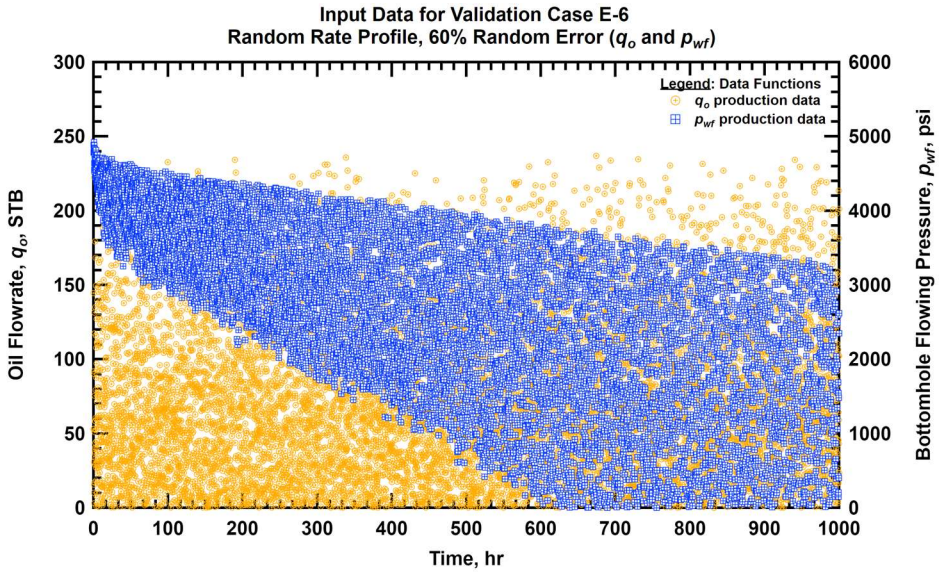

Figure E-6 — Input data for Validation Case E-6 (random rate, 60% random error).

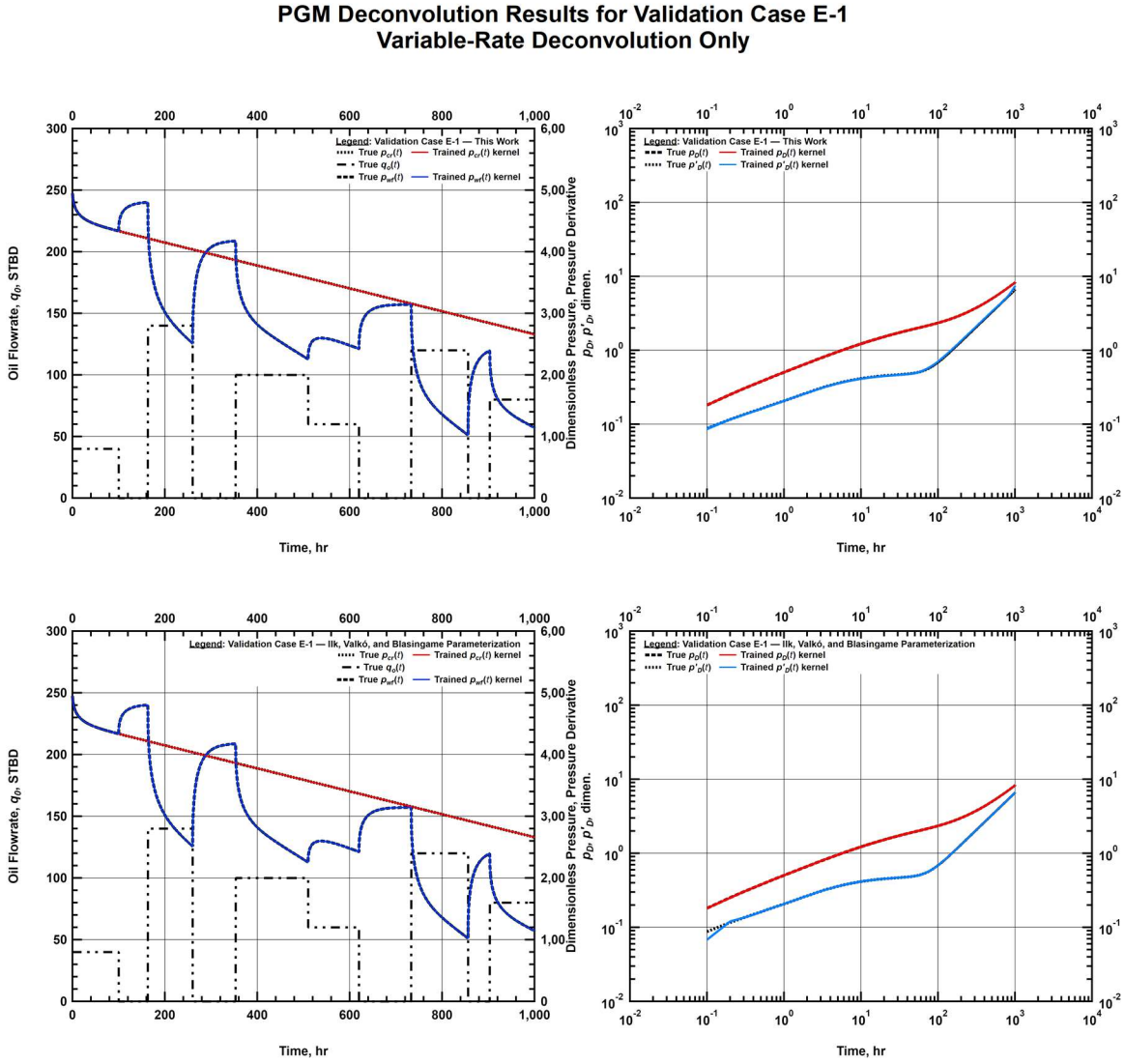

Figure E-7 — Deconvolution results for Validation Case E-1 (variable rate, no error).

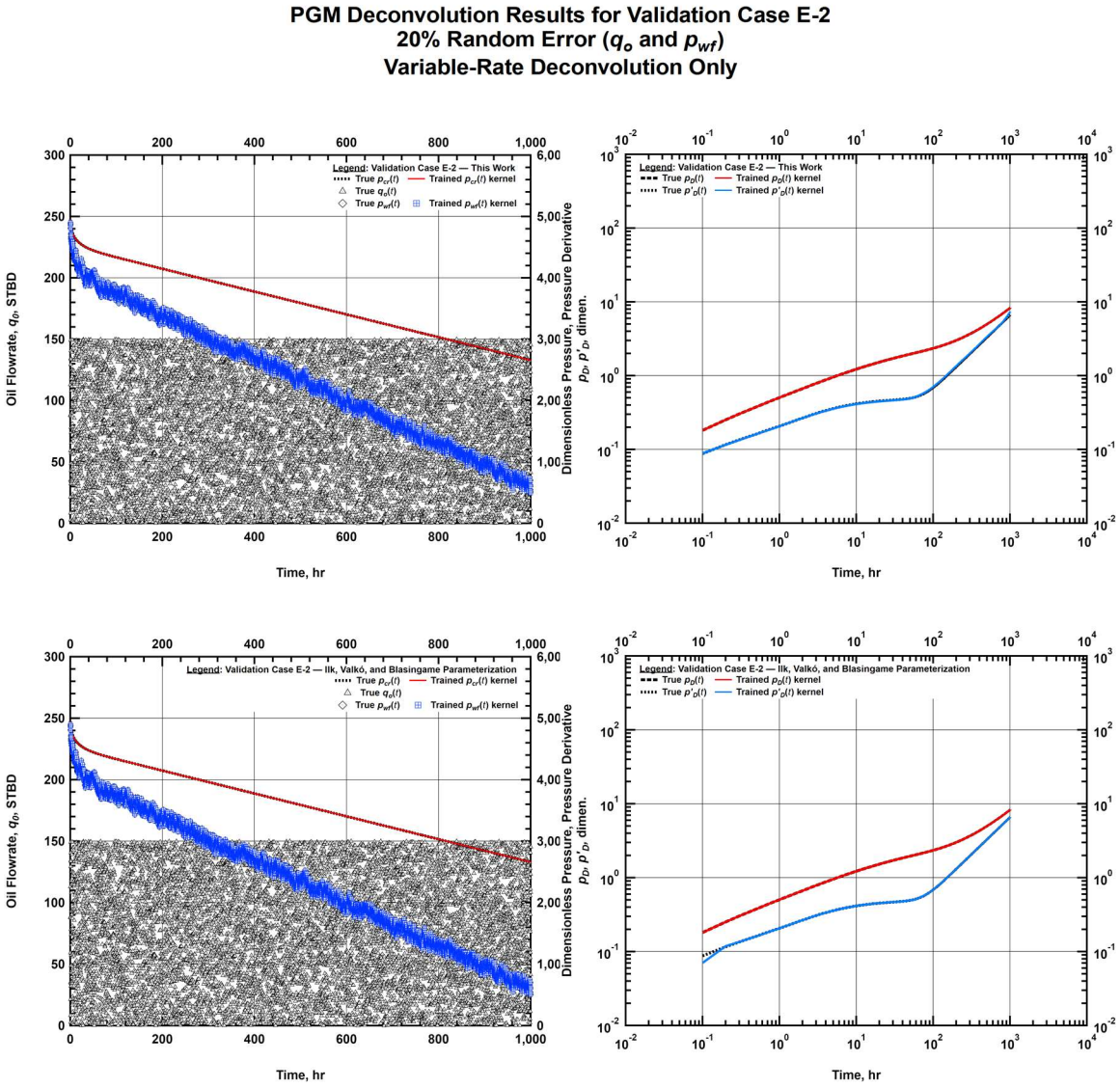

Figure E-8 — Deconvolution results for Validation Case E-2 (random rate, no error).

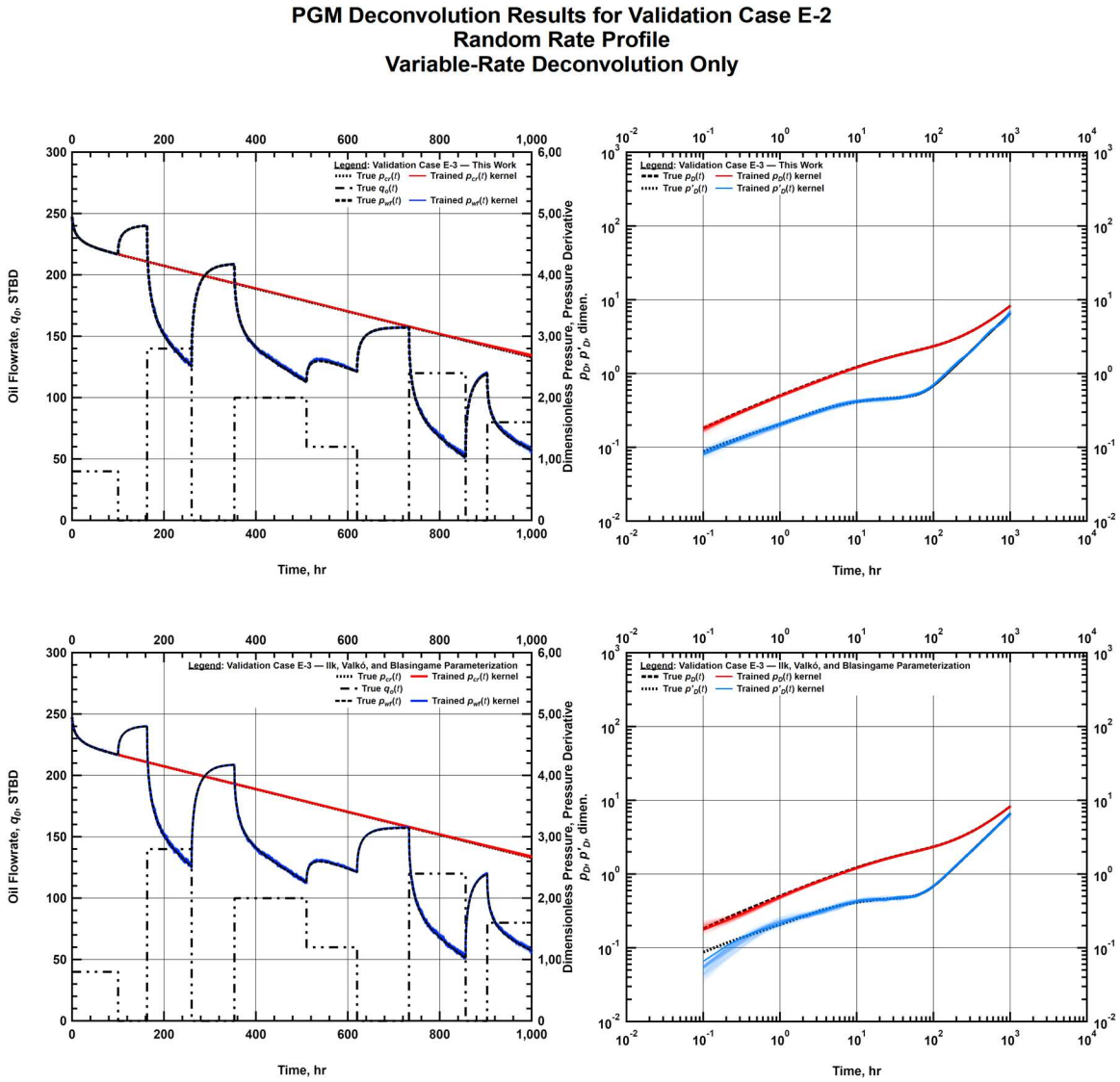

Figure E-9 — Deconvolution results for Validation Case E-3 (variable rate, 20% random error).

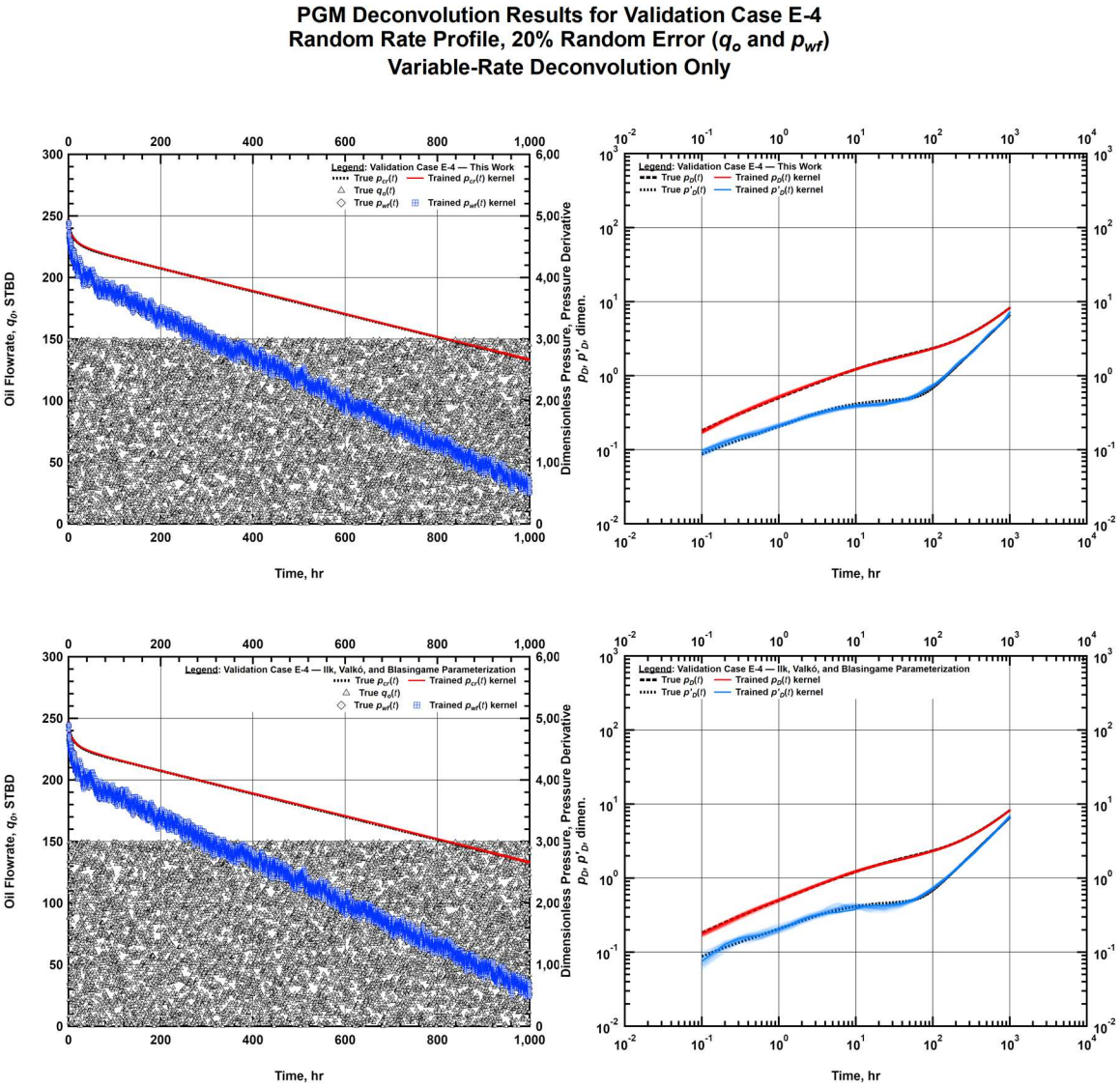

Figure E-10— Deconvolution results for Validation Case E-4 (random rate, 20% random error).

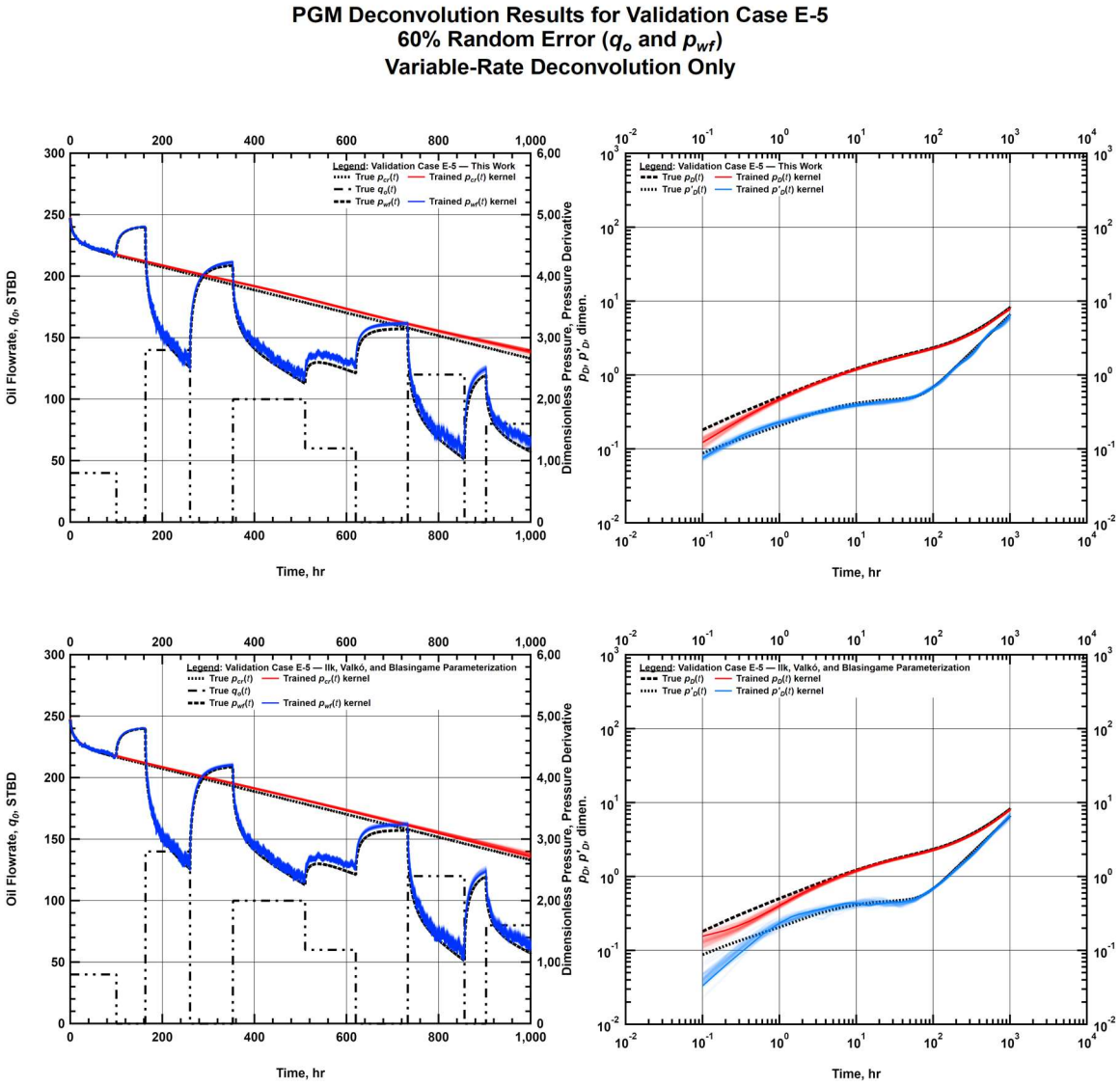

Figure E-11— Deconvolution results for Validation Case E-5 (variable rate, 60% random error).

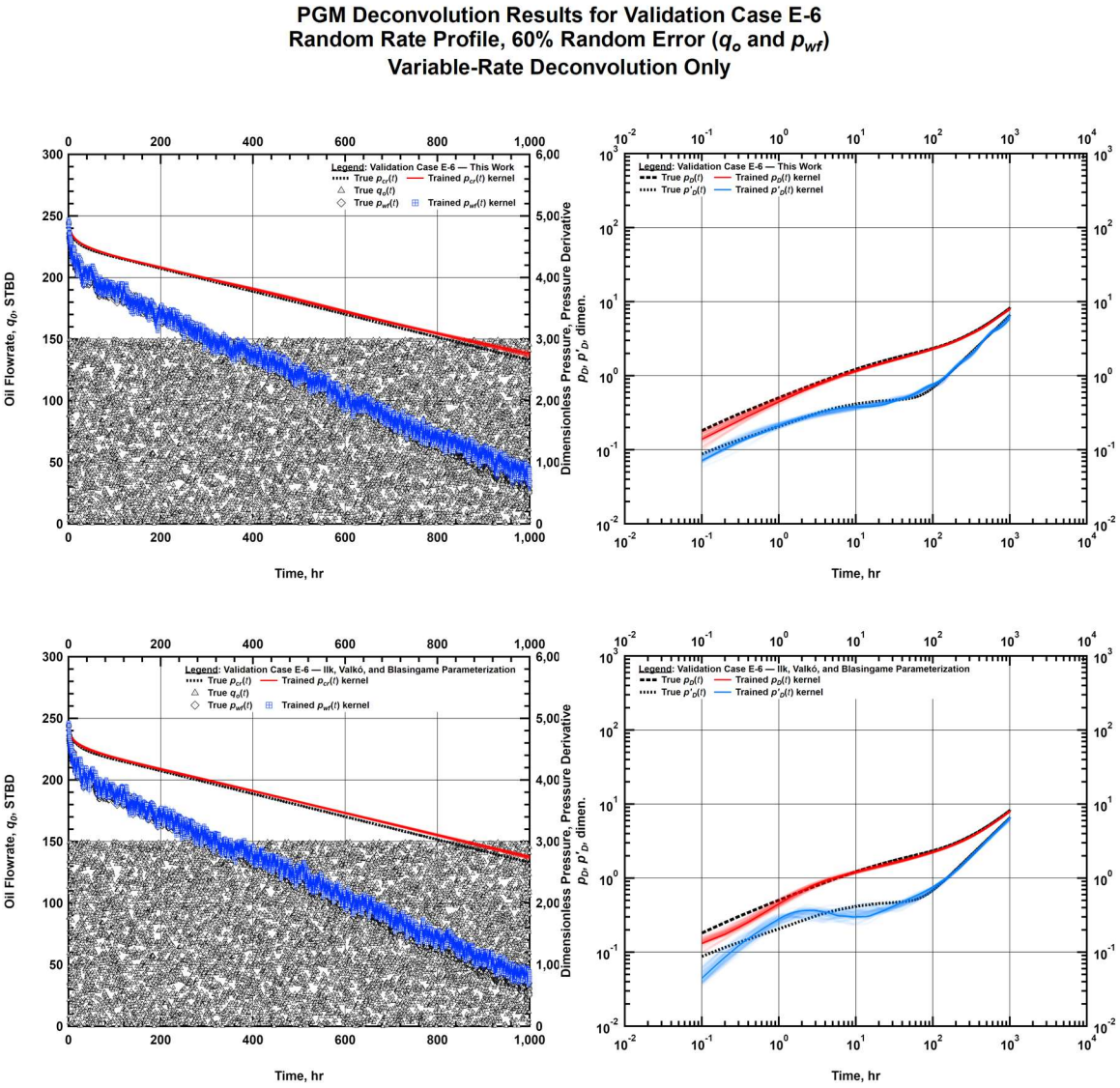

Figure E-12— Deconvolution results for Validation Case E-6 (random rate, 60% random error).