Dynamic Fault Tree Analysis with PEWMA Modeling of Event Rates and Bayesian Updating of Material Balance Parameters using MCMC

By

Alf Christian Johansen

A Thesis submitted to the School of Graduate Studies

In partial fulfillment of the requirements for the degree of Master of Engineering

Faculty of Engineering and Applied Science

Department of Process Engineering

Memorial University of Newfoundland

St. John's, Newfoundland and Labrador, Canada

May 2016

Abstract

This research explores Bayesian updating as a tool for estimating parameters probabilistically by dynamic analysis of data sequences. Two distinct Bayesian updating methodologies are assessed. The first approach focuses on Bayesian updating of failure rates for primary events in fault trees. A Poisson Exponentially Moving Average (PEWMA) model is implemnented to carry out Bayesian updating of failure rates for individual primary events in the fault tree. To provide a basis for testing of the PEWMA model, a fault tree is developed based on the Texas City Refinery incident which occurred in 2005. A qualitative fault tree analysis is then carried out to obtain a logical expression for the top event. A dynamic Fault Tree analysis is carried out by evaluating the top event probability at each Bayesian updating step by Monte Carlo sampling from posterior failure rate distributions. It is demonstrated that PEWMA modeling is advantageous over conventional conjugate Poisson-Gamma updating techniques when failure data is collected over long time spans. The second approach focuses on Bayesian updating of parameters in non-linear forward models. Specifically, the technique is applied to the hydrocarbon material balance equation. In order to test the accuracy of the implemented Bayesian updating models, a synthetic data set is developed using the Eclipse reservoir simulator. Both structured grid and MCMC sampling based solution techniques are implemented and are shown to model the synthetic data set with good accuracy. Furthermore, a graphical analysis shows that the implemented MCMC model displays good convergence properties. A case study demonstrates that Likelihood variance affects the rate at which the posterior assimilates information from the measured data sequence. Error in the measured data significantly affects the accuracy of the posterior parameter distributions. Increasing the

likelihood variance mitigates random measurement errors, but casuses the overall variance of the posterior to increase. Bayesian updating is shown to be advantageous over deterministic regression techniques as it allows for incorporation of prior belief and full modeling uncertainty over the parameter ranges. As such, the Bayesian approach to estimation of parameters in the material balance equation shows utility for incorporation into reservoir engineering workflows.

Acknowledgements

I would like to thank my supervisor Dr. Faisal Khan for sharing his enthusiasm for this reasearch area and for providing guidance throughout my studies. I would also like to thank Selina Boland and my dad Thormod Johansen for supporting me throughout this work.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Chapter 2 - Symbols, Nomenclature and Abbreviations

PEWMA	Poisson Exponentially Weighted Moving Average
y_t	number of observed Poisson counts at time t
μ	Expected number of Poisson counts
n	number of count data points
α	Gamma shape parameter
β	Gamma scale parameter
ɾ	Gamma function
ω	PEWMA discounting factor
VC	Vapor cloud forms outside blowdown drum
B	Blowdown drum overfills
N	Release of vapor cloud from blowdown drum
S	Release of vapor cloud from sewer
R	Raffinate splitter tower overfills
O	Operator unaware of raffinate splitter tower liquid level
O1	Operator not following regulations
O2	Operator unaware of raffinate liquid level due to alarm failure
O3	Operator unaware of raffinate liquid level due to lack of training
M	Maintenance failure (sight glass)
V	Raffinate splitter tower blowdown valve fails closed
L	Raffinate splitter tower level indicator fails to function
LS	Raffinate splitter tower level indicator alarm system fails

A	Raffinate splitter tower high level alarm system fails
A1	Alarm associated with level indicator fails to function
A2	Raffinate splitter tower hardwired alarm fails to function
SA	Severe liquid level alarm failure

Chapter 3 - Symbols, Nomenclature and Abbreviations

W_e	Aquifer influx
p_a	Aquifer pressure
W_a	Aquifer volume
W_i	Initial aquifer volume
c_t	Total Compressibility
c_w	Water Compressibility
c_r	Rock Compressibility
n_o,n_w,n_g	Corey Exponents (oil, water and gas)
k_{ro},k_{rw},k_{rg}	Relative permabilities (oil, water and gas)
$k_{ro,max},k_{rw,max},k_{rg,max}$	Maximum Relative permabilities (oil,water, gas)
S_o,S_w,S_g	Phase saturations (oil, water, gas)
S_{wc},S_{gc}	Connate phase saturations (water,gas)
S_{gt}	Threshold gas saturation

S_{or}	Residal oil saturation
x	Parameters
y	Observed data
f(y|x)	Posterior distribution
f(y|x)	Likelihood function
f(x)	Prior distribution
f(y)	Probability of observed data
C_x	Gaussian prior covariance matrix
C_y	Likelihood function covariance matrix
σ	Standard deviation
g(x)	Forward model
ϵ	Likelihood error/variance
π(x)	Markov Chain target distribution
X_t	Markov Chain
T(a, b)	Finite state space transition matrix
$P(X_{t+1}\|X_t)$	General state space transition kernel
S	State Space

α_{ij}	MCMC Acceptance Ratio
MCMC	Markov Chain Monte Carlo
PVT	Pressure Volume Temperature
GOR	Gas Oil Ratio
ρ_{XY}	Pearson′s correlation coefficient
PV_i	Initial Pore Volume
PV	Pore Volume
B_{oi}	Initial Oil Formation Volume Factor
B_o	Oil Formation Volume Factor
N	Stock Tank Oil Originally in Place
W_{res}	Stock tank water in place
G_p	Incremental Gas produced
R_s	Solution GOR
W_e	Incremental Aquifer Influx
N_p	Incremental Oil Production
B_g	Gas Formation Volume Factor
B_w	Water Formation Volume Factor

c_f	Formation Compressibility
c_w	Water compressibility
p_r	Reservoir pressure
p_a	Aquifer pressure
J_w	Aquifer index

Appendices

Appendix A - PEWMA Python Scripts

Appendix B – Bayesian Material Balance Python Scripts

Appendix C - PVT Correlations

Appendix D - Procedure for Generating Random Simulation Model Properties

Appendix E - Capillary Pressure Model

Appendix F - Corey Functions for Relative Permeability

Appendix G - Eclipse input file

1 THESIS OVERVIEW

1.1 Organization of Thesis

The Bayesian updating methodology is here applied to two distinct types of parameter estimation problems. Chapter 2 presents an application of Bayesian updating to probabilistic Fault Tree Analysis, while Chapter 3 explores Bayesian updating in the context of probabilistic parameter fitting. Due to the uniqueness of each methodology, Chapter 2 and Chapter 3 are organized as independent chapters with separate introductions, research objectives, literature reviews, conclusions and reference lists.

1.2 Relevance of this Research

This research builds on previous work within the field of Bayesian updating and further establishes this statistical techqniue as viable engineering tool for reducing uncertainty by sequentially assimilating model parameters to measured data. Chapter 2 demonstrates how a fault tree can be evaluated dynamically by incorporating accident precursor data into to a Bayesian updating framework. Despite its utility in modeling long term failure data and simplicity of implementation, PEWMA appears to be underutilized for dynamic risk assessments. This work therefore seeks to further establish probabilistic fault tree analysis with PEWMA updating for event rates as a viable technique for dynamic risk assessments.

Chapter 3 demonstrates a more general and non-linear/multivariate Bayesian updating technique, which is applied to estimate parameters in the hydrocarbon material balance equation by assimilating measured reservoir pressure data. Little work has been carried out to model material balance parameters in a fully probabilistic manner. This work seeks to further establish Bayesian material balance as a viable statistical technique for application in reservoir engineering workflows.

2 DYNAMIC FAULT TREE ANALYSIS WITH PEWMA MODELING OF EVENT RATES

2.1 Introduction

In this section Bayesian updating is explored in the context of probabilistic fault tree analysis and Bayesian updating of event failure rates. A review of relevant background theory on Quantitative Risk Assessment (QRA), Fault Tree Analysis (FTA), Bayesian updating and Poisson Exponentially Weighted Average (PEWMA) modeling of event rates is provided. A fault tree is developed based on the ISOM unit at the Texas City refinery incident. The resulting fault tree is evaluated qualitatively to generate a logic expression for the top event and is used identify safety improvements. PEWMA is implemented to model event failure rates as it is preferable over conventional conjugate Poisson-Gamma updating when accident precursor data is collected over long time spans. Real accident precursor data obtained from the Texas City accident reports is presented and used to model the event rates probabilistically with the PEWMA model. The fault tree top event probability is integrated through time by Monte Carlo sampling from posterior PEWMA event rates to provide a dynamic risk profile for the Texas City ISOM unit up until the time of the refinery incident.

2.2 Research Objectives

- Implement a Poisson Exponentially Weighted Moving Average (PEWMA) model for modeling Poisson event rates in a Bayesian framework.
- Develop a fault tree for the ISOM unit that caused the 2005 Texas City refinery incident.
- Evaluate the resulting fault tree qualitatively and analyze minimal cut sets to identity potential safety improvements.
- Collect accident precursor data from the published reports on the Texas City refinery incident and apply PEWMA to the resulting data set.
- Evaluate the developed fault tree probabilistically through time by Monte Carlo sampling of the posterior distributions resulting from the PEWMA model.

2.3 Literature Review and Background

2.3.1 Dynamic Risk Assessment

Khan and Abbasi (1998) presents a review of the available Quantitative Risk Assessment (QRA) tools. The most important techniques are Hazard and Operability Study (HAZOP), Failure Mode and Effect Analysis (FMEA), Event Tree Analysis (ETA) and Fault Tree Analysis (FTA). In recent years, several authors have studied Bayesian Networks (BN) as an alternative to Fault Tree modeling. The advantages of BN models is that are they are able to model non-exponential failure distributions, multi-state variables, noisy gates, common cause failures and simple sequentially dependent failures. Examples of applications of BN can be found in Bobbio et. al. (2001) and Marquez et. al. (2010). A more comprehensive analysis technique can be obtained by combining Event Trees and Fault Tere using a Bow-Tie (BT) technqiue, thereby allowing modeling accident scenarios from causes to effects. Examples of BT modeling are provided in Dianous VD et. al. (2006) and Khakzad et.al (2013). Generic failure rate data from publicly available data sources are often used to estimate static failure rates for primary events in a fault tree. An example of such a source is the Offshore Reliability Data Handbook (OREDA, 2002), which contains failure rates for valves and level indicators in terms of calendar time. Failure rates can in turn be evaluated as failure probabilities by using a stochastic model such as the Poisson process. QRA as a whole lacks the ability to model how risk levels are changing over time (Kalantarnia, 2011) and the disadvantage associated with using a generic source is that the resulting failure rates are static and not representative of the actual system. One of the first attempts at bridging this gap was made in the nuclear industry, where (Bier and Mosleh, 1990)

carried out a dynamic risk assessment by modeling system specific failure rates in a Bayesian updating framework. Bayesian updating is a technique that has later been re-visited by several authors. Shafaghi (2008) demonstrates how Bayesian statistics can be used to model the Poisson failure rate for individual process equipment units. Meel (2006) and Kalantarnia (2011) carry out integrated Bayesian analyses by modeling multiple plant components simultaneously. Khakzad et. al. (2013) shows that the BT technique combined with Bayesian updating using accident precursor data can be used to facilitate a dynamic risk assessment. A problem associated with incorporating plant specific accident precursor data into a Baysian updating framework is that abnormal events are often overlooked or unnoticed by operators due to underestimation of their adverse impacts (Meel, 2006). Another factor to consider is that the number of recorded incidents depends on the quality and frequency of safety audits and inspections. Common to most attempts at implementing Bayesian updating for dynamic risk assessments is that conjugate probability distributions are used. When applicable, conjugate distributions are desirable because they result in closed-form expressions that are not prone to the errors associated with sampling based techniques. As an example, the conjugate prior to the Poisson likelihood function is the gamma distribution. If a gamma prior distribution does not adequately characterize prior knowledge for a particular component, sampling based techniques are required (Thodi, 2010). Lindhe et. al. (2009) shows how probabilistic Fault Tree analysis can be used as part of risk-based decision making and uses a Monte Carlo techqniue for top event integration with all primary events modeled as random variables.

2.3.2 Fault Tree Analysis

Fault Tree Analysis (FTA) is a deductive, top-down technique used to determine the root causes leading to a defined failure event, often called a top event. It is a standardized technique with designated symbols for expressing events and logical interrelationships. The OR gate is used when the output event requires one or more of the input events occur, while the AND gate is used when the output event requires all input events to occur (Figure 1). A square box signifies events that are consequences resulting from AND/OR gates.

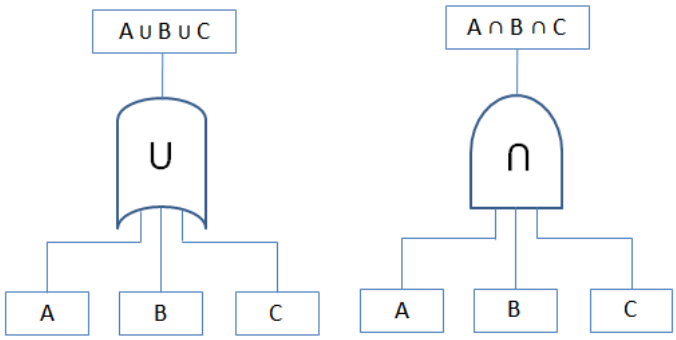


Figure 1 - Fault Tree Gate Types

A circle indicates an independent primary failure event that does not require further development. A diamond shape indicates an event that has not been fully developed because the underlying causes are not fully known. An inhibit gate is used when a particular condition must be satisfied in order to allow a fault to propagate. Finally, a house symbol denotes a normally occurring event that is not a fault. The circle, diamond, inhibit gate and house symbols are all illustrated in (Figure 2). A more comprehensive list of fault tree symbols can be found in Atwood (2003).

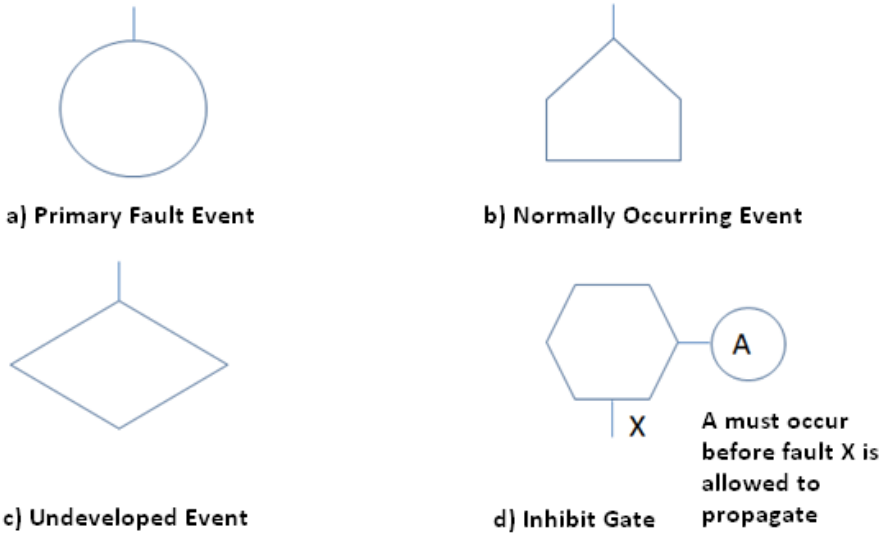
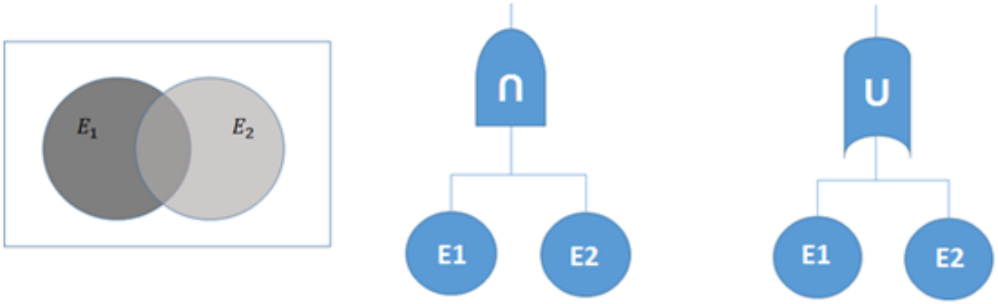


Figure 2 - Common Fault Tree Symbols

Fault tree analysis does not account for all possible system failures, but rather focuses on a particular failure mode. The development of the fault tree is a process where possible root causes within the defined system boundary are mapped out by working backwards from the top event. As a graphical aid, a Fault Tree allows system management and non-experts to visualize hazards. A fault tree can also be evaluated in a probabilistic manner by incorporating component failure data. A challenge with Fault Tree Analysis is that it does not easily allow for common-cause modeling, but rather assumes all primary events to be independent. Failures can be classified as primary, secondary or command faults. Primary faults occur in an environment the component was qualified for. Secondary faults occur when component fails in an environment it was not designed for. Finally a command fault occurs if a component operates correctly but at the wrong time. A qualitative fault tree evaluation involves developing a logical expression for the top event as a function of the primary fault events. A minimal cut-set requires all its associated primary faults to occur for system failure to occur (Vesely et. al., 1981). Smaller fault trees can

be evaluated manually, but for more complex trees computer codes are required in order to determine the minimal cut sets. By ordering the minimal cut-sets according to their size a qualitative measure of realative importance can be established. Quantitative Fault Tree evaluation requires estimation of the failure probabilities associated with the primary events in the Fault Tree. A typical approach is to calculate the failure probability for each primary event in the Fault Tree as the Poisson probability of observing at least one failure over the next time step (Eq. 1). All events in the Fault Tree are assumed to be independent and the top event probability is calculated by evaluating the associated logic expressions resulting from a series of AND/OR gates (Figure 3). Sample logic expression for AND/OR gates are provided in (Eq. 2 and Eq. 3).

$$P(at\ least\ one\ failure) = 1 - Poisson(X = 0) = 1 - \frac{e^{-\mu}\mu^{0}}{0!} = 1 - e^{-\mu} \quad \text{Eq. 1}$$



.

Figure 3 - AND/OR Gates

$$P(E_1 \cap E_2) = P(E_1) \cdot P(E_2) \quad \text{Eq. 2}$$

$$P(E_1 \cup E_2) = P(E_1) + P(E_2) - P(E_1) \cdot P(E_2) \quad \text{Eq. 3}$$

2.3.3 Monte Carlo Integration of Top Event Probability

Because failure rates are modeled as random variables, each term in the expression for the top event is associated with a probability distribution. Algebraic methods have been developed for determining the probability distribution function for combinations of random variables, but exact solutions are usually only possible for simple cases such as the sum of two independent distributions, which is also known as a convolution (Vose, 2008). A more generally applicable solution approach is to apply Monte Carlo sampling to integrate the top event numerically. A classic example that is used to explain Monte Carlo sampling is the problem of evaluating the expectation of a function (Eq. 4), where x is a vector of random variables distributed according to $f(x)$. This integral can be numerically approximated by drawing independent and identically distributed (i.i.d.) random samples from $f(x)$ (Eq. 5). By the law large numbers, the accuracy of the approximation will improve with the number of samples drawn (Eq. 6).

$$E[h(x)] = \int h(x)f(x)dx \qquad \text{Eq. 4}$$

$$E[h(x)] \approx \bar{h}_N = \frac{1}{N}\sum_{i}^{N} h(X_n) \qquad \text{Eq. 5}$$

$$\bar{h}_N = \frac{1}{N}\sum_{i}^{N} h(X_i) \rightarrow E[h(x)] \ \ as\ N \rightarrow \infty \qquad \text{Eq. 6}$$

2.3.4 Poisson Process

The Poisson process (Vose, 2008) is a stochastic model that has been widely applied to count processes, such as the number of calls arriving at a call center. The Poisson distribution is fully characterized by a single variable, i.e., the expected number of event counts over a given time period and yields the probability of observing a discrete number of counts given a rate and a given length of exposure time (Eq. 7). It can be formulated as a likelihood function for use in Bayesian Updating by setting y_t equal to the observed number of counts and treating the expected number of failures μ as a random variable.

$$p_{y_t} = \frac{e^{-\mu}\mu^{y_t}}{y_t!} \quad for\ y_t = 0,1,\dots,n \qquad \text{Eq. 7}$$

2.3.5 Poisson Exponentially Moving Average Model

The posterior distribution is closed-form if the prior and likelihood function fall in a conjugate pair of probability distributions. Combining a gamma prior distribution (Eq. 8) with a Poisson likelihood function (Eq. 9) results in a posterior gamma distribution (Eq. 10). The closed-form update equations for the posterior Gamma shape α and scale β factors shown in Eq. 11 and Eq. 12.

$$Prior = P(\mu_t; \alpha, \beta) = \frac{e^{-\beta\mu}{\mu_t}^{\alpha-1}}{\Gamma(\alpha)\beta^{-\alpha}} \qquad \text{Eq. 8}$$

$$Likelihood\ P(y_t|\mu_t) = \mu_t^{y_t} e^{-\mu_t}/y_t! \quad \text{Eq. 9}$$

$$Posterior = P(y_t|\mu_t) \cdot P(\mu_t; \alpha, \beta) = \frac{\mu_t^{y_t} e^{-\mu_t}}{y_t!} \cdot \frac{e^{-\beta\mu} {\mu_t}^{\alpha-1}}{\Gamma(\alpha)\beta^{-\alpha}} \quad \text{Eq. 10}$$

$$\alpha_{post} = \alpha_t = \alpha_{prior} + y_t \quad \text{Eq. 11}$$

$$\beta_{post} = \beta_t = \beta_{prior} + t \quad \text{Eq. 12}$$

A drawback with the traditional conjugate Poisson-Gamma approach is that all events are given equal weight. This is undesirable when modeling event counts over long time spans where the underlying event rate is likely to be changing. To mitigate this problem, Harvey (1989) introduced the Poisson Exponentially Weighted Moving Average (PEWMA), which models Poisson time series count data using a state space solution similar to that of the Kalman filter. The PEWMA model has been applied to count time series problems in political science (Brandt, 1998), disease control (Holloway, 2011) and nuclear risk analysis (Rangel, 2012). PEWMA reduces the weight associated with past data points (Eq. 13 and Eq. 14) by means of a discounting factor ω which controls the responsiveness of the model to measured data. Harvey uses conjugate Gamma/Poisson distributions and thereby retains a closed form solution. The mean is constant over the updating step, while the variance increases (Eq. 15 and Eq. 16). When count observations y_t become available, the updating step is applied and results in Eq. 17 and Eq. 18.

$$\alpha_{t|t-1} = \omega\alpha_{t-1} \quad \text{Eq. 13}$$

$$\beta_{t|t-1} = \omega\beta_{t-1} \quad \text{Eq. 14}$$

$$E(\mu_t|y_{t-1}) = \frac{\alpha_{t|t-1}}{\beta_{t|t-1}} = \frac{\omega\alpha_{t-1}}{\omega\beta_{t-1}} = \frac{\alpha_{t-1}}{\beta_{t-1}} = E(\mu_{t-1}|y_{t-1}) \quad \text{Eq. 15}$$

$$Var(\mu_t|y_{t-1}) = \frac{\alpha_{t|t-1}}{\beta_{t|t-1}^{\ 2}} = \frac{\omega\alpha_{t-1}}{(\omega\beta_{t-1})^2} = \frac{\alpha_{t-1}}{\omega\beta_{t-1}} = \frac{Var(\mu_{t-1}|y_{t-1})}{\omega} \quad \text{Eq. 16}$$

$$\alpha_t = \alpha_{t|t-1} + y_t = \omega\alpha_{t-1} + y_t \quad \text{Eq. 17}$$

$$\beta_t = \beta_{t|t-1} + t = \omega\beta_{t-1} + t \quad \text{Eq. 18}$$

From a Bayesian perspective, one could apply a distribution on *ω* as well, but this would remove the natural conjugate form and prevent a closed-form solution. A common approach is therefore to rather use maximum a posteriori or maximum likelihood techniques to estimate hyper-parameters such as *ω* (Harvey, 1989). Here, the log is taken of the posterior predictive distribution (Eq. 19). The function is then optimized with respect to *ω* to determine its optimal value before moving to the next updating step. Note that the maximum likelihood solution for *ω* simply provides an optimal fit between observed data and model output. This is not necessarily a conservative approach. The analyst may therefore want to experiment with different *ω* values and assess how past values are weighted before deciding to optimize.

$$\log\big(L(\omega)\big) = \sum_{t=\tau+1}^{T} \big\{\log\Gamma(a_{t|t-1} + y_t) - \log(y_t!) - log\Gamma\big(a_{t|t-1}\big) + a_{t|t-1} log b_{t|t-1} - (a_{t|t-1} + y_t)\log(1 + b_{t|t-1})\big\} \quad \text{Eq. 19}$$

Prior data is most valuable initially before a significant amount failure data becomes available. As an increasing amount of failure data is incorporated, the measured data will eventually dominate the posterior. The prior information is overwhelmed by the likelihood function more rapidly for smaller ω values, which are associated with heavier discounting of past data points. The prior probability distribution can be developed based on expert opinion or historic data from similar process installations. In cases where an uninformative prior is sought, Jeffrey's prior is often used (Atwood et. al., 2003). For the Gamma distribution, Jeffrey's prior is obtained when the shape and scale parameters of the gamma distribution are set to $\alpha_{prior} = 0.5$ and $\beta_{prior} = 0$. When plant specific data is not available, generic data from published sources can be used instead. An example of generic failure rate data obtained from the Offshore Reliability Data Handbook (OREDA, 2002) which contains failure rates for valves and level indicators in terms of calendar time. Sample data from OREDA (2002) is provided in Table 1.

Table 1 - Generic Failure Rates from OREDA

Failure Mode	**Failure rate per (10^6 hours)**			
	Lower	Mean	Upper	SD
Erratic output from level indicator – Taxonomy No 4.2.2.3	0.05	3.8	12.22	4.42
Blowdown valve fail to open on demand – Taxonomy No. 4.4.1	0	4.66	22.67	9.43
Alarm failure to function on demand – Taxonomy No 4.2.2	0	0.46	1.72	0.63

2.4 Model Implementation

2.4.1 PEWMA

For this study, the PEWMA model is implemented using the Python scripting language. A summary of the implemented code is provided in Algorithm 1. The complete set of Python scripts used to carry out the PEWMA analysis is provided in Appendix A. The PEWMA code is progressed through time and updates posterior Gamma parameters (α, β) for all primary events that have failure data available. For events that do not have failure data available the gamma parameters retain prior distribution values (α_0, β_0) throughout. Figure 4 and Figure 5 show how the PEWMA filter responds to a generic step function. Smaller ω values discount older points more heavily and lead to a faster response in the modeled failure rate. As such, ω models the amount of noise associated with the process, which is demonstrated in Figure 6.

Algorithm 1 - PEWMA

1 $Load\ failure\ data\ for\ each\ event\ i - \ y_t(i)$

2 $Load\ prior\ gamma\ parameters\ for\ each\ event\ i - \ \alpha_0^{(i)}\ and\ \beta_0^{(i)}$

2 $If\ (optimize\_omega = False)$

3 $Set\ \omega = \ constant\ value \sim (0,1]$

4 $For\ i = 1\ to\ n_{evemts}$

5 $For\ t = 1\ to\ n_{time}(i)$

6 $If(optimize\_omega = True)$

7 $$\omega = scipy.optimize.minimize(\sum_{t=\tau+1}^{T}\{\log\Gamma({a_{t|t-1}}^{(i)} + {y_t}^{(i)}) - \log({y_t}^{(i)}!) - log\Gamma({a_{t|t-1}}^{(i)}) + {a_{t|t-1}}^{(i)} log {b_{t|t-1}}^{(i)} - ({a_{t|t-1}}^{(i)} + {y_t}^{(i)})\log(1 + {b_{t|t-1}}^{(i)})\})$$

8 $Else$

9 $\alpha_t^{(i)} = \omega\alpha_{t-1}^{(i)} + y_t^{(i)}$

10 $\beta_t^{(i)} = \omega\beta_{t-1}^{(i)} + t$

11 END

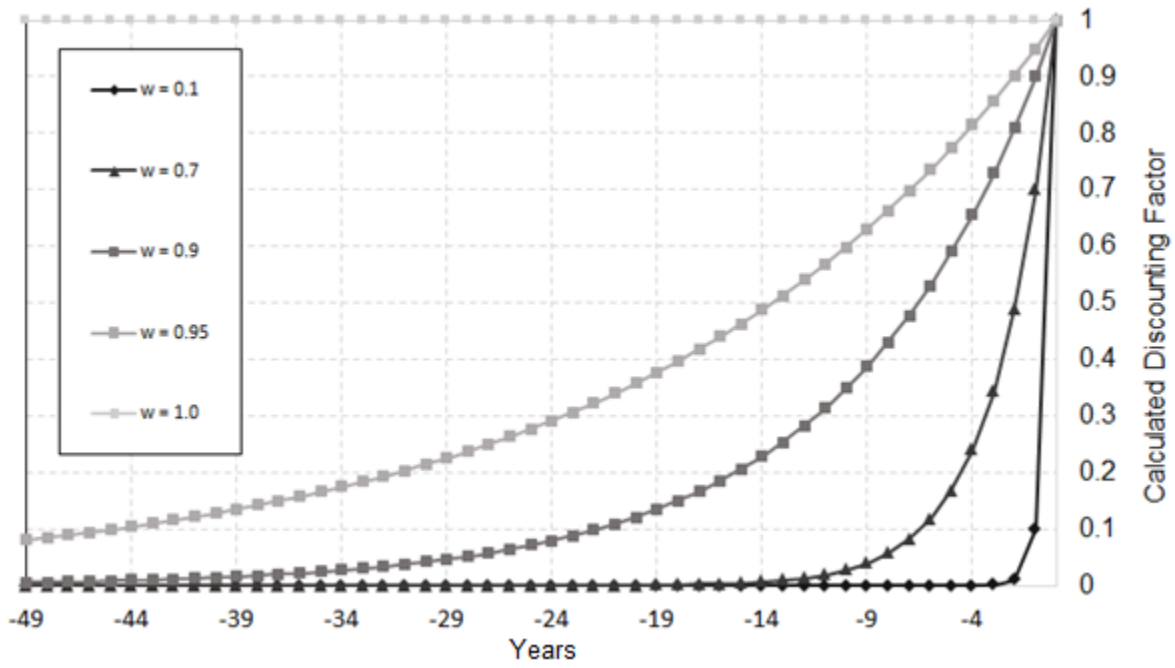


Figure 4 - Influence of PEWMA weighting factor

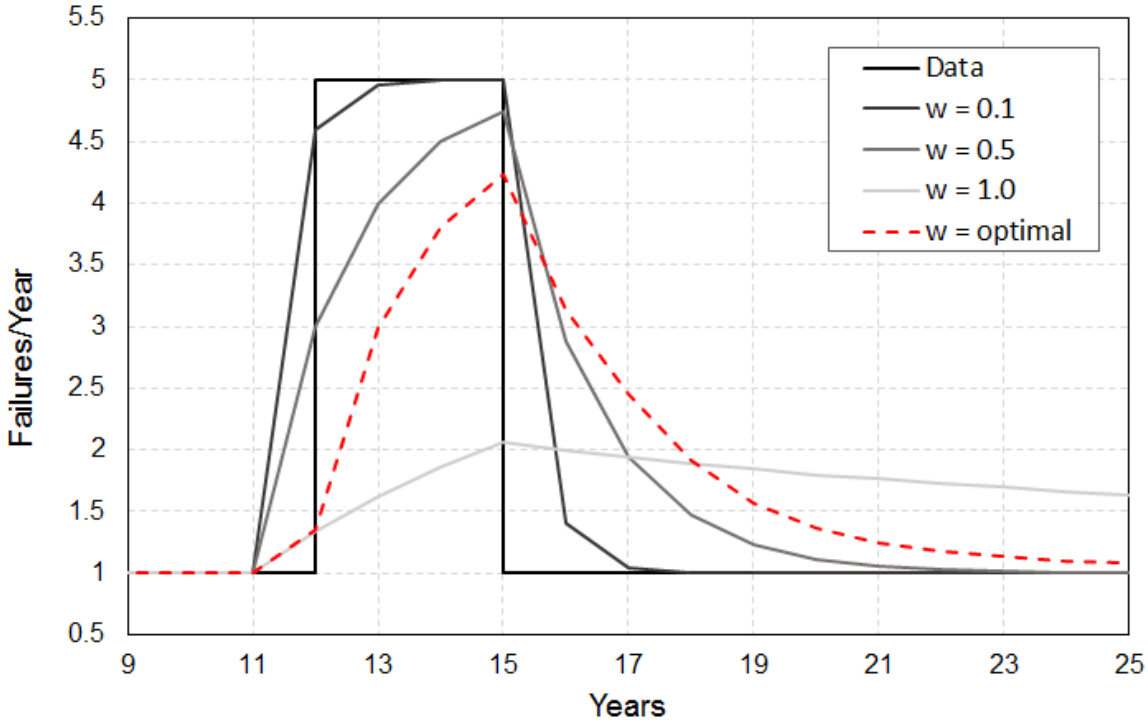


Figure 5 - PEWMA responsiveness

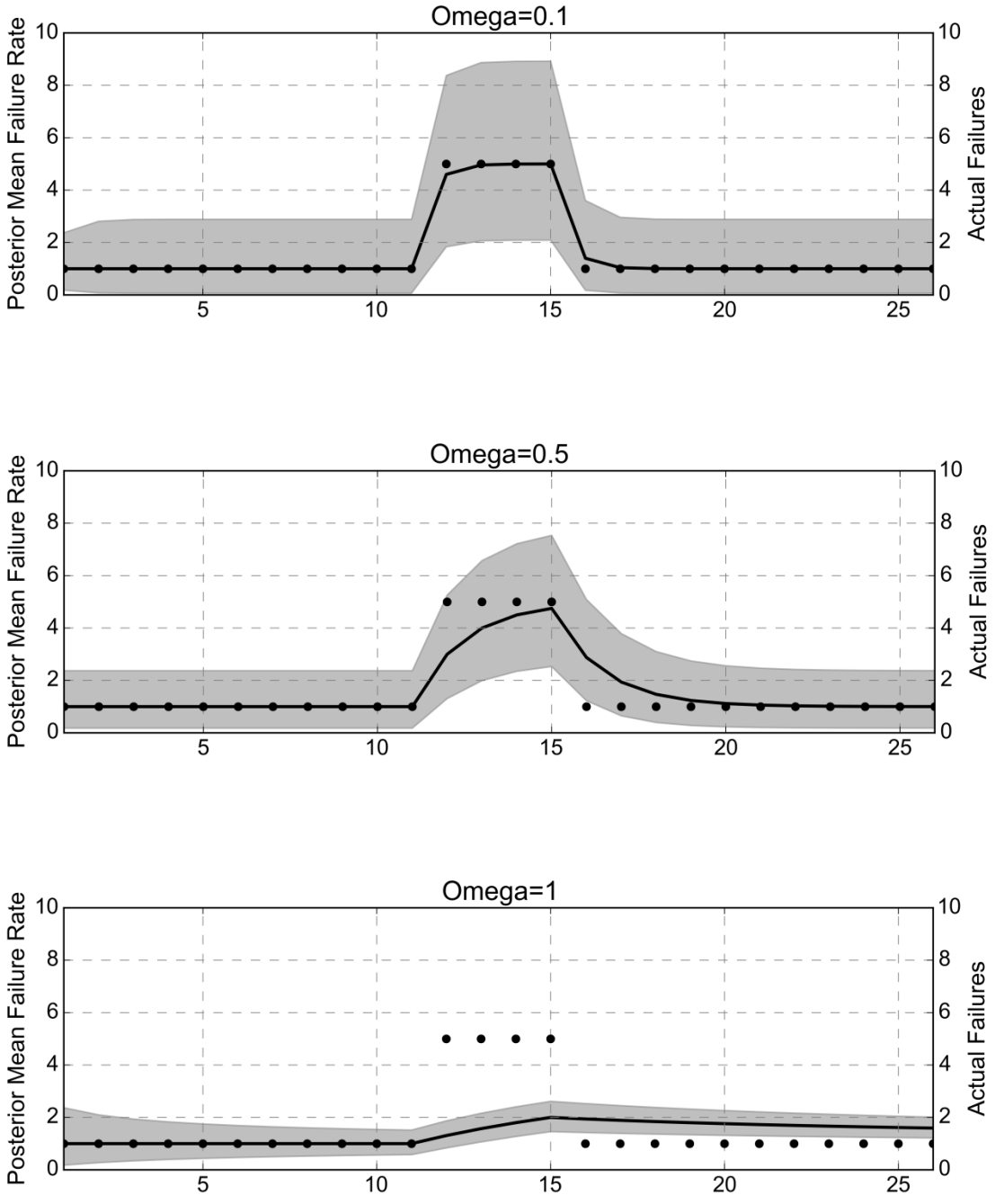


Figure 6 - Influence of omega factor

2.4.2 Top-Event Integration

A visual overview of the procedure for integrating the Fault Tree top event probability at each Bayesian updating step is provided in Figure 7. This figure emphasizes the fact that the process starts with PEWMA modeling of failure rates associated with the primary events in the fault tree. The posterior distributions for the primary events are then sampled and used to integrate the probability for the top event. A summary of the Python script implemented to carry out Monte Carlo integration of the top event probability is provided in Algorithm 2.

Algorithm 2 - Monte Carlo Top Event Integration

1 $Load\ Gamma\ parameters\ \alpha(i,t)\ and\ \beta(i,t)\ for\ all\ primary\ events$

2 $For\ t = 0\ to\ n_{time}$

3 $For\ s = 1\ to\ n_{MC\ samples}$

4 $For\ i = 1\ to\ n_{events}$

5 $Failure\ Rate\ Sample(s,i) \sim Gamma\ \big(\alpha(i,t), \beta(i,t)\big)$

6 $Failure\ Probability(s,i) = 1 - \exp[Failure\ Rate\ Sample(s,i)]$

7 $Top\ Event\ Prob(s) = f[Failure\ Probability(s,i)]$

8 $Mean(t) = statistics.mean(P(Top\ Event))$

9 $Percentile(t) = numpy.percentile(P(Top\ Event\ Prob), 0.95)$

10 END

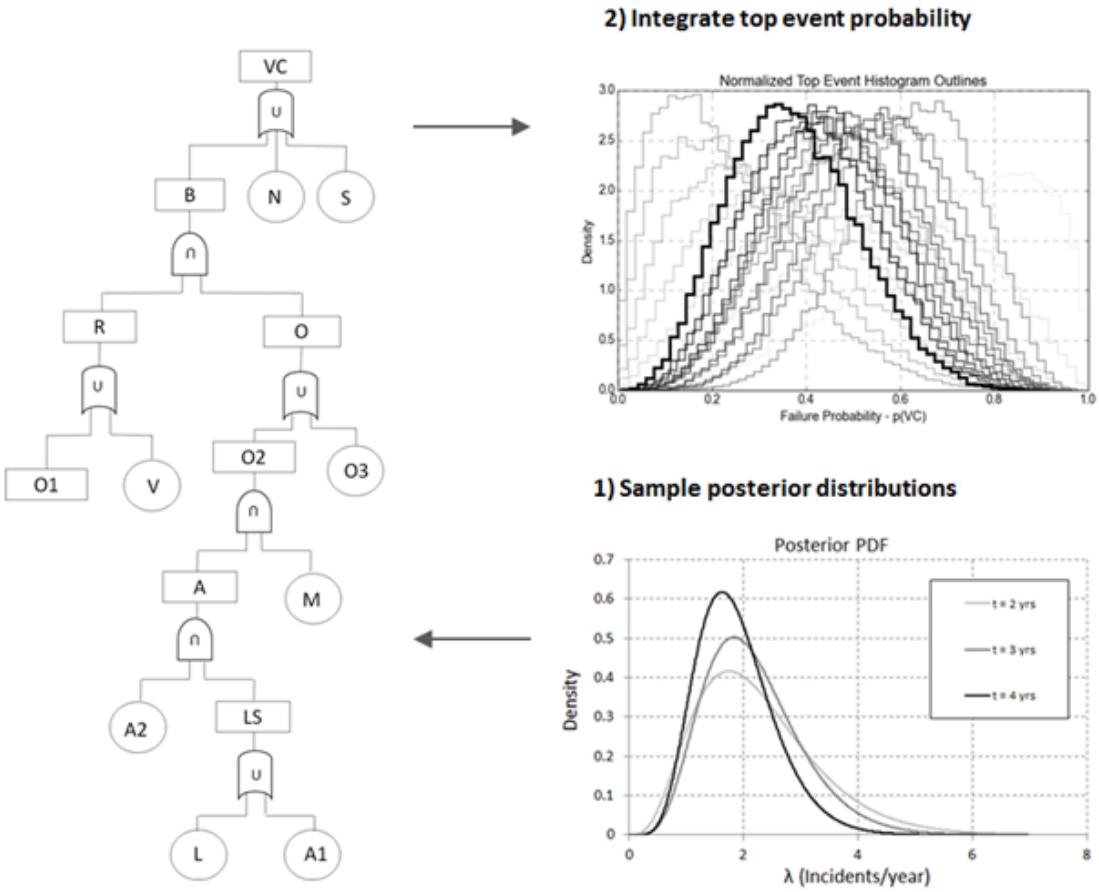


Figure 7 - Monte Carlo Sampling Procedure for Integration Top Event Probability

2.5 Case Study - Texas City Fault Tree Analysis

2.5.1 Development of Texas City Fault Tree

The BP Texas City Oil Refinery incident occurred on March 23 2005 in the isomerization unit (ISOM) of the oil refinery, which converts linear molecules to higher-octane branched molecules for blending into gasoline or feed to alkylation units (CSB, 2007). The incident was an explosion caused by heavier–than-air hydrocarbon vapours combusting after coming into contact with an ignition source (BP, 2005 and CSB, 2007). Hydrocarbon vapors were released due to overfilling of liquids in the raffinate splitter tower, causing both hydrocarbon liquids and vapors to overflow into the blowdown drum and discharge into the atmosphere (BP, 2005 and CSB, 2007). An overview of the unit taken from the U.S. Chemical and Hazard Investigation Board (CSB, 2007) is shown in Figure 8. Investigations carried out by BP and the U.S. Chemical Safety and Hazard Investigation Board (BP, 2005 and CSB, 2007) revealed that the incident occurred due to a complex series of events involving maintenance issues, lack of training of key personnel, lack of safety culture, instrumentation and equipment failure and unsafe designs.

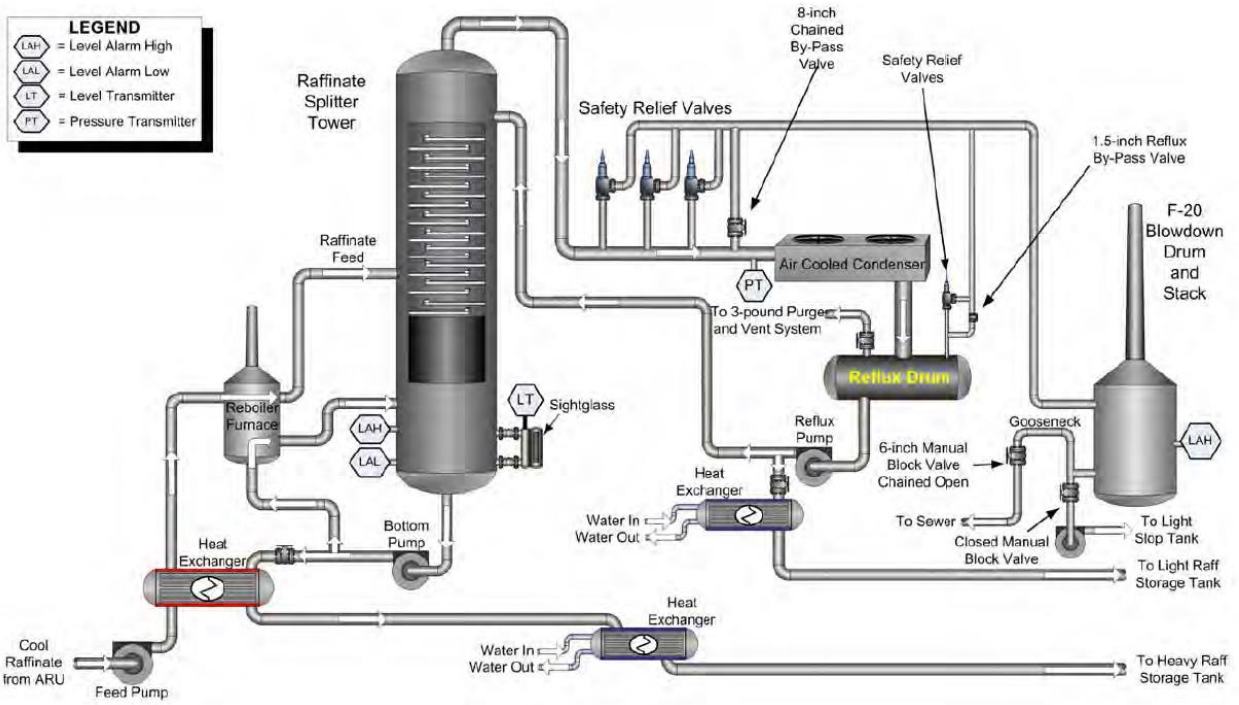


Figure 8 - Overview of the Texas City Refinery ISOM Unit (CSB 2007)

The top event for the Texas City refinery's ISOM unit is defined as the formation of a flammable vapor cloud outside the blowdown drum. The primary fault leading to the top event is defined as the release of a flammable vapour cloud from the blowdown drum during normal operations. The sewer release event is not developed further in this Fault Tree because the system boundary is limited to the ISOM unit. The only fault which is developed further for this gate is the event which caused the Texas City incident, namely the blowdown drum overfilling with hydrocarbon liquids and vapours.

The raffinate splitter tower can potentially overflow with liquid hydrocarbons in two different manners: 1.) The first scenario occurred during the Texas City incident and involved the blowdown valve being closed while continuous feed was introduced to the tower from the

Aromatics Recovery Unit (ARU). At some point after the splitter tower was filled far beyond the allowable limit, the operator deemed that the tower was likely to be overfilled and ordered bottom hydrocarbons to be taken out of the tower. This caused overheated bottom hydrocarbons to exchange heat with the incoming feed, which in turn vaporized on its way into the raffinate splitter tower. The vaporization of the incoming feed caused the raffinate splitter tower to overflow and a combination of liquids and vapours to flow into the tower's overhead line and consequently into the blowdown drum. 2.) The second scenario occurs if the liquid level is allowed to rise continuously for a sufficient amount of time, eventually causing the tower to overfill with single-phase liquids. These two scenarios are similar in that the former simply represents an accelerated version of the latter, whereby heat exchange and vaporization of incoming feed caused the raffinate splitter tower to overfill earlier than it would have had the operators decided not to open the bottom valve late in the start-up of the raffinate splitter tower. In this Fault Tree the second scenario is adopted.

The liquid level in the raffinate splitter tower will continuously rise if the inflow rate exceeds the outflow rate from the tower and the operator does not take action to either stop the feed into the raffinate splitter tower or to open the bottom valve. Continuous feed to the raffinate splitter tower is not considered to be a fault, but rather an external event that occurs during normal operations and it is therefore indicated with a house symbol. As a worst case scenario, it is assumed that liquids can only leave the tower through the bottom valve, since a high liquid level would effectively prevent vapor formation and subsequent flow of vapors into to the tower's overhead lines. It is also assumed that the operator will not take action to either stop feed into the raffinate splitter tower or to open the bottom valve of the splitter tower if he/she is unaware of the rising

liquid level. The operator being unaware of the liquid level due to instrumentation failure is considered an intermediate fault that can be attributed to the simultaneous occurrence of the raffinate tower's alarm system failing and the level sight glass associated with the raffinate splitter tower being impossible to see through. The raffinate tower's alarm system is assumed to fail when both the alarm system associated with the level indicator fails and the redundant hard-wired high level alarm fails. The operator being unaware of the rising liquid level can be attributed to a lack of system understanding due to insufficient personnel training. The operator being unaware of the liquid level due to being unconscious is an undeveloped fault which is included for completeness, though is not developed further here. The bottom valve of the raffinate splitter tower being closed during start-up is assumed to be an intermediate fault caused by the operator not following start-up regulations or the bottom valve failing in the closed position. The complete fault tree is shown in Figure 9.

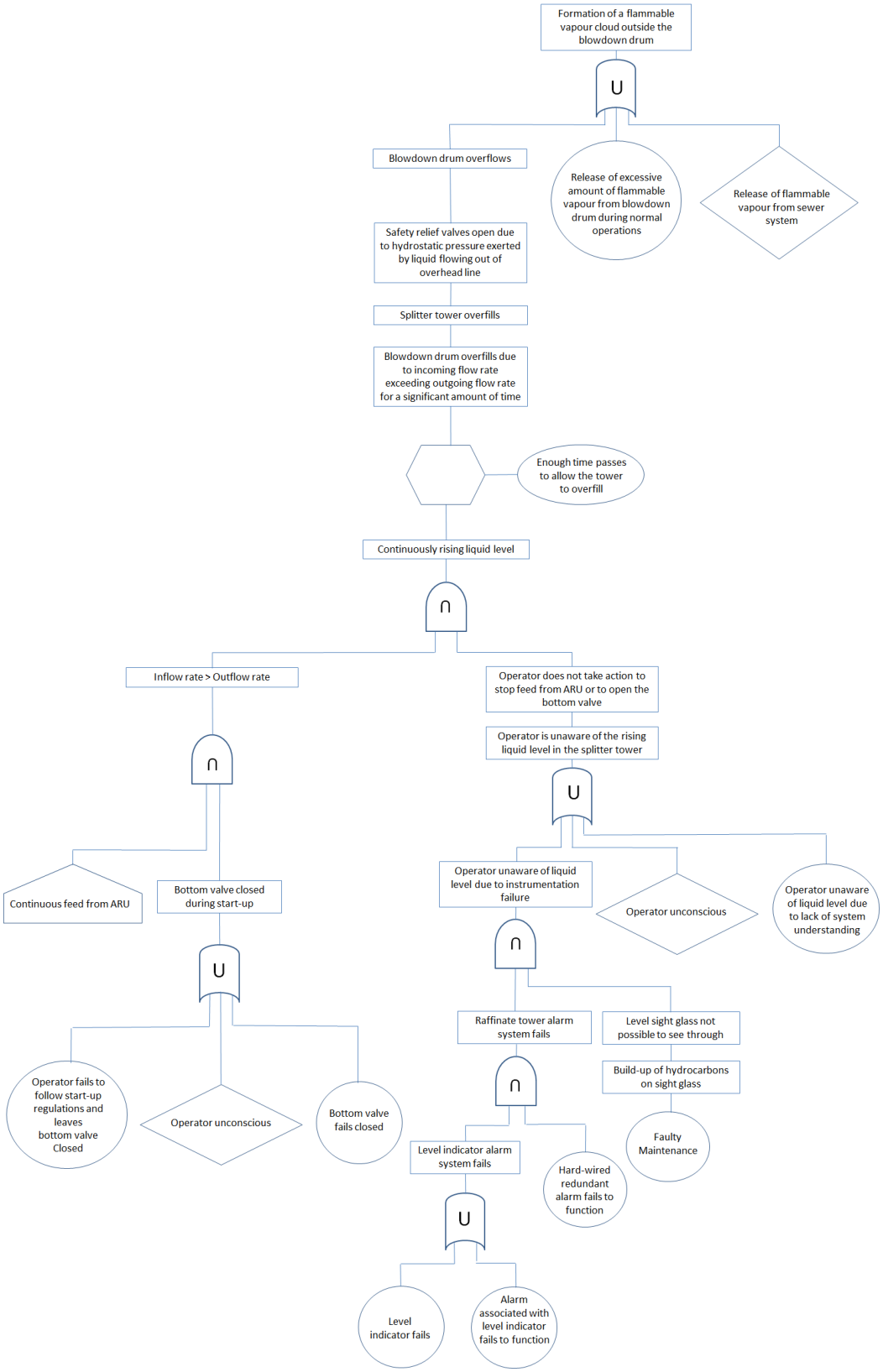


Figure 9 - Complete Texas City ISOM unit Fault Tree

2.5.2 Qualitative Analysis of Texas City Fault Tree

For the purpose of developing a fault tree that can be evaluated qualitatively in terms of minimal cut-sets, only the primary events are included here (Figure 10). Starting with the top event and working downwards, the logical expressions are developed as follows from Eq. 20 to Eq. 25. The final logical expression for the top event is provided in Eq. 26.

$$VC = B + N + S \quad \text{Eq. 20}$$

$$VC = (R \cdot O) + N + S \quad \text{Eq. 21}$$

$$VC = (O1 + V) \cdot (O2 + O3) + N + S \quad \text{Eq. 22}$$

$$VC = (O1 + V) \cdot ((A \cdot M) + O3) + N + S \quad \text{Eq. 23}$$

$$VC = (O1 + V) \cdot (((A2 \cdot LS) \cdot M) + O3) + N + S \quad \text{Eq. 24}$$

$$VC = (O1 + V) \cdot (((A2 \cdot (L + A1)) \cdot M) + O3) + N + S \quad \text{Eq. 25}$$

$$VC = (O1 \cdot L \cdot A2 \cdot M) + (O1 \cdot O3) + (O1 \cdot A1 \cdot A2 \cdot M) + (L \cdot A2 \cdot M \cdot V) + (A1 \cdot A2 \cdot M \cdot V) + (V \cdot O3) + N + S \quad \text{Eq. 26}$$

The failure probability often decreases by orders of magnitude as the size of the cut sets increase. The ranking of the minimal cut sets according to size therefore provides an indication of relative failure probabilities and the importance of each cut set. The smallest cut sets are placed at the top of the list, and the larger cuts-sets follow accordingly below.

Minimal cut-sets:

N

S

$(V \cdot O3)$

$(O1 \cdot O3)$

$(A1 \cdot A2 \cdot M \cdot V)$

$(L \cdot A2 \cdot M \cdot V)$

$(O1 \cdot A1 \cdot A2 \cdot M)$

$(O1 \cdot L \cdot A2 \cdot M)$

From the above minimal cut-sets it is clear that the most important events are: 1) the release of a flammable vapor cloud from the blowdown drum during normal operation (N) and 2) the release of an excessive amount of flammable hydrocarbons from the blowdown drum through the sewer system (S). As such, the reliability of the system can be improved by: 1) installing a pressure vessel and flare to the blowdown drum to reduce the probability of vapor cloud release 2) enhancing the sewer design system to prevent a vapor cloud from forming after dumping of hydrocarbons.

The second order cut sets involve the operator being unaware of the rising liquid level in the raffinate splitter tower due to a lack of training (O3) and understanding of the system. This can be prevented by installing a severe overfilling alarm connected to an automatic shut-down mechanism. The improved design is incorporated into the Fault Tree and is shown in (Figure 10). The severe overfilling alarm affects all cut sets except for the events N (Vapour release from

blowdown drum) and S (vapour release from sewer system) and is expressed in (Eq. 27). Overall, this case study demonstrates that a qualitative assessment of the cut sets can reveal valuable design improvements prior to a quantitative assessment.

$$VC = (O1 \cdot L \cdot A2 \cdot M \cdot SA) + (O1 \cdot O3 \cdot SA) + (O1 \cdot A1 \cdot A2 \cdot M \cdot SA) + (L \cdot A2 \cdot M \cdot V \cdot SA) + (A1 \cdot A2 \cdot M \cdot V \cdot SA) + (V \cdot O3 \cdot SA) + N + S \quad \text{Eq. 27}$$

VC = Vapor cloud forms outside blowdown drum

B = Blowdown drum overfills

N = Release of vapor cloud from blowdown drum

S = Release of vapor cloud from sewer

R = Raffinate splitter tower overfills

O = Operator unaware of raffinate splitter tower liquid level

O1 = Operator not following regulations

O2 = Operator unaware of raffinate liquid level due to alarm failure

O3 = Operator unaware of raffinate liquid level due to lack of training

M = Maintenance failure (sight glass)

V = Raffinate splitter tower blowdown valve fails closed

L = Raffinate splitter tower level indicator fails to function

LS = Raffinate splitter tower level indicator alarm system fails

A = Raffinate splitter tower high level alarm system fails

A1 = Alarm associated with level indicator fails to function

A2 = Raffinate splitter tower hardwired alarm fails to function

SA = Severe liquid level alarm failure

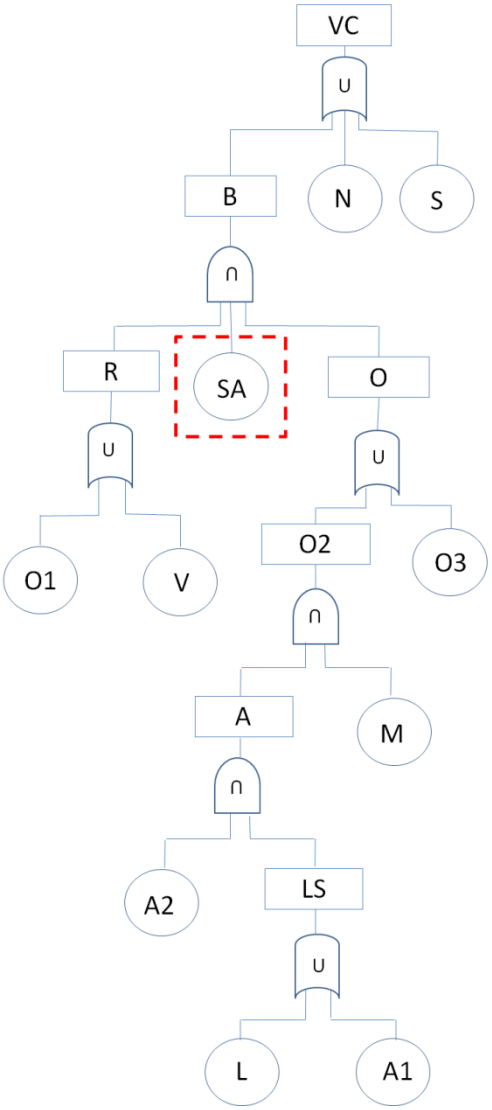


Figure 10 - Simplified Fault Tree including improved alarm system (SA)

2.5.3 PEWMA Updating using Texas City Failure Data

Here, the failure rates for primary events in the Texas City Fault Tree are modeled using PEWMA with accident precursor data from the CSB (2007) report. The data set provided in CSB (2007) is not likely to be complete, however, does illustrate how plant specific parameter estimation can be used as part of dynamic risk assessments. The PEWMA model is applied to model how the failure rate is changing over time for the individual primary events. For this case study, a PEWMA omega value $\omega = 0.9$ is used, which means that data points that are 5 years old are discounted by about 40%, while data points 10 years old are discounted by about 70%. A summary of the Gamma prior parameters used in the Texas City PEWMA model is shown in Table 2.

Table 2 - Prior Gamma Parameters for PEWMA Analysis

$\boldsymbol{Variable}$	$\boldsymbol{\alpha}$	$\boldsymbol{\beta}$	$\boldsymbol{Mean = \alpha/\beta}$	$\boldsymbol{Variance = \alpha/\beta^2}$
N	2	1	2	2
S	1	1	1	1
O1	2	1	1	2
O3	2	1	2	2
M	2	1	2	2

2.5.3.1 Blowdown drum vapour cloud release rate (N)

Exposure time for the blowdown drum is counted from 1987 when the last major capacity increase to the splitter tower was made. The incident data is summarized in Table 3. Shortly after the capacity increase, safer alternatives to the blowdown drum was proposed by the Amoco Refining and Planning Department (ARPD) and the Occupational Safety and Health Administration (OSHA), however due to cost constraints the unsafe blowdown drum design remained in place until the Texas City incident in 2005 (CSB report, 2007). Figure 11 shows the mean and 95th percentiles vs. time and the posterior gamma distributions through time.

Table 3 - Blowdown drum vapour cloud release incidents (B)

Incident Description	**Year**	**t (years)**	**y_t (failures)**
Vapours from blowdown drum	1994	7	1
Vapours from blowdown drum	1994	7	2
Relief valve discharge to blowdown drum	1994	7	3
Significant blowdown drum release	1995	8	4
Oil mist from blowdown drum	1995	8	5
Blowdown drum vapours disturbing workers	1995	8	6
Significant blowdown drum release	1999	12	7
Liquid hydrocarbon release to blowdown drum	2003	16	8

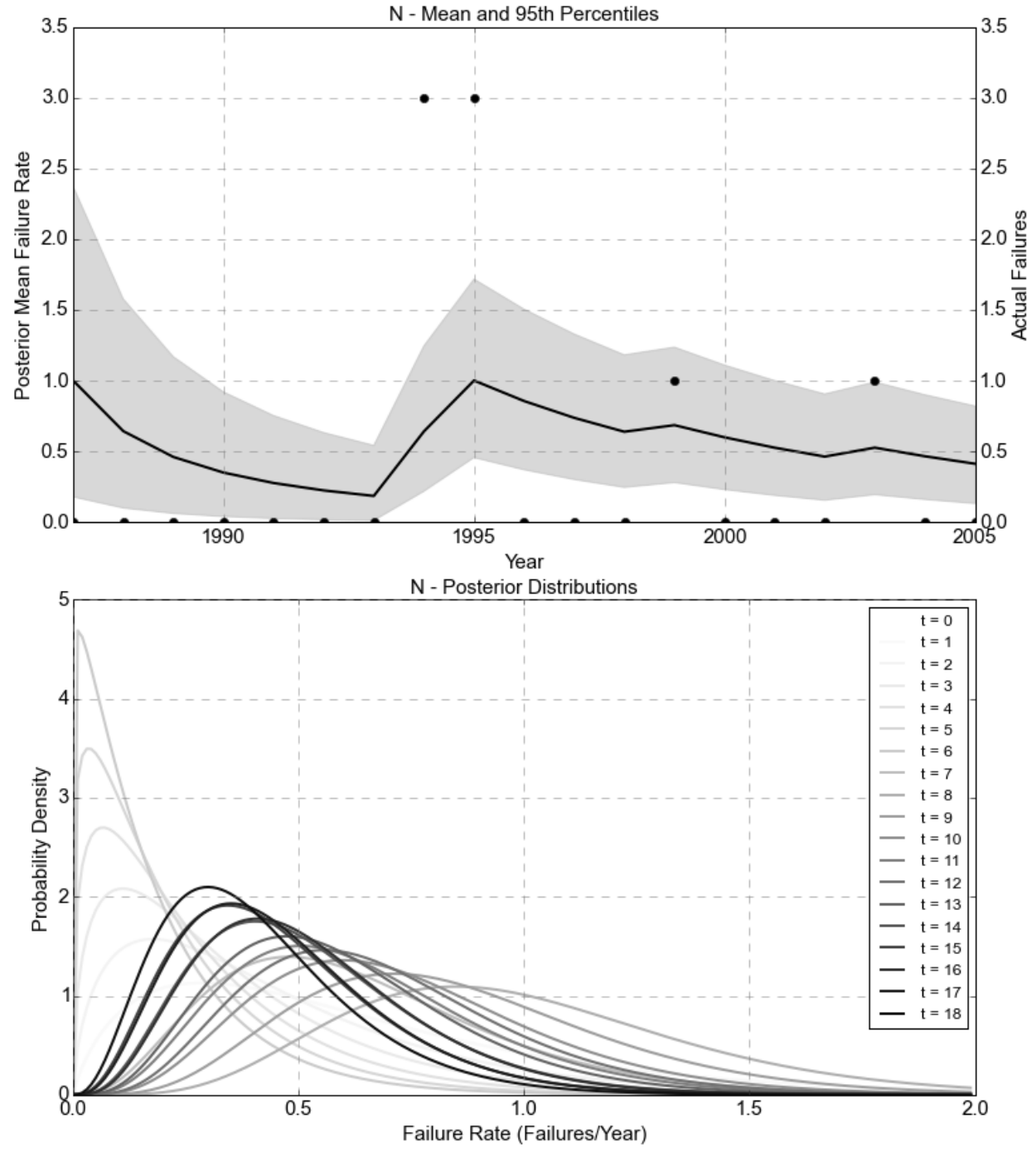


Figure 11 - PEWMA model ouput (N - Blowdown Drum)

2.5.3.2 Unsafe Sewer Disposal Rate (S)

The CSB report (2007) documents one incident of unsafe disposal of hydrocarbons into the sewer occurring in 1999 and resulting in the formation of a dangerous vapour cloud. The total exposure time is here counted from 1987 when the last major capacity increase of the splitter tower was made. Figure 12 shows the mean and 95^{th} percentiles vs. time and the posterior gamma distributions through time.

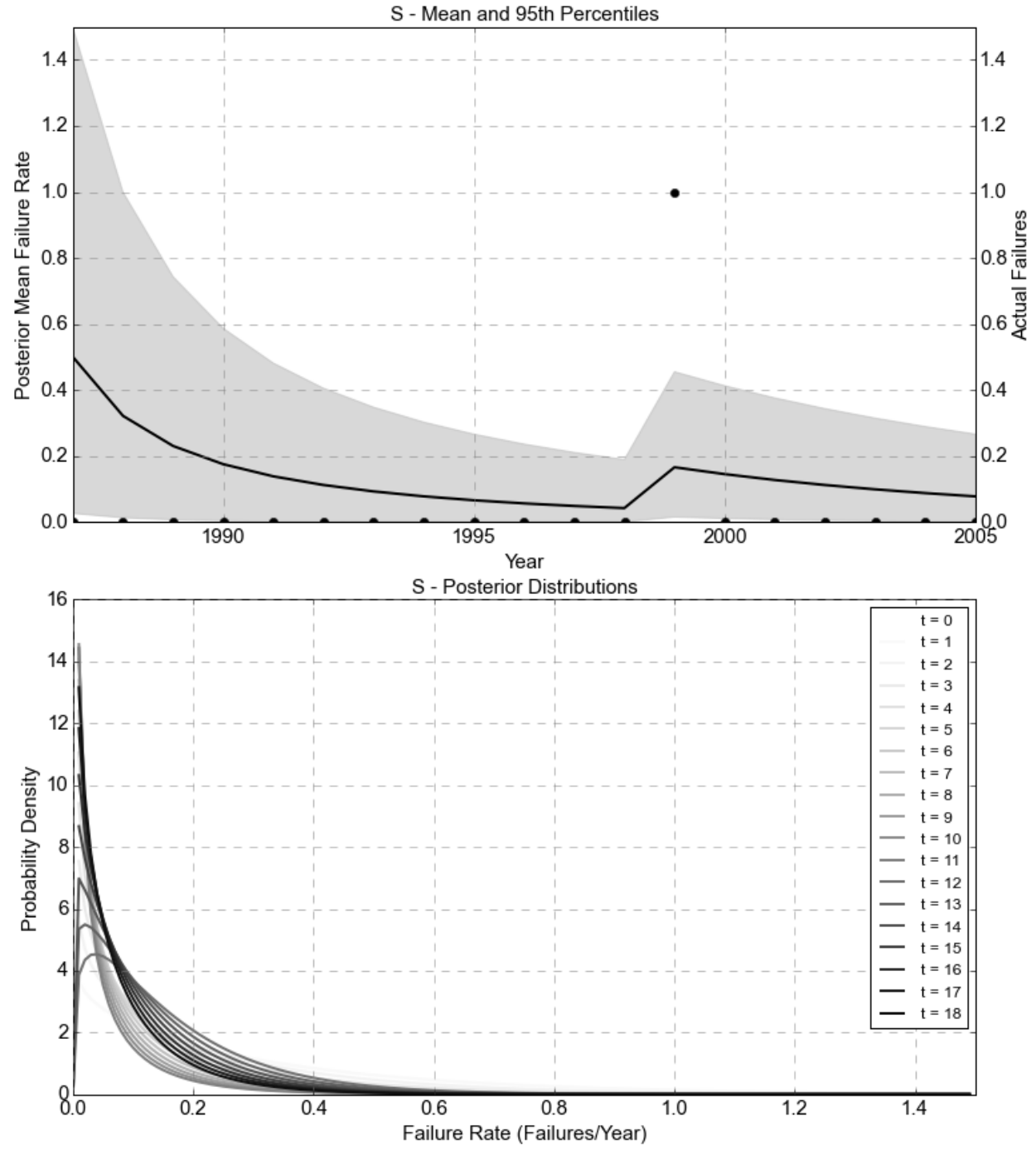


Figure 12 - PEWMA Output (S - Sewer Release)

2.5.3.3 Insufficient Operator Training (O3)

Non-compliance with training requirements and the inability of plant personnel to learn from previous incidents is defined here as training insufficiency. Failure data is collected from CSB (2007). The total exposure time is counted from 1997 when safety audit systems for behavioral safety were implemented at the Texas City refinery (CSB report, 2007) and the incident data is summarized in Table 4. Figure 13 shows the mean and 95th percentiles vs. time and the posterior gamma distributions through time.

Table 4 - Insufficient Training incidences (O3)

Incident Description	**Year**	**t (yrs)**	y_t **(failures)**
The ISOM unit's HAZOP revalidation does not address previous incidents	1998	1	1
The ISOM blowdown stack catches fire, no investigation to learn about causes	2000	3	2
BP's learning & development center unable to get training simulators for Texas City	2000	3	3
PSM audit finds a number of PHA items past due dates	2001	4	4
BP Group report reveals that root causes for accidents are not being investigated	2002	5	5
ISOM unit HAZOP revalidation again does not address previous incidents	2003	6	6
OCAM audit reveals no individual operator development plans in place	2003	6	7
GHSER audit determines that training and incident investigation are insufficient	2003	6	8
PSM audit reveals inadequate learning from previous incidents and lack of training	2004	7	9
GHSER assessment grades Texas City as "poor" due to lack of learning from incidents	2004	7	10
Telos survey finds serious safety issues related to inadequate training	2004	7	11
OCAM audit reveals deficencies in training of operators at Texas City	2004	7	12
Texas City incident - operator unaware of liquid levels due to lack of training	2005	8	13

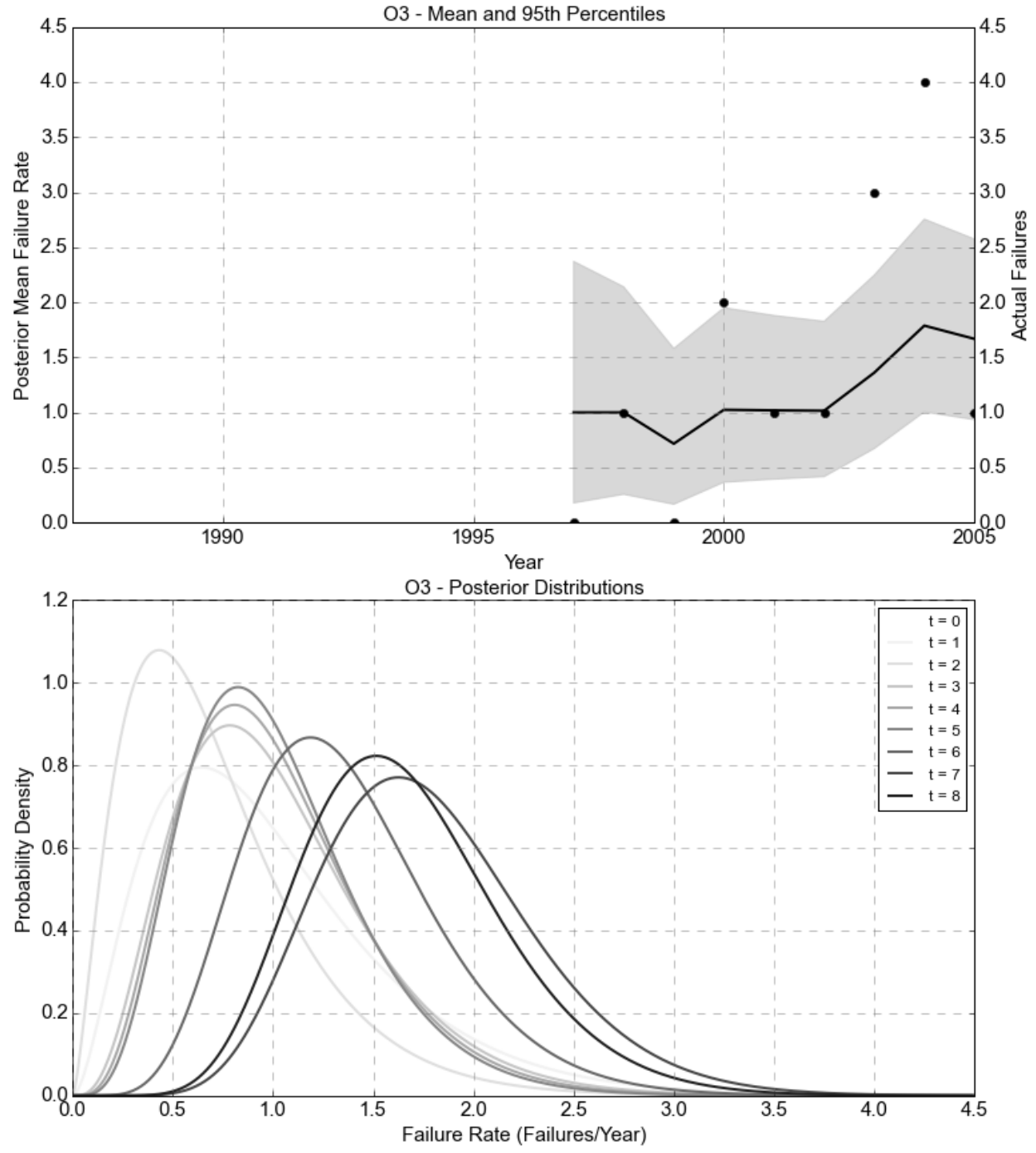


Figure 13 - PEWMA Output (O3 - Operator Training)

2.5.3.4 Maintenance Failure Rate (M)

Maintenance failure is assessed based on audits and investigations made at the Texas City refinery as documented in CSB (2007). Prevalent equipment corrosion problems, failure to conform to maintenance standards, repair schedules and problems with mechanical integrity are defined as failure. Total exposure time is counted from 2001, when the BP Group issued a "Process Safety/Integrity Management" standard outlining the minimum requirements to prevent catastrophic incidents. Incident data is summarized in Table 5. Figure 14 shows the mean and 95th percentiles vs. time and the posterior gamma distributions through time.

Table 5 - Maintenance failure incidences (M)

Incident Description	**Date (year)**	**t (yrs)**	y_t **(failures)**
Gap assessment reveals maintenance/mechnical integrity problems at Texas City	2003	2	1
Inspection of blowdown drum reveals damage to trays - no repair recommended	2003	2	2
PSM requires review of ISOM relief valves - the study is never completed	2003	2	3
Major corrosion damage on the blowdown drum	2003	2	4
Significant corrosion detected on the raffinate splitter tower	2004	3	5
Texas city scores low on PSM metrics such as action item completion (maintenance)	2004	3	6
Texas City incident (alarms, sight glass, etc. Not properly maintained)	2005	4	7

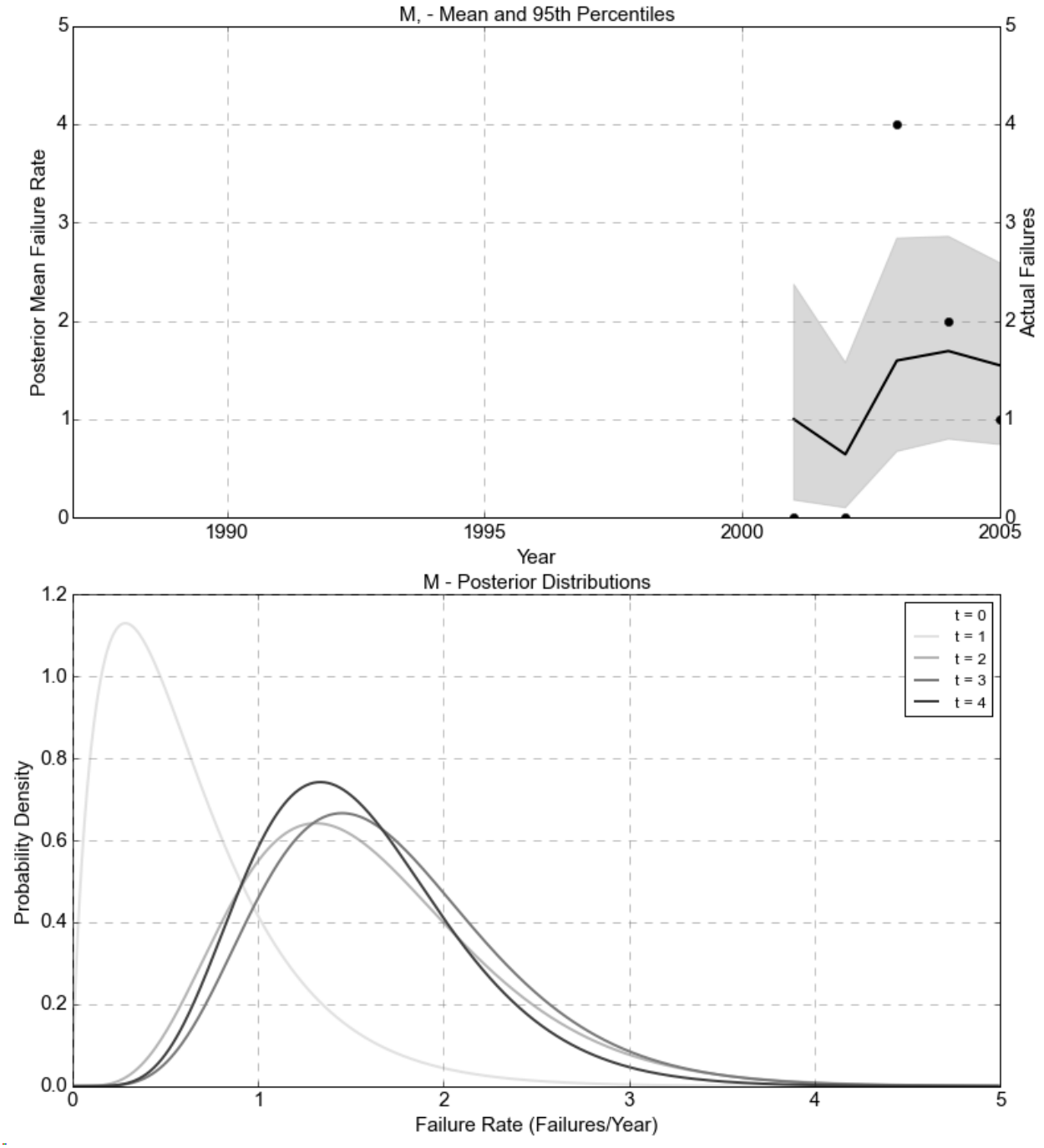


Figure 14 - PEWMA Output (M – Maintenance Failure)

2.5.3.5 Regulations Non-Compliance rate (O1)

Analysis of the Texas City incident revealed that operational problems at the refinery were not corrected over time. Operators were found to have deviated from established procedures, such as leaving the bottom valve of the raffinate splitter tower in the closed position during start-up of the ISOM unit. The operator failing to follow start-up regulations is assumed to be a primary fault, with an occurrence rate that can be estimated based on historical safety audits at the plant (BP report, 2005 and CSB report, 2007). The exposure time is counted from 1993 when the first HAZOP was conducted at the Texas City Refinery and the incident data is summarized in Table 6. Figure 15 shows the mean and 95th percentiles vs. time and the posterior gamma distributions through time.

Table 6 - Regulations non-compliance incidences (O1)

Incident Description	**Year**	**t (yrs)**	y_t **(failures)**
DIH distillation tower in the ISOM unit is overfilled and results in a vapour cloud	1994	1	1
ISOM stabilizer tower emergency relief valves are open 5-6 times over 4 hours	1994	1	2
8-inch chain vent valve (raffinate splitter) is left open for 20 hours	1995	2	3
PSM audit reveals that operating procedures at Texas City are not current	2001	8	4
PSM requires review of ISOM relief valves, study is never completed	2003	10	5
Use of pressure relief valves against regulations in raffinate unit	2004	11	6
Poor PSM scores on items related to action item completion	2004	11	7
Texas City incident, splitter tower was filled beyond regulations, alarms ignored	2005	12	8

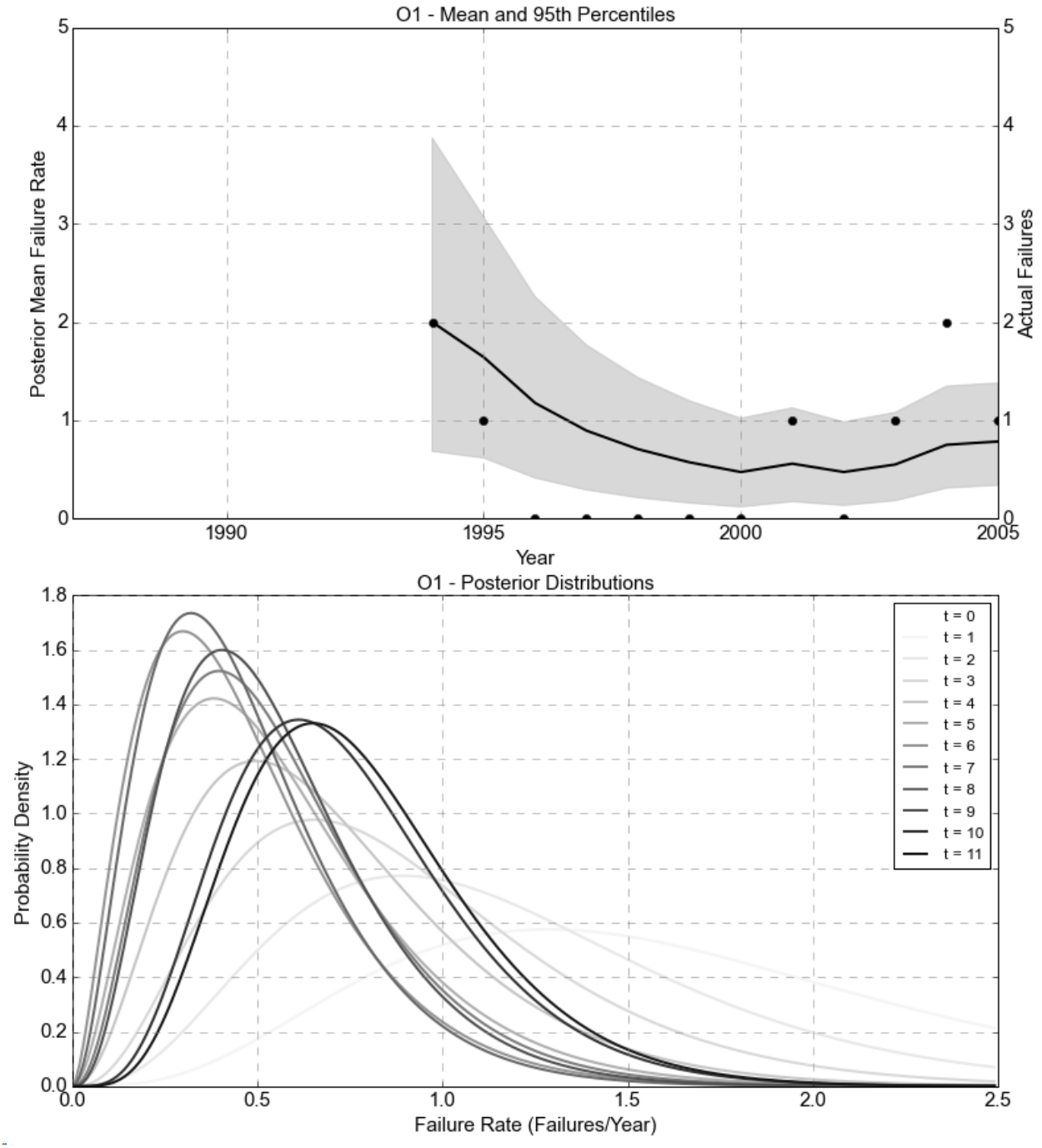


Figure 15 - PEWMA Output (O1 - Regulations Non-Compliance)

2.6 Probabilistic Analysis of Texas City Fault Tree

All primary events in the Texas City fault tree are modeled as random variables with individual failure rate distributions resulting from the PEWMA analysis. The resulting gamma parameters are summarized in Table 7. Grey fields indicate static parameters and white fields indicate dynamic PEWMA parameters. Eq. 28 is the expression for the top event resulting from the qualitative analysis. This expression is integrated using Monte Carlo sampling at each Bayesian updating step. For each sample obtained from the posterior failure rate distributions, a corresponding failure probability is calculated using the Poisson process by assuming at least one failure over an exposure time of one year (Eq. 35).

$$P(VC) = P(B) + P(N) + P(S) - P(B) \cdot P(N) - P(B) \cdot P(S) - P(N) \cdot P(S) + P(B) \cdot P(N) \cdot P(S) \quad \text{Eq. 28}$$

Where,

$$P(B) = P(R) \cdot P(O) \quad \text{Eq. 29}$$

$$P(R) = P(O1) + P(V) - P(O1) \cdot P(V) \quad \text{Eq. 30}$$

$$P(O) = P(O2) + P(03) - P(O2) \cdot P(O3) \quad \text{Eq. 31}$$

$$P(O2) = P(A \cap M) = P(A) \cdot P(M) \quad \text{Eq. 32}$$

$$P(A) = P(LS \cap A2) = P(LS) \cdot P(A2)$$ Eq. 33

$$P(LS) = P(L) + P(A1) - P(L) \cdot P(A1)$$ Eq. 34

$$P(at\ least\ one\ failure) = 1\ -\ Poisson(X = 0) = 1 - \frac{e^{-\mu}\mu^{0}}{0!} = 1 - e^{-\mu}$$ Eq. 35

Figure 16 shows posterior histogram outlines through time and Figure 17 shows mean top event failure probability and the associated 95th percentiles, resulting from Monte Carlo sampling of the top event probability. It is evident that the failure probability is cyclic, which demonstrates the dynamic effects of incorporating plant-specific accident precursor data into the analysis. The mean failure probability decreases initially because the collected data indicates a lower failure probability than the prior data. This is also demonstrated in the variance, which initially increases because the collected data conflicts with the prior information. Finally, the severe overfilling alarm (SA), is incorporated into the expression for the top event by multiplying the expression for P(B) by P(SA). The resulting mean failure probability and posterior histograms are shown in Figure 18 and Figure 19. Compared to the failure probability calculated without the severe overfilling alarm in place, the improvement in top-event failure probability is significant. This analysis shows the value of carrying out a qualitative Fault Tree analysis for identifying safety improvements and a quantitative fault tree analysis for quantifying the effect of the improvements.

Table 7 - Summary of Gamma Parameters

	Posterior Parameters from PEWMA model										Static Parameters based on OREDA					
	N		**M**		**O1**		**O3**		**S**		L		V		A	
Year	**α**	**β**	**α**	**β**	**α**	**β**	**α**	**β**	**α**	**β**	**α**	**β**	**α**	**β**	**α**	**β**
1987	2.00	2.00	2.00	2.00	4.00	2.00	2.00	2.00	1.00	2.00	0.66	21.51	0.26	6.18	0.33	97.49
1988	1.80	2.80	2.00	2.00	4.00	2.00	2.00	2.00	0.90	2.80	0.66	21.51	0.26	6.18	0.33	97.49
1989	1.62	3.52	2.00	2.00	4.00	2.00	2.00	2.00	0.81	3.52	0.66	21.51	0.26	6.18	0.33	97.49
1990	1.46	4.17	2.00	2.00	4.00	2.00	2.00	2.00	0.73	4.17	0.66	21.51	0.26	6.18	0.33	97.49
1991	1.31	4.75	2.00	2.00	4.00	2.00	2.00	2.00	0.66	4.75	0.66	21.51	0.26	6.18	0.33	97.49
1992	1.18	5.28	2.00	2.00	4.00	2.00	2.00	2.00	0.59	5.28	0.66	21.51	0.26	6.18	0.33	97.49
1993	1.06	5.75	2.00	2.00	4.00	2.00	2.00	2.00	0.53	5.75	0.66	21.51	0.26	6.18	0.33	97.49
1994	3.96	6.17	2.00	2.00	4.00	2.00	2.00	2.00	0.48	6.17	0.66	21.51	0.26	6.18	0.33	97.49
1995	6.56	6.56	2.00	2.00	4.60	2.80	2.00	2.00	0.43	6.56	0.66	21.51	0.26	6.18	0.33	97.49
1996	5.90	6.90	2.00	2.00	4.14	3.52	2.00	2.00	0.39	6.90	0.66	21.51	0.26	6.18	0.33	97.49
1997	5.31	7.21	2.00	2.00	3.73	4.17	2.00	2.00	0.35	7.21	0.66	21.51	0.26	6.18	0.33	97.49
1998	4.78	7.49	2.00	2.00	3.35	4.75	2.80	2.80	0.31	7.49	0.66	21.51	0.26	6.18	0.33	97.49
1999	5.30	7.74	2.00	2.00	3.02	5.28	2.52	3.52	1.28	7.74	0.66	21.51	0.26	6.18	0.33	97.49
2000	4.77	7.97	2.00	2.00	2.72	5.75	4.27	4.17	1.15	7.97	0.66	21.51	0.26	6.18	0.33	97.49
2001	4.30	8.17	2.00	2.00	3.44	6.17	4.84	4.75	1.04	8.17	0.66	21.51	0.26	6.18	0.33	97.49
2002	3.87	8.35	1.80	2.80	3.10	6.56	5.36	5.28	0.93	8.35	0.66	21.51	0.26	6.18	0.33	97.49
2003	4.48	8.52	5.62	3.52	3.79	6.90	7.82	5.75	0.84	8.52	0.66	21.51	0.26	6.18	0.33	97.49
2004	4.03	8.67	7.06	4.17	5.41	7.21	11.04	6.17	0.76	8.67	0.66	21.51	0.26	6.18	0.33	97.49
2005	3.63	8.80	7.35	4.75	5.87	7.49	10.94	6.56	0.68	8.80	0.66	21.51	0.26	6.18	0.33	97.49

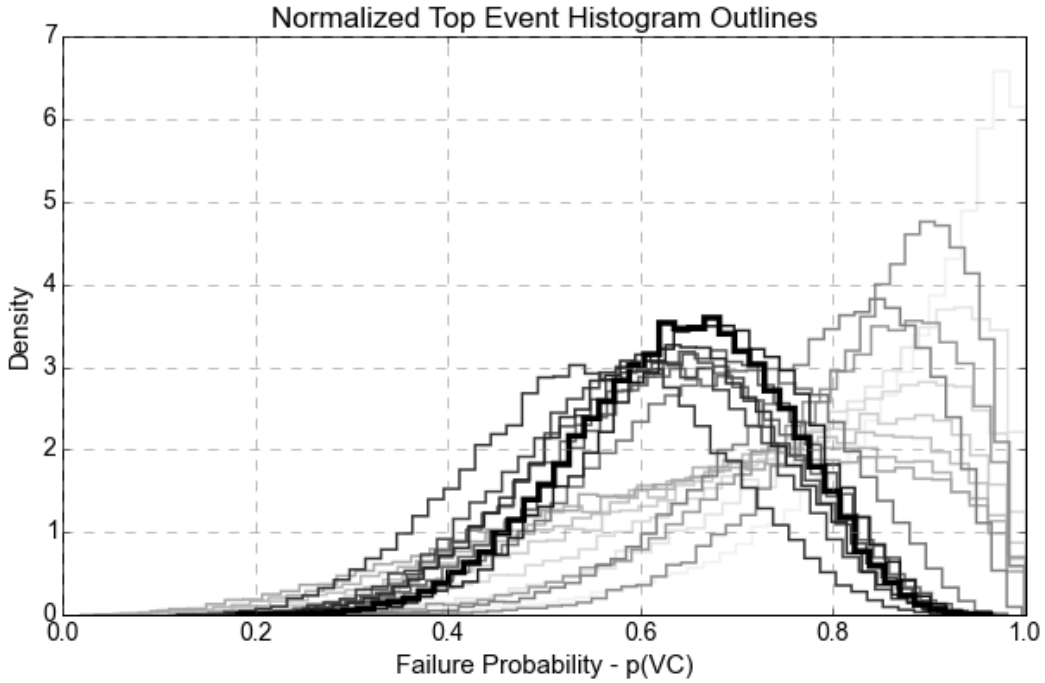


Figure 16 - Case 1 - Top Event Marginal Histograms

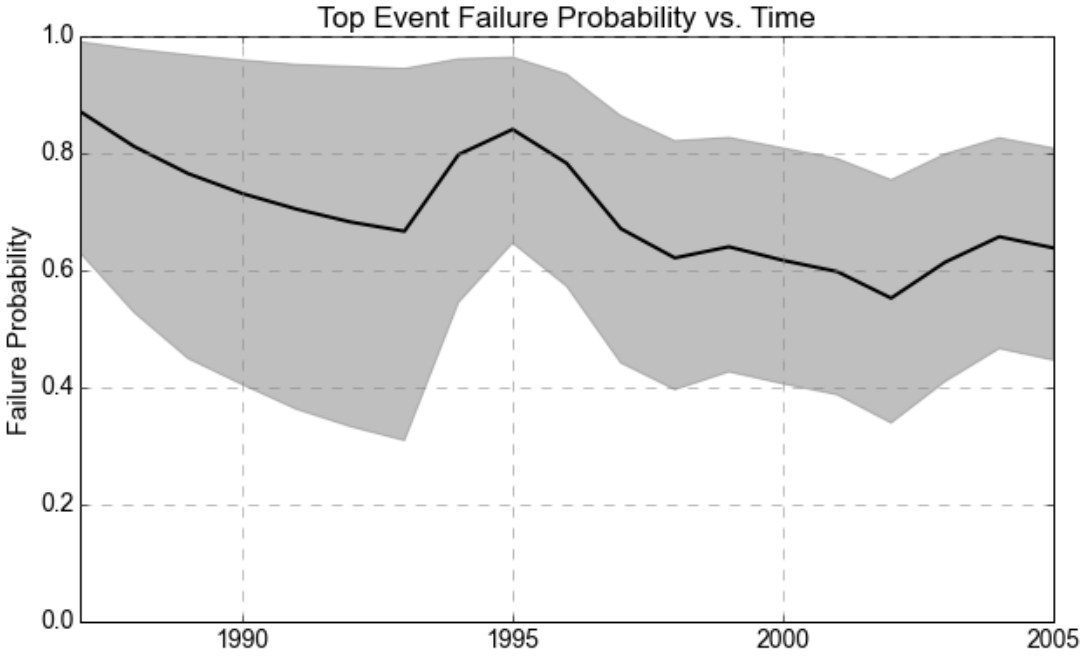


Figure 17 - Case 1 - Top Event Probability and 95th Percentile vs. Time

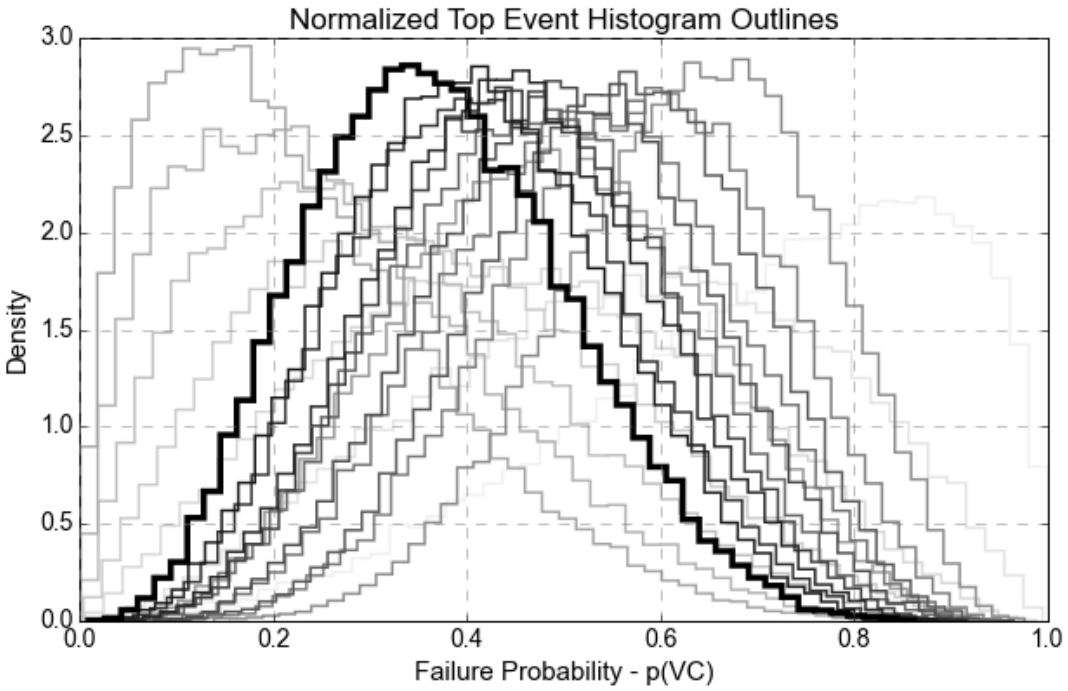


Figure 18 - Case 2 - Top Event Probability Marginal Histograms

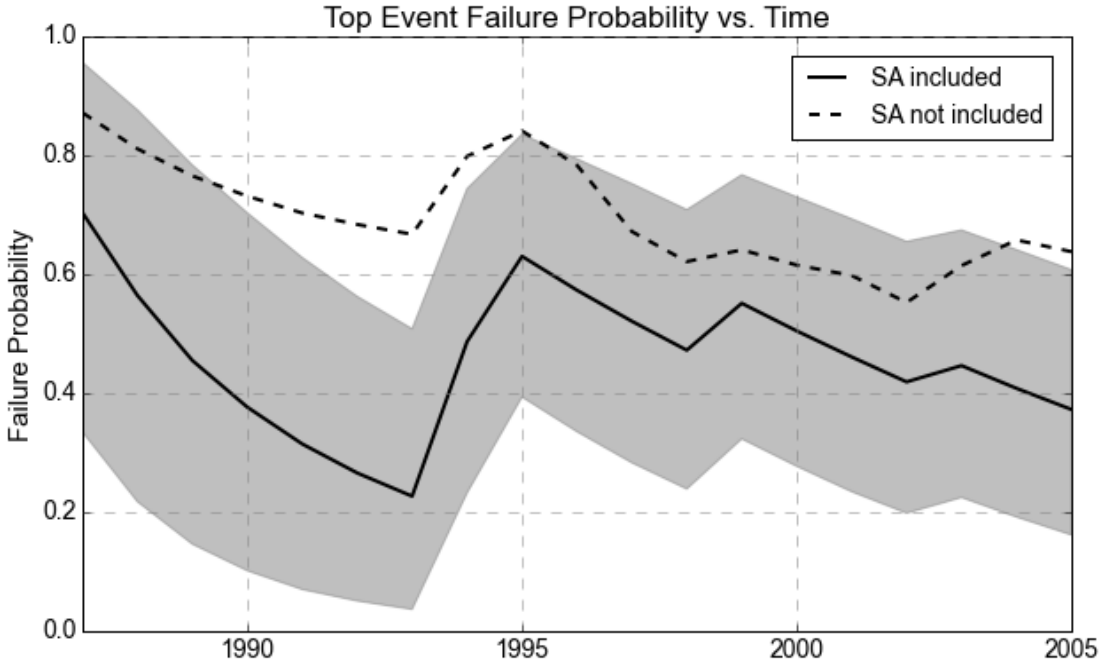


Figure 19 - Case 2 - Top Event Probability Mean and 95th Percentiles

2.7 Conclusion

This chapter presents a Bayesian analysis of the fault tree developed based on the BP Texas City ISOM unit, at which a catastrophic failure occurred in 2005. It is shown that fault tree analysis combined with Bayesian updating can be used as part of dynamic risk assessments and ongoing risk surveillance of plant processes. A summary of key findings is provided below.

- The process of developing the fault tree enables a safety analyst to learn about the system and the potential root causes that may lead to the undesired top event.
- A qualitative analysis of the fault tree establishes the minimal cut-sets and allows the analyst to determine the relative importance of the various parts of the fault tree from a qualitative perspective.
- The quantitative analysis incorporates failure and occurrence rates for the primary faults in the system and allows for an order of magnitude estimate of the top event probability. This provides a basis for comparing incremental improvements in safety designs.
- Bayesian parameter estimation establishes how parameter uncertainty changes with time as plant specific failure data is collected.
- Longer term failure data associated with processing plants can be modeled using Bayesian state space models such as PEWMA, which allows discounting of older data points.
- Qualitative and quantitative fault tree analysis can help establish a basis for continuous monitoring of safety systems and design improvements.

2.8 References

Atwood, C.I, LaChance, J. L., Martz, H.F., Anderson, D.J., Englehardt, M., Whitehead, D., Wheeler, T. (2003). Handbook of Parameter Estimation for Probabilistic Risk Assessment, NUREG/CR-6823. Sandia National Laboratories, U.S. Nuclear Regulatory Commission Office of Nuclear Regulatory Research.

Bier, V.M. and Mosleh, A. (1990), The Analysis of Accident Precursors and Near Misses: Implications for Risk Assessment and Risk Management, Relability Engineering and System Safety, 27, 91-101

Bobbio, A., Portinale, L., Minichino, M., Ciancarmerla, E. (2001), Improving the analysis of dependable systems by mapping fault trees into Bayesian networks, Elsevier Reliability Engineering and System Safety 71 (2001) 249-260

Brandt, P. T., Williams, J.T., Fordham, B.O and Pollins, B. (2000). Dynamic Modelling For Persistent Event Count Time Series. American Journal of Political Science 44(4): 823-843.

BP (2005). Fatal accident investigation report, final report. Texas City, TX, www.bpresponse.org.

CSB (2007). Final Investigation Report: Refinery Explosion and Fire. U.S. Chemical and Hazard Investigation Board, Paper No. ISOPE-2007-SBD-03, www.csb.gov.

Det Norske Veritas (2002). Offshore Reliability Data Handbook, Fourth Edition. Det Norske Veritas.

Dianous VD, Fievez C. ARAMIS project: a more explicit demonstration of risk control through the use of bow-tie diagrams and the evaluation of safety barrier performance. Journal of Hazardous Materials 2006;130:220–233.

Holloway, J. P., (2011) Time series analysis of count data with an application to the incidence of cholera, Master's Thesis, Department of Statistical Sciences, University of Cape Town

Kalantarnia, M. (2011), Dynamic Risk Assessment Using Accident Precursor Data and Bayesian Theory, Master's Thesis, Faculty of Engineering & Applied Science, Memorial University of Newfoundland

Khan, F.I, Amyotte, P.R. (2005). Modeling of BP Texas City refinery incident. Journal of Loss Prevention in the Process Industries Volume 20, Issues 4-6, July-November 2007, Pages 387-395.

Khan, F., and Abbasi, S. (1998), Techniques and Methodologies for Risk Analysis in Chemical Process Industries, Journal of Loss Prevention in Process Industries, V. 11, 261-277.

Khakzad, N., Khan, F., Amyotte P. (2011), Safety analysis in process facilities: Comparison of fault tree and Bayesian network approaches, Elsevier Reliability Engineering and System Safety 96 (2011) 925-932

Khakzad, N., Khan, F., Amyotte P. (2013), Dynamic safety analysis of process systems by mapping bow-tie into Bayesian network, Elsevier Process Safety and Environmental Protection Journal

Lindhe, A., Rosen, L., Norberg, T.,Bergstedt O (2009), Fault tree analysis for integrated and probabilistic risk analysis of drinking water systems, Elsevier water research 43 (2009) 1641-1653

Marquez D, Neil M, Fenton N. Improved reliability modeling using Bayesian networks and dynamic discretization. Reliability Engineering and System Safety 2010;95:412–425.

Meel S., Seider, W.D. (2006). Plant-specific dynamic failure assessment using Bayesian theory., Chemical Engineering Science, Volume 61, Issue 21, 6 November 2006, Pages 7036–7056.

Rangel, L. E., Levesque, F., How did Fukushima-Daiichi core meltdown change the probability of nuclear accidents? 2012, <hal-00740684>

Shafaghi, A. (2008), Equipment Failure Rate Updating – Bayesian Estimation. Elsevier, Journal of Hazardous Materials, Volume 159, Issue 1, 15 November 2008, Pages 87–91.

P. N. Thodi, F. I. Khan, Mahmoud R. Haddara (2010). The Development of Posterior Probability Models in Risk-Based Integrity Modeling. Risk Analysis, Vol. 30, No. 3, 2010.

Vesely, W.E., Goldberg, F.F., Roberts, N.H., Haasl, D.F. (1981). Fault Tree Handbook, NUREG-0492. Systems and Reliability Research Office of Nuclear Regulatory Research, Washington D.C., 20555.

Vose, D., Risk Analysis – A Quantitative Guide, Wiley, 2008

3 BAYESIAN UPDATING OF MATERIAL BALANCE PARAMETERS USING MCMC

3.1 Introduction

In this Chapter, Bayesian updating is applied to estimate the uncertainty associated with the parameters in the general hydrocarbon material balance equation. Gaussian distributions are used to model prior information and likelihood error in the implemented Bayesian updating models. Because the material balance equation is a non-linear forward model, the posterior distribution is not closed-form and requires sampling based solution methods. A Markov Chain Monte Carlo (MCMC) Metropolis algorithm is implemented to solve for the posterior distribution at each time step. A structured grid approximation of the posterior is also implemented to allow for calibration of the MCMC algorithm. Since a public data set is not available for this study, a synthetic data set is developed using the Eclipse reservoir simulation software. This provides the additional benefit that it provides the ability to directly assess the accuracy of the Bayesian updating models. A case study is provided to present the results obtained from running the structured grid model and the MCMC model on the synthetic data set.

3.2 Research Objectives

- Implement the general material balance equation as a forward model for use in a Bayesian updating framework.
- Implement structured grid and MCMC based Bayesian updating models.
- Generate a synthetic data set using the Eclipse reservoir simulator, to which the Bayesian updating models can be applied.
- Run both structured grid and MCMC Bayesian updating models on the synthetic data set and assess accuracy of the Bayesian updating models for material balance parameter estimation.
- Assess influence of Likelihood error on the rate of data assimilation of the posterior distribution.
- Assess impact of both consistent and random measurement noise in measured data set.
- Assess convergence properties of the implemented MCMC model by comparison to the structured grid approach and by carrying out a graphical convergence analysis.

3.3 Literature Review and Background

3.3.1 Bayesian Updating of Multivariable Forward Models

Bayesian updating is a recursive parameter estimation technique based on Bayes theorem (Eq. 36). A summary of the terms associated with Bayes theorem is provided in Table 8. Central to Bayesian updating is the implementation of a likelihood function, which probabilistically quantifies the goodness-of-fit associated of the parameters in a forward model. Comprehensive reviews of the technique can be found in Tarantola (2001) and Oliver (2008). Bayesian updating is an alternative to deterministic model fitting techniques such as linear regression, which simply returns optimal point estimates based on minimizing a loss function. Tarantola (2001) focuses mostly on linear inverse problems, but addresses both grid and sampling based solution strategies to solve for the posterior distribution. If the forward model is non-linear or the prior and likelihood distributions are non-conjugate, the posterior distribution is not be closed-form and sampling based techniques such as Markov Chain Monte Carlo (MCMC) are required. Traditional Bayesian updating techniques are practical for forward models involving a small intermediate number of input variables. For forward models requiring assimilation on a very large number of variables, such as the case of history matching of reservoir simulation or weather prediction models, modified Bayesian techniques such as the Ensemble Kalman filter are more suitable (Evensen, 2003).

$$f(x|y) = \frac{f(x)f(y|x)}{\int f(x)\, f(y|x)dx} \qquad \text{Eq. 36}$$

Table 8 - Bayesian Updating Equation Summary

Component	**Bayesian Nomenclature**	**Meaning**
$f(x\|y)$	Posterior Distribution	Probability of model parameters conditional to data
$f(y\|x)$	Likelihood Function	Probability of the data given model parameters
$f(x)$	Prior Distribution	Prior probability of model parameters
$\int f(x)\, f(y\|x) dx$	Normalizing Constant	Probability of the data

3.3.1.1 Prior Distribution

The prior distribution quantifies prior belief of the model parameters and is ideally based on expert opinion (Vose, 2008). In the case of material balance modeling, the expert can be a reservoir engineer or a geologist with a few years of experience. Reservoir engineers and geologists form prior opinions by analyzing information resulting from seismic surveys, core sampling, PVT testing and pressure transient analyses (Kelkar, 2002). If field specific data is sparse, it may also be necessary to rely on analog reservoir data. As such, prior information may only be available in the form of rough parameter ranges such as maximums and minimums for which uniform prior distributions are suitable. If belief about modal value(s) is also available, standard probability distributions such as the Gaussian or Lognormal distributions can be used to convey prior belief about parameters. The Gaussian prior distribution is expressed mathematically in (Eq. 37), where x is an array of mean model variables, C_x is the covariance matrix and μ_0 is the prior mean of the model variables.

$$f(x) = \left[\frac{1}{(2\cdot\pi)^{n/2}(\det(C_x))^{1/2}}\right]\cdot exp\left[-\frac{1}{2}(x-\mu_0)^T C_x^{-1}(x-\mu_0)\right] \quad \text{Eq. 37}$$

3.3.1.2 Likelihood Function

The likelihood function comprises of a mathematical forward model that produces measurable output(s) and probabilistic loss function that quantifies the likelihood of forward model outputs relative to the measured data points. The likelihood function is associated with variance due to measurement and/or theoretical/model discrepancies. Measurement errors stem from inherent randomness in the measurement equipment due to for instance voltage fluctuations or varying instrumentation response to changing conditions. Theoretical errors, on the other hand, exist due to simplifying assumptions associated with the forward model such as numerical discretization. A common approach is to model the likelihood error as a combined measurement/theoretical error (Oliver, 2008), which is expressed in Eq. 38.

$$y - g(x) = \epsilon_{theoretical} + \epsilon_{measuemrent} = \epsilon_{total} \quad \text{Eq. 38}$$

A Bayesian likelihood function can be viewed as a probabilistic loss function that produces uncertainty ranges for the model parameters. If multiple data points are available for assimilation, the likelihood function can be evaluated sequentially and can be expressed as (Eq. 39) provided each updating step i is statistically independent.

$$f(y|x) = \prod_{i=1}^{n} f(y_i|x_i) \quad \text{Eq. 39}$$

A likelihood distribution is generated by evaluating the likelihood function over ranges of model input parameters. Figure 20 illustrates how likelihood variance/error increases the spread of the distribution. The peak of the likelihood distribution is referred to the Maximum Likelihood Estimate (MLE). The variance of the likelihood function reduces with each Bayesian updating step as long as each data point confirms the same model parameters. Data points conflicting with previously inferred parameter values will, however, cause likelihood variance to increase.

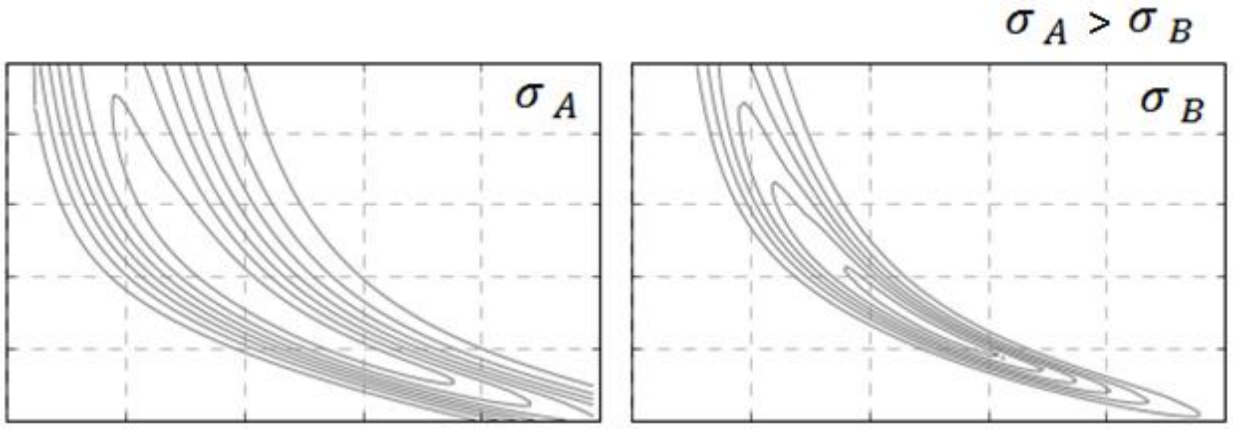


Figure 20 - Effect of standard deviation on Likelihood Distribution

The Bayesian updating framework allows any type of probability distribution to be used as a likelihood function, however, errors are commonly assumed to be Gaussian, which results in Eq. 40. If the variance associated with each Bayesian updating step is assumed to be uncorrelated and constant, the covariance matrix C_y with identical elements σ_y on the diagonal results (Eq. 41 and Eq. 42). Algebraically, the evaluation of the Gaussian likelihood up to updating step n reduces to Eq. 43. For the case of Gaussian likelihood functions, the measured data is typically set equal to the mean of the distribution. This is reasonable for situations where one believes the measured data point to be associated with the highest likelihood. The reference point, however, does not need to coincide with the mean of the likelihood function and can be offset to

accommodate a situation where the maximum likelihood is believed to occur for a value smaller or larger than the measured data point.

$$f(y|x) = \frac{1}{(2\pi)^{n/2}\left(det\left(C_y\right)\right)^{1/2}} exp\left(-\frac{1}{2}\left((g(x)-y)^T {C_y}^{-1}(g(x)-y)\right)\right) \quad \text{Eq. 40}$$

$$C_y = \begin{bmatrix} {\sigma_{y,1}}^2 & 0 & 0 & 0 \\ 0 & {\sigma_{y,2}}^2 & 0 & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & {\sigma_{y,n}}^2 \end{bmatrix} \quad \text{Eq. 41}$$

$${\sigma_{y,i}}^2 = {\sigma_{y,1}}^2 = {\sigma_{y,2}}^2 = \cdots = {\sigma_y}^2 \quad \text{Eq. 42}$$

$$f(y|x) = \frac{1}{(2\pi)^{n/2}\left(det\left(C_y\right)\right)^{1/2}} \cdot \exp\left(-\frac{1}{\sigma^2}\sum_{i=1}^{n}\left(g(x_i)-y_i\right)^2\right. \quad \text{Eq. 43}$$

3.3.1.3 Posterior Distribution

The posterior distribution $f(x|y)$ is the solution to the Bayesian updating problem and incorporates both prior and likelihood information (Tarantola, 2010). The Posterior distribution is Gaussian and closed-form if the prior and likelihood functions are both Gaussian and the forward model is linear (Oliver, 2008). The closed-form linear Gaussian solution to the inverse problem has found great utility in for instance the Kalman filter, which is often used as part of machine learning topics subjects such as signal processing and robotics (Särkkä, 2013). If the forward model is non-linear or the prior and likelihood distributions are non-conjugate, the posterior distribution is non-Gaussian and requires integration by sampling. An example of a prior, likelihood and posterior distribution is shown as a two-variable contour plot in (Figure 21). As demonstrated in this figure, the posterior behaves like a compromise between the prior and the likelihood function. Provided the data that is incorporated into the analysis confirms the same model parameter values, the variance of the likelihood function will decrease with each updating step and eventually dominate the posterior distribution. The rate at which the posterior assimilates the likelihood distribution depends on the relative variance between the likelihood function and the prior. If the prior is associated with low variance relative to the likelihood function, the posterior will be slow to incorporate information from the data. Conversely, if the prior is associated with high variance relative to the likelihood function, the posterior will incorporate information from the data faster.

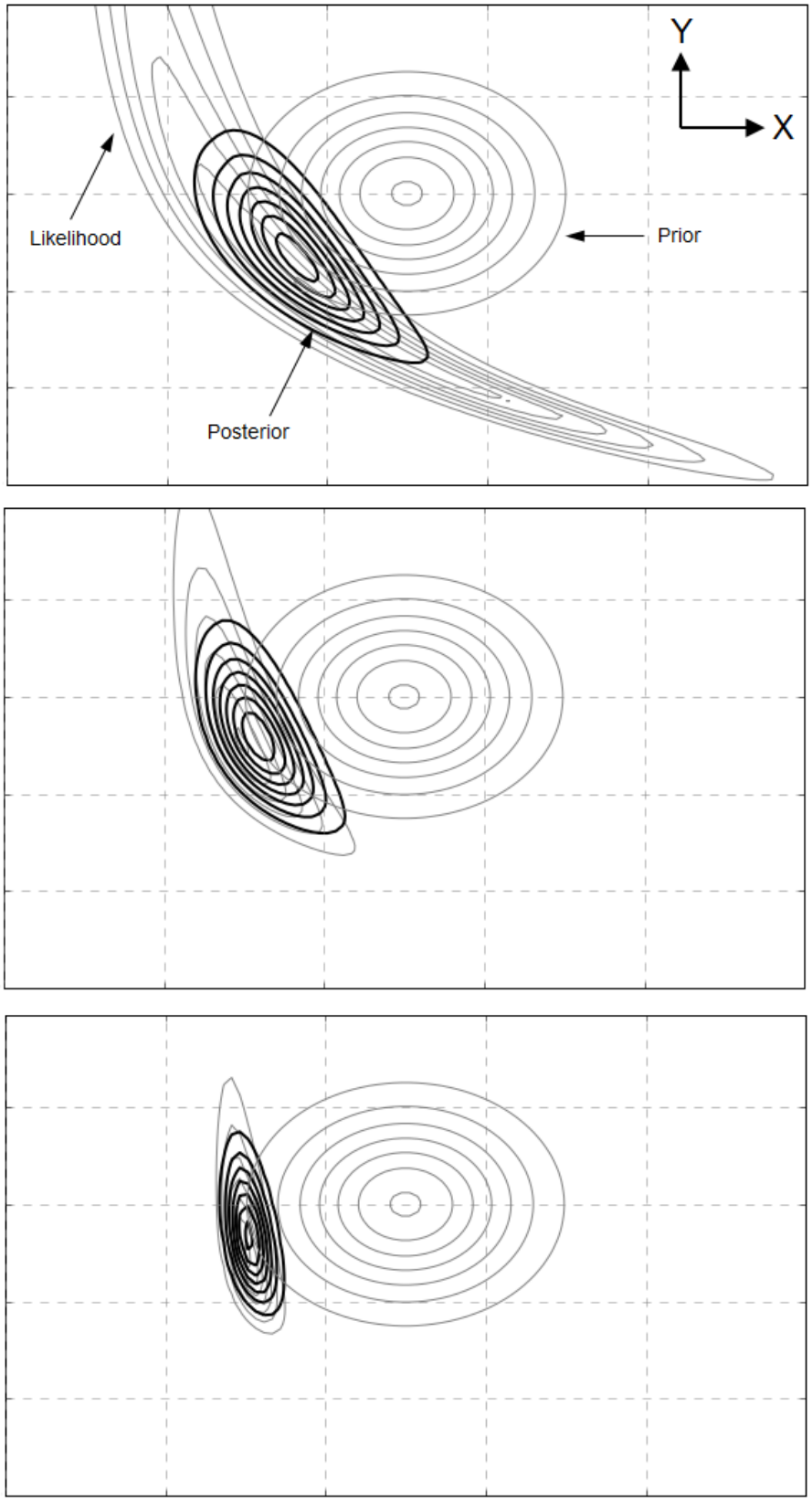


Figure 21 - Prior, Likelihood and Posterior distributions vs. Bayesian Updating Steps

3.3.1.4 Bayesian Updating on a Structured Grid

The simplest approach for approximating the posterior distribution $f(x|y)$ involves creating a structured grid of input parameters and solving the forward model at each grid location (Figure 22). This approach is useful for two-variable problems, as it allows the resulting prior, likelihood and posterior distributions to be visualized on three-dimensional surface and contour plots. A drawback associated with the structured grid method is that it wastes computational effort and array memory because it requires the forward model $g(x)$ to be evaluated in both low and high probability regions. The structured grid approach also requires a sufficiently fine grid to capture specific characteristics of the posterior distribution, such as multi-modal peaks. For problems involving more than two variables, the structured grid solution becomes intractable because of the associated computational cost.

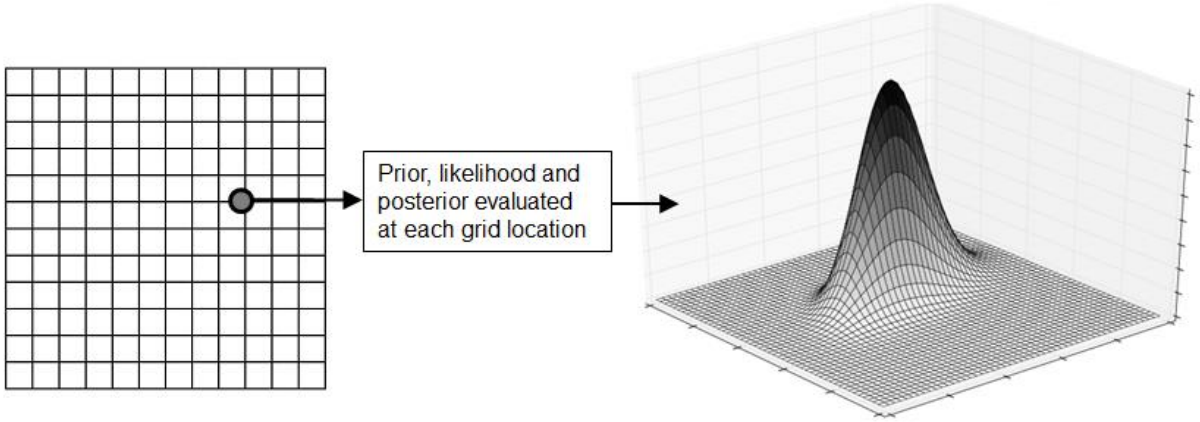


Figure 22 - Structured grid solution strategy

3.3.1.5 Bayesian Updating using Markov Chain Monte Carlo

To solve higher-order problems, Monte Carlo based sampling techniques are required in order to reduce the required number of forward model evaluations. A standard Monte Carlo algorithm may end up spending significant number of iterations sampling regions of low probability, which can in turn cause incorrect estimates of the posterior distribution. Markov Chain Monte Carlo (MCMC) is a more generally applicable strategy because it preferably returns samples from high probability regions and therefore samples the posterior more efficiently. When running an MCMC algorithm the end result is a Markov chain of parameter states.

3.3.1.6 Markov Chain Theory

A Markov Chain is a sequence of random variables, where the probability of the state at time $t+1$ depends only the preceding state at time t (Figure 23). As such, Markov Chains are dependent samples, compared to conventional Monte Carlo which draws independent samples. In the context of Markov Chains, time refers to the sample number, where the chain is thought of as being progressed forwards in time as the number of samples increases. Markov chain can be stated mathematically as Eq. 44, where $P(X_{i+1}|X_i)$ is known as a transition kernel and governs the probability of transitioning from one state to the next throughout the possible states of the system.

$$P(X_{t+1}|\, X_1, X_2, \dots, X_t) = P(X_{t+1}|X_t) \qquad \text{Eq. 44}$$

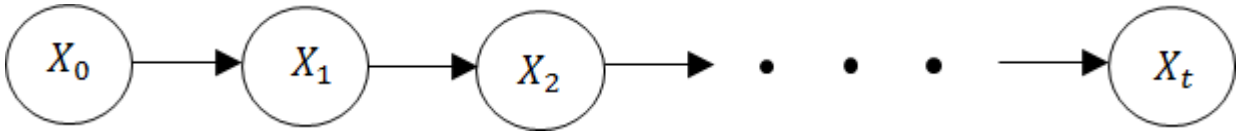


Figure 23 - Markov Chain

The ergodic theorem is concerned with the asymptotic convergence properties of Markov Chains as $t \to \infty$. For the purpose of outlining the concepts associated with the ergodic theorem, the discussion is here based on finite state spaces. This is sufficient for understanding the underlying properties and can be extended to general state spaces (Gamerman, 2002). For a finite state space, the transition kernel $P(X_{t+1}|X_t)$ is a stochastic matrix T_{ab}, containing the probabilities of transitioning between the discrete states of the system (Eq. 45).

$$T(a,b) = P(X_{t+1} = b|X_t = a) \quad for\ \forall(a,b) \in S \qquad \text{Eq. 45}$$

The ergodic theorem states that if the Markov Chain is stationarity, irreducible and aperiodic, then the resulting chain $(X_1, X_2, \dots, X_n)$ converges to the true mean of the dsitribution $E[h(x)]$ (Eq. 46 and Eq. 47) as the number of samples approaches infinity $t \to \infty$ (Gamerman, 2002).

$$E[h(x)] = \int h(x)f(x)dx \qquad \text{Eq. 46}$$

$$\frac{1}{N}\sum_t^n h(X_n) \to E[h(x)] \quad as\ \ t \to \infty \qquad \text{Eq. 47}$$

The target distribution π is stationary if Eq. 48 is satisfied (Gamerman, 2002). A stationary distribution π is time invariant, meaning that all samples in the Markov Chain are drawn from the same distribution.

$$\sum_{a} \pi(a)T(a,b) = \pi(b) \quad for\ \forall(a) \in S \qquad \text{Eq. 48}$$

A Markov Chain is irreducibile if the probability of reaching any state b from an arbitrary starting state a in a finite number of moves is greater than zero (Gamerman, 2002). This is formally stated in Eq. 49 and implies that the entire state space S can be reached by the Markov Chain.

$$P(X_t = b|X_0 = a) > 0 \quad for\ \forall(a,b) \in S \qquad \text{Eq. 49}$$

An irreducible Markov Chain (X_t) is aperiodic if for any state a the greatest common divisor (g.c.d.) of return times to any particular state a is equal to one (Eq. 50). A value greater than one implies that the Markov Chain gets stuck in cycles, which prevents exploration of the entire state space.

$$gcd\{t: P(X_i = a|X_0 = a) > 0\} = 1 \quad for\ \forall(a) \in S \qquad \text{Eq. 50}$$

3.3.1.7 Metropolis-Hastings Algorithm

The Metropolis algorithm (Metropolis, 1953) is a Markov Chain Monte Carlo algorithm that can be used to sample from a probability mass function π or to approximate the expected value of a function $E[h(x)]$. The algorithm was generalized by Hastings (1970) and named Metropolis-Hastings. The algorithm has been applied extensively in Bayesian Inference, because it only requires the posterior distribution $\pi(x)$ to be known up to the normalizing constant z (Eq. 51).

$$\pi(x) = \frac{\tilde{\pi}(x)}{z} \quad for\ z > 0 \qquad \text{Eq. 51}$$

The first step of the Metropolis-Hastings algorithm is to initialize the Markov Chain to an initial state $X_0 \in S$. The next candidate sample X_{t+1} is generated from a proposal distribution q (Eq. 52), which is centered at the current state X_t. Knowing the current state X_t and the candidate state Y_{t+1}, an acceptance ratio α is calculated (Eq. 53). Next, a random number $u \sim [0,1]$ is sampled from the uniform distribution. If the acceptance ratio α is greater than u, the candidate state Y_{t+1} is accepted and appended to the Markov Chain as X_{t+1}. If the the acceptance ratio α is less than u, the candidate state Y_{t+1} is rejected, causing the Markov Chain to remain in the current state, i.e., $X_{t+1} = X_t$. A summary is provided in Algorithm 3.

$$Y_{t+1} = q(Y_{t+1}|X_t) \qquad \text{Eq. 52}$$

$$\alpha(X_t, Y_{t+1}) = min\left\{1, \frac{\tilde{\pi}(Y_{t+1})q(X_t|Y_{t+1})}{\tilde{\pi}(X_t)q(Y_{t+1}|X_t)}\right\} \quad \text{Eq. 53}$$

In the original algorithm developed by Metropolis (1953), a symmetric proposal distribution is assumed ($q_{ab} = q_{ba}$), such that the decision of accepting a state is based only on the ratio of the probability of being in the two states (Eq. 54). An example of a symmetric proposal distribution q is the Gaussian distribution.

$$\alpha(X_t, Y_{t+1}) = min\left\{1, \frac{\tilde{\pi}(Y_{t+1})}{\tilde{\pi}(X_t)}\right\} \quad \text{Eq. 54}$$

To ensure that the MCMC algorithm converges to the target distribution π the requirements of irreducibility, stationarity and aperiodicity must be satisfied. In particular, if detailed balance (Eq. 55) is satisfied, then the Markov Chain converges asymptotically to the target distribution $\pi(x)$ as $t \to \infty$. In order to prove that detailed balance holds for the Metropolis algorithm, it is necessary to consider two separate cases. For simplicity of proof, denote a as the current state and b as the candidate state. Detailed balance can be expressed as Eq. 55, where $T(b, a)$ represents the Markov Chain transition kernel which quantifies the probability of transitioning from state a to state b. For the Metropolis algorithm in particular, the transition kernel can be expressed as Eq. 56.

$$\pi(a)T(a, b) = \pi(b)T(b, a) \quad for\ \forall(a, b) \in S \quad \text{Eq. 55}$$

$$T(b,a) = P(X_{t+1} = b | X_t = a) = q(b,a) min\left\{1, \frac{\pi(b)}{\pi(a)}\right\} \quad \text{Eq. 56}$$

For the case of a rejected candidate state $a = b$, detailed balance is trivially satisfied. To prove that detailed balance holds for the case of an accepted candidate sample $a \neq b$, the transition probability equation is applied to the left hand side of the detailed balance equation (Eq. 57). Next, the transition probability equation is applied to the right-hand side of the detailed balance equation (Eq. 58). Because the min operator is symmetric and because a symmetric proposal distribution $q(a,b) = q(b,a)$ is used, detailed balance is satisfied (Eq. 59).

$$\pi(a)T(a,b) = \pi(a)q(a,b) min\left\{1, \frac{\pi(b)}{\pi(a)}\right\} = q(a,b) min\{\pi(a), \pi(b)\} \quad \text{Eq. 57}$$

$$\pi(b)T(b,a) = \pi(b)q(b,a) min\left\{1, \frac{\pi(a)}{\pi(b)}\right\} = q(b,a) min\{\pi(b), \pi(a)\} \quad \text{Eq. 58}$$

$$q(b,a) min\{\pi(b), \pi(a)\} = q(a,b) min\{\pi(a), \pi(b)\} \quad \therefore \quad \text{Eq. 59}$$

Finally, it is necessary to assess whether irreducibility and aperiodicity criterions are satifisfied. In terms of irreducbility, the proposal distribution must be able to draw samples from the entire parameter space S over which π is defined. Practically speaking, this means that proposal distribution must be defined over the entire parameter space S. The Metropolis algorithm is Aperiodic because it allows for rejection of candidate samples.

Algorithm 3 - MCMC - Metropolis Algorithm

1 $Select\ Starting\ Point\colon\ X_0$

2 $for\ i = 1\ to\ n_{\max samples}$

3 $Sample\ candidate\ point, Y_{t+1} \sim q(Y_{t+1}|X_t)$

4 $Sample\ random\ number, u \sim [0,1]$

5 $Calculate\ acceptance\ ratio, \alpha(X_t, Y_{t+1}) = min\left\{1, \frac{\tilde{\pi}(Y_{t+1})}{\tilde{\pi}(X_t)}\right\}$

6 $if(\alpha(X_t, Y_{t+1}) > u)$

7 $Accept\colon\ X_{t+1} = Y_{t+1}$

8 $Else$

9 $Reject\colon\ X_{t+1} = X_t$

10 END

3.3.1.8 MCMC Convergence Diagnostics

Convergence analysis is an important and necessary aspect of MCMC sampling. If the chain does not properly sample from the posterior distribution, important quantities such as the mean, mode and variance will be incorrectly estimated. A method for assessing whether the MCMC chain is converging to the posterior distribution is to run several chains in parallel using different initial values and comparing inter-chain results. If all chains provide the same posterior quantities, convergence can be assumed with reasonable confidence. Commonly applied graphical techniques for assessing mixing and convergence properties are time series, running means and autocorrelation plots (Gamerman, 2002). Autocorrelation (Eq. 60) reveals 'non-randomness' in data, such as trends or clustering and should for MCMC sampling ideally appear as random noise around a value of zero. A time series plot of sample values vs. MCMC iteration number should ideally show that the algorithm is thoroughly sampling the posterior region. A running mean plot shows calculated sample mean .vs. MCMC iteration and should converge to a stable value as the chain length increases. Quantitatively determining MCMC convergence is challenging, however, advanced methods involving the assessment of inter-chain/between-chain variance and spectral analysis have been published by for instance Geweke (1992). The number of samples required before the Markov chain converges to the stationary distribution is referred to the burn-in period. The burn-in samples should be removed before computing summary statistics such as sample mean and variance (Eq. 61). In Figure 24, this would entail removing samples $0\ to\ m$ from the chain.

$$r_k = \frac{\sum_{i=1}^{N-k}(Y_i - \bar{Y})(Y_{i+k} - \bar{Y})}{\sum_{i=1}^{N}(Y_i - \bar{Y})^2}$$ Eq. 60

$$E[X] \approx \frac{1}{n-m}\sum_{i=m+1}^{n} X_i$$ Eq. 61

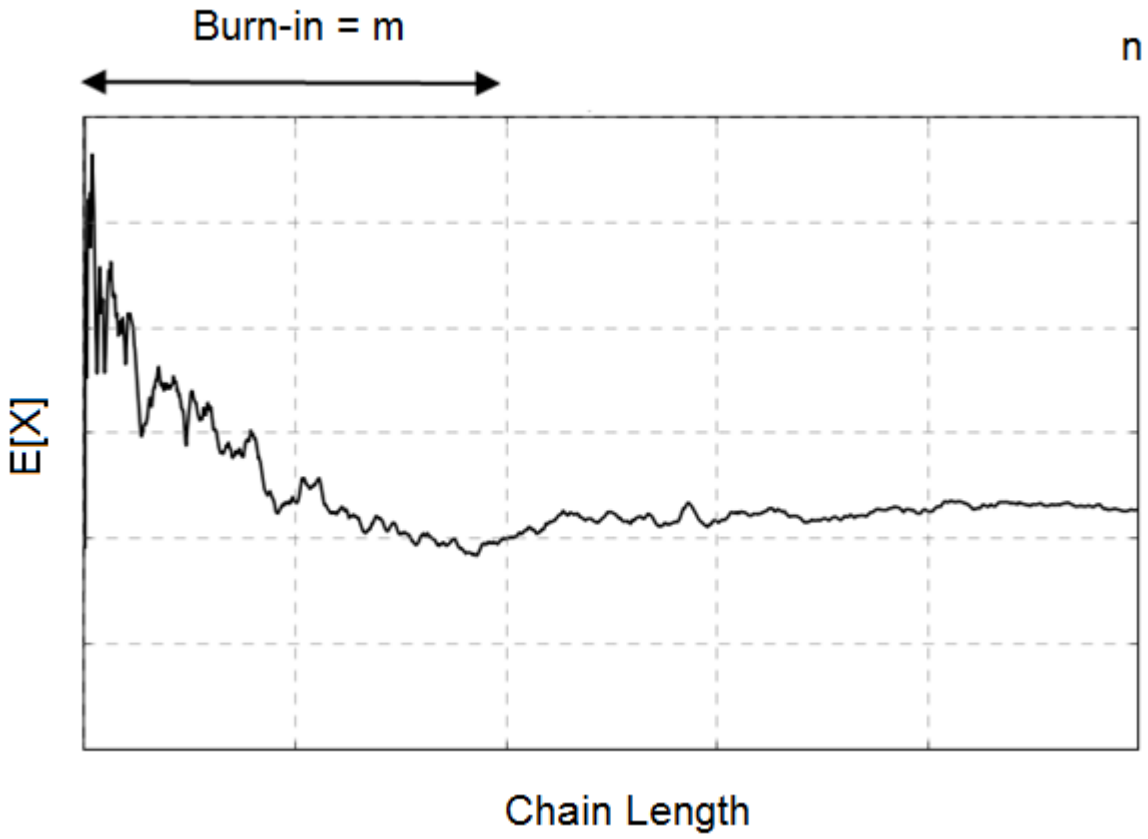


Figure 24 - Burn-in

3.3.1.9 MCMC Output Analysis

For sampling based solutions involving three or more variables, the solution is usually represented as marginal histograms with associated means and variances. A challenge with a marginal histogram is that it represents an orthogonal projection into the solution space. As such, correlation between variables is potentially masked in the marginal histograms and the variance may therefore appear to be higher than it actually is. Pairwise scatter plots of the Markov chains can be used to assess correlation effects and overcomes the limitation associated with marginal histograms. A useful summary statistic that reveals linear correlation between variables is Pearson's correlation coefficient (Eq. 62). This coefficient can can be analyzed together with marginal histograms and scatter plots as shown in Figure 25. For Bayesian updating problems, it is also useful to plot the posterior sample mean (Eq. 63) and percentiles at each updating step t to provide a basis for understanding how the collected data is affecting the posterior distribution through time (Figure 26).

$$\rho_{XY} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \quad \text{Eq. 62}$$

$$E[X_t] \approx \frac{1}{n}\sum_{i=1}^{n} X_{t,i} \quad \text{Eq. 63}$$

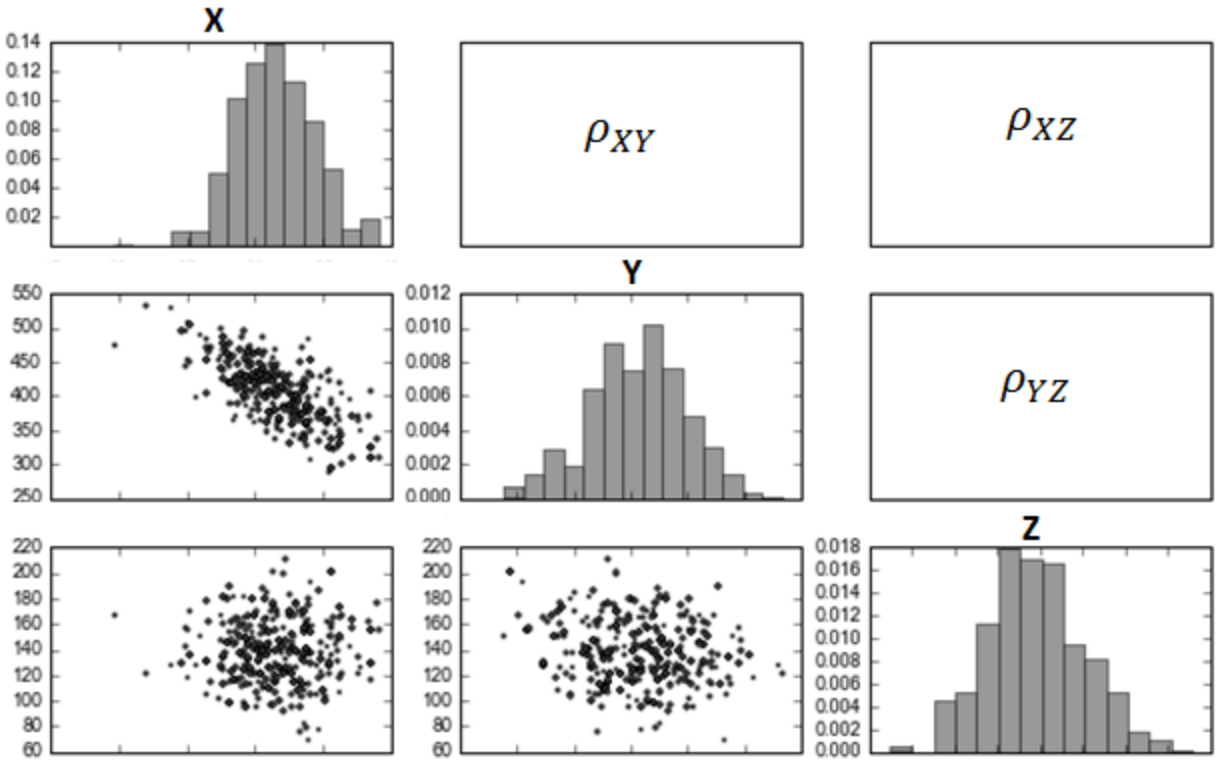


Figure 25 - Posterior Diagnostics Plot

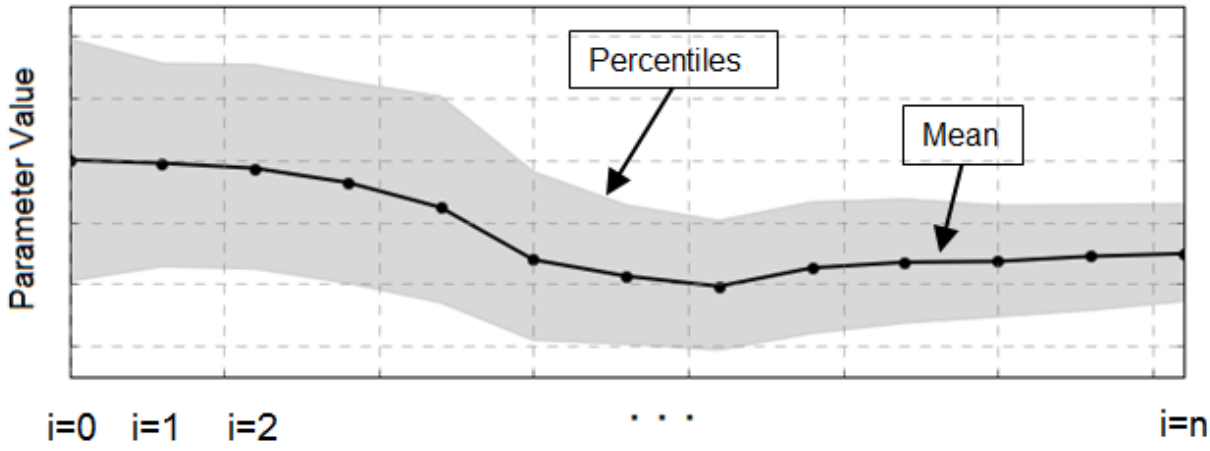


Figure 26 - Mean and percentiles

3.3.1.10 Material Balance

Material balance is a standard reservoir engineering tool often used in conjunction with more advanced techniques such as reservoir simulation to estimate original hydrocarbons in place and to quantify drive mechanisms. It assumes that the reservoir can be modeled as a compressible tank with average pressures and rock properties throughout (Dake, 2001), which is only approximately true for any reservoir. For tight reservoirs with low permeability, large pressure gradients will exist and prevents usage of the material balance technique. A visual representation of the material balance technique is provided in (Figure 27).

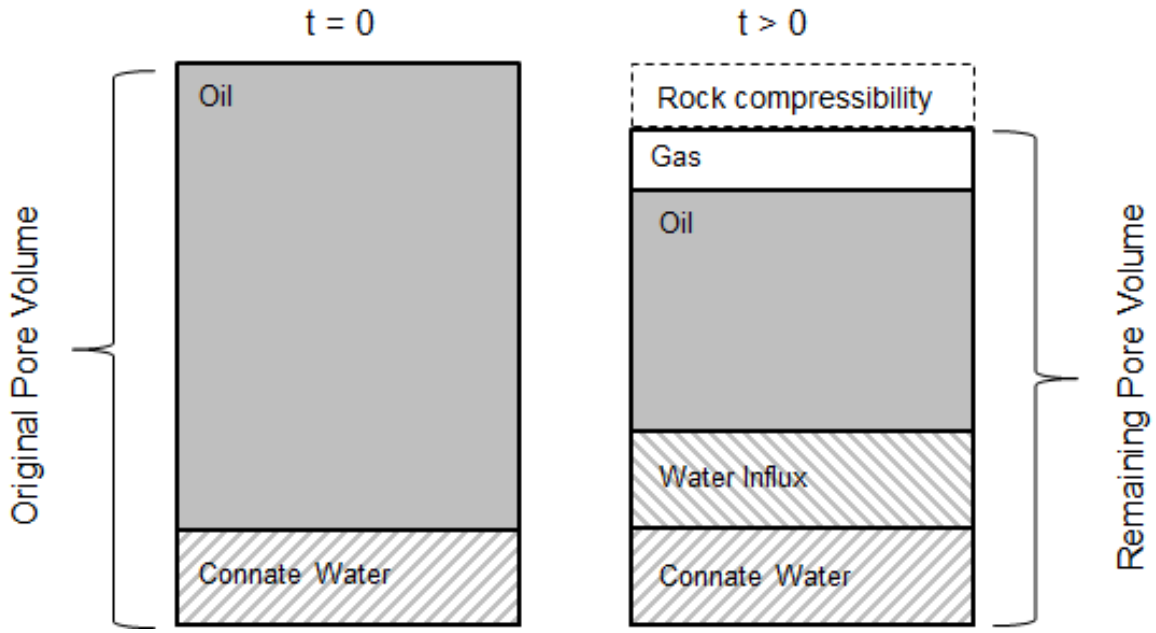


Figure 27 - Hydrocarbon material balance summary

By combining the drive effects associated with single phase expansion of oil/water/gas, liberation of solution gas, gas cap expansion, connate water expansion, pore compaction and aquifer influx, Dake (2001) formulates the material balance equation in terms of cumulative volumes as Eq. 64. Note that all volumetric terms are stated in terms of underground volumes, which is why the aquifer influx term W_e does not contain a water formation volume factor.

$$N_p\left[B_o + \left(R_p - R_s\right)B_g\right] + W_p B_w = N\left[(B_o - B_{oi}) + (R_{si} - R_s)B_g\right] + mNB_{oi}\left(\frac{B_g}{B_{gi}} - 1\right) + \frac{(1+m)NB_{oi}\left(c_w S_w + c_f\right)\Delta p}{1 - S_{wc}} + W_e \quad \text{Eq. 64}$$

Aquifer response cannot be directly measured and a model is therefore required to estimate the influx. Several aquifer models have been published in literature, with varying levels of rigor in terms of geometrical representation and transient behavior. Here, a Fektovich type analytical aquifer model (Fektovich, 1971) is implemented because of its generality. The Fetkovich aquifer model assumes that the reservoir-aquifer system behaves like a two-tank system (Figure 28) and that the reservoir pressure remains constant over each time step while the aquifer pressure varies. The derivation starts with defining the aquifer influx equation as a function of the aquifer index and the drawdown between the aquifer and reservoir (Eq. 65). Next, the concepts of total compressibility (Eq. 66) and isothermal compressibility (Eq. 67) are combined to obtain a separable differential equation that can be integrated for pressure and time (Eq. 68). Algebraic manipulation leads to closed form equations for aquifer pressure and aquifer influx as a function of time (Eq. 69 and Eq. 70).

$$(dW/dt) = J_w \cdot (p_a - p_r)$$ Eq. 65

$$c_t = c_w + c_r$$ Eq. 66

$$c_t = -(1/W) \cdot (dW/dp)$$ Eq. 67

$$\int_{p_a(0)}^{p_a(t)} dp/(p - p_r) = -\int_0^t (J_w/c_t \cdot W) \cdot d\tau$$ Eq. 68

$$p_a(t) = p_r + (p_a(0) - p_r) \cdot exp\left(-\frac{J_w}{c_t \cdot W_i} \cdot t\right)$$ Eq. 69

$$\Delta W(t) = W_e = c_t \cdot W_i \cdot \left(p_{a,i} - p_r(t)\right) \cdot \left(1 - exp\left(-\frac{J_w \cdot t}{c_t \cdot W_i}\right)\right)$$ Eq. 70

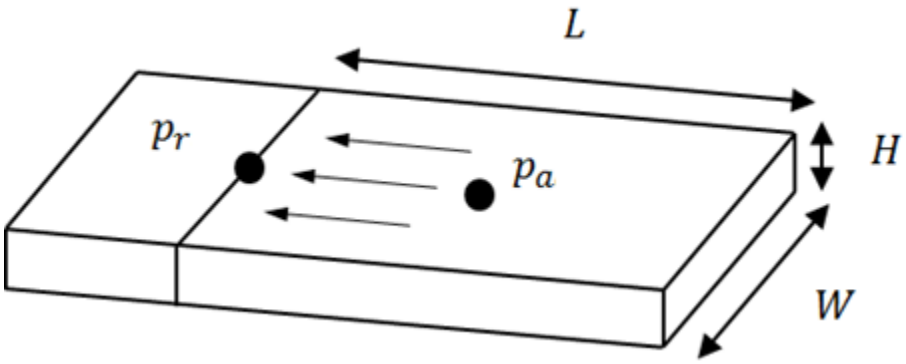


Figure 28 - Aquifer Model

3.3.1.11 Review of Material Balance Parameter Fitting

Van Everdingen et. al. (1953) presents a regression based technique for estimating material balance parameters. McEwen (1961) builds on the work by Van Everdingen et. al. (1953) and develops a regression methodology that better handles noise in the pressure data, but does not characterize uncertainty in the solution. Fair (1994) provides regression based material balance modeling that includes summary statistics and confidence intervals for the estimated parameters. Sills (1996) reports a regression based material balance regression technique similar to that of McEwen (1961) that is less sensitive to pressure noise due to a reduction in the number of regression parameters. Confidence intervals for resulting material parameters are, however, not provided in the analysis. The first attempt at fully characterizing uncertainty of the material balance parameters in a Bayesian framework is provided by Ogele (2006), who proposes a grid based Bayesian inversion strategy for the hydrocarbon material balance equation using two unknowns and Gaussian distributions. The structured grid solution presented in Ogele's work allows for three-dimensional visualization of the prior, likelihood and posterior, which is only practical for material balance problems involving two unknown variables. Aprilla (2006) presents a similar analysis to that of Ogele (2006), but introduces a third variable and thereby demonstrates the challenge associated with calculating and summarizing grid solutions in higher dimensions. Finally, Ottah (2015) presents a sampling based methodology for matching aquifer size using particle swarms. The particle swarm method generates an ensemble of solutions making it possible to estimate the uncertainty bandwidth and confidence intervals for the model parameters.

3.4 Model Implementation

3.4.1 Python Code Overview

The following section provides an overview of the Bayesian updating models that are implemented using the Python scripting language (Appendix B). Produced oil/gas/water (N_p, G_p, W_p) volumes and PVT data (B_o, B_g, B_w, R_s) are treated as deterministic constants in the material balance model. Subsurface reservoir quantities such as original oil/gas in place (N, G) and aquifer characteristics (W_i, J_w) are treated as random variables. The first Bayesian model evaluates prior, likelihood and posterior distributions on a structured grid. The second model implements an MCMC-Metropolis algorithm. Figure 29 shows the overall flow of the code for the grid based and MCMC based techniques. The grid based and MCMC based model have separate main routines, where the required function calls are made and solution data is summarized. Both the grid and MCMC based techniques share the same likelihood and material balance functions. In order to calculate a likelihood value for each combination of material balance parameters, the material balance model is reverse for pressure at each time step using a Newton-Raphson algorithm.

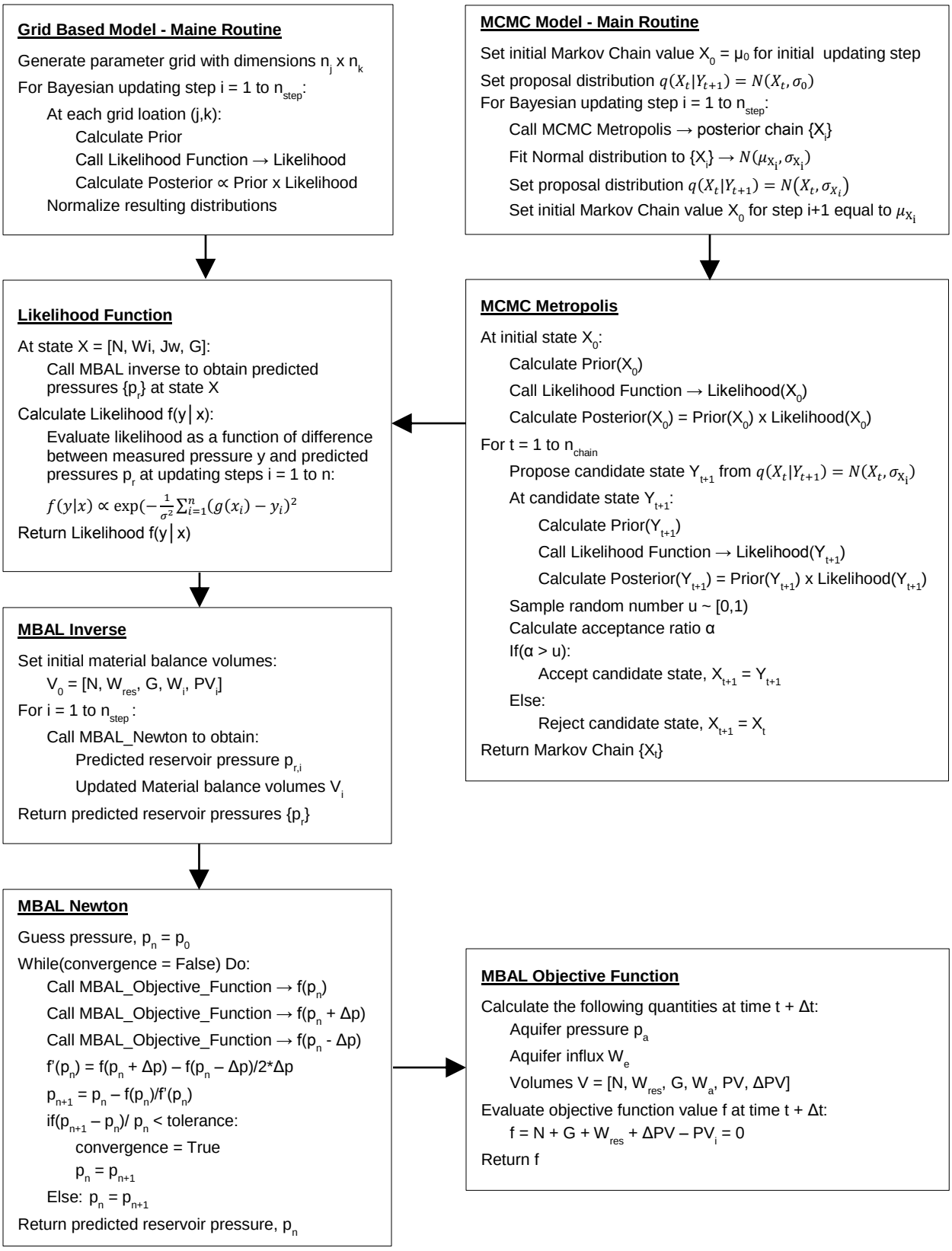


Figure 29 - Overview of implemented Python functions

3.4.2 Proposal Distribution

The proposal density is a multivariate Gaussian distribution centered at the current state X_t (Eq. 71). At each assimilation step i, the standard deviation of the proposal distribution is obtained by fiting a Gaussian distribution to the posterior output from the Metropolis algorithm at the previous assimilation step $i-1$ (Eq. 72). This ensures that the proposal distribution adapts to the variance of the posterior distribution throughout the Bayesian updating steps. The initial Markov chain value at each updating step is set equal to the mean of the posterior distribution at the previous updating step (Eq. 73).

$$Proposal\ distrbution, updating\ step\ i = \ q(X_t|Y_{t+1}) = N\left(X_t, \sigma_{X_{i-1}}\right) \quad \text{Eq. 71}$$

$$\{X_i\} \xrightarrow{fit\ normal\ distribution} N\left(\mu_{X_i}, \sigma_{X_i}\right) \quad \text{Eq. 72}$$

$$Markov\ Chain\ start\ value, updating\ step\ i = \ X_{o,i} = \mu_{X_{i-1}} \quad \text{Eq. 73}$$

3.4.3 Time-Discretization of Material Balance Equation

For the purpose of applying the material balance equation as a forward model in the likelihood function it is formulated as an incremental objective function (Eq. 74) and solved iteratively for pressure at each time step by using a Newton-Raphson algorithm. Given that the objective function is based on PVT tables, it is not a closed form equation and the pressure derivatives are obtained by means of central difference approximation. The procedure is initialized by calculating the initial pore volume in the tank (Eq. 75). The program then re-calculates the amount of rock compaction as well as oil, gas and water volume in the pore space at each time step (Eq. 76 - Eq. 79).

$$f = N(t+\Delta t) + G(t+\Delta t) + W_{res}(t+\Delta t) + \Delta PV(t+\Delta t) - PV_i = 0 \quad \text{Eq. 74}$$

$$PV_i = PV(t=0) = N \cdot B_{oi}/(1 - S_{wc}) \quad \text{Eq. 75}$$

$$N(t+\Delta t) = N(t) \cdot B_o(t+\Delta t) - N_p(t+\Delta t) \cdot B_o(t+\Delta t) \quad \text{Eq. 76}$$

$$G(t+\Delta t) = G(t) \cdot B_g(t+\Delta t) - G_p \cdot B_g(t+\Delta t) + N(t) \cdot [R_s(t) - R_s(t+\Delta t)] \cdot B_g(t+\Delta t) - \left[N(t) - N_p(t)\right] \cdot R_s(t+\Delta t) \cdot B_g(t+\Delta t) \quad \text{Eq. 77}$$

$$W_{res}(t+\Delta t) = W_{res}(t+\Delta t) \cdot B_w(t+\Delta t) - W_p(t+\Delta t) \cdot B_w(t+\Delta t) + W_e(t+\Delta t) \quad \text{Eq. 78}$$

$$\Delta PV(t+\Delta t) = PV_i \cdot c_f \cdot \left(p_{r,i} - p_r(t+\Delta t)\right) \quad \text{Eq. 79}$$

The aquifer influx $W_e(t+\Delta t)$ cannot be measured and must therefore be estimated with an aquifer model. Time steps are therefore limited to 10 days in this implementation of the

Fetkovich aquifer model is chosen based on sensitivity analysis. A summary of the time-discretized aquifer response equations is provided in Eq. 80 - Eq. 82.

$$p_a(t+\Delta t) = p_r(t) + \left(p_a(t) - p_r(t)\right) \cdot exp(-[J_w \cdot \Delta t]/[c_t \cdot W_a(t)]) \quad \text{Eq. 80}$$

$$W_e(t+\Delta t) = c_t \cdot W_a(t) \cdot \left(p_a(t) - p_r(t)\right) \cdot \left(1 - exp(-[J_w \cdot \Delta t]/[c_t \cdot W_a(t)])\right) \quad \text{Eq. 81}$$

$$W_a(t+\Delta t) = W_a(t) - W_e(t+\Delta t) \quad \text{Eq. 82}$$

3.5 Synthetic Data Set

In order to test the Bayesian updating model, a synthetic data set is required. Since real field production data is not available for this study, a synthetic data set is generated using the Eclipse reservoir simulation software. This comes with the additional benefit that it allows for direct testing the accuracy of the Bayesian updating technique. The resulting Eclipse input file can be found in Appendix G. The simulation model properties are based on correlations and typical reservoir properties found in published works on commercial oil fields. A Python script is used to populate the simulation grid with random properties in the standard Eclipse input file format.

3.5.1 Model Geometry and Grid Properties

A rectangular reservoir structure representative of a reservoir fault block with a small dip is created and a corner point grid structure is applied. The overall simulation model geometry, including distance to the oil-water contact is provided in (Figure 30). Grid block properties such as porosity and absolute permeability are generated by random sampling to fit a linear log(k) vs. porosity relationship; a trend which is commonly observed in commerical oil fields (Figure 31). Details on the process associated with generating random grid properties are provided in Appendix D. The total fluids in place and initial reservoir pressure is provided in Table 9.

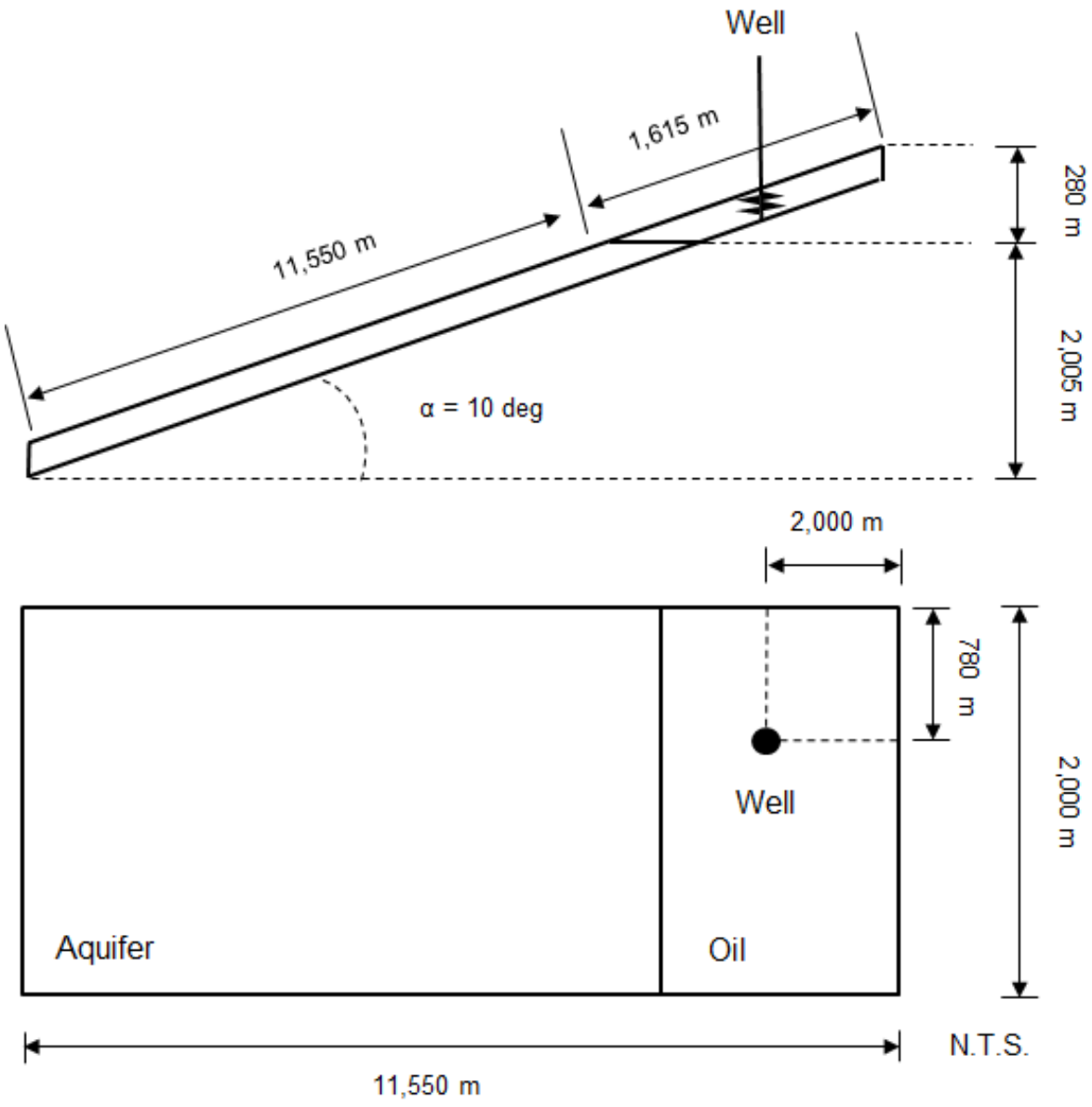


Figure 30 - Eclipse model overview 5x exaggerated in the vertical direction

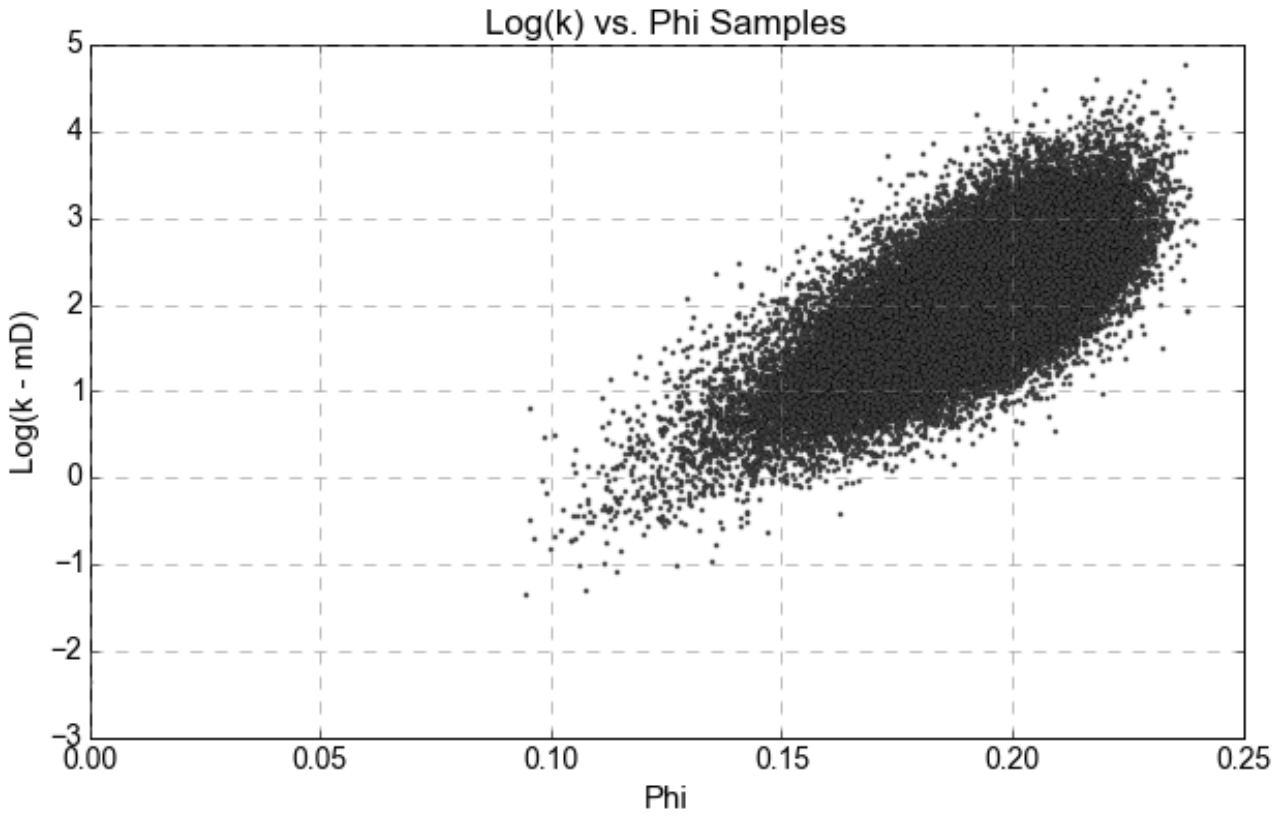


Figure 31 - Log(Permeability) vs. Porosity Plot

Table 9 - Summary of Simulation Model Parameters

Item	**Value**	**Unit**
Original Oil in Place	31.6	M Sm^3
Water in place in Aquifer	444.6	M Rm^3
Number of cells in i-direction	50	-
Number of cells in j-direction	90	-
Number of cells in k-direction	10	-
Total number of cells	45,000	-
Initial Reservoir Pressure	296.4	bar(a)
Initial Solution GOR	120	Sm^3/Sm^3
Rock compressibility	5.0E-5	1/bar(a)

3.5.2 PVT Data

Fluid properties are generated by using Black Oil PVT correlations suitable for the chosen pressure/temperature ranges and overall fluid properties (Table 10). The resulting PVT data set is applied to both simulation model and material balance model (Table 11). The PVT correlations used to generate the PVT table are provided in Appendix C.

Table 10 - PVT/Reservoir Properties

Property	**Metric Value**	**Oil Field Value**
Reservoir Temperature (T)	80 deg C	194 deg F
Formation Water Salinity (w_s)	100,000 ppm	100,000 ppm
Gas Specific Gravity (γ_g)	0.7	0.7
Oil API Gravity (γ_{API})	35	35
Formation GOR (R_s)	120 Sm/Sm^3	673.7 SCF/STB

Table 11 - PVT Data Table

P *bar(a)*	Rs *Sm^3/Sm^3*	co *1/bar*	Bo *Rm^3/Sm^3*	mu_o *mPa-s*	Z *-*	Bg *Rm^3/Sm^3*	mu_g *mPa-s*	cw *1/bar*	Bw *Rm^3/Sm^3*	mu_w *mPa-s*
500.0	120.000	9.157E-05	1.301	0.834	1.170	0.00291	0.054	2.557E-05	1.011	0.650
475.0	120.000	9.639E-05	1.303	0.804	1.137	0.00297	0.052	2.538E-05	1.013	0.637
450.0	120.000	1.017E-04	1.304	0.774	1.104	0.00305	0.051	2.518E-05	1.015	0.623
425.0	120.000	1.077E-04	1.306	0.745	1.071	0.00313	0.049	2.498E-05	1.016	0.610
400.0	120.000	1.145E-04	1.308	0.716	1.039	0.00323	0.047	2.478E-05	1.017	0.598
375.0	120.000	1.221E-04	1.310	0.689	1.007	0.00334	0.046	2.459E-05	1.019	0.586
350.0	120.000	1.308E-04	1.313	0.663	0.977	0.00347	0.044	2.439E-05	1.020	0.574
325.0	120.000	1.409E-04	1.316	0.638	0.948	0.00362	0.042	2.419E-05	1.021	0.562
300.0	120.000	1.526E-04	1.320	0.614	0.920	0.00381	0.040	2.399E-05	1.022	0.551
275.0	120.000	1.665E-04	1.324	0.592	0.896	0.00405	0.038	2.379E-05	1.023	0.541
250.0	120.000	1.831E-04	1.329	0.571	0.875	0.00435	0.036	2.360E-05	1.024	0.530
227.3*	120.000	2.014E-04	1.334	0.555	0.861	0.00470	0.034	2.342E-05	1.025	0.521
200.0	102.254	-	1.284	0.608	0.849	0.00527	0.031	2.320E-05	1.026	0.511
175.0	87.021	-	1.242	0.666	0.846	0.00600	0.029	2.300E-05	1.027	0.501
150.0	72.705	-	1.203	0.735	0.849	0.00703	0.027	2.281E-05	1.028	0.492
125.0	59.254	-	1.167	0.820	0.861	0.00855	0.025	2.261E-05	1.028	0.484
100.0	46.614	-	1.135	0.927	0.880	0.01093	0.024	2.241E-05	1.029	0.476
75.0	34.722	-	1.107	1.068	0.905	0.01499	0.022	2.221E-05	1.029	0.468
50.0	23.482	-	1.081	1.263	0.936	0.02325	0.021	2.202E-05	1.030	0.460
35.0	16.975	-	1.067	1.425	0.955	0.03390	0.021	2.190E-05	1.030	0.456
25.0	12.679	-	1.058	1.564	0.968	0.04811	0.021	2.182E-05	1.030	0.453
10.0	6.066	-	1.045	1.858	0.988	0.12268	0.020	2.170E-05	1.030	0.449
5.0	3.633	-	1.041	2.004	0.994	0.24692	0.020	2.166E-05	1.030	0.448
1.0	1.235	-	1.037	2.178	0.999	1.24075	0.020	2.163E-05	1.031	0.447

**Bubble point pressure, p_b*

3.5.3 Capillary Pressure and Relative Permeability Model

Generic logarithmic expressions are used to generate drainage and imbibition capillary pressure curves. The drainage curve models the initial fluid distributions, while the imbibition curve models capillary pressure behavior associated with water encroaching into the oil zone from the aquifer. Because the surface tension between oil and gas is orders of magnitudes less than that of water and oil, oil/gas capillary pressure is assumed to be zero. The resulting capillary pressure curves are shown in Figure 32. The drainage curve corresponds to a transition zone about 60 m thick. The capillary pressure drainage parameters are summarized in Table 14. Appendix E contains details about the capillary pressure model used.

Two-phase relative permeability curves are modeled using Corey functions (Corey 1954), which are power-law correlations for gas and oil relative permeability. Details about Corey functions can be found in Appendix F. Reasonable Corey exponents are chosen for Oil/Water (Table 12) and Oil/Gas (Table 13) based on published data on simulation of North Sea reservoirs (Tangen, 2012). The Eclipse default model is used to model three-phase relative permeability. The resulting Oil/Water and Oil/Gas relative permeability curves are provided in Figure 33 and Figure 34. Note that the oil-water relative permeability curve in Figure 33 is extrapolated to a value of 1.0, which is necessary because the water saturation is equal to 1.0 at the free water level and below. The default three-phase relative permeability model in Eclipse was implemented (Eclipse manual, 2010). It assumes complete segregation of gas and water within each cell, which is often a reasonable assumption because in most reservoir simulation studies the number of grid blocks where three phase flows occurs is relatively small.

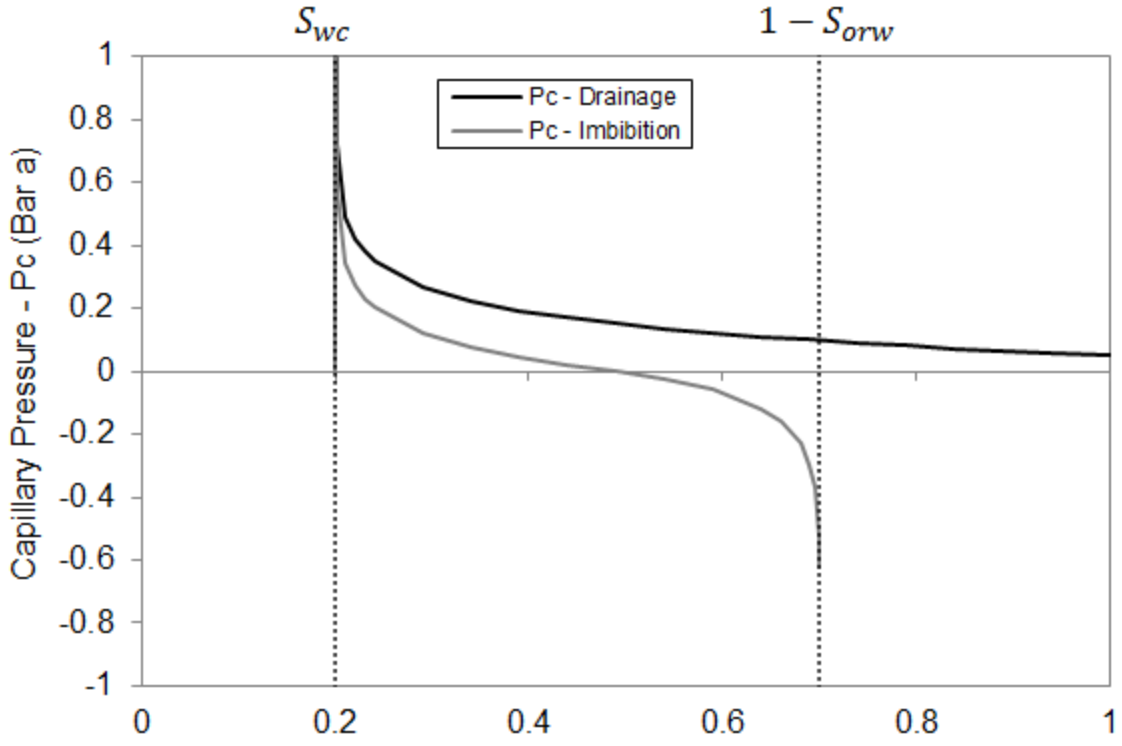


Figure 32 - Drainage and Imbibition Capillary Pressure Curves

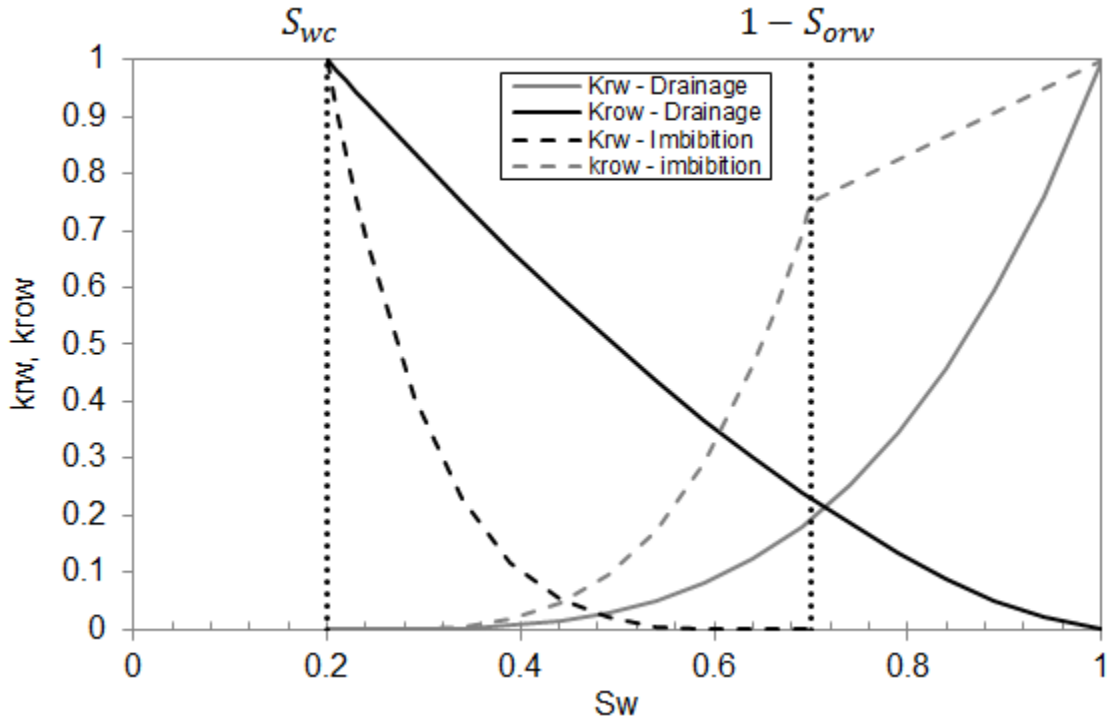


Figure 33 - Oil/Water Relative Permeability Curves

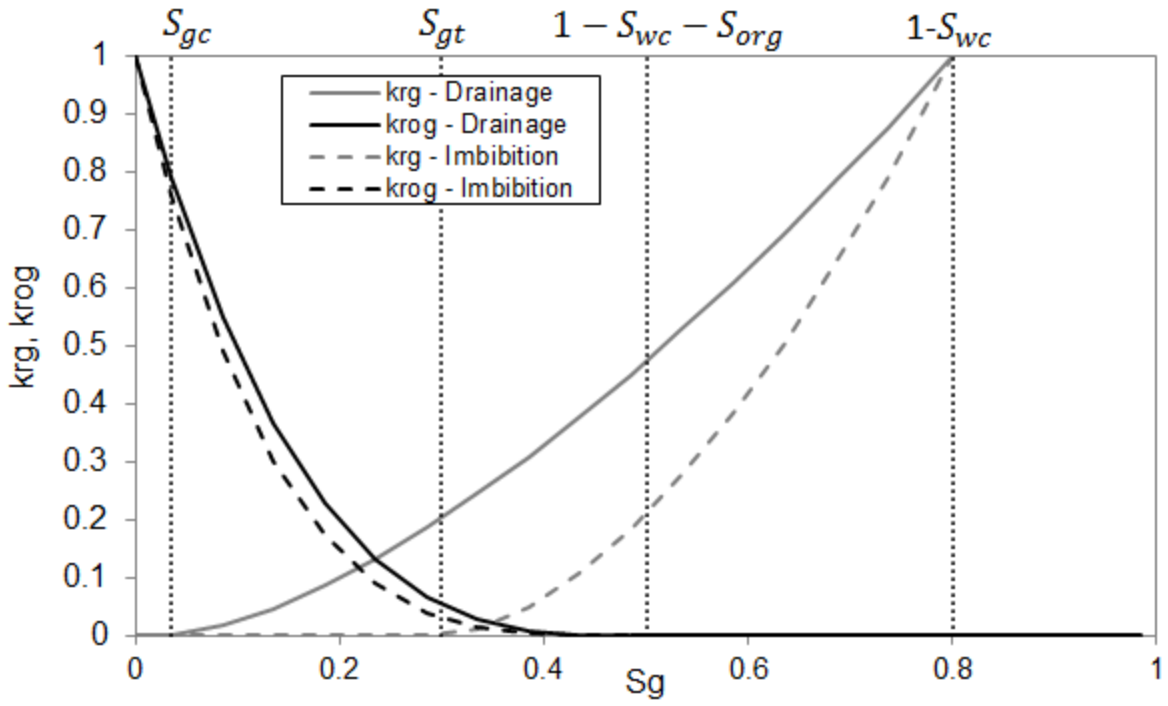


Figure 34 - Oil/Gas Relative Permeability Curves

Table 12 - Oil/Water Corey Parameters

Process	**no**	**nw**	**Krow,max**	**Krw, max**
Oil/Water Drainage	1.5	3.5	1.0	1.0
Oil/Water Imbibition	4.5	3.8	1.0	0.75

Table 13 - Oil/Gas Corey Parameters

Process	**no**	**ng**	**Krog,max**	**Krg, max**
Oil/Gas Drainage	3.2	1.5	1.0	1.0
Oil/Gas Imbibition	3.8	1.7	1.0	1.0

Table 14 - Logarithmic Capillary Pressure Parameters

Process	α_1	α_2	S_{wx}	P_{th} **(Bara)**
Drainage	-1.0	0.5	1.0	0.05
Imbibition	-0.1	0.1	1.0	-

3.5.4 Synthetic Reservoir Parameters and Production Data

The intial oil in place N, gas in place G and aquifer size W_i are obtained from the initialization report from the reservoir simulation model. The aquifer index cannot be extracted directly from the simulation model because the grid properties are populated based on random sampling from probability distributions. The true value of the aquifer index J_w is therefore calculated based on the average horizontal permeability below the oil-water contact. The geometry and viscosity used for this J_w calculation is taken from Figure 30 and Table 11 in Section 3.5.1, respectively. A summary of the parameter values obtained from the simulation model are provided in Table 15.

$$J_w = \frac{K \cdot H \cdot W}{\mu \cdot L} = \frac{370\ mD \cdot 100\ m\ \cdot 2000m}{0.55cp \cdot 11550m} = 100.68 \sim 100\ \text{R m}^3/\text{bar(a)} \cdot \text{day}$$

Table 15 - Summary of Synthetic Model Parameters

Parameter	Value	Unit
N	31.6	M Sm^3
Wi	444.6	M Sm^3
Jw	100	Rm^3/bar(a)*day
G	0	N Sm^3

In terms of production data, the simulation model runs for 4,000 days with a maximum liquid rate constraint of 1,750 Sm^3/day. A reasonable minimum bottom hole pressure of 90 bar(a) is specified. The resulting production rates can be viewed in Figure 35, while the average pressure decline trend for the oil column is shown in Figure 36. Note that the average reservoir pressure is extracted from the simulation model by averaging the pressure in the oil column at each time step. In reality, it is necessary to apply pressure transient analysis techniques to determine the average pressure in the reservoir. After about 2,000 days, oil production starts dropping below 1,750 Sm^3 as water from the aquifer reaches the producer. It is also worth noting that the gas rate increases as the reservoir pressure drops below the bubble point pressure and a secondary gas gap is formed. The formation of the secondary gas cap and encroachment of water into the oil zone can also be observed in the ternary plots provided in Figure 37. At about 3,000 days, the oil rate drops rapidly as the bottom hole pressure reaches the minimum limit of 90 bar(a). At this point, the rates drop because the bottom hole pressure cannot be lowered further to facilitate the drawdown required to maintain a total liquid rate of 1,750 Sm^3/day.

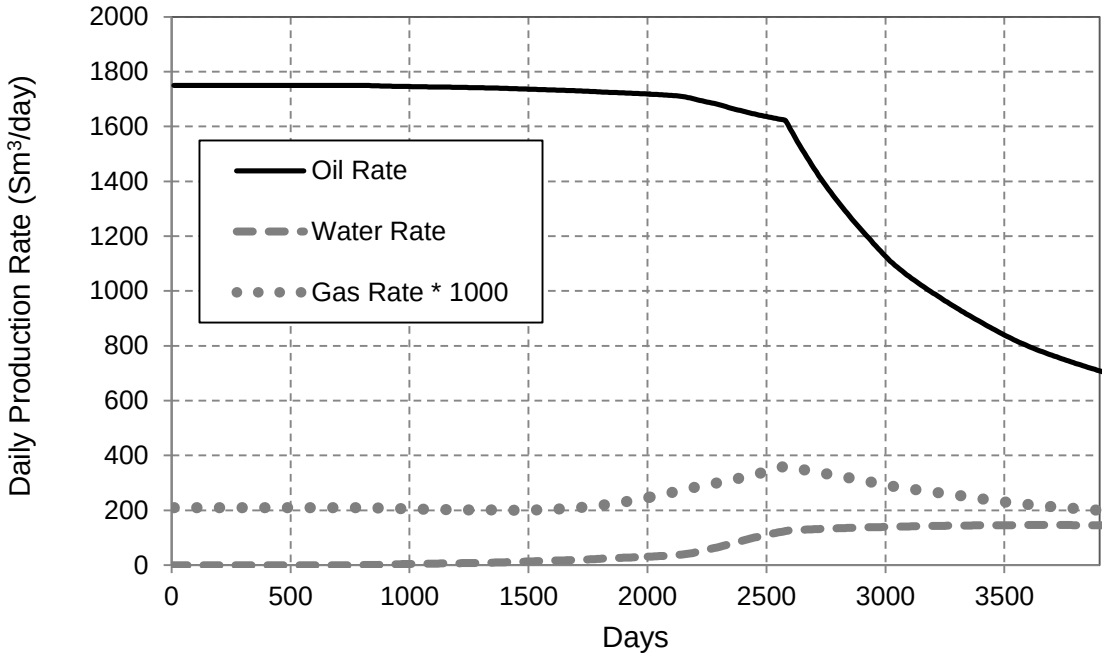


Figure 35 - Production Rates from Synthetic Model

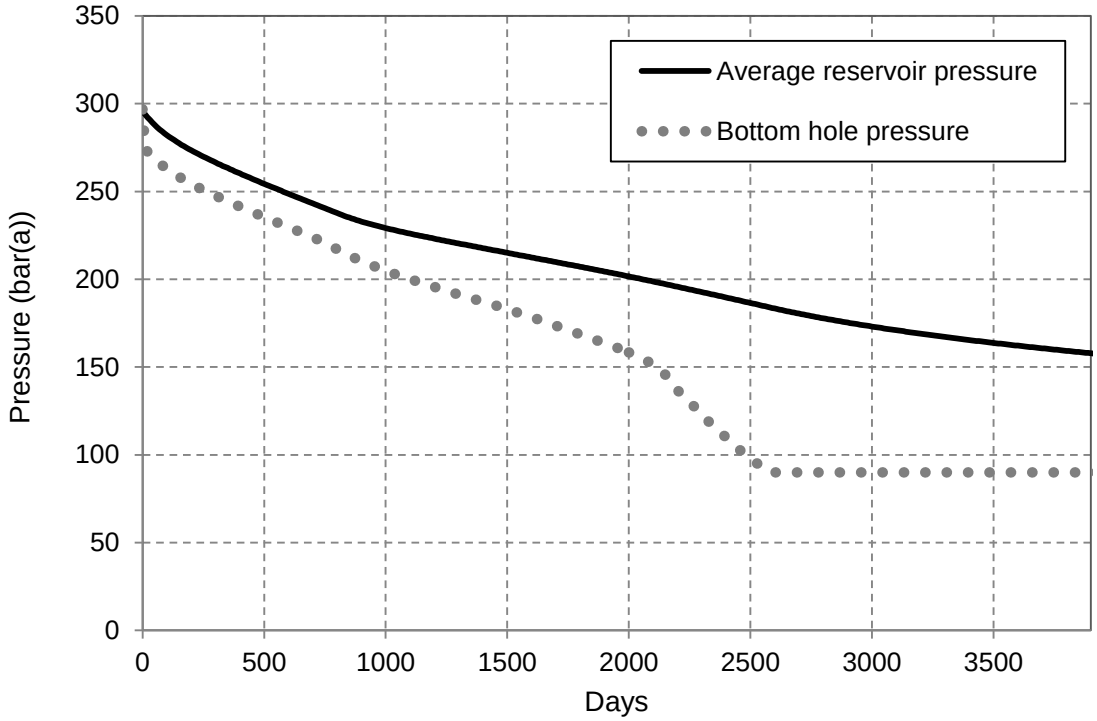


Figure 36 - Reservoir Pressure and BHP from Synthetic Model

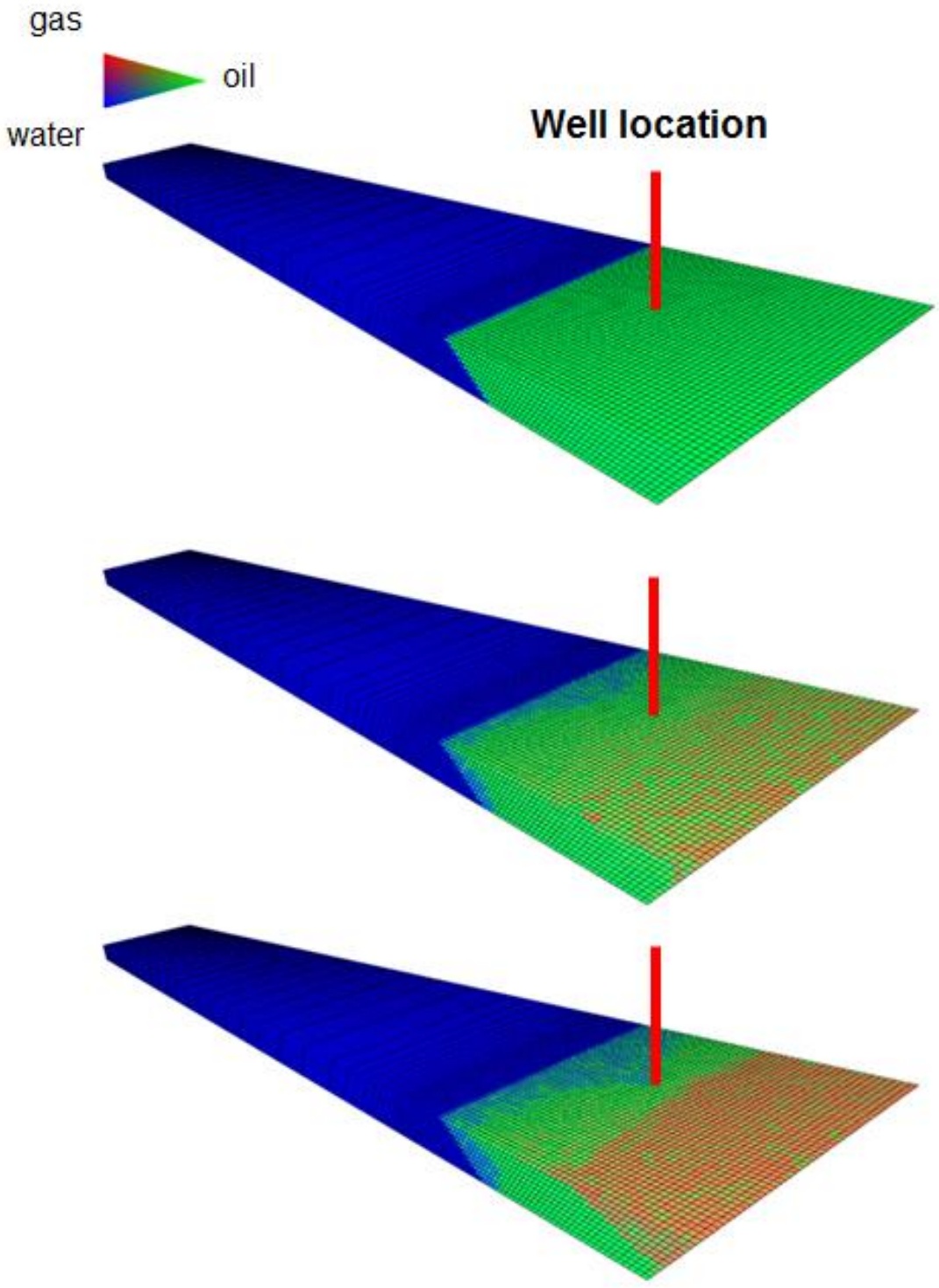


Figure 37 - Ternary Plot of Synthetic Model

3.5.5 Material Balance Model Response to Synthetic Production Data

This section demonstrates how the material balance model responds to the synthetic production data from the reservoir simulator (Figure 35). The material balance is run forward with the true reservoir parameters, which are listed in Table 16. From Figure 38 it is clear that the material balance model matches the simulation model pressure response very well and demonstrates that the synthetic reservoir behaves like a perfect tank. This is despite the random grid properties and overall complexity associated with the simulation model. The observed accuracy is attributed to the fact that true parameter values from the simulation model are used. Furthermore, we observe that there is a kink in the pressure predicted by the material balance model at 900 days. This occurs because the bubble point is reached at this point, which causes solution gas drive to take effect. The abrupt kink is not present in the pressure trend from the simulation model, as the bubble point is not reached in all simulation cells simultaneously. The oneset of solution gas drive causes the aquifer influx rate to drop quite significantly due to the back-pressure provided by liberated gas (Figure 39). Finally, after 2,800 days the aquifer influx drops steadily due to overall lower extraction rates from the reservoir. For reference, Figure 40 shows material balance fluid saturations vs. time.

Table 16 - Tank Model Parameters

Parameter	Value	Unit
N	31.6	M Sm^3
Wi	444.6	M Sm^3
Jw	100	Rm^3/bar(a)*day
G	0.0	M Sm^3
Swc	0.2	-
cw	2.8E-5	1/bar(a)
cf	5.0E-5	1/bar(a)
PVT	Same as simulation model	-

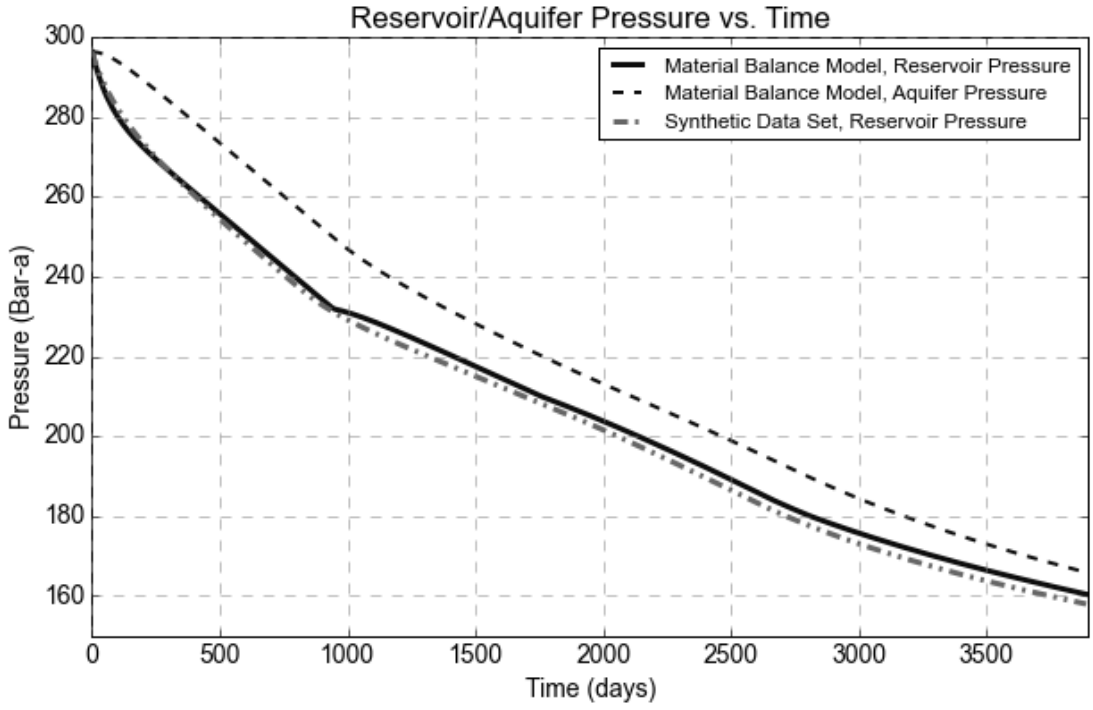


Figure 38 - Material Balance vs. Synthetic Data Reservoir Pressure

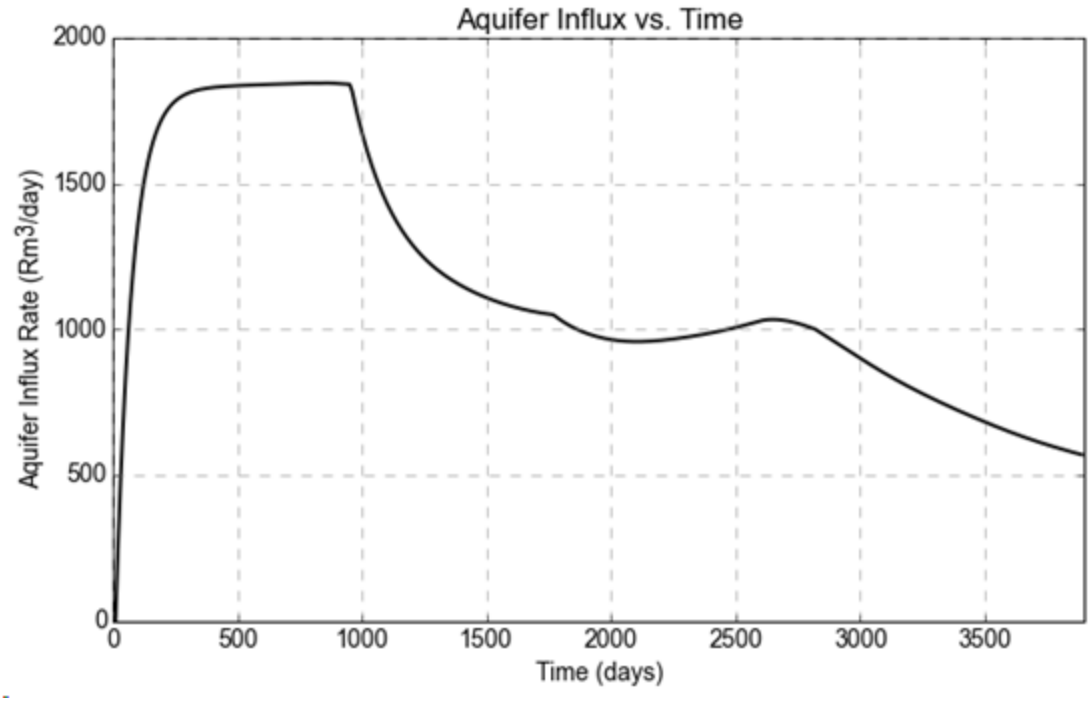


Figure 39 - Material Balance Aquifer Influx Prediction

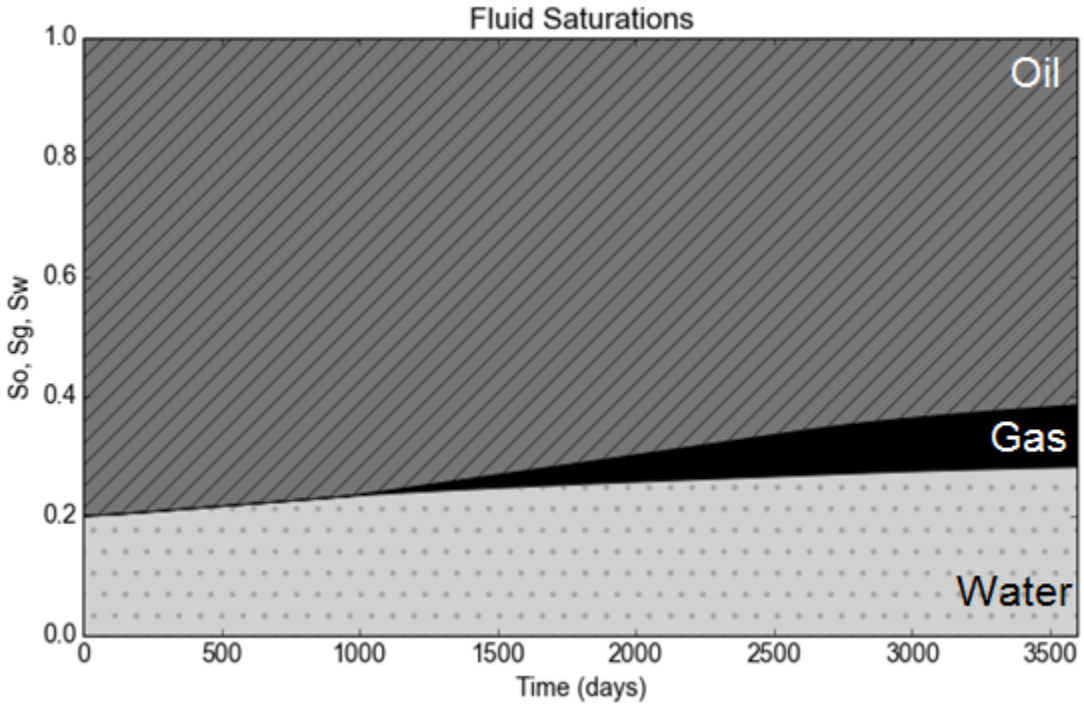


Figure 40 - Material Balance Fluid Saturation Predictions vs. Time

Next, an ad-hoc sensitivity analysis is provided to show the material balance responds to changes in input parameters in the vicinity of the true parameter values. Summaries of the sensitivity cases are provided in Table 17 and Figure 41. For the parameter ranges tested, it is clear that N and W_i overall have more effect on the predicted pressure than does J_w. It is also worth noting that the aquifer influx increases as N decreases. This occurs because smaller values of N are associated with more rapid reservoir pressure decline, which in turn creates a larger pressure differential between the oil tank and the aquifer tank, leading to larger aquifer influx rates.

Table 17 - Deterministic Sensitivities for Material Balance Model

Material Balance Sensitivity Case	**N**	**Wi**	**Jw**
1	25	444.6	100
	45	444.6	100
	65	444.6	100
2	31.6	200	100
	31.6	400	100
	31.6	600	100
3	31.6	444.6	50
	31.6	444.6	100
	31.6	444.6	150

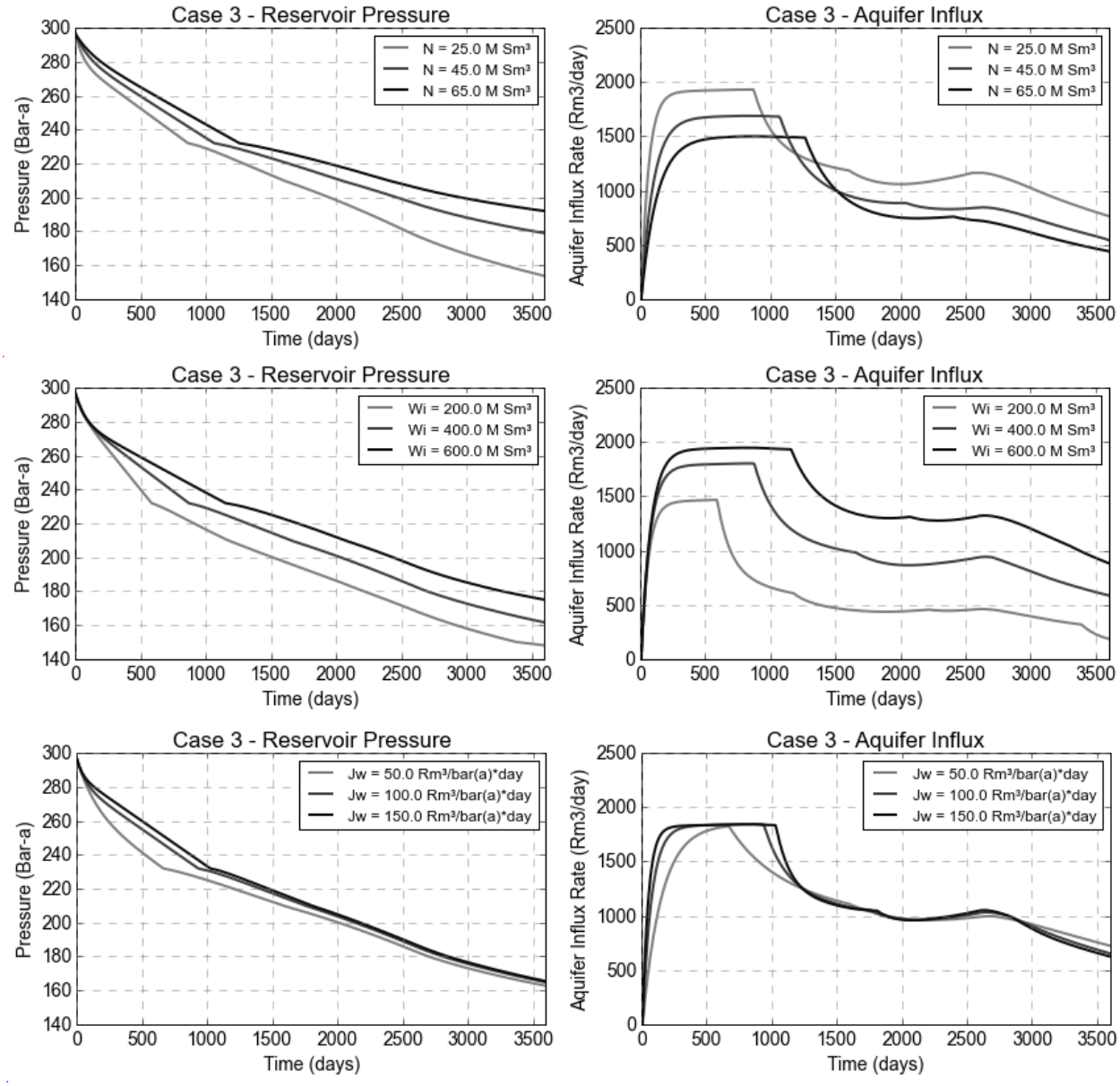


Figure 41 - Material Balance Model Sensitivities

3.6 Bayesian Updating Case Study

In this section Bayesian updating is applied to infer the material balance parameters by assimilating measured data resulting from the synthetic data set. Four different cases are evaluated and summarized in Table 18. The initial gas in place G is kept as a deterministic constant equal to zero for all cases. This is often a reasonable assumption for real reservoirs, provided that PVT data and pressure transient analysis provide sufficient evidence to rule out the existence of an initial gas cap.

Table 18 - Case Study Summary

Case	**Purpose**
1	Demonstrate structured grid solution procedure for two-variable Bayesian updating problems and examine 3D surface plots, 2D contour plots and marginal distributions for the prior, likelihood and posterior distributions.
2	Demonstrate MCMC based solution by comparing to the two-variable structured grid solution in Case 1.
3	Demonstrate the MCMC based solution strategy on a three-variable problem that cannot be visualized in 3D surface plots. Provide a comprehensive set of diagnostic plots that are useful for assessing model behavior and MCMC convergence properties.
4	Demonstrate the effect of both consistent and random measurement errors on the posterior parameter estimates

3.6.1.1 Case 1 - Two-Variable Structured Grid Solution

In the first case, initial oil in place N and aquifer index J_w are treated as random variables, while aquifer size W_i is treated as a deterministic constant. The numerical value for W_i is extracted directly from the simulation model. This allows studying the structured grid solution with full visualization of the prior, likelihood and posterior distributions in three-dimensional surface plots. A summary of the input parameters associated with Case 1 is provided in (Table 19). A 100x100 grid is used to generate the prior, likelihood and posterior distributions. This involves running the material balance model at 10,000 grid locations for all 12 time steps, running to 3,600 days. The resulting three-dimensional surface plots are shown in Figure 42, Figure 43 and Figure 44. The effect of the likelihood variance is assessed by setting the variance to 10 and 50 bar(a)^2 and the results are compared on contour plots shown in Figure 45. This figure shows that the posterior assimilates the likelihood distribution faster for smaller likelihood variance values. The same behavior can be observed in the marginal posterior histograms, shown in (Figure 46). Furthermore, the peak of the likelihood distribution, also known as the Maximum Likelihood (MLE), corresponds to the parameter values associated with the best overall fit to the measured reservoir pressure decline trend. As such, the posterior is shown to be a compromise between prior belief and an optimal parameter fit. As more data is incorporated into the analysis, the posterior becomes increasingly similar to the likelihood distribution.

Table 19 - Case 1 Parameters

Item	**Value**	**Unit**
Grid size	100 x 100	-
Bayesian updating steps	12	-
Time steps	300*12	days
Likelihood variance	10 and 50	$bar(a)^2$
Aquifer Size (constant)	444.6	M Sm^3
Gas in place (constant)	0	M Sm^3
Prior - Oil in Place (N) [mean, standard deviation]	[50, 100]	M Sm^3
Prior - Aquifer Index (Jw) Prior [mean, standard deviation]	[150, 1000]	Rm^3/bar(a)*day

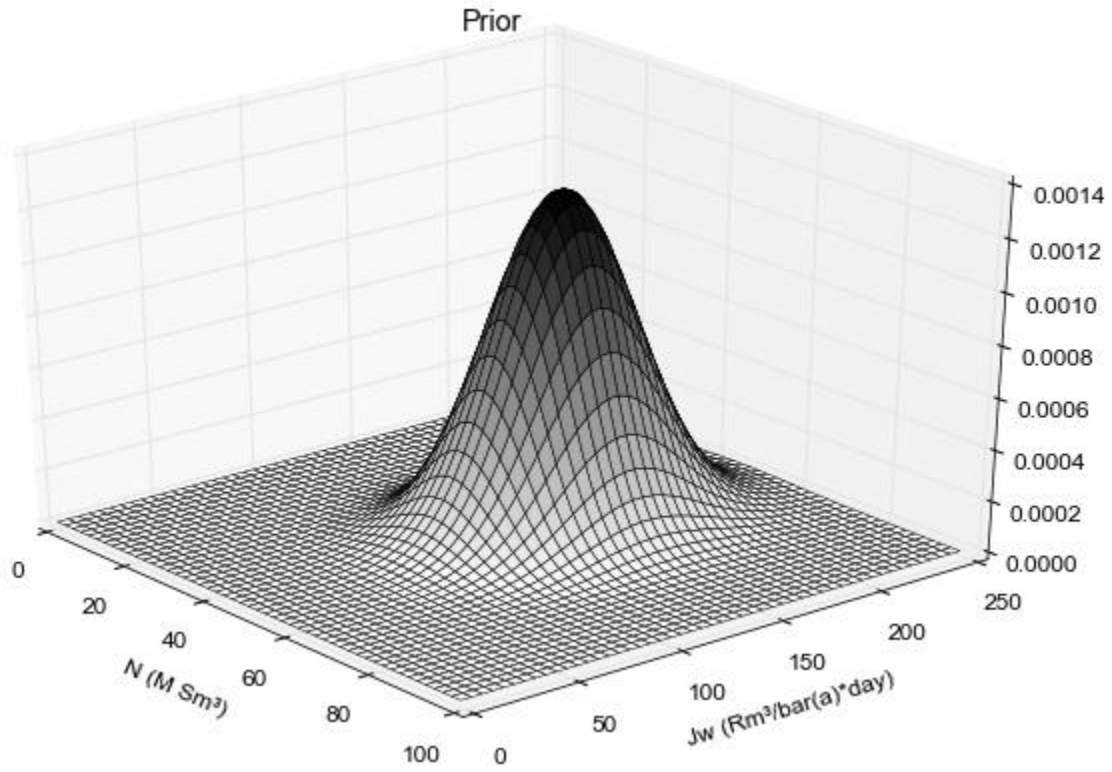


Figure 42 - Case 1, Prior Distribution

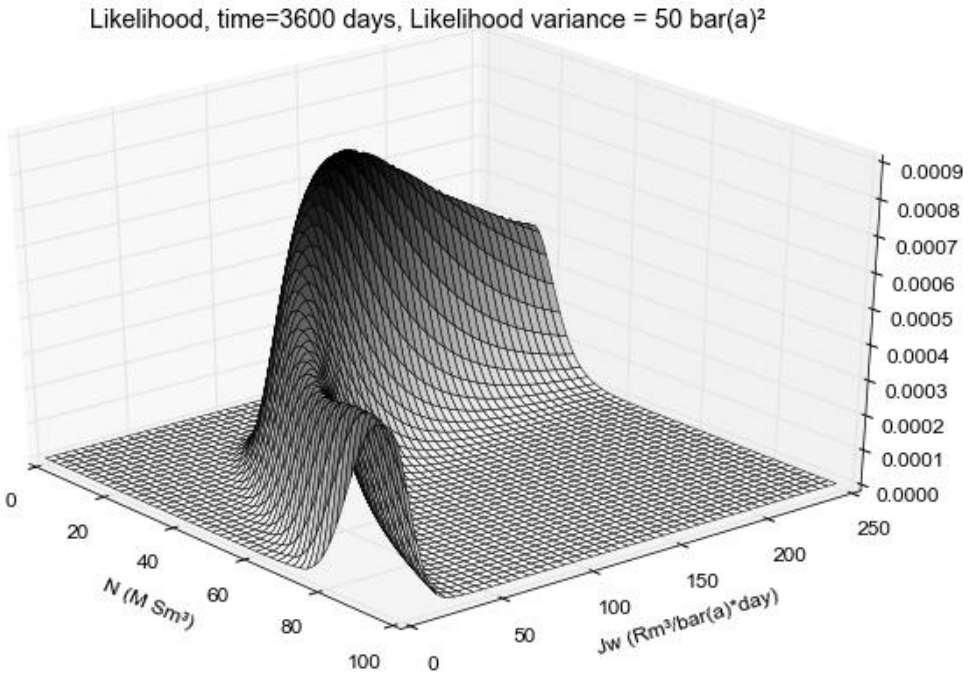


Figure 43 - Case 1, Likelihood Distribution

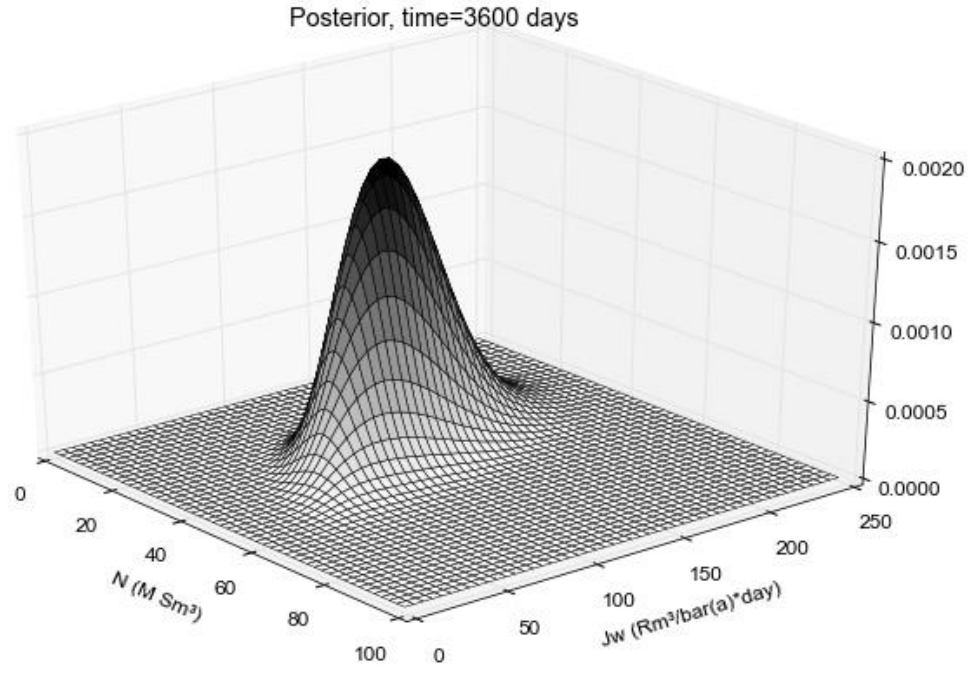


Figure 44 - Case 1, Posterior Distribution

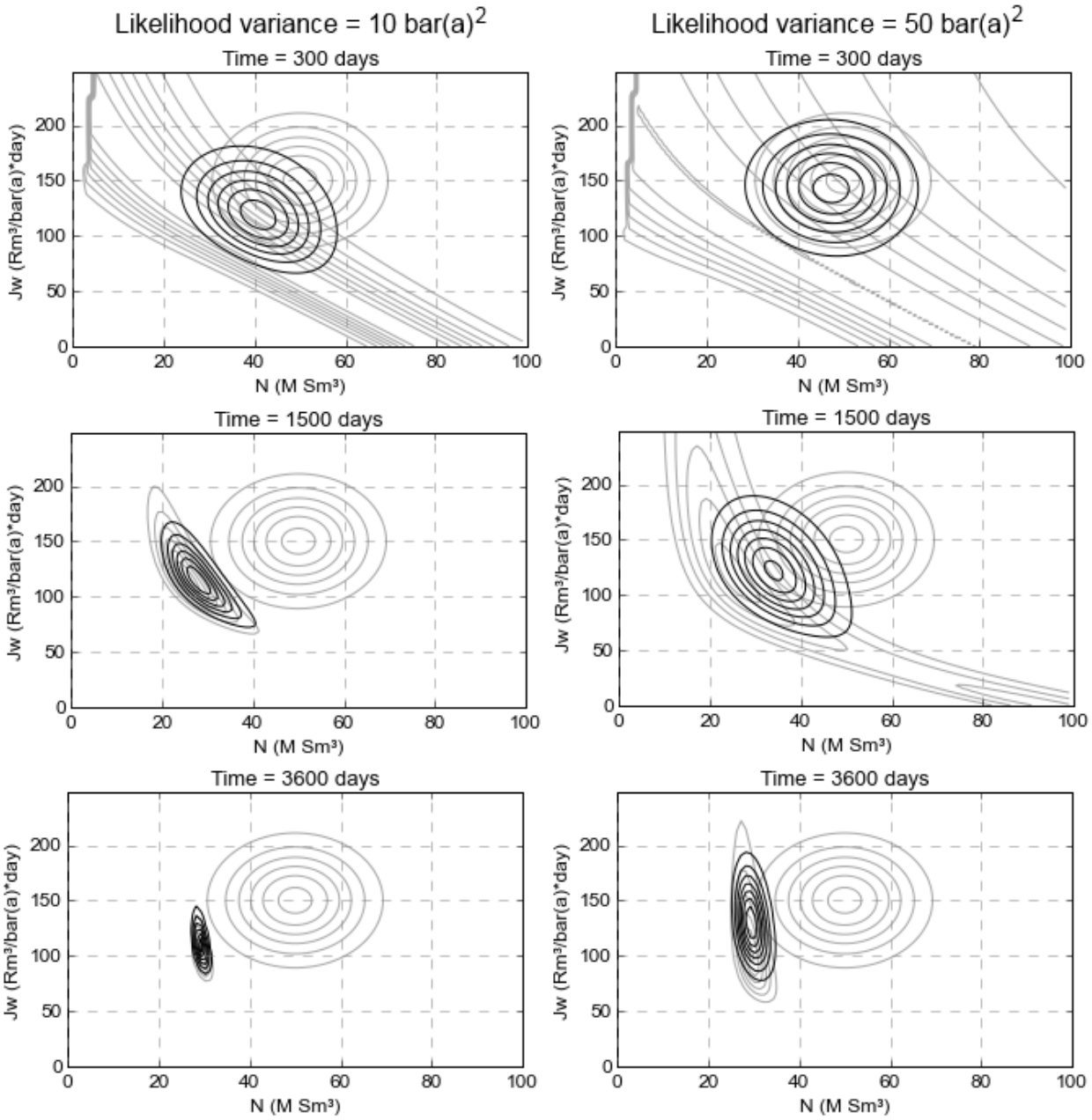


Figure 45 - Case 1, Effect of Error on Likeihood and Posterior

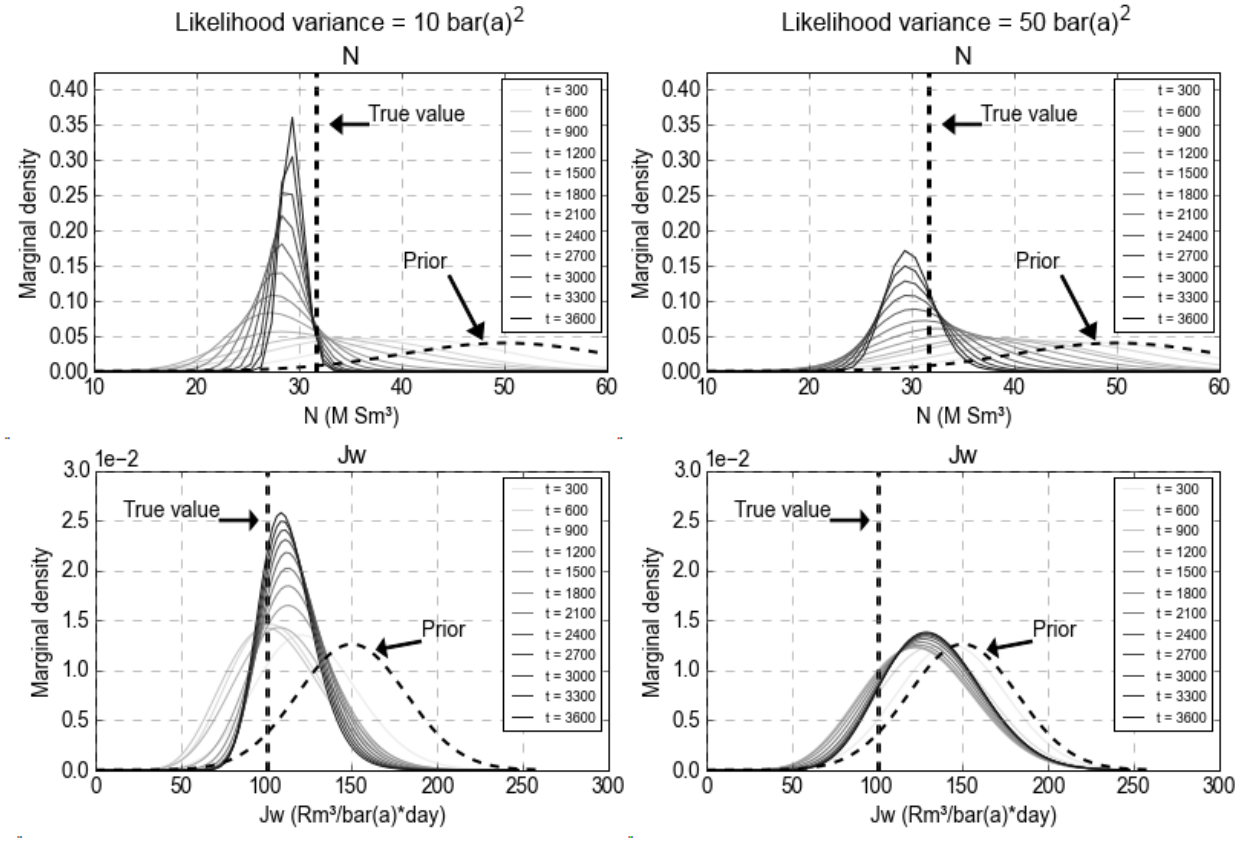


Figure 46 - Case 1, Effect of Error on Marginal Posteriors

3.6.1.2 Case 2 - Two-Variable MCMC Sampling Based Solution

The purpose of Case 2 is to provide a comparison between the MCMC sampling based algorithm and the grid based solution. Case 2 parameters are summarized in Table 20. Figure 47 shows a comparison between the marginal distributions and histograms resulting from the grid based solution and the MCMC algorithm at time = 1500 days. Figure 48 shows a scatter plot of the MCMC samples at times 300, 1500 and 3600 days plotted on top of prior/likelihood/posterior contours resulting from the grid based solution. Both Figure 47 and Figure 48 display good correspondence between the grid solution and the MCMC solution, thus providing confidence in the convergence properties of the MCMC algorithm. Figure 49 displays posterior marginal histogram outlines vs. time and shows how posterior variance decreases as the number of Bayesian assimilation steps increases. The same behavior can be observed in Figure 50, which shows posterior fitted normal distributions through time. Figure 51 displays acceptance ratios vs. sample number for each Bayesian assimilation step. It is evident that reasonable acceptance ratios ranging between 0.3 and 0.5 are achieved. Figure 52 displays running means for all time steps and show that the Markov Chain converges quickly and that the burn-in is achieved after about ~1,000 samples, thus showing the increased efficiency associated with the MCMC algorithm vs. the grid based approach. Figure 53 displays time series plots at times 300, 1500 and 3600 days and shows that the posterior region is being adequately sampled. Finally, Figure 54 shows autocorrelation vs. sample number at times 300, 1500 and 3600 days. Autocorrelation hovers around a value of zero, which indicates that the Markov Chains have good mixing properties.

Table 20 - Case 2 Parameters

Item	**Value**	**Unit**
Number of MCMC samples	10,000	-
Bayesian updating steps	12	-
Time steps	300 x 12	days
Likelihood variance	10	$bar(a)^2$
Aquifer Size (deterministic constant)	444.6	M Sm^3
Initial gas in place (deterministic constant)	0	M Sm^3
Oil in Place (N) Prior [mean, standard deviation]	[50, 10]	M Sm^3
Aquifer Index (Jw) Prior [mean, standard deviation]	[150, 31.68]	Rm^3/bar(a)*day

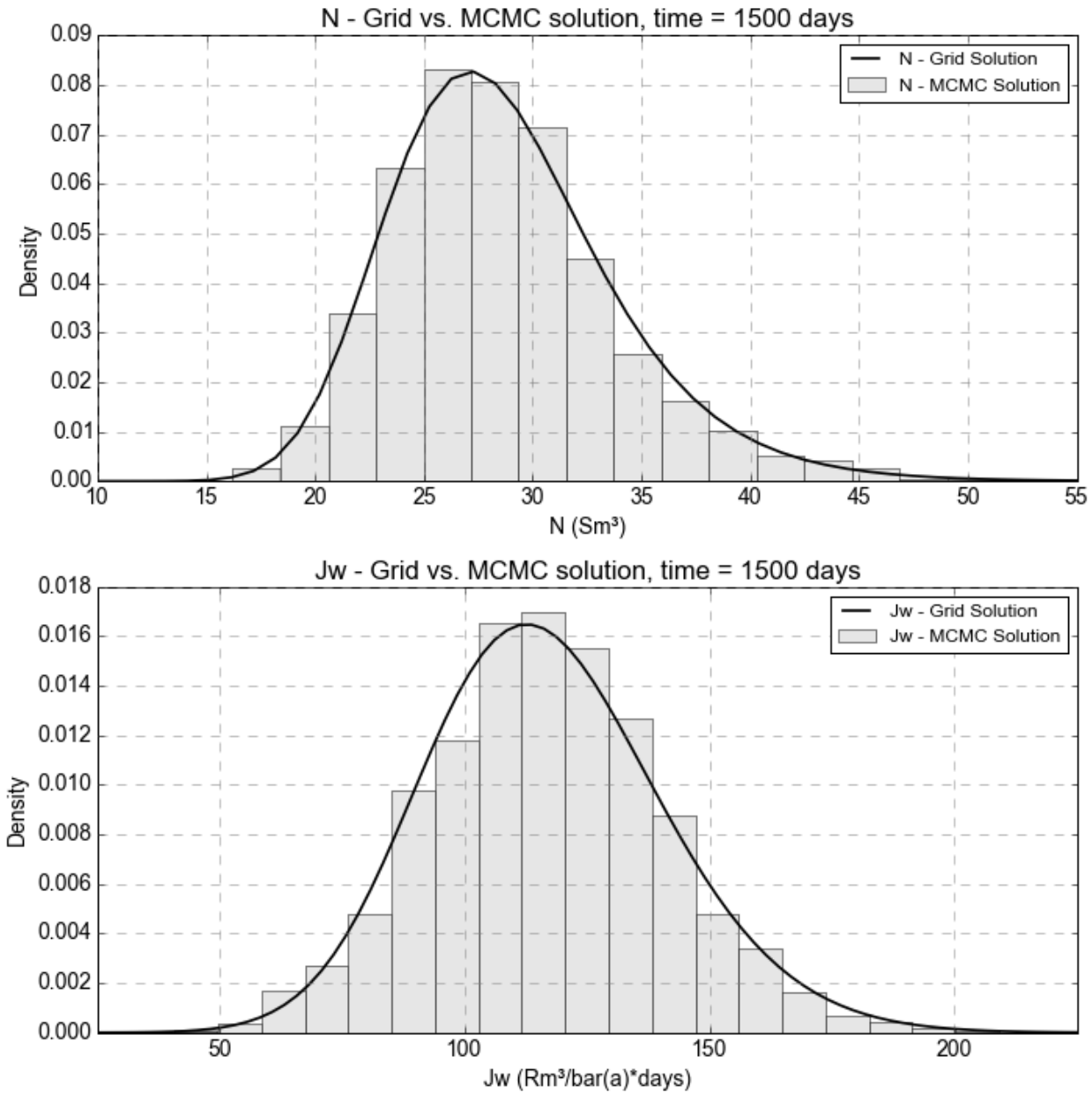


Figure 47 – Case 2, Grid vs. MCMC marginal posterior distributions at time = 1500 days

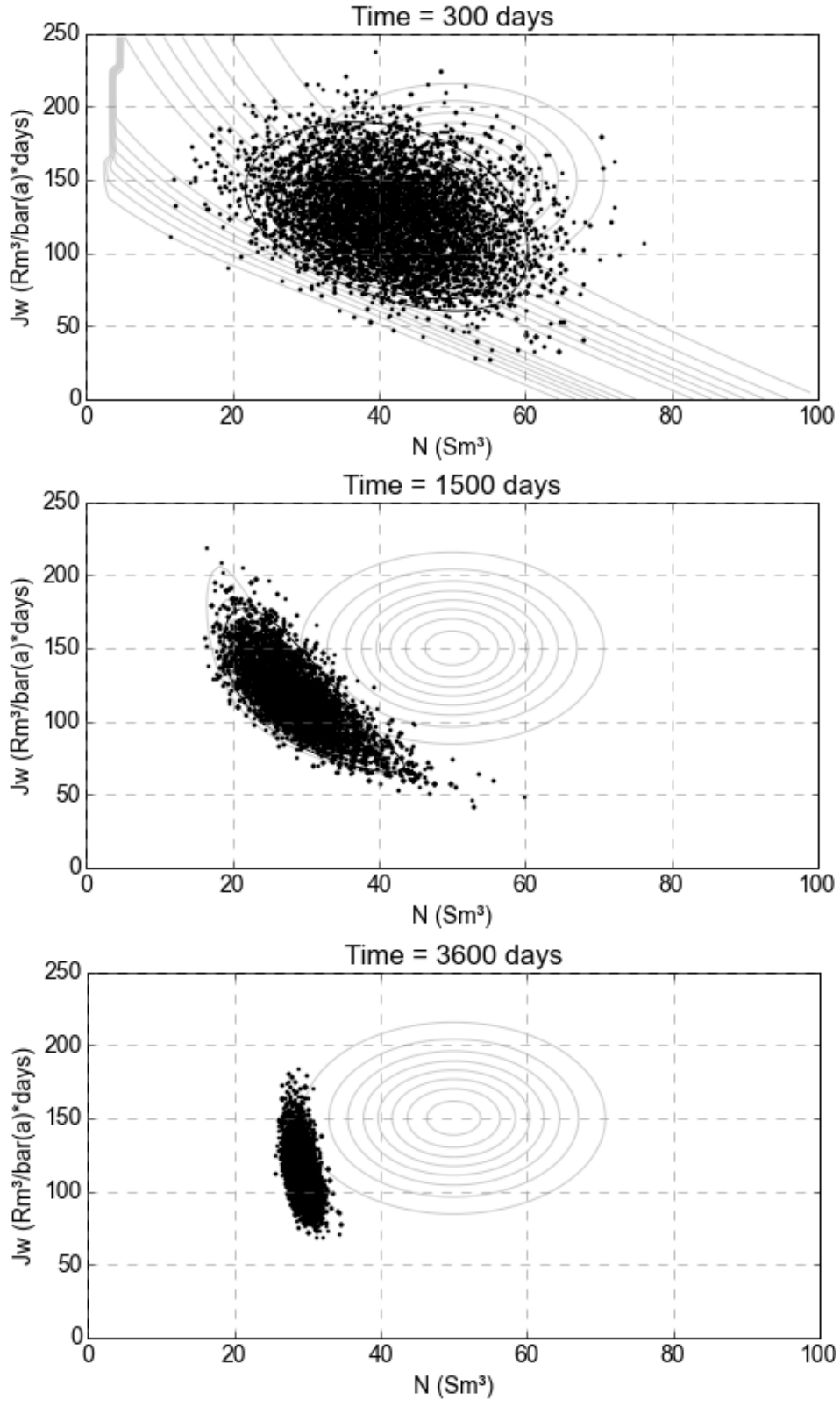


Figure 48 – Case 2, Posterior scatter plots

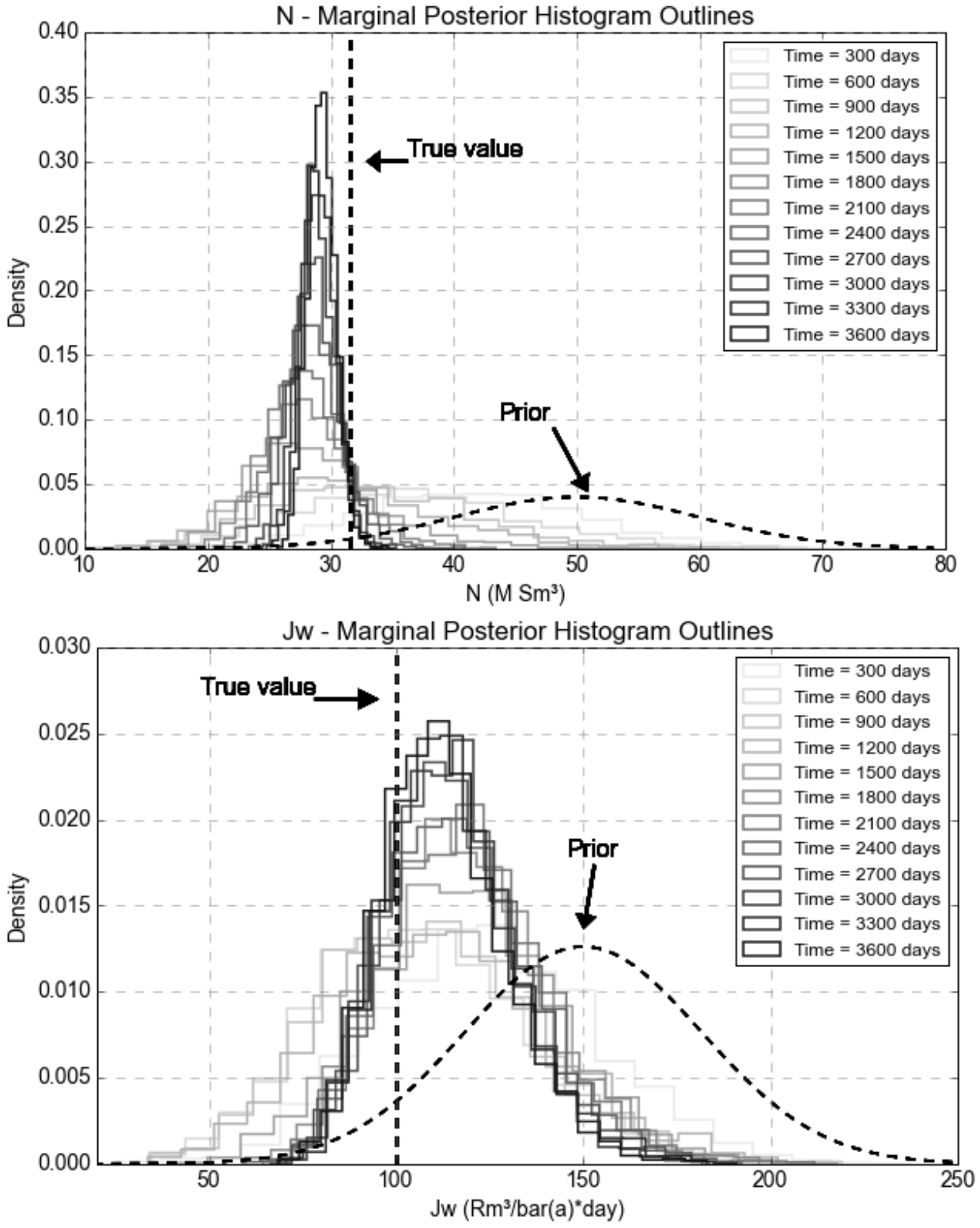


Figure 49 – Case 2, Posterior Marginal Histogram Outlines

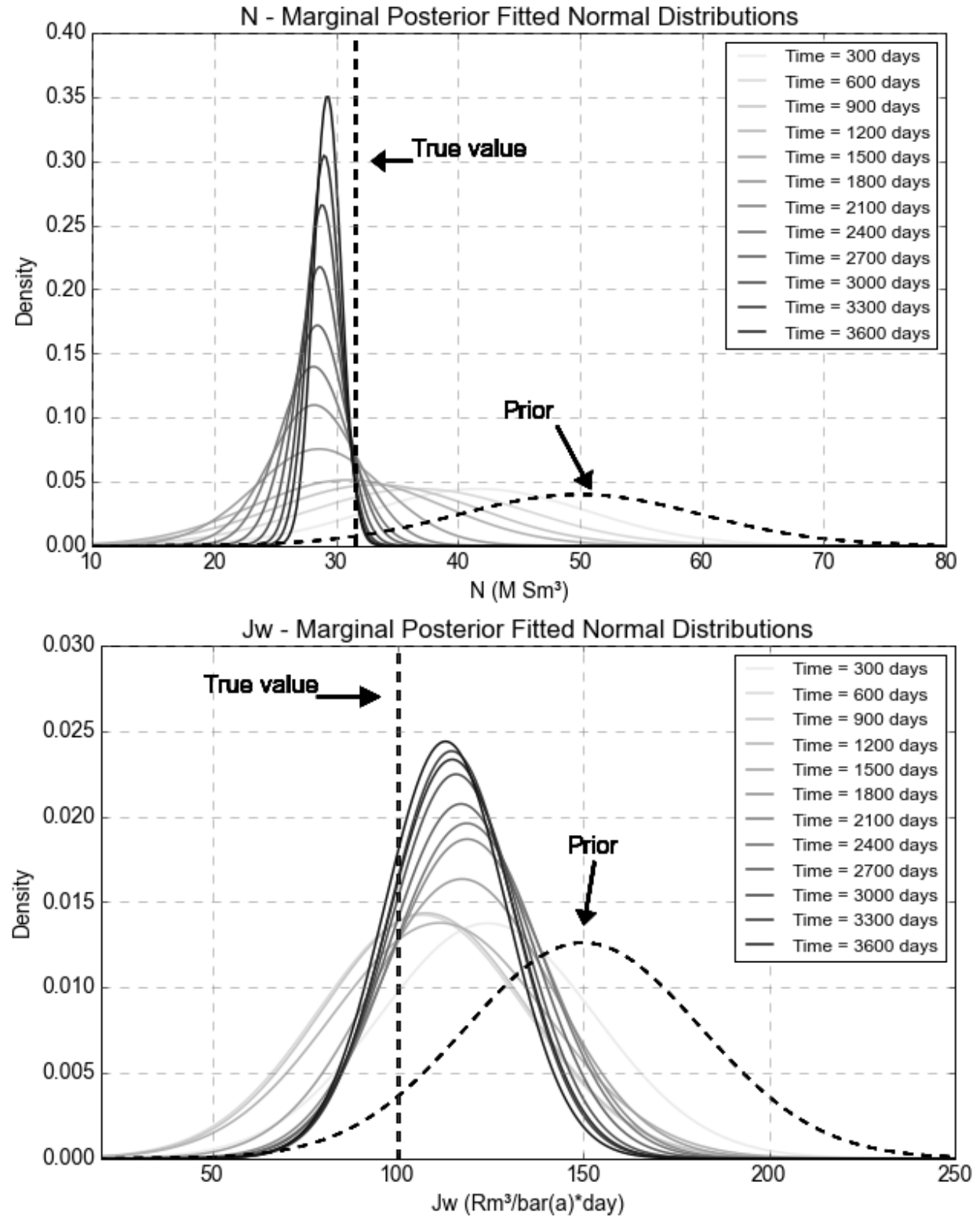


Figure 50 – Case 2, Posterior Marginal Fitted Normal Distributions

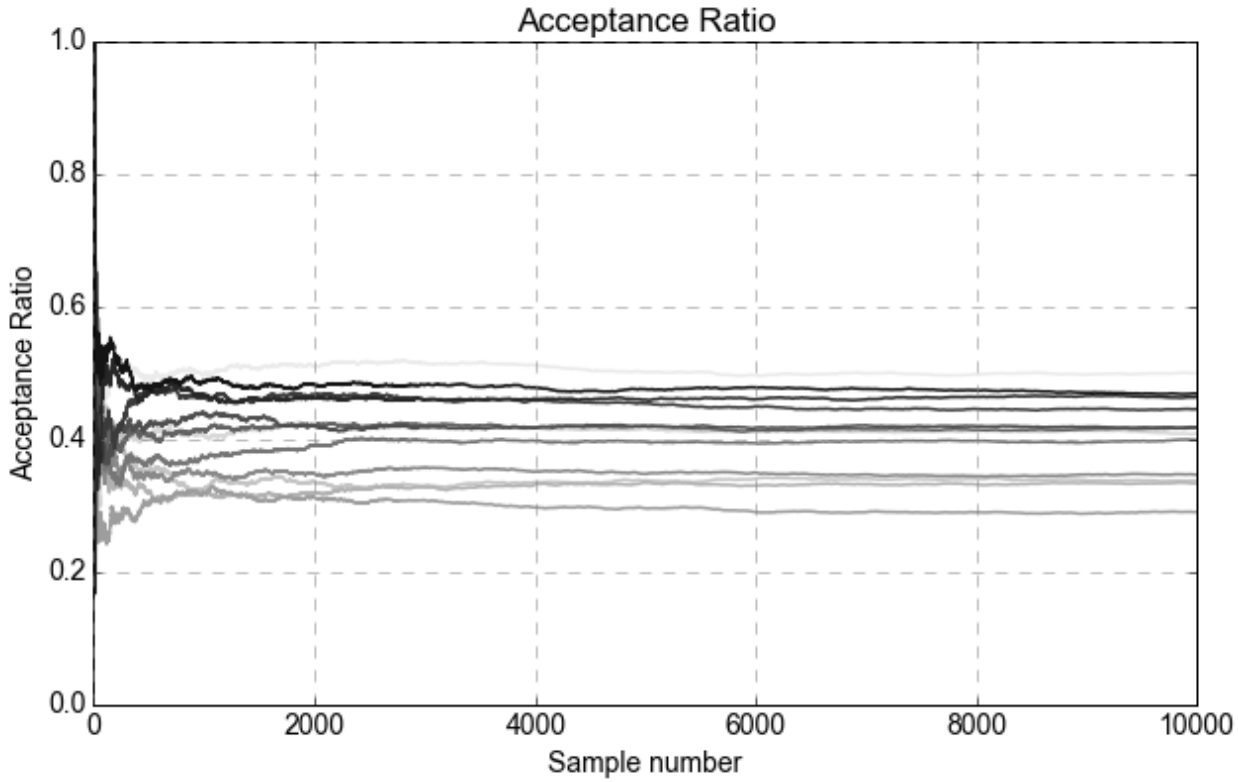


Figure 51 – Case 2, Acceptance Ratios

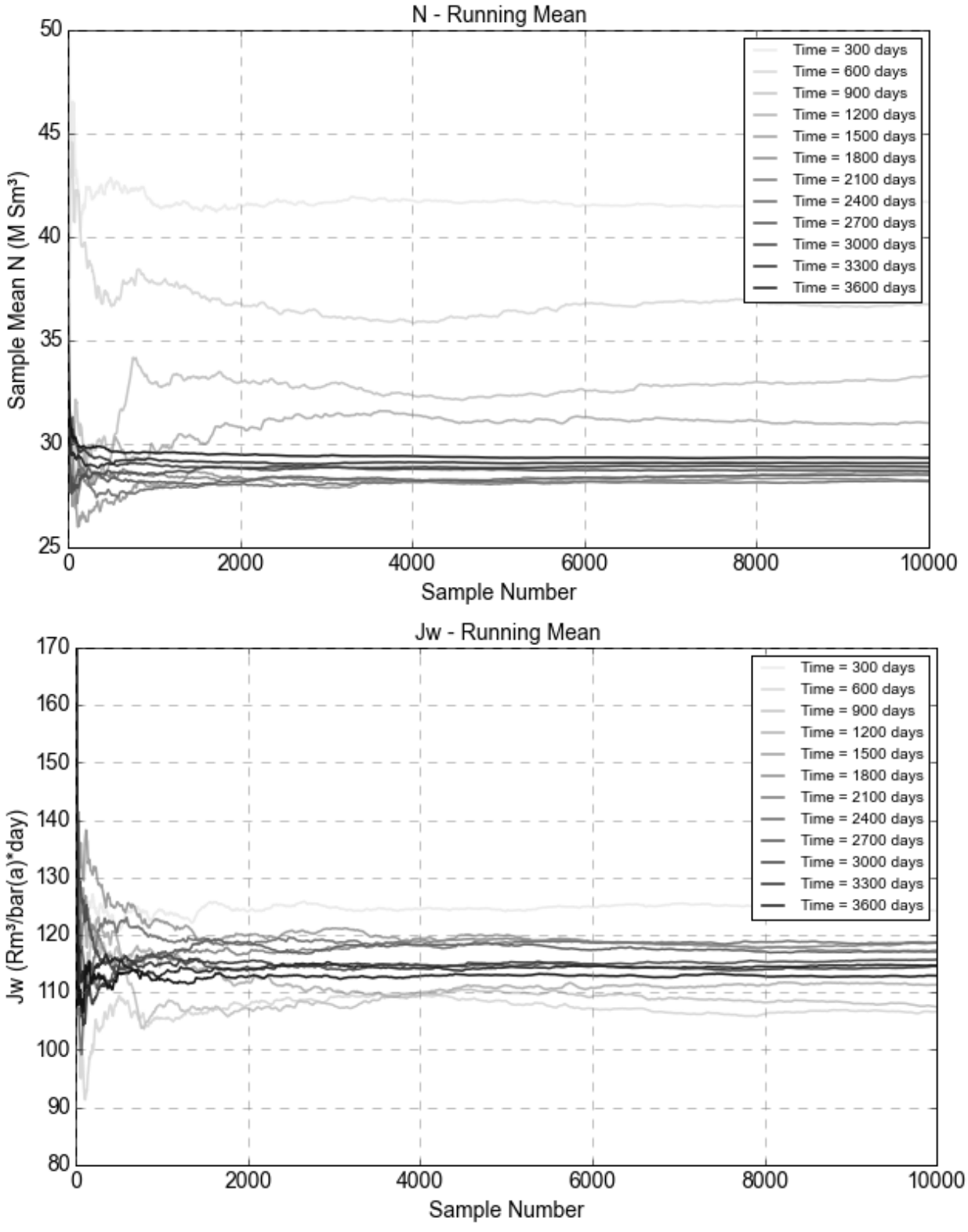


Figure 52 – Case 2, Running Mean Plots

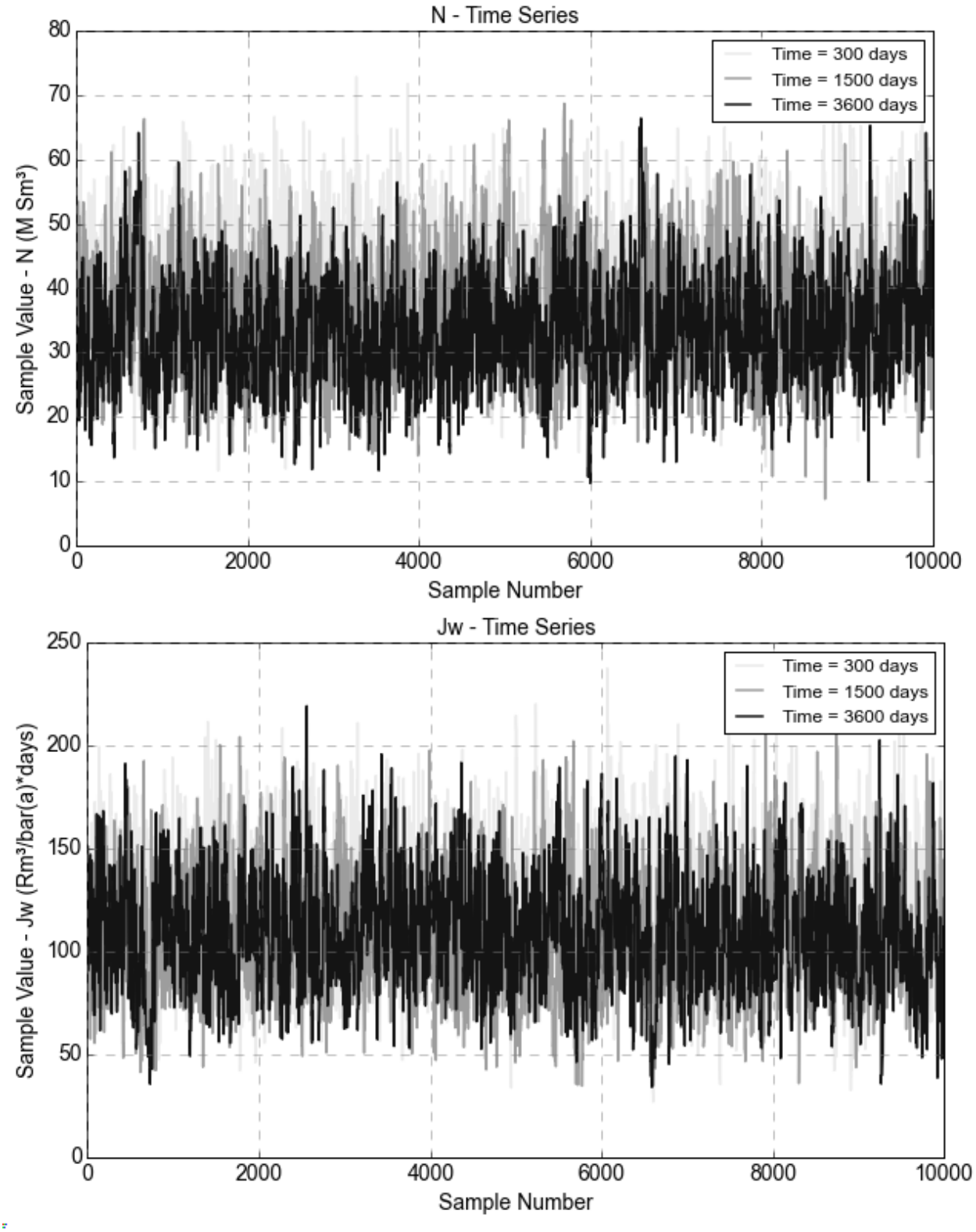


Figure 53 – Case 2, Time Series Plots

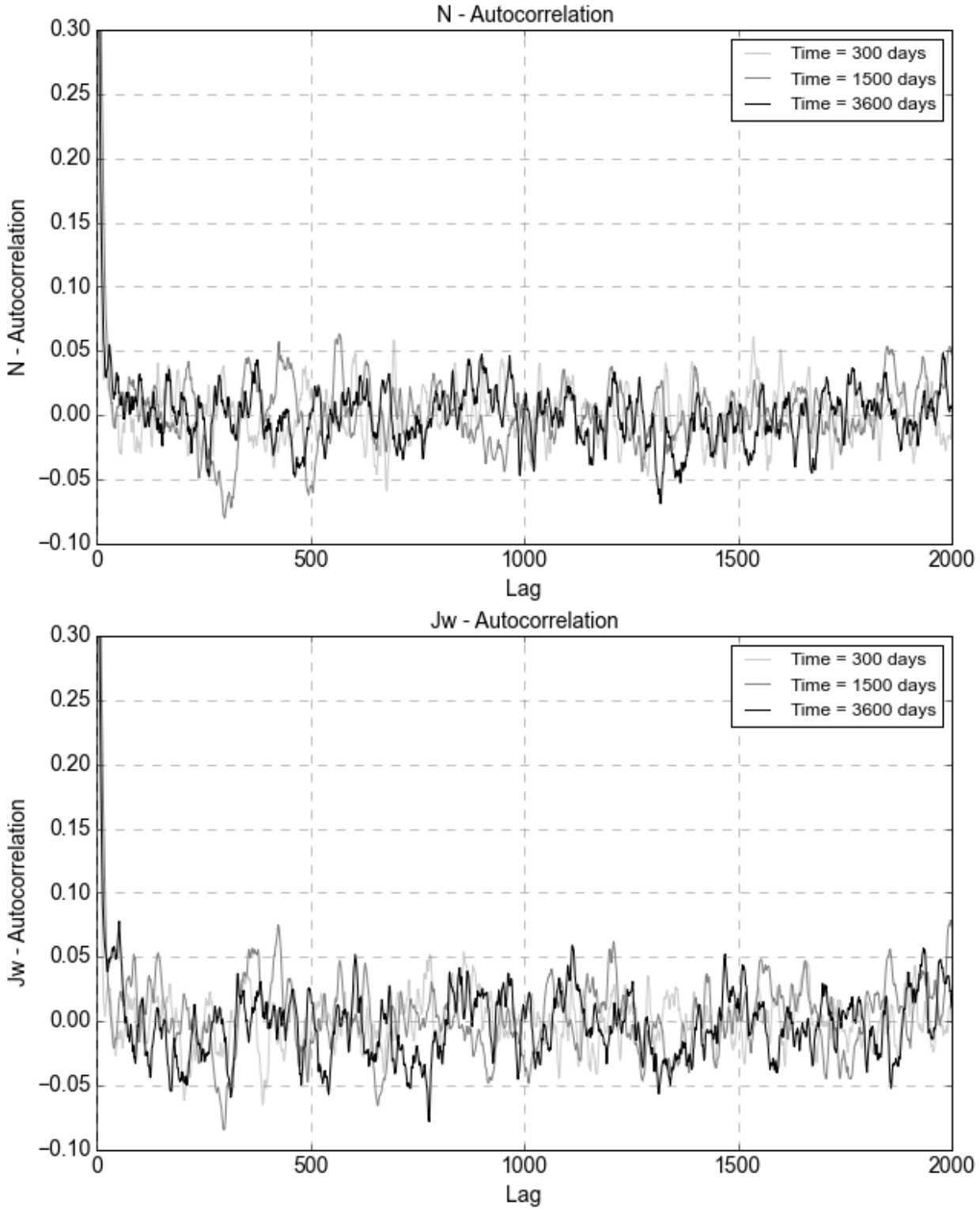


Figure 54 – Case 2, Autocorrelation Plots

3.6.1.3 Case 3 – Three-variable MCMC Sampling Based Solution

The purpose of Case 3, aquifer size W_i is added as a third random variable, meaning that the prior, likelihood and posterior cannot be visualized in three-dimensional plots. Instead, diagnostic plots useful for analyzing MCMC outputs are provided. Case 3 parmeters are summarized in Table 21.

Table 21 - Case 3 Parameters

Item	**Value**	**Unit**
Number of MCMC samples	10,000	-
Bayesian updating steps	12	-
Time steps	12 x 300	days
Likelihood variance	10	bar(a)^2
Aquifer Size Prior (Wi) - [mean, standard deviaton]	[600, 10]	M Sm^3
Oil in Place Prior (N) - [mean, standard deviation]	[50, 100]	M Sm^3
Aquifer Index Prior (Jw) Prior [mean, standard deviation]	[50, 20]	Rm^3/bar(a)*day

Figure 55 shows marginal posterior histogram outlines vs. time. It is evident that posterior variance decreases as data is assimilated. The same behavior is displayed in Figure 56, which shows fitted normal distributions vs. Bayesian updating steps. Overall, it is evident that the MCMC algorithm is correctly moving towards the true syntethic reservoir parameters as more data is incorporated. Figure 57 summarizes marginal histograms, scatter plots and correlation statistics at t = 300 days and t = 3600 days. The diagonal on these two 3x3 plots contanin the

marginal histograms. The plots above the diagonals show the calculated Pearson correlation coefficients for each variable pair. Initially, all variable combinations show little correlation. After about 12 updating steps (t = 3600 days), however, there is a exists strong linear correlation of -0.92 between oil in place N and aquifer size W_i. This occurs because a larger oil in place must correspond to a small aquifer and vice versa from a pressure response perspective. The scatter plots below the diagonal confirm the correlation between N and W_i as the samples fall on nearly straight line with slope -0.92. Figure 58, Figure 59 and Figure 60 display running mean, time series and autocorrelation vs. bayesian assimiliation step. All three plots show that the resulting Markov Chains exhibit good convergence and mixing properties. It is evident from the running mean plot (Figure 58) that burn-in is achieved after ~1,000 samples. Figure 61 shows that the acceptance ratio is relatively stable around a value of 0.2-0.3. This is lower than in case 1, which is explained by the fact that an additional random variable was introduced in case 3, thus causing the posterior region to grow in size and becoming more challenging for the MCMC algorithm to explore. Figure 62 shows posterior mean deviation from the true parameter values vs. time. The average deviation is over 40-60% initially, but after 12 updating steps the deviation has reuduced to about 5-15% of the true parameter values. Figure 63 shows posterior sample means and 95th percentile vs. time. It is clear that variances reduces as data is incoroprated into the analysis. Figure 64 shows 50 material balance realizations based on random N, Jw and W_i samples drawn from the posterior distributions at t = 300, 1500 and 3600 days. It is evident that uncertainty reduces as data is incorporated. As such, the difference between the measured data and the predicted data becomes increasingly smaller as the variance associated with the likelihood function reduces and starts dominiating in the posterior distribution.

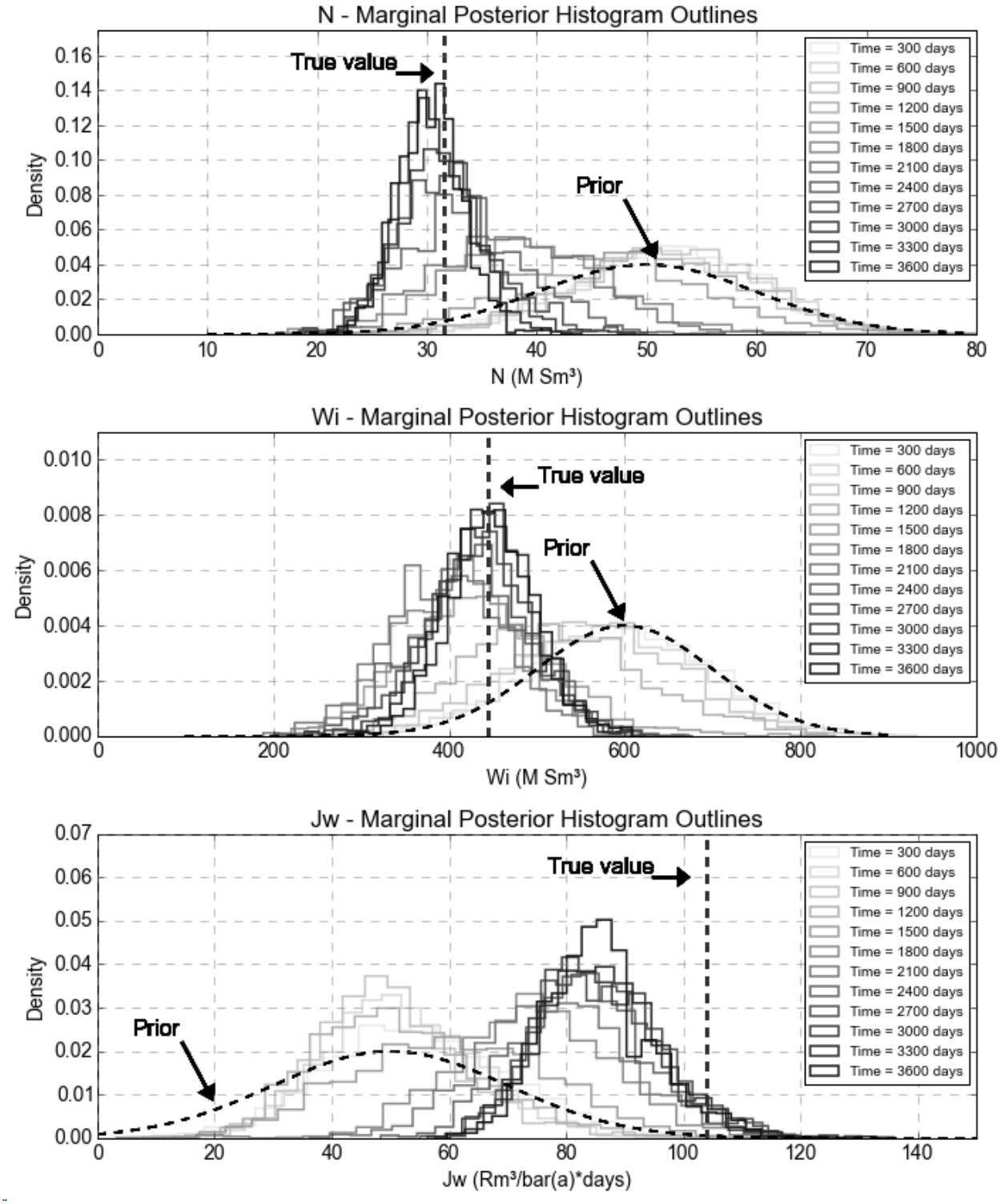


Figure 55 - Case 3, Posterior Marginal Histograms

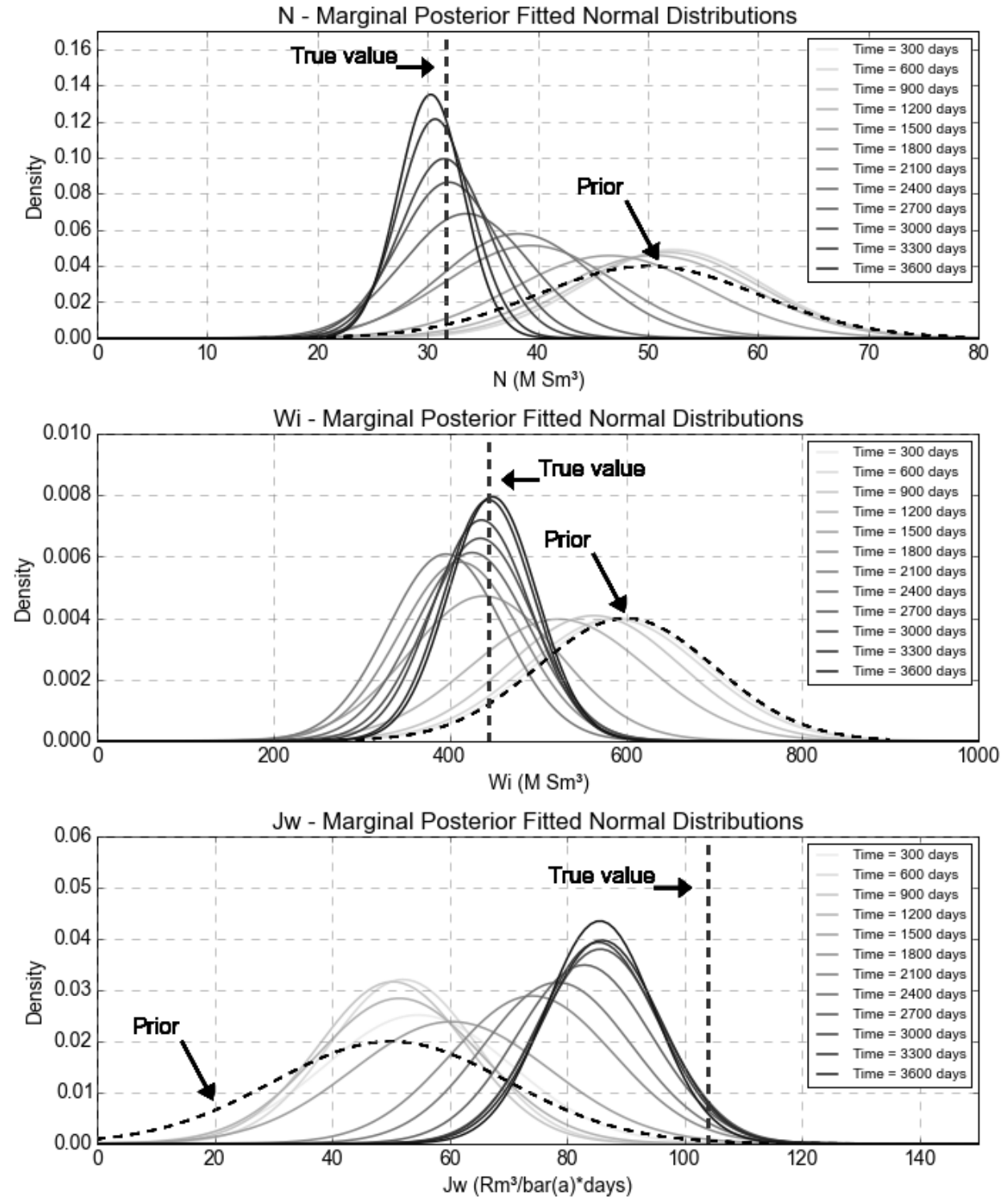


Figure 56 - Case 3, Posterior Fitted Normal Distributions

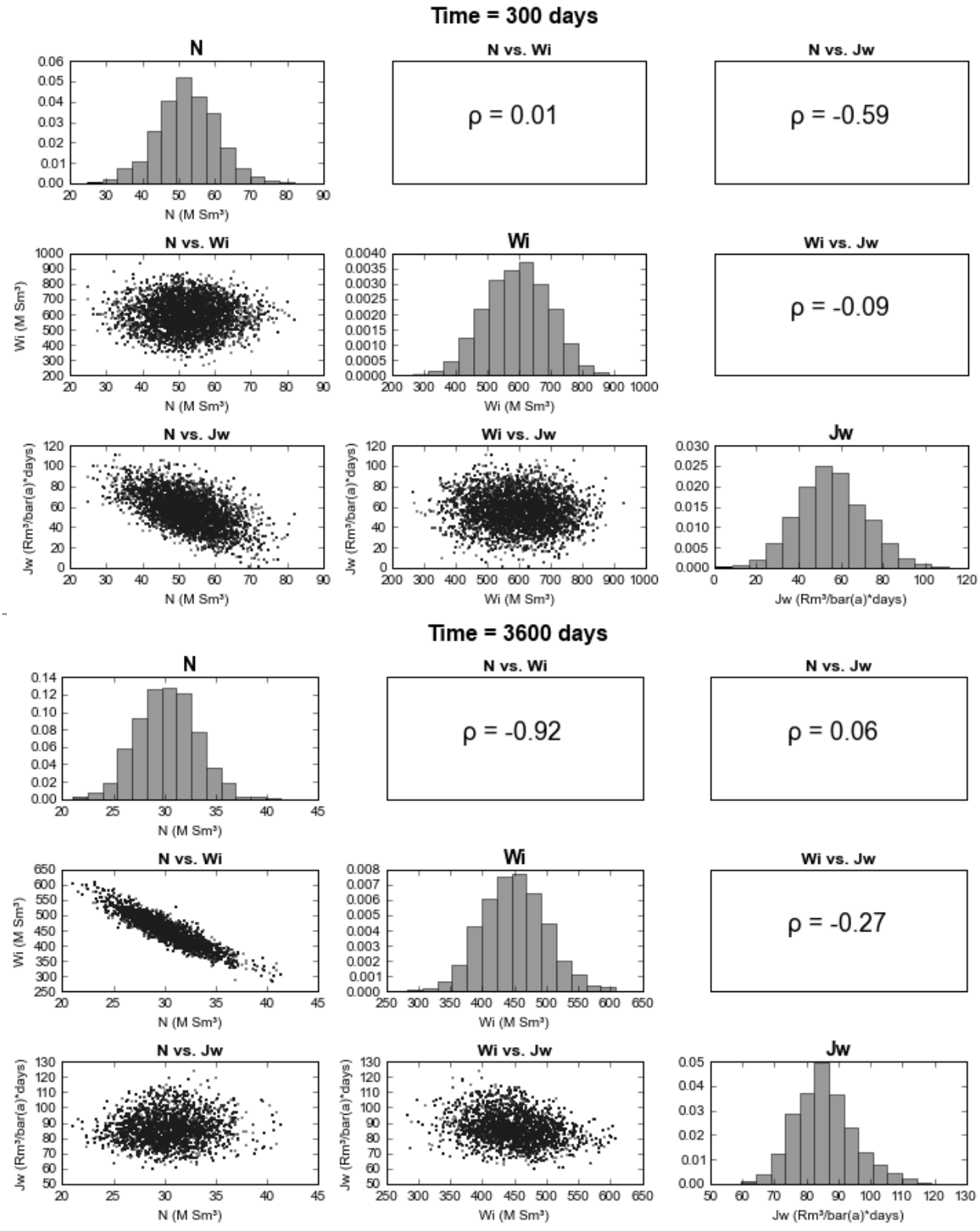


Figure 57 - Case 3, MCMC Summary Plot at time = 300 and 2600 days

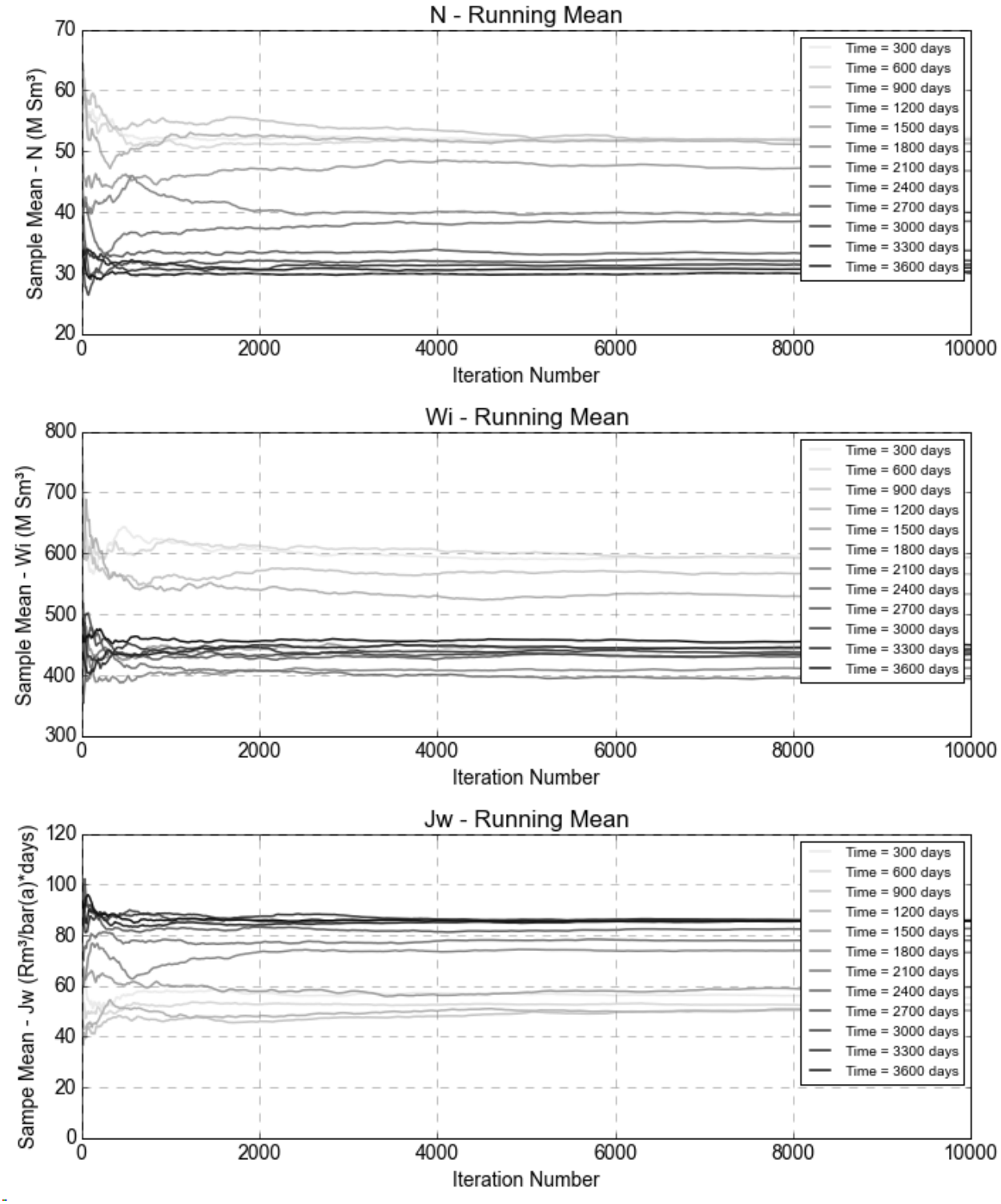


Figure 58 - Case3, Running Mean Plots

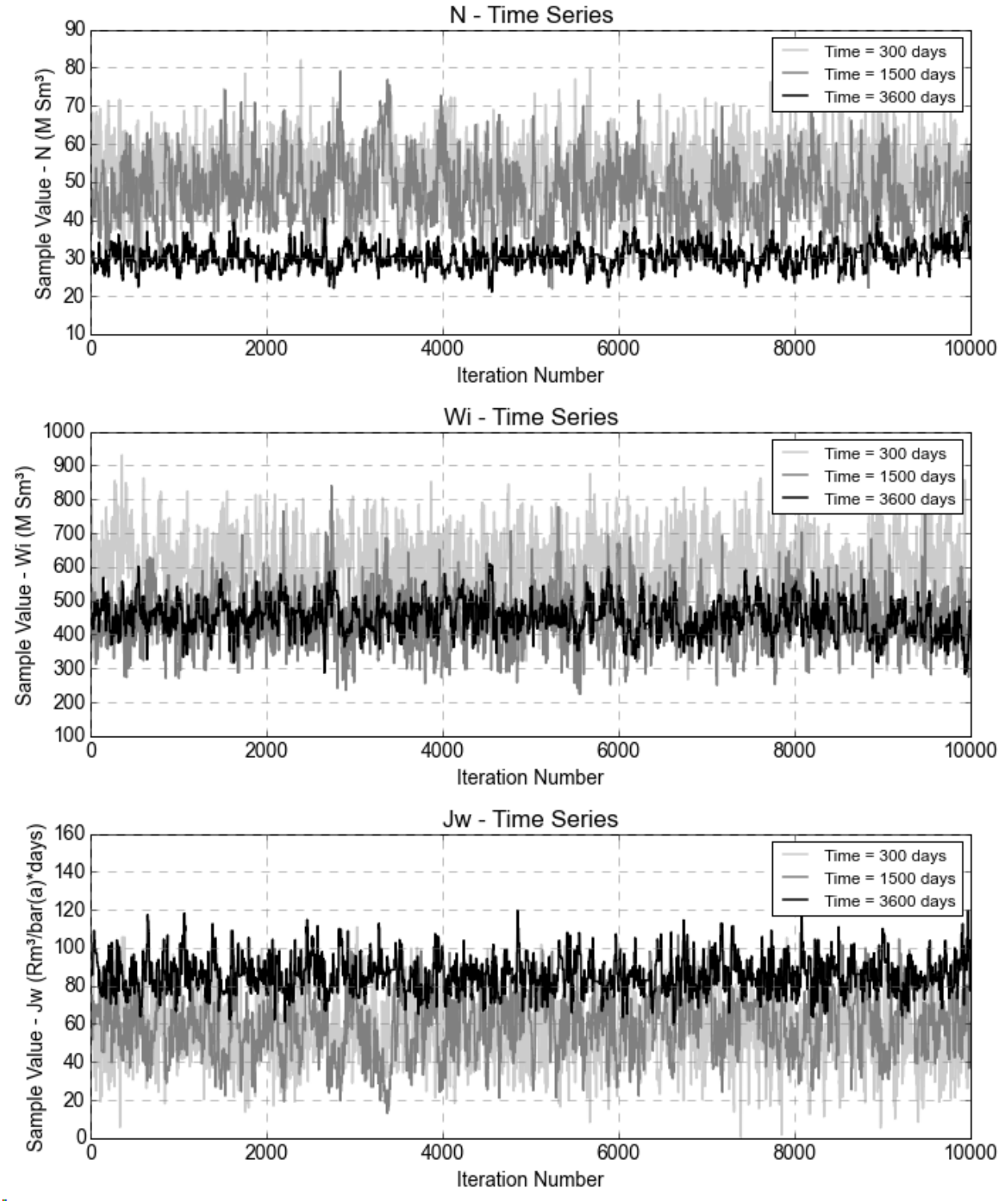


Figure 59 - Case 3, Time Series Plots

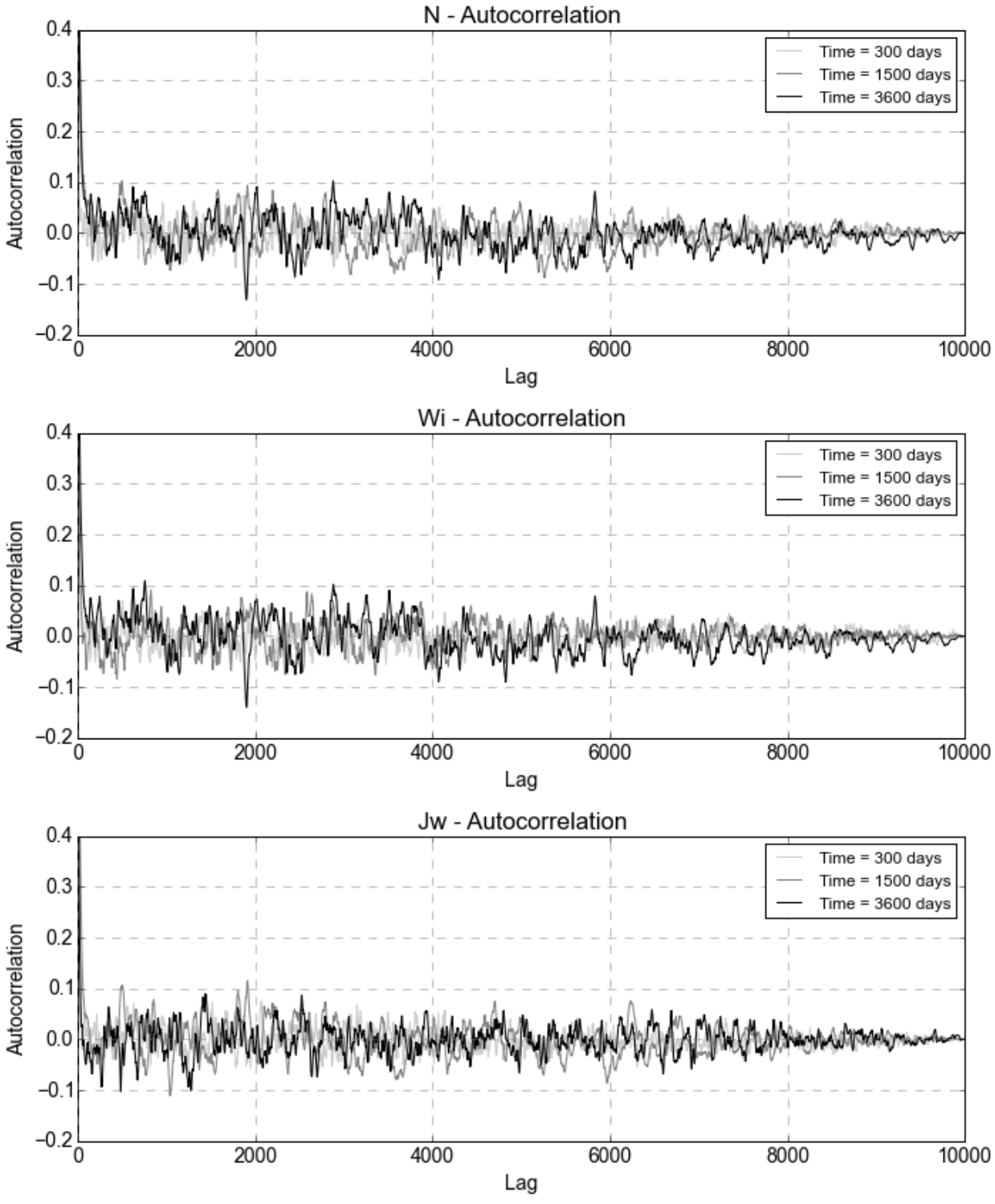


Figure 60 - Case 3, Autocorrelation Plots

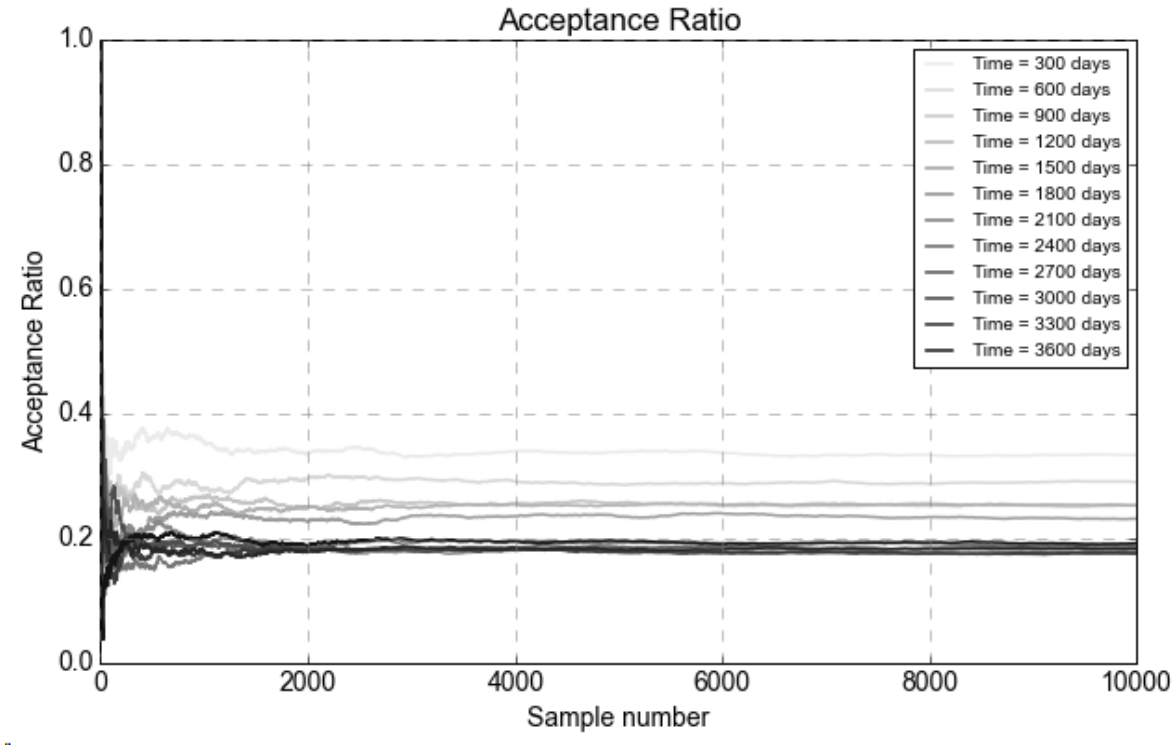


Figure 61 - Case3, Acceptance Ratios

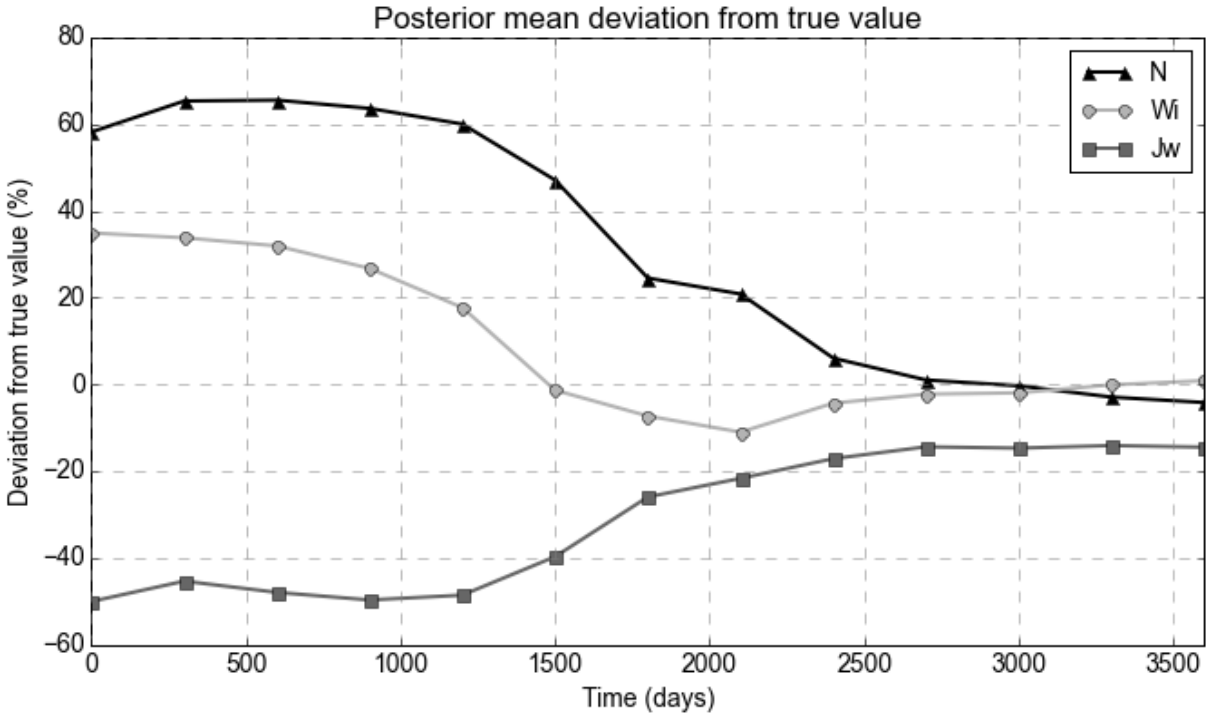


Figure 62 - Case 3, Deviation From True Parameter Values

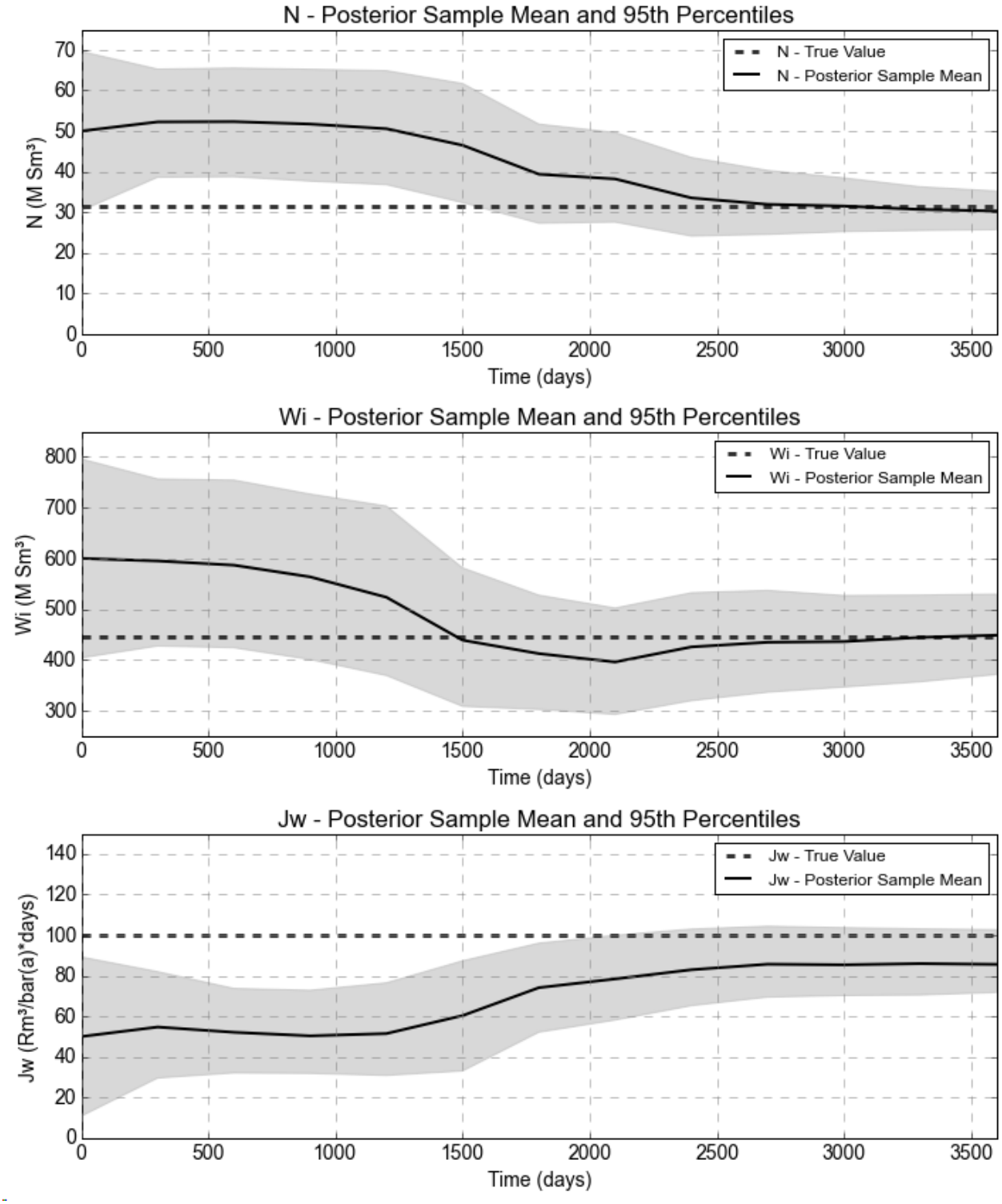


Figure 63 - Case 3, MCMC Posterior Means and 95th Percentiles

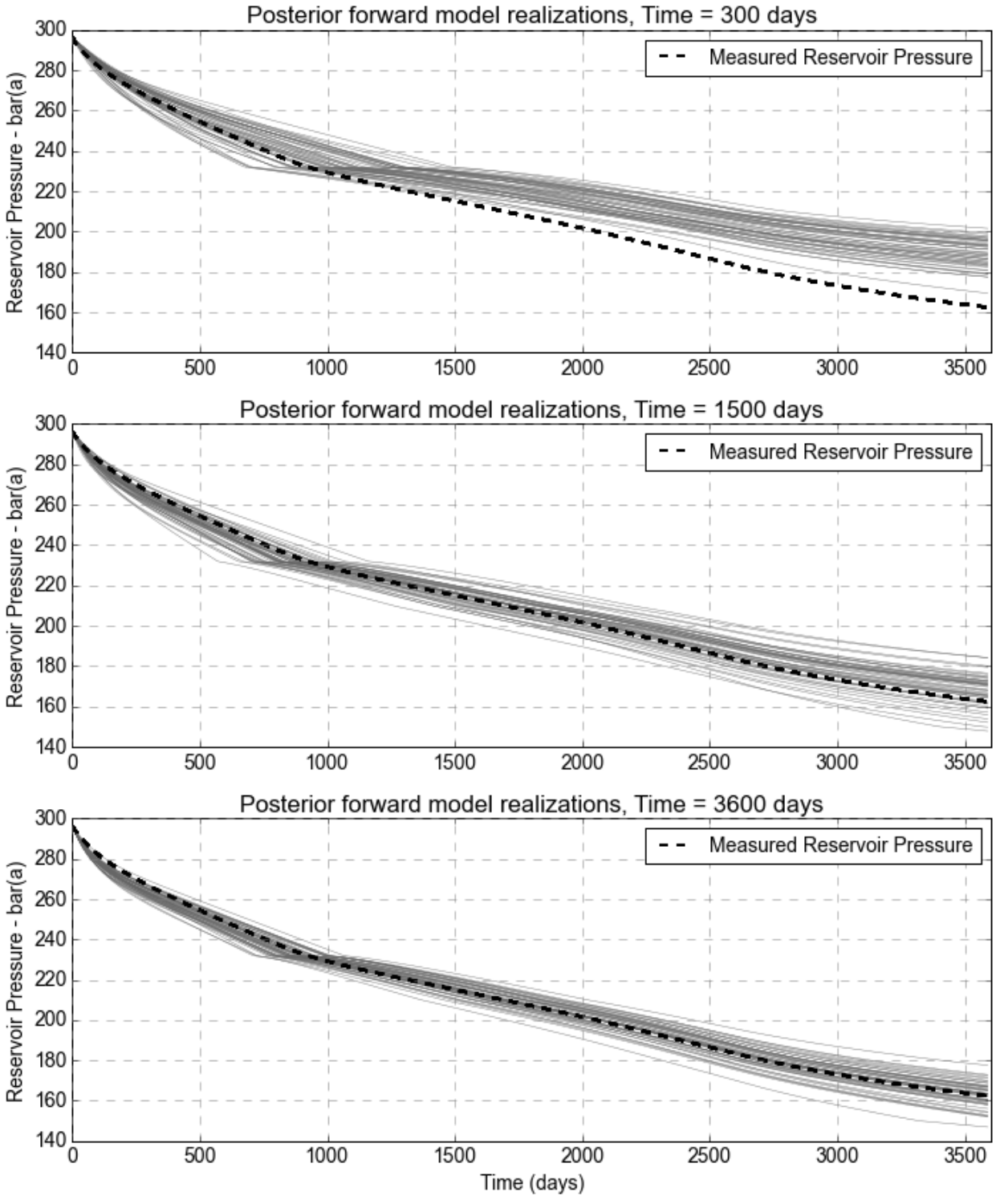


Figure 64 - Case 3, Posterior Material Balance Realizations, t=300, 1500 and 3600 days

3.6.1.4 Case 4 – Effect of Measurement Error

The purpose of this case is to demonstrate the effect of both consistent and random measurement errors. The general case parameters are summarized in Table 22. In this section only the posterior sample mean and associated 95th percentile plots are included as the previous sections have already demonstrated a full suite of MCMC convergence and diagnostic plots. First, a large constant pressure differential of 30 bar(a) is added to the synthetic reservoir pressure. This simulates consistent over-prediction of the measured pressure (Figure 65). It is clear that this measurement error causes the Bayesian material balance model to infer values different from the true parameter values (Figure 66). In particular, the oil in place N is estimated to be about 10 M Sm^3 larger than the true value. The error associated with the posterior distribution, however, is shown to somewhat mitigates this discrepancy by including the true parameter value in the vicinity of the 95th percentile of the posterior distribution. This demonstrates how Bayesian updating is an improvement over deterministic regression analyses, in that it provides a range of plausible parameters in addition to the parameters associated with maximum likelihood. To fully mititgate consistent measurement errors one would either have to shift the measured data prior to analysis with the Bayesian material balance model or apply a skewed likelihood function. To simulate inference on a noisy data set, random errors are added to the measured pressure (Figure 65). The random noise causes the posterior distribution to vary as each data point contradicts the parameters that were inferred in the previous time step (Figure 67). Next, the likelihood variance is increased to 50 bar(a)^2 to match the standard deviation of the random noise. This stabilizes the posterior distributions and increases overall posterior variance, but does not change the inferred mean values significantly (Figure 68). Overall, measurement noise has a significant effect on the

posterior parameter estimates. Since material balance is a longer-term analysis technique, it is recommended that long term trends are fitted to measured pressure data to reduce noise prior to incorporation into the Bayesian updating model.

Table 22 - Case 4 Main Parameters

Item	**Value**	**Unit**
Number of MCMC samples	10,000	-
Bayesian updating steps	12	-
Time steps	12 x 300	days
Aquifer Size Prior (Wi) - [mean, standard deviaton]	[600, 100]	M Sm^3
Oil in Place Prior (N) - [mean, standard deviation]	[50, 10]	M Sm^3
Aquifer Index Prior (Jw) Prior [mean, standard deviation]	[50, 20]	Rm^3/bar(a)*day

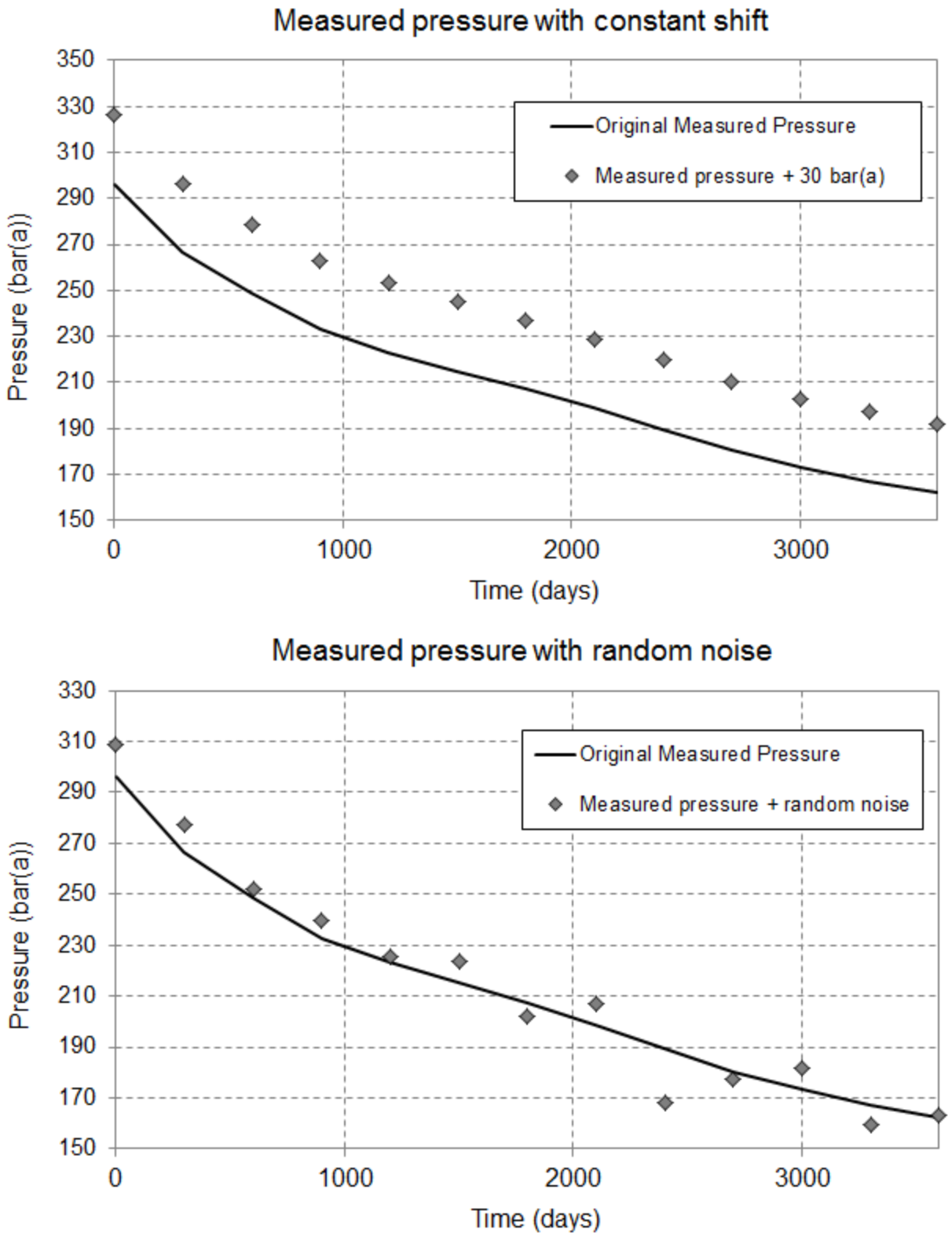


Figure 65 - Pressure data with constant shift and pressure data with random noise

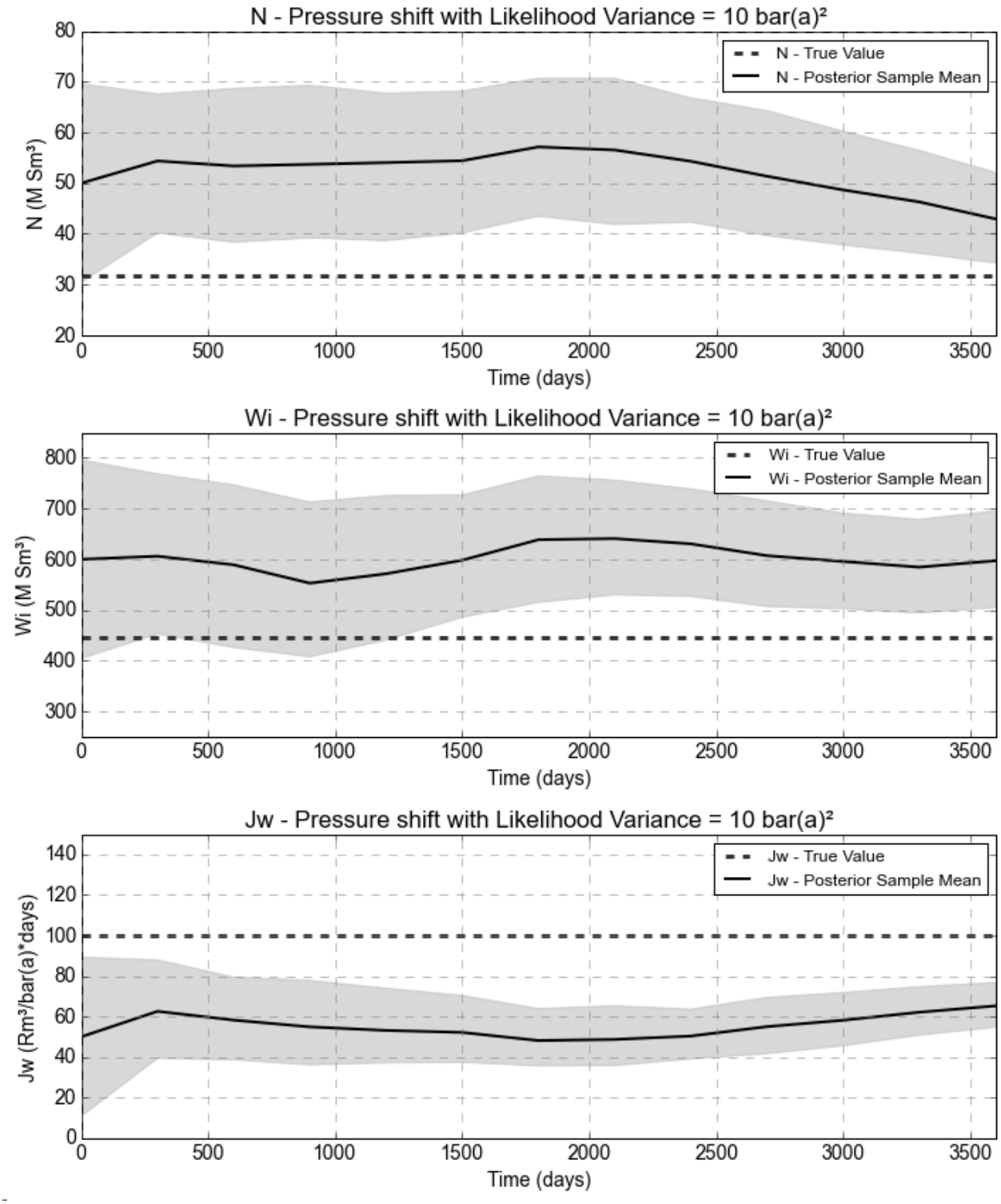


Figure 66 - Effect of constant pressure shift on posterior means and 95th percentiles

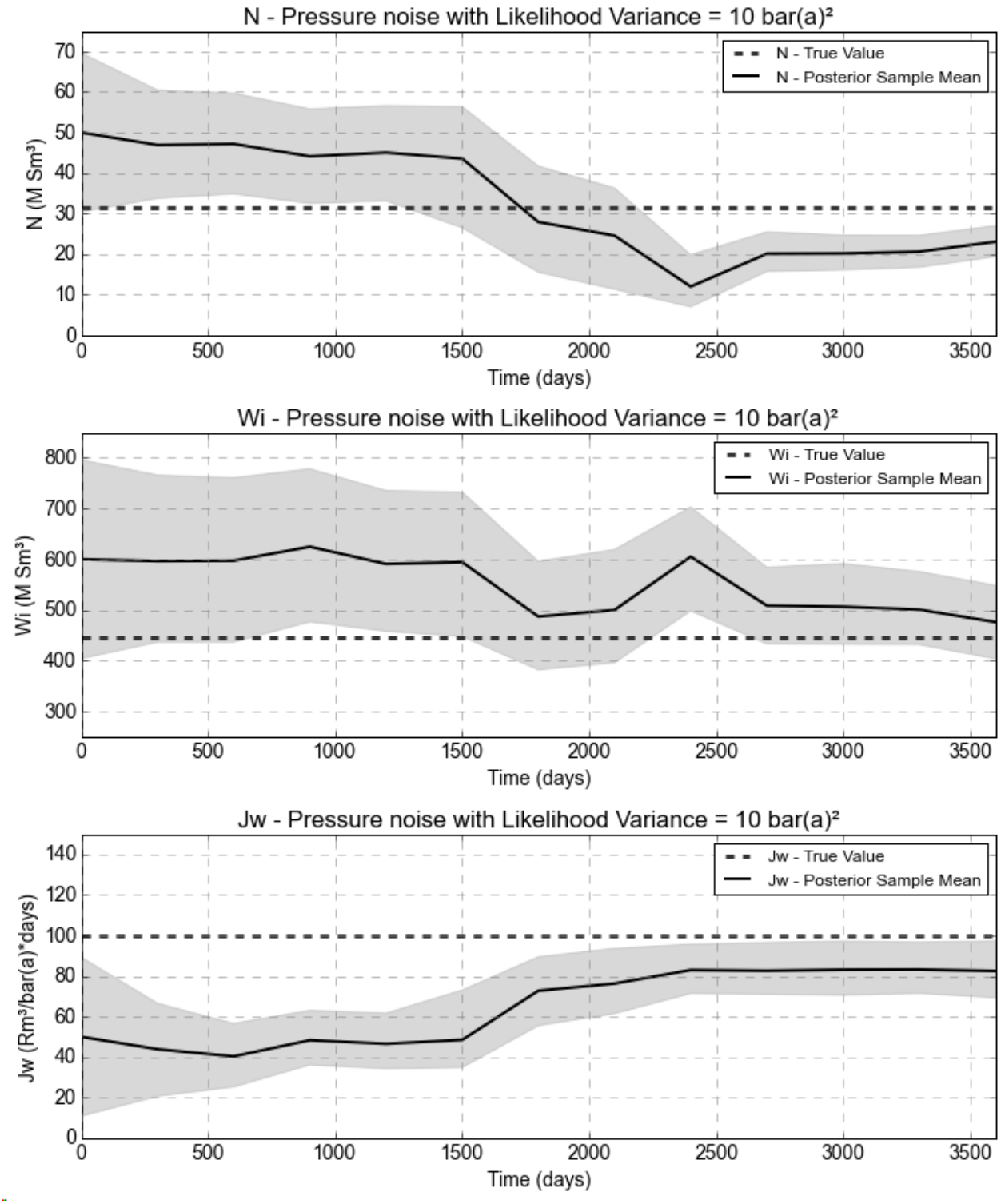


Figure 67 - Effect of random noise on posterior means and 95th percentiles

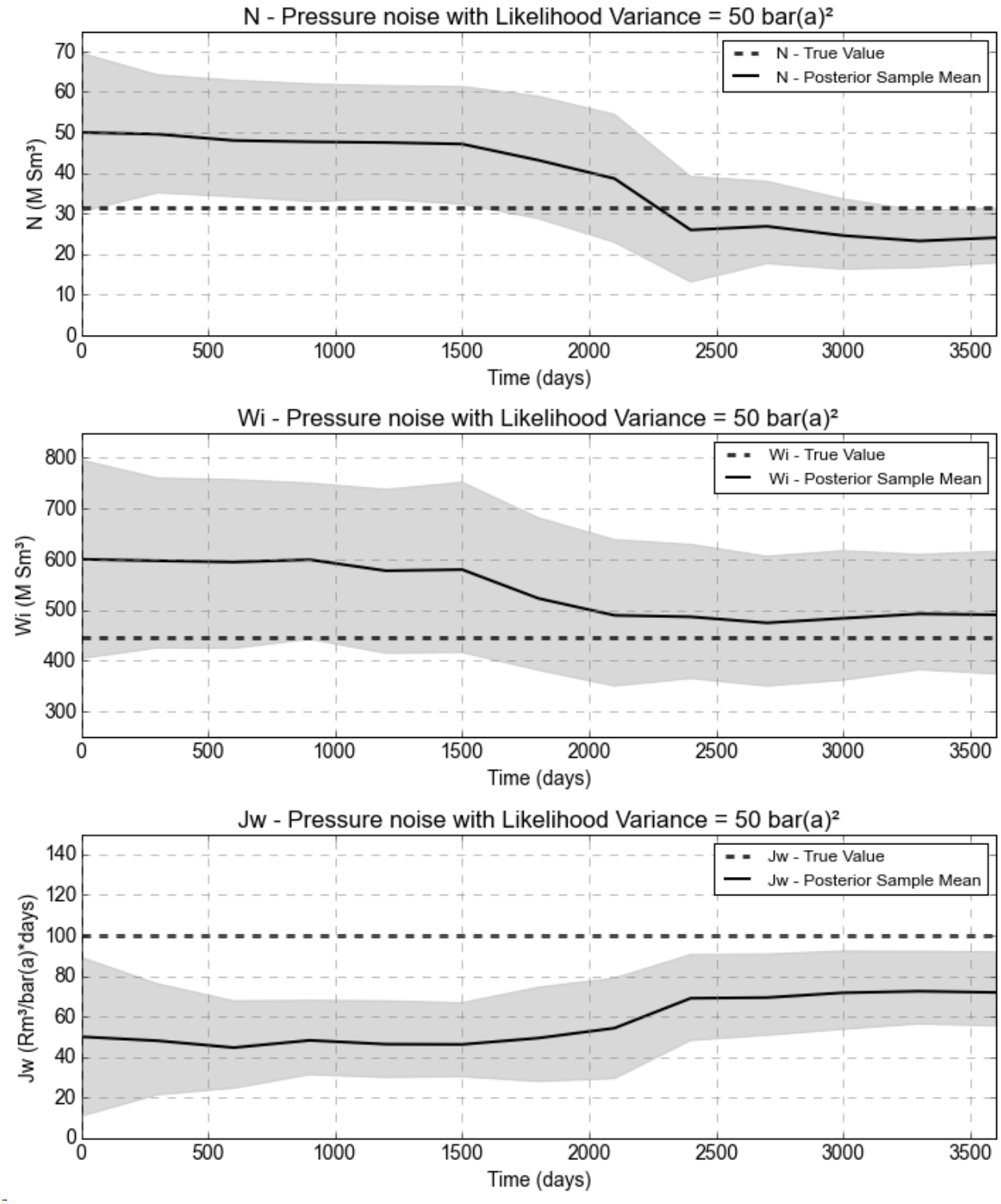


Figure 68 - Effect of noise on posterior means and 95th percentiles

3.7 Conclusion

This chapter presents a Bayesian analysis of the general material balance equation. Bayesian updating is shown to be a useful for estimating material balance parameters because the technique allows for full characterization of uncertainty. A summary of key findings is provided below:

- The grid based approach to Bayesian updating is shown to only be practical for two dimensions, due to the large number of additional forward model evaluations that are required with each added grid dimension.
- The MCMC approach is more efficient than the grid based approach as it reduces the number of required forward model evaluations.
- Good correspondence between the synthetic data set and the Bayesian updating models is observed. The MCMC model shows good convergence properties and replicates the posterior resulting from the grid based solution with high accuracy.
- Likelihood variance affects the rate at which the posterior assimilates information contained in the data.
- Data noise can have a significant effect on parameter estimates. Increasing the likelihood variance helps mitigate errors associated with random measurement noise.
- Material Balance with Bayesian updating can be placed into the broader reservoir engineering workflow as a technique for validating drive mechanisms and volumes in place probabilistically. The methodology represents an improvement over deterministic material balance in that it allows for full characterization of uncertainty. A sample

application would be to reconcile simulation model inputs with results obtained from Bayesian Material balance modeling (Figure 69).

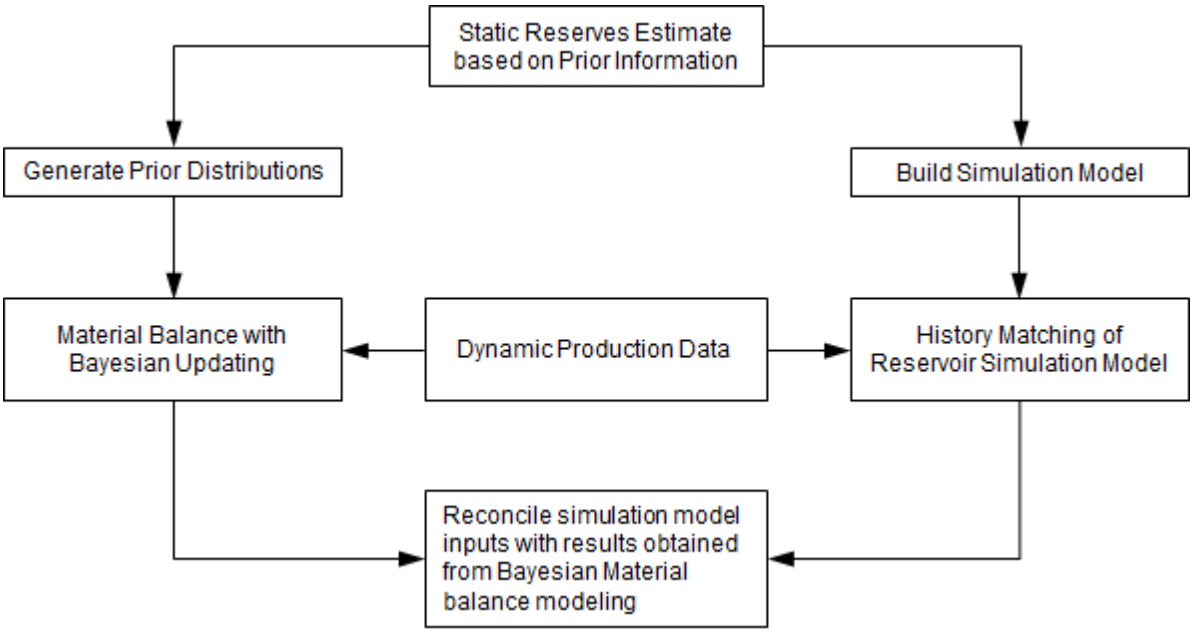


Figure 69 - Bayesian Material Balance in Reservoir Engineering Context

3.8 References

Aprilla, A.W., Li, Z., McVay D.A., Lee, W.J. (2006), Quantifying Uncertainty in Original-Gas-in-Place Estimates with Bayesian Integration of Volumetric and Material Balance Analyses, SPE 100575

Beggs , H.D. and Robinson, J.R. (1975), Esimtating the Viscosity of Crude Oil Systems, JPT, 27, 1140-1141

Brill, J. P. and Beggs, H. D. (1974) Two-Phase Flow in Pipes, INTERCOMP Course, The Hague

Brooks, S. (1998), Markov Chain Monte Carlo Method and Its Application, Journal of the Royal Statistical Society

Dake, L.P., (2001) The Practice of Reservoir Engineering (Revised Edition), Developments in Petroleum Science #36, Elsevier

Eclipse Reservoir Simulation Software (2010), Version 2010.2, Reference Manual

Fair, W.B. (1994), A Statistical Approach to Material Balance Methods, SPE 28629

Fetkovich, M. J. (1971), A Simplified Approach to Water Influx Calculations – Finite Aquifer Systems, Journal of Petroleum Technology

Gamerman, D. (2002), Markov Chain Monte Carlo, Stochastic Simulation for Bayesian Inference, Chapman & Hall/CRC

Geweke J. (1992), Evaluating the accuracy of sampling based approaches to the calculation of posterior moments. Bayesian Statistics 4, pp. 169-193,

Glaso, O. (1980), Generalized Pressure-Volume-Temperature Correlations, JPT ,785-95.

Hastings, W.K (1970), Monte Carlo sampling methods using Markov Chains and their applications, Biometrika, 57(1), 97-109

Kelkar, M., Perez G. (2002), Applied Geostatistics for Reservoir Characterization, SPE Book

Lee, A.M, Gonzalez, M.H. and Eakin, B.E. (1966), The Viscosity of Natural Gases, JPT, 997-1000.

McEwen, C.R. (1962), Material Balance Calculations with Water Influx in the Presence of Uncertainty in Pressures, SPE 225

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N, Teller, A.H & Teller, E. (1953), Equation of state calculations by fast computing machines, The Journal of Chemical Physics 21(6), 1087

Ogele, C. (2005), Integration and Quantification of Uncertainty of Volumetric and Material Balance Analyses Using a Bayesian Framework, SPE Conference Paper

Oliver, D.S., Reynolds, A.C., Liu, N. (2008), Inverse Theory for Petroleum Reservoir Characterization and History Matching,Cambridge University Press

Ottah, D. G. (2015), Aquifer Matching with Material Balance Using Particle Swarm Optimization Algorithm – PSO, SPE-178319-MS

Sills, S.R. (1996), Improved Material-Balance Regression Analysis for Waterdrive Oil and Gas Reservoirs, SPE 28630

Sutton, R.P (1985), Compressibility Factors for High Molecular Weight Reservoir Gases, SPE 14265, Proc. Of 60th Ann. Tech. conf

Tangen, M. (2012), Wettability Variations within the North Sea Oil Field Froy, Master's Thesis, Noregian University of Science and Technology, Earth Sciences and Petroleum Engineering Department

Tarantola, A. (2005), Inverse Problem Theory and Methods for Model Parameter Estimation, SIAM

Van Everdingen, A. F., Timmerman, E.H. (1953), Application of the Material Balance Equation to a Partial Water-Drive Reservoir, SPE 229-G, vol, 198

Vasquez, M. and Beggs, H. D (1980), Correlations for Fluid Physical Property Prediction, JPT 968-970

S. Särkkä (2013), Bayesian Filtering and Smoothing, Cambridge University Press

G. Evensen (2003), The Ensemble Kalman Filter: Theoretical Formulation and Practical Implementation , Ocean Dynamics, 53, 343-367

Appendix A – PEWMA Python Code

```
import argparse
import numpy as np
import scipy as sp
from scipy import special, optimize
import matplotlib.pyplot as plt
from scipy.special import gamma as Gamma
from pylab import *
import scipy.stats as ss
from scipy.stats import gamma
from scipy.stats import beta
from scipy.optimize import minimize
import scipy.misc
import pylab as pl
import statistics

plt.rc('font',family='Arial')

def PEWMA_filter(y,omega,a_prior,b_prior):

    a_predicted = np.zeros(len(y))
    b_predicted = np.zeros(len(y))
    a_updated = np.zeros(len(y))
    b_updated = np.zeros(len(y))
    return_array = []

    for t in range(0,len(y)):

        if(t == 0):
            a_predicted[t] = a_prior
            b_predicted[t] = b_prior
            a_updated[t] = a_predicted[t]+ y[t]
            b_updated[t] = b_predicted[t] + 1
        elif(t > 0):
            a_predicted[t] = a_updated[t-1]*omega
            b_predicted[t] = b_updated[t-1]*omega
            a_updated[t] = a_predicted[t] + y[t]
            b_updated[t] = b_predicted[t] + 1

    return_array = np.column_stack((a_predicted, b_predicted, a_updated,b_updated))

    return return_array

def omega_log_likelihood(w,y,a_prior,b_prior,index_non_zero):

    a_updated = np.zeros(len(y))
    b_updated = np.zeros(len(y))
    a_predicted = np.zeros(len(y))
    b_predicted = np.zeros(len(y))
    log_likelihood = 0
    for i in range(0,len(y)):

        if(i == 0):
            a_predicted[i] = a_prior
            b_predicted[i] = b_prior
            a_updated[i] = a_predicted[i] + y[i]
```

```
            b_updated[i] = b_predicted[i] + 1
        else:
            a_predicted[i] = a_updated[i-1]*w
            b_predicted[i] = b_updated[i-1]*w
            a_updated[i] = a_predicted[i] + y[i]
            b_updated[i] = b_predicted[i] + 1

        if i > index_non_zero:
            log_likelihood = log_likelihood + (math.log(math.gamma(a_predicted.item(i) + y[i])) - math.log(math.gamma(y[i] + 1)) - math.log(math.gamma(a_predicted.item(i))) +
a_predicted.item(i)*math.log(b_predicted.item(i)) - (a_predicted.item(i) + y[i])*math.log(1+b_predicted.item(i)))

    return -log_likelihood

def PEWMA(failure_data, a_prior, b_prior, optimize_omega, constant_omega):

    omega = np.zeros(len(failure_data))
    a_predicted = np.zeros(len(failure_data))
    b_predicted = np.zeros(len(failure_data))
    a_updated = np.zeros(len(failure_data))
    b_updated = np.zeros(len(failure_data))
    year = failure_data[:,0]
    y = failure_data[:,1]

    index_non_zero = 0
    #Finding the first zero in the array
    for t in range(0,len(y)):
        if(y[t] > 0.0):
            index_non_zero = t
            break

    for t in range(0,len(y)):

        if(optimize_omega == 1):
            if(t < index_non_zero):
                omega[t] = 1
            else:
                temp_optimal = scipy.optimize.minimize(omega_log_likelihood, [0.5],args=(y[0:t+1],a_prior,b_prior,index_non_zero),method='L-BFGS-B',bounds=((0.01,0.99),))
                omega[t] = temp_optimal.x
        else:
            omega[t] = constant_omega

        PEWMA_output = PEWMA_filter(y[0:t+1],omega[t],a_prior,b_prior)
        a_predicted[t] = PEWMA_output[t,0]
        b_predicted[t] = PEWMA_output[t,1]
        a_updated[t] = PEWMA_output[t,2]
        b_updated[t] = PEWMA_output[t,3]

    dist_type = np.zeros(len(y))
    return_array = np.column_stack((dist_type, year, a_updated, b_updated,a_predicted,b_predicted))
    return return_array

def generate_samples(input_data,num_samples):
    sample_array = []
    if(input_data[0] == 1):
        sample_array = np.random.uniform(input_data[2],input_data[3],num_samples)
    elif(input_data[0] == 0):
        sample_array = np.random.gamma(input_data[2],1/input_data[3],num_samples)
    return sample_array
```

```
def plot_PEWMA_mean_vs_time(PEWMA, failure_data, xlim, ylim, fignum, figname, title): #, title):

    PEWMA_mean = np.zeros(len(PEWMA))
    PEWMA_95 = np.zeros(len(PEWMA))
    PEWMA_5 = np.zeros(len(PEWMA))

    #Original dims, 10, 6
    fig = plt.figure(fignum, figsize=(11,12),dpi=1100)
    for t in range(0, len(PEWMA)):
        PEWMA_95[t] = gamma.ppf(0.95,PEWMA[t,2], 0, 1/PEWMA[t,3])
        PEWMA_5[t] = gamma.ppf(0.05,PEWMA[t,2], 0, 1/PEWMA[t,3])
        PEWMA_mean[t] = PEWMA[t,2]/PEWMA[t,3]

    plt.figure(fignum)
    ax = fig.add_subplot(211)
    ax.fill_between(PEWMA[:,1], PEWMA_5, PEWMA_95, color=str(0.7), alpha='0.5')
    ax.plot(PEWMA[:,1], PEWMA_mean, linewidth=2, color=str(0))
    ax2 = ax.twinx()
    ax2.plot(failure_data[:,0], failure_data[:,1], linestyle="None",marker="o",markersize=6,color='black')
    ax.set_xlim((xlim[0],xlim[1]))
    ax2.set_ylim((ylim[0],ylim[1]))
    ax.set_ylim((ylim[0],ylim[1]))

    ax.grid(color='gray',linestyle='dashed')
    plt.setp(ax.get_yticklabels(), rotation='horizontal', fontsize=15, style='normal', Family="Arial")
    plt.setp(ax2.get_yticklabels(), rotation='horizontal', fontsize=15, style='normal', Family="Arial")
    plt.setp(ax.get_xticklabels(), rotation='horizontal', fontsize=15, style='normal', Family="Arial")

    ax.set_ylabel("Posterior Mean Failure Rate (Failures/Year)", fontsize=15,family="Arial")
    ax.set_xlabel("Year", fontsize=15,family="Arial")
    ax.set_ylabel("Posterior Mean Failure Rate", fontsize=15,family="Arial")
    ax2.set_ylabel("Actual Failures", fontsize=15,family="Arial")
    plt.legend(fontsize=11)
    ax.set_title(title, fontsize=15,family="Arial")

    plt.savefig(figname,dpi=600)

def plot_PEWMA_mean_vs_omega(failure_data, a_prior, b_prior, omega_values, legend_loc, ylim, fignum, figname, title):

    fig = plt.figure(fignum, figsize=(10,10),dpi=1100)

    plt.figure(fignum)
    ax = fig.add_subplot(211)

    for i in range(1,len(omega_values)):

        PEWMA_output = PEWMA(failure_data,a_prior,b_prior,0,omega_values[i])

        plt.figure(fignum)
        ax = fig.add_subplot(111)
        ax.plot(PEWMA_output[:,1], PEWMA_output[:,2]/PEWMA_output[:,3], linewidth=2, color=str(0.9 - i/6), label="Omega = " + str(omega_values[i]))
        ax.set_xlim((1987,2005))
        ax.set_ylim((0,ylim))
        plt.setp(ax.get_yticklabels(), rotation='horizontal', fontsize=10, style='normal', Family="Arial")

    PEWMA_output = PEWMA(failure_data,a_prior,b_prior,1,1)
```

```
    plt.figure(fignum)
    ax = fig.add_subplot(111)
    ax.plot(PEWMA_output[:,1], PEWMA_output[:,2]/PEWMA_output[:,3], linewidth=1.5, color="black", linestyle="dashed",label="Optimized")
    ax2 = ax.twinx()
    ax2.plot(failure_data[:,0], failure_data[:,1], linestyle="None",marker="o",markersize=5,color='black')
    ax.set_xlim((1987,2005))
    ax2.set_ylim((0,ylim))
    ax.set_ylim((0,ylim))

    ax.grid(color=str(0.8),linestyle='dashed')
    plt.setp(ax.get_yticklabels(), rotation='horizontal', fontsize=15, style='normal', Family="Arial")
    plt.setp(ax2.get_yticklabels(), rotation='horizontal', fontsize=15, style='normal', Family="Arial")
    plt.setp(ax.get_xticklabels(), rotation='horizontal', fontsize=15, style='normal', Family="Arial")

    ax.set_ylabel("Posterior Mean Failure Rate (Failures/Year)", fontsize=15,family="Arial")
    ax2.set_ylabel("Actual Failures", fontsize=15,family="Arial")
    ax.legend(loc=legend_loc, shadow=False, fontsize='x-large')
    ax.set_title(title, fontsize=15,family="Arial")

    plt.savefig(figname,dpi=600)

def plot_gamma_updating(PEWMA, a_prior, b_prior, legend_loc, xlim, fignum, figname, title):

    fig = plt.figure(fignum, figsize=(9.5,10),dpi=1100)
    ax = fig.add_subplot(212)
    plt.tight_layout()

    years = np.linspace(0,len(PEWMA),len(PEWMA))

    x = np.arange(0,xlim,0.01)
    y = np.zeros(len(x))

    norm = matplotlib.colors.Normalize(vmin=np.min(years[0]), vmax=np.max(years[len(years)-1]))
    c_m = matplotlib.cm.Greys
    s_m = matplotlib.cm.ScalarMappable(cmap=c_m, norm=norm)
    s_m.set_array([])

    for t in range(0, len(PEWMA)):

        for i in range(0,len(x)):
            y[i] = gamma.pdf(x[i],PEWMA[t,2],0,1/PEWMA[t,3])
        plt.figure(fignum)

        #ax.plot(x,y, linewidth=1.5, color=s_m.to_rgba(t))
        ax.plot(x,y, linewidth=2, color=s_m.to_rgba(t), label="t = " + str(t))

    plt.figure(fignum)
    ax.set_xlim((0,xlim))
    ax.grid(color='gray',linestyle='dashed')
    ax.set_ylabel("Probability Density", fontsize=15,family="Arial")
    ax.set_xlabel("Failure Rate (Failures/Year)", fontsize=15,family="Arial")
    plt.setp(ax.get_yticklabels(), rotation='horizontal', fontsize=15, style='normal', Family="Arial")
    plt.setp(ax.get_xticklabels(), rotation='horizontal', fontsize=15, style='normal', Family="Arial")
    #ax.legend(loc=legend_loc, shadow=False, fontsize='small')
    plt.legend(fontsize=12)
    #plt.colorbar(s_m)
    ax.set_title(title, fontsize=15,family="Arial")
```

```
    plt.savefig(figname,dpi=600, figsize=(8,5))

def sample_exponential_failure_times(failure_rate):

    failure_times = []
    cum_time = 0
    integer_failure_time = 0
    failure_array = zeros(12)

    cum_time = 0

    while(cum_time < 12):

        random_number = np.random.uniform(0,1)
        temp = -np.log(1 - random_number)/failure_rate
        cum_time = cum_time + temp
        integer_failure_time = int(cum_time)
        failure_times.append(cum_time)
        if(cum_time <= 12):
            failure_array[integer_failure_time] = failure_array[integer_failure_time] + 1

    return failure_array

def OREDA_gamma_fit_parameters(x1, p1, x2, p2):

    # Standardize so that x1 < x2 and p1 < p2
    if p1 > p2:
        (p1, p2) = (p2, p1)
        (x1, x2) = (x2, x1)

    # function to find roots of for gamma distribution parameters
    def objective(alpha):
        return ss.gamma.ppf(p2, alpha) / ss.gamma.ppf(p1, alpha) - x2/x1

    # The objective function we're wanting to find a root of is decreasing.
    # We need to find an interval over which is goes from positive to negative.
    left = right = 1.0
    while objective(left) < 0.0:
        left /= 2
    while objective(right) > 0.0:
        right *= 2
    alpha = optimize.bisect(objective, left, right)
    beta = x1 / ss.gamma.ppf(p1, alpha)

    return (alpha, beta)

def main():

    optimize_omega = 0
    constant_omega = 0.9
    num_samples = 5000

    N_failure_data = np.loadtxt(r"C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python Scripts\PART_A_Data_Files\N_failure_data.txt", delimiter="\t")
    M_failure_data = np.loadtxt(r"C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python Scripts\PART_A_Data_Files\M_failure_data.TXT", delimiter="\t")
    O1_failure_data = np.loadtxt(r"C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python Scripts\PART_A_Data_Files\O1_failure_data.TXT", delimiter="\t")
    O3_failure_data = np.loadtxt(r"C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python Scripts\PART_A_Data_Files\O3_failure_data.TXT", delimiter="\t")
    S_failure_data = np.loadtxt(r"C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python Scripts\PART_A_Data_Files\S_failure_data.TXT", delimiter="\t")
    step_failure_data = np.loadtxt(r"C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python Scripts\PART_A_Data_Files\step_data.TXT", delimiter="\t")
```

```
L_static_data = np.loadtxt(r"C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python Scripts\PART_A_Data_Files\L_static_data.TXT", delimiter="\t")
A1_static_data = np.loadtxt(r"C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python Scripts\PART_A_Data_Files\A1_static_data.TXT", delimiter="\t")
A2_static_data = np.loadtxt(r"C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python Scripts\PART_A_Data_Files\A2_static_data.TXT", delimiter="\t")
V_static_data = np.loadtxt(r"C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python Scripts\PART_A_Data_Files\V_static_data.TXT", delimiter="\t")

L_static_data_10x = np.loadtxt(r"C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python Scripts\PART_A_Data_Files\L_static_data10x.TXT", delimiter="\t")
A1_static_data_10x = np.loadtxt(r"C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python Scripts\PART_A_Data_Files\A1_static_data10x.TXT", delimiter="\t")
A2_static_data_10x = np.loadtxt(r"C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python Scripts\PART_A_Data_Files\A2_static_data10x.TXT", delimiter="\t")
V_static_data_10x = np.loadtxt(r"C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python Scripts\PART_A_Data_Files\V_static_data10x.TXT", delimiter="\t")

L_static_data_100x = np.loadtxt(r"C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python Scripts\PART_A_Data_Files\L_static_data100x.TXT", delimiter="\t")
A1_static_data_100x = np.loadtxt(r"C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python Scripts\PART_A_Data_Files\A1_static_data100x.TXT", delimiter="\t")
A2_static_data_100x = np.loadtxt(r"C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python Scripts\PART_A_Data_Files\A2_static_data100x.TXT", delimiter="\t")
V_static_data_100x = np.loadtxt(r"C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python Scripts\PART_A_Data_Files\V_static_data100x.TXT", delimiter="\t")

M_static_data = np.loadtxt(r"C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python Scripts\PART_A_Data_Files\M_static_data.TXT", delimiter="\t")
O1_static_data = np.loadtxt(r"C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python Scripts\PART_A_Data_Files\O1_static_data.TXT", delimiter="\t")
O3_static_data = np.loadtxt(r"C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python Scripts\PART_A_Data_Files\O3_static_data.TXT", delimiter="\t")

#L_static_data = L_static_data_10x
#A1_static_data = L_static_data_10x
#A2_static_data = L_static_data_10x
#V_static_data = L_static_data_10x

N_a_prior = 2
N_b_prior = 1
M_a_prior = 2
M_b_prior = 1
O1_a_prior = 2
O1_b_prior = 1
O3_a_prior = 2
O3_b_prior = 1
S_a_prior = 1
S_b_prior = 1

#step_a_prior = 1
#step_b_prior = 1

N_PEWMA = PEWMA(N_failure_data,N_a_prior,N_b_prior,optimize_omega,constant_omega)
M_PEWMA = PEWMA(M_failure_data,M_a_prior,M_b_prior,optimize_omega,constant_omega)
O1_PEWMA = PEWMA(O1_failure_data,O1_a_prior,O1_b_prior,optimize_omega,constant_omega)
O3_PEWMA = PEWMA(O3_failure_data,O3_a_prior,O3_b_prior,optimize_omega,constant_omega)
S_PEWMA = PEWMA(S_failure_data,S_a_prior,S_b_prior,optimize_omega,constant_omega)
#step_PEWMA = PEWMA(step_failure_data,step_a_prior,step_b_prior,optimize_omega,constant_omega)


#plot_PEWMA_mean_vs_time(N_PEWMA, N_failure_data, [1987,2005], [0,3.5], 1, r'C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python
Scripts\PART_A_FIGURES\N_mean_and_percentiles.png', "N - Mean and 95th Percentiles")
#plot_PEWMA_mean_vs_time(M_PEWMA, M_failure_data, [1987,2005], [0,5], 2, r'C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python
Scripts\PART_A_FIGURES\M_mean_and_percentiles.png', "M, - Mean and 95th Percentiles")
#plot_PEWMA_mean_vs_time(O1_PEWMA, O1_failure_data, [1987,2005], [0,5], 3, r'C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python
Scripts\PART_A_FIGURES\O1_mean_and_percentiles.png', "O1 - Mean and 95th Percentiles")
#plot_PEWMA_mean_vs_time(O3_PEWMA, O3_failure_data, [1987,2005], [0,4.5], 4, r'C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python
Scripts\PART_A_FIGURES\O3_mean_and_percentiles.png', "O3 - Mean and 95th Percentiles")
#plot_PEWMA_mean_vs_time(S_PEWMA, S_failure_data, [1987,2005], [0,1.5], 5, r'C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python
Scripts\PART_A_FIGURES\S_mean_and_percentiles.png', "S - Mean and 95th Percentiles")
#plot_PEWMA_mean_vs_time(_PEWMA, step_failure_data, [1,26], [0,10], 999, r'C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python
Scripts\PART_A_FIGURES\step_mean_and_percentiles.png', "Omega=" + str(constant_omega))
```

```
    #plot_gamma_updating(N_PEWMA, N_a_prior, N_b_prior, "upper right", 2.0, 1, r'C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python
Scripts\PART_A_FIGURES\N_gamma_updating.png', "N - Posterior Distributions")
    #plot_gamma_updating(M_PEWMA, M_a_prior, M_b_prior, "upper right", 5, 2, r'C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python Scripts\PART_A_FIGURES\M_gamma_updating.png',
"M - Posterior Distributions")
    #plot_gamma_updating(O1_PEWMA, O1_a_prior, O1_b_prior, "upper right", 2.5, 3, r'C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python
Scripts\PART_A_FIGURES\O1_gamma_updating.png', "O1 - Posterior Distributions")
    #plot_gamma_updating(O3_PEWMA, O3_a_prior, O3_b_prior, "upper right", 4.5, 4, r'C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python
Scripts\PART_A_FIGURES\O3_gamma_updating.png', "O3 - Posterior Distributions")
    #plot_gamma_updating(S_PEWMA, S_a_prior, M_b_prior, "upper right", 1.5, 5, r'C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python Scripts\PART_A_FIGURES\S_gamma_updating.png',
"S - Posterior Distributions")

    #omega_values = [0.5,0.6,0.7,0.8,0.9,1]
    #plot_PEWMA_mean_vs_omega(N_failure_data, N_a_prior, N_b_prior, omega_values, "upper right", 3.5, 11, r'C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python
Scripts\PART_A_FIGURES\N_omega_value_sensitivity.png', "N, Omega Sensitivities")
    #plot_PEWMA_mean_vs_omega(M_failure_data, M_a_prior, M_b_prior, omega_values, "upper left", 5, 12, r'C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python
Scripts\PART_A_FIGURES\M_omega_value_sensitivity.png', "M, Omega Sensitivities")
    #plot_PEWMA_mean_vs_omega(O1_failure_data, O1_a_prior, O1_b_prior, omega_values, "upper left", 2.5, 13, r'C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python
Scripts\PART_A_FIGURES\O1_omega_value_sensitivity.png', "O1 Omega Sensitivities")
    #plot_PEWMA_mean_vs_omega(O3_failure_data, O3_a_prior, O3_b_prior, omega_values, "upper left", 4.5, 14, r'C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python
Scripts\PART_A_FIGURES\O3_omega_value_sensitivity.png', "O3 Omega Sensitivities")
    #plot_PEWMA_mean_vs_omega(S_failure_data, S_a_prior, S_b_prior, omega_values, "upper left", 1.5, 15, r'C:\Users\Christian\Google Drive\Masters\MASTERS_2015\01 - Python
Scripts\PART_A_FIGURES\S_omega_value_sensitivity.png', "S Omega Sensitivities")


    M_COMBINED = np.row_stack((M_static_data,M_PEWMA))
    O1_COMBINED = np.row_stack((O1_static_data,O1_PEWMA))
    O3_COMBINED = np.row_stack((O3_static_data,O3_PEWMA))
    N_COMBINED = N_PEWMA
    S_COMBINED = S_PEWMA
    L_COMBINED = L_static_data
    A1_COMBINED = A1_static_data
    A2_COMBINED = A2_static_data
    V_COMBINED = V_static_data

    p_N = np.zeros(num_samples)
    p_M = np.zeros(num_samples)
    p_O1 = np.zeros(num_samples)
    p_O3 = np.zeros(num_samples)
    p_S = np.zeros(num_samples)
    p_L = np.zeros(num_samples)
    p_A1 = np.zeros(num_samples)
    p_A2 = np.zeros(num_samples)
    p_V = np.zeros(num_samples)
    p_SA = np.zeros(num_samples)

    p_LS = np.zeros(num_samples)
    p_A = np.zeros(num_samples)
    p_O2 = np.zeros(num_samples)
    p_O = np.zeros(num_samples)
    p_R = np.zeros(num_samples)
    p_B = np.zeros(num_samples)
    p_VC = np.zeros(num_samples)

    VC_95_percentile = np.zeros(len(N_COMBINED))
    VC_5_percentile  = np.zeros(len(N_COMBINED))
    VC_mean = np.zeros(len(N_COMBINED))
    VC_mode = np.zeros(len(N_COMBINED))
```

```
p_VC_mean = np.zeros(num_samples)

plt.figure(97,figsize=(10,6),dpi=600)
plt.figure(98,figsize=(10,6),dpi=600)
plt.figure(99,figsize=(10,6),dpi=600)

#plt.figure(201,figsize=(11,4.6),dpi=600)
plt.figure(202,figsize=(10,6),dpi=600)
plt.figure(203,figsize=(10,6),dpi=600)

iterations = np.arange(0,num_samples,1)

cutset_1 = zeros(num_samples)
cutset_2 = zeros(num_samples)
cutset_3 = zeros(num_samples)
cutset_4 = zeros(num_samples)
cutset_5 = zeros(num_samples)
cutset_6 = zeros(num_samples)
cutset_7 = zeros(num_samples)
cutset_8 = zeros(num_samples)

cutset_1_mean = zeros(len(N_COMBINED))
cutset_2_mean = zeros(len(N_COMBINED))
cutset_3_mean = zeros(len(N_COMBINED))
cutset_4_mean = zeros(len(N_COMBINED))
cutset_5_mean = zeros(len(N_COMBINED))
cutset_6_mean = zeros(len(N_COMBINED))
cutset_7_mean = zeros(len(N_COMBINED))
cutset_8_mean = zeros(len(N_COMBINED))


for t in range(0,len(N_COMBINED)):

    N_samples = generate_samples(N_COMBINED[t,:],num_samples)
    M_samples = generate_samples(M_COMBINED[t,:],num_samples)
    O1_samples = generate_samples(O1_COMBINED[t,:],num_samples)
    O3_samples = generate_samples(O3_COMBINED[t,:],num_samples)
    S_samples = generate_samples(S_COMBINED[t,:],num_samples)
    V_samples = generate_samples(V_COMBINED[t,:],num_samples)
    A1_samples = generate_samples(A1_COMBINED[t,:],num_samples)
    A2_samples = generate_samples(A2_COMBINED[t,:],num_samples)
    SA_samples = generate_samples(A2_COMBINED[t,:],num_samples)
    L_samples = generate_samples(L_COMBINED[t,:],num_samples)

    temp_sum = 0

    for i in range(0,num_samples):

        p_N[i] = 1-np.exp(-N_samples[i])
        p_M[i] = 1-np.exp(-M_samples[i])
        p_O1[i] = 1-np.exp(-O1_samples[i])
        p_O3[i] = 1-np.exp(-O3_samples[i])
        p_S[i] = 1-np.exp(-S_samples[i])
        p_V[i] = 1-np.exp(-V_samples[i])
        p_A1[i] = 1-np.exp(-A1_samples[i])
        p_A2[i] = 1-np.exp(-A2_samples[i])
        p_L[i] = 1-np.exp(-L_samples[i])
        p_SA[i] = 1-np.exp(-SA_samples[i])
```

```
    p_LS[i] = p_L[i] + p_A1[i] - p_L[i]*p_A1[i]
    p_A[i] = p_LS[i]*p_A2[i]
    p_O2[i] = p_A[i]*p_M[i]
    p_O[i] = p_O2[i] + p_O3[i] - p_O2[i]*p_O3[i]
    p_R[i] = p_O1[i] + p_V[i] - p_O1[i]*p_V[i]
    p_B[i] = p_R[i]*p_O[i] #*p_SA[i]

    cutset_1[i] = p_O1[i]*p_L[i]*p_A2[i]*p_M[i]*p_SA[i]
    cutset_2[i] = p_O1[i]*p_A1[i]*p_A2[i]*p_M[i]*p_SA[i]
    cutset_3[i] = p_L[i]*p_A2[i]*p_M[i]*p_V[i]*p_SA[i]
    cutset_4[i] = p_A1[i]*p_A2[i]*p_M[i]*p_V[i]*p_SA[i]
    cutset_5[i] = p_O1[i]*p_O3[i]*p_SA[i]
    cutset_6[i] = p_V[i]*p_O3[i]*p_SA[i]
    cutset_7[i] = p_N[i]
    cutset_8[i] = p_S[i]

    p_VC[i] = p_B[i] + p_N[i] + p_S[i] - p_B[i]*p_N[i] - p_B[i]*p_S[i] - p_N[i]*p_S[i] + p_B[i]*p_N[i]*p_S[i]

    temp_sum = temp_sum + p_VC[i]
    p_VC_mean[i] = temp_sum/i

if(t == len(N_COMBINED)-1):
    plt.figure(9999,figsize=(10,6),dpi=600)
    plt.hist(p_VC, color="gray", bins=20)


if(t == len(N_COMBINED)-1):
    plt.figure(97)
    plt.hist(p_VC, bins = 50, histtype='step', normed=True,linewidth=3.5,color='black', cumulative=False)
else:
    plt.figure(97)
    plt.hist(p_VC, bins = 50, histtype='step', normed=True,linewidth=1.5,color=str(0.95 - t/20), cumulative=False)

VC_mean[t] = statistics.mean(p_VC)
VC_95_percentile[t] = np.percentile(p_VC,95)
VC_5_percentile[t] = np.percentile(p_VC,5)

plt.figure(98)
plt.plot(iterations,p_VC_mean,color=str(0.95 - t/20),label="Time = " + str(t) + " years")
plt.ylim((0.4,1))
plt.legend(fontsize=10)

cutset_1_mean[t] = statistics.mean(cutset_1)
cutset_2_mean[t] = statistics.mean(cutset_2)
cutset_3_mean[t] = statistics.mean(cutset_3)
cutset_4_mean[t] = statistics.mean(cutset_4)
cutset_5_mean[t] = statistics.mean(cutset_5)
cutset_6_mean[t] = statistics.mean(cutset_6)
cutset_7_mean[t] = statistics.mean(cutset_7)
cutset_8_mean[t] = statistics.mean(cutset_8)
```

Appendix B – Python code for Bayesian Updating of Material Balance Equation

B.1 Functions

```
def interpolate_PVT(PVT_data,p,index):

    return_value = 0
    slope = 0

    if(p < PVT_data[0,0]): #interpolate

        slope = (PVT_data[1,index] - PVT_data[0,index])/(PVT_data[1,0] - PVT_data[0,0])
        return_value = PVT_data[0,index] + (p-PVT_data[0,0]*slope)

    elif(p > PVT_data[len(PVT_data)-1,0]):  #extrapolate

        slope = (PVT_data[len(PVT_data)-1,index] - PVT_data[len(PVT_data)-2,index])/(PVT_data[len(PVT_data)-1,0] - PVT_data[len(PVT_data)-2,0])
        return_value = PVT_data[len(PVT_data)-1,index] + (p-PVT_data[len(PVT_data)-1,0]*slope)

    else:

        for i in range(0,len(PVT_data)-1):

            if(p >= PVT_data[i,0] and p <= PVT_data[i+1,0]):

                slope = (PVT_data[i+1,index] - PVT_data[i,index])/(PVT_data[i+1,0] - PVT_data[i,0])
                return_value = PVT_data[i,index] + (p-PVT_data[i,0])*slope

    return return_value

def MBAL_objective_function(pres_iteration, initial_pore_volume, initial_pressure, initial_water, MBAL_variables_prev_time_step, MBAL_production_data, PVT_data):

    pres = MBAL_variables_prev_time_step[1]
    N = MBAL_variables_prev_time_step[2]
    G = MBAL_variables_prev_time_step[3]
    W_res = MBAL_variables_prev_time_step[4]
    W_Aq = MBAL_variables_prev_time_step[6]
    pa = MBAL_variables_prev_time_step[7]
    Jw = MBAL_variables_prev_time_step[8]

    delta_t = MBAL_production_data[0]
    Np = MBAL_production_data[1]
    Gp = MBAL_production_data[2]
    Wp = MBAL_production_data[3]

    Bo2 = interpolate_PVT(PVT_data,pres_iteration,1)
    Bg2 = interpolate_PVT(PVT_data,pres_iteration,2)
    Bw2 = interpolate_PVT(PVT_data,pres_iteration,3)
    Rs2 = interpolate_PVT(PVT_data,pres_iteration,4)
    Rs1 = interpolate_PVT(PVT_data,pres,4)

    cf = 5E-5
    cw = 2.77182E-05
    ct = cf + cw
```

```
    aquifer_pressure = pres + (pa - pres)*exp(-Jw*(delta_t)/(ct*W_Aq))
    aquifer_influx = ct*W_Aq*(pa - pres)*(1-np.exp(-Jw*(delta_t)/(ct*W_Aq)))
    We = aquifer_influx
    delta_PV = initial_pore_volume*cf*(initial_pressure - pres_iteration)

    N_updated = N*Bo2 - Np*Bo2
    G_updated = G*Bg2 - Gp*Bg2 + N*Rs1*Bg2 - (N-Np)*Rs2*Bg2
    W_Aq_updated = W_Aq - aquifer_influx
    W_res_updated = W_res*Bw2 - Wp*Bw2 + We
    Vp_updated = initial_pore_volume - delta_PV

    if(N_updated < 0):
        N_updated = 0
    if(G_updated < 0):
        G_updated = 0
    if(W_res_updated < initial_water*Bw2):
        W_res_updated = initial_water*Bw2

    f = N_updated + G_updated + W_res_updated + delta_PV - initial_pore_volume

    return_variable = [f, pres_iteration, N_updated/Bo2, G_updated/Bg2, W_res_updated/Bw2, Vp_updated, W_Aq_updated, aquifer_pressure, Jw]

    return return_variable

def MBAL_Newton(MBAL_variables_prev_time_step, p_guess, initial_pore_volume, initial_pressure, initial_water, production_data, PVT_data):

    flag = True
    x_old = p_guess
    return_value = 0
    derivative_delta_p = 0.01
    tolerance = 0.0001
    max_count = 5
    temp_count = 0

    while (flag == True):

        p_delta_1 = x_old - derivative_delta_p
        p_delta_2 = x_old + derivative_delta_p

        f_delta_1 = MBAL_objective_function(p_delta_1, initial_pore_volume, initial_pressure, initial_water, MBAL_variables_prev_time_step, production_data, PVT_data)
        f_delta_2 = MBAL_objective_function(p_delta_2, initial_pore_volume, initial_pressure, initial_water, MBAL_variables_prev_time_step, production_data, PVT_data)
        f_x_old = MBAL_objective_function(x_old, initial_pore_volume, initial_pressure, initial_water, MBAL_variables_prev_time_step, production_data, PVT_data)

        derivative = ((f_x_old[0] - f_delta_1[0])/derivative_delta_p + (f_delta_2[0] - f_x_old[0])/derivative_delta_p)/2

        x_new = x_old - f_x_old[0]/derivative

        if(x_new < 1):
            x_new = 1
        elif(x_new > 1000):
            x_new = 1000

        if(np.abs((x_new-x_old)/x_old) < tolerance):

            return_value = f_x_old
            flag = False
        else:
            x_old = x_new
```

```
            temp_count = temp_count + 1

        if(temp_count > max_count):
            flag = False
            return_value = f_x_old

    return return_value

def MBAL_inverse(model_variables, production_data, PVT_data, end_time_index, Swc, return_all_data_flag):

    MBAL_variables = np.zeros((int(end_time_index/10 + 1),11))
    MBAL_production_data = np.zeros(4)

    initial_oil_volume = model_variables[0]
    initial_aquifer_volume = model_variables[1]
    initial_aquifer_index = model_variables[2]
    initial_gas_volume = model_variables[3]
    initial_reservoir_pressure = production_data[0,1]
    initial_water_volume = ((initial_oil_volume*interpolate_PVT(PVT_data, initial_reservoir_pressure, 1)/(1-Swc))*Swc)/interpolate_PVT(PVT_data, initial_reservoir_pressure, 3)
    initial_aquifer_pressure = initial_reservoir_pressure
    initial_pore_volume = initial_oil_volume*interpolate_PVT(PVT_data, initial_reservoir_pressure, 1)/(1-Swc)

    MBAL_variables[0][0] = 0
    MBAL_variables[0][1] = initial_reservoir_pressure
    MBAL_variables[0][2] = initial_oil_volume
    MBAL_variables[0][3] = initial_gas_volume
    MBAL_variables[0][4] = initial_water_volume
    MBAL_variables[0][5] = initial_pore_volume
    MBAL_variables[0][6] = initial_aquifer_volume
    MBAL_variables[0][7] = initial_aquifer_pressure
    MBAL_variables[0][8] = initial_aquifer_index
    MBAL_variables[0][9] = 0 #time
    MBAL_variables[0][10] = initial_reservoir_pressure

    p_guess = 300

    for t in range(1,int(end_time_index/10)+1):

        MBAL_production_data[0] = production_data[t,0] - production_data[t-1,0]
        MBAL_production_data[1] = (production_data[t,2] - production_data[t-1,2])
        MBAL_production_data[2] = (production_data[t,3] - production_data[t-1,3])
        MBAL_production_data[3] = production_data[t,4] - production_data[t-1,4]

        MBAL_output = MBAL_Newton(MBAL_variables[t-1,:], p_guess, initial_pore_volume, initial_reservoir_pressure, initial_water_volume, MBAL_production_data, PVT_data)

        if(isnan(MBAL_output[1]) == True):
            p_guess = 200
        elif(MBAL_output[1] < 0):
            p_guess = 200
        else:
            p_guess = MBAL_output[1]

        MBAL_variables[t][0] = MBAL_output[0]
        MBAL_variables[t][1] = MBAL_output[1]
        MBAL_variables[t][2] = MBAL_output[2]
        MBAL_variables[t][3] = MBAL_output[3]
        MBAL_variables[t][4] = MBAL_output[4]
        MBAL_variables[t][5] = MBAL_output[5]
```

```
        MBAL_variables[t][6] = MBAL_output[6]
        MBAL_variables[t][7] = MBAL_output[7]
        MBAL_variables[t][8] = MBAL_output[8]
        MBAL_variables[t][9] = production_data[t,0]
        MBAL_variables[t][10] = production_data[t,1]


    if(return_all_data_flag == False):
        return_variable = MBAL_variables[0,:]
        temp_count = 0
        for i in range(0,len(MBAL_variables)):
            if(temp_count == 30):
                return_variable = np.vstack((return_variable, MBAL_variables[i,:]))
                temp_count = 0
            temp_count = temp_count + 1
    else:
        return_variable = MBAL_variables

    return return_variable

def likelihood_function(production_data, PVT_data, model_variables, error, big_end_time_index, production_data_end_time, Swc):

    #Gaussian Likelihood
    n = big_end_time_index
    temp_exp_sum = 0
    const = 1/(pow(2*np.pi,n/2)*pow(error,n/2))
    MBAL_output = MBAL_inverse(model_variables, production_data, PVT_data, production_data_end_time, Swc, False)

    for t in range(1, len(MBAL_output)):

        if(isnan(MBAL_output[t,1]) == True):
            temp_exp_sum = temp_exp_sum
        elif(MBAL_output[t,1] < 0):
            temp_exp_sum = temp_exp_sum
        else:
            temp_exp_sum = temp_exp_sum + pow((MBAL_output[t,10]-MBAL_output[t,1]),2)/error

    likelihood = const*np.exp(-0.5*temp_exp_sum)

    return likelihood


def MCMC_Metropolis(Swc, model_variables, proposal_parameters, production_data, PVT_data, error, t, big_time_step, chain_length, proposal_type=0):

    markov_chain = np.zeros((chain_length,4))
    markov_mean = np.zeros((chain_length,4))
    markov_acceptance = np.zeros((chain_length,1))
    N_temp_sum = 0
    Wi_temp_sum = 0
    Jw_temp_sum = 0
    G_temp_sum = 0
    acceptance_sum = 0
    acceptance_ratio = 0

    #Proposal parameters
    N_proposal_mean = proposal_parameters[0]
    N_proposal_std = proposal_parameters[1]
    Wi_proposal_mean = proposal_parameters[2]
```

```
Wi_proposal_std = proposal_parameters[3]
Jw_proposal_mean = proposal_parameters[4]
Jw_proposal_std = proposal_parameters[5]
G_proposal_mean = proposal_parameters[6]
G_proposal_std = proposal_parameters[7]

if(model_variables[0][0] == 0):
    N = model_variables[0][1]
    N_prior_probability = 1
elif(model_variables[0][0] == 1):
    N_prior_mean = model_variables[0][1]
    N_prior_std = model_variables[0][2]
    N = N_proposal_mean
    if(N < 0):
        N = N_prior_mean
    N_prior_probability = ss.norm.pdf(N, loc=N_prior_mean, scale=N_prior_std)

if(model_variables[1][0] == 0):
    Wi = model_variables[1][1]
    Wi_prior_probability = 1
elif(model_variables[1][0] == 1):
    Wi_prior_mean = model_variables[1][1]
    Wi_prior_std = model_variables[1][2]
    Wi = Wi_proposal_mean
    if(Wi < 0):
        Wi = Wi_prior_mean
    Wi_prior_probability = ss.norm.pdf(Wi, loc=Wi_prior_mean, scale=Wi_prior_std)

if(model_variables[2][0] == 0):
    Jw = model_variables[2][1]
    Jw_prior_probability = 1
elif(model_variables[2][0] == 1):
    Jw_prior_mean = model_variables[2][1]
    Jw_prior_std = model_variables[2][2]
    Jw = Jw_proposal_mean
    if(Jw < 0):
        Jw = Jw_prior_mean
    Jw_prior_probability = ss.norm.pdf(Jw, loc=Jw_prior_mean, scale=Jw_prior_std)

if(model_variables[3][0] == 0):
    G = model_variables[3][1]
    G_prior_probability = 1
elif(model_variables[3][0] == 1):
    G_prior_mean = model_variables[3][1]
    G_prior_std = model_variables[3][2]
    G = G_proposal_mean
    if(G < 0):
        G = G_prior_mean
    G_prior_probability = ss.norm.pdf(G, loc=G_prior_mean, scale=G_prior_std)

prior = N_prior_probability*Wi_prior_probability*Jw_prior_probability*G_prior_probability
likelihood = likelihood_function(production_data, PVT_data, [N*1000000, Wi*1000000, Jw, G*1000000], error, t, big_time_step, Swc)

if(isnan(likelihood) == True):
    likelihood = 0

p_x_old = likelihood*prior
x_old = [N, Wi, Jw, G]
```

```
markov_chain[0,0] = x_old[0]
markov_chain[0,1] = x_old[1]
markov_chain[0,2] = x_old[2]
markov_chain[0,3] = x_old[3]

N_temp_sum = x_old[0]
Wi_temp_sum = x_old[1]
Jw_temp_sum = x_old[2]
G_temp_sum = x_old[3]

markov_mean[0,0] = N_temp_sum
markov_mean[0,1] = Wi_temp_sum
markov_mean[0,2] = Jw_temp_sum
markov_mean[0,3] = G_temp_sum

acceptance_sum = 1
markov_acceptance[0] = acceptance_sum

for i in range(1,chain_length):

    if(model_variables[0][0] == 0):
        N = model_variables[0][1]
        N_prior_probability = 1
    elif(model_variables[0][0] == 1):
        N_prior_mean = model_variables[0][1]
        N_prior_std = model_variables[0][2]
        N = np.random.normal(loc=markov_chain[i-1,0], scale=N_proposal_std)
        N_prior_probability = ss.norm.pdf(N, loc=N_prior_mean, scale=N_prior_std)

    if(model_variables[1][0] == 0):
        Wi = model_variables[1][1]
        Wi_prior_probability = 1
    elif(model_variables[1][0] == 1):
        Wi_prior_mean = model_variables[1][1]
        Wi_prior_std = model_variables[1][2]
        Wi = np.random.normal(loc=markov_chain[i-1,1], scale=Wi_proposal_std)
        Wi_prior_probability = ss.norm.pdf(Wi, loc=Wi_prior_mean, scale=Wi_prior_std)

    if(model_variables[2][0] == 0):
        Jw = model_variables[2][1]
        Jw_prior_probability = 1
    elif(model_variables[2][0] == 1):
        Jw_prior_mean = model_variables[2][1]
        Jw_prior_std = model_variables[2][2]
        Jw = np.random.normal(loc=markov_chain[i-1,2], scale=Jw_proposal_std)
        Jw_prior_probability = ss.norm.pdf(Jw, loc=Jw_prior_mean, scale=Jw_prior_std)

    if(model_variables[3][0] == 0):
        G = model_variables[3][1]
        G_prior_probability = 1
    elif(model_variables[3][0] == 1):
        G_prior_mean = model_variables[3][1]
        G_prior_std = model_variables[3][2]
        G = np.random.normal(loc=markov_chain[i-1,3], scale=G_proposal_std)
        G_prior_probability = ss.norm.pdf(G, loc=G_prior_mean, scale=G_prior_std)

    x_j = [N, Wi, Jw, G]
```

```
        prior = N_prior_probability*Wi_prior_probability*Jw_prior_probability*G_prior_probability
        likelihood = likelihood_function(production_data, PVT_data, [N*1000000, Wi*1000000, Jw, G*1000000], error, t, big_time_step, Swc)

        if(isnan(likelihood) == True):
            likelihood = 0

        p_x_j = prior*likelihood
        if(p_x_old > 0):
            alpha = np.min((1,p_x_j/p_x_old))
        elif(p_x_old == 0):
            alpha = 0
        elif(isnan(p_x_old) == True):
            alpha = 0
        random_number = np.random.uniform(0,1)

        if(alpha >= random_number):
            x_new = x_j
            p_x_new = p_x_j
            acceptance_sum = acceptance_sum + 1
            acceptance_ratio = acceptance_sum/i
        else:
            x_new = x_old
            p_x_new = p_x_old
            acceptance_ratio = acceptance_sum/i

        markov_chain[i,0] = x_new[0]
        markov_chain[i,1] = x_new[1]
        markov_chain[i,2] = x_new[2]
        markov_chain[i,3] = x_new[3]

        N_temp_sum = N_temp_sum + x_old[0]
        Wi_temp_sum = Wi_temp_sum + x_old[1]
        Jw_temp_sum = Jw_temp_sum + x_old[2]
        G_temp_sum = G_temp_sum + x_old[3]

        markov_mean[i,0] = N_temp_sum/i
        markov_mean[i,1] = Wi_temp_sum/i
        markov_mean[i,2] = Jw_temp_sum/i
        markov_mean[i,3] = G_temp_sum/i

        markov_acceptance[i,0] = acceptance_ratio

        x_old = x_new
        p_x_old = p_x_new

    return_variable = np.hstack((markov_chain, markov_mean, markov_acceptance))

    return return_variable

def update_proposal_parameters(markov_output_incremental, spread_factor):

    N_mean, N_std = norm.fit(markov_output_incremental[:,0])
    Wi_mean, Wi_std = norm.fit(markov_output_incremental[:,1])
    Jw_mean, Jw_std = norm.fit(markov_output_incremental[:,2])
    G_mean, G_std = norm.fit(markov_output_incremental[:,3])

    N_std = N_std*spread_factor
    Wi_std = Wi_std*spread_factor
```

```
Jw_std = Jw_std*spread_factor
G_std = G_std*spread_factor

if(N_std < 0.5):
    N_std = 1
if(Wi_std < 0.5):
    Wi_std = 1
if(Jw_std < 0.5):
    Jw_std = 1
G_std = 1

proposal_parameters = [N_mean, N_std, Wi_mean, Wi_std, Jw_mean, Jw_std, G_mean, G_std]

return proposal_parameters
```

B.2 Main Python Routine for Grid Based Solution

```
def main():

    home = True

    uni_string = "C:/Users/alfchris/Google Drive/Masters/MASTERS_2015/01 - Python Scripts/"
    home_string = "C:/Users/Christian/Google Drive/Masters/MASTERS_2015/01 - Python Scripts/"

    noise = [10,20,50]

    if(home==True):
        file_dir_string = home_string
    else:
        file_dir_string = uni_string

    #Loading files
    PVT_data = np.loadtxt(file_dir_string + "PART_B_Data_Files/PVT_data.txt", delimiter="\t")
    production_data = np.loadtxt(file_dir_string + "PART_B_Data_Files/production_data.txt", delimiter="\t")
    end_time_index = len(production_data)
    big_time_steps = np.arange(0,4200,300)
    Swc = 0.2

    for k in range(0,3):

        if(home==True):
            file_dir_string = home_string
        else:
            file_dir_string = uni_string

        #Loading files
        PVT_data = np.loadtxt(file_dir_string + "PART_B_Data_Files/PVT_data.txt", delimiter="\t")
        production_data = np.loadtxt(file_dir_string + "PART_B_Data_Files/production_data.txt", delimiter="\t")

        Swc = 0.2
        #error = 20
        error = noise[k]
        end_time_index = len(production_data)
        big_time_steps = np.arange(0,4200,300)
        Wi = 444600000
        G = 0
        rho = 0
        grid_points = [100,100]
        N_bounds = [0,2,100]
        Jw_bounds = [0,25,250]
        deltas = [((N_bounds[2]-N_bounds[0])/(grid_points[0]-1)),((Jw_bounds[2]-Jw_bounds[0])/(grid_points[1]-1))]

        N_grid = np.arange(N_bounds[0],N_bounds[2],deltas[0])
        Jw_grid = np.arange(Jw_bounds[0],Jw_bounds[2],deltas[1])

        prior = np.zeros((len(N_grid),len(Jw_grid),13))
        likelihood = np.zeros((len(N_grid),len(Jw_grid), 13))
        posterior = np.zeros((len(N_grid),len(Jw_grid),13))

        likelihood_sum = 0
        posterior_sum = 0
        prior_sum = 0
```

```
N_prior = 50
Jw_prior = 150

N_prior_std = 100
Jw_prior_std = 1000

const = pow((1/sqrt(2*np.pi)),2)
C_prior = ([[N_prior_std,rho],[rho,Jw_prior_std]])
const_prior = const/sqrt(det(C_prior))

time = np.zeros(13)

N_grid_mesh,Jw_grid_mesh = np.meshgrid(N_grid,Jw_grid)

N_marginal_posterior = np.zeros((len(N_grid), len(time)))
N_marginal_likelihood = np.zeros((len(N_grid), len(time)))
N_marginal_prior = np.zeros((len(N_grid), len(time)))
Jw_marginal_posterior = np.zeros((len(Jw_grid), len(time)))
Jw_marginal_likelihood = np.zeros((len(Jw_grid), len(time)))
Jw_marginal_prior = np.zeros((len(Jw_grid), len(time)))

time_index = [1,13]

for t in range(time_index[0],time_index[1]):

    time[t] = t
    #print(t)

    for i in range(0,len(N_grid)):

        for j in range(0,len(Jw_grid)):

            prior[j,i,t] = const_prior*exp(-0.5*np.dot(np.dot([N_grid[i]-N_prior,Jw_grid[j]-Jw_prior],linalg.inv(C_prior)),[N_grid[i]-N_prior,Jw_grid[j]-Jw_prior]))

            temp = likelihood_function(production_data, PVT_data, [N_grid[i]*1000000, Wi, Jw_grid[j], G], error, t, big_time_steps[t], Swc)
            if(np.isnan(temp) == True):
                likelihood[j,i,t] = 0
            else:
                likelihood[j,i,t] = temp

            posterior[j,i,t] = prior[j,i,t]*likelihood[j,i,t]

            N_marginal_posterior[i,t] = N_marginal_posterior[i,t] + posterior[j,i,t]
            N_marginal_likelihood[i,t] = N_marginal_likelihood[i,t] + likelihood[j,i,t]
            N_marginal_prior[i,t] = N_marginal_prior[i,t] + prior[j,i,t]

            Jw_marginal_posterior[j,t] = Jw_marginal_posterior[j,t] + posterior[j,i,t]
            Jw_marginal_likelihood[j,t] = Jw_marginal_likelihood[j,t] + likelihood[j,i,t]
            Jw_marginal_prior[j,t] = Jw_marginal_prior[j,t] + prior[j,i,t]

            if(likelihood[j,i,t] >= 0):
                likelihood_sum = likelihood_sum + likelihood[j,i,t]

            if(posterior[j,i,t] >= 0):
                posterior_sum = posterior_sum + posterior[j,i,t]

            if(prior[j,i,t] >= 0):
```

```
            prior_sum = prior_sum + prior[j,i,t]

likelihood_normalized = likelihood/likelihood_sum
prior_normalized = prior/prior_sum
posterior_normalized = posterior/posterior_sum

print(k)

#Saving calculations to file:
np.save(file_dir_string + "GRID_BASED_2_VAR_OUTPUTS/" + str(noise[k]) + "_likelihood.npy", likelihood_normalized)
np.save(file_dir_string + "GRID_BASED_2_VAR_OUTPUTS/" + str(noise[k]) + "_prior.npy", prior_normalized)
np.save(file_dir_string + "GRID_BASED_2_VAR_OUTPUTS/" + str(noise[k]) + "_posterior.npy", posterior_normalized)

np.save(file_dir_string + "GRID_BASED_2_VAR_OUTPUTS/" + str(noise[k]) + "_N_marginal_posterior.npy", N_marginal_posterior)
np.save(file_dir_string + "GRID_BASED_2_VAR_OUTPUTS/" + str(noise[k]) + "_N_marginal_prior.npy", N_marginal_prior)
np.save(file_dir_string + "GRID_BASED_2_VAR_OUTPUTS/" + str(noise[k]) + "_N_marginal_likelihood.npy", N_marginal_likelihood)

np.save(file_dir_string + "GRID_BASED_2_VAR_OUTPUTS/" + str(noise[k]) + "_Jw_marginal_posterior.npy", Jw_marginal_posterior)
np.save(file_dir_string + "GRID_BASED_2_VAR_OUTPUTS/" + str(noise[k]) + "_Jw_marginal_prior.npy", Jw_marginal_prior)
np.save(file_dir_string + "GRID_BASED_2_VAR_OUTPUTS/" + str(noise[k]) + "_Jw_marginal_likelihood.npy", Jw_marginal_likelihood)

np.save(file_dir_string + "GRID_BASED_2_VAR_OUTPUTS/" + str(noise[k]) + "_N_grid.npy", N_grid)
np.save(file_dir_string + "GRID_BASED_2_VAR_OUTPUTS/" + str(noise[k]) + "_Jw_grid.npy", Jw_grid)

np.save(file_dir_string + "GRID_BASED_2_VAR_OUTPUTS/" + str(noise[k]) + "_time_index.npy", time_index)
```

B.3 Main Python Routine for MCMC Based Solution

```
def main():

    home = True

    uni_string = "C:/Users/alfchris/Google Drive/Masters/MASTERS_2015/01 - Python Scripts/"
    home_string = "C:/Users/Christian/Google Drive/Masters/MASTERS_2015/01 - Python Scripts/"

    if(home==True):
        file_dir_string = home_string
    else:
        file_dir_string = uni_string

    #Loading files
    PVT_data = np.loadtxt(file_dir_string + "PART_B_Data_Files/PVT_data.txt", delimiter="\t")
    production_data = np.loadtxt(file_dir_string + "PART_B_Data_Files/production_data.txt", delimiter="\t")

    Swc = 0.2
    error = 10  #this is a variance, not a standard deviation
    chain_length = 10000
    proposal_spread_factor = 1
    num_forward_model_samples = 50

    end_time_index = len(production_data)
    big_time_steps = np.arange(0,4800,300)

    N_initial = [1,50,10]   #50
    Wi_initial = [0,444.6,100]    #600, 100
    Jw_initial = [1,150,31.6228]   #150,31.6228          50,20
    G_initial = [0,0,0.01]

    total_markov_output = zeros((chain_length*end_time_index,4))
    model_variables = [N_initial,Wi_initial,Jw_initial,G_initial]

    N_mean_proposal = N_initial[1]
    N_std_proposal = N_initial[2]*proposal_spread_factor
    Wi_mean_proposal = Wi_initial[1]
    Wi_std_proposal = Wi_initial[2]*proposal_spread_factor
    Jw_mean_proposal = Jw_initial[1]
    Jw_std_proposal = Jw_initial[2]*proposal_spread_factor
    G_mean_proposal = G_initial[1]
    G_std_proposal = G_initial[2]*proposal_spread_factor

    proposal_parameters_incremental = [N_mean_proposal, N_std_proposal, Wi_mean_proposal, Wi_std_proposal, Jw_mean_proposal, Jw_std_proposal, G_mean_proposal, G_std_proposal]
    total_proposal_parameters = proposal_parameters_incremental

    time_index = [1,13]
    temp_count = 0

    for t in range(1,13):

        print(t)

        if(t == 1):
```

```
        markov_output_incremental =  MCMC_Metropolis(Swc, model_variables, proposal_parameters_incremental, production_data, PVT_data, error, t, big_time_steps[t], chain_length, 0)
        total_markov_output = markov_output_incremental
        proposal_parameters_incremental = update_proposal_parameters(markov_output_incremental, proposal_spread_factor)
        total_proposal_parameters = np.vstack((total_proposal_parameters,proposal_parameters_incremental))

    else:

        markov_output_incremental =  MCMC_Metropolis(Swc, model_variables, proposal_parameters_incremental, production_data, PVT_data, error, t, big_time_steps[t], chain_length, 0)
        total_markov_output = np.vstack((total_markov_output,markov_output_incremental))
        proposal_parameters_incremental = update_proposal_parameters(markov_output_incremental, proposal_spread_factor)
        total_proposal_parameters = np.vstack((total_proposal_parameters,proposal_parameters_incremental))


    if (t == 1 or t==7 or t==12):
        N_mu,N_std = norm.fit(markov_output_incremental[:,0])
        Wi_mu,Wi_std = norm.fit(markov_output_incremental[:,1])
        Jw_mu,Jw_std = norm.fit(markov_output_incremental[:,2])
        N_samples = np.random.normal(N_mu, N_std, num_forward_model_samples)
        Wi_samples = np.random.normal(Wi_mu,Wi_std,num_forward_model_samples)
        Jw_samples = np.random.normal(Jw_mu, Jw_std, num_forward_model_samples)

        for i in range(0,len(N_samples)):
            incremental_MBAL_output = MBAL_inverse([N_samples[i]*1000000,Wi_samples[i]*1000000,Jw_samples[i],0], production_data, PVT_data, big_time_steps[time_index[1]], Swc, True)
            #print(shape(incremental_MBAL_output[0:360,1]))
            #plt.figure(444)
            #plt.plot(production_data[0:360,0], incremental_MBAL_output[0:360,1])
            #print(shape(incremental_MBAL_output))
            if(temp_count == 0):
                total_MBAL_output = incremental_MBAL_output[0:360,1]
            else:
                total_MBAL_output = np.vstack((total_MBAL_output,incremental_MBAL_output[0:360,1]))
            temp_count = temp_count + 1



#Save the output to binary files
np.save(file_dir_string + "SAMPLING_BASED_OUTPUTS/total_markov_output.npy", total_markov_output)
np.save(file_dir_string + "SAMPLING_BASED_OUTPUTS/total_proposal_parameters.npy", total_proposal_parameters)
np.save(file_dir_string + "SAMPLING_BASED_OUTPUTS/chain_length.npy", chain_length)
np.save(file_dir_string + "SAMPLING_BASED_OUTPUTS/big_time_steps.npy", big_time_steps)
np.save(file_dir_string + "SAMPLING_BASED_OUTPUTS/time_index.npy", time_index)
np.save(file_dir_string + "SAMPLING_BASED_OUTPUTS/num_forward_model_samples.npy", num_forward_model_samples)
np.save(file_dir_string + "SAMPLING_BASED_OUTPUTS/total_MBAL_output.npy", total_MBAL_output)
```

Appendix C – PVT Correlations

Bubble-point pressure (Glaso 1980):

$$p_b = 10^{1.7669+1.7447\log(p_b^*)-0.30218\left(\log(p_b^*)\right)^2}$$

$$p_b^* = \left(\frac{R_{sb}}{\gamma_g}\right)^{0.816} \cdot \frac{T^{0.172}}{\gamma_{API}^{0.989}}$$

Solution GOR (Glaso 1980):

$$R_s = \gamma_g \left[N_{pb}\left(\frac{\gamma_{API}^{0.989}}{T^{0.172}}\right)\right]^{1.2255}$$

$$N_{pb} = 10^{\left[2.8869-\left[14.1811-3.3093log(p_b)\right]^{0.5}\right]}$$

Oil Formation Volume Factor p< pbub (Glaso 1980):

$$B_{ob} = 1 + 10^{\left[-6.58511+2.91329log(B_{ob}^*)-0.27683[log(B_{ob}^*)]^2\right]}$$

$$B_{ob}^* = R_{sb}\left(\frac{\gamma_g}{\gamma_o}\right)^{0.526} + 0.986T$$

Oil compressibility factor & Formation Volume Factor, p > p_bub:

$$c_o = \left(-1433 + 5R_s - 17.2T - 1180.0\gamma_g + 12.61\gamma_{API}\right)/(p \cdot 10^5)$$

$$B_o = B_{ob} exp\left(-c_o \cdot (p - p_{bub})\right)$$

Dead Oil Viscosity:

$$\mu_{od} = 10^{A} - 1$$

$$log(A) = 3.0324 - 0.02023\gamma_{API} - 1.163T$$

Oil with dissolved gas viscosity below bubble point:

$$\mu_{ob} = C{\mu_{od}}^{B}$$

$$C = 10.715(R_s + 100)^{-0.515}$$

$$B = 5.44(R_s + 150)^{-0.338}$$

Oil viscosity above bubble-point:

$$\mu_o = \mu_{ob}(p/p_b)^{D}$$

$$D = 2.6p^{1.187}exp(-11.513 - 8.98 \cdot 10^{-5}p)$$

Gas Critical Pressure and Temperature (Sutton 1985):

$$p_c = 756.8 - 131\gamma_g - 3.6{\gamma_g}^2$$

$$T_c = 169.2 + 349.5\gamma_g - 74{\gamma_g}^2$$

Gas Compressibility Factor, Z (Brill & Beggs 1974):

$$A = 1.39(T_r - 0.92)^{0.5} - 0.36T_r - 0.10$$

$$B = (0.62 - 0.23T_r)p_r + \left(\frac{0.066}{T_r - 0.86} - 0.037\right){p_r}^2 + \frac{0.32{p_r}^6}{10^E}$$

$$C = 0.132 - 0.32log(T_r)$$

$$D = 10^F$$

$$E = 9(T_r - 1)$$

$$F = 0.3106 - 0.49T_r + 0.1824{T_r}^2$$

$$Z = A + \frac{1-A}{e^B} + C{p_r}^D$$

$$p_r = p/p_c \qquad T_r = T/T_c$$

Gas Formation Volume Factor (Real Gas Law):

$B_g = 0.0283 * Z * \frac{T}{P}$, where T = Rankine

Gas Viscosity (Lee et. al. 1966):

$$\mu_g = 10^{-4} a \exp\left(b\left(\rho_g/62.43\right)^c\right)$$

$$\rho_g = pM_g/(ZRT)$$

$$M_g = \gamma_g M_{air}$$

Water Density (McCain 1990):

$$\rho_w = 62.328 + 0.438603\, w_s + 1.60074 \cdot 10^{-3}\, {w_s}^2$$

Water Viscosity:

$$\mu_w = \mu_{wT}\,(0.9994 + 4.0295 \cdot 10^{-5}\, p + 3.1062 \cdot 10^{-9}\, p^2)$$

$$\mu_{wT} = 109.574 - 2.63951 \cdot 10^{-2}\, w_s + 6.79461 \cdot {w_s}^2 + 5.47119 \cdot 10^{-5}\, {w_s}^3 - 1.55586 \cdot 10^{-6}\, {w_s}^4$$

$$D = 1.12166 - 2.63951 \cdot 10^{-2}\, w_s + 6.79461 \cdot 10^{-4}\, {w_s}^2 + 5.47119 \cdot 10^{-5} {w_s}^3 - 1.55586 \cdot 10^{-6} {w_s}^4$$

Water Compressibility (Meehan 1980):

$$C_w = 10^{-6}\,(C_0 + C_1 T + C_2 T^2)\,\psi_s$$

$$C_0 = 3.8546 - 0.000134\, p$$

$$C_1 = -0.01052 + 4.77 \cdot 10^{-7}\, p$$

$$C_2 = 3.9267 \cdot 10^{-5} - 8.8 \cdot 10^{-10}\, p$$

$$\psi_s = 1 + (-0.052 + 2.7 \cdot 10^{-4} T - 1.14 \cdot 10^{-6} T^2 + 1.121 \cdot 10^{-9} T^3) w_s$$

Water Formation Volume Factor (McCain 1990):

$$B_w = \left(1 + \Delta V_{wp}\right) \cdot (1 + \Delta V_{wT})$$

$$\Delta V_{wp} = -(3.58922 \cdot 10^{-7} + 1.95301 \cdot 10^{-9}\, T)p - \ (2.25341 \cdot 10^{-10} + 1.72834 \cdot 10^{-13}\, T)p^2$$

$$\Delta V_{wp} = -1.0001 \cdot 10^{-2} + 1.33391 \cdot 10^{-4} T + 5.50654 \cdot 10^{-7} T^2$$

$$B_w = {B_w}^{ref} exp\left(-C_w(p - p^{ref})\right)$$

Appendix D – Procedure for Generating Random Simulation Model Properties

The Weibull distribution is used to generate the porosities for the reservoir simulation grid. The Weibull Probability density function (pdf) and cumulative density function (cdf) are defined as shown in equation 1 and 2. To characterize Porosity variability in the synthetic model, Weibull parameters of $\lambda = 0.2$ and $k = 12$ are used. The resulting Porosity distribution is shown in Figure E1.

$$Weibull\ PDF: \quad f(x) = \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k} \qquad x \geq 0$$

$$Weibull\ CDF: \quad f(x) = 1 - e^{-\left(\frac{x}{\lambda}\right)^k} \qquad x \geq 0$$

Porosity is often assumed to be a principal geologic parameter that permeability co-varies with. As such, the sampling strategy employed here, starts by drawing Monte Carlo samples for Porosity using the Weibull CDF. Once a set of Porosity samples have been obtained a second loop is started, where for each Porosity sample a random noise component is sampled from a Gaussian distribution (mean = 0, std=0.017) , before a corresponding permeabiltiy is calculated by using a log K vs. Porosity characteristic line. Figure X shows the resulting Porosity vs. log K line.

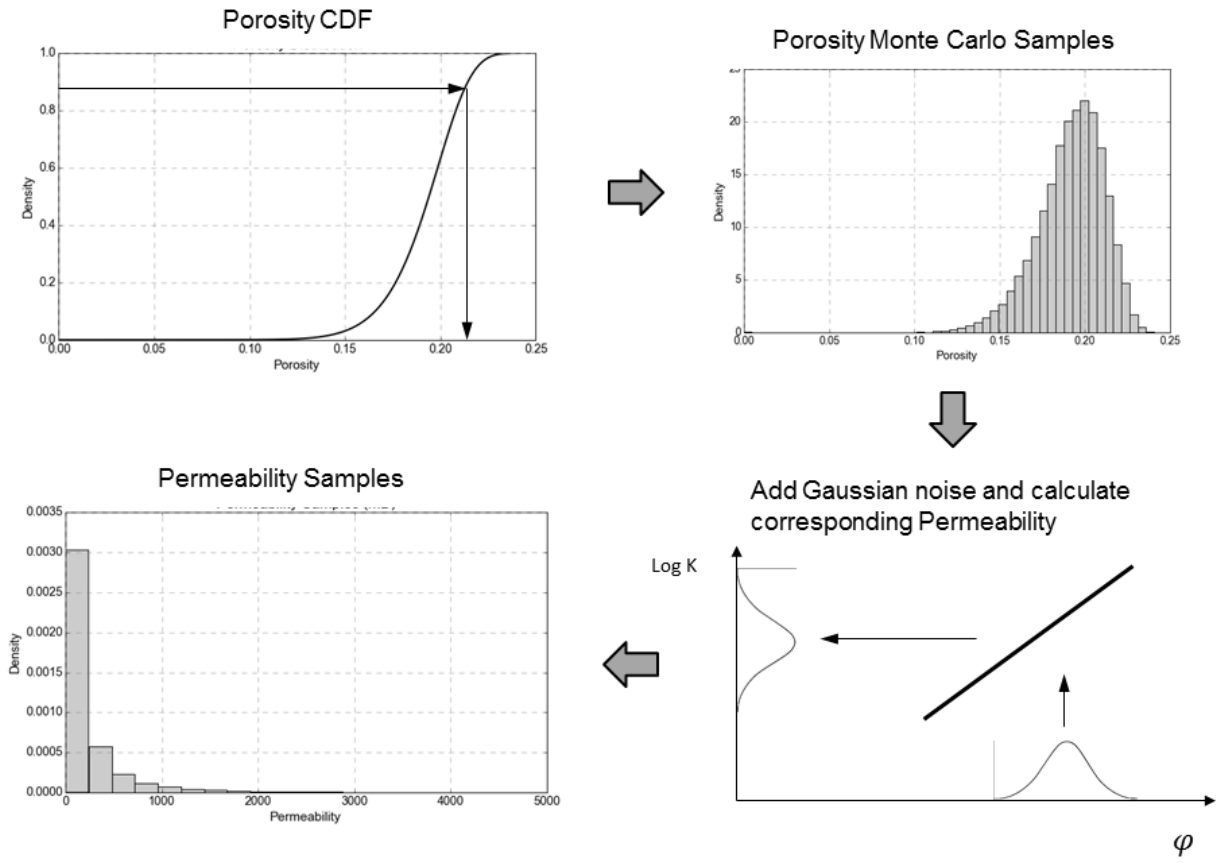
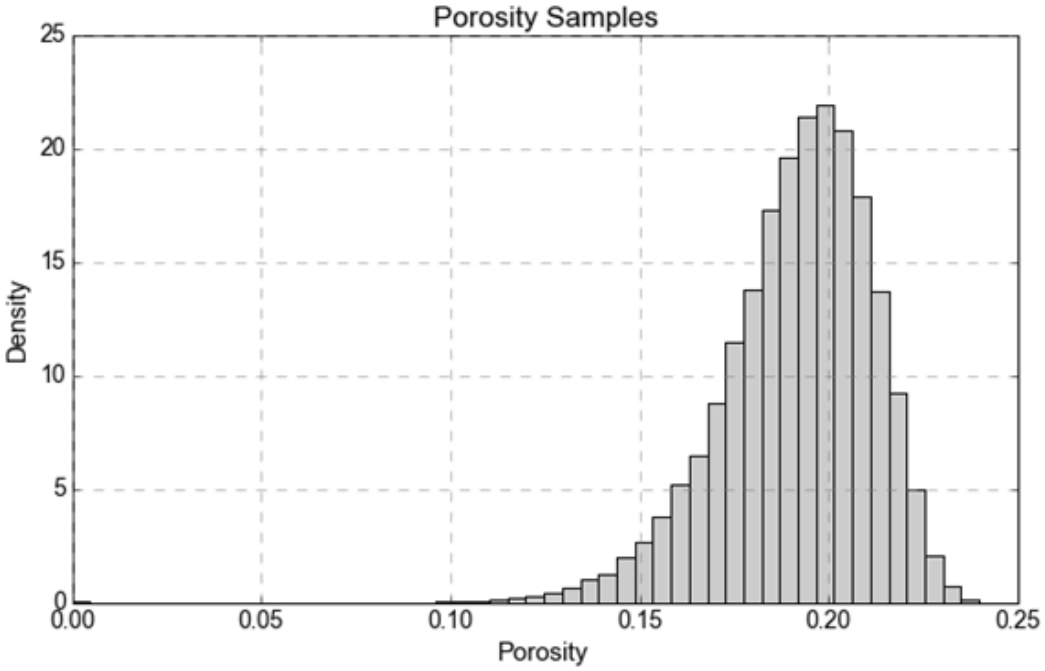
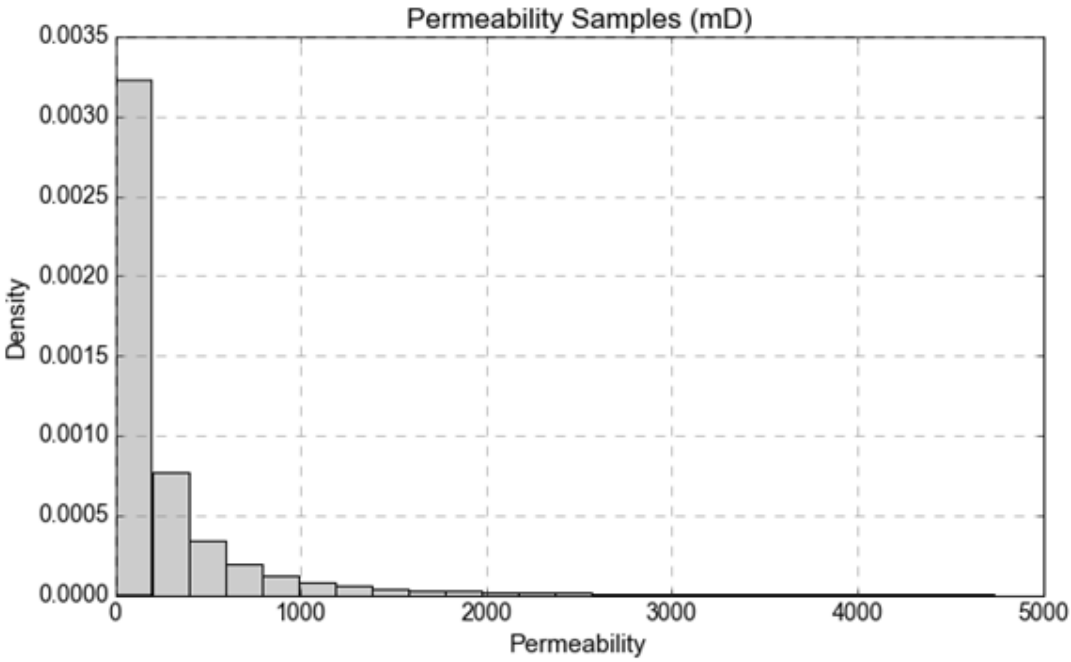


Figure E1 – Porosity/Permeability Sampling Process


Porosity Samples
Density
Porosity
0
5
10
15
20
25
0.00
0.05
0.10
0.15
0.20
0.25


Permeability Samples (mD)
Density
Permeability
0.0000
0.0005
0.0010
0.0015
0.0020
0.0025
0.0030
0.0035
0
1000
2000
3000
4000
5000

Appendix E – Capillary Pressure Model

In general, the shape of the capillary pressure curve will depend mostly on the permeability of the reservoir. Highly permeable reservoirs will be associated with sharper transition zones, while less permeable reservoirs will have longer transition zones. The following logarithmic expressions were used to generate capillary pressure curves for the simulation model in this study.

$$P_{cwo}(S_w) = \alpha_1 ln\left(\frac{1 - S_{wx} - S_{wc}}{1 - S_w - S_{orw}}\right) \qquad S_w > S_{wx}\ and\ S_w \leq 1 - S_{orw}$$

$$P_{cow}(S_w) = \alpha_2 ln\left(\frac{1 - S_{wx} - S_{wc}}{S_w - S_{wc}}\right) \qquad S_w > S_{wc}\ and\ S_w \leq S_{wx}$$

$$\alpha_1 = constant$$

$$\alpha_2 = -\left(\frac{S_{wx} - S_{wc}}{1 - S_{wx} - S_{orw}}\right)$$

Appendix F – Corey Functions for Relative Permeability

The Corey exponents n_o, n_w, n_g range in values from 1 to 6 and depend on the wettability characteristics of the rock and direction of change of the wetting phase saturation. A drainage process occurs when the wetting phase is decreasing and an imbibition process occurs when the wetting phase increases. In this study, the reservoir is assumed to be water-wet initially, meaning that the drainage process for the oil-water system is associated with decreasing oil saturation. Similarly, the imbibition process for a water wet oil-water system is associated with increasing water saturation due to either water injection or aquifer influx. If the pressure drops below the bubble point, gas will come out of solution. For a gas-oil system that occurs after dropping below the bubble point, the oil is assumed to be the wetting phase in the presence of connate water. The drainage process for the oil-gas system is therefore associated with decreasing oil saturation and the imbibition process associated with increasing oil saturation. Practically, speaking the drainage process for an oil-gas system can be visualized as a primary or secondary gas cap moving into the oil zone. Likewise, the imbibition process for an oil-gas system can be visualized as the gap cap moving upwards due to pressurization of the oil column during down-dip water injection or aquifer influx. The upwards gas gap movement is undesirable because it leaves behind trapped gas and reduces recoverable oil (also referred to as 'smearing' of oil zone).

Oil-Water Drainage (water saturation decreasing):

$$k_{ro}(S_w) = k_{row,max,d} \cdot \left(\frac{1 - S_w}{1 - S_{wc}}\right)^{n_{o,d}} \qquad k_{row,max,d} = 1.0$$

$$k_{rw}(S_w) = k_{rw,max,d} \cdot \left(\frac{S_w - S_{wc}}{1 - S_{wc}}\right)^{n_{w,d}} \qquad k_{rw,max,d} = 1.0$$

Oil-Water Imbibition (water saturation increasing):

$$k_{ro}(S_w) = k_{row,max,i} \cdot \left(\frac{1 - S_w - S_{orw}}{1 - S_{orw} - S_{wc}}\right)^{n_{ow,i}} \qquad k_{row,max,i} = 1.0$$

$$k_{rw}(S_w) = k_{rw,max,i} \cdot \left(\frac{S_w - S_{orw}}{1 - S_{orw} - S_{wc}}\right)^{n_{w,i}} \qquad k_{rw,max} < 1.0$$

Oil-Gas Drainage (oil saturation decreasing):

$$k_{rog}(S_g) = k_{rog,max,d} \cdot \left(\frac{1 - S_g - S_{wc} - S_{org}}{1 - S_{wc} - S_{org}}\right)^{n_{og,d}} \qquad k_{rog,max} = 1.0$$

$$k_{rg}(S_g) = k_{rg,max,d} \cdot \left(\frac{S_g - S_{gc}}{1 - S_{wc} - S_{gc}}\right)^{n_{g,d}} \qquad k_{rg,max} = 1.0$$

Oil-Gas Imbibition (oil saturation increasing):

$$k_{rog}(S_g) = k_{rog,max,i} \cdot \left(\frac{1 - S_g - S_{wc} - S_{org}}{1 - S_{wc} - S_{org}}\right)^{n_{og,i}} \qquad k_{rog,max} = 1.0$$

$$k_{rg}(S_g) = k_{rg,max,i} \cdot \left(\frac{S_g - S_{gt}}{1 - S_{wc} - S_{gt}}\right)^{n_{g,d}} \qquad k_{rg,max} = 1.0$$

Appendix G – Eclipse Input File

```
RUNSPEC
TITLE
MODEL FOR ACJ MASTERS SYNTHETIC DATA

INCLUDE
DIMENS.INC /

UNIFOUT

OIL
GAS
WATER
DISGAS

METRIC

EQLDIMS
1 100 /

-- #wells #cell connections
WELLDIMS
1        10       1        1        /

START
1 'APR' 2011   /

TABDIMS
2 1 30 30 1* 30/

SATOPTS
'HYSTER' /

GRID
ECHO

GRIDFILE
   1    /

INCLUDE
COORD.INC /

INCLUDE
ZCORN.INC /
```

```
INCLUDE
PORO.INC /

INCLUDE
PERMX.INC /

INCLUDE
PERMY.INC /

INCLUDE
PERMZ.INC /

INIT

PROPS

PVTO
1.234771169   1       1.036596191   2.177904133 /
3.633249464   5       1.040876084   2.003647206 /
6.066287279   10      1.045363114   1.857590312 /
12.67882075   25      1.058227986   1.56361029 /
16.97507028   35      1.067055875   1.425068732 /
23.48203042   50      1.081038623   1.263476231 /
34.72165746   75      1.106665652   1.068075371 /
46.61378097   100     1.135429946   0.927375103 /
59.25354089   125     1.167465161   0.82007224 /
72.70470983   150     1.202855793   0.735005819 /
87.02103867   175     1.241669255   0.665624204 /
102.2536347   200     1.283966774   0.607781132 /
120     227.3493064   1.334219525   0.554545083
        250     1.328696557   0.571395807
        275     1.323676955   0.591791657
        300     1.319508442   0.613894154
        325     1.315991494   0.637538502
        350     1.312984429   0.662573523
        375     1.310383864   0.688856576
        400     1.308112596   0.716250111
        425     1.306111805   0.744619359
        450     1.304335894   0.773830867
        475     1.302748967   0.803751663
        500     1.301322384   0.834248875 /
/

PVDG
1 1.24075 0.01261
5 0.24692 0.01264
```

10 0.12268 0.01269
25 0.04811 0.01289
35 0.0339 0.01306
50 0.02325 0.01337
75 0.01499 0.01402
100 0.01093 0.01484
125 0.00855 0.01583
150 0.00703 0.01697
175 0.006 0.01822
200 0.00527 0.01956
227.34931 0.0047 0.02107
250 0.00435 0.02231
275 0.00405 0.02366
300 0.00381 0.02497
325 0.00362 0.02622
350 0.00347 0.02742
375 0.00334 0.02858
400 0.00323 0.0297
425 0.00313 0.03078
450 0.00305 0.03183
475 0.00297 0.03285
500 0.00291 0.03384
/

PVTW
-- Pref Bw Cw ViscW
300 1.02239 2.77182E-05 0.551284 0.00E+00 /
/

DENSITY
849.010 1071.864 0.7 /

ROCK
300 5E-05 /

SWOF
0.2 2.45986277400839E-58 1 2
0.200000005 1.9301011073869E-29 0.999999990625 1.93906843737245
0.201 6.90533966002491E-11 0.998125586059627 0.718461172766793
0.21 2.18366013427717E-07 0.981308716396119 0.488202663467388
0.22 2.47052942200657E-06 0.96273536083391 0.418887945411393
0.23 1.02119678073829E-05 0.944280686885525 0.378341434600577
0.24 2.79508497187475E-05 0.925945462756851 0.349573227355399
0.29 0.000477566471366412 0.836089063362869 0.268480205733766
0.34 0.00224198741485303 0.749343462639129 0.224296930505862
0.39 0.00652863890732859 0.665824500619345 0.193758765550744

0.44 0.0147885090526395 0.585662018573853 0.170397280432594
0.49 0.0286799267756021 0.509002931597648 0.151473080468741
0.54 0.0500450647136885 0.436015338032964 0.135566611005772
0.59 0.0808930424957353 0.366894164746456 0.121846498854424
0.64 0.123387002329054 0.301869176962472 0.109783700075562
0.69 0.1798338017963 0.241216804711031 0.0990206336563255
0.74 0.252675541391583 0.18527850657861 0.0893042588109607
0.79 0.34448243812698 0.134491228803219 0.0804489190768162
0.84 0.457946721791958 0.0894427190999913 0.0723143551314209
0.89 0.595877329988409 0.0509863645987824 0.0647920130076622
0.94 0.76119524130925 0.0205395959064435 0.0577961541469711
1 1 0 0.05
/
0.200000000000005 4.71782331762052E-54 0.999999999999955 2
0.200000005 2.98580377317919E-31 0.999999955000001 1.79098551207127
0.201 4.1588690589309E-11 0.991031447539367 0.57037824746562
0.210000000000005 2.62406897746886E-07 0.91309789296114 0.340119738166166
0.220000000000005 3.65501556233797E-06 0.832186274715232 0.270805020110196
0.230000000000005 1.70622387560118E-05 0.756964201418113 0.230258509299388
0.240000000000005 5.09100137063666E-05 0.687139988129428 0.201490302054214
0.290000000000005 0.0011094220230758 0.409413666613231 0.120397280432588
0.340000000000005 0.00594650491928197 0.228032229770755 0.0762140052046861
0.390000000000005 0.0189774969603207 0.116348986013036 0.0456758402495689
0.440000000000005 0.0461081784607557 0.0527248127890013 0.0223143551314189
0.490000000000005 0.0946428127598162 0.0201660949082708 0.00339015516756642
0.540000000000005 0.173218842295269 0.00593164160151488 -0.0223143551314241
0.590000000000005 0.291756643231723 0.00109875803423297 -0.0597837000755667
0.640000000000005 0.46141894042188 7.18316110914689E-05 -0.120397280432602
0.660000000000005 0.546329947435499 1.15852375029537E-05 -0.160943791243423
0.680000000000005 0.642232011705081 5.11999999999408E-07 -0.23025850929943
0.690000000000005 0.694576924669561 2.26274169979171E-08 -0.299573227355451
0.695000000000005 0.721896610789948 9.99999999995357E-10 -0.368887945411497
0.699000000000005 0.744315940855691 7.15541752783312E-13 -0.52983173665532
0.699600000000005 0.747722552374498 1.15852375022818E-14 -0.621460809843521
1 1 0 -2
/

SGOF
0 0 1 0
0.035 0 0.792766765644481 0
0.085 0.016709476881206 3 0.550870981716026 0
0.135 0.0472615376511234 0.365286104871042 0
0.185 0.0868249887784407 0.227976330113399 0
0.235 0.133675815049651 0.131124179221773 0
0.285 0.186817630874193 0.0671582016022023 0
0.335 0.245578153366725 0.0287901950785023 0

0.385 0.309463842559586 0.00906836673681601 0
0.435 0.378092301208987 0.00146089407393082 0
0.485 0.451155875792571 0.000013390229330315 0
0.535 0.528400054545388 0 0
0.585 0.609609717838873 0 0
0.635 0.694599910227526 0 0
0.685 0.783209383857804 0 0
0.735 0.875295926423717 0 0
0.785 0.97073288527125 0 0
0.8 1 0 0
/
0 0 1 0
0.035 0 0.758988515516814 0
0.085 0 0.492602658159541 0
0.135 0 0.302431392045342 0
0.185 0 0.172780245967101 0
0.235 0 0.0895876829741821 0
0.285 0 0.0404743731401657 0
0.335 0.0108809463266352 0.0148031101771948 0
0.385 0.0491771285083817 0.00375461816423567 0
0.435 0.10797380322933 0.000429521332995507 0
0.485 0.184477298822442 1.63328046596546E-06 0
0.535 0.277055233633678 0 0
0.585 0.384580495492682 0 0
0.635 0.506205855581335 0 0
0.685 0.641260121229155 0 0
0.735 0.789192100436947 0 0
0.785 0.949537122383164 0 0
0.8 1 0 0
/

EHYSTR
1* 0 /

RPTPROPS
'PVTO' 'PVTW' 'PVDG' /

REGIONS

SATNUM
45000*1 /

IMBNUM
45000*2 /

SOLUTION

```
EQUIL
1500 250 2250 1.5 1500 0 1 0 1* /

RSVD
1500 120
9000 120
/

RPTSOL
  -- Initialisation Print Output
  --
'SWAT' 'RESTART=2' 'FIP=1' /

SUMMARY

FPR

FOPT

FGPT

FWPT

FGOR

FWCT

FOPR

FGPR

FWPR

WBHP
'OP' /

EXCEL

SCHEDULE
--
--  WELSPECS and COMPDAT define well information in both
--  standard and LGC models.
--

DEBUG
```

```
20* 1 /

EXTRAPMS
4 /

WELSPECS
--wname group i j BHP prefphase
'OP' 'GROUP' 20 30 2159 'OIL' /
/

COMPDAT
-- ic  jc  k_hi  l_lo
'OP'   20   30   1   10 'OPEN'  1*    1*    0.2   1*      1*      1* /
/

WCONPROD
'OP','OPEN','LRAT'  1* 1* 1* 1750  1* 90  /
/
RPTSCHED
  'RESTART=1' 'FIP=1' 'WELLS=5' 'SUMMARY=3' 'CPU=2' 'WELSPECS' 'NEWTON=1' /
/

RPTRST
BASIC=3 FREQ=30 PBPD /

TSTEP
   400*10
/

RPTONLY

--WELTARG
--'OP' 'LRAT' 0 /
--/
RPTSCHED
  'RESTART=2' 'FIP=1' 'WELLS=1' 'SUMMARY=1' 'CPU=2' 'WELSPECS' 'NEWTON=1'
/


END
```