Louisiana State University

LSU Scholarly Repository

LSU Master's Theses | Graduate School

2014

Investigating Different Coding Environments for Simplified Reservoir Characterization Models

Atheer Mohammad Al Attar
Louisiana State University and Agricultural and Mechanical College

Follow this and additional works at: https://repository.lsu.edu/gradschool_theses

Part of the Petroleum Engineering Commons

Recommended Citation

Al Attar, Atheer Mohammad, "Investigating Different Coding Environments for Simplified Reservoir Characterization Models" (2014). *LSU Master's Theses*. 1809.
https://repository.lsu.edu/gradschool_theses/1809

This Thesis is brought to you for free and open access by the Graduate School at LSU Scholarly Repository. It has been accepted for inclusion in LSU Master's Theses by an authorized graduate school editor of LSU Scholarly Repository. For more information, please contact gradetd@lsu.edu.

INVESTIGATING DIFFERENT CODING ENVIRONMENTS FOR SIMPLIFIED RESERVOIR CHARACTERIZATION MODELS

A Thesis

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Masters of Science

in

The Department of Petroleum Engineering

by
Atheer M. Al Attar
B.S., Mechanical University of Basra, 2004
December 2014

ACKNOWLEDGMENTS

I would like to thank my advisor Dr. Richard Hughes, for his continuous support, patience and guidance along the way. His advice made a huge impact on my research and the way I deal with problems. At this point of my life I feel I am lucky to have the chance to work with a professor like him.

I would also like to thank my other two committee members Dr. Thompson and Dr. Tyagi for their feedback, support and advice on my research and to the way I run my tasks.

My Dad and Mom who keep calling me all the time asking about the research deadlines and results, I am just blessed having you in my life.

My lovely fiance Reeman, Thank you very much for all of calls overseas asking and worrying about when is this task will be achieved.

.

This work is dedicated to my dad Mohammad and my mom Salima.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF NOMENCLATURE

RM	is some really long explanation of this value
CM	is the capacity model suggested by Al-Yousef (2006)
CRM	is capacitance resistivity model suggested by Sayarpour (2008)
τ	time constant- units 1/time where the time is the input data time increment
λ	is the connectivity
$\hat{q}$	is the calculated value of the production rate
β_o	is the unbalanced model term
i	is the injection rate
ε	is an error term
σ_{II}	is the injector-injector covariance
σ_{Ij}	is the injector-producer covariance
β_{ij}	is the connectivity value in Albertoni model
R^2	is the correlation coefficient
C_t	Compressibility (1/psi)
k	Permeability (mD)
μ	Viscosity
ϕ	Porosity
V_p	Pore volume
J	Productivity index
P_{wf}	Well flowing pressure
δn	is the time step
i'	Filtered injection
P'_{wf}	Filtered well flowing pressure
ARE	Average Relative Error
MAE	is Mean Average Error
MSE	is Mean Squared Error
CRMP	Capacitance Resistivity Model-Producer
CRMIP	Capacitance Resistivity Model-Producer/Injector
σ_{pp}	Primary production covariance
C^T_{pp-I}	Transpose of primary production-injection covariance
C^T_{pp-BHP}	Transpose of primary production-bottom hole pressure covariance
C_{pp-I}	Primary production-injection covariance
C_{II}	Injection-Injection covariance
C^T_{I-BHP}	Transposed Injection-Bottom hole pressure covariance
C_{pp-BHP}	Primary production-Bottom hole pressure covariance
$C_{BHP-BHP}$	Bottomhole pressure -Bottomhole pressure covariance
C_{I-BHP}	Injection-Bottomhole pressure covariance
λ_p	Connectivity value as per Al-Yousef (2006)
ν_{kj}	Pressure change coeficient for prodcers in Al-Yousef (2006)
σ_{pp-q_j}	Covariance of production rate and primary production
C_{BHP-q_j}	Covariance of bottom hole pressure and production
BFGS	BroydenFletcherGoldfarbShanno algorithm

L-BFGS-B	Limited-memory BFGS
NelderMead	The NelderMead method or downhill simplex method
SSE	Sum of sequared error
CMG-IMEX	a black oil simulator by CMG (Computer Modeling Group)
LOF	Local Outlier Factor
MLR	Multivariate Linear Regression
USGS	U.S. Geological Survey
UBMLR	Unbalanced Multivariate Linear Regression
λ^{-1}	Inverse lambda, a notation suggested by the author was used to differentiate it from λ. It was used to calculate the missing injection rates.

ABSTRACT

Reservoir characterization is one of the most important tasks that determines the recovery plan for a specific reservoir. This process incorporates a significant amount of data acquisition and processing to finally develop an acceptable model that matches the production history and can forecast the future production behavior. The model also should be able to adapt to changes along the way: adding or removing producers or injectors, changing the injection pattern, recompletions and converting wells are all examples of possible changes that are common in the oil and gas industry. These goals will not be realistic if an approximate understanding of the subsurface structures and heterogeneities in the system are not understood. The more accurate the understanding, the less effort and cost will be spent on secondary or tertiary recovery operations, since the ultimate goal is maximization of recovery at minimal operating costs. In this thesis the focus will be on the simple models and three of the most common simple models were investigated: the Resistance Model (RM) by Albertoni et al (2003), the Capacitance Model (CM) by Al-Yousif (2006), the Capacitance-Resistance Model (CRM) by Sayarpuor (2008). These models were coded using R-project and Matlab®, and these codes were generalized to handle any case size (i.e. number of injectors and producers). Both R-Project and Matlab® were found to be capable of solving RM problems. Several optimization algorithms were tested in R-project and Matlab® to solve the CM and CRM non-linear optimization problems.

Only the Matlab® solution solved these models efficiently. To test these implementations, data for two synthetic cases were used. A 5×4 and 16×25 i.e. (producers×injectors) and the results were discussed. The issue of having missing or corrupted data in the field data was studied. The maximum acceptable rate of missing data was found to be 6%. Having more than this amount of missing data resulted in solutions that were unacceptable compared to complete data results. To deal with the missing data, two options were evaluated: truncate the data, which means less information for the model, or correctly estimate the missing value. The RM model was utilized in the process of estimating the missing values. Two cases: were tested: 15% and 30% of missing data rates respectively. The lowest value of R^2 obtained between the good dataset and the missing dataset was ~ 0.7. Finally, a field case study was performed which had 189 producers and 65 injectors.

CHAPTER 1
INTRODUCTION

The finite difference reservoir simulation model is the main tool that reservoir engineers use to develop an understanding of the reservoir. There are many black oil simulators in the market. All share limitations that make the process of quick reservoir model inference almost impossible without significant effort. Some of these limitations are:

- The need for large computational power and storage space for realistic problems.
- Simulation models require data preparation effort to build the model in addition to the time required to run field-scale simulations.
- The need for highly trained individuals to perform the tasks of model creation, interpretation, updating and prediction.

For many problems, full-scale reservoir characterization and modeling is unnecessary and simple models are sufficient to infer flow characteristics. There are several simple models that have been proposed. For the resistance model (RM), injection and production rates and well locations are all the data that is needed to understand flow behavior trends in the reservoir and connectivity between injector/producer pairs (Albertoni et al., 2003). The Capacitance Model (CM) proposed by Al-Yousef (2006) added knowledge of bottomhole pressures (BHPs) to the required data. The whole approach in this model was different from what Albertoni et al. (2003) did, as he started with a simple mass balance over the drained volume (V_p) between each injector/producer pair in the reservoir. The output of this model was a set of values that represent the time constants (τ) and the connectivity (λ) for each injector/producer pair. Sayarpour (2008) presented an analytical solution to the continuity equation using superposition in time and space and called this the Capacitance-Resistance Model (CRM). The model output was also a set of variables for each injector/producer pair. The difference here is the objective function used which will be discussed later. Simple models provide quick understanding to reservoir properties and guide the process of performance optimization. In this work, these different types of simplified models will be developed to evaluate their use for reservoir characterization problems and to explore whether these tools can simplify or improve the modeling process or model results either through less user interaction or faster model performance. This thesis will first discuss the different modeling tools to be evaluated. Then an implementation of the models will be developed followed by proposed enhancements. Results from these models will be presented and further work will be proposed.

Chapter 3 includes the coding work that has been done using Matlab® and R-project programming languages. Codes were implemented to solve the RM, CM, and CRM models and to represent the results. These codes were generalized to handle any case size with as little user input as possible. Several issues in these two coding platforms were identified and a comparison has been done for the two implementations. Chapter 4 presents two synthetic field cases from which data was used to run the RM, CM, and CRM implementations. The issue of missing and corrupted data has been studied in Chapter 5, where different patterns and scenarios of missing data have been simulated. Several recommendations will be given about the maximum amount of missing data to work with. A complete decision procedure was implemented in Matlab® to handle different missing data scenarios and patterns. Chapter 6 presents a field case study, were the data from 189 producers and 65 injectors is presented. The use of the RM model resulted in about 49% of the injector-producer pairs having unphysical connectivity values and the approach that was suggested by Albertoni et al. (2003) to handle this issue was implemented in addition to CM and CRM models. Several conclusions were made and an integrated procedure was developed that handles the missing and corrupted data automatically, runs the previously mentioned models and presents the data.

CHAPTER 2
LITERATURE SURVEY

2.1 Resistance model (RM)

Albertoni et al (2003) presented the resistance model (RM) where multivariate correlation between the injection and production rates was used to infer injection well influences on production well behavior. The approach of the work was to assume that the production from each well was a linear combination of the injection from each injector. Thus this is a multivariable linear regression between injection rates and the production rates using the model:

$$\hat{q}_j = \beta_{oj} + \sum_{i=1}^{I} \beta_{ij} i_i + \varepsilon \tag{2.1}$$

where β_{oj} represents the primary production for each well in the case where total injection and total production are not balanced (equal). This term is omitted in cases where injection and production rates are balanced. Minimizing the squared differences between the calculated production rates, $\hat{q}_j$ and observed production rates is how the values of β_{ij} for each injector-producer pair are obtained.

$$\begin{pmatrix} \sigma_{11}^2 & \sigma_{12}^2 & \cdots & \sigma_{1I}^2 \\ \sigma_{21}^2 & \sigma_{22}^2 & \cdots & \sigma_{2I}^2 \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{I1}^2 & \sigma_{I2}^2 & \cdots & \sigma_{II}^2 \end{pmatrix} \begin{pmatrix} \beta_{1j} \\ \beta_{2j} \\ \vdots \\ \beta_{Ij} \end{pmatrix} = \begin{pmatrix} \sigma_{1j}^2 \\ \sigma_{2j}^2 \\ \vdots \\ \sigma_{Ij}^2 \end{pmatrix} \tag{2.2}$$

where the $\sigma_{11}^2 \ldots \sigma_{II}^2$ are the injector-injector covariance values and the $\sigma_{1j}^2 \ldots \sigma_{Ij}^2$ are the injector-producer covariance values.

They proposed the concept of diffusivity filters to account for any time lag between the injector-producer pairs in field applications. The generic steps of the RM are:

- Acquire and prepare the injection and the production data.
- Check the records for any missing or zero values.

- Limit the time frame for application of the technique to where there are no missing data or zero injection rate values (this thesis will present an approach for handling missing data).
- Compute the injector-injector covariance matrix and injector-producer covariance vector obtaining values for well pair connectivity values β_{ij} to solve Eqn 2.1.
- Plot the predicted values and calculate the value of the residual squared errors, R^2.

2.1.1 Diffusivity filters

The development of diffusivity filters (DF) was proposed to account for any time lag between when the fluid is injected and when it is produced caused by the effect of permeability, porosity and the existence of different fluids with different compressibility and viscosity. The main task of the DF is to modify the injection rate to be an effective or time-shifted injection rate. All these factors were grouped into one value called the diffusivity constant.

$$\eta = \frac{k}{\phi \mu C_t} = \frac{1}{d} \tag{2.3}$$

Albertoni et al. (2003) called the reciprocal of the diffusivity constant the dissipation value (d).

2.1.2 Mathematical development

Albertoni (2002) presented the following mathematical development for the diffusivity filters. Pressure at any point of the reservoir caused by a change in an injection rate is given by:

$$\Delta P = C_1 \times E_i\left(-d\frac{r^2}{t}\right) \tag{2.4}$$

Using the superposition principle:

$$\Delta P = \begin{cases} C_1 \times E_i\left(-d\frac{r^2}{t}\right) & \text{if } t \leq 1 \\ C_1 \times \left[E_i\left(-d\frac{r^2}{t}\right) - E_i\left(-d\frac{r^2}{t-1}\right)\right] & \text{if } t > 1 \end{cases} \tag{2.5}$$

where E_i is the exponential integral function, C_1 is a constant, r is the distance from the point to the injector and d is the dissipation term described above. Utilizing the productivity index equation (Muskat, 1937):

$$q = J(\bar{P} - Pwf) \tag{2.6}$$

Eq. 2.7 can be written as:

$$\Delta q = \begin{cases} C_3 \times E_i\left(-d\frac{r^2}{t}\right) & \text{if } t \leq 1 \\ C_3 \times \left[E_i\left(-d\frac{r^2}{t}\right) - E_i\left(-d\frac{r^2}{t-1}\right)\right] & \text{if } t > 1 \end{cases} \tag{2.7}$$

and the (filtered) injection will be given by:

$$i_i^c(t) = \sum_{n=0}^{I} \alpha_{ik}^n i_i(t-n) \tag{2.8}$$

where α_{ik}^n terms are given by:

$$\alpha_{ij}^n = \frac{\int\limits_{t=n}^{t=n+1} \Delta q dt}{\int\limits_{t=0}^{t=12} \Delta q dt} \tag{2.9}$$

and hence the calculated production is given by:

$$\hat{q}_j(t) = \sum_{i=1}^{I} \lambda_{ij} i_i^c(t) \tag{2.10}$$

2.2 Capacitance model (CM)

One issue with the RM is the difference between the length of the diffusivity filters and the time step. In some cases the diffusivity filter may be less than the time step, and will not add any effect on the injection schedule. There are several works and versions of the capacitance model. Al-Yousef (2006) replaced the diffusivity filters in the RM model with a time constant. The CM is a mass balance over the drained pore volume, V_p, for each injector-producer well pair. Starting with the mass balance Al-Yousef (2006) presented this equation as:

$$C_t V_p \frac{d\bar{p}}{dt} = i(t) - q(t) \tag{2.11}$$

where C_t and V_p represent the total compressibility and the drained pore volume respectively. When the productivity equation (Eq. 2.6) is substituted into 2.11 a new constant is introduced. This constant is $\tau = \frac{C_t V_p}{J}$ and the revised form of Eqn 2.11 will be:

$$\tau \frac{d\bar{p}}{dt} + q(t) = i(t) - \tau J \frac{dp_{wf}}{dt} \tag{2.12}$$

Integrating this equation yields:

$$q(t) = q(t_o)e^{\frac{-(t-t_o)}{\tau}} + \frac{1}{\tau}e^{\frac{-(t)}{\tau}} \int\limits_{\zeta=t_o}^{\zeta=t} e^{\frac{\zeta}{\tau}} i(\zeta)d\zeta + J\left[p_{wf}(t_o)e^{\frac{-(t-t_o)}{\tau}} - p_{wf}(t_o) + e^{\frac{-(t-t_o)}{\tau}} \int\limits_{\zeta=t_o}^{\zeta=t} p_{wf}(\zeta)d\zeta\right] \tag{2.13}$$

where the first part of Eq. 2.13 represents the contribution of the primary production to the total production, the integral accounts for the production that comes from the injection and the term in brackets accounts for the effects of producer bottomhole pressure changes on the production. The final discrete model for the system will be:

$$q_j(n) = \lambda_p q_j(n)\ e^{\frac{-(t-t_o)}{\tau_{pj}}} + \sum_{i=1}^{I} i_i^{'}(n)\lambda_{ij} + \nu_j \left[p_{wf}(n)\ e^{\frac{-(t-t_o)}{\tau_p}} - p_{wf}(n) + p_{wf}^{'}(n)\right]_j \tag{2.14}$$

where:

$$i_i^{'}(n) = \sum_{m=n_o}^{m=n} \frac{\Delta n}{\tau_{ij}} e^{\frac{m-n}{\tau_{ij}}}\, i_i(m) \tag{2.15}$$

$$p_{wfj}^{'}(n) = \sum_{m=n_o}^{m=n} \frac{\Delta n}{\tau_{ij}} e^{\frac{m-n}{\tau_{kj}}}\, p_{wfj}(m) \tag{2.16}$$

So the equations above introduces the following unknowns:

1. λ_{ij}: refers to the connectivity between each injector i and producer j; initially this is unknown and it will be initialized by using the inverse of the distance between injector *i* and producer *j* i.e. $\lambda_{ij} = \frac{1}{distance(i,j)}$. It is an $I \times J$ matrix (where I is the number of the injectors and J is the number of the producers) as suggested by Sayarpour (2008).

2. τ_{ij}: are the time constants for each injector *i* and producer *j*; these are also initially unknown and will be initialized using the method suggested by Sayarpour (2008) described in section 3.2.4. This is also an $I \times J$ matrix.

3. ν_j: are coefficients that account for the effect of pressure changes in producers; initially they are unknown and they will be obtained from a matrix multiplication that will be presented in Ch. 3 (Eq. 3.1).

4. τ_{kj}: are the producer-producer time constants for each producer (Eq 2.16). Initially they are unknown and will be initialized in the same way as τ_{ij}; this is a $J \times J$ matrix.

5. τ_{pj}: are the time constants for each producer that accounts for the primary production. Initially the values are unknown and will be initialized in the same way τ_{ij}. This is a vector with J elements.
6. λ_p: refers to the primary production contribution to the total production. Initially unknown and they will be initialized by taking values less than the minimum value of the λ values obtained in step 1. This is also a vector with J elements.
7. n_o: refers to the first time step that will be used as the primary production contribution to the total production. Therefor $q_j(n_o)$ is the first flow rate value in the time sequence for each well.

Figure 2.1 shows these parameters and the relation between them in the control volume between each injector and producer pair. The solution of the set of equations given by 2.14, 2.15 and 2.16 is through nonlinear optimization to minimize the squared difference between the observed flow rates and calculated flow rates for each well.

$$MIN\left[\sum_{n-1}^{N}\left(q_j(n)-\hat{q}_j(n)\right)^2\right] \tag{2.17}$$

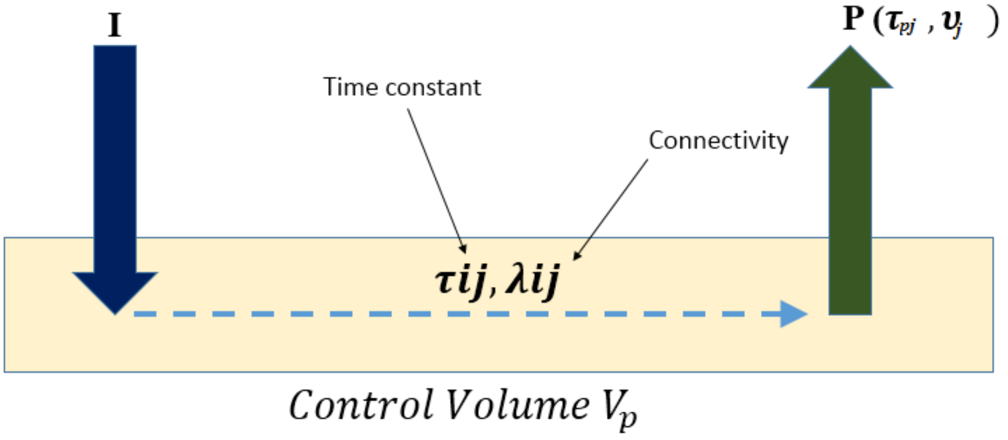


Figure 2.1: Illustrative diagram shows the main parameters of the CM Model.

2.3 Capacitance Resistance Model (CRM)

Sayarpour (2008) and Sayarpour et al. (2009a) presented analytical solutions for the continuity equation using superposition in time and space and introduced three cases for the control volume drained: 1) for the entire reservoir volume (CRMT); 2) for the volume drained by each producer (CRMP); 3) for the volume drained by each producer and injector pair (CRMIP). The CRM as presented by Sayarpour (2008) minimizes the Average Relative Error or Mean Average Error (ARE or MAE) as its objective function. The ARE (or MAE) is defined as:

$$ARE = MAE = min\left\{ \frac{\sum\limits_{n=1}^{Ndata} \left| \frac{Q_{obs} - Q_{est}}{Q_{obs}} \right|}{N_{data}} \times 100 \right\} \quad (2.18)$$

The MAE measures the absolute value of the errors between the simulated and the calculated rates and averages them. It has been reported to be the most suitable to time-related data and continuous variables. Also it is a linear score which treats all errors with equal weight. In other words it is considered as a natural measure to the error (Willmott and Masuura, 2005). The mean square error is defined as:

$$MSE = \sum_{n=1}^{N_{data}} \frac{(Q_{obs} - Q_{est})^2}{n} \times 100 \quad (2.19)$$

In this work these two criteria will be compared when used for the CM and CRM optimization routines. In addition only CRMT and CRMIP will be implemented since CRMP was replaced with CRMIP in Sayarpour (2008).

2.3.1 CRMT (tank model)

In the tank representation the producers and the injectors are summed into one pseudo-injector and one pseudo-producer. The governing equation is written in terms of the field (summed) parameters.

$$q_F(t_k) = q_F(t_{k-1}) e^{-\frac{\Delta t_k}{\tau_F}} + \left[f_{ij} I_F^k \left(1 - e^{\frac{\Delta t_k}{\tau_F}} \right) \right] \quad (2.20)$$

An MAE objective function is written to minimize the difference between the actual field rate at a given time step and the rate computed using Eq. 2.20 to obtain the field time constant, τ_F. This value will be used later to initialize the injector-producer model i.e. the CRMIP values for the time constants.

2.3.2 CRMIP (Injector, producer model)

For the CRMIP model, production at any time can be written as:

$$\hat{q}_j(t_n) = q_j(t_0)e^{-\frac{t_n - t_0}{\tau_{ij}}} + \sum_{i=1}^{N_{inj}} \left[f_{ij} \left(i_i(t_n) - e^{\frac{\Delta t}{\tau_{ij}}} i_i(t_n - 1) \right) \right] - \tau_{ij} \left(1 - e^{\frac{\Delta t_n}{\tau_{ij}}} \right) \left[\sum_{i=1}^{N_{inj}} f_{ij} \frac{\Delta i_i^n}{\Delta t_n} + J_i \frac{\Delta p_{wf,j}}{\Delta t_n} \right] \tag{2.21}$$

where f_{ij} represents the fraction of injection that is directed from injector i toward each producer at steady-state, i.e it is the percentage that the i(th) injector contributes towards the j(th) producer. The minimization algorithm to obtain the unknowns for this model will be presented in section 3.2

2.3.3 Modifications and similar models

There have also been some modifications to these simple models and especially to the CRM model. Liang et al. (2007) introduced an optimization model along with a power law model that accounts for oil production in terms of the cumulative injected water. A non-linear optimization method was then presented to maximize the produced oil and predict the future production.

Tiab and Dinh (2008) introduced a different approach for these models. By changing the permeability at the well pairs and holding injection well permeability values constant, a match can be found between the actual and modeled rates. The main difference here is that Tiab and Dinh (2008) used the bottomhole pressure to obtain the production well permeability values. Another assumption made was that the injection should reach pseudo-steady state at the end of each time step.

Weber et al. (2009) used the CRM model to maximize the net production value and to optimize the inje*ction schedule. They suggested identifying the data points within more than two standard deviations as outliers. The method of compensating the outliers is to replace it with the average of five consecutive data points.

2.4 Current work

A CRM model code has been implemented as a part of this project. The code developed and results will be presented in chapters 3 and 4. Two minimization methods were used: MAE (Mean Average Error) and a traditional minimization based on the mean square error, MSE. In this approach the weights of the differences will be squared, so small values will be much smaller and vice versa. Thus the MSE is a more stringent measure for the optimization. From Figure 2.2 we can see that the MSE takes more iterations to obtain at least the same value of R^2 than the MAE. These two approaches may have an impact on the optimization process speed. Thus both approaches were tried in this work and the results are reported in terms of the number of the iterations to obtain the same residual error. Both methods took almost the same time to run but the number of iterations in the MSE is higher, thus the MAE converges faster.

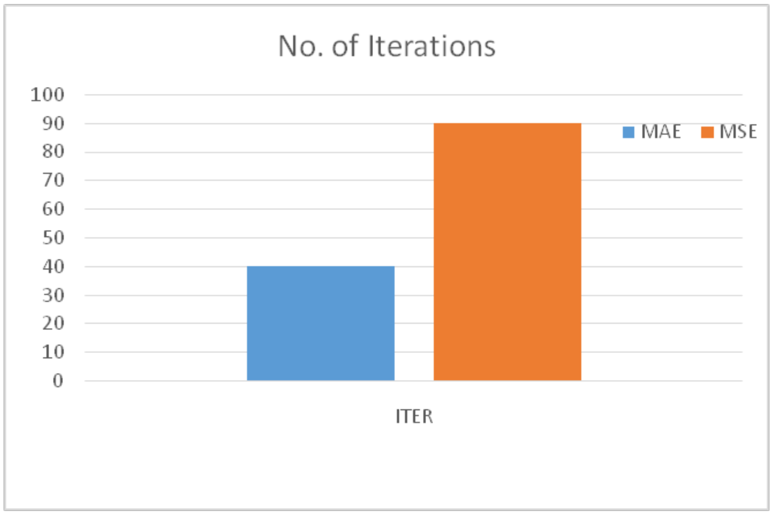


Figure 2.2: The Difference in number of iterations between the MSE and the MAE for the CRM method

CHAPTER 3
CODING WORK AND PRELIMINARY RESULTS

3.1 RM coding work (R and Matlab$^{®}$)

The RM was programmed using the R-Statistical Package and the code allows a user-defined number of time periods as well as a user-defined number of producers and injectors in the model. All results matched values provided in Albertoni et al. (2003) for similar cases and an example is shown in Figure 3.1. Utilizing R provided a fast and efficient method for obtaining these results outside the spreadsheet options typically used. The complete R- code for implementation of RM is shown below:

```
PRD=read.csv(file="PRD.csv", header=F)

INJ=read.csv(file="INJ.csv",header=F)

INJCOV=matrix(nrow=ncol(INJ), ncol=ncol(INJ))

for (i in 1:ncol(INJCOV))

for (j in 1:ncol(INJCOV))

{INJCOV[i,j]=cov(x=INJ[,i],y=INJ[,j])}

INJPRCOV=matrix(nrow=ncol(INJ),ncol=ncol(PRD))

for (j in 1:ncol(INJPRCOV))

for (i in 1:nrow (INJPRCOV))  {INJPRCOV[i,j]=cov(x=INJ[,i],y=PRD[,j]);3

INJPRCOV[is.na(INJPRCOV)] <- 0}

for (j in 1:ncol(BETAS.NOMISS)) {BETAS.NOMISS[,j]=INJPRCOV[,j]%*%solve(INJCOV)}

write.csv(BETAS.NOMISS, file="BETAS.csv")
```

A sample of the input data format and the output format is shown in section 4.2. The β values for each well are shown in Figure 3.1 and show a good match between the R code and what was found in an Excel-spreadsheet provided by Albertoni (2006). The running time was ∼ .14-.15 seconds for the case of 16 producers and 25 injectors. The code is generalized to deal with any case size with no modifications needed. Cases with larger numbers of producers and injectors have been tested. Run times are shown in Figure. 3.2 and show that the running time is relatively fast. Note that the biggest time sink is in data preparation and cleaning; the data needs to be in a specific format and the data needs to be checked for any missing values as they will cause problems obtaining the solution. This data preparation step needs to be done regardless of the tool being used. A technique to account for missing data will be described as part of this thesis. Exactly equal injection and production rates cause singular matrices. To prevent this from occurring data records can be altered by adding a small amount of white noise to each record that will prevent the matrices from approaching singularity as suggested by Al-Yousef (2006). As a result it can be concluded that using R-code to process the RM method is suitable in terms of the processing time and the only time sink will be the data preparing and processing before running.

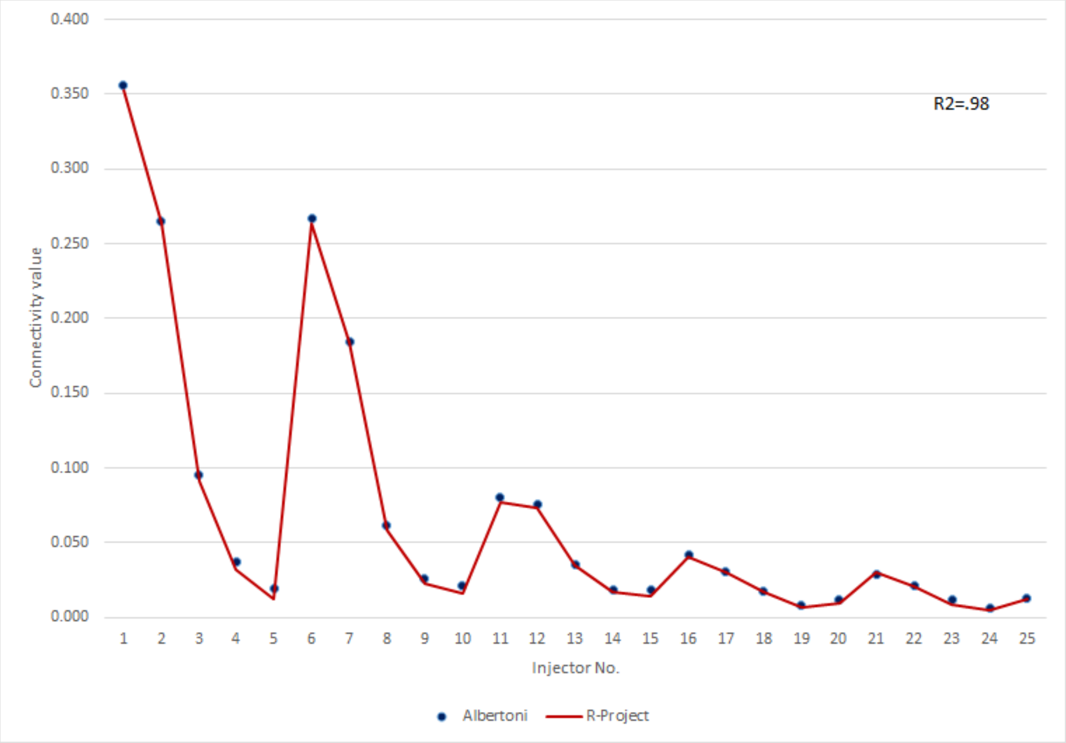


Figure 3.1: Validation between this work using R-project and Albertoni (2003) work in Excel

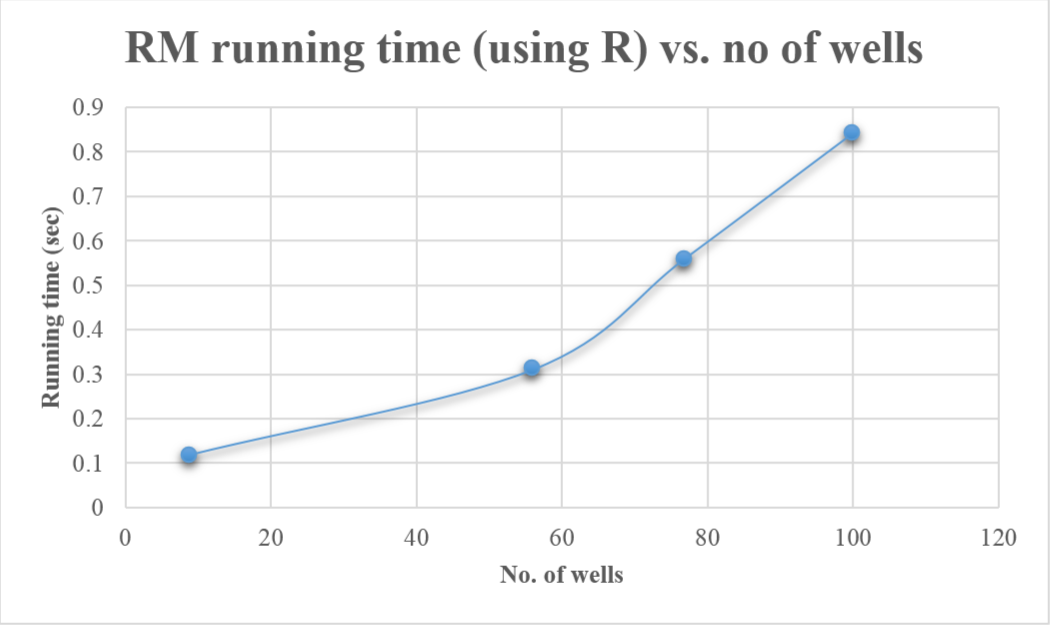


Figure 3.2: The relationship between the RM running time in R and the number of the wells

3.2 Coding work for both the CRM and CM

3.2.1 Coding work for both the CRM and CM using R-project

The system of equations to obtain the unknown parameters in Eq. 2.18 for well j is shown in Eqn 3.1. There are I+J+1 (I=No. of injectors, J=No. of producers) variables in the unknown vector for this non-linear problem. The unknowns vector shown is λ_{pj}, λ_{ij} and ν_{kj} and there are unknowns that influence the covariance values in the matrix and in the righthand side vector. These are τ_{pj}, τ_{ij} and τ_{kj}

$$\begin{pmatrix} \sigma_{pp} & C^T_{pp-I} & C^T_{pp-BHP} \\ C_{pp-I} & C_{II} & C_{I-BHP} \\ C_{pp-BHP} & C^T_{I-BHP} & C_{BHP-BHP} \end{pmatrix} \begin{pmatrix} \lambda_p \\ \lambda_{ij} \\ \nu_{kj} \end{pmatrix} = \begin{pmatrix} \sigma_{pp-q_j} \\ C_{I-q_j} \\ C_{BHP-q_j} \end{pmatrix} \tag{3.1}$$

The variables in Eq. 3.1 are:

- σ_{pp}: is the primary production variance a single value computed for each producer. The primary production vector values can be obtained using:

$$\lambda_{pj} q_j(n_0)\, e^{\frac{-(t-t_O)}{\tau_{pj}}} \tag{3.2}$$

and the initial estimates for the values for λ_{pj}, τ_{pj} were explained in section 2.2.

- C_{pp-I}: is the covariance between the filtered injection values calculated using Eq. 2.15 and the primary production values (Eq. 3.2. There are I values in this vector.
- C_{pp-BHP}: is the covariance between the bottomhole pressure term in Eq. 2.16 and the primary production for producer j. There are J values in this vector.
- C_{II} is the filtered injection covariance matrix, which can be calculated using Eq. 2.15. There are I×I values in this matrix.
- C_{I-BHP}: is the covariance between the filtered injection values (Eq. 2.15) and filtered bottomhole pressure values (Eq. 2.16). There are I×J values in this matrix.
- $C_{BHP-BHP}$: is the variance of the filtered bottomhole pressure values for each producer (Eq. 2.16) and there are J×J values in this matrix.
- σ_{pp-q_j}: is the covariance between the primary production value and the production values for producer j. This is a single value.
- C_{I-q_j}: is the covariance between the filtered injection values (Eq. 2.15) with the production values for producer j. There are I values in this vector.
- C_{BHP-q_j}: is the covariance of the filtered bottomhole pressure (Eq. 2.16) and the production values for producer j. There are J values in this vector.

The coefficient vector in the matrix was explained in section 2.2. The above terms need to be assembled into one master matrix. Figure. 3.3 shows the final master matrix with the expected dimensions.

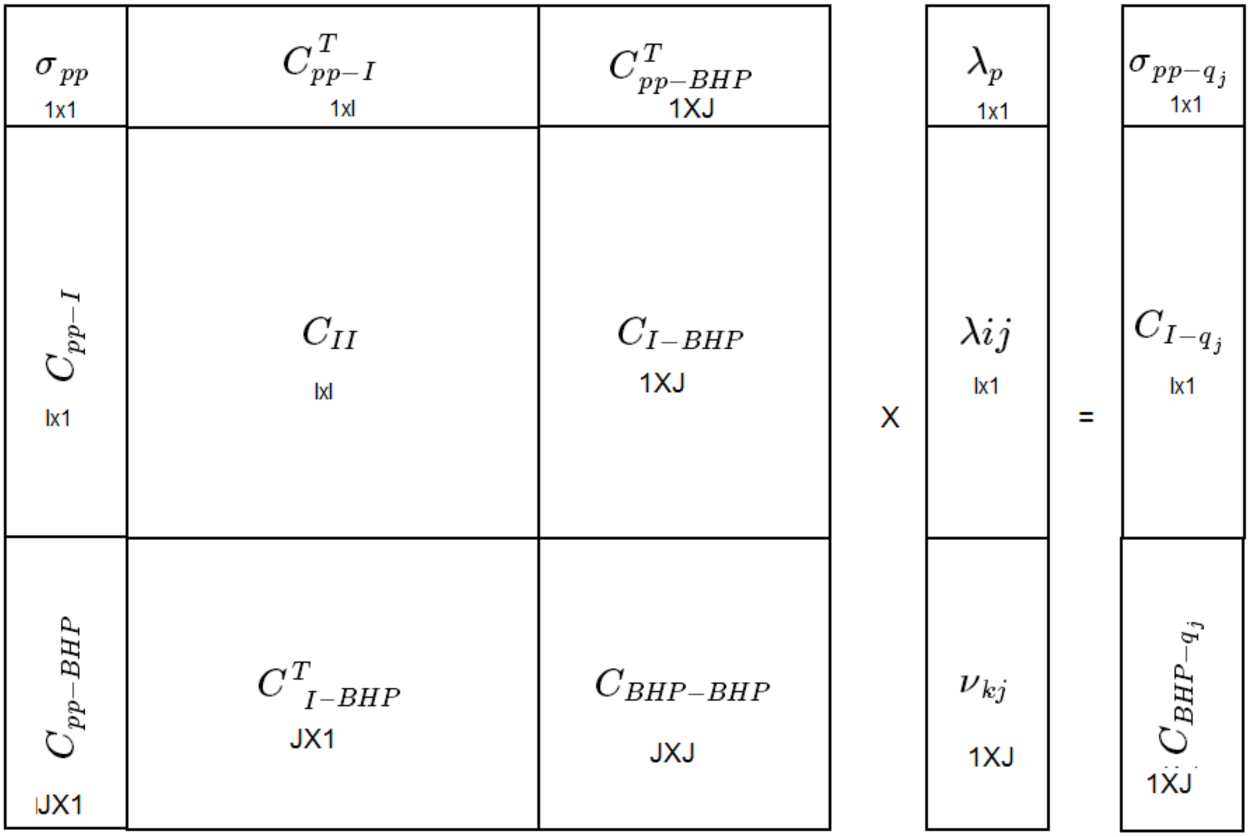


Figure 3.3: The final matrix for the CM model

A flowchart of the optimization process is shown in Figure. 3.4 and this flowchart is how the code was implemented in R-Project. Both CRM and CM were also coded using the R-project statistical package. The first case tested was a synthetic case with five injectors and four producers where pressure, injection and production data for 100 time intervals was generated using the IMEX black oil reservoir simulator. The optimization algorithm that was used was BFGS (Broyden-Fletcher-Goldfarb-Shanno) which is an approximation for Newton's method (R-documentation, 2013). The following issues were observed:

- The RM matrix calculations were reasonably fast and the results matched the ones calculated using an Excel sheet from Albertoni et al. (2003).
- The CM optimization gave a good match between the observed and the calculated values of the production values i.e. R^2=.98. Figure 3.5 shows the optimization results plotted against the actual production along with the R^2 value calculated.

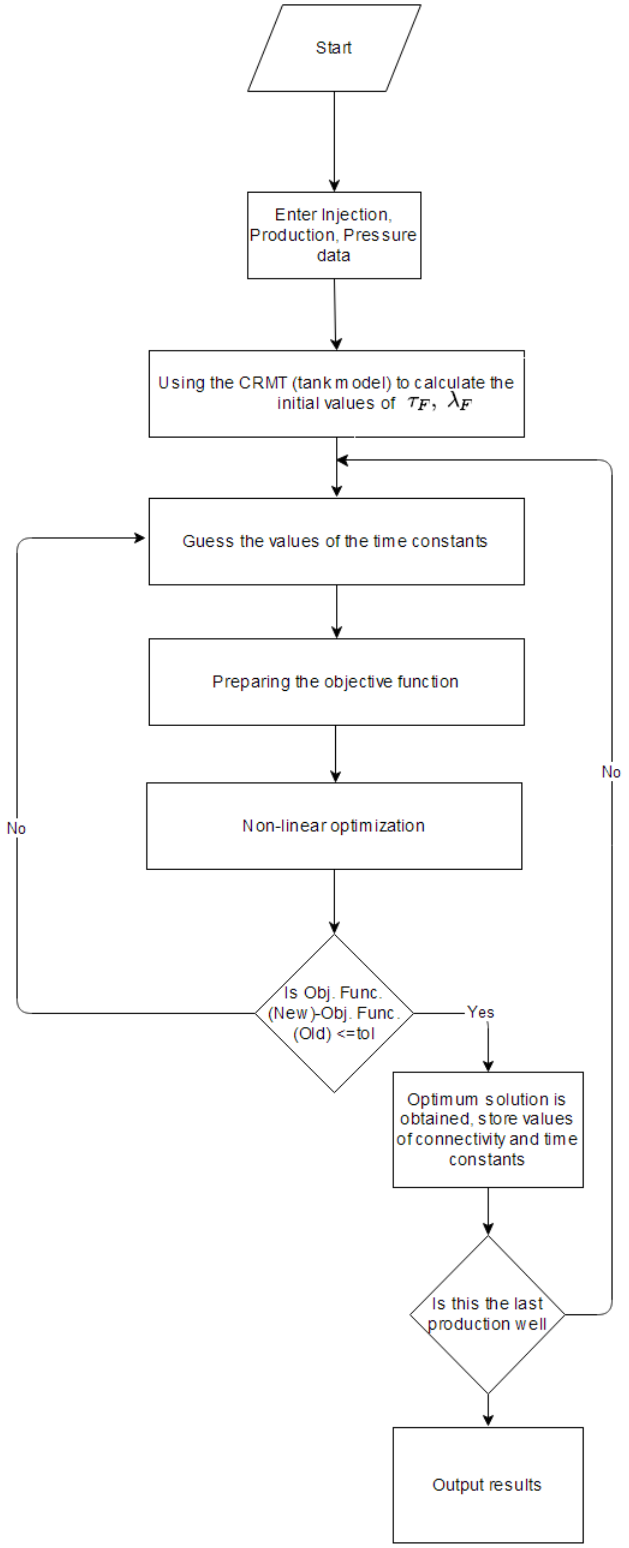

Start
Enter Injection, Production, Pressure data
Using the CRMT (tank model) to calculate the initial values of τ_F, λ_F
Guess the values of the time constants
Preparing the objective function
Non-linear optimization
Is Obj. Func. (New)-Obj. Func. (Old) <=tol
Yes
No
Optimum solution is obtained, store values of connectivity and time constants
Is this the last production well
No
Output results

Figure 3.4: The flow chart of the procedure in R Optimization

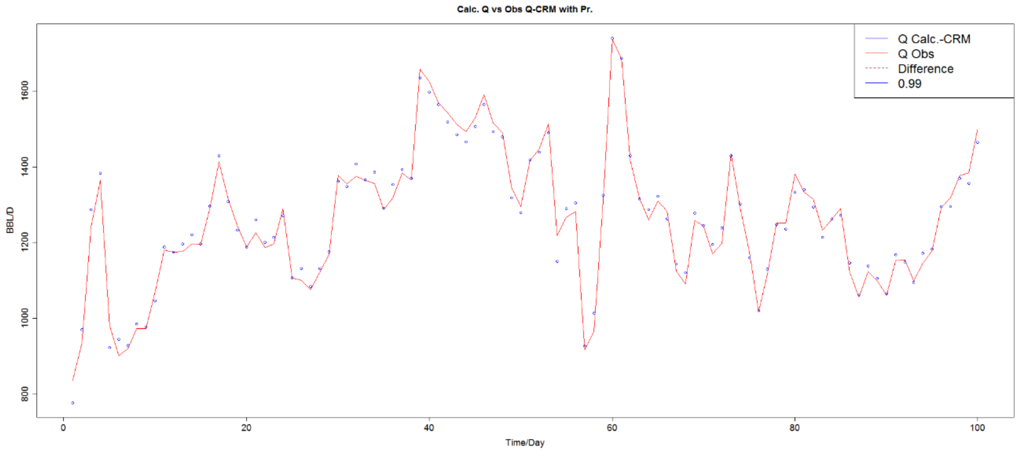


Figure 3.5: Optimization results plotted against the real production along with the R^2 value calculated for well no. 1

3.2.2 Coding work for both the CRM and CM using Matlab®

The R-package was used to perform the CRM optimization but took an extraordinary amount of time to converge: a session for 5 injectors and 4 producers took almost an hour to get to a minimum value of the objective function ($MSE \leq tol$). Conclusions from the CM and CRM models coded using R is that the data preparation and cleaning effort will be the same as for the RM model but the main issue is the time to reach convergence. Figure. 3.6 shows a comparison between the CRM, CM and RM in terms of running time in R. Several optimization algorithms were tested to try to improve the convergence speed (BFGS, Nelde-Mead, L-BFGS-B) in R and they gave similar results in terms of running time. Consultants from the LSU Statistics Department stated that the R-package lacks the computational power required for optimization processes because it is mainly designed to handle statistical problems. The conclusion is that it is not recommended to implement the CRM or CM methods using the available R-project optimization packages as these packages are not optimized enough to handle such optimization problems. This leads to the conclusion that R-project is suitable to implement the RM models only.

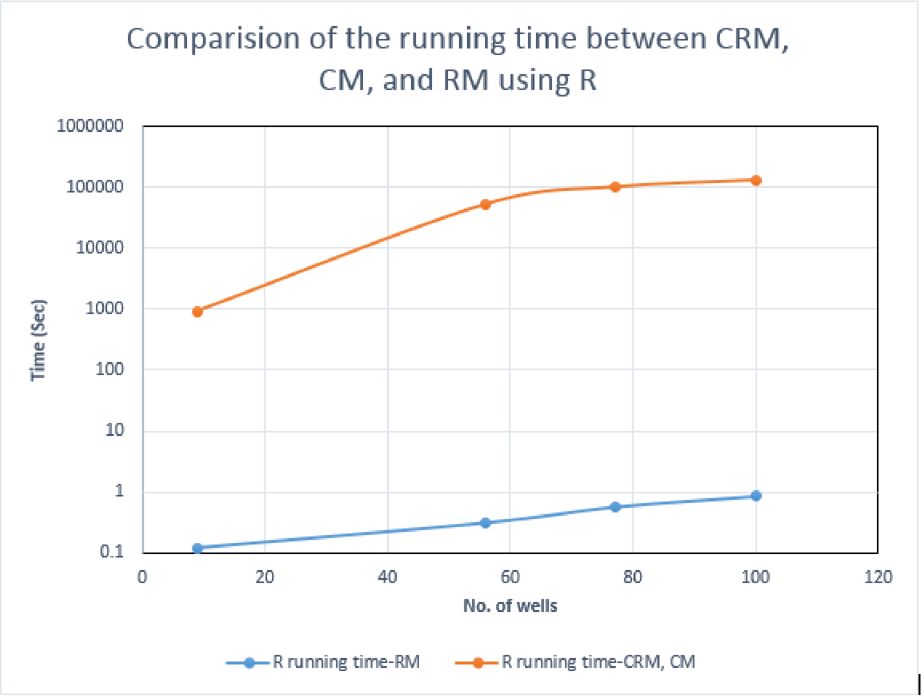
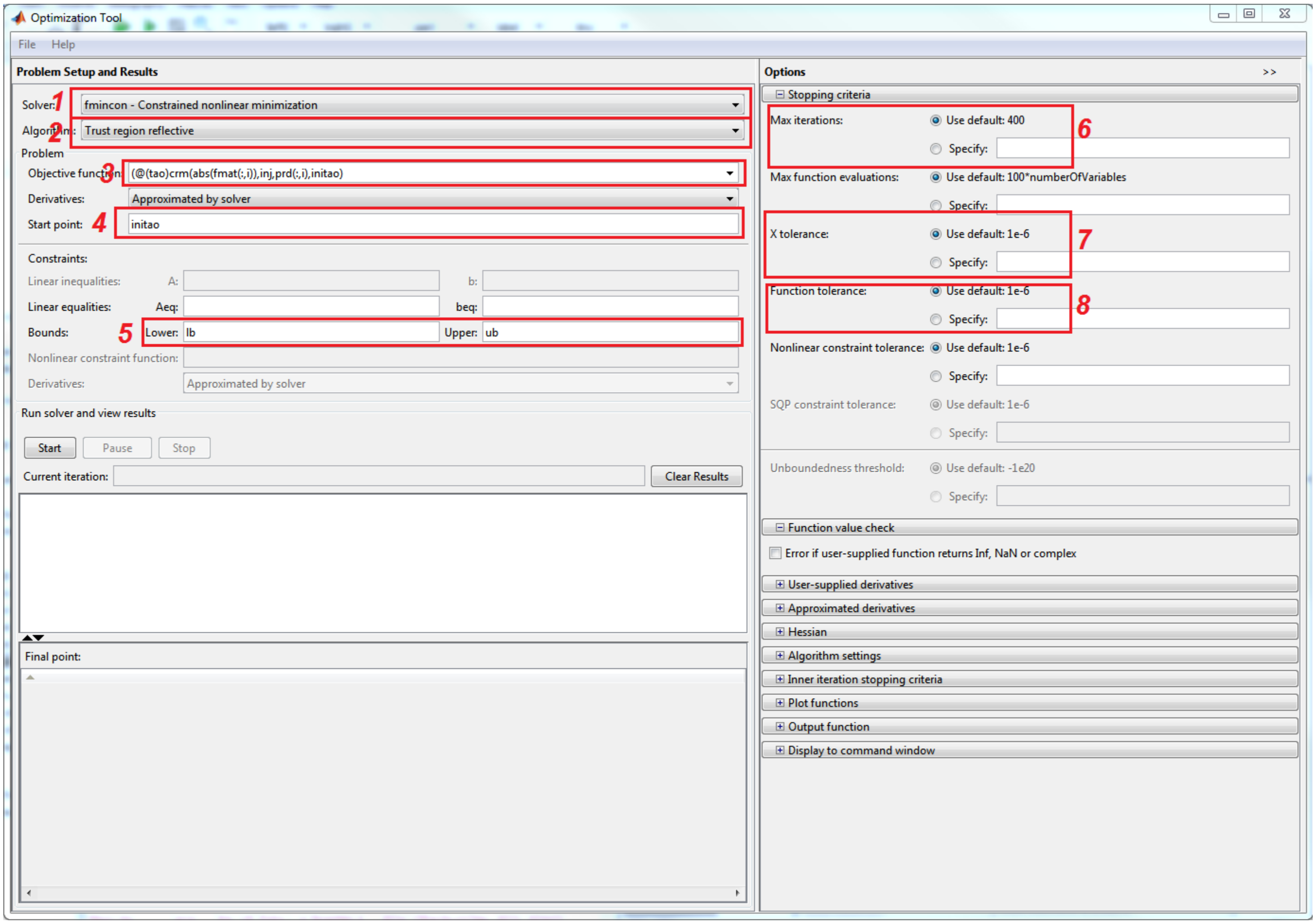


Figure 3.6: A comparison between CRM, CM and RM models running time in R

3.2.3 Matlab® optimization toolbox® work

The Optimization toolbox® in Matlab® was utilized to run the same optimization problem and this implementation reduced the time needed to reach an optimum value of the objective function significantly. Tables 3.1 and 3.2 show comments comparing the R-Project code and Matlab® for CRM, CM, and RM implementations in terms of running time and implementation effort. Figure 3.7 shows the optimization box window from the Matlab® interface. The important inputs have been highlighted and are explained as follows:

1. Solver: Several solvers can be found in Matlab® (depends on the version of the software). The solver that was used in this work was Constrained nonlinear minimization as the problem is a constrained minimization (constraints in λ_{ij} and τ_{ij}).
2. Algorithm: The minimization algorithms can be selected from the panel shown. The selection criteria in general depends on whether the problem derivative is provided or not.



Solver: fmincon - Constrained nonlinear minimization

Algorithm: Trust region reflective

Problem

Objective function: (@(tao)crm(abs(fmat(:,i)),inj,prd(:,i),initao)

Derivatives: Approximated by solver

Start point: initao

Constraints:

Linear inequalities: A: b:

Linear equalities: Aeq: beq:

Bounds: Lower: lb Upper: ub

Nonlinear constraint function:

Derivatives: Approximated by solver

Run solver and view results

Current iteration:

Final point:

Max iterations: ● Use default: 400 ○ Specify:

Max function evaluations: ● Use default: 100*numberOfVariables ○ Specify:

X tolerance: ● Use default: 1e-6 ○ Specify:

Function tolerance: ● Use default: 1e-6 ○ Specify:

Nonlinear constraint tolerance: ● Use default: 1e-6 ○ Specify:

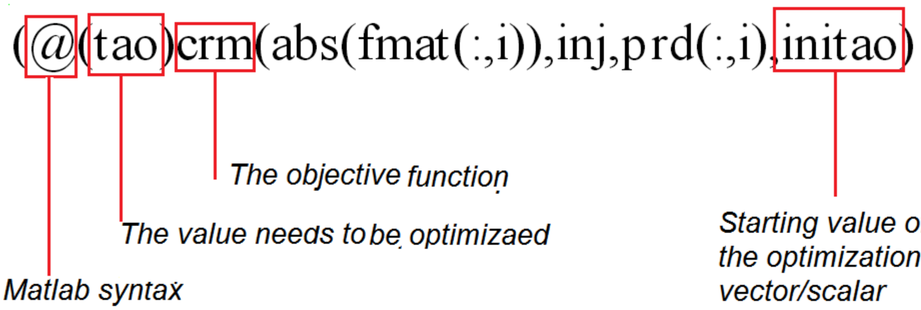
SQP constraint tolerance: ● Use default: 1e-6 ○ Specify:

Unboundedness threshold: ● Use default: -1e20 ○ Specify:

□ Error if user-supplied function returns Inf, NaN or complex

Figure 3.7: Matlab optimization toolbox ® main entries for the CRM, CM model

3. Objective function: The function that needs to be optimized. The way this function should be entered is shown in Figure. 3.8 for the CRM.



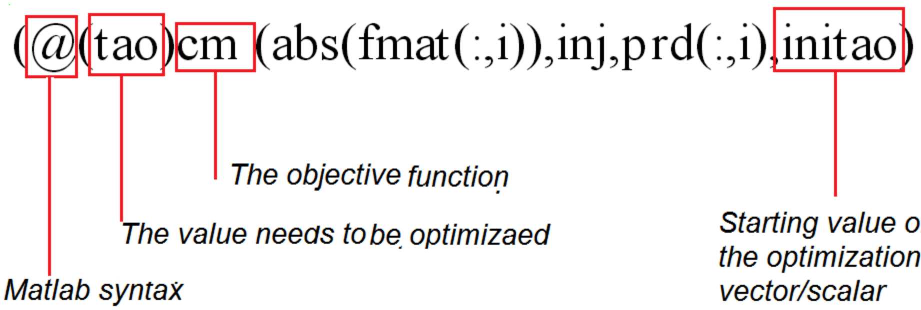
```
(@(tao)crm(abs(fmat(:,i)),inj,prd(:,i),initao)
```

Figure 3.8: Matlab® optimization toolbox® objective function syntax entry for the CRM optimization

4. Start point: The starting (initial) values of the optimization vector or scalar.
5. Bounds: The lower and upper bounds for the unknown values are to be entered here. In the case of the λ_{ij} values, the lower bounds are 0 and the upper bounds should be 1. For the τ_{ij} values, a lower bound of 0 is all that is required.
6. Max iteration: The maximum number of iterations is set to 400 by default.
7. X tolerance: Specifies the termination tolerance for X (X is the final values of the optimization vector, Eq. 2.18)
8. Specifies the termination tolerance for the objective function value (Eq. 2.18)

   Items 6, 7 and 8 represent the stopping criteria that Matlab® uses for the optimization of the objective function (Eq. 2.18). The flowchart for these steps is shown in Figure. 3.11.

   The only difference in the CM method syntax is the name of the function that Matlab® calls. Figure. 3.9 shows the syntax for the CM method.



```
(@(tao)cm (abs(fmat(:,i)),inj,prd(:,i),initao)
```

The objective function
The value needs to be optimizaed
Matlab syntax
Starting value of the optimization vector/scalar

Figure 3.9: Matlab® optimization toolbox® objective function syntax entry for the CM optimization

Figure. 3.10 shows the results for the same problem from section 3.5 (4 producer× 5 injector) graphically. The value for R^2 is slightly smaller (in R yielded $R^2 = .998$ while Matlab® yielded 0.996) and it is believed that the decrement is not significant comparing with the difference in the time needed to perform the optimization. Increasing the maximum iterations function iteration to more than 400 might improve the value of R^2.

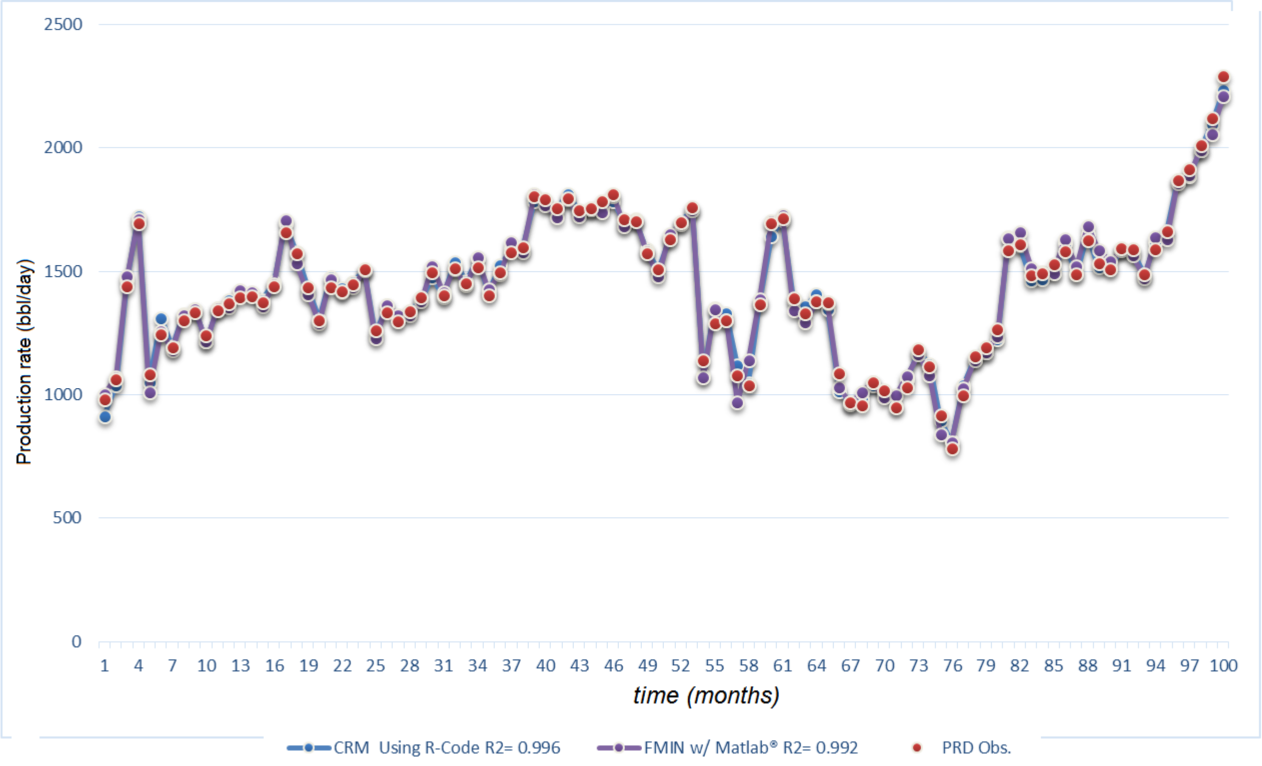


Figure 3.10: Results of Matlab® optimization comparing with R-Proj, Qobs.

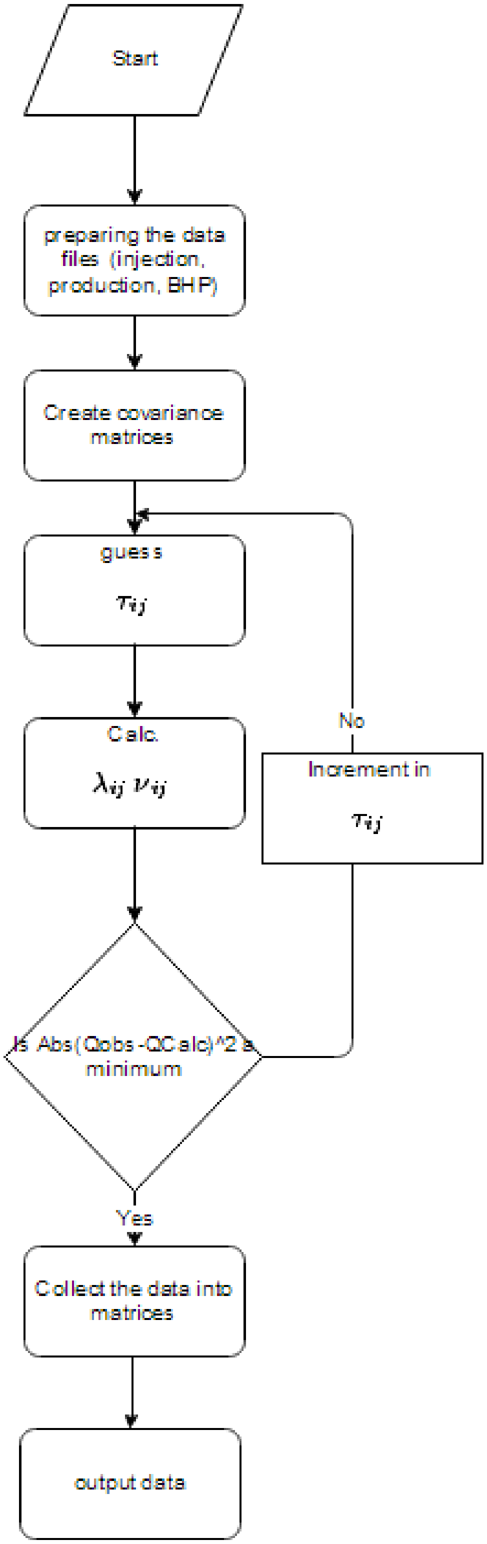

Start
preparing the data files (injection, production, BHP)
Create covariance matrices
guess
τ_{ij}
Calc.
λ_{ij} ν_{ij}
Is Abs(Qobs-QCalc)^2 a minimum
No
Increment in
τ_{ij}
Yes
Collect the data into matrices
output data

Figure 3.11: Matlab® Optimization toolbox implementation flowchart for the CRM and CM models

Table 3.1: R-Project usage observations

CM, CRM Implementation	Iteration	Function Evaluations	Running time	RM Implementation	Running time
One time implementation and the code was generalized. More than one non-linear optimization method was tested (Optim, Optimx, nlm) and they resulted good matches but with long run times. In this implementation we have two time sinks, the data preparation and cleaning and the optimization time required.	120	130	60 min	Less effort than CRM and CM. No optimization tools need to be used. The code is generalized to handle any case size. The only time sink is the data cleaning and preparation and handling the missing data. The run times are much shorter than the CRM, CM.	1-2 mins

Table 3.2: Matlab® usage observations

CM and CRM Implementation	Iteration	Function Evaluations	Running time	RM Implementation	Running time
A generalized code to handle any case size. The Optimization toolbox® was used. The functions fmincon and fminunc were tested, run times were significantly less than R-Project implementation with more function evaluations. Results were used to predict the same values and yield good correlation.	94	4000	10 min	No issues with RM in R-Project. Matlab® code was generalized. The same data preparation effort is required. The only difference here is that Matlab® offers many aids to represent the data and is more flexible to manipulate the variable values. The run times were 10% shorter than R- Project	1-2 mins

3.2.4 The mechanics of picking the starting values for the time constants

The optimization process for both the CM and the CRM is sensitive to the starting values for the time constants in terms of the function convergence speed. Thus there should be a reasonable guess that can save the process time and memory needs. For the CRM model, Sayarpour (2008) suggested that the production and injection data should be added into one pseudo-producer and one pseudo-injector (a tank model) and then the CRM model should be run to obtain an estimate of the time constants. In Figure. 3.12 the time constants are shown for the case of the tank model as well as the final values for the time constants obtained after convergence. Also shown is the tank model values divided by the number of injector-producer pairs. The figure shows that the usage of the tank model (CRMT) to obtain the starting values of the time constants provides an upper limit for the converged time constants. Dividing the tank model value by the number of injector-producer pairs provides a lower limit. For further investigation the relation between the initial guess of the time constant and the final error has been investigated by taking multiple values and computing the corresponding error. It has been found that there is an optimum value that yielded the least error for this problem. Figure 3.13 shows this trend and the optimum value location. For balanced water flooding the time constant value usually is low as will be explained in Chapter 4. For a better start value for the time constants, it is suggested that the maximum value to start with is:

$$\tau_{initial} = \frac{Lower\ bound + Upper\ bound}{4} \tag{3.3}$$

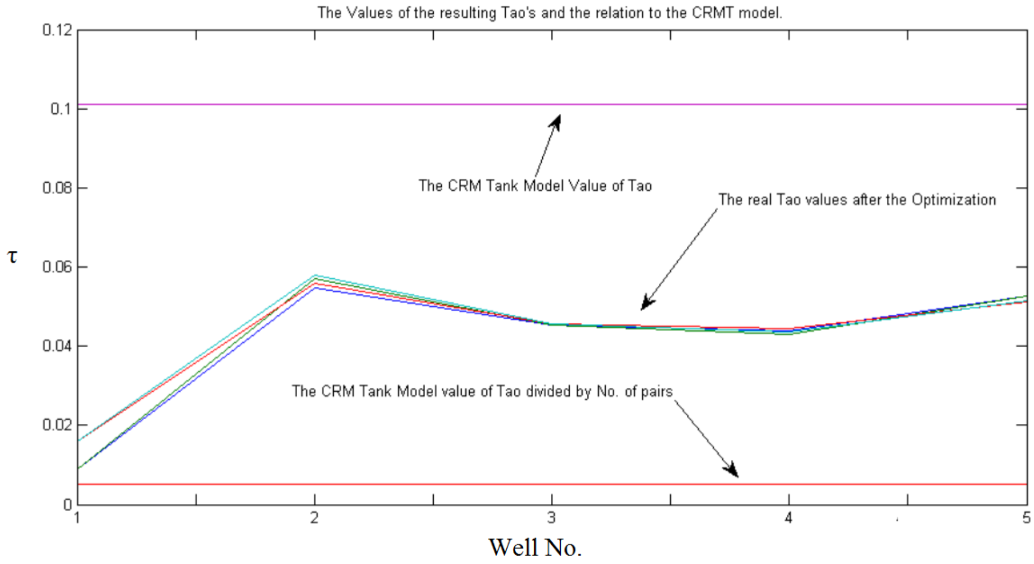


Figure 3.12: The upper and lower limits for the time constants for the case of the CRMT (tank model)

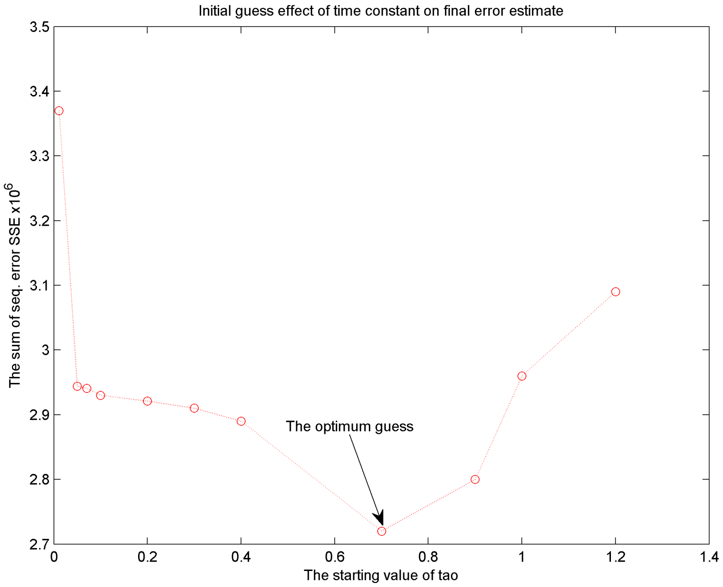


Figure 3.13: Effect of the starting value for the time constant on the final error SSE

3.2.5 Issues with using Matlab® optimization toolbox® to solve CRM, CM models

Although the procedure of using the optimization toolbox® in Matlab® yielded good results, some values for the injector-producer connectivity were found to be either negative or greater than one and some values of the time constants were found to be negative. These values do not have physical meaning. This is another motivation that it could be a good idea to implement a different procedure to solve the problem. But first it is important to test whether there are local minima that the connectivity values or the time constant values could converge to. In order to answer this question the value for the SSE (Sum of Squared Errors) of the function as λ_{ij} for one injector-producer was increased from 0 to 1 (holding other parameters constant). Figure 3.14 shows the that there is only one value (for this problem) that the function converges to. Furthermore, the value for the τ_{ij} vs. SSE was investigated (holding other parameters constant). The result is shown in Figure 3.15 where there is again a single minima.

The next step was to implement a procedure for an appropriate solution sequence. The suggested procedure is shown in Figure 3.17 for the CM and the steps are as follows:

1. Data input (injection and production rates, wells locations, pressure data if available)
2. Data cleaning and preparation. Start with production well 1.
3. Guess values for τ_{pi}, τ_{ij} and τ_{kj} (Use Eq. 3.3)
4. Calculate the initial values for λ_{ij} and λ_{pj} using Eq. 3.1
5. Predict the production rates from steps 3 and 4
6. Calculate a value for the SSE
7. Do one λ_{ij} increment using $\lambda_{ij} = \lambda_{ij} + SI \times \Delta\lambda_{ij}$ (SI is a sign changing factor default=either 1 or -1)
8. Predict the production rates and the new values for λ_{ij} using the time constants from step 3
9. Calculate a new value for SSE
10. If the SSE from step 9 is greater than the SSE from step 6 there will be a sign change in the calculation of λ_{ij} (SI=$-$SI)
11. Iterate between steps 7 and 10 until the difference between the new value for the SSE and the old value for the SSE is less than or equal to the tolerance (10^{-6} for this work)
12. The value for this SSE is saved for further comparisons
13. Increase the values for τ_{ij} by a factor (0.1 for the results presented here)
14. Repeat steps 4-12

15. Compare the SSE value from steps 9 and 12 until the difference is less than the tolerance level.
16. Again save the SSE value
17. Increase τ_{kj} by a factor and perform steps 4-16 until the difference in SSE values between steps 9 and 16 is less than the tolerance.
18. Save the SSE value
19. Increase τ_{pj} by a factor and perform steps 4-17 until the difference between SSE values is less than the tolerance
20. Repeat steps 3-19 for each injector
21. Calculate the production rates and R^2 values for the current well
22. Repeat steps 3-21 for each production well
23. Data representation (well connectivity values graph and the predicted values plot matrix)

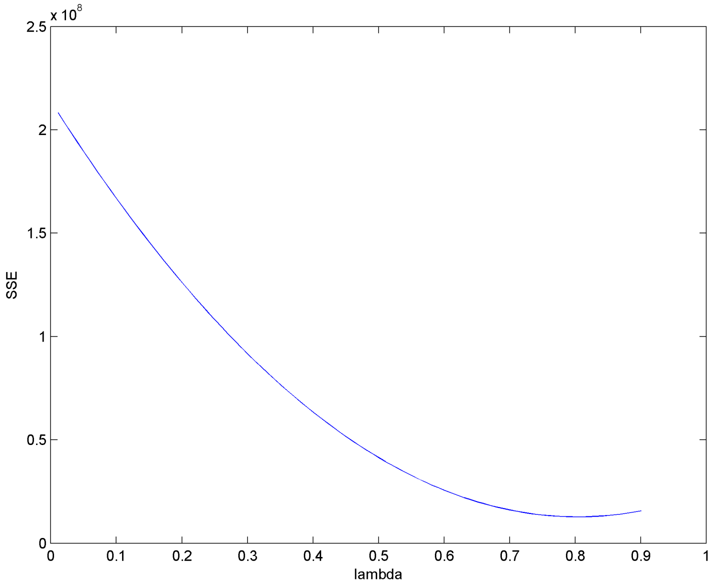


Figure 3.14: The function convergence with the change of connectivity value λ_{ij} for well no. 1

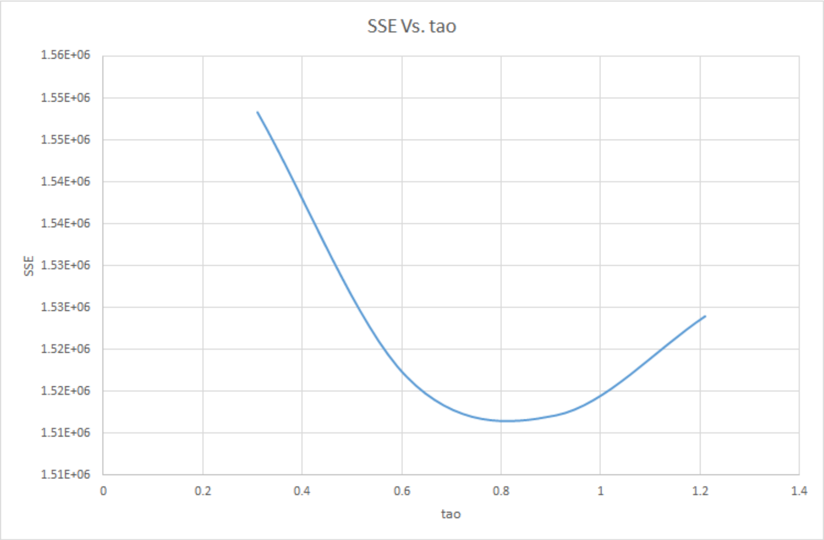


Figure 3.15: The function convergence with the change of time constant τ_{ij} for well no. 1

A similar procedure for each producer has been implemented to solve the CRM model and is shown in Fig. 3.16 with some differences, The steps are:

1. Data input
2. Data cleaning and preparing. Start with production well 1
3. Guess τ_{ij} values (using Eq. 3.3) and guess f_{ij} using the reciprocal of the distance between producer j and injector i as suggested by Sayarpour (2008) i.e. $f_{ij} = \frac{1}{distance(i,j)}$
4. Predict the production values using Eq 2.21
5. Calculate a value for the SSE and store it for further comparison
6. Increase f_{ij} values by a factor (0.01).
7. Calculate a new value for SSE and store it for further comparison
8. if the new value of SSE is greater than the old value of the SSE there will be a sign change (SI=-SI)
9. Save the last computed SSE for further comparison
10. Repeat steps 5-8 until the new value of SSE - the old value of SSE is less than or equal to the tolerance of SSE
11. Increase τ_{ij} values by a factor (0.1).

12. Predict the production values
13. Calculate the new SSE and compare it with the stored SSE from step 5
14. If the new value of SSE is greater than the old value of the SSE there will be a sign change (SI= –SI)
15. Repeat steps 11-13 until the new value of SSE - the old value of SSE is less than or equal to the tolerance.
16. Repeat steps 3-15 for each production well
17. Data representation (Production plots matrices)

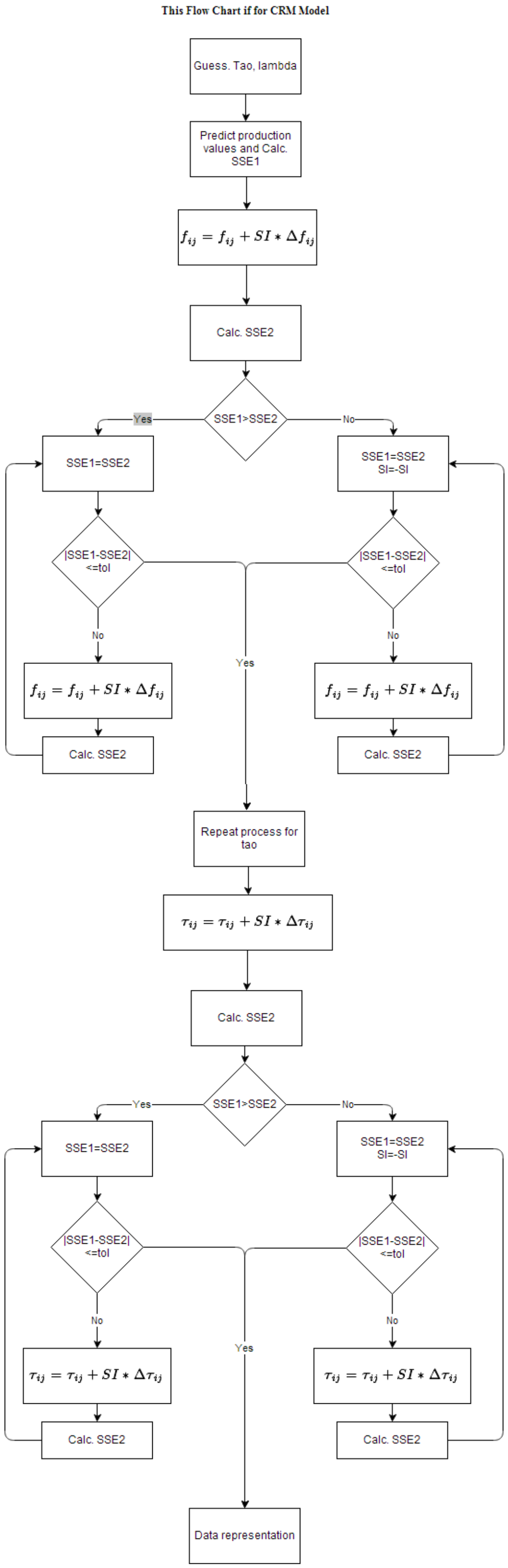

This Flow Chart if for CRM Model
Guess. Tao, lambda
Predict production values and Calc. SSE1
$f_{ij} = f_{ij} + SI * \Delta f_{ij}$
Calc. SSE2
SSE1>SSE2
Yes
No
SSE1=SSE2
SSE1=SSE2 SI=-SI
|SSE1-SSE2| <=tol
|SSE1-SSE2| <=tol
No
Yes
No
$f_{ij} = f_{ij} + SI * \Delta f_{ij}$
$f_{ij} = f_{ij} + SI * \Delta f_{ij}$
Calc. SSE2
Calc. SSE2
Repeat process for tao
$\tau_{ij} = \tau_{ij} + SI * \Delta \tau_{ij}$
Calc. SSE2
SSE1>SSE2
Yes
No
SSE1=SSE2
SSE1=SSE2 SI=-SI
|SSE1-SSE2| <=tol
|SSE1-SSE2| <=tol
No
Yes
No
$\tau_{ij} = \tau_{ij} + SI * \Delta \tau_{ij}$
$\tau_{ij} = \tau_{ij} + SI * \Delta \tau_{ij}$
Calc. SSE2
Calc. SSE2
Data representation

Figure 3.16: The CRM solution routine flow chart

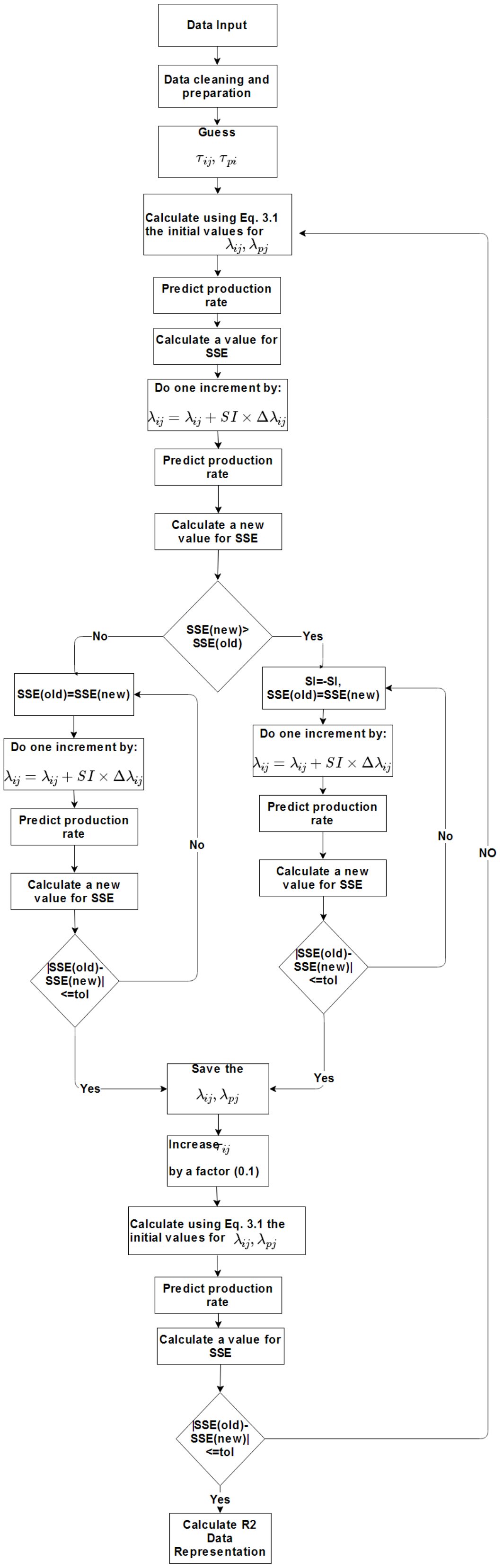

Data Input
Data cleaning and preparation
Guess
τ_{ij}, τ_{pi}
Calculate using Eq. 3.1 the initial values for $\lambda_{ij}, \lambda_{pj}$
Predict production rate
Calculate a value for SSE
Do one increment by:
$\lambda_{ij} = \lambda_{ij} + SI \times \Delta\lambda_{ij}$
Predict production rate
Calculate a new value for SSE
SSE(new)> SSE(old)
No
Yes
SSE(old)=SSE(new)
SI=-SI, SSE(old)=SSE(new)
Do one increment by:
$\lambda_{ij} = \lambda_{ij} + SI \times \Delta\lambda_{ij}$
Do one increment by:
$\lambda_{ij} = \lambda_{ij} + SI \times \Delta\lambda_{ij}$
Predict production rate
Predict production rate
No
No
NO
Calculate a new value for SSE
Calculate a new value for SSE
|SSE(old)-SSE(new)| <=tol
|SSE(old)-SSE(new)| <=tol
Yes
Yes
Save the
$\lambda_{ij}, \lambda_{pj}$
Increase τ_{ij}
by a factor (0.1)
Calculate using Eq. 3.1 the initial values for $\lambda_{ij}, \lambda_{pj}$
Predict production rate
Calculate a value for SSE
|SSE(old)-SSE(new)| <=tol
Yes
Calculate R2 Data Representation

Figure 3.17: The CM solution routine flow chart

3.2.6 The effect of time constant (τ_{ij}) on production rates estimation

The time constant accounts for the pressure dissipation until a response is seen in a production well. So from the equation of the time constant, $\tau = \frac{C_t V_p}{J}$ a large value for time constant would be expected when the system is highly compressible (high GOR), when there is a large pore volume or for low permeability formations i.e. low J. Also a high time constant and pressure dissipation occurs when producers are far from injectors. Figure 3.18 shows the relationship between an assumed time constant for a well (τ_{ij}) and the predicted production rate curve assuming fixed values for the other unknowns. It can bee seen that the larger the time constant, the smoother and more dampened the predicted production will be.

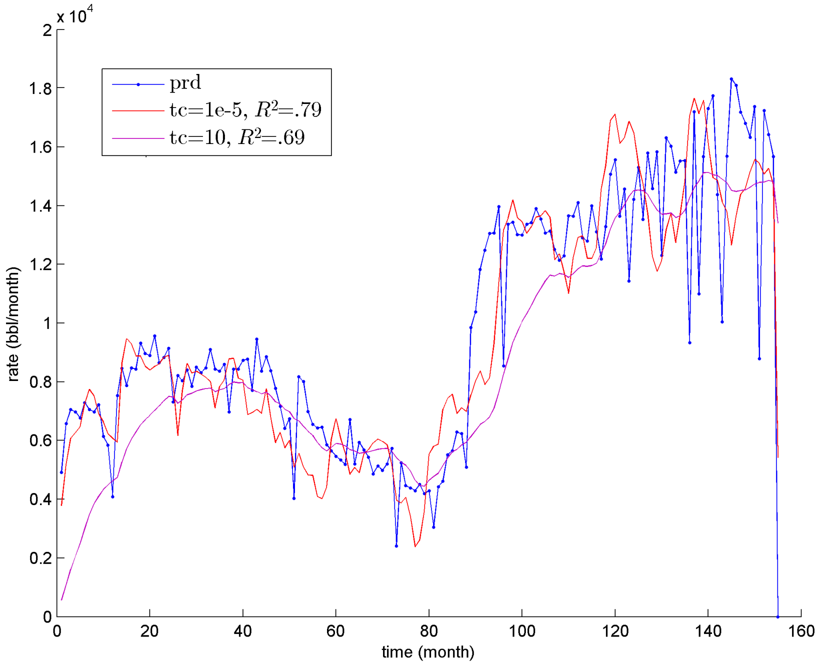


Figure 3.18: The effect of the time constant on the estimated production rate for one well

3.2.7 The effect of the connectivity (λ_{ij}) on the estimated production rate

Connectivity accounts for the amount of the injection that goes toward, or contributes to the production (Albertoni et al., 2003). The larger the connectivity, the more the production rate is influenced by a particular injector, and the less magnitude difference there is between the injection and production curves. The effect of the assumed connectivity values for fixed values of the time constants has been calculated in a 5×4 case (Figure 3.19). It can be seen that as the connectivity value increases, the curve will be shifted up to match the actual production curve. For this case then, almost all of the injection is going toward this producer($\lambda_{ij} \simeq 1$). Variations in the curves were very similar with very little dampening. This indicates that τ_{ij} values are small and the optimization is most influenced by the connectivity values.

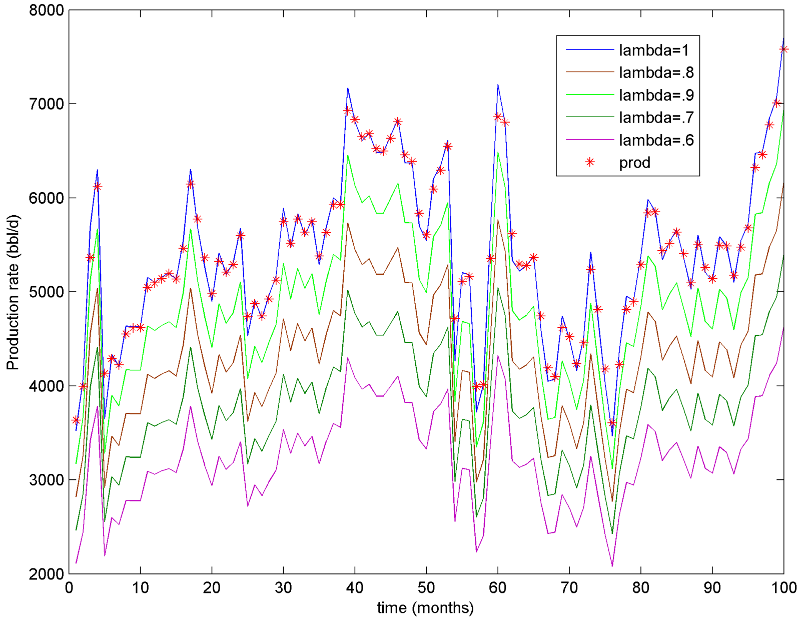


Figure 3.19: Connectivity effect on the estimated production rate for pseudo-producer/pseudo-injector

3.2.8 Graphical representation for connectivity values

The code was extended to be capable of superimposing the values of the connectivity on a layout of all the wells in order to have a graphical illustration for the connectivity trends. This routine has been automated and generalized to work with and adapt to any case size. Figure 3.20 shows a sample representation for this feature in the code, where the arrow direction refers to the relation between the current injector and the producer that it is pointing to, and the arrow length refers to the magnitude of the connectivity value in that direction. This method of representation has been used by Albertoni et al. (2003), Al-Yousif (2006), Sayarpour (2008), Ogunyomi (2009) and Gherabati et al. (2012).

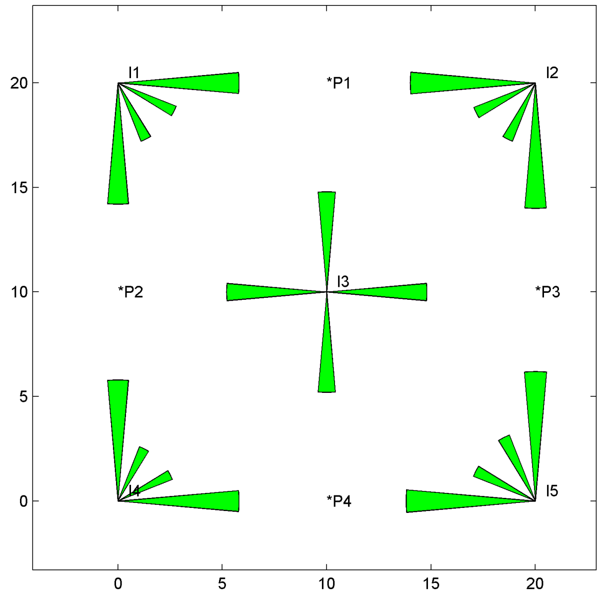


Figure 3.20: Graphical representation of the connectivity values for the CRM model

3.3 Summary

Coding work has been done for all the three models using two coding environments R and Matlab ®. The resulting connectivity values λ_{ij} , and the time constants τ_{ij} were used calculate production data for the same time interval and comparisons were made for both R and Matlab ®. In the following sections detailed observations will be made for each method.

3.3.1 Ease of implementation

RM Model : As mentioned previously the RM model is a multivariate linear regression model (Albertoni et al.,2003). The implementation was simple and easy in both R and Matlab; not that much of effort was needed to develop a fully generalized and automated system that handles all the cases and produces a good graphical representation for most cases.

CM : The implementation of the CM model was more complicated compared to RM. There are several covariance matrices that need to be coded and these matrices were combined into a master matrix that is needed to calculate the required parameters. The optimization process was also long and a bit complicated due to the non-linear nature of the function. Also collecting and sorting out the results was not as direct as for RM.

CRM : The CRM model falls between the CM and the RM in terms of ease of implementation. Because the method to initialize the solution is by inverting the distances, there is no need to calculate the covariance matrices. The optimization was easier than the CM model. Although the function is also non-linear, the terms were less complicated than CM.

3.3.2 Running time

RM : The running time in R and Matlab® for the RM model was not an issue as was explained in section 3.1. The model is a linear model so there is no iteration or substitution.

CM, CRM : Using R, the running time for the CM model was excessive. Although many scenarios and methods were tested to speed the process, run time was still long. One reason is that the procedure has more variables than RM. Another is that these are non-linear problems that contains iterations and substitution finally and the most important thing is that R is not optimized to handle such high computational demand problems (as per the Faculty of Statistics Department at LSU). In Matlab® the time was significantly less than in R using the optimization toolbox® . Even when a simple iteration procedure was used, the run time was much less than in R.

CHAPTER 4
SYNTHETIC CASE APPLICATIONS

The process of code implementation for the three models (RM, CM, and CRM) was described in Chapter 3. Recommendations were made for each coding platform and each model in addition to investigation of some parameter effects. In this chapter, the implemented code will be applied to synthetic case that was described by Albertoni et al. (2003) and others to measure the interwell connectivity values using these three models. Usually in these synthetic case the desire is to test the model with basic settings without any real complications. A homogeneous, isotropic model will be introduced to infer the values of these parameters. The procedure used was that part of the data was used to obtain the connectivity values and the values were then used to predict the production response.

4.1 Application of RM model

4.1.1 4×5 Case

In this case 4 producers and 5 injectors were positioned in an inverted 5-spot injection pattern which is the same as case has been described by Albertoni et al. (2003). Using the CMG IMEX reservoir simulator, production and injection data were created to have a bit of fluctuation in order for these models to work correctly. Also the model is a balanced injection case whereby the total injection rate is equal to the total production rate with no losses and no aquifer support. This means that the balanced RM model can be used (Albertoni et al., 2003). The well pattern is shown in Figure 4.1. Boundary conditions are known with variable injection rates for the injection wells and constant bottomhole pressure constraints for the producers. Thus there are known production rates with no changes (adding, removing, or re-completion of producers) during the process of data acquisition. For synthetic data, the data cleaning process is formatting the data from the numerical simulator for use in the simplified methods. The values for the connectivity obtained from the RM model are listed in Table 4.1 and shown in Figure 4.1:

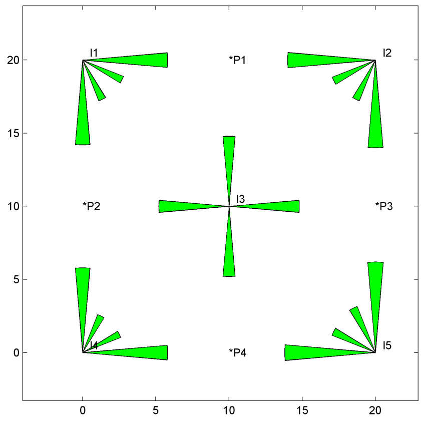


Figure 4.1: Connectivity values representation for the RM model

Table 4.1: λ_{ij} values for the RM model

	P01	P02	P03	P04
I01	0.2961	0.2958	0.1534	0.153
I02	0.3032	0.1593	0.303	0.1591
I03	0.2474	0.2454	0.2442	0.2423
I04	0.146	0.2959	0.1466	0.2965
I05	0.1701	0.1682	0.3143	0.3124

These results were used to predict the production history for the four producers and the R^2 values were all greater than 0.98. Figures 4.2 and 4.3 show simulated and calculated total field production rates respectively. Both the connectivity values and the prediction of rates agree with the results obtained by Albertoni et al. (2003).

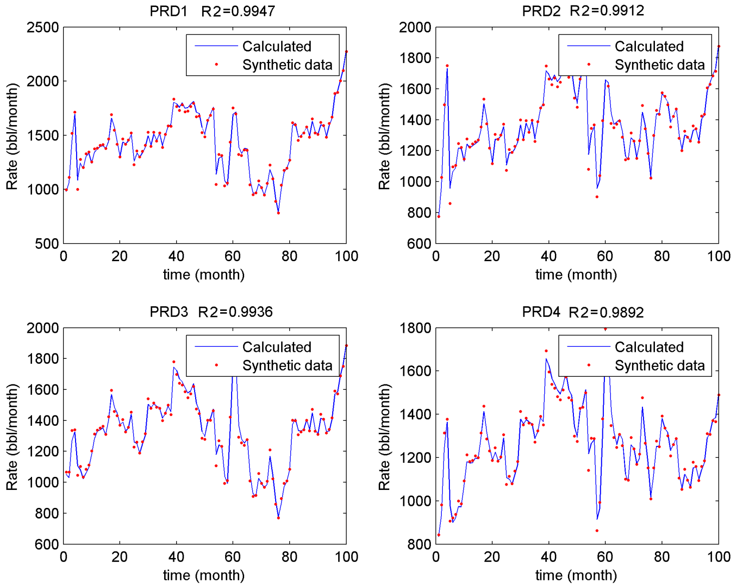


Figure 4.2: Prediction production rate values using RM model, 4×5 model

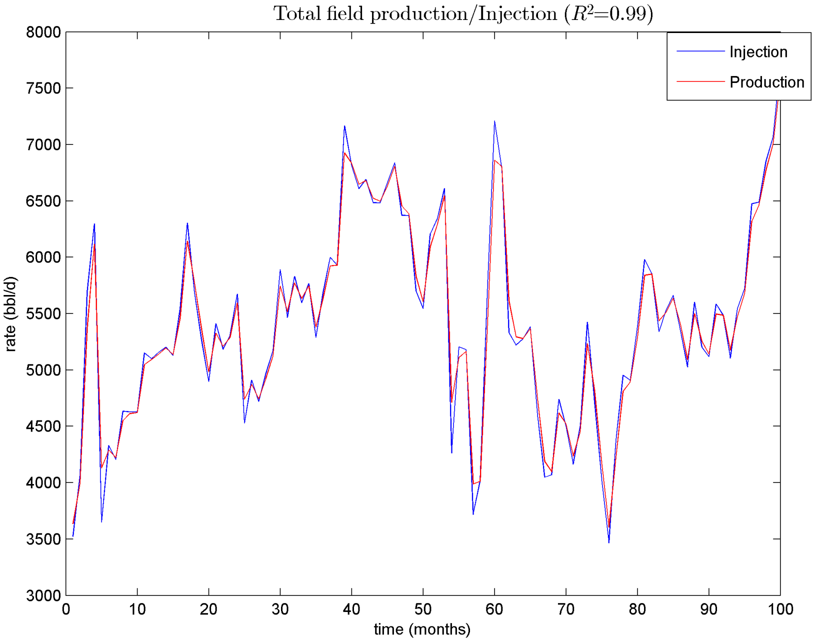


Figure 4.3: Total field injection/production prediction using the RM model

4.1.2 16×25 Case

This is an extended case of the previous one balanced injection and production as shown in Figure 4.4, Figure 4.5 shows the relative connectivities that were obtained from the RM model.

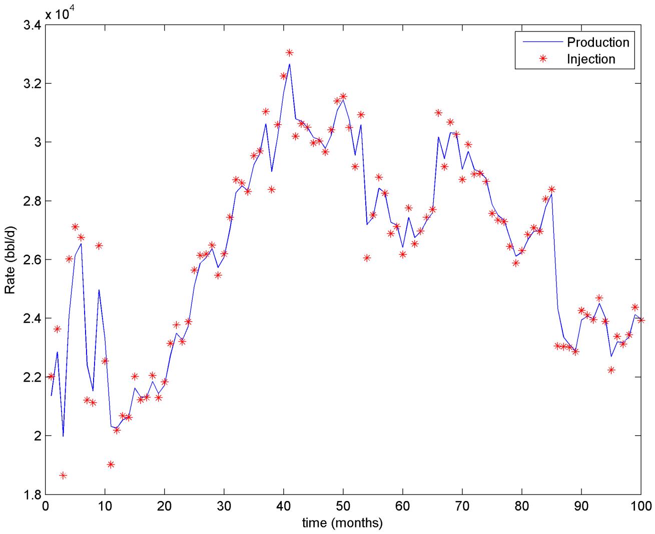


Figure 4.4: Total field injection/production prediction 16×25 using RM model

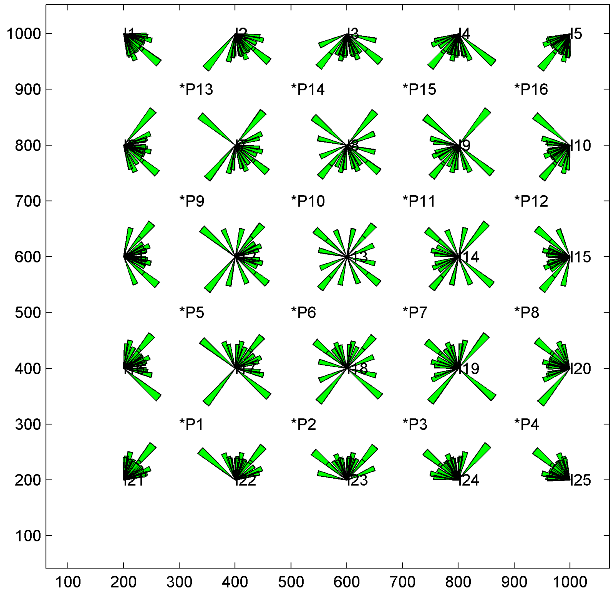


Figure 4.5: Connectivity values for the 16×25 case using the RM model

These values were used to reproduce the production rates and were plotted. Again, very high R^2 values were obtained for the predicted rates. Table 4.2 shows the connectivity values for all the well pairs. Similar to cases in (Albertoni et al. 2003) there are some small negative values that appear in the connectivity results. These negative values especially for injector-producer pairs where another injector with stronger influence on the producer is positioned between the injector and producer with the negative value. Figure 4.6 shows the predicted production values for 16×25 case

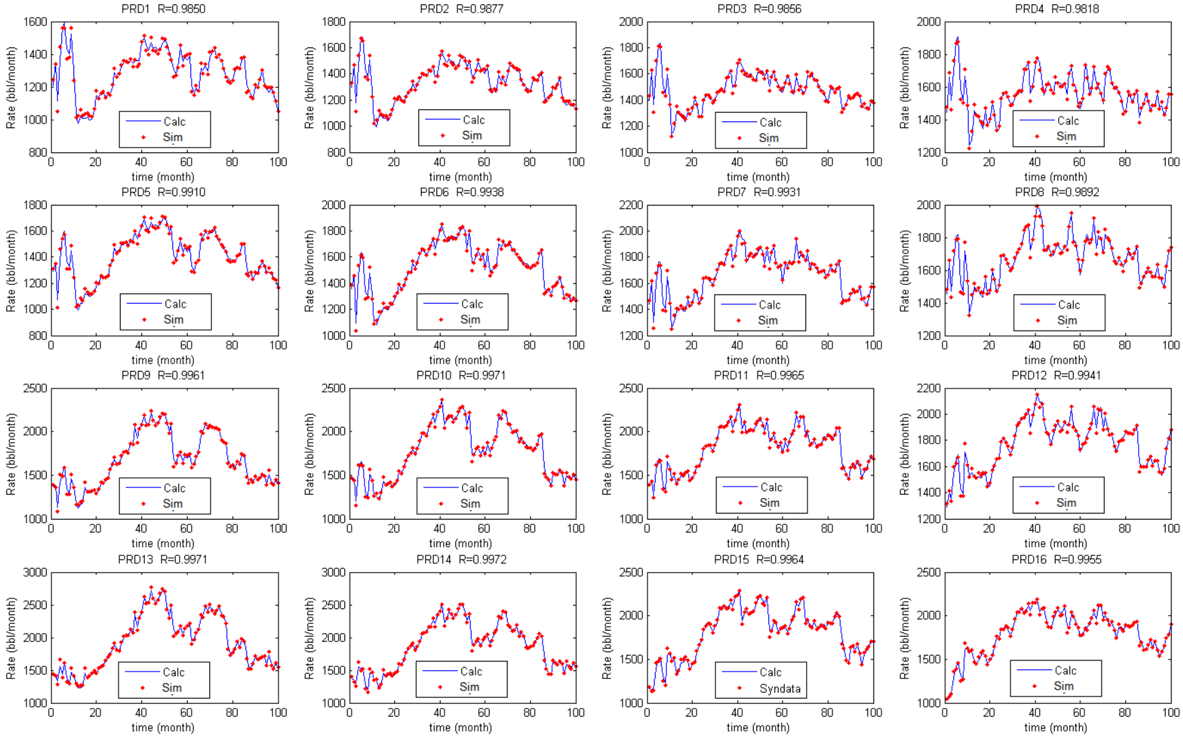


Figure 4.6: Prediction using RM model for 16×25 wells

4.2 RM model application discussion

The RM method was applied for these cases since they were balanced waterflooding cases. The properties were homogeneous and isotropic reservoirs in both cases. In the first case (4x5) the results were almost symmetrical and identical for each injector. For the second case the results were more directed and variable. Taking into consideration that for every injector there are 16

Table 4.2: Connectivity values for 16× model

	P 1	P 2	P 3	P 4	P 5	P 6	P 7	P 8	P 9	P 10	P 11	P 12	P 13	P 14	P 15	P 16
I 1	0.346	0.135	0.049	0.011	0.142	0.070	0.025	0.000	0.079	0.036	0.009	-0.007	0.077	0.030	0.001	-0.012
I 2	0.230	0.204	0.095	0.068	0.075	0.063	0.044	0.043	0.007	0.006	0.004	0.010	-0.022	-0.022	-0.024	-0.016
I 3	0.110	0.199	0.193	0.088	0.066	0.061	0.047	0.026	0.044	0.013	-0.013	-0.025	0.043	-0.002	-0.038	-0.059
I 4	0.028	0.068	0.183	0.212	0.010	0.022	0.051	0.068	0.005	0.005	0.012	0.021	0.014	0.007	0.006	0.008
I 5	0.070	0.082	0.127	0.301	0.058	0.063	0.075	0.113	0.050	0.043	0.044	0.051	0.061	0.041	0.034	0.033
I 6	0.238	0.092	0.035	0.011	0.204	0.072	0.024	0.002	0.084	0.043	0.016	-0.003	0.049	0.030	0.013	-0.007
I 7	0.165	0.141	0.052	0.024	0.146	0.127	0.048	0.026	0.065	0.056	0.040	0.030	0.043	0.039	0.038	0.035
I 8	0.011	0.086	0.087	0.010	0.009	0.083	0.083	0.012	0.001	0.021	0.021	0.008	0.001	0.012	0.015	0.012
I 9	0.007	0.031	0.111	0.127	0.004	0.024	0.097	0.109	0.001	0.008	0.023	0.028	0.003	0.001	0.002	0.001
I 10	0.017	0.026	0.065	0.198	0.011	0.019	0.051	0.169	-0.001	0.011	0.031	0.060	-0.016	0.002	0.019	0.028
I 11	0.083	0.068	0.084	0.108	0.183	0.088	0.092	0.109	0.190	0.096	0.097	0.106	0.085	0.079	0.087	0.093
I 12	0.049	0.035	0.014	0.004	0.126	0.107	0.030	0.008	0.133	0.114	0.037	0.013	0.068	0.055	0.033	0.019
I 13	0.008	0.022	0.024	0.014	0.025	0.089	0.090	0.026	0.032	0.096	0.095	0.030	0.027	0.039	0.038	0.025
I 14	0.014	0.024	0.044	0.057	0.018	0.038	0.112	0.130	0.021	0.042	0.118	0.135	0.022	0.035	0.057	0.071
I 15	-0.028	-0.012	0.015	0.056	-0.028	-0.009	0.032	0.157	-0.037	-0.016	0.027	0.156	-0.051	-0.028	0.004	0.053
I 16	0.038	0.023	0.013	0.005	0.081	0.045	0.023	0.014	0.201	0.078	0.039	0.027	0.231	0.096	0.051	0.038
I 17	0.026	0.022	0.018	0.016	0.056	0.046	0.028	0.020	0.147	0.127	0.047	0.025	0.174	0.147	0.057	0.029
I 18	0.022	0.023	0.022	0.019	0.031	0.042	0.041	0.028	0.049	0.114	0.113	0.045	0.057	0.128	0.127	0.055
I 19	0.012	0.011	0.014	0.015	0.016	0.022	0.038	0.045	0.023	0.042	0.117	0.134	0.031	0.055	0.137	0.159
I 20	0.023	0.026	0.031	0.039	0.019	0.027	0.048	0.085	0.017	0.035	0.080	0.209	0.018	0.040	0.095	0.242
I 21	0.062	0.052	0.044	0.041	0.080	0.060	0.046	0.040	0.149	0.091	0.058	0.046	0.346	0.148	0.078	0.058
I 22	0.035	0.032	0.027	0.023	0.053	0.046	0.038	0.032	0.112	0.088	0.056	0.038	0.276	0.224	0.090	0.044
I 23	0.040	0.037	0.033	0.034	0.038	0.039	0.038	0.039	0.059	0.072	0.071	0.057	0.110	0.196	0.191	0.091
I 24	-0.013	-0.005	0.005	0.018	-0.009	-0.007	0.006	0.022	-0.001	0.006	0.033	0.058	0.015	0.043	0.159	0.199
I 25	0.053	0.045	0.044	0.048	0.055	0.057	0.066	0.081	0.066	0.079	0.111	0.167	0.082	0.106	0.181	0.380

values of connectivity (one for each producer), the process of interpreting theses values is more difficult. It can be seen that the injection has no strong preference in terms of the magnitude of the connectivity. Because the model is homogeneous but with variable injection amounts it can be seen from Figure 4.5 that the model has some symmetry but this symmetry is broken due to the variable injection.

4.2.1 Running time

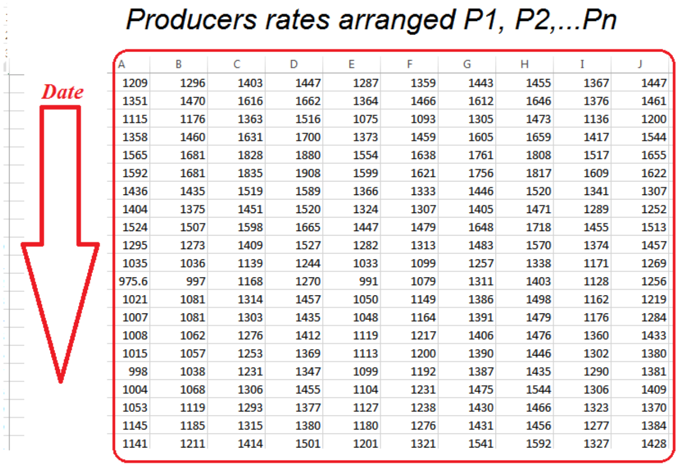
The time required to obtain the 400 connectivity values was less than one second. This time also includes the time necessary to use R to generate the rose diagrams and to do the prediction routine. The code has been automated and generalized to perform all these tasks with little effort if the data is in the proper format. The main time sink here is data cleanup and preparation into the appropriate format.

4.2.2 Prediction accuracy

The prediction yielded R^2 values for all the producers that were greater thant .98 with most greater than 0.99.

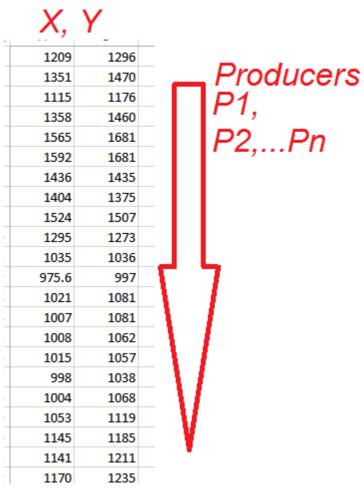
4.2.3 Input files

Two groups of input data are needed: firstly, the injection and production data and secondly the locations of the wells in the form of xy coordinates and the names of the wells for the purpose of representation. Figure 4.7 show a typical input spreadsheet for injection and production data in which each row is a particular date and each column is the production or injection rate for that date. The spreadsheet must be saved as a comma separated values (CSV) file if a spreadsheet is used for the input data. A location file is shown in Figure 4.8 and there should be separate files for the injection well locations and for the production well locations. Each row of the file is the x and y location of a well and the well order in the location file is the column order in the production/injection rate file. Units for the production and injection rates files should be consistent and should be flow rate per time period. Rates in this thesis are barrels per month.



A	B	C	D	E	F	G	H	I	J
1209	1296	1403	1447	1287	1359	1443	1455	1367	1447
1351	1470	1616	1662	1364	1466	1612	1646	1376	1461
1115	1176	1363	1516	1075	1093	1305	1473	1136	1200
1358	1460	1631	1700	1373	1459	1605	1659	1417	1544
1565	1681	1828	1880	1554	1638	1761	1808	1517	1655
1592	1681	1835	1908	1599	1621	1756	1817	1609	1622
1436	1435	1519	1589	1366	1333	1446	1520	1341	1307
1404	1375	1451	1520	1324	1307	1405	1471	1289	1252
1524	1507	1598	1665	1447	1479	1648	1718	1455	1513
1295	1273	1409	1527	1282	1313	1483	1570	1374	1457
1035	1036	1139	1244	1033	1099	1257	1338	1171	1269
975.6	997	1168	1270	991	1079	1311	1403	1128	1256
1021	1081	1314	1457	1050	1149	1386	1498	1162	1219
1007	1081	1303	1435	1048	1164	1391	1479	1176	1284
1008	1062	1276	1412	1119	1217	1406	1476	1360	1433
1015	1057	1253	1369	1113	1200	1390	1446	1302	1380
998	1038	1231	1347	1099	1192	1387	1435	1290	1381
1004	1068	1306	1455	1104	1231	1475	1544	1306	1409
1053	1119	1293	1377	1127	1238	1430	1466	1323	1370
1145	1185	1315	1380	1180	1276	1431	1456	1277	1384
1141	1211	1414	1501	1201	1321	1541	1592	1327	1428

Figure 4.7: Injection/Production input format



1209	1296
1351	1470
1115	1176
1358	1460
1565	1681
1592	1681
1436	1435
1404	1375
1524	1507
1295	1273
1035	1036
975.6	997
1021	1081
1007	1081
1008	1062
1015	1057
998	1038
1004	1068
1053	1119
1145	1185
1141	1211
1170	1235

Figure 4.8: Injection/Production locations input format to CRM, CM, and RM model codes in Matlab®

There are then 4 input files to prepare (Injection data, Production data, Injector locations, Producer locations). The routines require that there be no missing data in these files. If there is missing data (i.e. dates with production or injection values of 0) this thesis will present techniques for dealing with this missing data prior to running the model.

4.3 Application of CRM Model

4.3.1 4×5 model

Using the same set of data used in section 4.1.1 the CRM model was applied to the 4x5 case and the connectivity results (λ_{ij} values) are shown in Figure 4.9 and Table 4.3.

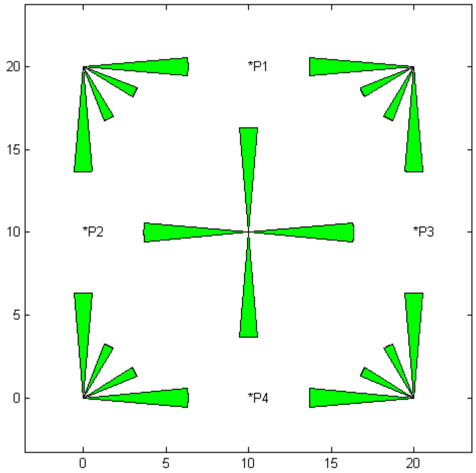


Figure 4.9: Connectivity values for 4x5 case using CRM model

The same production history was predicted using the obtained values for connectivity and time constants, these results are shown in Figure 4.10

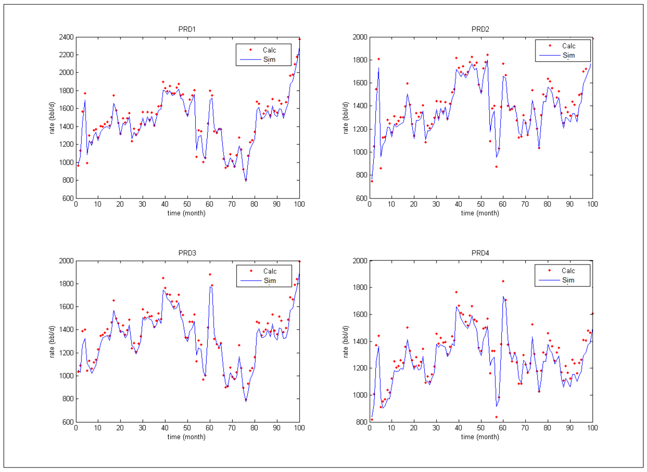


Figure 4.10: Production predicted history for 4x5 case using the CRM model

Table 4.3 shows the values for the connectivity.

Table 4.3: λ_{ij} values for the CRM model

	P01	P02	P03	P04
I01	0.316777	0.316777	0.174834	0.174834
I02	0.316777	0.174834	0.316777	0.174834
I03	0.316777	0.316777	0.316777	0.316777
I04	0.174834	0.316777	0.174834	0.316777
I05	0.174834	0.174834	0.316777	0.316777

4.3.2 16×25 model

Using the same set of data used in section 4.1.1 the CRM model was applied to the 16x25 case. Production rates were predicted as shown in Figure 4.11 using the obtained values for the connectivity and the time constants.

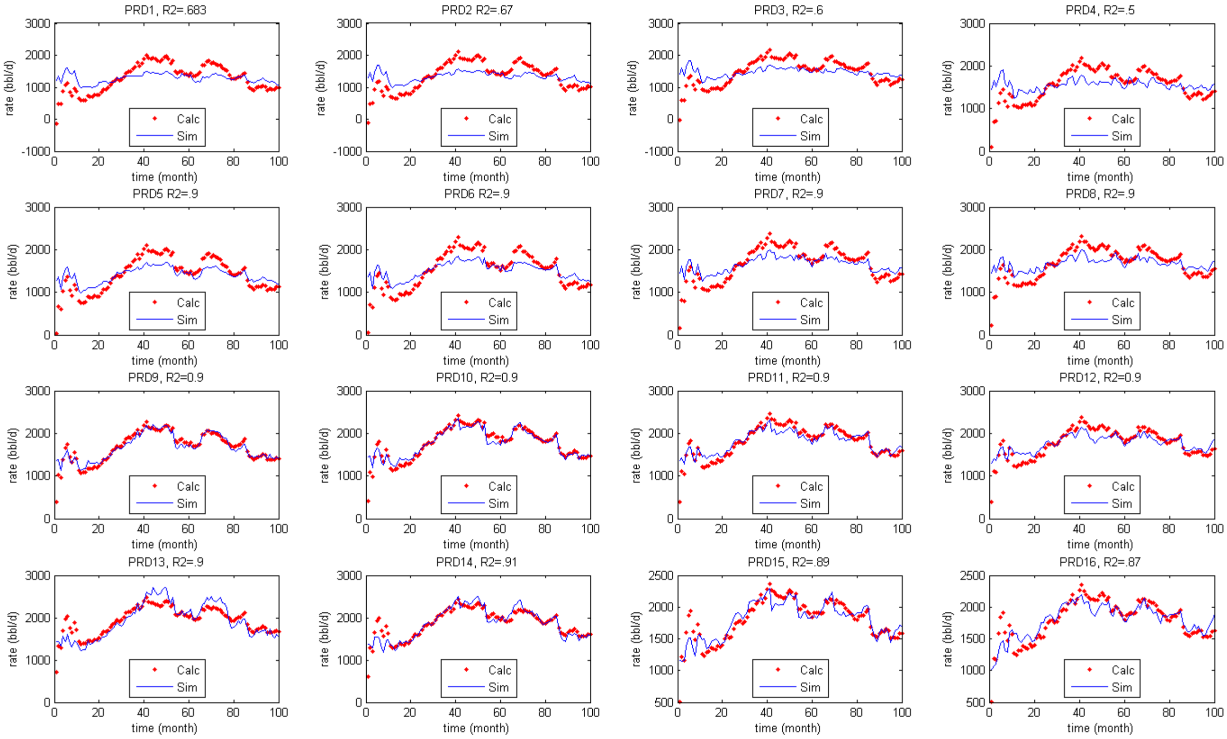


Figure 4.11: Production predicted history for 16×25 case using the CRM model

For this case the optimization was forced to stop after a number of iterations (1000 for each variable) as there was no more change in the value of the variables. It was noted also that the values of τ_{ij} do not have a significant effect on the predicted value of the production rates for the same time period. It was noted during the optimization that the value of τ_{ij} did not have large changes from the initial assumptions calculated using the CRMT (tank) model. This could be due to the fact that the synthetic field is homogeneous. Figures 4.12 and 4.13 show the time constant distribution for both the 16×25 and the 4×5 case.

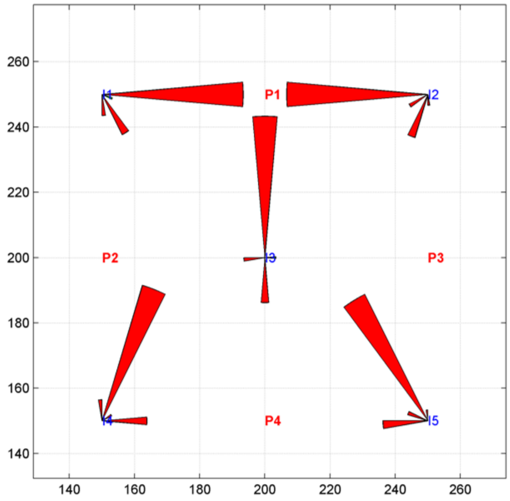


Figure 4.12: Values of the time constant for 4×5

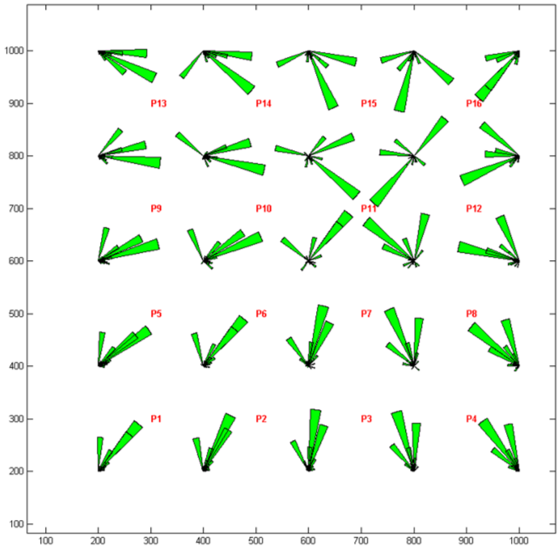


Figure 4.13: Values of the time constant for 16×25

CHAPTER 5
MISSING AND CORRUPTED DATA TREATMENT

“the only thing we know for sure about a missing data point is that it is not there, and there is nothing that the ma

Howard Wainer- American Statistician

5.1 Introduction

Oil field data are not always accurate and complete. According to Nobakht et al. (2009) $60 billion per year is the cost of corrupted and missing data. Unless extreme caution is taken to collect the data, every dataset is expected to have at least 1-5% error from different sources. Some of the error sources are:

1. Human error: these errors include the misreading and interpretation of the data that has been recorded by field personnel.
2. Data collection setup error: the way that data is collected might have a probability of error to propagate as shown Figure 5.1., where the flow rate of all the four producers is obtained by dividing the total flow rate by 4. This assumes that the rates from the four producers are equal, and in fact they most likely are not.

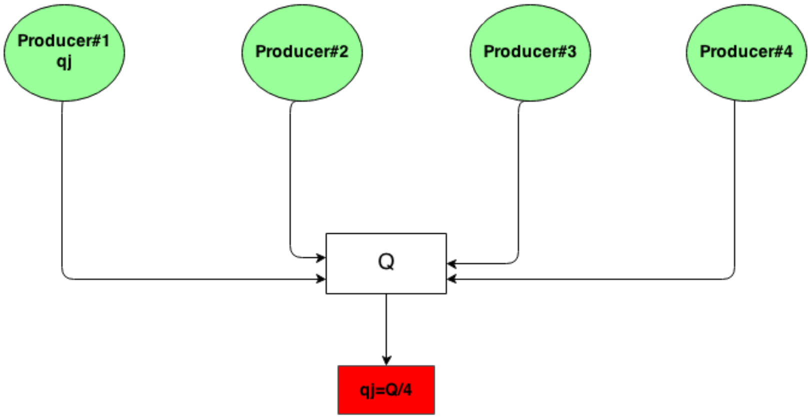


Figure 5.1: Missing data setup error case example

Some examples were mentioned by Nobakht et al. (2009) and the ones related to production and injection data include:

- Averaging the rates of a specific well from an adjacent group of wells.
- Incorrect assumptions (Single phase flow while having increasing GOR for example)
- Incorrect meter location. (i.e. it is downstream while the choke is not fully opened)
- Incorrect measure synchronization, (i.e. taking the injection and production data with a time shift between them and treating them as being measured at the same time)

3. Device malfunction which results in corrupted data.

The missing or corrupted data can have an impact on the decisions that need to be made. The error in one data set may propagate to parameters that depend on these data sets. As discussed previously, the mechanisms causing the missing data will be discussed from a statistical point of view.

5.2 Missing data

5.2.1 Detection

Detection of the missing data is easy to implement. Simply the missing data will be denoted by NaN (Not-A-Number) in the Matlab®coding environment. The code also will be extended to provide the ratio of the missing data.

5.2.2 Missing patterns

According to Schafer and Graham (2002) there are several typical patterns to missing data that might narrow down the distribution of missing points. These patterns are characterized in terms of how many response variables are contributing to the missing data. If one of the variables has missing data that is called a univariate pattern. A monotone pattern is if the same data point is missing in all the variables. An Arbitrary pattern means that any N data points could be missing for any variable Y without any dependency. Figure 5.2 shows these patterns in order where Figure 5.2A is a univariate pattern, Figure 5.2B is a monotone pattern and Figure 5.2C is an arbitrary pattern. A fourth case D which is an extension of case A is when more than one variable is missing in the model data which is more likely to happen in oil field data.

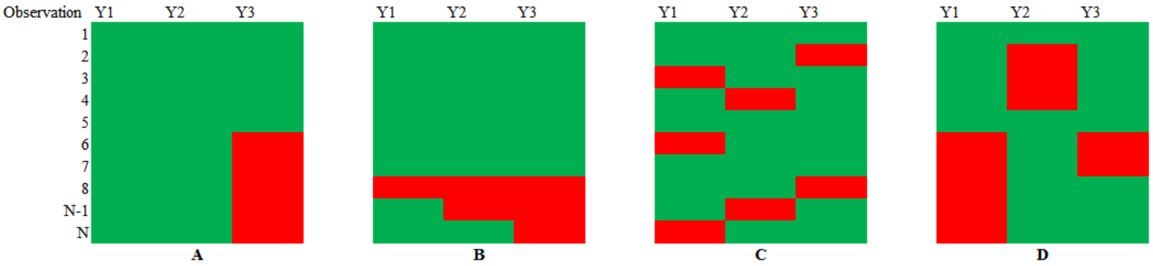


Figure 5.2: Missing data patterns as per (Schafer 2002)

5.2.3 Missing mechanisms

Schafer and Graham (2002) and Nobakht et al. (2009) divide the missing mechanisms into three main methods depending on the probability for a specific observation in a specific variable to be missing.

Missing completely at random (MCAR): As per Schafer and Graham (2002), MCAR is defined when you have missing data that is independent from the data set itself.

$$P(R|Y_{com}) = P(R) \tag{5.1}$$

where R is an indicator variable that defines what is missing and what is not missing, Y_{com} is the complete data set i.e. $Y_{com} = \{Y_{obs}, Y_{miss}\}$.

Missing at random (MAR) In this case the probabilities of the missing data are allowed to depend on the observed data not the missing data.

$$P(R|Y_{com}) = P(R|Y_{obs}) \tag{5.2}$$

Missing not at random (MNAR): The distribution of R here depends on the missing data itself (Schafer and Graham 2002). As an example, in Figure 5.3 if there is a problem in valve no.1, both Prod1 and Prod2 data will be missing and the missing data for Prod1 here depended on Prod2 because of the location of the malfunctioned valve. While if the problem location is in value no.2, Prod1 data still may be acquired.

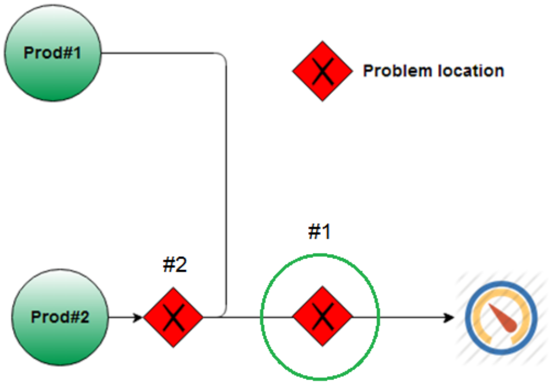


Figure 5.3: MNAR Example

5.2.4 Simulated missing patterns

In order to simulate some of these missing data situations, a randomized process was implemented to create these patterns. These patterns depend on some accidents that might happen in the process of data collection and there might be other patterns depending on the conditions and mechanism of the acquisition. All of these patterns were taken from a statistical study done by Schafer and Graham (2002) and were coded into MATLAB®. The process starts by asking the user to input the desired ratio of the missing data and then it will generate the patterns explained below according to the ratio provided. An explanation will also be given for when it is expected that a specific pattern might happen under oilfield conditions.

5.2.4.1 Arbitrary missing pattern

This pattern consists of individual random data missing in each data column (injector/producer). Such a pattern could happen due to conditions like measuring device malfunction, human error, simply missing a gauging opportunity for a specific period of time, etc. This pattern of data loss can be found in everyday data not only in oilfield related data. The way this was coded into Matlab® was to generate a number of random locations equal to the user input ratio of missing data. The actual values were replaced at these locations with NaN values (Matlab® reads NaN as missing). Figure 5.4 shows the pattern for a missing data ratio of 6%. It can be seen from the Figure that there are more than random missing data in all the columns of the data, each column represents one injector or producer and also the amount of the missing data per column is not constant.

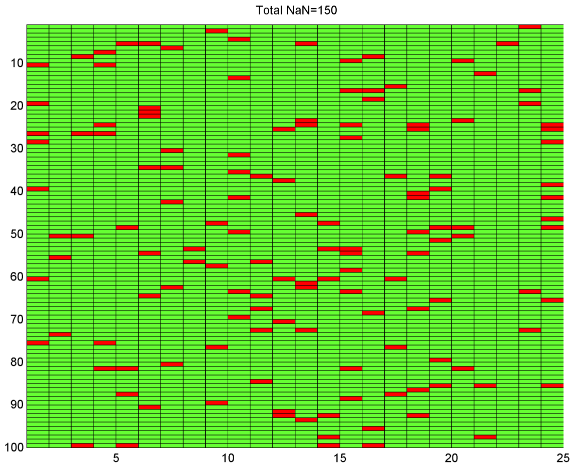


Figure 5.4: Arbitrary missing pattern

5.2.4.2 Monotone missing pattern

This pattern is created by having missing data that starts at some time and has different lengths of time where details are missing before returning to normal data collection. This pattern would be expected to happen if there was any general trouble with the measuring system across the field. The author has personally experienced a user-input data acquisition issue like this in the Rumaila field in Iraq. The missing data for this pattern was also generated using a user input ratio of the amount of data missing. This pattern also has been discussed by Schafer and Graham (2002) and is shown in Figure 5.5.

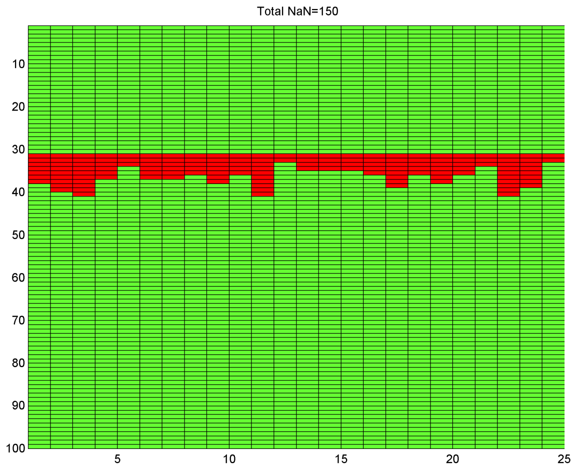


Figure 5.5: Monotone missing pattern

5.2.4.3 Multivariate missing pattern

This pattern is expected to happen when wells or measuring equipments have some regular maintenance activities that take a fixed length time period to finish and the starting time depends on the well records. So these would be randomly placed periods of missing data with nearly equal length. Figure 5.6 shows this pattern.

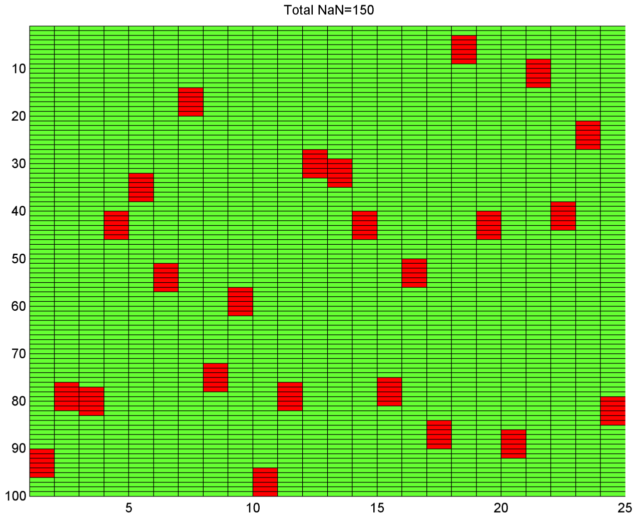


Figure 5.6: Multivariate missing pattern

5.2.4.4 Modified multivariate pattern

This pattern occurs when there are different lengths of missing data that occur at different times for each well. This pattern is expected to happen when there is any random accident or problem in a single measurement device. Figure 5.7 shows an example of this pattern.

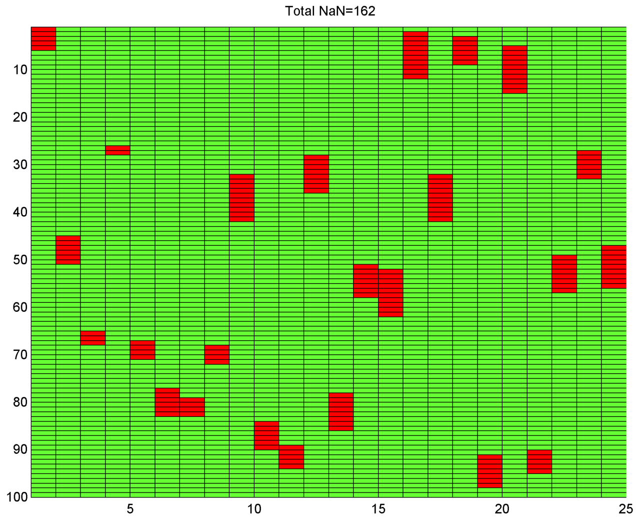


Figure 5.7: Modified multivariate missing pattern

5.3 Dealing with missing data

In general there are three options to deal with missing or corrupted data. First, Albertoni et al. (2003) stated that missing data causes singular covariance matrices so full data sets are required. The effect is then to limit the time frame to these full data sets and drop all remaining data either up to the time of the missing data or after. Second, the missing data time steps could be discarded but the rest of the data would be used. The third method would be to find a method to impute a (presumably) good data estimate. This is often done by taking advantage of data dependency to implement some linear models to be used in prediction. To find the missing data is not a difficult process but deciding whether the data is corrupted or not is the difficulty. Finally, a situation-dependent method to treat these issues must be developed. In this chapter an integrated approach will be introduced that combines several methods from the statistical point of view to deal with the missing data.

5.3.1 Dropping the missing points

Presuming that oil and gas data has some missing data and restricting the simple models to complete data sets sometimes might not be an option, so dropping the data for the missing time period is the next least preferable action. The data must be synchronized, i.e. all the readings for all the wells should be at the same time. So to imagine the case, if there are 16 wells and each well has 100 data points, and assuming that there is a missing pattern where each well is off by one time period as shown in Figure 5.8, then dropping data would cause the entire data set to be useless. There must then be a cutoff ratio of the the lost data whereby the drop method cannot be used. In the following sections this cut off ratio will be investigated.

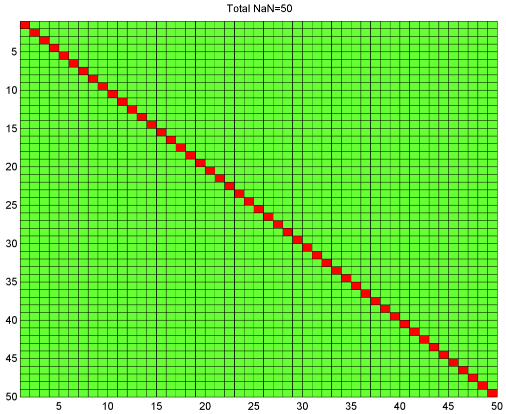


Figure 5.8: Example on a case that dropping the missing data will result non-usable set of data

5.3.1.1 Methodology

As per Sayarpour (2008) the minimum recommended no. of points is $4 \times N_{inj} \times N_{pro}$, for a case of 16x25 the minimum required data points are around 40 points. In the approach in this work missing data will be generated in a random pattern as in Figure 5.4 starting with 1% and increasing the missing data in 1% increments.The error in the connectivity values will be computed using the full data set values as the actuals. These errors will then be plotted vs the percentage of the missing data. The process was simulated 100 times to overcome any random behavior although the function used is a pseudo-random function in Matlab (rand) with a normal distribution from 0-1. Figure 5.10 shows the relationship between the missing ratio and the error encountered in calculating the value of a particular λ_{ij}. It can be seen that error is propagating as the missing ratio increases. There are times when the error decreases as the missing ratio increases, which can be explained by the missing data occurring during a flat part of the curve (i.e. almost equal value data points). Figure 5.9 shows one random realization were the SSE error in the values of λ_{ij} and R^2 value are plotted against the missing ratio. The 100 random realizations were simulated to show the effect of the missing ratio on the SSE and results shown in Figure 5.10. The SSE term is defined as:

$$SSE = \sum_{i=1}^{N} (\lambda_{actual} - \lambda_{calculated})^2 \tag{5.3}$$

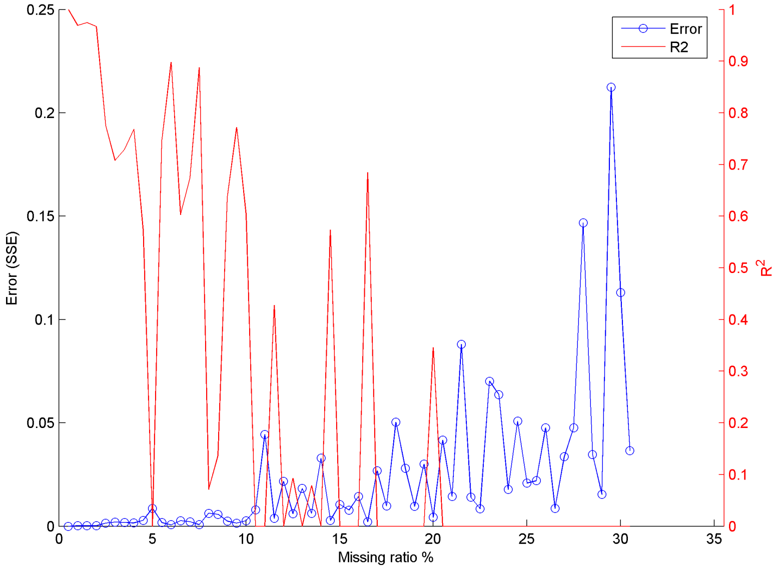


Figure 5.9: Error encountered (SSE) in the value of λ_{ij} and R^2 vs. missing ratio for one realization

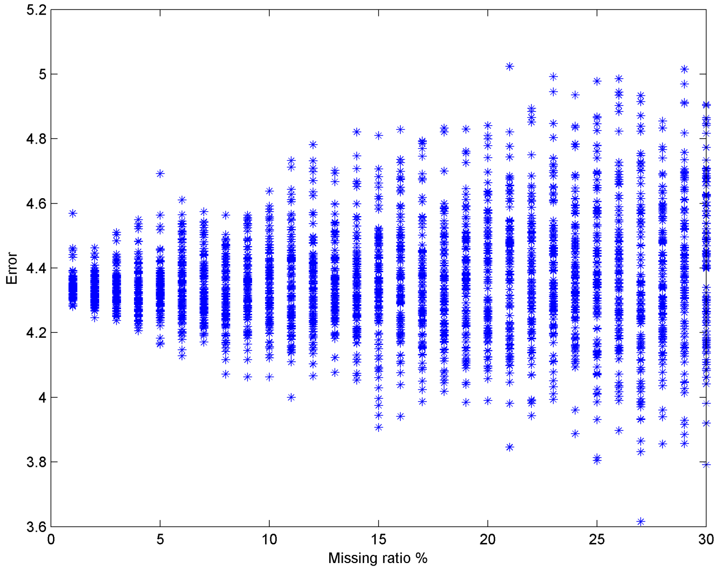


Figure 5.10: Error encountered (SSE) vs. missing ratio

The value of R^2 was also investigated vs. the missing ratio with a change of missing ratio from 0-5%. The random patterns were generated 100 times to view the effect on R^2 which are shown in Figure 5.11, In general the value of R^2 drops drastically as the missing data increases.

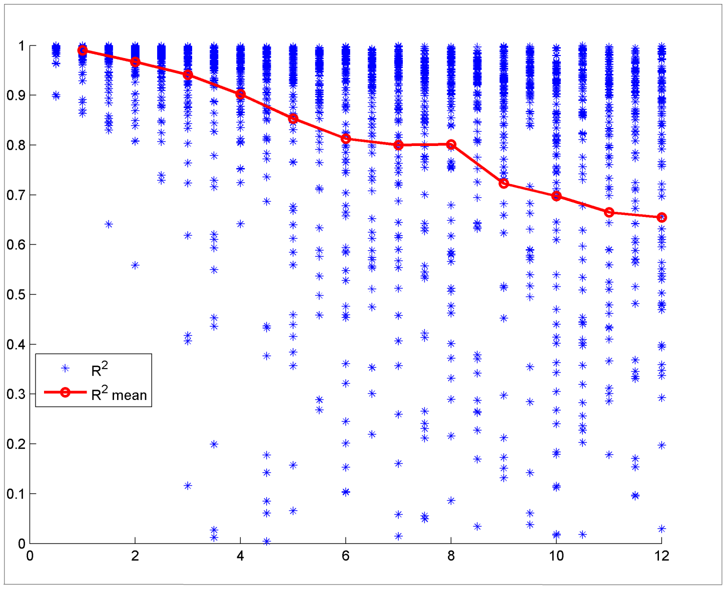


Figure 5.11: R^2 vs. missing ratio for arbitrary pattern

The multivariate pattern was simulated in the same way. Figure 5.14 shows that the value of R^2 decreases faster than the arbitrary pattern (which can be seen by the denser distribution as the missing rate increases). The explanation here is that this is due to the blocks of missing data rather than the small individual missing points. Figure 5.12

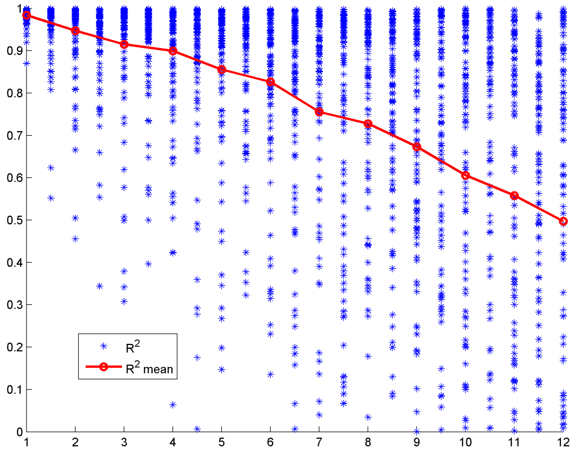


Figure 5.12: R^2 vs. missing ratio for multivariate pattern

Figure 5.13 shows R^2 trend for the modified multivariate missing pattern. It can be seen that multivariate and modified multivariate have the largest R^2 drop.

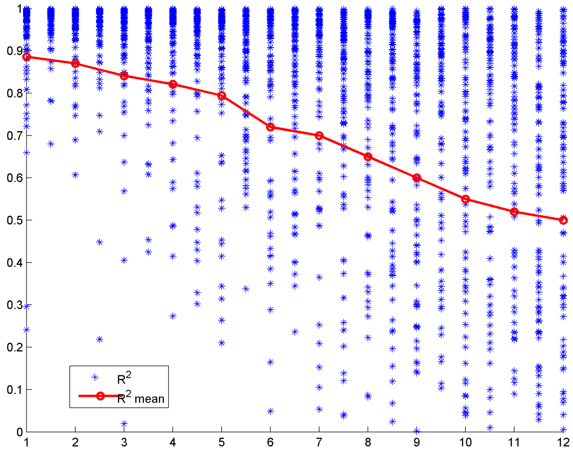


Figure 5.13: R^2 vs. missing ratio for modified multivariate pattern

The group of plots in Figure 5.14 summarizes and visualizes pattern related error in a different way by showing the values of λ_{ij} for the case with no missing data compared to that with the maximum missing data of 15%. From these figures the pattern-related error effect from the least affected to the most affected will be:

1. Arbitrary error (pattern shown in Figure 5.4)
2. Monotone error (pattern shown in Figure 5.5)
3. Multivariate error (pattern shown in Figure 5.6)
4. Modified multivariate error (pattern shown in Figure 5.7)

5.3.1.2 Conclusion about dropping the missing data

From the above graphs we can conclude that the error propagation is maximum when the modified multivariate mode is in effect i.e. this pattern is expected to happen when there is a systematic accident or problem in the rate measurement devices. Thus, it takes time until the system can be returned to normal measurement. The least amount of error occurs when there is an arbitrary missing data which could happen at anytime. The recommendation here is that dropping is not an appropriate method if there is more than 3% missing data. The value of R^2 approaches 60% quickly once the amount of missing data hits 2% for all the patterns. This also helps explain the reasoning behind using only full data sets.

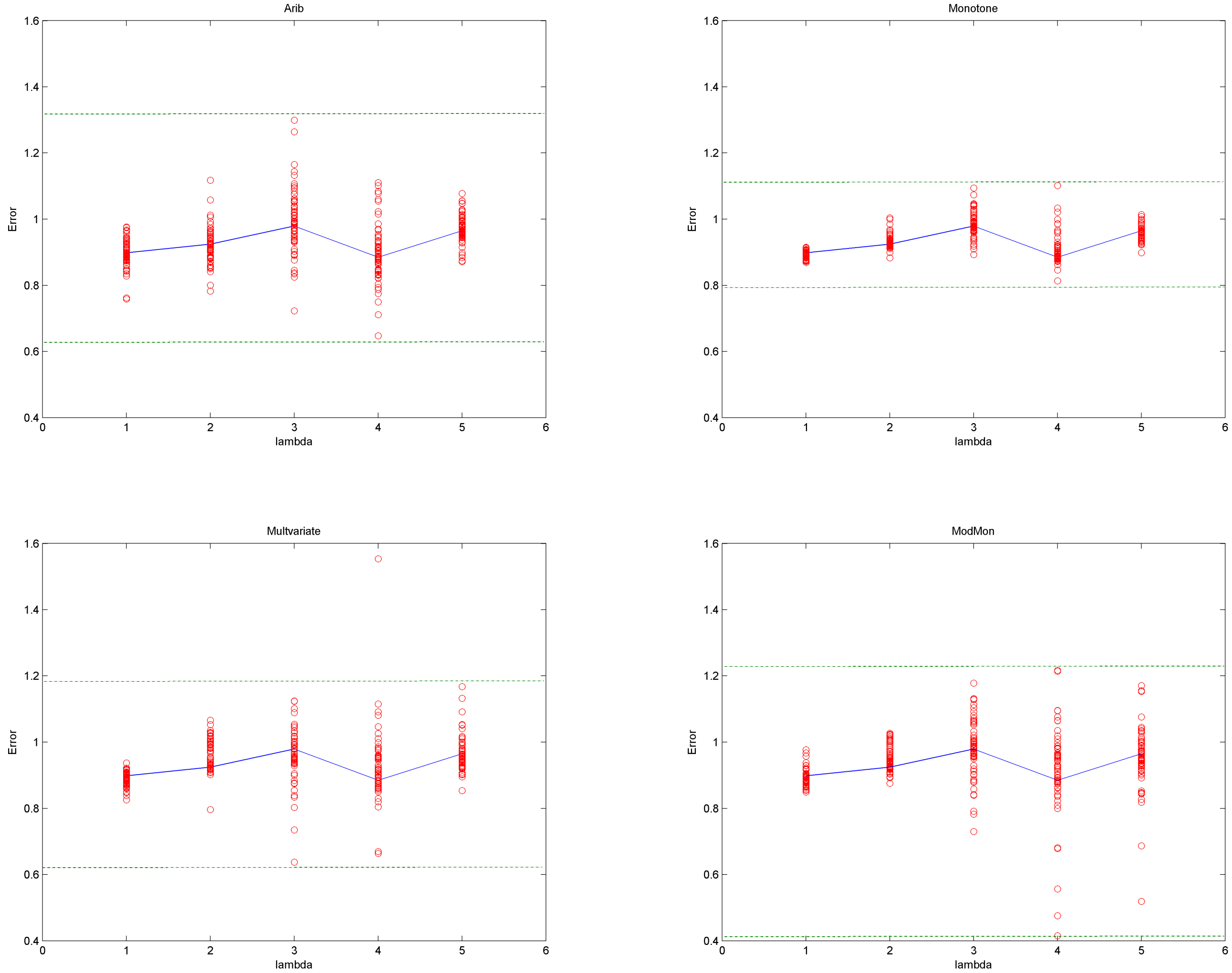


Figure 5.14: Subplots show the pattern related error in connectivity values for 15% missing ratio case

5.3.2 Missing data imputation

Using imputation methods when dealing with the missing data are more complicated than simply dropping the missing points but allow the inclusion of more data in the analysis. There are several methods for imputation discussed in Kabacoff (2011). The imputation techniques are more likely case-specific methodologies, hence the data dependency between values influence the imputation approach that needs to be taken. For injection and production data there is a significant dependency between these sets of data especially in a case were the injection and production is approximately balanced. In such a case the production is likely due to the injection displacement. This relation has been discussed in detail by Albertoni et al. (2003) and others. We will use this relation with minor modifications to make the imputation of the missing data.

5.3.3 Imputation suggested methods

In this section we will investigate the suggested imputation methods and develop the required code to implement these methods. It is good to mention that we will concentrate on the two extreme patterns of missing data (arbitrary and modified multivariate) while all the rest of the patterns represent intermediate cases between these patterns.

5.3.3.1 Linear dependency relations

The linear dependency between injection and production data is well known and has been discussed in the literature Albertoni et al. (2003) while Al-Yousef (2006) and Sayarpour (2008) went further and developed nonlinear models. In this section will use the linear relation suggested by Albertoni et al. (2003). The assumption here is a balanced case. The case that was used in the beginning of this chapter will be used here.

A. Missing values in production data:

The production in any producer is given by Albertoni et al. (2003) as:

$$\hat{q}_j = \lambda_o + \sum_{i=1}^{I} \lambda_{ij} i_i + \varepsilon \tag{5.4}$$

This means that if an estimate for the values of (λ_{ij}) can be made, values for the missing production data can be imputed from Eq. 5.4. For this case it is assumed that there is a complete data set. But what if there are missing injection rates? A decision algorithm that accounts for this case is required.

B. Missing values in injection data

Assuming a balanced injection case, implies that every injection rate, i_i, is distributed to all producers in an amount α_{ij} for each producer j. This also implies that each producer has a corresponding relationship with each producer. Starting from Eq. 5.4 above, for a balanced case every injector rate can be broken down to:

$$\hat{i}_i = \alpha_o + \sum_{j=1}^{J} \alpha_{ij}\, q_j + \varepsilon \tag{5.5}$$

where:

- α_{ij} has been used here to differentiate from the connectivity value λ_{ij}
- $\hat{i}_i$: is the calculated injection rate.
- α_o: is the remaining amount that does not contribute to the production. A different symbol has been used here to recognize it from the main equation in 5.4.
- ε: random error, typically assumed to be normally distributed.

Now, there are two equations 5.4 and 5.5 that can be used start from to impute the missing injection and production values. An algorithm needs to be developed that organizes these cases and decisions. Figure 5.15 shows the proposed flowchart to handle all the cases for missing injection and production data. This flowchart was coded and tested and will be discussed later in this chapter.

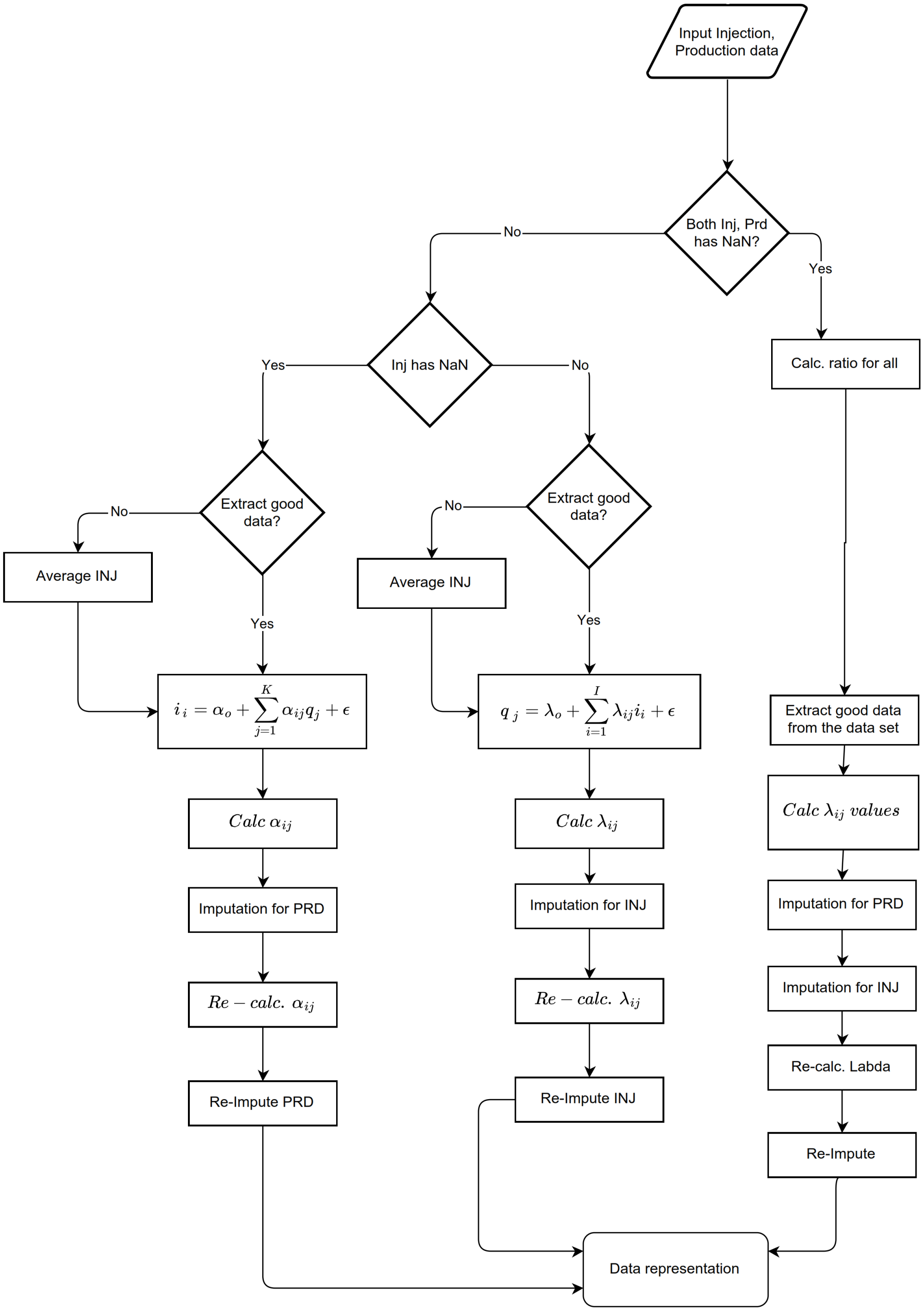


Figure 5.15: Imputation method decision making flowchart

5.3.3.2 Horizontal missing data (same time-step missing in injection and production data)

The most difficult cases to handle have been when data is missing for multiple injection and production wells at the same time, (i.e. when missing data is in the same row for multiple wells). This case will be called horizontal missing data in this thesis. For this case an approach that might ensure the least damage to the data will be followed. The injection data between two "good" points will be averaged assuming that the injection process fluctuates less than the production. After this averaging is done the matrices will be imputed again in an iterative process in an attempt reduce the error. The flowchart in Figure 5.15 has been modified to be able to handle the horizontal missing cases as shown in Figure 5.16. In general the imputation algorithm seems to performs well when the data missing is on the production side. Different scenarios and missing rates have been investigated and will be discussed later in this chapter.

Terms that will be used to investigate some missing scenarios are:

Horizontal missing: Having data missing in the same row (i.e. at the same time) in both the injection and production rate datasets.

Good data: Having data in the same row in both injection and production data sets with no missing data.

Averaging: Imputation option where the missing rates are assumed to be the average of two good data points prior to and after the missing value.

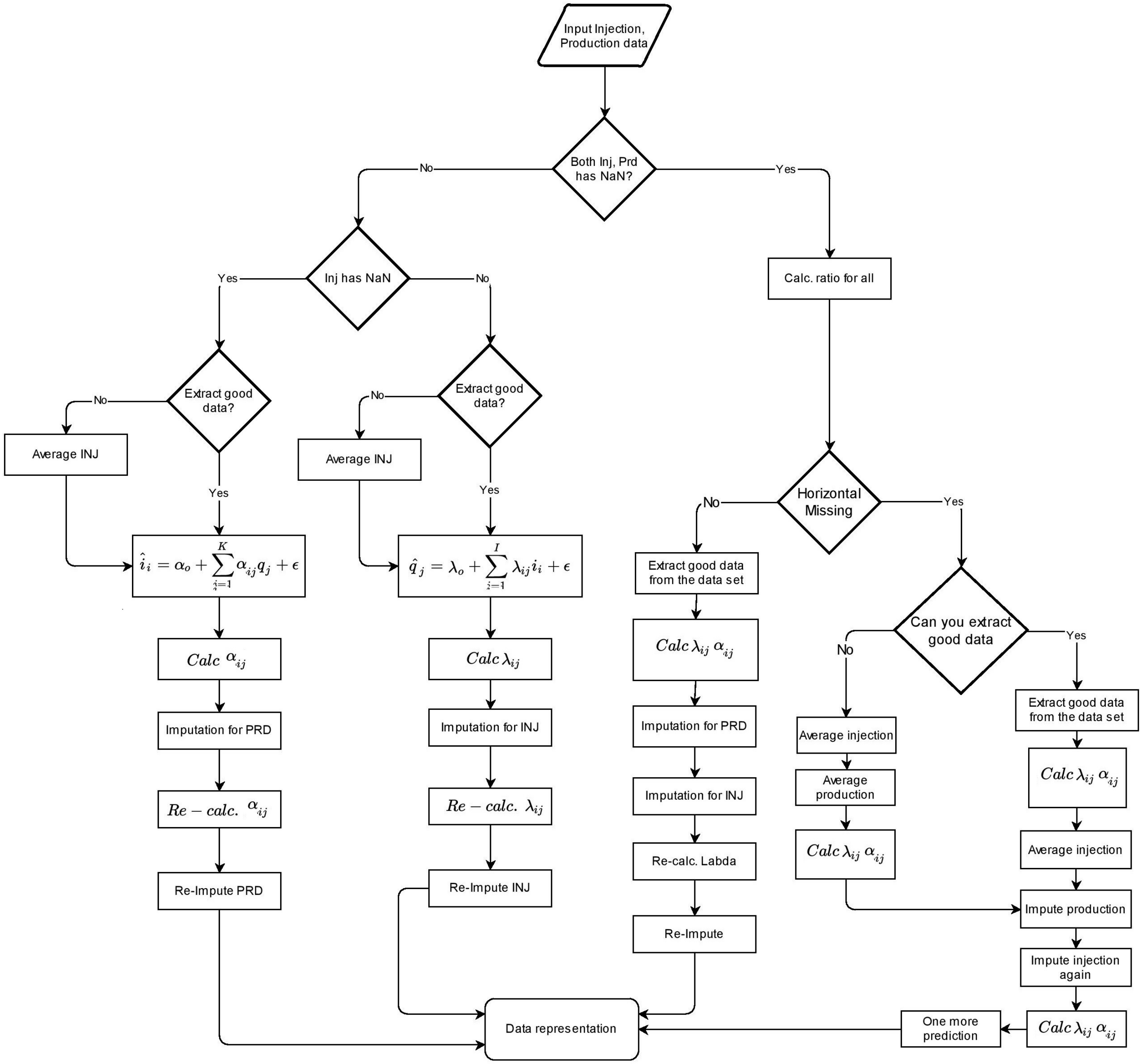


Figure 5.16: Decision flowchart for modified imputation method including the horizontal missing case

5.3.3.3 Horizontal missing /Production and Injection with a possibility to extract good data

A random pattern of missing data has been created for this case in both the production and injection information. Two medium and two high missing data rates were tested. The following strategy was used:

1. Check for horizontal missing data.
2. If data is missing then check for the possibility of extracting good data.
3. If it is possible, compute the values of λ_{ij} and α_{ij} using Equations 5.4 and 5.5 respectively.
4. Compute the values of $\lambda_{ij}, \alpha_{ij}$
5. Compute the average of two good data points prior to and after the missing value and use these to impute the missing injection value.
6. Use λ_{ij} and α_{ij} values to compute the averaged injection and production.
7. Repeat steps 3-5 until the calculated injection and production rates in step 5 are unchanged.

Figure 5.17 shows a case with 15% missing data. An R^2 value of 0.85 was obtained using the imputation method suggested using Eq. 5.5. Some of the imputed points at time steps 25, 39, 42, 45, 46, 83 and 85 have been imputed with a close estimate to the actual value of the data point.

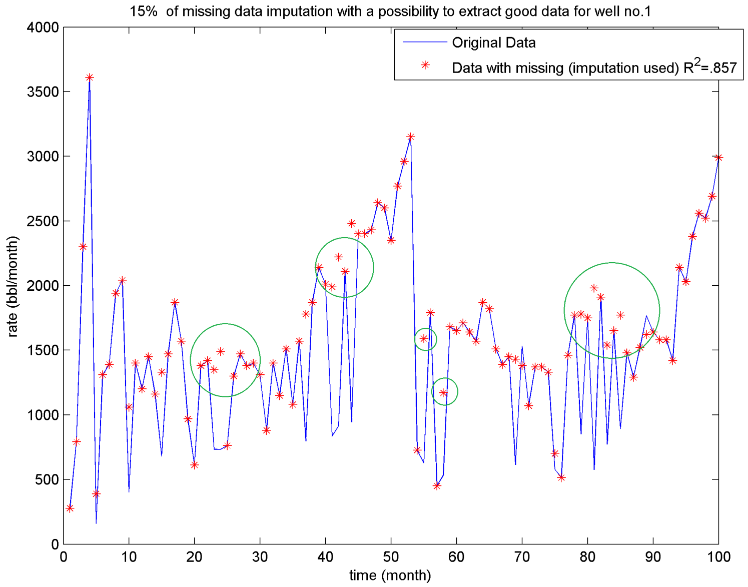


Figure 5.17: Results of missing data imputation for a missing rate of 15% with a possibility to extract good data (circles values show imputed data)

5.3.3.4 Horizontal Missing /Production and Injection with no possibility to extract good data

The case discussed in section 5.3.3.3 was for 15% missing data. In this section the amount of missing data was increased to 30% in order to reduce the chances of extracting good data from the model. The strategy followed in this case is almost the same as in 5.3.3.3 except in this case it was not possible to extract good data. Averaging was used to create some good usable data. These good data will be used to calculate the values $\lambda_{ij}, \alpha_{ij}$. As was mentioned previously an acceptable number of missing data should not exceed 6%. Figure 5.18 shows the results for 5x4 case where the production rates were imputed and plotted against a good dataset. The rates deviated in some places due to the high rate of missing data, but overall the smallest R^2 value was 0.88 which is reasonable.

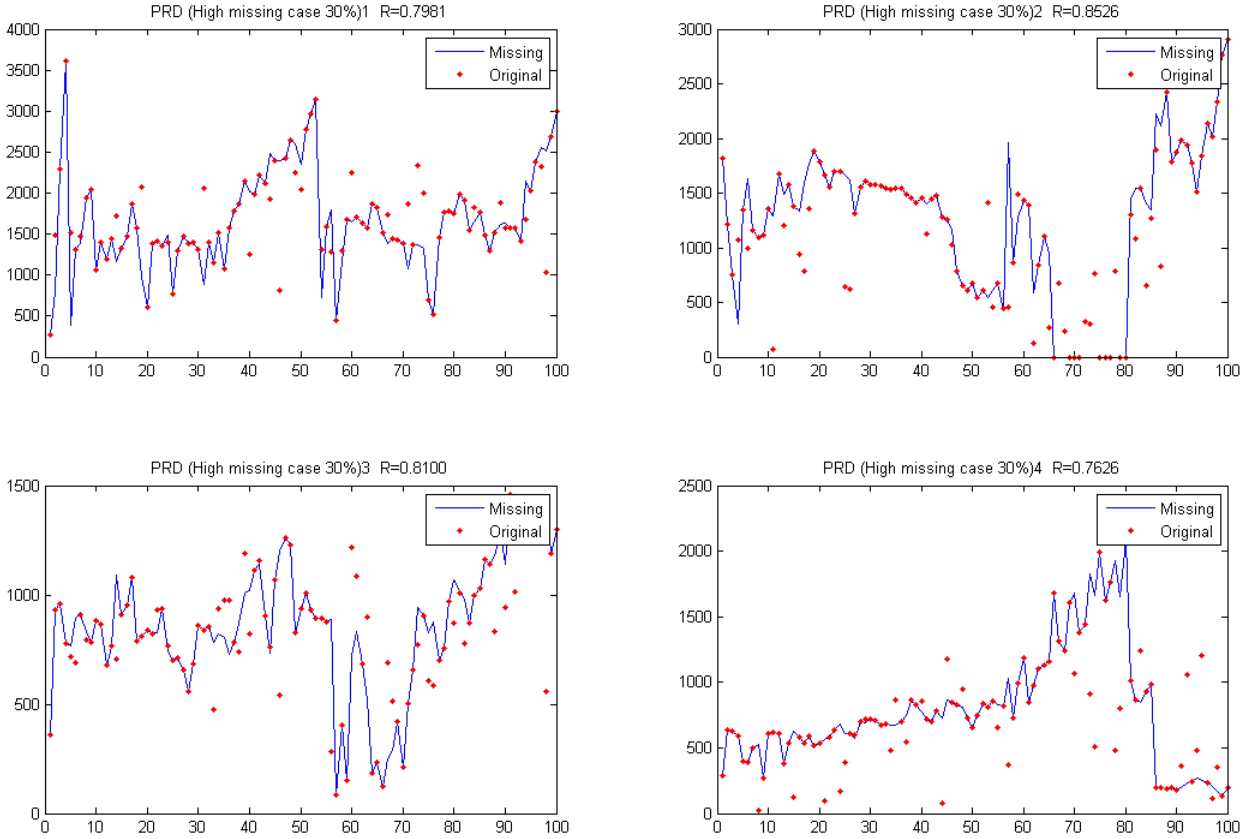


Figure 5.18: Horizontal Missing /Production and Injection with no possibility to extract good data (High missing 30% case)

5.3.3.5 Production only missing case

In this case a random pattern in the production data was created. The injection data remains intact. The strategy of solution here was simpler than for previous cases.

1. Can you extract good data? More than 40 data points at least as per (Sayarpour 2008).
2. If yes then use it to calculate the values for λ_{ij}.
3. If not, use averaging to construct usable production data and use it.
4. Impute and re-calculate.

One high and one moderate missing data rates were created with both good data extraction possibility and no possibilities to extract good data. Figures 5.19, and 5.20 show these two cases with the production rates plotted against the synthetically generated data. There is a very slight decrease in the R^2 for each well, but these changes are very slight and should not affect any perceived influence an injector has on a producer.

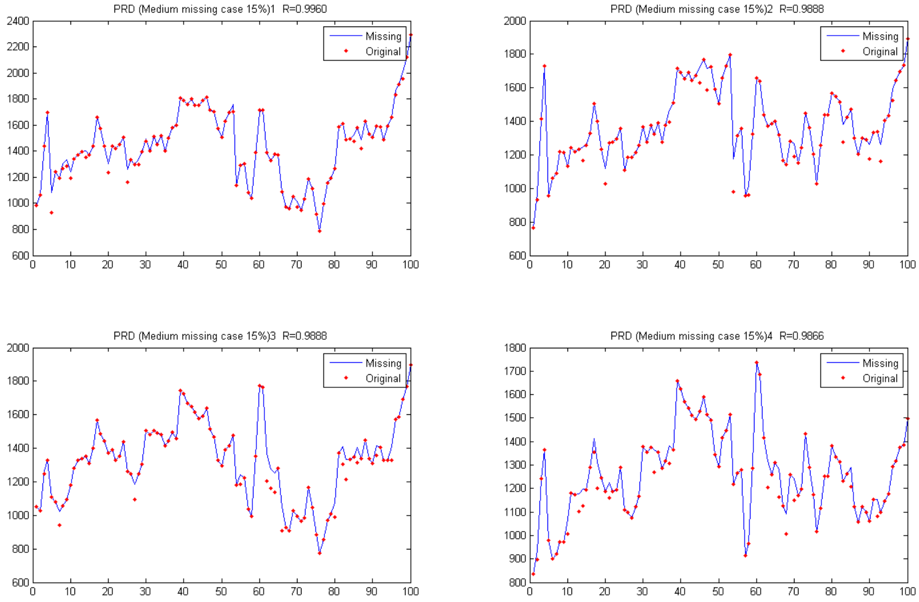


Figure 5.19: Production only missing case, Moderate missing case (15%)

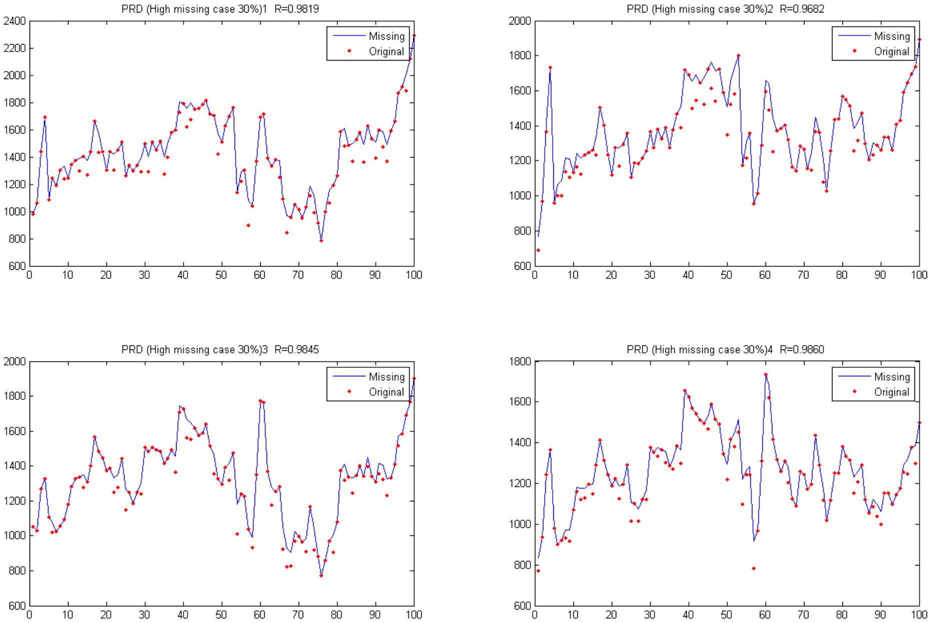


Figure 5.20: Production only missing case, High missing case (30%)

5.3.3.6 Injection only missing case -RM Model reversing

Assuming good production data the next case consisted of creating some random patterns of missing on the injection data side. The strategy used here is to try to extract some good usable data from the injection side. using Eqn. 5.5

It can be noticed here that imputing the injection data resulted in more deviation than imputing the production data. A 15% missing data case was tested and the results are shown in Figure 5.21. It can be seen that the algorithm yielded prediction values for the missing rates with R^2 higher than 0.8 in all 5 injection wells. A high missing ratio case of 30% was tested. This 30% missing rate case was tested to see weather the algorithm would be able to handle extreme cases. Figure 5.21 shows that with a moderate missing 15% the average value of R^2 obtained was around 0.95 and Figure 5.22 shows high missing rate for the injection.

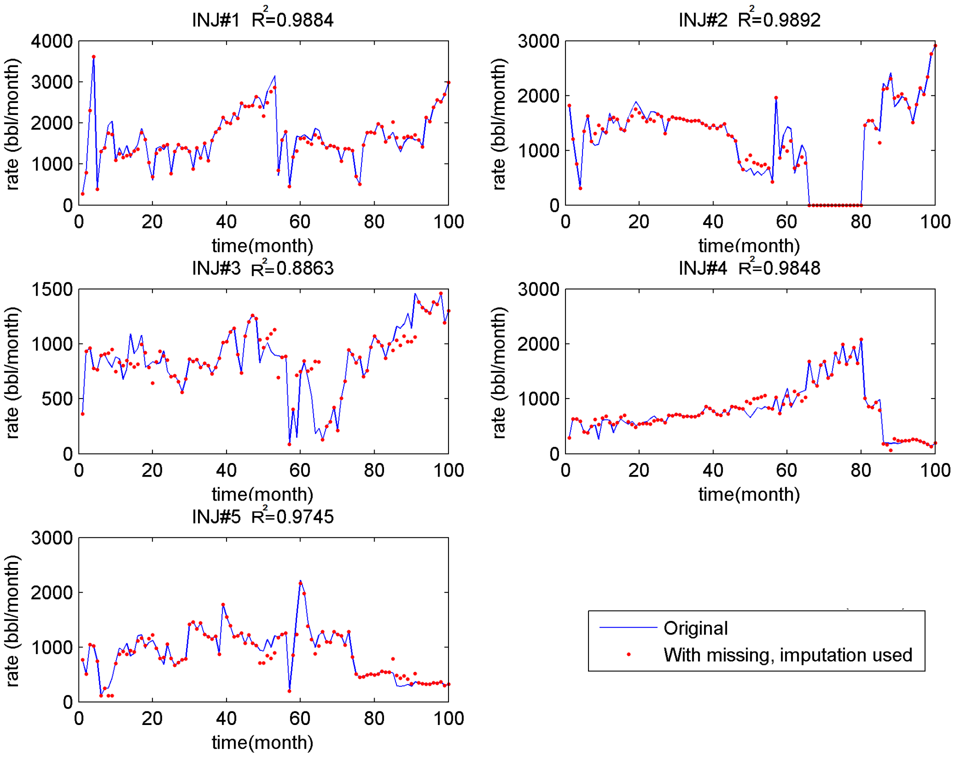


Figure 5.21: Injection only missing case, Moderate missing case (15%)

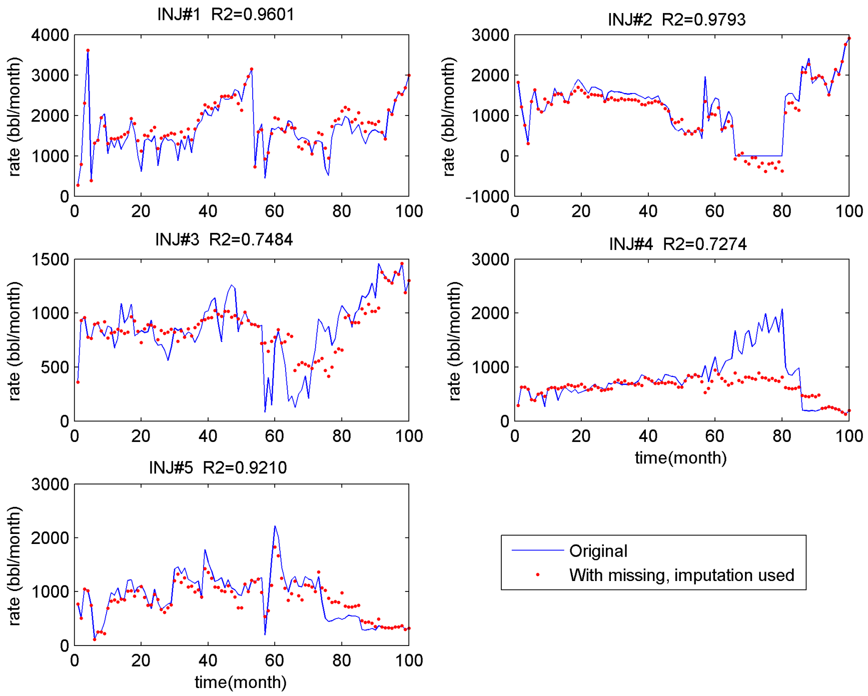


Figure 5.22: Injection only missing case, High missing case (30%)

5.4 Missing data treatment summary

In the previous sections treatment of missing input data has been investigated. Several patterns were presented based on scenarios that might happen at data acquisition time. These patterns were randomly generated in order to prevent any sampling biases that might affect the results. There are two ways to deal with the missing data: drop them or impute them. The preference between the two methods will depend on how much data is missing and where the data is missing; is it in the production data or the injection data? It has been found that missing more than 6% of the data is the limit for dropping data since the model R^2 drops very fast when the amount of missing data is more than 6%. The dropping technique has been investigated with each pattern and the most severe case was found to be when the modified multivariate missing pattern occurs. The imputation technique was investigated and several methods were introduced. Horizontal missing data patterns were discussed and two methods were introduced and discussed: using RM models as in equation 5.5 and rate averaging. A complete decision tree was created (Figure 5.16), coded and tested to simulate a number of possible scenarios and choose between them. The case used to test the results was a 5x4 case for all the attempted scenarios.

5.5 Corrupted data (outliers) detection

The definition of corrupted data here is similar to that stated by (Hawkins 1980) “an outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism”. In addition to missing data which are not there from the beginning, another form of input data problem is outliers. This case is more complicated than having missing data because there is no explicit approach on how to detect and treat the outliers. A significant part of the solution to this problem is coming from our understanding of the expected behavior within the dataset. Once an insight to the dataset is developed, the detection of outliers and a suitable model to adjust for them can be used. Most of the approaches to detect outliers depend on a multivariate form of the data. By measuring the change in the residual errors when a value is removed from the data set, a decision can be taken on whether this value is an outlier or not (Chawla and Sun, 2006); Kriegel et al. 2009; Franklin and Brodeur, 1997; (Mathworks b). The other part of these techniques are that they are sample dependent methods or model based on spatial proximity (Kriegelet al. 2009). In this thesis the model dependent procedures will be ignored because implementing a model from a problematic data set will not be useful in case they have to be used again for outlier detection. In this thesis we will use an assumed normal distribution of data to detect the outliers in addition to another sample dependent method to be discussed later.

5.5.1 Standard Deviation assisted method (STD)

It is safe here to assume the data is normally distributed when the number of values exceeds 30 according to the Central Limit Theorem (Feller,2008). The normal distribution has characteristics:

$$1\% - 99\% = \mu \pm 3\sigma \tag{5.6}$$

where μ is the mean of the distribution and σ is the standard deviation Figure 5.23 shows the probability distribution for the normal distribution

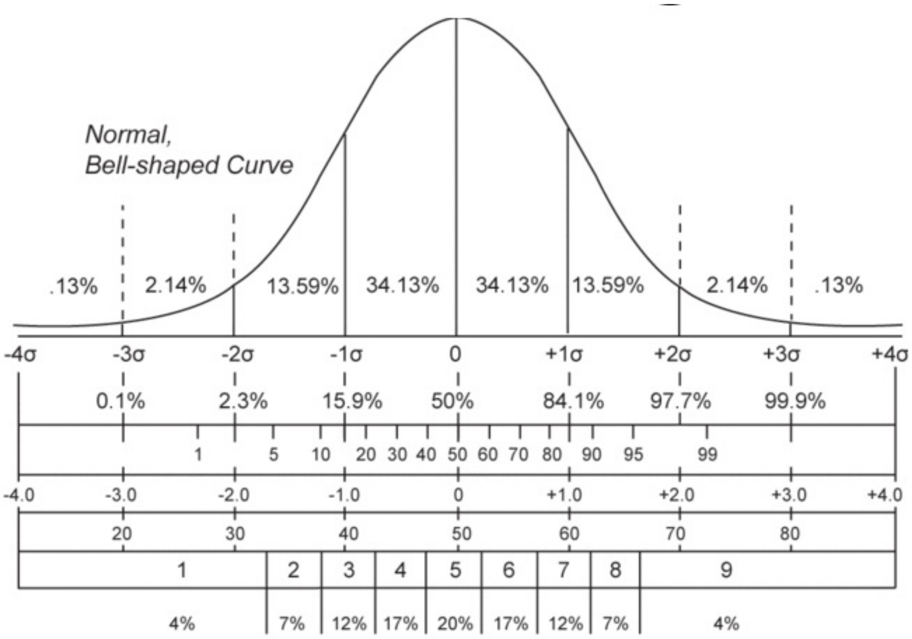


Figure 5.23: Normal Distribution (Close, 2014)

The standard deviation will be used to identify any data values that fall above or below $\mu \pm 3\sigma$. The proposed strategy is to flag the data points outside this range for further investigation; they cannot be removed unless a full understanding of the data set trends is achieved. The flow chart for missing data was modified to handle outlier detection and is shown in Figure 5.29. Also the method was coded as a part of the main code extensions. A set of contaminated data was inserted into the injection data in order to test this method. The code was modified to flag the potential outliers points. Two test cases were utilized: the first was to insert some low outliers (values less than $\mu - 3\sigma$) and the other case was to insert some high outliers (values more than $\mu + 3\sigma$). Figure 5.24 shows results for 4 injectors, the upper row in the figure was for high value outliers and it can be seen that the method was able to detect these potential outliers. The lower two rows in the figure show the testing for low outliers. It is good to know that these results are more warning signs than deleting justifications. Having these data points marked does not mean to start removing the data, but it is good to check these points as they are suspected to be potential outliers.

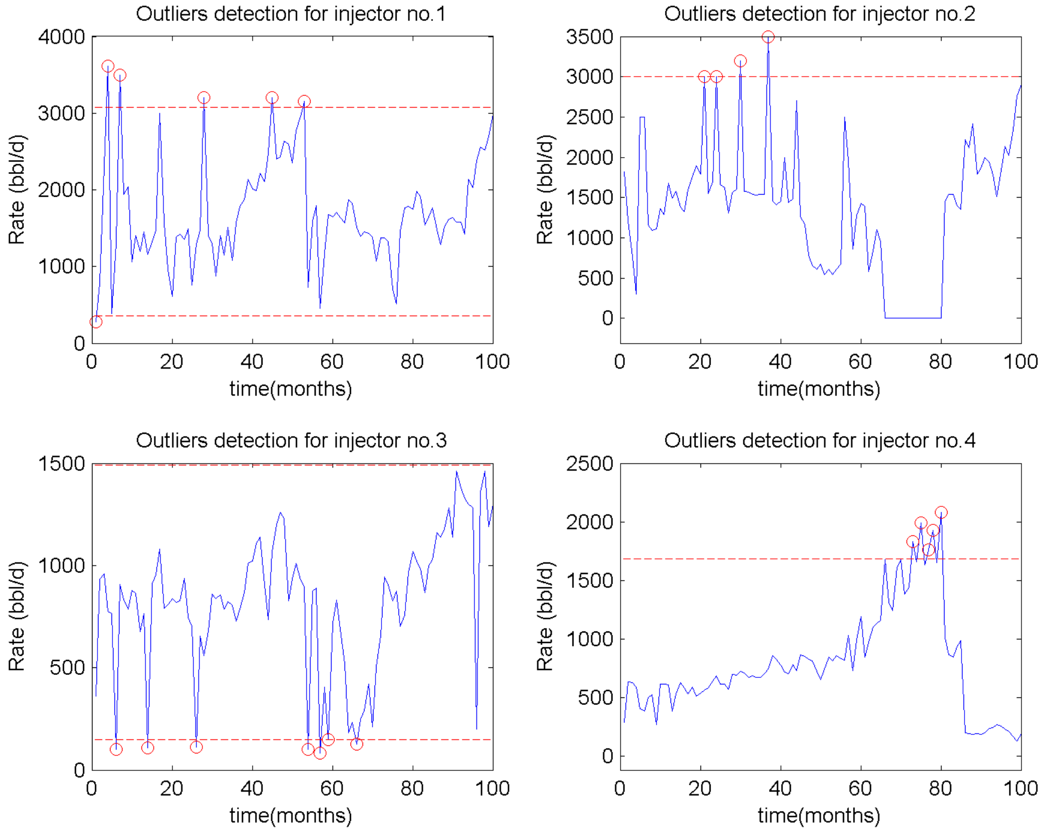


Figure 5.24: Standard Deviation assisted method (STD) results

5.5.2 Local Outlier Factor (LOF)

This method was suggested by Breunig et al. (2000) to decide whether a point is an outlier or not. The approach is by checking the “degree of isolation” of that point, or what they called the ”local outlier factor”. This procedure is done by measuring the density of a specific point in relation to the k-th neighboring points. A low density point suggests that this point is a potential outlier. In the cases presented here, the number of neighborhood points is 2 from each side. Figure 5.25 shows this approach. The neighborhood points are then 5 in the cases in this work, and the magnitude of the differences between the data points were calculated. Any outliers identified are replaced with NaNs and the methodology presented previously for dealing with missing data is applied to the revised dataset.

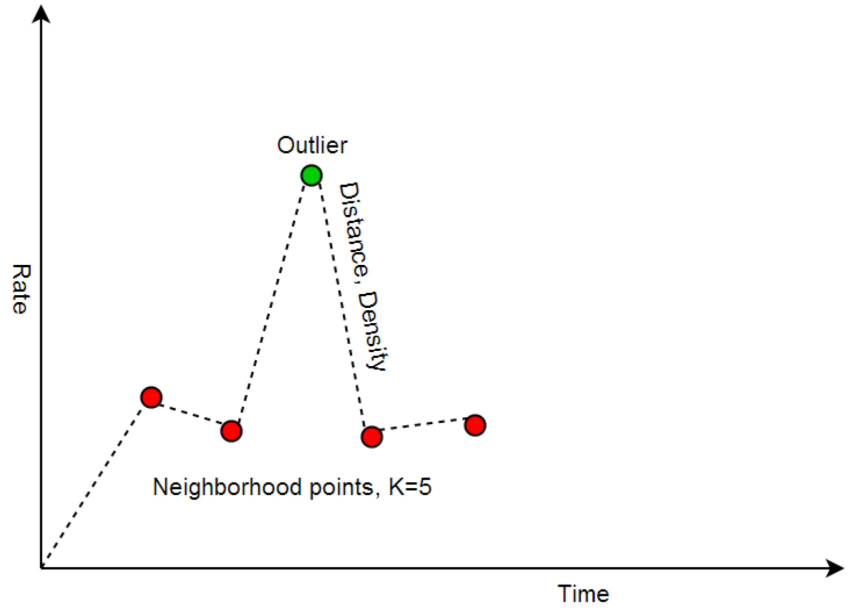


Figure 5.25: Local Outlier Factor (LOF) method definition

5.5.3 LOF mathematical development

Breunig et al. (2000) presented the following mathematical development for the LOF. As shown in Figure 5.26, one can define the k-distance (A) as the distance from point A to the k-th nearest neighbor. The set of k points will be noted as $N_k(A)$. Then the reachability distance between point A and the group of points $N_k(A)$ will be defined as:

$$Reachability\ distance_k(A,B) = max\{k - distance(A), d(A,B)\} \tag{5.7}$$

where:

$d(A,B)$ is the distance between A and B i.e. $|Y(A) - Y(B)|$

B $\in N_k(A)$. i.e. the reachability distance between two points A and B is the true distance and at least k-distance(B).

The local reachability density is defined by:

$$lrd(A) := \left(\frac{\sum B \in N_k(A)\ Reachability\ distance_k(A,B)}{n(N_k(A))} \right) \tag{5.8}$$

while the LOF factor will be:

$$LOF_k(A) = \frac{\sum B \in N_k(A) \frac{lrd(B)}{lrd(A)}}{n(N_k(A))} / lrd(A) \tag{5.9}$$

where $n(N_k(A))$ is the number of elements in $N_k(A)$.

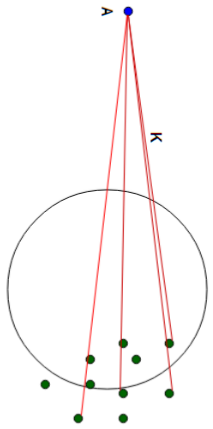


Figure 5.26: Basic idea of LOF, by comparing the density of a point with the that of its adjacent points

5.5.4 Testing and results

The same problematic data sets used in the standard deviation assisted method was used for testing the LOF method. Figure 5.27 shows the results of the LOF method. The k^{th} neighborhoods are set to 5 and the outliers were selected be the top-10 outliers which means it is 1% of the data.

It can be noticed here that LOF detected some of the neighbor points to the potential outliers in addition to flagging the actual outliers. In general the LOF method was able to detect all the outliers that were inserted into the data and were detected by STD. A superimposed plot in Figure 5.28 between the STD method and the LOF method shows the inserted outliers were detected by both methods. The highlighted circles are mutual outliers that were detected by both the STD and LOF.

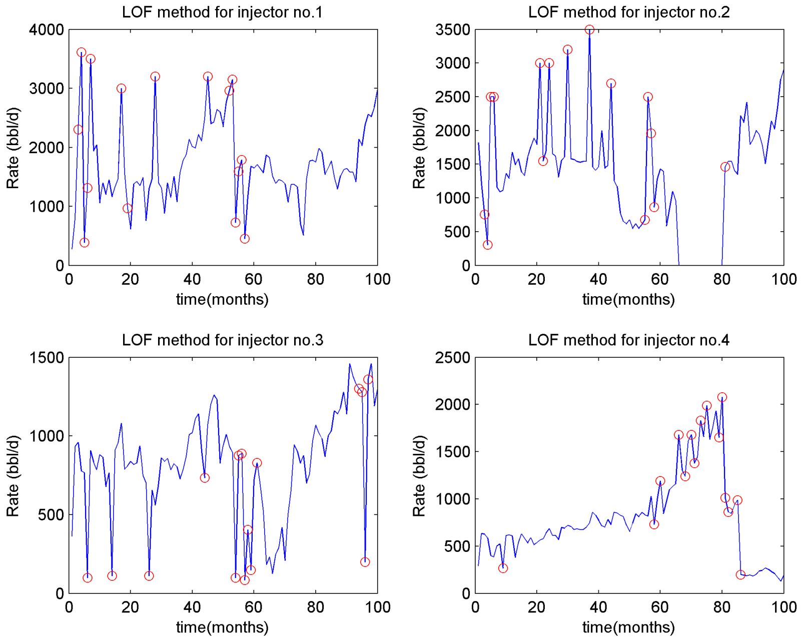


Figure 5.27: Local Outlier Factor (LOF) method results

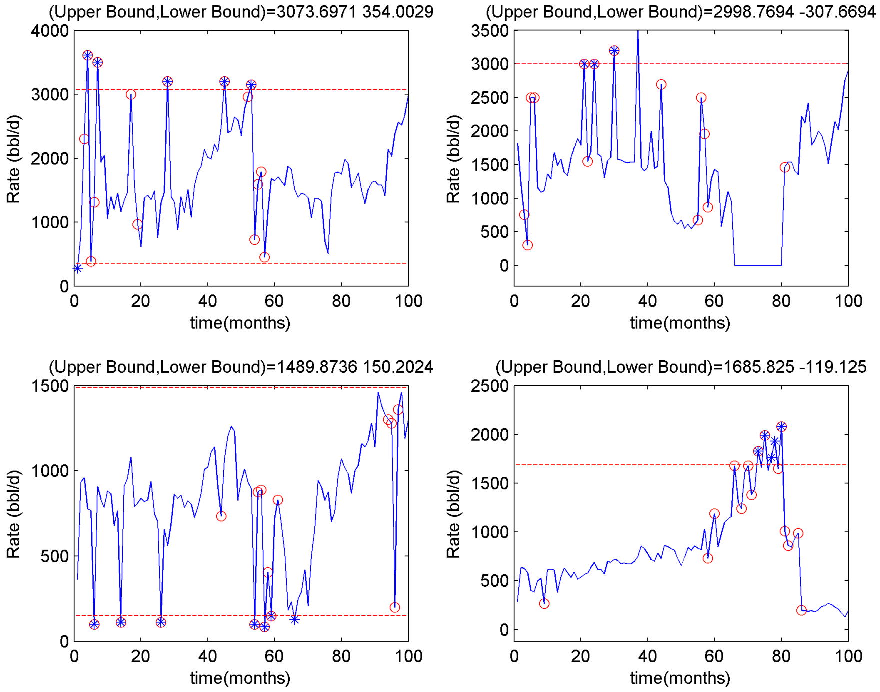


Figure 5.28: Comparison between LOF, STD results

5.6 Corrupted (outliers) data treatment and detection summary

In section 5.5 and its subsections the second big issue in input data (outliers) was discussed and two methods were investigated. Looking at the statistical literature there are mainly two approaches to deal with outliers: a modeling approach which assumes that the data is related in a multivariate linear model; and a sample dependent method which depends on learning from the data, or data mining. The modeling approach was not used in this work due to the fact that there is already problematic data so it will not be that useful to infer a model from such a dataset. Descriptive statistics are widely used in the data mining literature, so the standard deviation and mean will be calculated and conclusions can be inferred from that. Comparing each data point with the upper and lower limits of the distribution is what was tested in this work on a contaminated data set and results have been presented. Another sample dependent method known as the distances methods and is related to calculating how far every point is from its neighbor. Several methods could have been chosen but the Local Outlier Factor (LOF) method as suggested by Breunig et al. (2000) was selected. The method was implemented and tested on the same set of data and the results were compared to the STD method. One issue that this method has is detecting adjacent points to the outlier as outliers. This issue needs to be investigated throughly or other models may need to be implemented. Despite this issue the method was able to detect all the outliers that were inserted into the dataset. A complete flowchart in Figure 5.29 to account for the corrupted and missing data in both injection and production was coded in Matlab

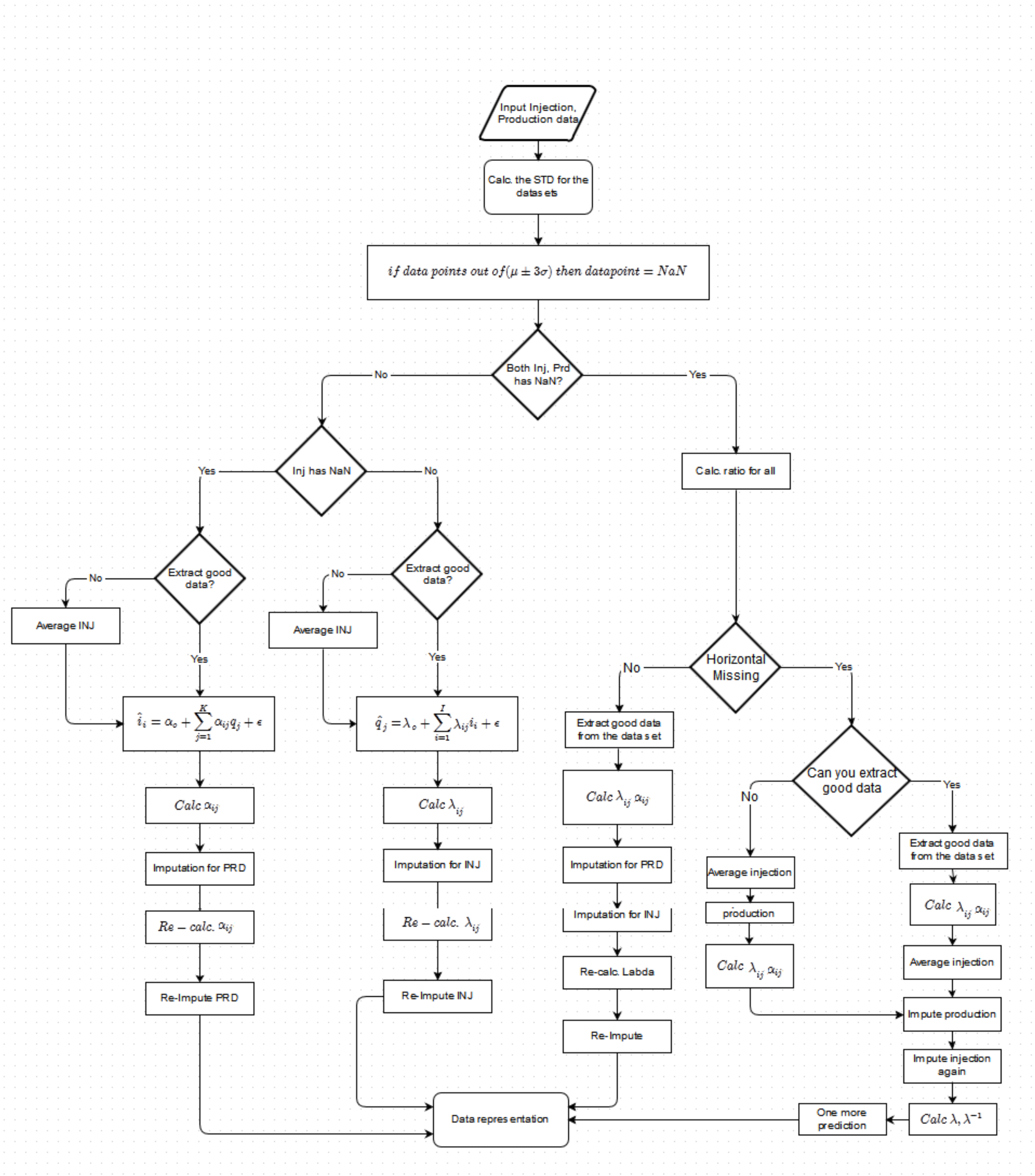

Input Injection, Production data
Calc. the STD for the datas ets
$if\ data\ points\ out\ of(\mu \pm 3\sigma)\ then\ datapoint = NaN$
Both Inj, Prd has NaN?
No
Yes
Inj has NaN
Yes
No
Calc. ratio for all
Extract good data?
No
Average INJ
Yes
Extract good data?
No
Average INJ
Yes
Horizontal Missing
No
Yes
$\hat{i}_i = \alpha_o + \sum_{j=1}^{K} \alpha_{ij} q_j + \epsilon$
$\hat{q}_j = \lambda_o + \sum_{i=1}^{I} \lambda_{ij} i_i + \epsilon$
Extract good data from the data s et
Can you extract good data
No
Yes
$Calc\ \alpha_{ij}$
$Calc\ \lambda_{ij}$
$Calc\ \lambda_{ij}\ \alpha_{ij}$
Extract good data from the data s et
Imputation for PRD
Imputation for INJ
Imputation for PRD
Average injection
$Calc\ \lambda_{ij}\ \alpha_{ij}$
$Re - calc.\ \alpha_{ij}$
$Re - calc.\ \lambda_{ij}$
Imputation for INJ
production
Average injection
$Calc\ \lambda_{ij}\ \alpha_{ij}$
Re-calc. Labda
Impute production
Re-Impute PRD
Re-Impute INJ
Re-Impute
Impute injection again
Data repres entation
One more prediction
$Calc\ \lambda, \lambda^{-1}$

Figure 5.29: Modified final flowchart for missing and outliers detection

CHAPTER 6
FIELD CASE STUDY

6.1 Introduction

To test the implemented simplified models on field data can only be done if the field data is available and can be utilized. Many operators consider rate and location information as proprietary and withhold publication rights to protect this data. An operator has provided the necessary information but has requested that we provide very limited description of the location to prevent its identification. The following provides the geologic setting in summary form and should provide a number of citations to literature provided by the operator or found from traditional literature searches. Rather than providing these directly, they will be listed as “Operator (2014)” in the citation below in order to protect the location information.

The field is a very complex, layered turbidite channel sand in the western United States. The channels are oriented roughly northeast-southwest. Structurally the field is an anticline with some faulting present near and slightly off the crest of the anticline. The anticline strike is roughly aligned with the channel direction and dip angles off the crest of the structure range from 20 to 60 degrees. The sands that make up the channel sequence are contained within a shale system and there does not appear to be a strong aquifer supporting the field (Operator, 2014).

6.2 Data preparation and analysis

The provided data contains injection and production rates for 189 producers and 65 injectors in addition to the well locations. The time period is nearly 40 years of monthly data. There have been a number of both producer to injector conversions as well as injector to producer conversions. The statistics of these conversions are shown in Figure 6.1 and were taken into consideration.

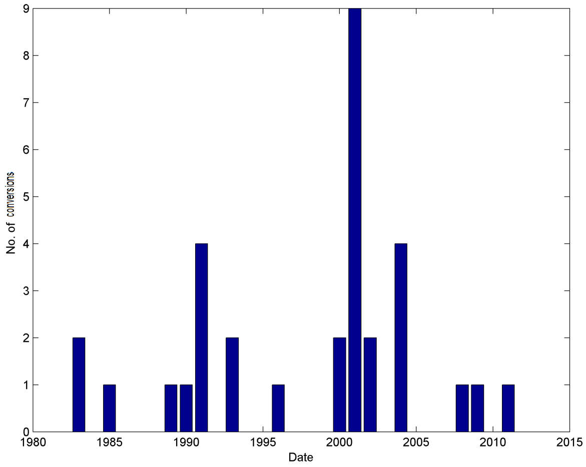


Figure 6.1: No of wells conversions per year

Figure 6.2 shows the well locations for the field along with a selected window for a simplified modeling effort. The window boundary was chosen by evaluating the data availability and cleanliness. The selected window contains 24 injectors and 31 producers. The number of data points available for the study was 156 monthly time intervals.

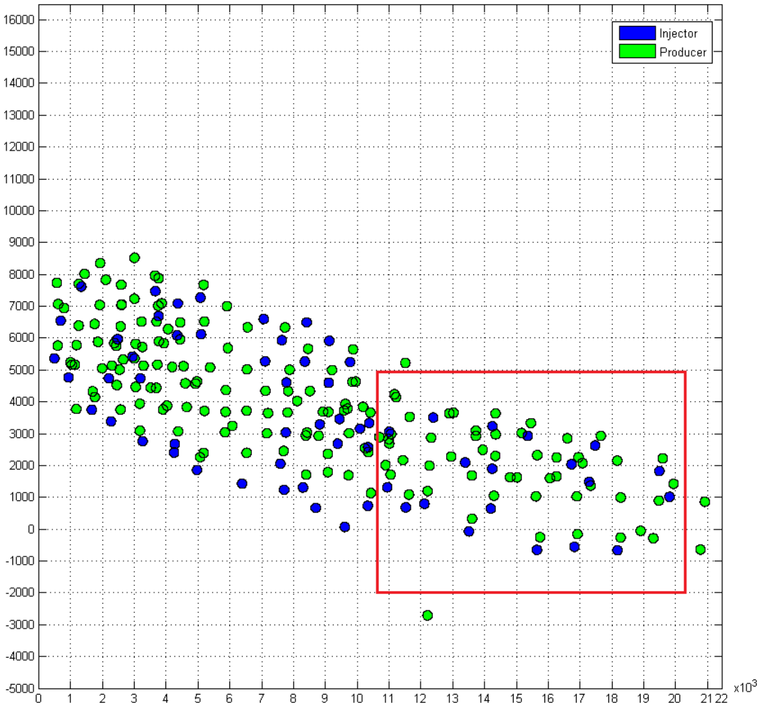


Figure 6.2: Wells selected for case study

Over the course of this time period there are many rate values that are 0. This might be indicating missing data or that the well was shut in for that month. These zero rate intervals are problematic especially for the RM model and were avoided as much as possible. Figure 6.3 shows the data availability vs. time with the chosen time span indicated. Figure 6.4 shows the total injection and production rates for the selected time interval. Injection exceeds production by as much as 100,000 bbl/month.

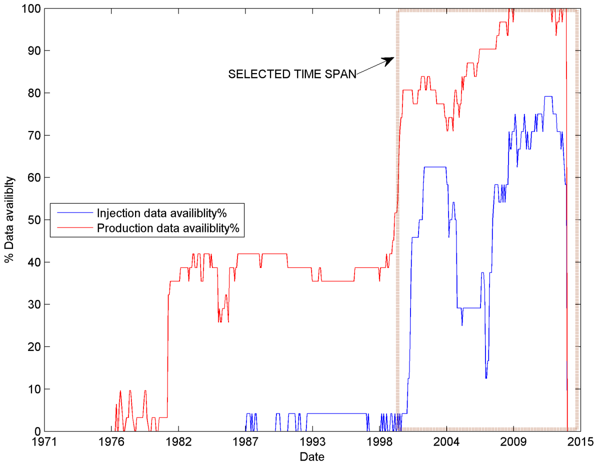


Figure 6.3: Time span selection for the selected case

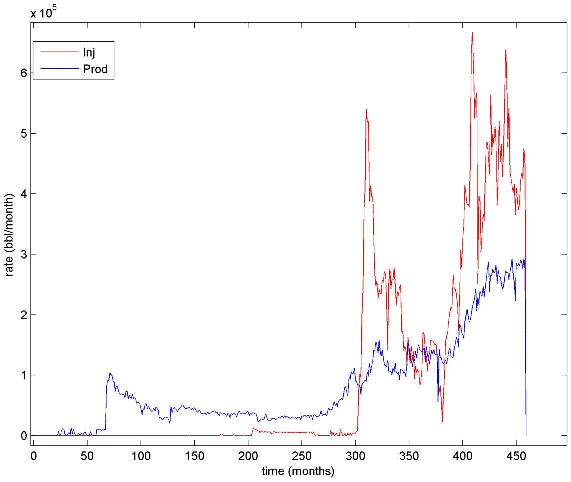


Figure 6.4: Sum of the injection and production for the selected span

The data has been checked for outliers and corruption using the two methods described in Chapter 5 and several outliers were detected. However, no action was taken to treat these points because there is not enough information about the injection and production schedule to determine whether these are within normal behavior. Figure 6.5 and 6.6 shows the data values flagged as outliers for the production values and for the injection data.

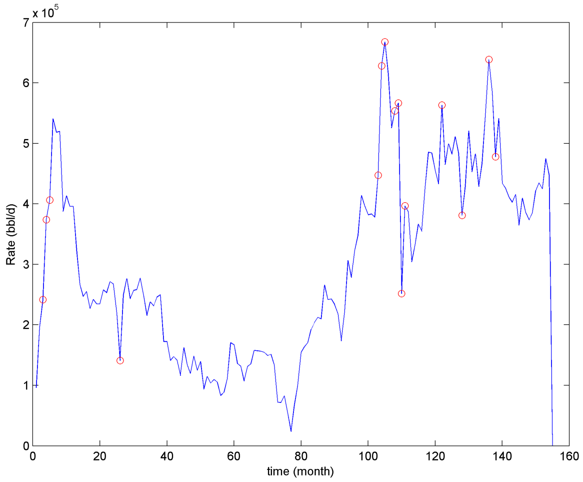


Figure 6.5: Outliers detection for production rates

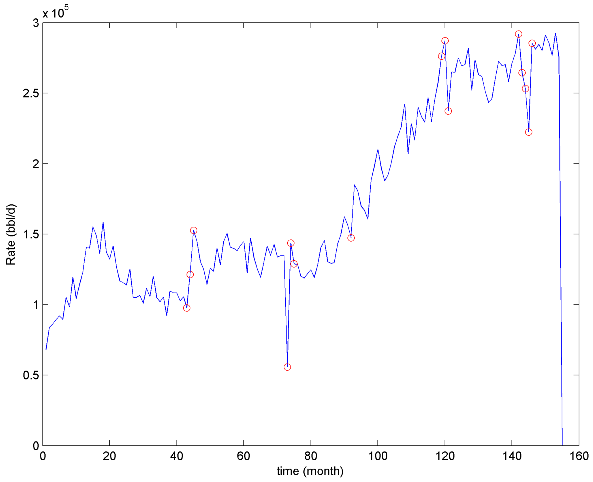


Figure 6.6: Outliers detection for injection rates

6.3 RM (unbalanced) model application

After the data preparation process, the RM model was applied to the window area and the resulting connectivities are shown in Table 6.1 and presented in Figure 6.7. For this case there were many negative connectivity values (Albertoni, 2003). This could be due to the chosen window and the severe imbalance in the injection and production. For large datasets there needs to be a technique to divide the field according to a specific criteria in order to reduce the number of coefficients per injector. The calculated values for the connectivities are used to predict the rates for the same time interval and the results were plotted in Figure 6.8. The results show $R^2 > 0.8$ in most of the simulated wells. In the unbalanced RM method, values of β_{oj} need to be obtained. These values are obtained using Eq. 6.1 and Eq. 6.2 and are shown in Table 6.2:

$$\hat{q}_j = \beta_o + \sum_{i=1}^{I} \beta_{ij} i_i + \varepsilon \tag{6.1}$$

$$\beta_{oj} = \bar{q}_j - \sum_{i=1}^{I} \beta_{ij} \bar{i}_j \tag{6.2}$$

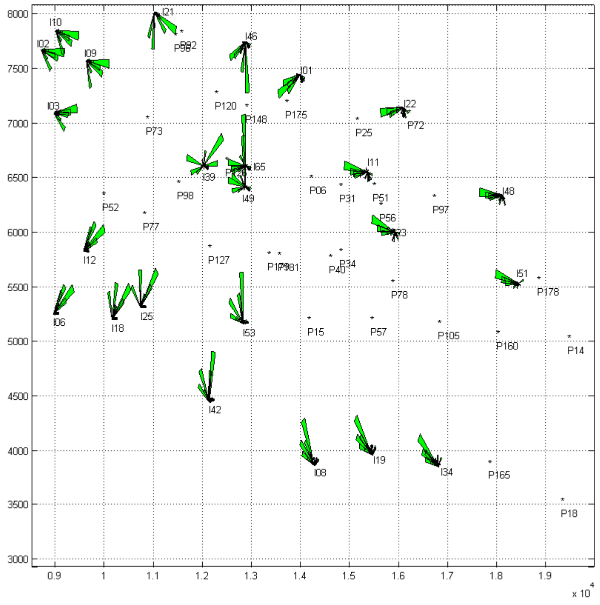


Figure 6.7: RM connectivities results

Table 6.1: Connectivity values for the field case using unbalanced RM

	P06	P14	P15	P18	P25	P31	P34	P40	P51	P52	P56	P57	P72	P73	P77	P78
I01	0.004	-0.016	-0.004	0.007	0.015	0.005	-0.012	-0.025	-0.011	0.087	-0.008	-0.030	0.009	-0.115	-0.050	-0.030
I02	-0.010	-0.007	0.011	-0.001	-0.001	-0.008	0.000	-0.006	-0.012	0.016	-0.005	0.007	0.005	0.023	-0.025	-0.021
I03	-0.056	-0.004	0.158	0.015	-0.009	-0.006	-0.039	-0.009	-0.026	-0.002	0.000	-0.005	-0.002	0.037	0.050	-0.026
I06	0.149	-0.172	0.524	-0.039	0.413	0.119	0.134	-0.211	0.053	-0.121	0.055	-0.062	0.013	0.513	0.553	-0.082
I08	-0.008	-0.026	-0.110	0.004	0.022	-0.015	-0.009	-0.036	0.011	-0.234	-0.001	0.011	-0.001	-0.113	-0.090	-0.013
I09	0.048	-0.005	-0.080	-0.009	0.002	0.017	0.036	0.054	-0.031	-0.033	0.014	-0.010	0.008	0.099	0.053	0.008
I10	0.012	0.006	0.005	0.000	0.007	0.011	0.003	0.000	0.027	0.022	0.004	-0.006	-0.001	0.012	0.062	0.025
I11	-0.073	-0.020	-0.011	-0.014	0.068	-0.018	-0.023	-0.018	0.063	-0.047	0.018	-0.002	-0.011	0.118	-0.109	0.015
I12	0.009	0.006	0.080	0.002	0.062	0.010	0.022	0.027	-0.007	0.265	0.009	-0.009	0.001	0.086	0.091	0.015
I18	0.072	0.022	0.100	0.014	-0.027	-0.009	0.004	0.003	-0.015	-0.028	-0.012	0.031	0.001	-0.018	0.200	-0.009
I19	-0.006	0.047	-0.181	-0.013	0.042	0.002	-0.011	0.008	-0.047	-0.178	0.019	0.006	-0.018	0.010	0.076	0.008
I21	0.018	0.036	-0.272	-0.002	0.029	-0.004	0.002	0.019	0.000	-0.135	0.030	-0.003	-0.010	0.098	0.071	0.027
I22	0.101	-0.028	-0.633	-0.003	-0.215	0.063	0.072	0.080	0.122	-0.055	0.074	0.118	0.007	0.081	0.473	0.210
I23	0.036	0.017	0.120	-0.004	0.025	-0.009	-0.004	0.023	0.046	0.046	-0.007	-0.004	0.000	-0.065	-0.076	0.084
I25	0.028	-0.021	0.144	0.010	0.038	-0.012	-0.017	0.024	0.001	0.037	-0.014	0.000	0.003	-0.001	-0.097	-0.008
I34	-0.006	0.009	0.078	0.013	-0.086	-0.016	-0.016	0.024	0.010	0.275	-0.027	0.008	0.004	-0.092	-0.039	0.004
I39	0.002	0.006	0.205	-0.011	0.093	-0.003	0.005	0.014	0.001	0.033	0.011	0.001	-0.005	-0.014	0.037	0.059
I42	-0.042	-0.020	-0.039	-0.016	0.019	0.021	0.015	-0.013	0.000	-0.045	0.018	-0.026	0.000	0.077	-0.064	0.000
I46	0.077	0.030	0.032	0.006	0.039	0.025	0.031	0.024	0.003	0.049	0.021	0.025	0.000	0.169	0.133	0.026
I48	-0.012	0.002	0.255	0.006	-0.038	0.008	-0.001	0.023	-0.011	-0.041	0.001	0.018	0.014	0.005	-0.036	0.015
I49	0.025	0.012	-0.001	0.012	-0.045	0.015	0.002	-0.027	-0.010	0.079	0.015	-0.010	-0.003	0.066	-0.006	0.002
I51	0.068	-0.040	0.057	0.020	0.121	0.011	0.015	0.008	0.029	0.116	-0.009	0.018	0.017	0.044	-0.062	0.088
I53	0.035	0.003	-0.023	0.007	0.015	0.009	0.023	-0.002	0.015	0.043	-0.001	0.008	0.004	-0.018	0.066	-0.001
I63	0.089	0.012	-0.390	-0.011	0.158	0.055	0.052	-0.009	-0.102	0.074	0.052	0.150	-0.002	0.070	0.123	0.019

Table 6.2: Example for β_o values for the field case

P06	P14	P15	P18	P25	P31	P34	P40	P51
6442.701	619.1948	5544.924	145.1128	3941.68	1057.728	2221.96	461.5375	1083.117
P56	**P57**	**P72**	**P73**	**P77**	**P78**	**P92**	**P96**	**P97**
1499.525	2129.629	-190.42	9064.471	8854.427	169.8024	860.848	3015.466	1365.098

Figure 6.7 shows the unbalanced RM (UBRM) rose diagram for the same data.

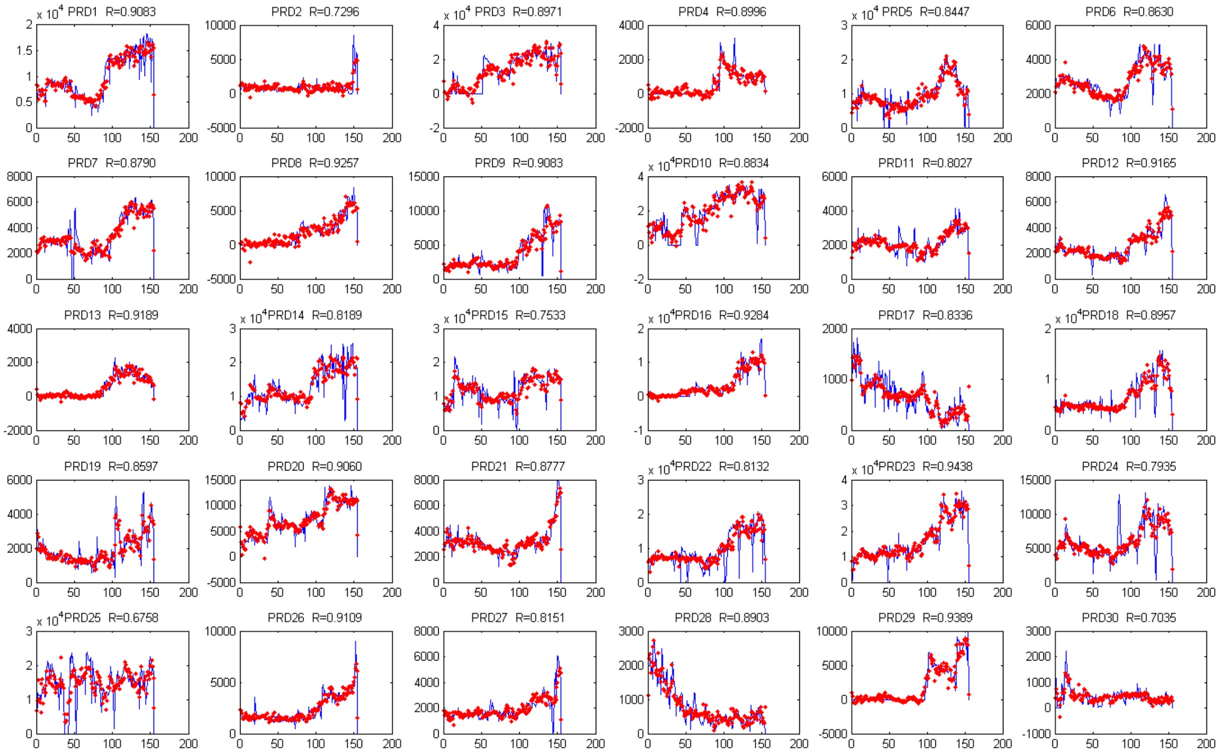


Figure 6.8: Production rates prediction for the field case

Note that there are some peaks in the actual data that were not represented by the model. These peaks were also previously identified as outliers. That might suggest that the method used to identifying outliers has found actual outliers. Note also that the injection trends seen in the rose diagram generally follow a northeast-southwest trend. This would indicate that injection follows the stated direction of the channel sands as well as the direction of the anticline. All that can be said with certainty is that there is a preferential flow pattern in the southeast-northwest direction according to the analysis of the connectivity values.

6.4 Full field modeling

Full field unbalanced RM modeling was also run for all 65 injectors and 189 producers. There were again 153 months of data for each well. The time required to obtain the modeling parameters (λ_{ij}) was 1.5 seconds with 49% of the λ_{ij} values being negative. Figure 6.9 shows connectivity values for all wells on a rose diagram. This diagram again shows the strong northwest-southeast injection influence and it also shows a group of wells surrounded by injectors. The producers in the central part of this region receive strong support from the injectors, but the peripheral wells do not.

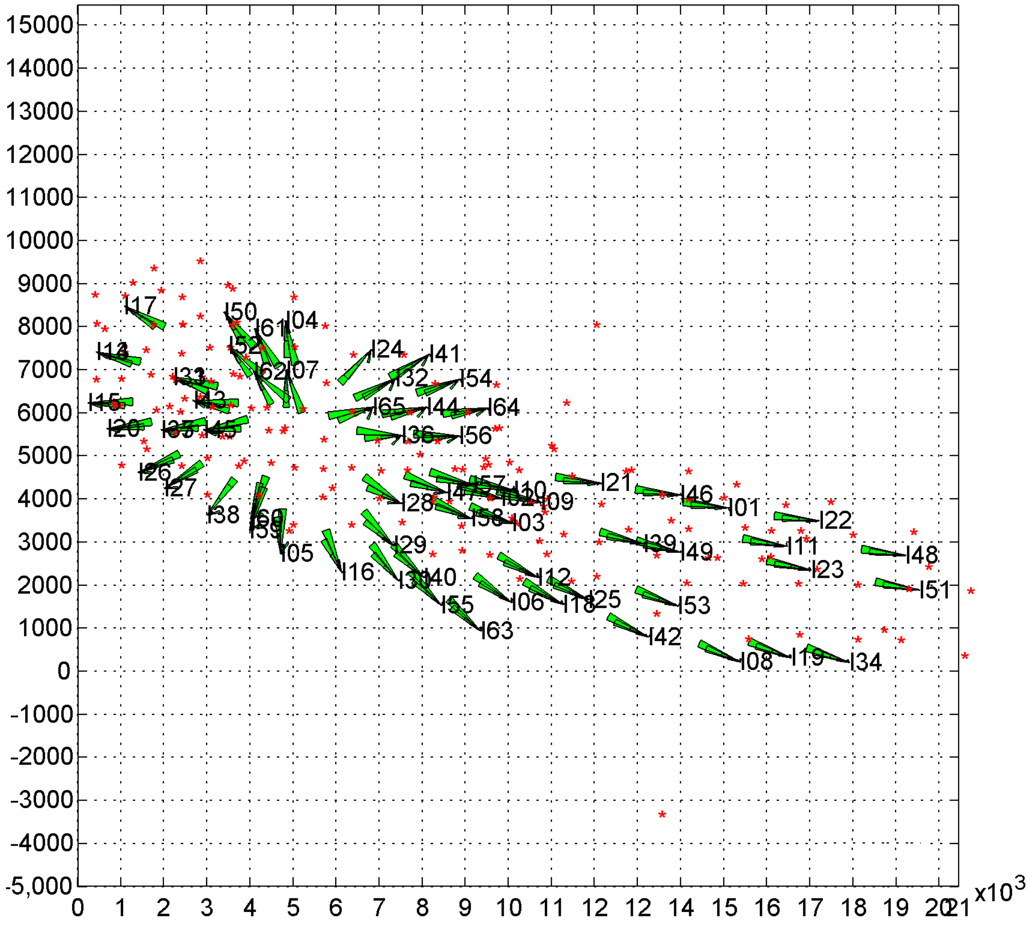


Figure 6.9: Rose diagram for full field modeling using RM model

The negative connectivity values for each injector were plotted spatially using a bubble map (Figure 6.10). It can be seen that the most negatively influencing injectors i.e. the ones that have the largest number of negative connectivity values, are located in the middle and the southwestern edge of the field, i.e. (I57, I28, I58, I09 and I39). Moderate to low negative connectivity value injectors are located on the southeastern edge and in areas where there are few close surrounding injectors. Negative connectivity values have generally indicated flow barriers or boundaries (Albertoni, et al, 2003; Oqunyomi, 2009).

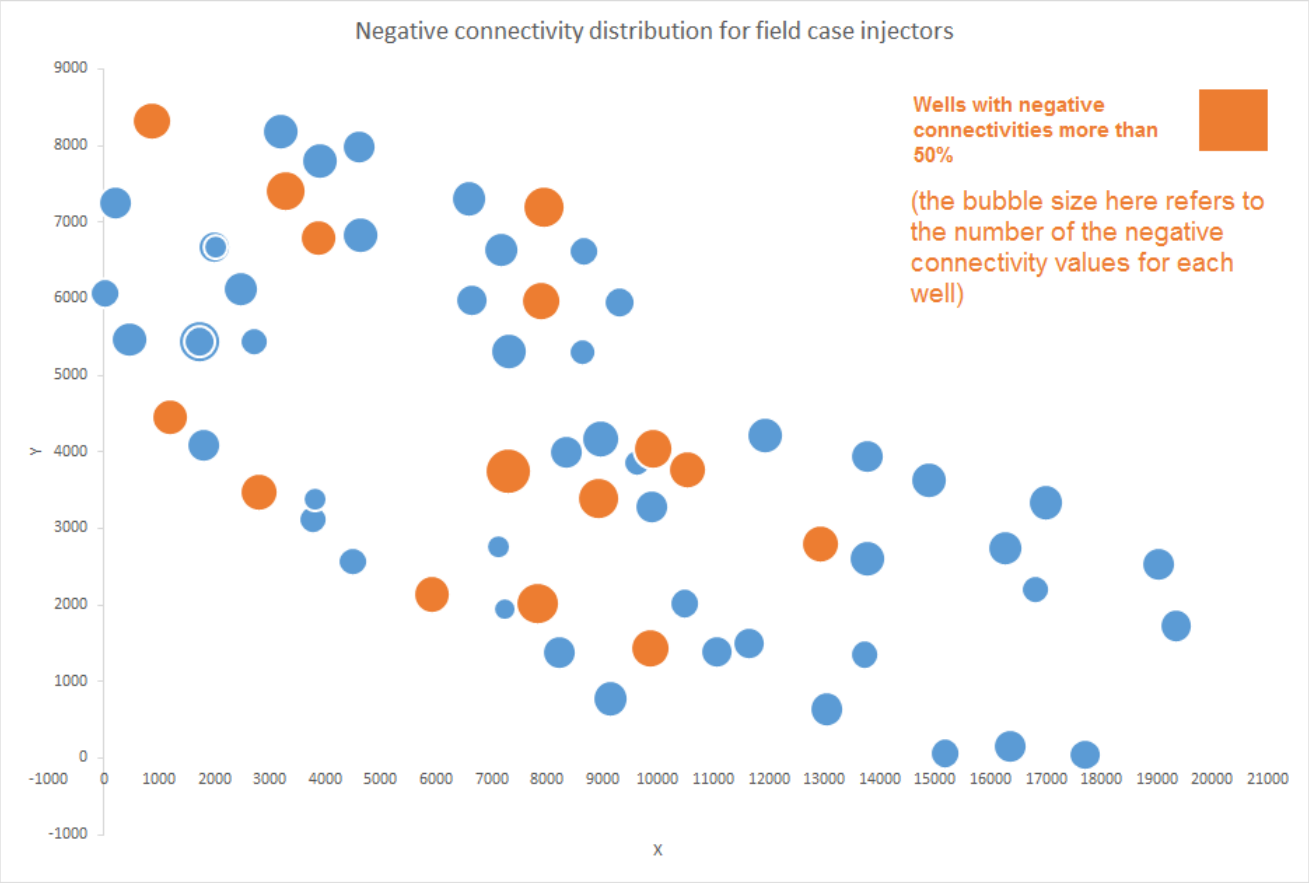


Figure 6.10: Map of negative connectivity values per injector for full field modeling (no. of negative connectivity/ Total connectivities)

The influence of the negative connectivity values as compared with the positive influence of injectors is relatively small. Figure 6.11 shows negative connectivity values in red and positive values in green. It can be seen that positive values are generally larger than the negative values.

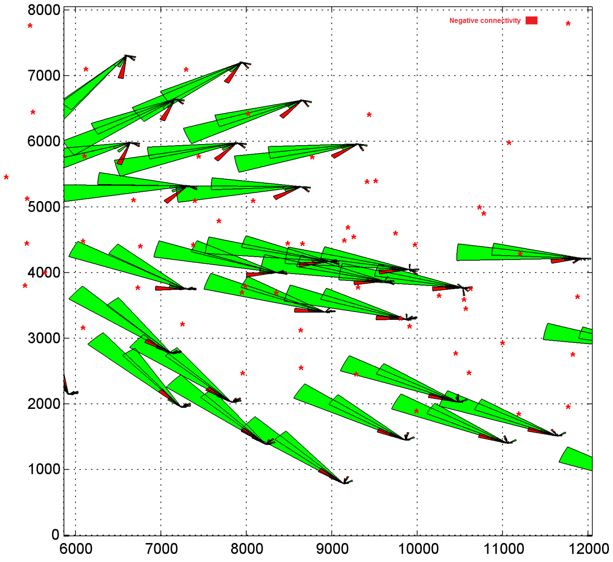


Figure 6.11: Negative connectivity values presented with positive values

The trend of negative connectivity values follows the same trend of the positive values from southeast to northwest. Figure 6.12 shows the negative values only for the full field case.

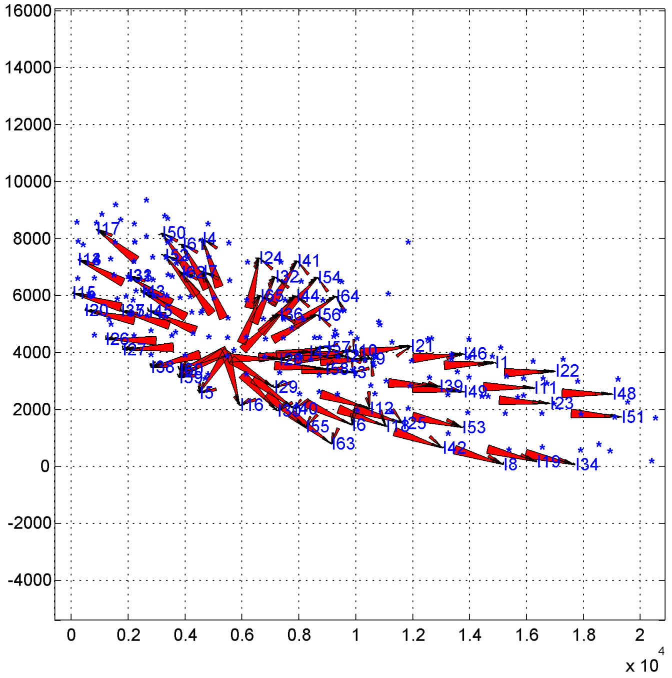


Figure 6.12: Negative connectivity values only for full field case, the values are scaled up to show the trend

6.5 Negative connectivity values treatment

The process to eliminate the negative connectivity values was suggested by Albertoni et al. (2003). Figure 6.13 shows a flowchart for eliminating both negative values and values that are larger than 1. These steps are:

1. Input the connectivity values matrix
2. Input actual production and injection rates values.
3. Input predicted production rates $\hat{q}$ values.
4. The following procedure will be performed for every element in the connectivity value matrix
    - 4.1 if the element is less than 0 or greater than 1
        - 4.1.1 Set this element=0
        - 4.1.2 Calculate the production values $\hat{q}$ using the updated matrix of connectivity values and the injection rate values.
        - 4.1.3 Set the actual production values = $\hat{q}$ in step 4.1.2
        - 4.1.4 Calculate connectivity matrix values again using the new actual production values and the injection values.
5. Again calculate predicted production $\hat{q}$
6. Calculate R^2 for each producer
7. Plot $\hat{q}$ vs. actual production rates

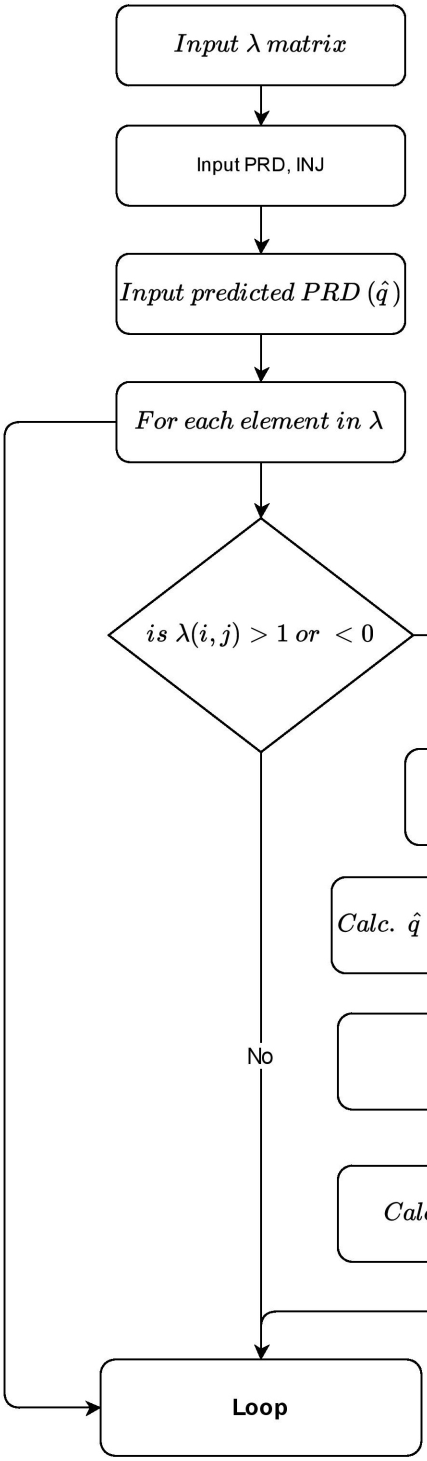

$Input\ \lambda\ matrix$
Input PRD, INJ
$Input\ predicted\ PRD\ (\hat{q})$
$For\ each\ element\ in\ \lambda$
$is\ \lambda(i,j) > 1\ or\ < 0$
Yes
$\lambda(i,j) = 0$
$Calc.\ \hat{q} = PRD(INJ, \lambda(i,j)$
No
$PRD = \hat{q}$
$Calc.\ \lambda(PRD, INJ)$
Loop

Figure 6.13: Process for elimination negative connectivity values and values larger than one

It has been noticed that once the negative connectivity values are set to zero, these values will appear back as a very small numbers (1^{-10}) when the matrix is recalculated and positive values will have increased during the course of the elimination. Figure 6.14 shows the the decline in R^2 for the field production prediction throughout the course of elimination. The decline looks drastic due to the chosen scale, but R^2 is still greater than 93% after eliminating the negative values.

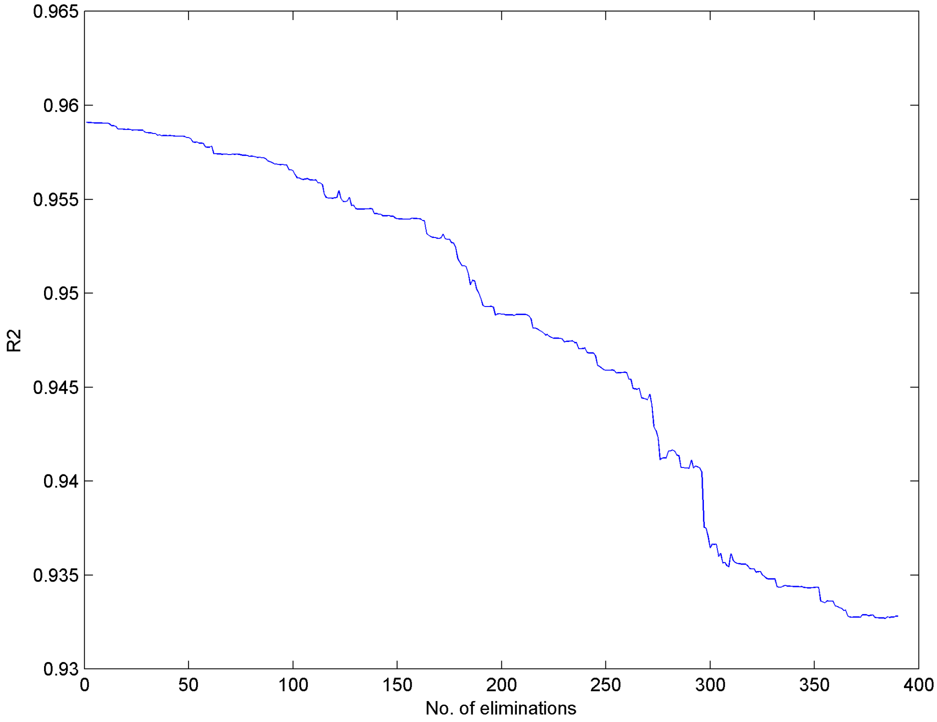


Figure 6.14: Coefficient of determination R^2 vs. number of droppings

This decrease is for total actual field production vs. total predicted field production. A higher decrease in R^2 is expected for each well that is removed. Figure 6.15 shows the connectivity values after the process of elimination. It can be seen that the flow trends are very similar (compare vs. Figure 6.9).

6.5.1 RM model field rate representation

The resulting values for the connectivity were used to predict the total field rate in order to compare it with the actual total field rate (Figure 6.16). The resulting R^2 was 0.868 which is 2.4% less than the average R^2 obtained for individual wells.

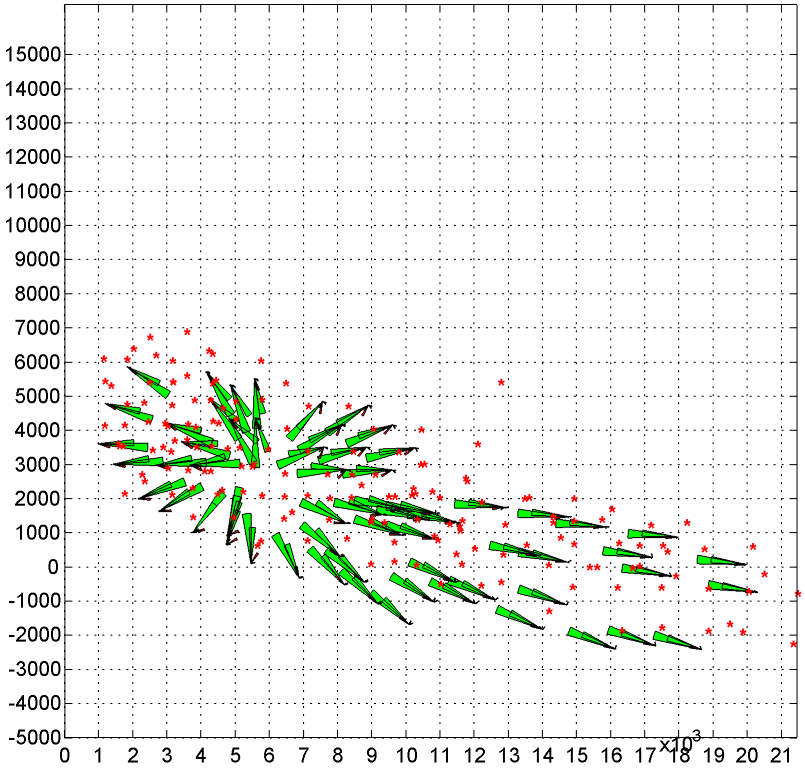


Figure 6.15: Field case connectivity values after negative connectivity values elimination

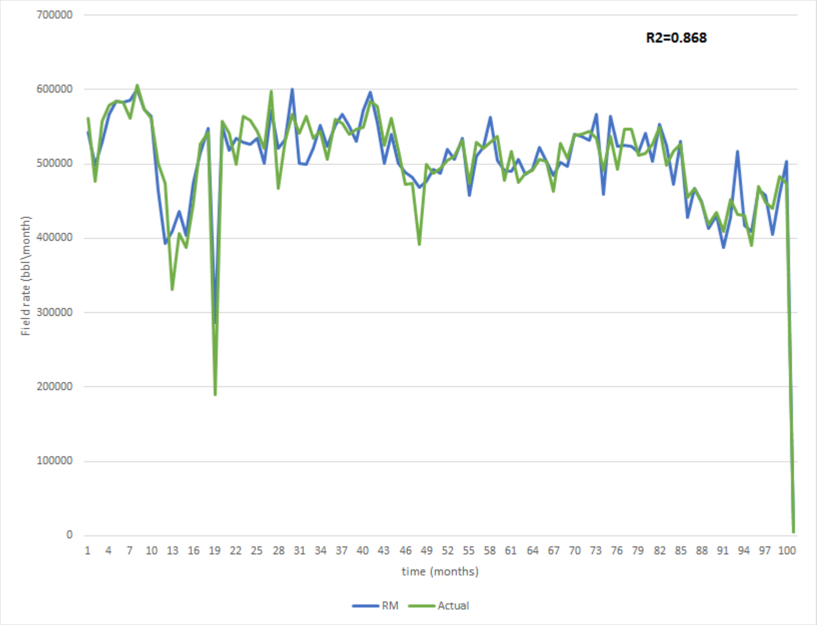


Figure 6.16: Predicted total field rate vs. the actual total field rate for the RM model

6.6 CRM model application

The same data points used in the RM model predictions were used to run the CRM model for 189 producers and 65 injectors. The implemented method in section 3.2.5 was run. The running time was around 30-40 min. The resulting parameters for the connectivity values and time constants were less representative of the actual data than the RM model. Several starting values for the time constant were tested and the final value for the time constants were not changing. The maximum value for R^2 was 0.787 and the average was 0.18 with 19.5% negative connectivity values. Figure 6.17 shows the rose diagram for the field case using the CRM model and there are some differences as compared with the RM result. First, there is a general angle change to the injection arrows. In the RM result injectors seem to be supporting a group of wells in a roughly radial pattern in the central portion of the northwest quadrant. In the CRM result, the arrows in the northwest quadrant point in a slightly different direction and the injection wells that separate the northwestern quadrant from the southeastern quadrant point to the southeast rather than the northeast as in the RM result. Second, most of the wells seem to be pointing more towards wells along the northern edge of the southeastern part of the field and possibly even “out of zone”. In both cases (RM and CRM) the arrows point along the northwest-southeast trend consistent with the depositional and structural directions.

6.7 CM model application

Using the same case data, the CM model was also run for all the 189 producers and 65 injectors using the method described in section 3.2.5 and the running time was around 55 min. The resulting parameters for the connectivity values are shown in Figure 6.18. There is 43.9% of negative connectivity values and the maximum values for R^2 was 0.89 with an average of 0.166. Figure 6.18 shows that flow trends especially in the southeast quadrant agree with what the RM results showed but the injection wells in the northwest part of the field show a trend that is different from similar wells in the RM results. Many of the injectors in this northwest quadrant are showing that injection along the north side are supporting wells more towards the far northwest corner rather than more towards the central part of this quadrant as in the RM result which support wells more towards the center.

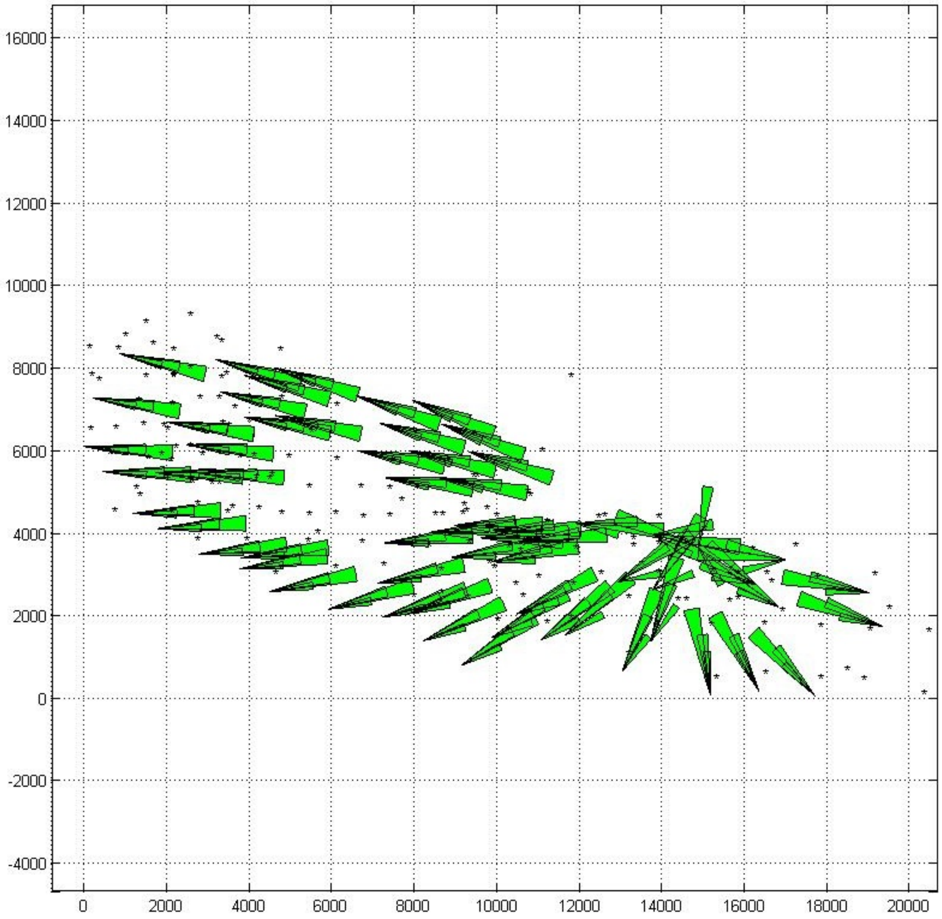


Figure 6.17: CRM model rose diagram for field case

6.8 Models comparison

6.8.1 R^2 values

The values of R^2 distribution for the three models in Figure 6.19 shows that RM resulted in the best R^2 values and the CM and CRM models respectively come after the RM model. These values are averaged and shown in Figure 6.20

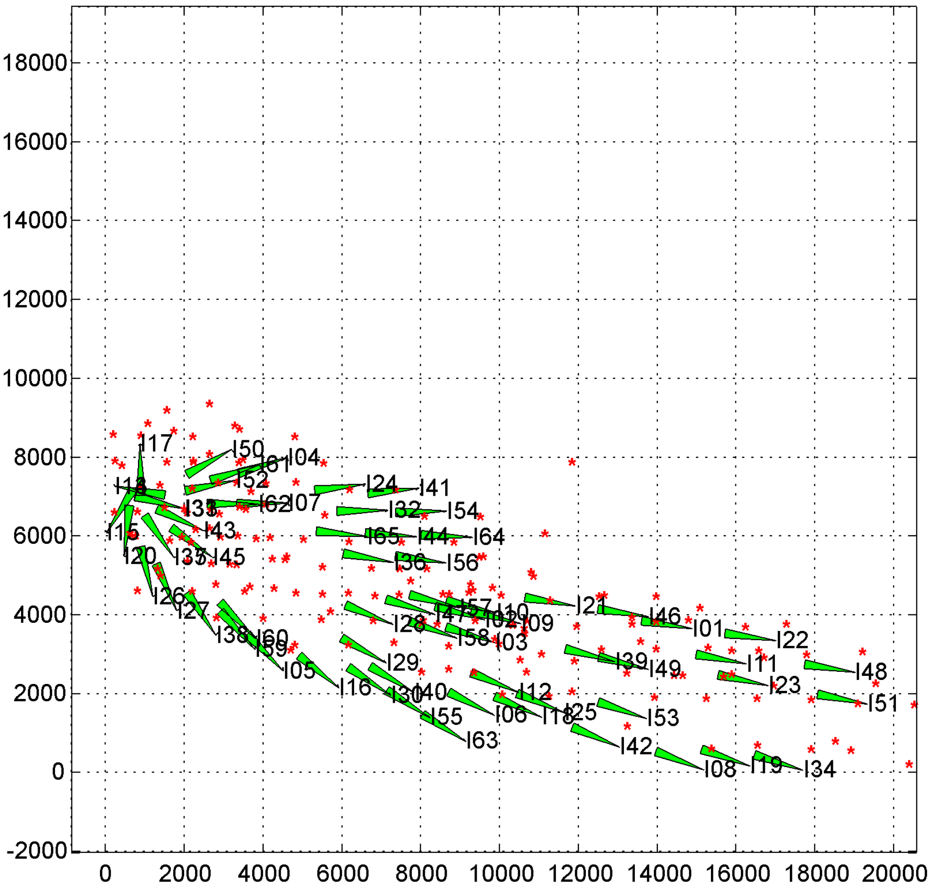


Figure 6.18: CM model rose diagram for field case

6.8.2 Negative connectivity values

All the three models yielded negative connectivity values, Figure 6.21 shows the RM and CM model ratios of negative connectivity values are close 49% and 43.9% respectively, the CRM model resulted in 19.5% negative values.

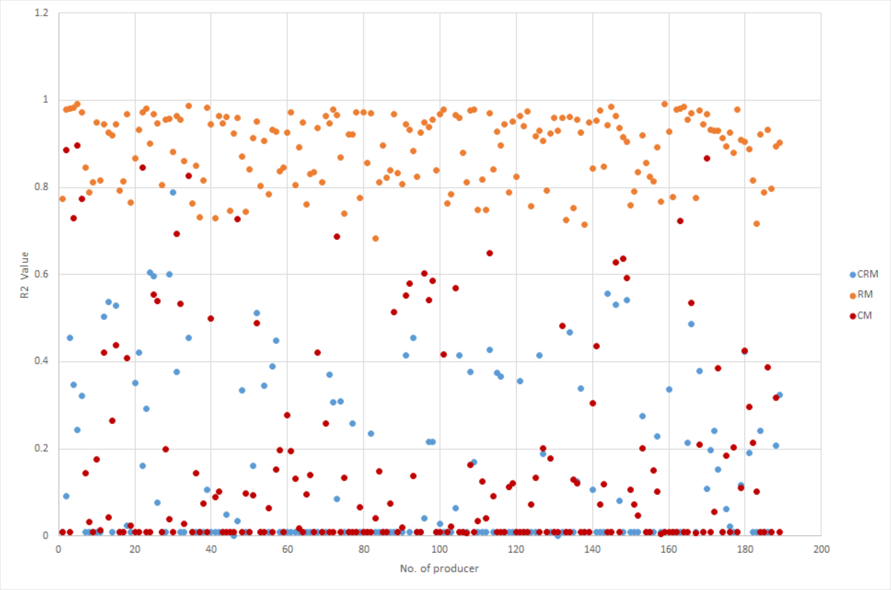


Figure 6.19: R^2 distribution for all the three models for the field case

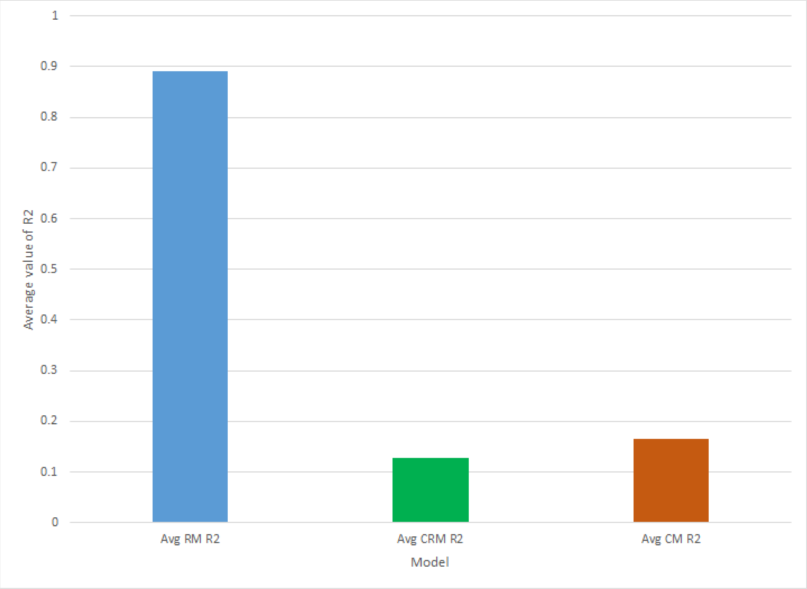


Figure 6.20: R^2 averages for all the three models for the field case

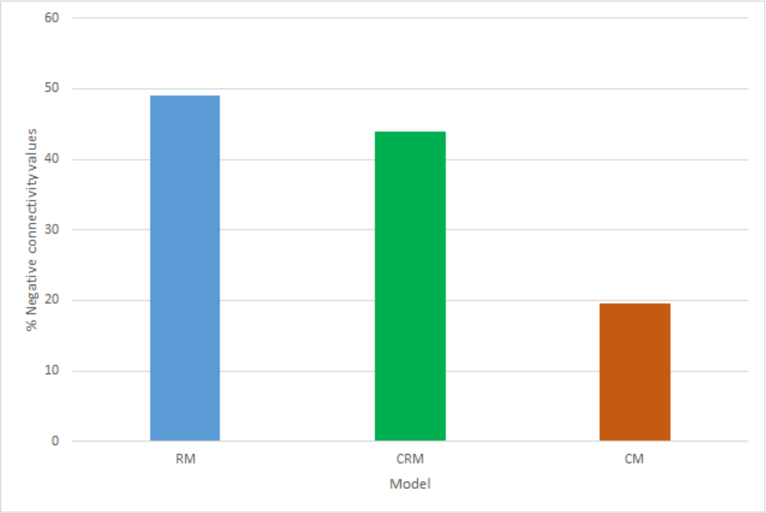


Figure 6.21: Negative connectivity values for RM, CRM, and CM models

6.8.3 Total field R^2

The connectivity values and time constants for both the CRM and CM model were used to predict the total field rate over the same modeling time period and compared with the actual total field rate. Both the CM and CRM models show a very low R^2 values for the total field rate and it is close to the obtained average R^2 values in both models. Figure 6.22 shows a comparison between the three models R^2 for the total field rate.

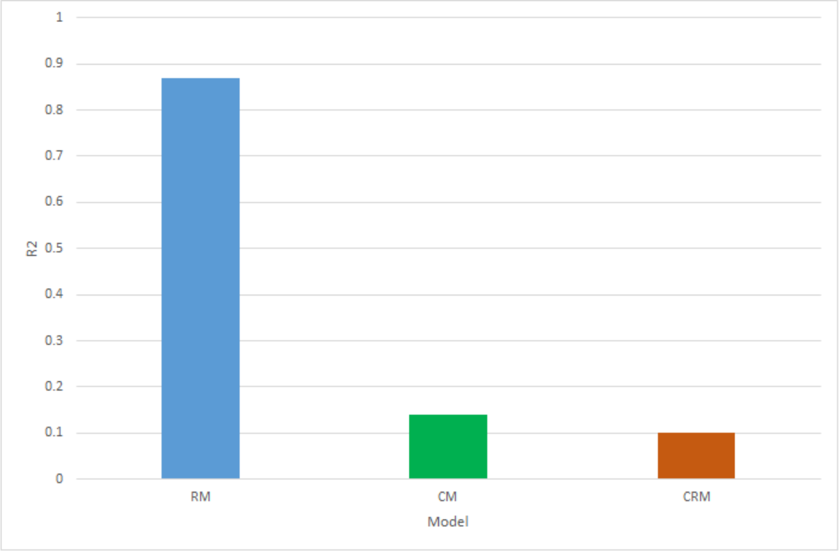


Figure 6.22: R^2 for the RM, CRM and CM for the total predicted field rate vs. the actual total field rate

6.8.4 The effect of the time constant in the field case

For the field case, the time constant loop did not seem to work the way that it did for the synthetic cases. The time constants did not converge to a reasonable fit to the data using the algorithm provided (the time constants appeared to be too large as the predicted flow rates were much smoother than the data). When smaller time constants were provided to the system, the time constants did not change from the initial value provided. Figure 6.23 shows the field data, results from the RM and three cases from the CRM process. The time constants for the best fit to the data were 1×10^{-20} month^{-1} which mimics the RM result. The figure also shows that when the time constants are set to 1 month^{-1} and 2 month^{-1} the predicted production variations are dampened and time shifted to the right (delayed). The fit to the data steadily gets worse. However, the prediction match is still reasonably good at 1 month^{-1}. Exactly what is causing the fast convergence in the time constant loop needs to be further investigated as there should be information in the time constant variations between wells that is not being considered.

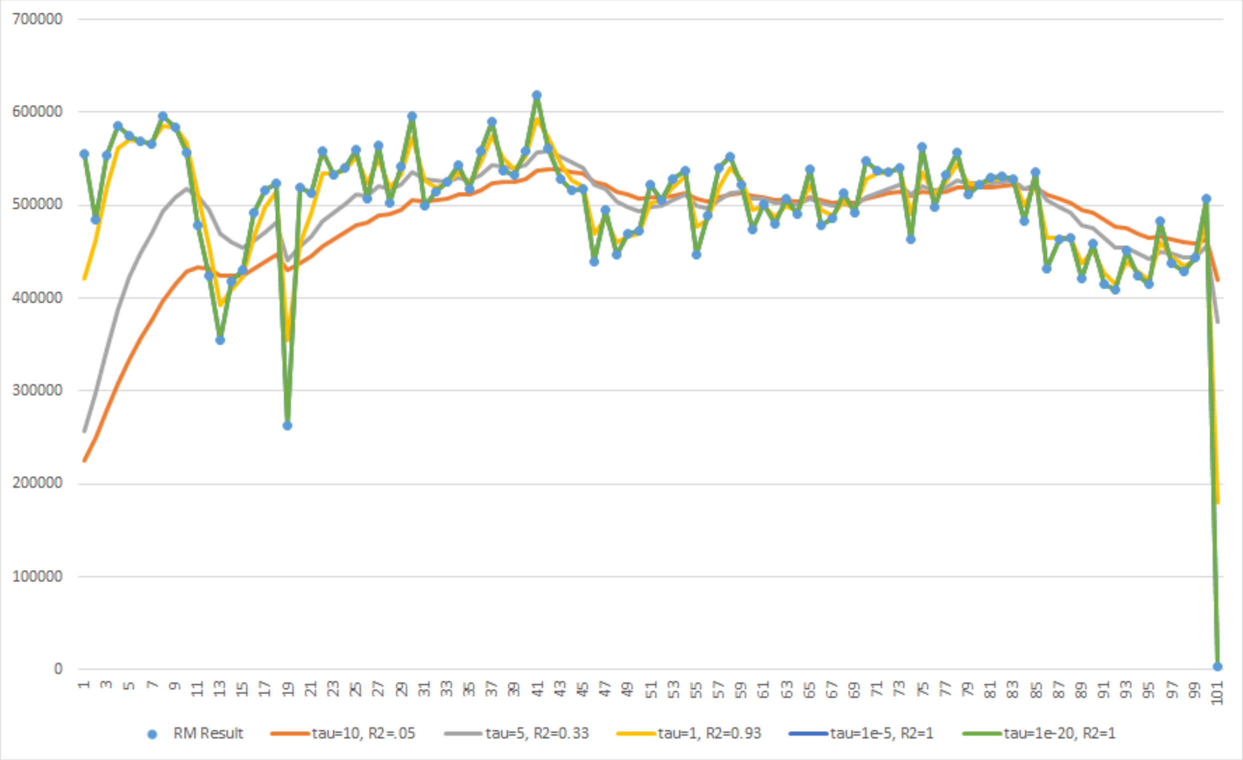


Figure 6.23: Effect of changing the time constant on production rates for the field case

CHAPTER 7
CONCLUSIONS-ONE INTEGRATED SOLUTION

7.1 Summary

In the previous chapters we discussed three simple models for reservoir characterization to investigate the possibility of getting all the models coded into one integrated package and come up with some recommendations and conclusions in this regard. Two coding environment were used to implement these coeds: Matlab and R-project where the codes have been written, tested in terms of coding feasibly, run time, data presentation and coding capabilities. We also tested some optimization packages needed for the CRM and the CM models in both Matlab and R. In general, RM was the easiest model to code in both R and Matlab, but it is recommended that Matlab be used to make use of its data representation process. For CRM and CM, R could not handle the optimization process although we tried a several optimization methods were tried. In addition to that, the LSU Statistics Department was asked for advice and they stated that “R is not optimized to handle these kind of high computational needs problems, while these packages are written by individuals and not optimized to run fast”. So Matlab was used to solve the optimization problem and several optimization methods were tested. They all yielded the same run time with a slightly different results. In general, Matlab is more suitable to solve the CRM and CM models of optimization.

7.2 Time constant and connectivity

The effect of the time constant (τ_{ij}) and connectivity (λ_{ij}) on synthetic cases was investigated, It has been noticed that the value of λ_{ij} controls the vertical shift of the curves. While the value of τ_{ij} controls how dampened or smooth the curve will be. Another common issue in field data is missing and corrupted data. It cannot be assumed that all the field data are 100% correct and complete with the many sources of errors like human error, device malfunction, measuring procedure flaws..etc.

Several statistical methods for handling missing data were reviewed and possible missing data patterns were identified. Matlab code was written to simulate these patterns in order to use them later on in the study. Basically there are two main options to deal with the missing data: Dropping and Imputation. This work found that it was not recommended to work with data with missing data rate of more than 6% as the error will propagate rapidly after this ratio. The most severe missing data pattern that could generate errors was found to be the "Moidified monotone pattern". Another preferable technique here is to impute and compensate the data by several methods. One is to infer a linear multivariate model and calculate the missing data by assuming that $(\hat{q}_j = \sum_{i=1}^{I} \lambda_{ij} i_i)$ for production imputation and $(\hat{i}_i = \sum_{i=1}^{J} \alpha_{ij} q_j)$ for injection. Before the imputation process starts there has to be a data check process that extracts some good data to construct the model. Several possible scenarios were studied and a decision tree was implemented and coded in order to take the right decision according to the given data and situation. The other issue was the corrupted data and its detection while it is another problem of the field-provided data. According to the statistical literature there are two main methods to detect the outliers in a set of data: Sample dependent and model dependent methods. The sample dependent method was used whereby descriptive statistics were used to point out the elements that lie out of the expected distribution of the data. For the model dependent methods a linear model is used to impute data and try to measure the deviation of the given elements against that model. This method requires a full understanding of the data, and as per the LSU Department of Statistics this method is less likely to be used, so it was not included in this work. Another way was found in the literature to detect the outliers which is a distance based method. This approach measures the distance of each point from the adjacent neighbors and by comparing this distance with a factor, a decision will be taken whether this point is an outlier or not. There are several types of these (distances) methods and the LOF: local Outlier Factor method was chosen. The two approaches (Standard deviation method (STD) and LOF) were coded and applied on a contaminated set of data. The results in the STD were more realistic and match the problematic points that were added, while the LOF method identified some of the outliers neighbors as additional outliers.

7.3 Field study

Access to field data which consisted of injection and production rates was obtained. The data was subjected to a cleaning procedure and the methods for outliers detection were applied. Some outliers were identified but because full understanding about the data collection conditions was not provided, no action could be taken to treat these possible outliers. The field was as an unbalanced injection process that has been under waterflooding at least 30 years ago. The usage of the RM yielded better results than CM and CRM in terms of data prediction.

7.4 One integrated package

In order to employ these models in one integrated package to get the most benefit from them, a complete code package was implemented. This package starts with the data which will undergo a thorough quality check procedure to detect any missing data or outliers and to find the best way to treat them. After this stage there is a routine that will processes that data using the three models and finishes with data representation. The models were coded under Matlab in a set of stand alone functions that can be called independently to perform the required tasks. A flow chart in Figure7.1 shows the entire procedure.

7.5 Future work recommendations

1. In all the three studied models we saw that every producer is the summation of the influence of all the other injectors. That could be a reasonable approach in synthetic systems and small field cases. In large field cases or for large synthetic cases this approach can be time consuming, computationally difficult and often is not correct for distant injector/producer pairs. A windowing technique needs to be developed for these large systems.

2. In every waterflood project, there will be some non-swept volumes due to preferential flow paths. Using the resulting values of the connectivities and their orientation in the reservoir could be a good starting point to investigate the non-swept volumes.
3. Some issues were faced during the CRM and CM optimizations (negative connectivity and time constant values, convergence issues, etc.). Different optimization algorithms should be investigated.
4. Studying the effect of applying these three models on deviated wells and identifying the factors that can be added to the model needs to be done.

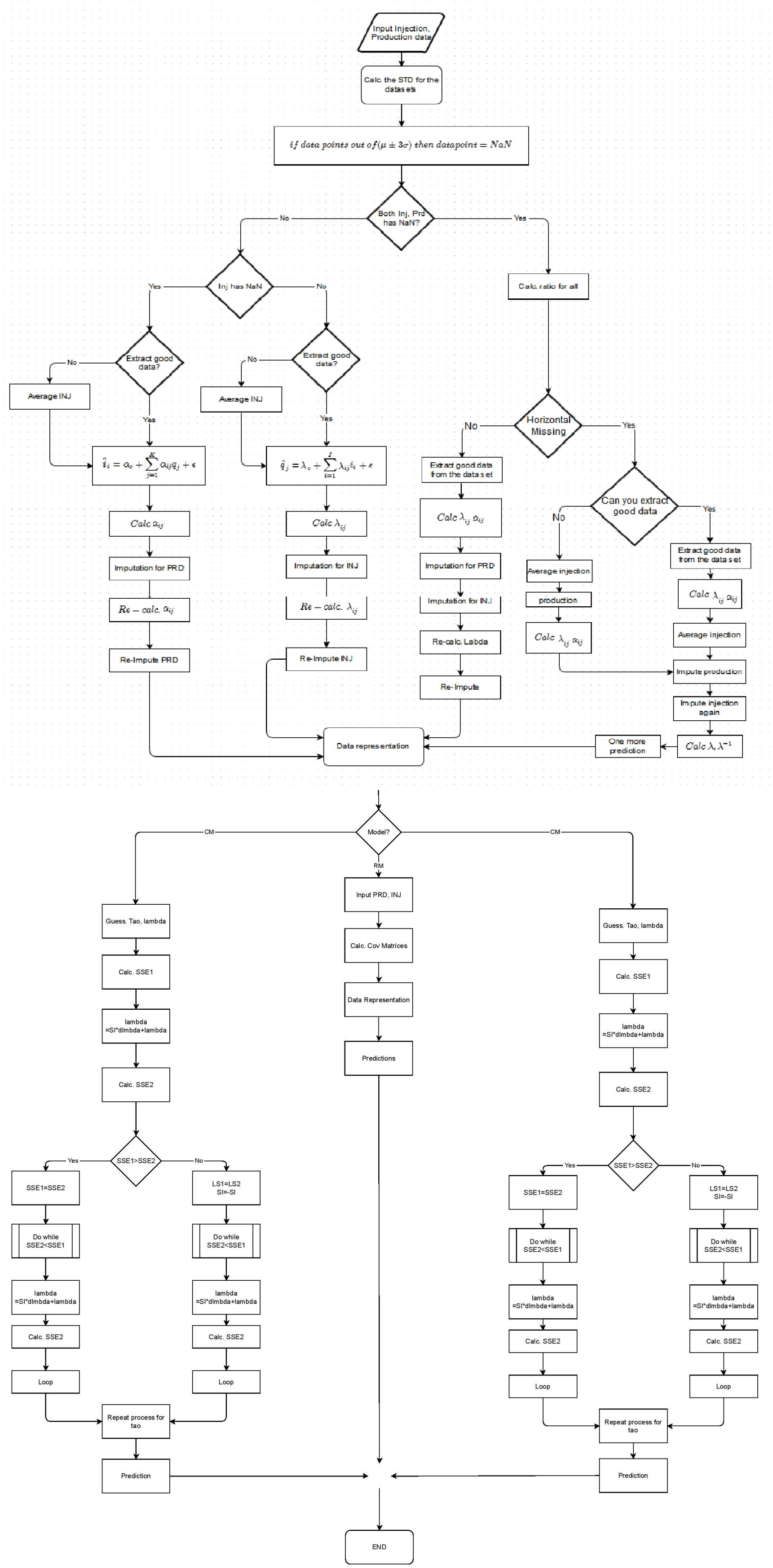


Figure 7.1: The suggested integrated model flow chart

Bibliography

Adler, Joseph. 2010. *R in a Nutshell.* " O'Reilly Media, Inc.".

Albertoni, Alejandro, Larry W Lake, et al. 2003. "Inferring interwell connectivity only from well-rate fluctuations in waterfloods." *SPE Reservoir Evaluation & Engineering* 6 (01): 6–16.

Allison, Paul D. 2001. *Missing data.* Sage.

———. 2012. "Statistical Horizons." *Haverford, PA, USA, Handling Missing Data by Maximum Likelihood, SAS Global Forum.*

Al-Yousef, Ali Abdallah. 2006. "Investigating statistical techniques to infer interwell connectivity from production and injection rate fluctuations." Ph.D. diss.

Braun, Stephan, Klaus Pantel, Peter Müller, Wolfgang Janni, Florian Hepp, Christina RM Kentenich, Stephan Gastroph, Artur Wischnik, Thomas Dimpfl, Günter Kindermann, et al. 2000. "Cytokeratin-positive cells in the bone marrow and survival of patients with stage I, II, or III breast cancer." *New England journal of medicine* 342 (8): 525–533.

Breunig, Markus M, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. 2000. "LOF: identifying density-based local outliers." *ACM Sigmod Record*, Volume 29. ACM, 93–104.

Cao, Fei. 2011. A new method of data quality control in production data using the capacitance-resistance model, MS thesis.

Chawla, Sanjay, and Pei Sun. 2006. "Outlier detection: Principles, techniques and applications." *Proceedings of the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Singapore.*

Coleman, Thomas, Mary Ann Branch, and Andrew Grace. 1999. "Optimization toolbox." *For Use with MATLAB. Users Guide for MATLAB 5, Version 2, Relaese II.*

CWillmott, Cort J, and Kenji Matsuura. 2005. "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance." *Climate Research* 30 (1): 79.

Draw.io. draw.io. https://www.draw.io/. (Visited on 07/02/2014).

DWeber, Daniel, Thomas F Edgar, Larry Wayne Lake, Leon S Lasdon, Sami Kawas, Morteza Sayarpour, et al. 2009. "Improvements in capacitance-resistive modeling and optimization of large scale reservoirs." *SPE Western Regional Meeting.* Society of Petroleum Engineers.

Feller, Willliam. 2008. *An introduction to probability theory and its applications.* Volume 2. John Wiley & Sons.

Franklin, Sarah, and Marie Brodeur. 1997. “A practical application of a robust multivariate outlier detection method.” *Proceedings of the Survey Research Methods Section, American Statistical Association*. 186–191.

Gherabati, Seyedmohammad Amin, Richard Gary Hughes, Hongchao Zhang, Christopher David White, et al. 2012. “A Large Scale Network Model To Obtain Interwell Formation Characteristics.” *SPE Western Regional Meeting*. Society of Petroleum Engineers.

Hawkins, Douglas M. 1980. *Identification of outliers*. Volume 11. Springer.

Hewahi, Nabil M, and Motaz K Saad. 2007. “Class outliers mining: Distance-based approach.” *International Journal of Intelligent Systems and Technologies*, vol. 2.

Kabacoff, Robert. 2011. *R in Action*. NY:Manning.

Kriegel, Hans-Peter, Peer Kröger, and Arthur Zimek. 2009. “Outlier detection techniques.” *Tutorial at the 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining*.

Lake, Larry W, Ximing Liang, Thomas F Edgar, Ali Al-Yousef, Morteza Sayarpour, Daniel Weber, et al. 2007. “Optimization of oil production based on a capacitance model of production and injection rates.” *Hydrocarbon economics and evaluation symposium*. Society of Petroleum Engineers.

Mathworks. Handling Missing Data and Outliers - MATLAB & Simulink. http://www.mathworks.com/help/ident/ug/handling-missing-data-and-outliers.html. (Visited on 06/17/2014).

———. Mahalanobis distance - MATLAB. http://www.mathworks.com/help/stats/mahal.html. (Visited on 06/21/2014).

———. Maximum Likelihood Estimation with Missing Data - MATLAB & Simulink. http://www.mathworks.com/help/finance/maximum-likelihood-estimation-with-missing-data.htmlf8-73149. (Visited on 06/16/2014).

MWei, M, AH Sung, J Wang, D Zhu, et al. 2005. “Detect Data Quality Problems In Large Data Sets By Mining Approximate Dependencies.” *Canadian International Petroleum Conference*. Petroleum Society of Canada.

Nobakht, M, L Mattar, et al. 2009. “Diagnostics of Data Quality for Analysis of Production Data.” *Canadian International Petroleum Conference*. Petroleum Society of Canada.

Ogunyomi, Gbemisola. 2009. Evaluation Of The Interwell Connectivity Of Little Creek Field, Mississippi From Production Data, MS thesis.

R Development Core Team. 2008. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. ISBN 3-900051-07-0.

Sayarpour, Morteza. 2008. “Development and Application of Capacitance-Resistive Models to Water/carbon Dioxide Floods.” Ph.D. diss., University of Texas.

Sayarpour, Morteza, C Shah Kabir, Larry W Lake, et al. 2009a. “Field applications of capacitance-resistance models in waterfloods.” *SPE Reservoir Evaluation & Engineering* 12 (06): 853–864.

Sayarpour, Morteza, Elizabeth Zuluaga, C Shah Kabir, and Larry W Lake. 2009b. “The use of capacitance–resistance models for rapid estimation of waterflood performance and optimization.” *Journal of Petroleum Science and Engineering* 69 (3): 227–238.

Schafer, Joseph L, and John W Graham. 2002. “Missing data: our view of the state of the art.” *Psychological methods* 7 (2): 147.

Tiab, Djebbar, Anh V Dinh, et al. 2008. “Inferring interwell connectivity from well bottomhole-pressure fluctuations in waterfloods.” *SPE Reservoir Evaluation & Engineering* 11 (05): 874–881.

University, Arizona State. lecture$_{34}.http://exoplanet.as.arizona.edu.(Visitedon08/04/2014)$.

MATLAB and R-PROJECT CODES

RM Model Codes

This code is under Matlab, the input to this code is the injection and production rates, the output will be a plotting matrix contains the simulated and calculated values and the connectivities matrix can be accessed by looking up the variable **mlrlambda**

```
PRD=prdmlr%csvread(’PRDB.csv’);
INJ=injmlr%csvread(’INJB.csv’);
nrowprd=size(PRD,1);
ncolprd=size(PRD,2);
ncolinj=size(INJ,2);
nrowinj=size(INJ,1);
injinj=cov(INJ);
%injprd cov
injprdcov=zeros(ncolinj, ncolprd);
for j=1:ncolprd
  for i=1:ncolinj
    tt=cov(INJ(:,i), PRD(:,j));
    injprdcov(i,j)=tt(1,2);
  end
end
mlrlambda=zeros(ncolinj, ncolprd);
mlrlambda=injprdcov’/(injinj);
mlrlambda=mlrlambda’;
%checking and plotting
predmatrix=ones(nrowprd, ncolprd);
for k=1:ncolprd;
  for n=1:nrowinj;
    temp=0;
    for j=1:ncolinj;
     temp=INJ(n,j)*mlrlambda(j,k)+temp;
    end
     predmatrix(n,k)=temp;
  end
end
for i = 1:ncolprd
  subplot(5,6,i);
  plot(PRD(:,i));hold on ;
plot (predmatrix(:,i),’Marker’,’.’,’LineStyle’,’none’,’Color’,[1 0 0])
  rs= corrcoef(PRD(:,i),predmatrix(:,i));
  title([’PRD’,num2str(i),’ R=’, num2str(sprintf(’%0.4f’,rs(1,2)))]);
  %legend(’Calculated’,’Simulated’)
end
```

CM Model codes

This code will calculate the CM model coefficients using the **fmincon** function in Matlab, the input are the injection and production rates and the output will be a plotting matrix contains the simulated and calculated values and the connectivities matrix can be accessed by looking up the variable **masterlambda**

```
%This is the main CRM model
PRD=csvread(’PRD.csv’);
PR=csvread(’PR.csv’);
INJ=csvread(’INJ.csv’);
nrowprd=size(PRD,1);
ncolprd=size(PRD,2);
ncolinj=size(INJ,2);
nrowinj=size(INJ,1);
nrowpr=size(PR,1);
ncolpr=size(PR,2);
masterlambda=ones(2*ncolinj+2*ncolpr,ncolpr);
for i=1:ncolprd
%Calling the Matrices Subroutine
lambdas=matrcies11(PRD(:,i),INJ);
masterlambda(1:ncolinj,i)=lambdas(2:ncolinj+1,1);
%masterlambda(ncolinj*2+ncolprd+1:ncolinj*2+2*ncolprd,i)=lambdas(ncolinj+2:ncolinj
    +1+ncolprd,1);
end
tao=.01*(ones(ncolinj,ncolprd));
nopressmat=zeros(2*ncolinj+1,ncolprd);
%modified optimization
for i=1:ncolprd
    options = optimset;
    options = optimset(options,’Display’, ’off’);
    options = optimset(options,’Algorithm’, ’interior-point’);
    options = optimset(options,’PlotFcns’, { @optimplotfval });
    x0=tao(:,i);
    lambda=masterlambda(2:ncolinj+1,i);
    %tao=tao(:,i);
    [x,fval,exitflag,output] = fmincon(@(tao)fn11(tao,lambda,INJ,PRD(:,i)),x0
        ,[],[],[],[],lb,ub,[],options);
%   [x,fval,exitflag,output] = fminunc(@(lambda)fn1(lambda,INJ,PRD(:,i),PR),x0,
    options);
    tao(:,i)=x
    for j = 1 : ncolprd
        x0=x
%         [x,fval,exitflag,output] = fminunc(@(lambda)fn1(lambda,INJ,PRD(:,i),PR),x0,
    options);
          [x,fval,exitflag,output] = fmincon(@(tao)fn11(tao,lambda,INJ,PRD(:,j)),x0
              ,[],[],[],[],lb,ub,[],options);

          tao(:,i)=x;
    end
    nopressmat(ncolinj+2:end,i)=x;
    nopressmat(1:ncolinj,i)=lambda;
end

%Calcluating qhat after getting the parameters
predmatrix=ones(nrowprd, ncolprd);
for i = 1 : ncolprd;
    predmatrix(:,i)= checking(masterlambda(:,i), INJ, PR);
end

%plotting
for i = 1:ncolprd
    subplot(2,2,i);
    plot(predmatrix(:,i));hold on ;
    plot (PRD(:,i),’Marker’,’.’,’LineStyle’,’none’,’Color’,[1 0 0])
   rs= corrcoef(PRD(:,i),predmatrix(:,i));
    title([’PRD’,num2str(i),’ R=’, num2str(sprintf(’%0.2f’,rs(1,2)))]);
    legend(’Sim’,’Act’)
end
```

This function calculates the value of $\hat{q}$ for any given set of injection rates, lambdas and pressures using the CM model.

```
function fin = fn1(lambda, INJ,PR)
nrow = size(INJ,1);
ncol = size(INJ,2);
nrowpr=size(PR,1);
ncolpr=size(PR,2);
idash = zeros(nrow, ncol);
%
% Constructing the i.dash matrix
for i = 1:ncol
    for j = 1:nrow
      temp=0;
       for k = 1:j
            temp= (1/lambda(ncol+i,1))*
exp((k-j)/lambda(ncol+i,1))*INJ(k,i)+ temp;

       end
      idash(j,i) =temp;
    end
end
%
% Preparing the idash matrix X lambda matrix
lambdai = zeros(nrow, ncol);
for i = 1:ncol
   lambdai(:,i)=lambda(i+1,1)*idash(:,i);
end
```

```
qhat = sum(lambdai,2);
%
% Prepating the BHP matrix
bhp1term=zeros(nrowpr, ncolpr);
for k=1:ncolpr
    for i = 1: nrowpr
        bhp1term(i,k)=PR(1,k)*
exp((1-i)/lambda(((2*ncol)+k),1));
    end
end
%
% preparing the 2nd term (Pwfkj)
bhp2term=PR;
%
%preparing the 3rd term (P'wfkj)
bhp3term=zeros(nrowpr, ncolpr);
for k=1:ncolpr
    for l=1:nrowpr
        temp=0;
        for m=1:l
            temp=(1/lambda(((2*ncol)+k),1))
*exp((m-l)/lambda(((2*ncol)+k),1))*PR(m,k)+temp;
        end
        bhp3term(l,k)=temp;
    end
end
%
%wrapping up
bhpmaster=bhp3term+bhp2term+bhp1term;
bhpnue=zeros(nrowpr, ncolpr);
for i=1:nrowpr
    for j=1:ncolpr
        bhpnue(i,j)= (bhpmaster(i,j)*
lambda(((2*ncol)+ncolpr+j),1))+bhpnue(i,j);
    end
end
fin=qhat+sum(bhpnue,2);
%fin=bhpmaster

return
```

This function creates the CM model sub-matrices and combine them into the master matrix. The inputs are production, injection and pressures. The output will be the **lambda** matrix that has IxJ values for connectives readings.

```
%This is the matrices creation subroutine
function lambdas=matrices(PRD, INJ, PR)
nrowprd=size(PRD,1);
```

```
ncolprd=size(PRD,2);
ncolinj=size(INJ,2);
nrowinj=size(INJ,1);
nrowpr=size(PR,1);
ncolpr=size(PR,2);
taop=ones(30,1).*.05; %this is the tao for the production Tk
taoipp=.2; %this is the tao for the injection Ti
ppmatrix=zeros(nrowprd,ncolprd);
ppvar=zeros(1,ncolprd);
cppi=zeros(ncolinj, ncolprd);
injinj=zeros(ncolinj, ncolinj);
ppq=zeros(1,ncolprd);
ciqj=zeros(ncolinj, ncolprd);
lhs=zeros(ncolinj+ncolpr+2,1);
lambda=zeros(2+ncolinj+ncolprd,1);
master=zeros(2+ncolinj+ncolpr,2+ncolinj+ncolpr);
iavg=zeros(1,ncolinj);
idash=zeros(nrowinj, ncolinj);
%preparing the BHP matrix
%Preparing the 1st term (Pwf(n0)e)
bhp1term=zeros(nrowprd, ncolprd);
for k=1:ncolpr
   for i=1:nrowpr
      bhp1term(i,k)=PR(1,k)*exp((1-i))/taop(k,1);
   end
end
%preparing the 2nd term (Pwfkj)
bhp2term=PR;
bhp3term=zeros(nrowpr, ncolpr);
%Preparing the 3rd term(P’wfkj)
for k=1:ncolpr
   for l=1:nrowpr
      temp=0;
      for m=1:l
         temp=(1/taop(k,1)*exp((m-l)/taop(k,1))*PR(m,k)+temp);
      end
    bhp3term(l,k)=temp;
   end
end
%wrapping up bhp...
bhpmaster=zeros(nrowpr, ncolpr);
bhpmaster=bhp1term+bhp2term+bhp3term;
% preparin the idash values
taoi=csvread(’taoi.csv’);
for i=1:ncolinj
   for j=1:nrowinj
      temp=0;
      for k=1:j
         temp=(1/taoi(i,1))*exp((k-j)/taoi(i,1))*INJ(k,i)+temp;
```

```
        end
        idash(j,i)=temp;
    end
end

%filling pp matrix for all producers
for j=1:ncolprd
    for i=1:nrowprd
        ppmatrix(i,1)=PRD(1,1)*exp((-i+1)/taoipp);
    end
end
%filling the cov matrix for pp
ppvar=var(ppmatrix);
%filling the cov of cppi and its trans.
for j=1:ncolprd
    for i=1:ncolinj
        tt=cov(ppmatrix(:,1),idash(:,i));
        cppi(i,j)=tt(1,2);
    end
end
cppit=transpose(cppi);
%filling the inj-inj cov
injinj=cov(INJ)

%filling sigmapp-qq matrix
for j=1:ncolprd
    tt=cov(ppmatrix(:,j),PRD(:,j));
    ppq(1,j)=tt(1,2);
end
%filling the ci-qj matrix
for j=1:ncolprd
    for i=1:ncolinj
        tt=cov(idash(:,i), PRD(:,j));
        ciqj(i,j)=tt(1,2);
    end
end
%preparing the BHP-BHP cov matrix
bhpbhp=zeros(ncolpr,ncolpr);
for i=1:ncolpr
    for j=1:ncolpr
        tt=cov(bhpmaster(:,i),bhpmaster(:,j));
       bhpbhp(i,j)=tt(1,2);
    end
end
% preparing the bhp.avg matrix and trans
bhpavg=transpose(mean(bhpmaster));
bhpavgt=transpose(bhpavg);
% preparing the cpp bhp and its trans
cppbhp=zeros(ncolpr, 1);
```

```
for i=1:ncolpr
   tt=cov(ppmatrix,bhpmaster(:,i));
   cppbhp(i,1)=tt(1,2);
end
cppbhpt=transpose(cppbhp);
%preparing the C - BHP matrix
cbhpqj=zeros(ncolpr,1);
for i=1:ncolpr
   tt=cov(bhpmaster(:,i), PRD(:,1));
   cbhpqj(i,1)=tt(1,2);
end
%preparing the ci-bhp matrix and its trans
cibhp=zeros(ncolinj, ncolpr);
for i=1:ncolinj
   for j=1:ncolpr
      tt=cov(idash(:,i), bhpmaster(:,j));
      cibhp(i,j)=tt(1,2);
   end
end
cibhpt=transpose(cibhp);
% preparing the avg matrix for the i dash
iavg=mean(idash);
%preparing the master matrix
ncolmaster=size(master,2);
nrowmaster=size(master,1);
master(1,1)=ppvar(1,1);
master(1,2:(ncolinj+1))=cppit;
master(1,2+ncolinj:ncolmaster-1)=cppbhpt;
master(1,ncolmaster)=mean(ppmatrix);
master(2:(ncolinj+1), 1)=cppi(:,1);
master((nrowmaster),1)=mean(ppmatrix);
master(2:(ncolinj+1),2:(ncolinj+1))=injinj;
master(2:(ncolinj+1),ncolmaster)=transpose(iavg);
master((nrowmaster),ncolmaster)=0;
master(2:(ncolinj+1), ncolmaster)=iavg;
master(ncolinj+2:nrowmaster-1, 1)=cppbhp;
master(ncolinj+2:nrowmaster-1,2:ncolinj+1)=cibhpt;
master((ncolinj+2):(nrowmaster-1),(2+ncolinj):(ncolmaster-1))=bhpbhp;
master(2:(ncolinj+1), (2+ncolinj):(ncolmaster-1))=cibhp;
master((ncolinj+2):(nrowmaster-1), ncolmaster)=bhpavg;
master(nrowmaster,2:ncolinj+1)=iavg;
master(nrowmaster,2+ncolinj:ncolmaster-1)=bhpavgt;
% filling the LHS matrix
nrowlhs=size(lhs,1);
lhs(1,1)=ppq;
lhs(2:(ncolinj+1),1)=ciqj(:,1);
lhs(ncolinj+2:nrowlhs-1,1)=cbhpqj;
lhs(nrowlhs,1)=mean(PRD);
%%%%%%lambda
```

```
lambdas=inv(master)*lhs;
return
```

This code is under R-Project and it solves the CM model. In order to access the values lookup the file **lambdaReplacement.csv** in the working directory and the code also will generate some comparison plots for the value of **lambdas**

```
#Bismilla-Reading the files and preparing the matrices------------CM Model
ptm <- proc.time()
PRD=read.csv(file="PRDCRM.csv")
INJ=read.csv(file="INJCRM.csv")
ms=4
PRD=do.call(cbind, PRD[ms])
INJ=do.call(cbind, INJ)
tao.p=.2
#Preparing the matrices
pp.matrix=matrix(ncol=ncol(PRD), nrow=(nrow(PRD))) #pp matrix
pp.var=matrix(ncol=ncol(PRD), 1) #pp variance matrix
cpp.I=matrix(ncol=ncol(PRD), nrow=ncol(INJ)) #cpp.I matrix
inj.inj=matrix(ncol=ncol(INJ), nrow=ncol(INJ)) #inj-inj matrix
pp.q=matrix(ncol=ncol(PRD)) #pp.q matrix
ci.qj=matrix(ncol=ncol(PRD), nrow=ncol(INJ)) #ci.qj matrix
lhs=matrix(nrow=ncol(INJ)+2,1) #lhs matrix
lambda=matrix(ncol=1, nrow=(1+ncol(INJ))) #lambda matrix
master=matrix(ncol=ncol(INJ)+2, nrow=(ncol(INJ)+2)) #master matrix
i.avg=matrix(1,ncol(INJ))
i.dash=matrix(ncol=ncol(INJ), nrow=(nrow(INJ)))
#############Preparing the i.dash values####################################
# preparing the i’ values
tao=do.call(cbind, read.csv(file="tao.csv",header=FALSE))
for (i in 1:ncol(INJ)){
  for (j in 1:nrow (INJ)){
    temp=0
    for (k in 1:j){
      temp=(1/tao[i,1])*exp((k-j)/tao[i,1])*INJ[k,i]+temp
    }
    i.dash[j,i]=temp
  }
}
########################filling PP MATRIX FOR ALL PRODUCERS###################
for (j in 1 : ncol(PRD))
  for (i in 1: nrow(PRD)) {
    pp.matrix[i,1]=PRD[1,1]*exp((-i+1)/tao.p)
  }
#######################filling THE COV. MATRIX FOR PP#########################
pp.var= var(pp.matrix)

#######################filling THE COV OF Cpp-I and its transpose############
```

```
for (j in 1:ncol(cpp.I))
  for (i in 1: nrow(cpp.I)){
    cpp.I[i,j]=cov(pp.matrix[,1],i.dash[,i] )
  }
cpp.IT=t(cpp.I)
#######################filling THE INJ-INJ COV MATRIX##########################
for (j in 1: ncol(inj.inj))
  for (i in 1: ncol(inj.inj)){
    inj.inj[i,j]=cov(i.dash[,j], i.dash[,i])
  }
#######################filling SIGMA.PP-QQ MATRIX#############################
for ( j in 1: ncol (pp.q)){
  pp.q[1,j]=cov(pp.matrix[,j], PRD[,j])
}
######################filling CI-QJ MATRIX####################################
for ( j in 1: ncol(ci.qj))
  for (i in 1: nrow(ci.qj)){
    ci.qj[i,j]=cov(i.dash[,i], PRD[,j])
  }
################Preparing the avg. matrix for i.dash###########################
i.avg=colMeans(i.dash)
###################ASSEMBLING THE MASTER MATRIX################################
master[1,1]=pp.var[1,1]
master[1,2:(ncol(master)-1)]=cpp.IT[1,]
master[1,ncol(master)]=mean(pp.matrix)
master[2:(nrow(master)-1), 1]=cpp.I[,1]
master[nrow(master),1]=mean(pp.matrix)
master[2:(nrow(master)-1),2:(ncol(master)-1)]=inj.inj
master[nrow(master),2:(ncol
                     (master)-1)]=t(i.avg)
master[nrow(master),ncol(master)]=0
master[2:(nrow(master)-1), ncol(master)]=i.avg
##################################filling THE LHS Matrix#####################
lhs[1,1]=pp.q
lhs[2:(nrow(lhs)-1),1]=ci.qj[,1]
lhs[nrow(lhs),1]=mean(PRD)
##############lambda###################Resulsts##############################
lambda=solve(master)%*% lhs
#for (i in 1 : nrow(lambda)){
if(lambda[i,1]<0) {lambda[i,1]=0}
#}
write.csv(lambda,file="lambdaReplacement.csv")
#############################plotting########################################
coff=read.csv(file="BETAS.CSV")

png(file=paste("CRM Lambdas",".png"),width=900,height=900)
plot (lambda[2:nrow(lambda),1], type="l",xlab="Injector No.",
      ylab="Lambda", col="green",axes=FALSE ,
      ylim=c(min(lambda,coff[,2] ),c(max(lambda, coff[,5]))))
```

```
axis(2)
axis(1,at=1:(ncol(INJ)), cex.axis=.7)
par(new=TRUE)
plot (coff[,ms+1], axes=FALSE, type="l", col="red",xlab=FALSE,ylab=FALSE)
#axis(4)
box()
title(main = (paste("Lambdas", lambda[1,1])))
dev.off()
##############checking out############################
# Preparing the lambda X i’ matrix
lambda.i=matrix(ncol=ncol(INJ),nrow=nrow(INJ))
for (i in 1: ncol(INJ)){
  lambda.i[,i]=lambda[i+1, 1]*i.dash[,i]
}
#Wrapping up to calculate Qhat
#preparing the on col qhat
q.hat=matrix(nrow=nrow(INJ),1 )
for (i in 1:nrow(INJ)){
  q.hat[i,1]=sum(lambda.i[i,1:ncol(INJ)])+lambda[1,1]*pp.matrix[i,1]

}
diff=matrix(nrow(INJ), 1)
diff=q.hat-PRD
png(file=paste("Qobs vs Qhat",".png"),width=2000,height=900)
plot(PRD, type="l", col="red", axes=TRUE,ylim=c(min(diff,PRD,q.hat),max(PRD,q.hat)))
legend (’topright’, c("Q Calc." ,"Q Obs","Difference"),lty=c(1,1,2),
      col=c("blue", "red", "brown"))
par(new=T)
plot(q.hat[,1], type="l", col="blue", axes=F)
par(new=T)
plot(diff, type="b", col="brown", axes=F,
    ylim=c(min(diff,PRD,q.hat),max(PRD,q.hat)))
dev.off()
Difference=colSums(q.hat)-colSums(PRD)
proc.time() - ptm
```

CRM model codes

This code will solve the CRM model using an ad-hoc algorithm. The inputs are the production and injection rates. The wells locations and names are also needed in order to plot the sectors. You may want to zoom in in order to spot the sectors. tic

```
clear all

PRDI=readtable(’PRD_Org.xlsx’);

INJI=readtable(’INJ_Org.xlsx’);

prd=table2array(PRDI(305:end,2:end-3));

inj=table2array(INJI(305:end,2:end-3));

XYII=readtable(’XYI.csv’);

XYPI=readtable(’XYP.csv’);

xyi=table2array(XYII(:,2:end));

xyp=table2array(XYPI(:,2:end));

% inj=csvread(’INJB.csv’);
%
% prd=csvread(’PRDB.csv’);
%
% xyi=csvread(’xyib.csv’);%rand(size(inj,2),2);
%
% xyp=csvread(’xypb.csv’);%rand(size(prd,2),2);;

%Calc. the initial values of f (which is the initial values of the

%connectvity

distance=zeros(size(xyi,1), size(xyp,1));

for k=1:size(xyp,1)

    for i =1:size(xyi,1)

        distance(i,k)=((abs(xyp(k,1)-xyi(i,1)).^2+(abs(xyp(k,2)-xyi(i,2)).^2))).^.5;

    end

end

%creating the matirx of the connectivity
```

```
fmat=zeros*distance;

for k=1:size(xyp,1)

    for i =1:size(xyi,1)

        fmat(i,k)=(1/distance(i,k))./(sum(1./distance(:,k)))+.06;

    end

end


%---------Creating the matrix of taos------------------------

tao=.06*ones(size(inj,2),size(prd,2));

qest=zeros*prd;

%-------------------------------------------------------------
tic
%---------Calculating Q from f & tao above-----------

for j= 1:size(prd,2)%to no. of prod.

    for i = 1:size(inj,1)%to no. of timesteps

        for kk=1:i%internal loop from 1-->i

            for m=1:size(inj,2)%no. of injetors

                qest(i,j)=qest(i,j)+(1-exp(-1/tao(m,j)))*...
                    (fmat(m,j)*inj(kk,m))*(exp((kk-i)/tao(m,j)));

            end

        end

    end

end

%------Calc. the new sum of sequared errors between Qest and Qact.---

% This new SSE is after one increment of fmat to check the convergence
```

```
%---Procedure of calculatin the optimum lambda for each injector
dfmat=.1;
si=1;
for j =1:size(prd,2)
for i = 1:size(fmat,1) % taking every lambada
    si=1;
    %---------checking the convergence----------------------
       fmatnew=fmat;% creating a new matrix
       fmatnew(i,j)=fmatnew(i,j)+si*dfmat; %modifying the new matrix
       lso=ls2(fmat(:,j),tao(:,j),size(prd,2),size(inj,2),prd(:,j),inj);
       lsn=ls2(fmatnew(:,j),tao(:,j),size(prd,2),size(inj,2),prd(:,j),inj);
       if (lso<lsn)% if true--> we are not on the track, flip the sign
           si=-2*si;
       end
       fmat=fmatnew;
       m=1     ;
       tol=mean(prd(:,j))*size(inj,1);
           for tt = 1:5
           abs(lso-lsn);
           lso=lsn;
           lso;
           lsn;
           fmat(i,j)=fmat(i,j)+si*dfmat; %starin the iteration
                      lsn=ls2(fmat(:,j),tao(:,j),size(prd,2),size(inj,2),prd(:,j),
                          inj);
       m=m+1;
```

```
        m;
        end
        lso=0;
        lsn=0;

end

end

% Calc. qest new from fmat and tao

 lso=0;

 lsn=0;

%------Starting the procedure for optimizing tao---------------------%
dtao=.01;

for j =1:size(prd,2)

    for i = 1:size(fmat,1) % taking every lambada

        si=1;
    %---------checking the convergence----------------------

        taonew=tao;% creating a new matrix

        taonew(i,j)=taonew(i,j)+si*dtao; %modifying the new matrix

        lso=(ls2(fmat(:,j),tao(:,j),size(prd,2),size(inj,2),prd(:,j),inj));

        lsn=(ls2(fmat(:,j),taonew(:,j),size(prd,2),size(inj,2),prd(:,j),inj));
        lsn-lso
%        if (lso<lsn)% if true--> we are not on the track, flip the sign
%
%            si=-2*si;
%
%        end

        taonew=tao;


        m=1;
        %for tt = 1:15
        taotest=zeros(100,2);
        while (lsn<lso) && (m<100)
            abs(lso-lsn);
            lso=lsn;
            lso;
            lsn;
            taotest(m,1)=tao(i,j)
            tao(i,j)=tao(i,j)+si*dtao; %starin the iteration
            lsn=(ls2(fmat(:,j),tao(:,j),size(prd,2),size(inj,2),prd(:,j),inj));
            taotest(m,2)=lsn;
            old=lsn-lso
            m=m+1;
            end
        m
        %lso=0;
        %lsn=0;

end

end

close all
for i =1:size(prd,2)
subplot(4,4,i)
qest=qescal(fmat(:,i),taonew(:,i),size(prd,2),size(inj,2),prd(:,i),inj);
plot(qest(:,1))
hold on
plot(prd(:,i),’r*’)
hold off
xlabel(’time (month)’)
ylabel(’rate (bbl/d)’)
legend(’Calculated’, ’Simulated’,’Location’,’south’)
title([’PRD’,num2str(i)])
lsn=ls2(fmat(:,i),taonew(:,i),size(prd,2),size(inj,2),prd(:,i),inj);
end

toc
%Sectors routine
%---------------------------------------------------------
xyi=csvread(’xyi.csv’);
xyp=csvread(’xyp.csv’);
prdnames=csvread(’prdnames.csv’);
injnames=csvread(’injnames.csv’);
angles=zeros(size(xyi,1),size(xyp,1));
arrlength=fmat
axislimit=max(xyi(:));
if max(xyp(:))>axislimit
    axislimit=max(xyp(:))
end
for i = 1 : size(angles,1)%filling per row for each injector
    for j =1:size(angles,2)
        angles(i,j)=atand((xyp(j,2)-xyi(i,2))/(xyp(j,1)-xyi(i,1)))
        if (xyp(j,2)-xyi(i,2))<0 && (xyp(j,1)-xyi(i,1))<0
            angles(i,j)=angles(i,j)+180
        end

        if (xyp(j,2)-xyi(i,2))==0 && (xyp(j,1)-xyi(i,1))<0
            angles(i,j)=angles(i,j)+180
        end

        if (xyp(j,2)-xyi(i,2))>0 && (xyp(j,1)-xyi(i,1))<0
            angles(i,j)=angles(i,j)+180
        end
    end
end
figure
for i = 1:size(angles,1)
    for j = 1:size(angles,2)
        plot_sect(angles(i,j),(angles(i,j)+10),xyi(i,1),xyi(i,2),arrlength(i,j),
            axislimit)
        hold on
    end
end

for i =1:size(prdnames,1)
    text(xyp(i,1),xyp(i,2),['*P', num2str(prdnames(i,1))])
end
for i =1:size(injnames,1)
    text(xyi(i,1),xyi(i,2),['I', num2str(injnames(i,1))])
end
```

This code is to solve the CRM method using the **fmincon** function in Matla. The output is the matrix of lambdas and the input will be the rates, and well locations. The lambdas matrix can be accessed by looking up the variable **fmat** clear all

```
PRDI=readtable('PRD_Org.xlsx');

INJI=readtable('INJ_Org.xlsx');

prd=table2array(PRDI(305:end,2:end-3));

inj=table2array(INJI(305:end,2:end-3));

XYII=readtable('XYI.csv');

XYPI=readtable('XYP.csv');

xyi=table2array(XYII(:,2:end));

xyp=table2array(XYPI(:,2:end));

%Calc. the initial values of f (which is the initial values of the
%connectvity

distance=zeros(size(xyi,1), size(xyp,1));

for k=1:size(xyp,1)

    for i =1:size(xyi,1)

        distance(i,k)=((abs(xyp(k,1)-xyi(i,1)).^2+(abs(xyp(k,2)-xyi(i,2)).^2))).^.5;

    end

end

%creating the matirx of the connectivity

fmat=zeros*distance;

for k=1:size(xyp,1)

    for i =1:size(xyi,1)

        fmat(i,k)=(1/distance(i,k))./(sum(1./distance(:,k)))+.06;

    end

end
%---------Creating the matrix of taos------------------------

tao=.06*ones(size(inj,2),size(prd,2));

qest=zeros*prd;

%------Optimization section lambda-------------------------
% the process will be for each producer, so the filling for lambda matrix
% will be vertical
lb=0*(fmat(:,1));
ub=1+lb;

for i=1:size(prd,2)

    for j=1:5

        inifmat=fmat(:,i);

        fmat(:,i)=fmincon(@(fmat)crm(abs(inifmat),inj,prd(:,i),tao(:,i)),inifmat
            ,[],[],[],[],lb,ub,[],options);
        inifmat=fmat(:,i);

    end

end

%------Optimization section tao----------------------------

lb=ub*.001;

for i=1:size(prd,2)

    for j=1:5

        initao=tao(:,i);

        tao(:,i)=fmincon(@(tao)crm(abs(fmat(:,i)),inj,prd(:,i),initao),initao
            ,[],[],[],[],lb,ub,[],options);

        initao=tao(:,i);

    end

end

%----------- checking and plotting-------------------------
close all
for i =1:size(prd,2)
subplot(4,4,i)
qest=qescal(fmat(:,i),tao(:,i),size(prd,2),size(inj,2),prd(:,i),inj);
plot(qest(:,1))
hold on
plot(prd(:,i),’r*’)
hold off
xlabel(’time (month)’)
ylabel(’rate (bbl/d)’)
legend(’Calculated’, ’Simulated’,’Location’,’south’)
title([’PRD’,num2str(i)])
lsn=ls2(fmat(:,i),tao(:,i),size(prd,2),size(inj,2),prd(:,i),inj);
end

%---------------sectors routine----------------------------
angles=zeros(size(xyi,1),size(xyp,1));
arrlength=fmat
axislimit=max(xyi(:));
if max(xyp(:))>axislimit
    axislimit=max(xyp(:))
end
for i = 1 : size(angles,1)%filling per row for each injector
```

```
    for j =1:size(angles,2)
        angles(i,j)=atand((xyp(j,2)-xyi(i,2))/(xyp(j,1)-xyi(i,1)))
        if (xyp(j,2)-xyi(i,2))<0 && (xyp(j,1)-xyi(i,1))<0
            angles(i,j)=angles(i,j)+180
        end

        if (xyp(j,2)-xyi(i,2))==0 && (xyp(j,1)-xyi(i,1))<0
            angles(i,j)=angles(i,j)+180
        end

        if (xyp(j,2)-xyi(i,2))>0 && (xyp(j,1)-xyi(i,1))<0
            angles(i,j)=angles(i,j)+180
        end
    end
end
figure
for i = 1:size(angles,1)
    for j = 1:size(angles,2)
        plot_sect(angles(i,j),(angles(i,j)+10),xyi(i,1),xyi(i,2),arrlength(i,j),
            axislimit)
        hold on
    end
end
```

VITA

Atheer Al Attar is a Mechanical engineer from Basra, Iraq born in 1982. He completed his B.Sc in Mechanical Engineering from the University of Basra in 2004. His last years of experience was with BP/South Oil Company joint venture in Rumaila field. He was awarded Fulbright scholarship with other 35 Iraqis in 2012. He is to be awarded Masters degree in petroleum engineering from Louisiana State in Fall 2014.