Comparison of Multi-Model History Matching Methods

by

Prashanth Neelakantan
Student Id: 4507843

in partial fulfillment of the requirements for the degree of

Master of Science
in Applied Earth Sciences

at the Delft University of Technology,
to be defended publicly on Monday October 30, 2017 at 14:00 PM.

Advisor	Dr. Ir. Olwijn Leeuwenburgh	TNO/TU Delft
Supervisor	Prof. Dr. Ir. Jan Dirk Jansen	TU Delft
Thesis Commitee	Dr. Ir. Femke Vossepoel	TU Delft
	Prof. Dr. Martin Verlaan	TU Delft

TU Delft

Acknowledgements

I wish to express my gratitude to Prof. Jan Dirk Jansen and Dr. Olwijn Leeuwenburgh for giving me the opportunity to work on this project. The progress meetings were opportunities to put my work in context and I thank them for their suggestions during these. I would also like to thank Olwijn for his patience as he explained concepts and ideas I had trouble with. His emphasis on the need to take one step at a time was of immense help during difficult times in the project.

These two years at TU Delft have been a roller coaster of emotions and experiences. I am thankful to my friends, here and at home for hearing me out during difficult times and showing me that there is always a silver lining. Finally, I am grateful to my family for convincing me of the need to pursue higher education- this has been a truly marvellous journey. Words are insufficient to express the support I have received from them, thank you.

Prashanth Neelakantan
Delft, October 2017

Summary

The oil industry is a high risk high reward venture. The capital and operating expenses runs into tens of millions of dollars with oil production being the primary source of revenue. During the initial development phase of the field, few wells are completed. Data collection during this period verifies assumptions made during the modelling phase and forms the basis for future development. History matching can play a significant role in these plans since it can determine uncertainties in future production.

History matching algorithms must be able to make an accurate estimates of uncertainty in future production while being computationally light. Towards this end, a number of history matching methods have been developed with emphasis being on the ensemble Kalman filter (EnKF) in recent times. While the variants of the EnKF seek to improve different aspects of the method, few have been successful in addressing all of these concerns.

The Distributed Gauss Newton (DGN) was developed with the same goal- accurate uncertainty prediction at low computational cost. It is not a variant of the EnKF but uses a sensitivity matrix determined through linear regression which decreases the computational load compared to existing gradient based techniques. In their tests, the authors report superior performance of the DGN compared to a Gauss-Newton scheme. This thesis aims to provide a detailed understanding of the method and its dependencies. This is followed up with a comparison of the DGN with an EnKF variant known as the ES-MDA.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

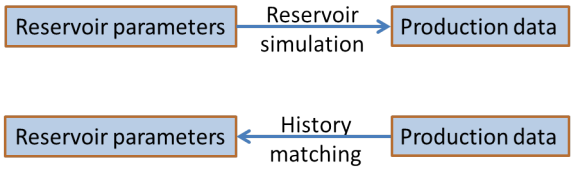

Figure 1.1: Relation between reservoir parameters and production parameters

With reservoir simulation, the reservoir parameters can be transformed into production parameters while the reverse is true with history matching. Hence simulation can be thought of as a forward problem while history matching is an inverse problem. In history matching, reservoir properties at each grid cell are variables with production parameters being the given data. Due to the complex relationship between the variables and the data, a direct relationship does not exist whereby the data can be used to determine the variables. Hence, history matching is done using a combination of reservoir simulation and variable updating schemes. There exist different configurations of reservoir properties that result in the same/similar production data making this an under-determined problem.

The reservoir parameters are properties that are populated in the grid cells and are subject to regional geology. This is usually an amalgamation of different lithologies which are heterogeneous, i.e., parameters vary in different directions. The properties of the subsurface are observed at discrete intervals (well locations) and interpolation techniques are used populate properties in the grid cells (for reservoir simulation). High costs limit extensive data collection while interpolation with limited data can give rise to errors in simulation results. Thus, there is an uncertainty in reservoir parameters used.

These parameters include permeability and porosity in the grid cells, oil-water-gas contacts, fault transmissibility, aquifer size and strength and so on. History matching can be used to tune these parameters to match measurements. The measurements are production parameters in the field such as bottom hole pressures and fractional flux rates. It will have some errors associated with it since no recording device is error-free. These are assumed to be Gaussian with

zero mean and some variance. Further, the measurements are assumed to be independent.

History matching is not an end. It is a means to appraise uncertainty in the field. Prior to producing the field, uncertainty quantification is performed with limited data leading to a production plan. Having produced from the field for a duration, more data is now available. This is incorporated into the uncertainty study through history matching techniques to obtain a better forecast of future production. The initial models used before forecasting form the prior data set while history matching generates models that sample a 'posterior' distribution based on minimization of an objective function (under Gaussian assumptions).

1.2 Literature review

The history matching problem can be phrased as determination of model parameters conditional to the recorded measurements [Oliver et al., 2008]. Bayes' theorem can then be used to relate the different conditional probabilities to determine the distribution of these parameters- called the posterior. Most of these parameters are approximated to be Gaussian resulting in the posterior having a form-

$$p(\mathbf{m}|\mathbf{d_{obs}}) = a * e^{-f} \tag{1.1}$$

where,

p is probability
$\mathbf{m}$ contains the model parameters
$\mathbf{d_{obs}}$ is the observed data
a is some constant
f is the objective function defined in Equation 2.1

f contains the sum of squares of differences between simulated data (of the models being history matched) and measurements. This is referred as data mismatch. While using more data decreases the effect of measurement errors, it also increases the value of the objective function causing a possible underestimation of uncertainty. This pitfall in bayesian history matching could be offset by modifying the covariance matrix associated with the data to ensure that the data mismatch does not inflate with increasing amounts of data used [Vink et al., 2015].

In Equation 1.1 models that minimize f sample the posterior distribution. One way of determining the model parameter space is to use a large number of models. If a sufficiently large set of models is used, then the entire posterior distribution could be identified. This is the premise behind the use of Monte Carlo methods. The Markov Chain Monte Carlo (MCMC) is a subset of this method and is used in history matching. It allows for complete sampling of the posterior distribution. However, it requires the use of an extremely large number of models and is not computationally feasible for practical problems. A variation, known as the RML may be used instead. Its advantage lies in the fact that it requires fewer models to determine a posterior that almost approximates the MCMC. However, the ensemble (set of models) size required is still large enough to make this method unattractive. Nevertheless, both these methods may be used for bench marking of other history matching techniques [Oliver and Chen, 2011]

Gradual deformation [Roggero et al., 1998] is a possible alternative to the Monte Carlo/RML approach owing to reduced computational requirements. The premise is to modify (deform) the prior model in steps so that the newly generated model continues to honor well data. With sufficient deformations, these models sample the true posterior distribution. A characteristic of

the algorithm is the updating of model parameters which occurs along random directions. After the first few iterations when the objective function has decreased, the probability of finding an update (that minimizes the objective) in a random direction decreases. This in turn makes the algorithm inefficient [Liu and Oliver, 2004].

Gradient based techniques could be an alternative since they can be used to determine an optimal search direction for updates. These methods usually require the computation of the sensitivity of observed data to the model parameters. Gauss-Newton steps can then be used to minimize an 'objective function' which contains the data mismatch term. However, the determination of gradients is often a tedious task.

Gao et al. developed the Distributed Gauss Newton (DGN) method where the gradient calculation is straightforward. In this method, the sensitivity of the measurements to the model parameters is computed for all the prior models used through linear regression. This is then used to calculate the gradients of the objective function using a Gauss-Newton scheme. The use of linear regression alleviates the cost of gradient computations. In [Gao et al., 2016a], they compare the DGN with a Gauss Newton scheme developed in prior research [Gao et al., 2016b] reporting the superior performance of the former. In [Gao et al., 2016c], using a toy model, the authors compare the posterior distribution generated by the DGN with that of the MCMC, making a tentative claim that the DGN is more efficient.

Among gradient free techniques, one that has become popular in recent times is based on the Ensemble Kalman Filter (EnKF). Introduced in the 1990's in meteorology and oceanography, it is a sequential data assimilation filter which can update multiple models simultaneously. It was introduced in the petroleum industry in 2001 and has since gained popularity especially in history matching. It is attractive since a single computation is sufficient to update a large number of models (forward runs of the models are still required). Various forms of the EnKF have been introduced since with the common aim of improving the posterior sampling of the EnKF vis-à-vis the MCMC. This includes forms which are iterative- have better performance when the model parameters are non-Gaussian, smoothers- the data assimilation step is modified resulting in a non-sequential filter or hybridized schemes- combination of the EnKF with the MCMC or RML. Of these, a variation of the smoother known as the ensemble smoother with multiple data assimilation (ES-MDA) has been very successful in estimating the posterior distribution [Emerick and Reynolds, 2013b].

Uncertainty quantification is the integral part of history matching. This may be characterized on the basis of differences in the model parameters, or the simulated measurements in a forecast period. Fenwick and Baycky describe this among other things through the use of metric space methods [Fenwick and Batycky, 2011]. This technique involves analysis by clustering the posterior models in a lower dimensional space.

1.3 Research objectives

The aim of this thesis is twofold- to understand the working of the DGN and later compare it with ensemble based methods on counts of computational efficiency and sampling of the posterior. Towards this end, tests are carried on the implemented code to understand its working. This implementation is based on [Gao et al., 2016a]. For comparison, the ES-MDA is chosen. This is because of its success in characterizing the posterior distribution while being

computationally feasible [Emerick and Reynolds, 2013b].

1.4 Report structure

Chapter 2 provides a theoretical background of the concepts used in the thesis. This is followed by an analysis of the workflow used in the DGN in Chapter 3. Chapter 4 deals with experiments done using the implemented code. The code is first verified through the use of toy problems following which synthetic reservoir model testing is carried out to identify dependencies. This is followed up by comparisons with the ES-MDA on counts of computational time and posterior variability (through metric space methods). The chapter ends with a comparison of the methods on a benchmarking study by [Emerick and Reynolds, 2013b]. Conclusions based on these tests are made in Chapter 5 which is followed by possible avenues for research based on the results obtained thus far.

Chapter 2

Theoretical concepts

2.1 History matching problem

The reservoir parameters estimated through history matching are referred to as model parameters. The reservoir models used during the process are evaluated based on data mismatch. This value is large prior to the process and is often quite small at the end. Being an under determined problem, there exists more than one reservoir model that has a sufficiently low mismatch. While it is desirable to find all possible reservoir models with low mismatch values, this is not computationally feasible. To limit the size of the solution space regularization terms can be used.

$$f(x) = \frac{1}{2}\{(\mathbf{x} - \mathbf{x_{prior}})^T {\mathbf{C_M}}^{-1}(\mathbf{x} - \mathbf{x_{prior}}]) + [\mathbf{g(x)} - \mathbf{d_{obs}}]^T {\mathbf{C_D}}^{-1}[\mathbf{g(x)} - \mathbf{d_{obs}}]^T\} \qquad (2.1)$$

where

$\mathbf{x}$ is a model parameter
$\mathbf{x_{prior}}$ is the mean of the prior of the parameter
$\mathbf{C_M}$ is the covariance of the parameters
$\mathbf{d_{obs}}$ is the measured data
$\mathbf{C_D}$ is the covariance of the measured data
$\mathbf{g(x)}$ is the simulated data

The objective function is defined in Equation 2.1. If N_p parameters and N_d observations are considered, then $\mathbf{x}$ and $\mathbf{x_{prior}}$ are vectors of size $1 \times N_p$, $\mathbf{C_M}$ is a matrix of $N_p \times N_p$, $\mathbf{g(x)}$ and $\mathbf{d_{obs}}$ are $1 \times N_d$ and $\mathbf{C_D}$ is $N_d \times N_d$ respectively. The first part of the equation is called the model error while the latter part is the data error. Data error is the mismatch that history matching aims at diminishing while model error is the regularization used. Here, it limits the solution space by controlling the 2-norm distance of an update from the prior.

History matching can thus be considered a minimization problem as the aim is to reduce mismatch between observed and simulated data. Since multiple solutions are present, this is a multi-modal problem. There are two methods of determining these solutions. One method is to repeatedly perform minimization by starting from different locations in the model space. In this way, the local minima closest to each starting point can be found. The other method is to simultaneously minimise from different locations. While the former is easy to implement, it

is computationally intensive. As a result, latter methods are preferred for the history matching.

2.2 Matching strategies

Section 2.1 introduces the history matching problem. Strategies used to reduce the data mismatch may be derivative based or derivative-free. The derivatives referred to are of the objective function with respect to the model parameters.

Derivative based methods, as the name suggests, use the derivative of the objective function to determine updates to the reservoir model. The objective function has the same form as defined in Equation 2.1. The derivatives of the objective function are shown in Equation 2.2-2.3. These are derived in Appendix A. The reader is referred to [Oliver et al., 2008] for further reading.

In these expressions, $\mathbf{J} = \nabla \mathbf{g}$ is the sensitivity of the simulated data to model parameters. It is known as the local sensitivity or the Jacobian. $\nabla \mathbf{f}$ and $\mathbf{H}$ are the first and second order derivatives of the objective function. $\mathbf{H}$ is usually referred to as the Hessian.

$$\nabla \mathbf{f} = (\mathbf{x} - \mathbf{x_{prior}})^T {\mathbf{C_M}}^{-1} + \mathbf{J}^T {\mathbf{C_D}}^{-1}[\mathbf{g}(\mathbf{x}) - \mathbf{d_{obs}}] \tag{2.2}$$

$$\mathbf{H} = {\mathbf{C_M}}^{-1} + \mathbf{J'}^T {\mathbf{C_D}}^{-1}[\mathbf{g}(\mathbf{x}) - \mathbf{d_{obs}}] + \mathbf{J}^T {\mathbf{C_D}}^{-1}\mathbf{J}, \qquad \mathbf{J'} = \nabla \mathbf{J} \tag{2.3}$$

Derivative based methods use these to determine update in model parameters. One such commonly used technique is the Newton-Rhapson scheme where the update is shown in Equation 2.4. A potential problem in using this method in history matching is the severe computation cost necessary to accurately capture the derivative of the Jacobian. This led to use of the Gauss-Newton method where the second term in the Hessian definition was neglected resulting in Equation 2.5 [Oliver et al., 2008]. This neglected term tends to zero as the mismatch between simulated and observed data approaches zero making this an acceptable approximation.

$$\mathbf{x_{update}} = \mathbf{x_{old}} - \mathbf{H}^{-1} \nabla \mathbf{f} \tag{2.4}$$

$$\mathbf{H} = {\mathbf{C_M}}^{-1} + \mathbf{J}^T {\mathbf{C_D}}^{-1}\mathbf{J} \tag{2.5}$$

Derivative free methods do not require the computation of gradients which decreases computational requirements. The ensemble Kalman filter (EnKF) falls in this category and has been in focus the past decade. The filter consists of an update step and a forecast step and data assimilation takes place one at a time. These are detailed in Section 2.3. In the forecast step, the initial guess of the model parameters is used to propagate the model forward in time so as to determine the simulated data. In the update step, the Kalman gain is calculated using this simulated data which is then used to update the model parameters. After the update step, the new forecast parameter is the analysed parameter of the previous update step while the same is true for the covariance of the model parameters. In this manner, the forecast and update step follow each other till defined convergence criteria are satisfied.

The DGN is a derivative based method because it uses derivatives to update the model parameters. It is different from traditional gradient based methods in that the gradient used at the start is an approximation and hence is not computationally taxing. The DGN is contrasted with a variation of the EnKF which is a derivative free method.

2.3 Ensemble smoother with multiple data assimilation (ES-MDA)

The ensemble smoother differs from the EnKF in the way data is assimilated. The underlying theory is the same for both methods. Since the EnKF is quite popular, we start with this method. It consists of a forecast and an update step shown below. It is a sequential data assimilation technique, i.e., data can be incorporated as it becomes available. This is attractive since the reservoir history prior to the data (to be assimilated) does not have to be simulated again. The reader is referred to [Aanonsen et al., 2009] and [Oliver and Chen, 2011] which provide insights into the evolution of the EnKF. [Evensen, 2003] gives a detailed discussion on implementation of the filter.

Forecast step

$$\mathbf{G}(x_f, t + \delta t) = \mathbf{G}(x_f, t) \tag{2.6}$$

where,

$\mathbf{x}_f$ is the forecast model parameter
δt is the length of a time step

Update Step

$$\begin{aligned} \mathbf{K} &= \mathbf{P_f}\mathbf{G}^T(\mathbf{G}\mathbf{P_f}\mathbf{G}^T + \mathbf{R})^{-1} \\ \mathbf{x_a} &= \mathbf{x_f} + \mathbf{K}[\mathbf{d} - \mathbf{g}(\mathbf{x})] \\ \mathbf{P_a} &= (\mathbf{I} - \mathbf{K}\mathbf{G})\mathbf{P_f} \end{aligned} \tag{2.7}$$

where,

$\mathbf{K}$ is the Kalman gain
$\mathbf{P_f}$ is the covariance of the model parameters used in the forecast step
$\mathbf{G}$ is the operator that relates the model parameters to data
$\mathbf{R}$ is the covariance of the perturbations applied to the observed data
$\mathbf{x}_a$ is the updated model parameter
$\mathbf{d}$ is the measured data
$\mathbf{P_a}$ is the covariance of the updated model parameters
$\mathbf{I}$ is an identity matrix of appropriate size

$\mathbf{G}$ is defined to be an operator relating the simulated data to the model parameters. But reservoir simulation being a complex non-linear process makes it impossible to determine such an operator. Hence the definition of the covariances are simplified such that it is not necessary to define $\mathbf{G}$. In the update equation, the operator, G is linear which is not true in this case. Thus, the operator is linearised and the model parameters augmented. In the remainder of the section, $\mathbf{G}$ refers to the linearised operator.

Consider that the ensemble of the augmented model parameters $\mathbf{A} = m[\mathbf{x_1}\mathbf{x_2}\ldots\mathbf{x_n}]$ where x_i refer to a vector of parameters that belong to one model and m is the number of ensemble equivalents added to augment the model parameters. In such a case, the perturbation in the ensemble can be defined as

$$\mathbf{A}' = \mathbf{A} - \overline{\mathbf{A}} \tag{2.8}$$

The covariance of the forecast model parameters is shown below. Here, N refers to the ensemble size.

$$\mathbf{P_f} = \frac{\mathbf{A}'\mathbf{A}'^T}{N-1} \tag{2.9}$$

From the definition of the EnKF, perturbations need to be added to the measurements. $\mathbf{R}$ represents the covariance of this ensemble of perturbations. If $\boldsymbol{\gamma}$ the ensemble of perturbations, then

$$\mathbf{R} = \frac{\boldsymbol{\gamma}\boldsymbol{\gamma}^T}{N-1} \tag{2.10}$$

When applied to the Kalman gain expression, we get

$$\begin{aligned}\mathbf{K} &= \frac{\mathbf{A'}\mathbf{A'}^T}{N-1}\mathbf{G}^T(\mathbf{G}\frac{\mathbf{A'}\mathbf{A'}^T}{N-1}\mathbf{G}^T + \frac{\boldsymbol{\gamma}\boldsymbol{\gamma}^T}{N-1})^{-1} \\ &= \mathbf{A'}\mathbf{A'}^T\mathbf{G}^T(\mathbf{G}\mathbf{A'}\mathbf{A'}^T\mathbf{G}^T + \boldsymbol{\gamma}\boldsymbol{\gamma}^T)^{-1} \\ &= \mathbf{A'}(\mathbf{G}\mathbf{A'})^T[(\mathbf{G}\mathbf{A'})(\mathbf{G}\mathbf{A'})^T + \boldsymbol{\gamma}\boldsymbol{\gamma}^T]^{-1}\end{aligned} \tag{2.11}$$

Similarly, the covariance at the update step becomes

$$\begin{aligned}\mathbf{P}_a &= (\mathbf{I} - \mathbf{K}\mathbf{G})\mathbf{P}_f \\ &= \frac{1}{N-1}(\mathbf{I} - \mathbf{K}\mathbf{G})\mathbf{A'}\mathbf{A'}^T \\ &= \frac{1}{N-1}[\mathbf{A'}\mathbf{A'}^T - \mathbf{K}(\mathbf{G}\mathbf{A'})\mathbf{A'}^T]\end{aligned} \tag{2.12}$$

Since G is an operator relating the model parameters and the observations and $\mathbf{A'}$ is the ensemble of model parameter anomalies, the product $G\mathbf{A'}$ is the ensemble of simulated data anomalies. Equations 2.11-2.12 are then used in Equation 2.7 to ease implementation.

The EnKF is a sequential data assimilation filter implying the measurements are used one at a time. This means there is a forecast and an update step that accompanies each assimilated measurement. If, on the other hand, all the measurements are assimilated together, that is there is just a single update step present, then the filter is known as a smoother. The ensemble smoother, thus, is not a sequential data assimilation filter. However, it is similar to the EnKF in that it follows the same methodology. The advantage of the smoother is that it does not require simulation restarts that may be necessary in geo-modelling workflows.

Ensemble smoother with multiple data assimilation, as the name suggests, comprises repeated assimilation of the same set of data. Since the main difference between the schemes lies in data assimilation, the equations shown above can be used to implement the smoother. Another difference between the schemes is in the covariance matrix for the observations. Equation 2.11 indicates that the data covariance has an inverse relation to the Kalman gain and consequently the model parameters. Increasing this will reduce the magnitude of updates. [Rommelse, 2009] noted that having a large magnitude for the update can result in poor predictions of parameters of state. He proposed increasing the covariance matrix to mitigate this while also deriving the requisite number of times the data assimilation needs to be repeated to find the 'correct' states. He worked with a single measurement whereas typical history matching involves more. [Emerick and Reynolds, 2012] extended this approach, testing it with synthetic as well as actual reservoir models. They show that when multiple data assimilations are performed with the same data set, the coefficients used to inflate the covariance of measurements satisfy Equation 2.13.

$$\sum_{i}^{n_\alpha} \frac{1}{\alpha_i} = 1 \tag{2.13}$$

where,

α is the inflation factor
i is a counter for number of assimilations
n_α is the number of assimilations

In the ES-MDA, the number of data assimilations need to be decided prior to history matching and the associated inflation factors (α) calculated. The process is iterative and proceeds till completion of all data assimilation steps. Since all the data is assimilated simultaneously, there is no need to calculate the analysed covariance matrix. The forecast is then repeated for the updated model parameters to find the new updates. In this manner, the forecasting and updating steps iterate till all data assimilation steps are complete. This is shown in the algorithm below.

Algorithm 1 ES-MDA

1: **for** $i = 1, 2, ...n_a$ **do**
2: Forecast for the entire history match period-
3: $\mathbf{G}(x_f, t + \delta t) = \mathbf{G}(x_f, t)$
4: Update model parameters-
5: $K = \mathbf{A}'(\mathbf{GA}')^T[(\mathbf{GA}')(\mathbf{GA}')^T + \alpha_i \boldsymbol{\gamma}\boldsymbol{\gamma}^T]^{-1}$
6: $\mathbf{x_a} = \mathbf{x_f} + \mathbf{K}[\mathbf{d} - \mathbf{g}(\mathbf{x})]$
7: **end for**

2.4 The Distributed Gauss Newton method

This method of history matching was recently proposed by Gao et al., [Gao et al., 2016a]. The authors claim that the DGN is superior to traditional history matching methods from a computational perspective while also providing a better sampling of the posterior distribution. The method used for comparison by the authors was a Gauss Newton algorithm with direct pattern search. The computational gain stems from the reduced number of reservoir simulations that need to be performed. Using a toy problem, they showed that the DGN required 3.6 times fewer reservoir simulations to determine the posterior. When applied to a synthetic facies based reservoir model, they report the DGN required 157 times fewer simulations. This motivates this study of the DGN.

Similar to the ensemble approach, the DGN also makes use of an ensemble of reservoir models. This ensemble incorporates prior data available on the reservoir. In order to generate solutions [1] that span the solution space [2], it is preferable to distribute the initial ensemble over the entire range of its parameters (i.e., permeability). The workflow used in the DGN is shown in Figure 2.1.

The DGN is a gradient based technique since it uses the derivatives of the objective function to determine an update to the model parameters. It is a Gauss Newton scheme in that the approximated Hessian is used. An advantage of the DGN is in the ease with which the sensitivity matrix and consequently gradient and Hessian are determined (uses linear regression). The objective function is constructed at discrete points using the model error and the data error.

[1] reservoir configuration that fits the production data
[2] set of all configurations that fit the production data

However, its functional form is unknown. Furthermore, its shape depends on the type of parameters being optimized [Oliver et al., 2008]. Hence, the objective function is approximated with different local models at different locations.

An ensemble member is a collection of values for each model parameter. In this sense, the ensemble can be considered a collection of points in N_p dimensional space. To construct the local model, derivatives are calculated at each of these points. Here, a quadratic model is used. Provided the model approximates the objective function well, minimizing the former is equivalent to minimization of the latter. [Gao et al., 2016a] choose a trust region strategy for this purpose while three different methods are tested in this thesis.

The goodness of fit between the objective and its approximation is not known till the objective function is evaluated for the update. If deemed sufficient, the update is accepted and a quadratic model is constructed at the accepted update to generate the next update. Otherwise, a new model is constructed at the original point to generate a different update. In this manner, local models are constructed at all ensemble members till they satisfy some convergence criteria.

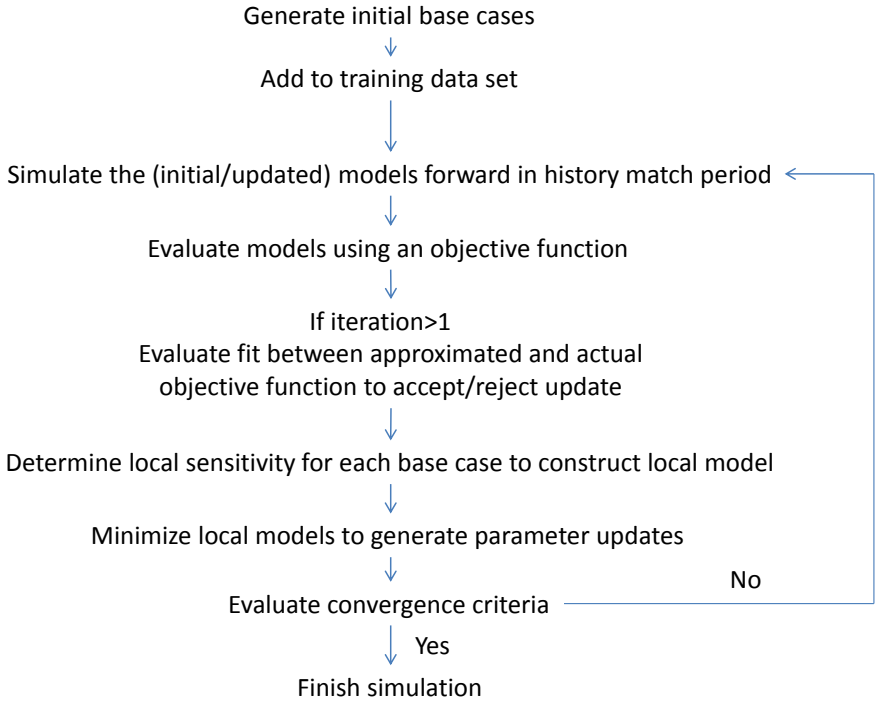

Figure 2.1: Workflow used by the DGN

2.5 Parameter based changes

The DGN uses linear regression to calculate the local sensitivity matrix. For a linear system to have a unique solution, the number of variables being solved for needs to equal the number

of available equations. In the synthetic reservoir model used in this thesis, there are 441 parameters which correspond to permeability in the 441 grid cells. In practice, reservoir models have hundreds of thousands of grid cells; if grid cell permeability is used for history matching, the number of equations required would be just as large meaning the ensemble will be similarly sized.

Given a set of parameters that need to be tuned, the ensemble size may be restricted by solving an under-determined linear system or by transforming the parameters to a different model space. The former may be achieved using the pseudo inverse while the latter is implemented through the use of principal component analysis.

2.5.1 Strategy 1: Pseudo-inverse

The More-Penrose definition allows for the determination of inverse for matrices in situations where one does not exist, i.e., the matrix has zero determinant or has a non-square shape. In the context of local sensitivity determination, having an insufficient ensemble size, i.e., more model parameters than ensemble members means a non-square coefficient matrix. Thus, in Equation 3.1, the actual inverse is replaced by the pseudo inverse. The matrix A^+ is said to be the Moore-Penrose pseudo inverse of A if it satisfies the following properties.

$$\begin{aligned} \mathbf{A} &= \mathbf{A}\mathbf{A}^+A \\ \mathbf{A}^+ &= \mathbf{A}^+\mathbf{A}\mathbf{A}^+ \\ \mathbf{A}\mathbf{A}^+ &= (\mathbf{A}\mathbf{A}^+)^T \\ \mathbf{A}^+\mathbf{A} &= (\mathbf{A}^+\mathbf{A})^T \end{aligned} \tag{2.14}$$

The generalized solution to a linear system of the form $\mathbf{AJ} = \mathbf{b}$ where $\mathbf{A}$ is the coefficient matrix, J is the vector of variables and b is the vector of constants is shown in 2.15. When the coefficient matrix is not square, the inverse $\mathbf{A}^+$ cannot be found through regular means and hence the pseudo-inverse is employed. Properties of the pseudo inverse can then be used to show that the resulting solution is the least norm solution to the system [Planitz, 1979]. In the DGN, sensitivity estimates are made for all ensemble members. If this becomes close to zero, no further improvement is possible in the vicinity of the member and it is said to have converged to a local minimum. In this sense, using the least norm solution can aid in convergence.

$$\mathbf{J} = \mathbf{A}^+b + (I - \mathbf{A}^+\mathbf{A})\mathbf{w} \tag{2.15}$$

where,

$\mathbf{w}$ is an arbitrary vector

In Matlab, the pseudo-inverse implementation is built into the function *pinv* where it is found through singular value decomposition. The coefficient matrix is decomposed to give singular value matrix and the left and right singular vectors (shown below). In the singular value matrix, the non-zero elements are replaced by their reciprocal with the result being the inverse of the matrix transposed.

$$\begin{aligned} [\mathbf{U}, \mathbf{S}, \mathbf{V}] &= svd(\mathbf{A}) \\ \mathbf{S}^+ &= 1./\mathbf{S} \quad \text{(for non zero elements)} \\ \mathbf{A}^{+T} &= \mathbf{U}\mathbf{S}^+\mathbf{V}^T \end{aligned}$$

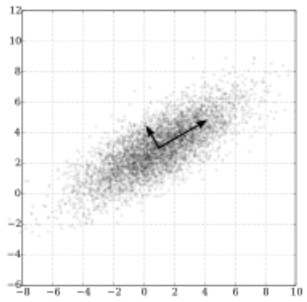

Figure 2.2: A random scatter of points in 2-D space. Image taken from wikipedia.org

where

. represents element wise operation

2.5.2 Strategy 2: Principal component analysis

Consider a large scatter of points as shown in Figure 2.2. Principal component analysis (PCA) can be used to correlate these points. Being in two dimensions, PCA calculates two orthogonal directions which characterizes these points. If these points were to exist in an N_p dimensional space, PCA would yield N_p orthogonal directions. Further, it also determines the relative impact of these directions in characterizing the data set in the form of singular values.

Now, consider the history matching problem. There exist multiple combinations of the model parameters that can minimize the objective function in Equation 2.1. Based on information gathered from well log, analogue studies, etc., multiple prior reservoir models are built. The parameters in these models are then tuned to provide history matched results. The start of this section mentioned the need to have an ensemble that is as large as the number of model parameters being tuned, in the DGN. Using PCA, some directions that characterize the data can be ignored based on their relative importance. Using the remaining directions, the existing data set can be transformed into a lower dimensional model space. With the decrease in model parameters, the ensemble size needed also decreases.

In this thesis, PCA was performed through singular value decomposition. Equation 2.16 shows the matrices formed through this process. If K contains the model parameters and each instance of a parameter is stored along a column then u is square matrix such that each column represents a direction of variability in the data set. These columns are also known as basis vectors. The number of model parameters desired in the lower dimensional space equals the number of basis vectors used for transforming the data set. Equation 2.17 shows the transformation of the data set.

$$[\mathbf{U}, \mathbf{A}, \mathbf{V}] = svd(\mathbf{K_{anomaly}}); \qquad \mathbf{K_{anomaly}} = \mathbf{K} - \overline{\mathbf{K}} \tag{2.16}$$

where,

K is the permeability data set

the overline refers to the mean value

$$\begin{aligned} \mathbf{K_{new}} &= \overline{\mathbf{K}} + \mathbf{K_{anomaly,new}} \\ \mathbf{K_{anomaly,new}} &= \sum_{i=1}^{n} \alpha_i \mathbf{U}_i \end{aligned} \tag{2.17}$$

where,

α is a matrix of scalars containing PCA coefficients
$\mathbf{U}$ contains the basis vectors

The α used is given by Equation 2.18. In Equation 2.17, if a portion of the basis vectors are used, the permeability will be transformed to a lower dimensional space. However, if all the basis vectors are used, it ensures $\mathbf{K_{anomaly,new}} = \mathbf{K_{anomaly}}$, i.e., the new permeability field will be the same as the original one. Equation 2.19 proves this. The substitution in the second line comes from Equation 2.18. The last line is a result of taking the dot product of orthogonal vectors. The result is a diagonal matrix containing the value of the required multipliers.

$$\alpha = \sum_{i=1}^{n} \mathbf{K}_i \cdot \mathbf{U}_i \tag{2.18}$$

$$\begin{aligned} \mathbf{K_{new}} &= \overline{\mathbf{K}} + \sum_{i=1}^{n} \alpha_i \mathbf{U_i} \\ &= \overline{\mathbf{K}} + \sum_{i=1}^{n} (\sum_{j=1}^{n} \mathbf{K_{anomaly}} \cdot \mathbf{U_j}) \cdot \mathbf{U_i} \\ &= \overline{\mathbf{K}} + \sum_{i=1}^{n} \sum_{j=1}^{n} \mathbf{K_{anomaly}} \cdot (\mathbf{U_j} \cdot \mathbf{Y_i}) \\ &= \begin{cases} \overline{\mathbf{K}} + \mathbf{K_{anomaly}} & \text{if } \mathbf{U_i} = \mathbf{U_j} \\ 0 & \text{if } \mathbf{U_i} \neq \mathbf{U_j} \end{cases} \\ &= \begin{cases} \mathbf{K_{original}} & \text{if } \mathbf{U_i} = \mathbf{U_j} \\ 0 & \text{if } \mathbf{U_i} \neq \mathbf{U_j} \end{cases} \end{aligned} \tag{2.19}$$

The final point to be discussed is the required number of parameters in the reduced model space such that it can model the original space accurately. This is done by using the retained variance parameter which in turn depends on the energy content of the reduced space. This is calculated as

$$retained\ variance = \frac{\sum_i^p \lambda_i}{\sum_i^{N_p} \lambda_i} \tag{2.20}$$

where,

λ_i is the i^{th} eigenvalue
N_p is the number of parameters
p is the dimension of the reduced model space

They are related to the singular values (a_i) as

$$\lambda_i = \frac{a_i^2}{N_p - 1} \tag{2.21}$$

where,

a_i is the i^{th} singular value

2.6 Optimization strategy

Optimization here refers to minimization of the quadratic approximation of the objective function. In their study, Gao et al [Gao et al., 2016b] compare line search and trust region methods to hybridize their pattern search algorithm (HJ-DPS) and conclude that trust region schemes are more robust when used with inaccurate gradients. Since the DGN uses linear regression to estimate its gradients, they are not exact and hence trust region methods were preferred.

The objective function $f(x)$ is approximated with a quadratic model. If a perturbation h were to be introduced at x, then Taylor series can be used to determine this approximation. To ensure it is quadratic, higher order terms (>2) are neglected.

$$f(x+h) = f(x) + h\frac{\partial f}{\partial x} + h^2\frac{\partial^2 f}{\partial x^2} + O(h^3) \tag{2.22}$$

There are two classes of methods that can be used to solve this problem. Line search algorithms are one type where the update to the model parameters are assumed to lie along a certain direction and a choice of step length results in finding the most appropriate update. These may be further classified based on whether the search direction is the steepest descent or the Newton direction. The other class of methods are trust region based where the update to the model is determined based on the goodness of fit between the approximated model and the true objective function.

2.6.1 Line search- Steepest descent

As the name suggests, the updates are chosen to lie along the steepest descent direction.

$$\mathbf{x_{n+1}} = \mathbf{x_n} + \alpha\mathbf{p} \tag{2.23}$$

where,

$\mathbf{x}$ is a vector of model parameters

$\mathbf{p}$ is a search direction

α is the step length along the search direction

If $\mathbf{f}$ is the objective and $\nabla\mathbf{f}$ its gradient, then the steepest descent direction is $-\nabla\mathbf{f}$. While it can determine local minima in smooth problems, implementation in non-smooth problems can result in convergence difficulties. This can be offset by including Hessian information in the search direction leading to Newton schemes. However, these were not explored in this thesis.

2.6.2 Trust region schemes

Two variations of the trust region scheme were used- the dog leg and conjugate gradient method. The former was used in a majority of tests done on the DGN. These methods differ in the way

the update is generated. The premise of the trust region is that an update be selected such that approximated model can be minimized within the trust region. The update is then used to determine the actual value of the objective function and these are compared. If the actual decrease in objective is at least a fraction of the predicted decrease, the update is accepted. Depending on the value of the ratio, the size of the trust region is increased. However, if this ratio is less than a user defined threshold, the update is rejected and the trust region shrunk. This process of generating updates, comparing ratios and updating trust region sizes (if necessary) is repeated till some convergence criteria are fulfilled. For a more detailed treatment of the methods, the reader is advised to refer [Wright and Nocedal, 1999],[Conn et al., 2000].

Dog-leg algorithm

This method uses 2 different search directions to find the optimum. Initially the steepest descent direction is followed after which it is modified to be closer to the Newton direction. These form what is known as the dog leg path. This can be represented as-

$$\mathbf{s} = \begin{cases} \tau \mathbf{s^U} & 0 \leq \tau \leq 1 \\ \mathbf{s^U} + (\tau - 1)(\mathbf{s^H} - \mathbf{s^U}) & 1 \leq \tau \leq 2 \end{cases} \tag{2.24}$$

where,

$\mathbf{s}$ is the search direction

τ is a some constant to be determined

$\mathbf{s^U}, \mathbf{s^H}$ are along the steepest descent and Newton directions respectively and are defined as

$$\mathbf{s^U} = -\frac{\mathbf{g}^T\mathbf{g}}{\mathbf{g}^T\mathbf{H}\mathbf{g}}\mathbf{g} \tag{2.25}$$

$$\mathbf{s^H} = -\mathbf{H}^{-1}\mathbf{g} \tag{2.26}$$

where,

$\mathbf{g}$ is the gradient of the objective function

$\mathbf{H}$ is the Hessian matrix

It is thus clear that the search direction lies somewhere within the space spanned by $\mathbf{s^U}, \mathbf{s^H}$. This form can be used only if the Hessian matrix is positive semidefinite. Since this cannot be guaranteed in the DGN, the search space is modified to $[\mathbf{g}, (\mathbf{H} + \alpha\mathbf{I})^{-1}\mathbf{g}]$. The second term is a modification of the Newton direction such that the Hessian becomes positive definite. α is a scalar that is used to ensure this transformation.

The trust region implementation in Matlab uses the dog leg strategy. Since the DGN has update schemes for the trust region size and the model updates, it is only required that the Matlab implementation minimize the approximated objective function. This was ensured by setting a trust region size limitation and specifying a single iteration count for *fmincon* after setting it up to use a trust region algorithm.

Steihaug-Toint algorithm

The conjugate gradient method forms the basis for the Steihaug -Toint algorithm. The premise behind the method is that there exist a set of orthonormal directions along which such mini-

mization can be carried out. In its basic form, it can be used to solve linear systems $\mathbf{Ax} = \mathbf{b}$. This can be posed as a quadratic minimization problem of the form

$$\frac{1}{2}\mathbf{x}^T\mathbf{Ax} - \mathbf{bx} \tag{2.27}$$

This is because minimizing requires determining $\mathbf{x}$ that causes the first order derivative to vanish which is the solution to the linear system. If the Hessian information is incorporated, this is known as the conjugate gradient Newton scheme. This scheme minimizes the quadratic

$$\frac{1}{2}\mathbf{x}^T\mathbf{Hx} + \mathbf{g}^T\mathbf{x} \tag{2.28}$$

The difference between the quadratic forms in Equations 2.27 and 2.28 is the use of Hessian information. Since this method is to be used within bounds, curvature information is included in the stopping criteria. This algorithm can deal with non-positive definite Hessian matrices in the conjugate gradient method [Steihaug, 1983]. This stems from the termination criteria that are used. Since a negative definite Hessian means that function is decreasing, the current search point is projected on the trust region boundary along its current direction. If, during determination of the updates, the update over shoots the trust region boundary, it is projected back on the boundary along the same direction. From an implementation perspective, the following pseudo-code was used.

Algorithm 2 Steihaug-Toint algorithm for solving trust region methods

$z_0 = 0$, $r_0 = \nabla f$, $d_0 = -\nabla f$
if $||r_0||_2 < \epsilon$ **then**
    $p = 0$
end if
for j=1,2,3... **do**
    if $d_{j-1}^T B d_{j-1} \leq 0$ **then**
        $p = \arg\min_p \{q(p) \mid p = z_j + \tau d_{j-1},\ ||p||_2 = \triangle\}$
        end for
    end if
    $\alpha_j = \frac{r^T r}{d_{j-1}^T B d_{j-1}}$
    $z_j = z_{j-1} + \alpha d_{j-1}$
    if then$||z_j||_2 > \triangle$
        $p = \arg\min_p \{q(p) \mid p = z_{j-1} + \tau d_{j-1},\ ||p||_2 = \triangle\}$
        end for
    end if
    $r_j = r_{j-1} + \alpha_j B d_{j-1}$
    if then$||r_j||_2 < \epsilon$
        $p = z_j$

        end for
    end if
    $\beta = \frac{r_j^T r_j}{r_{j-1}^T r_{j-1}}$
    $d_j = -r_j + \beta_j d_{j-1}$
end for

2.7 Uncertainty characterization

One aim in this thesis is to quantify uncertainty in the production so as to differentiate results of the DGN with those of ES-MDA. This means the posterior parameter ensemble needs to be compared. One way to do this is through differences of the tuning parameters with the 'truth model' and following up with statistics such as mean and standard deviation to understand the spread.

Multi-dimensional scaling (MDS) can also be used for this characterization. It is a form of parameter reduction scheme that preserves relative position of the ensemble as it is translated to a lower space. While PCA is also a parameter reduction scheme, it transforms the ensemble such that they lie along directions of maximum variance- which is not the aim here. With the preserved distances, a scatter plot of the ensemble can be made which gives the relative spatial positioning of the ensemble. If the posterior from DGN and ES-MDA are plotted on the same graph, then clustering can be used as a measure of variability.

MDS uses a measure known as dissimilarity to transform the ensemble. The dissimilarity is formed using the posterior permeability ensemble. It is a matrix that contains the pairwise 2-norm distances of the ensemble from the truth. These distances have no physical meaning and hence non-metric MDS is used. The transformation is such that the distances between the ensemble are preserved. This is ensured through the use of a *stress* condition, defined as-

$$stress = \sqrt{\frac{\sum (f(x) - d)^2}{\sum d^2}} \tag{2.29}$$

where,

$f(x)$ is the transformed dissimilarity
d is the original dissimilarity

Simply put, Equation 2.29 compares original dissimilarity matrix to the transformed one. A lower stress indicates that the dissimilarity is better preserved. Kruskal (1992) indicates some guidelines which are shown in Table 2.1. The *mdscale* function in Matlab is used to perform MDS in this thesis.

Table 2.1: Relation between stress and goodness of fit

Stress	Fit
20	poor
10	fair
5	good
2.5	excellent
0	perfect

Chapter 3

The DGN- An exposition

An attractive feature of the DGN is the reduced number of function evaluations (simulation jobs) that need to be performed to find a maximum a priori (MAP) estimate [Gao et al., 2016a]. The MAP represents the mode of the posterior distribution. It can then be used to reconstruct the posterior distribution using the model covariance matrix. They believe this could be a result of information sharing between the base cases during iterations of the DGN. To reach a MAP estimate, good quality [1] updates to the model parameters are necessary. This depends on the accuracy of the estimated local model as well as the method used to minimize it. The local model, in turn, is constructed using the gradient and Hessian of the objective function; both of which depend on the local sensitivity. Data sharing has an influence on the quality of this estimate. Tracking back far enough, the local sensitivity appears to play a significant role in determination of MAP estimates. Hence, this chapter begins with an analysis of the sensitivity determination and the associated role played by data sharing. This is followed by other aspects of the DGN, in the order in which they are used if one were to step through the work flow shown in Figure 2.1.

3.1 Calculating local sensitivity

Local sensitivity or Jacobian represents the rate of change of the simulated data with model parameters. The DGN uses linear regression to calculate it. This adds a minimum value constraint to the ensemble size which can be dealt with using methods described in Section 2.5.

The sensitivity calculation is shown in Equation 3.1. By definition, $\mathbf{J}$ contains as many rows as number of tuned parameters (N_p) and as many columns as measurements (used in history matching). To obtain a unique $\mathbf{J}$, the number of rows it contains must equal the the number of rows in $\mathbf{A}$. Since $\mathbf{A}$ consists of differences in model parameters, the minimum number of models required is $(N_p + 1)$. In the interest of reducing required computational power, the ensemble size used is not significantly larger than this minimum. This would mean that other ensemble members are used in the sensitivity calculation for any given member.

$$\mathbf{AJ} = \mathbf{B} \tag{3.1}$$

[1] Quality refers to match between objective function and its approximation. Good quality updates imply better approximation

Table 3.1: Gradient comparison at $x = 1$

x	gradient	
	$\frac{\triangle y}{\triangle x}$	$\frac{dy}{dx}$
2	3	2
3	4	2
4	5	2
5	6	2

$$\mathbf{J} = [\mathbf{J_1 J_2 J_3} \dots \mathbf{J_{N_p}}]^T \tag{3.2}$$

$$\mathbf{B} = [\mathbf{dy_1 dy_2} \dots \mathbf{dy_{N_p}}]^T \tag{3.3}$$

$$\mathbf{A} = \begin{bmatrix} dx_{11} & \cdots & dx_{1_{N_p}} \\ \vdots & \ddots & \vdots \\ dx_{N_p,1} & \cdots & dx_{N_p,N_p} \end{bmatrix} \tag{3.4}$$

$$dx_{ij} = x_{ij} - x \quad \mathbf{dy}_i = \mathbf{y}_i - \mathbf{y} \tag{3.5}$$

where

i, j are counters that cycle through the equations and the parameters respectively.
$x_{i,1\to N_p}$ results in y_i.
$\mathbf{y}_i$ may be a row vector such as $[y_{i1}, y_{i2} \dots y_{i,N_d}]$.

Say the ensemble is size is the minimum required value, i.e., $(N_p + 1)$. In the system formed by Equation 3.1, this is true if and only if the inverse of the coefficient matrix, $\mathbf{A}$, exists. This requires that $\mathbf{A}$ be square and non-singular. The restriction on the ensemble size ensures 'squareness' of the coefficient matrix. Non-singularity is achieved only if its rows are non-degenerate, i.e., no row can be expressed as a linear combination of other rows. To ensure the second condition, it is necessary to increase the minimum ensemble size. Testing on a synthetic reservoir model indicates an extra 30 ensemble members are sufficient.

Thus, an ensemble size larger than the number of tuned parameters is available. Since regression is used to determine the local sensitivity, using ensemble members that are closer can increase accuracy of estimation. As an analogy, consider the function $y = x^2$ for which gradients are being estimated at $x = 1$. Since there is just a single parameter, $\mathbf{x}$, only one point is required for estimation. Consider finite difference gradients being estimated at $x = 1$ with a set of points $x_1 = 2, 5$ with increments of 1. When compared with the actual gradient at the point, $2x$, Table 3.1 shows that finite difference gradients become more accurate as the distance from $x = 1$ decreases. The same is true in the local sensitivity estimation as well.

Having established the need for selecting models based on their distance for sensitivity determination, the next step is to weed out degenerate rows (if any). This was done by applying Gauss elimination to the coefficient matrix. In the absence of degenerate rows, this gives a coefficient matrix of the form shown in Equation 3.6. If say rows 2 and 3 are degenerate, then the third row will comprise of only zeros. The first n_p non-zero rows of this transformed coefficient matrix

are taken to form the linear system and will yield a unique solution for local sensitivity.

$$\mathbf{A}_{gauss} = \begin{bmatrix} 1 & a_{12} & a_{13} & \cdots & a_{1,N_p} \\ 0 & 1 & a_{13} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & & \vdots \\ 0 & 0 & \cdots & & a_{nn} \end{bmatrix} \tag{3.6}$$

where,

a_{ij} are some scalars

At the start of the DGN, the initial ensemble comprises the only available samples in the model parameter space for which the measured data are evaluated. Using the sensitivity determined above, say the intermediate steps leading to determination of updates have been carried out. These are then evaluated through reservoir simulation (for comparison of the approximated and actual objective function). Now, the sampled region of the model space has increased. If a selection of models based on distances (2-norm) is done in this 'sampled space' for use in Equation 3.1, a more accurate sensitivity could be found. Since the updates are determined through an approximate model which is ultimately dependent on this sensitivity, better quality updates can be generated.

Effectively, this means that updates of one model can be used to find the sensitivity of another one. In this manner, data sharing is achieved in the DGN. A storage variable known as the 'training data set' is used for this purpose. To prevent loss of data (sampled region of the model parameter space), it contains all the models and their updates regardless of whether or not the update was successful. The information stored is relevant to operations within the DGN such as objective function determination and sensitivity calculation. Thus, it includes values of the model parameters and the simulated data corresponding to measurements.

3.2 Determining updates

Having calculated the sensitivity, the next step is to construct a local model to approximate the objective function. The model chosen is a second order polynomial function obtained by truncating higher order terms in the Taylor series expansion. This model is then minimized to determine updates through methods in 2.6.

Let x be the vector of parameters that characterizes the local model with $\nabla\mathbf{f}, \mathbf{H}$ representing the gradient and Hessian of the objective function respectively. Then the change in model parameters, $\triangle\mathbf{x}$ can be expressed Equation 3.7. The condition in the equation is a constraint that is algorithm dependent. If line search algorithms are used, then the constraint is a maximum step length that can be taken along the search direction. If trust region methods are used, then it could be a 2-norm condition that keeps the update within ball shaped bounds.

$$\triangle\mathbf{x} = \arg\min_{\mathbf{x}} \ \mathbf{x}^T\mathbf{H}\mathbf{x} + \nabla\mathbf{f}^T\mathbf{x}, \quad \mathbf{x} \in \mathbf{x}_{valid} \tag{3.7}$$

Consider line search algorithms. From Section 2.6, the updated model parameters can be expressed as Equation 2.23. The steepest descent technique is based on the first order derivative and moving against the direction of maximum change in objective function. This means the

first term in Equation 3.7 is ignored. α is a step length which determines the magnitude of update. In trust region methods, both the dog-leg and Steihaug-Toint algorithm are used. The former was implemented using Matlab functions while the latter was programmed using the described pseudo code. The choice of algorithm used at this stage has a significant impact on the quality of updates that found.

3.3 Convergence criteria

This step evaluates termination criteria necessary to stop the DGN. [Gao et al., 2016a] indicate three possible criteria. They suggest the second criteria may not be suitable owing to numerical noise and propose the use of the third instead. This could be because the gradient is formed using local sensitivity which in turn is calculated with linear regression. The use of regression implies the sensitivity may not be accurate which would make the gradient an unreliable condition. The third condition is easier to evaluate. Further, if the step size taken is sufficiently small, the resulting change in objective function as a result will be negligible. The first and third criteria are used for termination of the DGN in this thesis.

1. Decrease in objective function between successive iterations is lower than a threshold value
2. Normalized norm of gradient is less than a threshold
3. Step size is lower than a threshold

It is necessary to ensure that the conditions used are satisfied at the same time. If the local model used does not sufficiently match the actual objective function, then the update may be rejected leading to no change in objective function. This model would then be considered converged when there is still room for improvement.

3.4 Quality of updates

The updates generated in the previous iteration of the DGN are simulated using the forward model are evaluated in this step. The approximated objective function for the updated model is compared with the objective function using the ratio ρ.

$$\rho = \frac{f_{i,k} - f_{i,k+1}}{f_{i,k} - q_{i,k}} \tag{3.8}$$

where,

subscripts i, k refer to the i^{th} ensemble member at the k^{th} iteration
f is the objective function
q is the estimate from the quadratic minimization

Because comparison of values are made across iterations, the necessary notations are defined. The subscript k represents value in the k^{th} iteration. f, q denote the objective function and its approximation respectively. Say, model parameters in the k^{th} iteration are used to find f_k, then its approximation q_k is determined in the same iteration to generate updated model parameters. These model updates are evaluated in the $(k+1)^{th}$ iteration where ρ_{k+1} is calculated to determine quality of updates.

q_k will always be less than or equal to f_k. This is because the update is generated conditional to obtaining a decrease in the approximation. If $f_{k+1} < f_k$, the updated model parameters are successful in decreasing the objective function and the update can be accepted ($\rho > 0$). If, on the other hand, $\rho < 0$, the update is rejected. In either case, there is a need to modify associated values in preparation for the subsequent iteration. This is defined in Equation 3.9.

$$\begin{aligned}
\text{if } \rho_{(i,k)} \geqslant \eta_v \text{ and } ||s^i|| > 0.5\triangle^{(i,k)} \quad & \begin{cases} \mathbf{x}_{(i,k+1)} = \mathbf{x}_{(i,k)} + s^i \\ \triangle^{(i,k+1)} = \gamma_e \triangle^{(i,k)} \end{cases} \\
\text{if } \rho_{(i,k)} > \eta_s \text{ or } \rho_{(i,k)} > \eta_v \text{ and } s^i \leqslant \triangle^{(i,k)} \quad & \begin{cases} \mathbf{x}_{(i,k+1)} = \mathbf{x}_{(i,k)} + s^i \\ \triangle^{(i,k+1)} = \triangle^{(i,k)} \end{cases} \\
\text{else} \quad & \begin{cases} \mathbf{x}_{(i,k+1)} = \mathbf{x}_{(i,k)} \\ \triangle^{(i,k+1)} = \triangle^{(i,k)} \end{cases}
\end{aligned} \tag{3.9}$$

In Equation 3.9, i refers to an ensemble member, $\triangle$ is size of the trust region for the i^{th} model in the k^{th} iteration. s refers to the magnitude of update. γ_e is the increase in trust region size in case a feasible point is found. η_v,η_s are constants to judge the quality of the update. Typically, $\eta_s = 0$ meaning the update is accepted as long as both q_k and f_{k+1} predict a decrease in objective. However, to enforce a more stringent update criterion, η_s may be a set to a small positive number.

Once the model parameters are updated, the local sensitivity is determined for these new models and the sections covered thus far in this chapter repeat till the convergence criteria are satisfied.

Chapter 4

Experiments

The previous chapter details the steps followed in the DGN. In this chapter, testing of the workflow is carried out. First, it is necessary to ensure the implemented workflow is correct. This is done through tests on toy models. The verified workflow is then tested on a synthetic reservoir model. Since the goal is to implement the DGN for history matching, a form of sensitivity analysis is done on the parameters of the DGN to determine their impact on results. Further, parameter reduction schemes discussed in Section 2.5 are implemented to understand how they affect results. With the optimal settings determined in the sensitivity study, different optimization schemes (for update generation) are tested. Armed with a good algorithm and optimal settings, the benchmark study [Emerick and Reynolds, 2013b] is approached. This study considers a non-linear history matching problem and analyses different history matching methods such as the MCMC (Markov Chain Monte Carlo), RML (Randomized Maximum Likelihood), EnKF and some of its variants. Using the problem stated here, the DGN is tested and its results added. This is the order of discussion in the chapter.

4.1 Toy Problems

The history matching objective function consists of the model term and the data term. A few different toy models are used for this purpose. First is the problem referenced in [Gao et al., 2016a] followed by the two forms of the Rosenbrock function. Since these models verify functioning of the implemented workflow, the objective function used is simplified- the use of prior information, which is a form of regularization, is ignored. Regularization reduces the possible solutions that can be found. Since the solutions for the toy problems are finite and known, this is unnecessary. The absence of prior information simplifies Equations 2.1, 2.2 and 2.5 where the terms containing the prior are omitted. In the absence of prior information, the initial ensemble is assumed to be uniformly distributed in the model space.

$\mathbf{C_D}$ contains information on measurement errors. In the objective function, it acts as a form of weighting between the data error and model error. Without prior information, there is no need for such a weight and hence $\mathbf{C_D}$ is assumed to be an identity matrix (of appropriate size). This reduces the objective function to a simple sum of squares.

Note: The toy problems have few parameters and can be visualized in a co-ordinate system. Hence, the ensemble members are referred to as points.

4.1.1 Toy Problem from literature

[Gao et al., 2016a] used a sinusoidal time dependent function of two model parameters to test the ability of DGN to determine multiple optima. It was chosen for this thesis so that a reference is available with which results can be compared. Thus, the same parameter settings are used where possible.

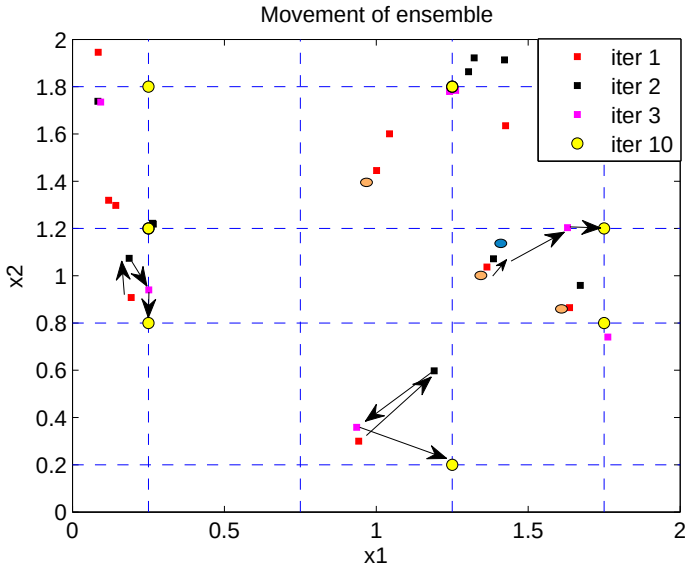

Figure 4.1: Results of testing the DGN on a toy problem. The intersection of the blue dashed lines represent local minima. The points in red are the initial ensemble. The arrows indicate how the ensemble members move over the iterations. The colored ellipses are used for description.

The functional form of the model is shown in Equation 4.1. At any time t, there exist multiple points in the parameter space with the same function value. With one point as reference, the DGN attempts to find all these other points.

$$m(x,t) = 2sin^2(\pi x_1)sin(t) + 6cos^2(\pi x_2)cos(t) \tag{4.1}$$

where,

x_1, x_2 are parameters in the space $[0,2] \times [0,2]$
$t \in [0,10]$ represents time

Within this space, the reference point was defined as $[k_1 \pm 0.25, k_2 \pm 0.2]$ where $k_1, k_2 \in I$. This identifies a set of 16 points with identical function value. Using one of these as reference, the DGN is used to find the remaining ones. The function value defined at the reference point is the observed data used for comparison. The goal is to minimize the objective function defined below. $f = 0$ at the 16 points defined above and are the local minima the DGN attempts to find.

$$f = \sum [m(x,t) - m(x_{obs},t)]^2$$

where,

x is a model for which the objective function is evaluated
x_{obs} is the reference point.

Table 4.1: Compares results between documented results and those of the workflow. A total of 16 minima exist

Ensemble size	Iterations	Number of minima found	
		literature	work flow
10	10	7	7
20	12	9	9
40	8	15	13
60	9	16	16

A number of tests are performed with varying ensemble sizes. With an ensemble size of 60, most of the models converged by the 9^{th} iteration. At this stage, a majority of the models had an objective function value less than 10^{-4}. These results match those reported in [Gao et al., 2016a] (Table 4.1).

To illustrate the trajectory towards local minima, an ensemble of 10 members is used. Figure 4.1 illustrates this. Since this is a 2-variable problem, two nearby points are needed for sensitivity calculations. The two points (red squares) denoted by the outer two orange ellipses are the closest to the point denoted by the central orange ellipse. The update is the black square denoted by the blue ellipse. In the next iteration, the two closest points for the blue ellipse are the lower two orange ellipses. These determine the next update shown by the pink square that lies due north east.

Figure 4.2 shows the results of the simulation when the ensemble size is varied. It can be observed that with increasing size, the ensemble is able to better span the parameter space resulting in the determination of more local minima.

4.1.2 Rosenbrock functions

Results from the previous toy problem indicate a successful implementation of the DGN since the results found match those reported by [Gao et al., 2016a]. To ensure this, two more tests are done- based on the Rosenbrock function. The first test is done on the two variable form while the other uses the three variable form. While the former is done to verify the workflow, the latter is done to test working of a more generic form of the implementation. The functional form of the Rosenbrock function used is shown below.

$$f(x_1, x_2) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2$$

$$f(x_1, x_2, x_3) = (1 - x_1)^2 + (1 - x_2)^2 + 100(x_2 - x_1^2)^2 + 100(x_3 - x_2^2)^2$$

The parameters were defined as $x_1, x_2 \in [-2, 2]$ for the two parameter case and $x_1, x_2, x_3 \in [-1, 1]$ for the three parameter case. The smaller size for the latter case stems from the shape of the iso-surface in the parameter space. Using a larger parameter range does not affect results.

The Rosenbrock function has one global minimum in both cases. The aim is not to find this point. Instead, an arbitrary point is chosen in the parameter space (which is not the global minimum) as reference. The work flow uses this reference to find all other points in the parameter space where the Rosenbrock function takes an identical function value. The nature

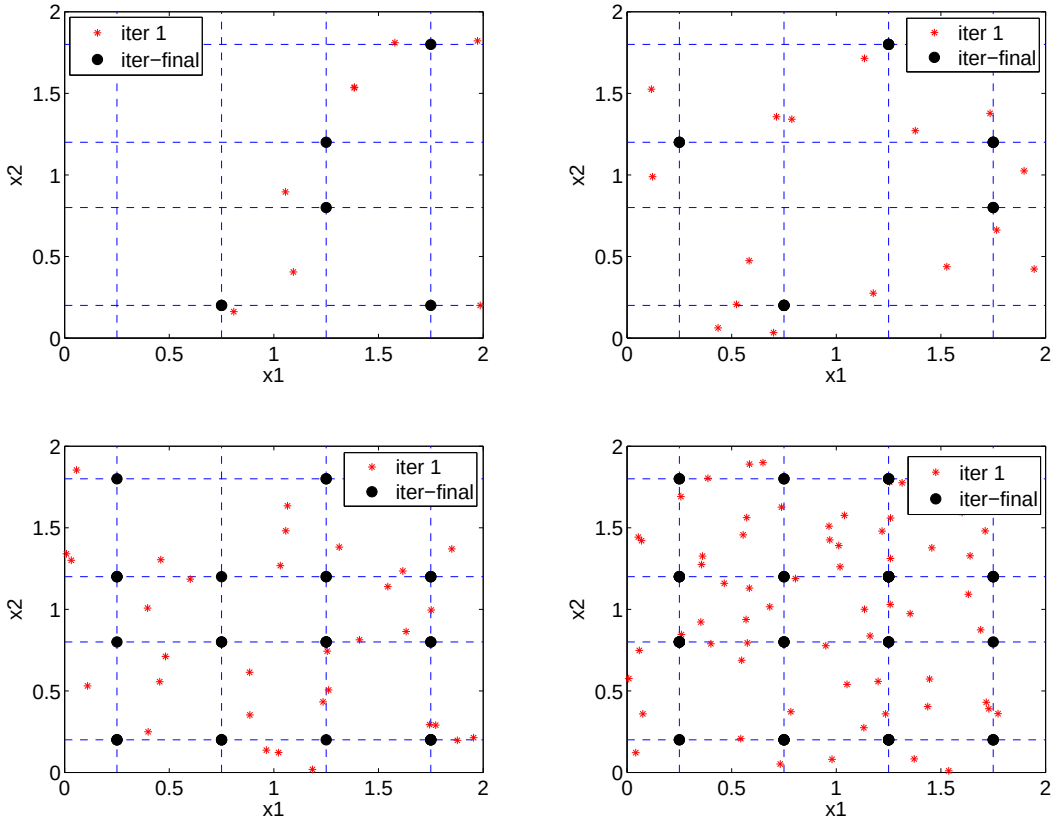

Figure 4.2: Comparison of results when ensemble size is varied. From top left, clockwise, the ensemble size are 8, 16, 64, 32.

of the function results in this being an exercise in identifying iso-lines (in 2-D) and iso-surfaces (in 3-D).

The objective function defined in Section 4.1.1 is used. The reference points are $x_{ref} = [0.01, -0.69]$ in 2-D and $x_{ref} = [0.1515, 0.3737, 0.0505]$ in 3-D. In two dimensions, the Rosenbrock function comprises rings of varying function value (Figure 4.3). Moving along the ring keeps the function value constant while cutting across it results in variation of function value. For this test, a point in the ring is chosen and DGN determines all other points on the ring.

The number of minima that exist is much larger than in the previous toy problem. In order to be capable of identifying as many as possible, the spatial distribution of the initial ensemble becomes important. Figures 4.4-4.5 shows the difference when individual variables (that form a distribution) are combined against one with uniform spatial distribution.

In both cases, the red points represent the initial ensemble while the black unfilled circles are the minima. More minima are found when the spatial distribution is uniform. Thus, a better spread in initial distribution of the ensemble can aid in determining more minima. If multiple reference points are chosen such that they lie on different rings, then an image similar to Figure 4.3 can be found.

In three dimensions, the rings (in 2-D) gain depth resulting in a surface. The surface may be

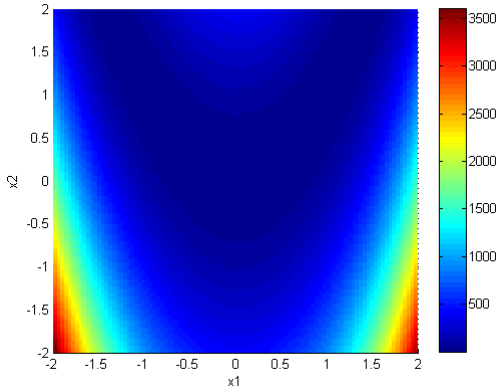

Figure 4.3: Contour plot of the bi-variable Rosenbrock function in the defined parameter space

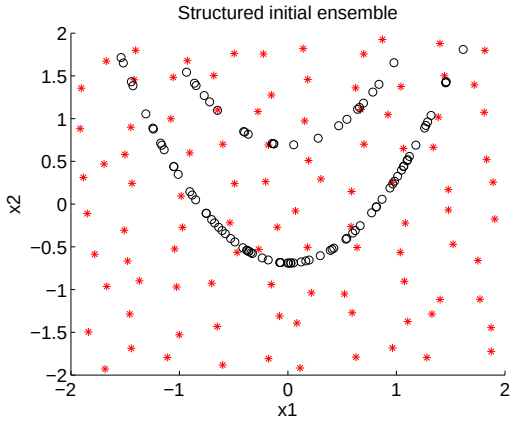

Figure 4.4: Initial (red) and converged (black) ensemble are compared. The curves formed by the converged ensemble are iso-lines. The initial ensemble is grid based.

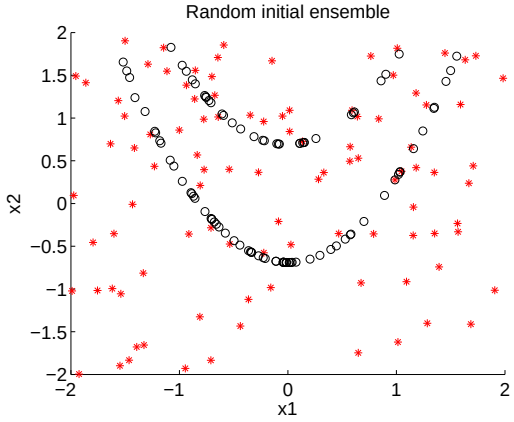

Figure 4.5: Initial (red) and converged (black) ensemble are compared. The curves formed by the converged ensemble are iso-lines. The initial ensemble is formed as a combination of uniform distributions

open or closed depending on the location of the reference point. To adequately visualize this

surface, sufficient sample points are required. Hence, an ensemble of 10000 points were used. To determine whether the solution obtained was correct, a package known as Sliceomatic (from Matlab File Exchange) was used. The surfaces generated by the DGN and using the external package are compared in Figure 4.6. It can be observed that the surfaces are quite similar validating the DGN implementation.

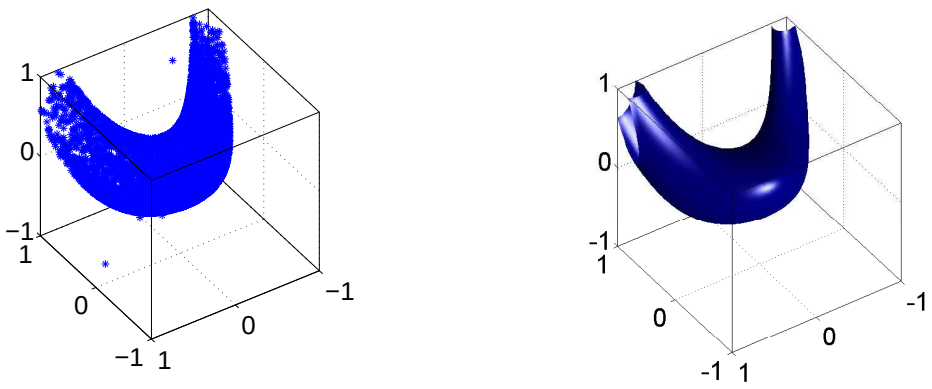

Figure 4.6: The cluster of points in blue on the left inset form the 3-D surface captured by the DGN. The blue surface in the right inset is the actual 3-D surface generated using Sliceomatic.

4.1.3 Location of initial ensemble

During testing with the toy model, it was observed that the spread of the initial ensemble had an impact on the number of minima found. To identify if the initial spread of the ensemble had an effect, it was necessary to have a uniform distribution of the ensemble in the parameter space. The ensemble was initially created using a uniform distribution function for each parameter separately and then combining the two parameters to form the ensemble.

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N_p} \\ x_{21} & x_{22} & \cdots & x_{2N_p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N_e1} & x_{N_e2} & \cdots & x_{N_eN_p} \end{bmatrix} \tag{4.2}$$

Consider the matrix, X. The columns correspond to the parameters while the rows represent the starting points. Say each parameter forms a uniform distribution within its limits and the parameters are individually uniformly distributed. Let the ensemble size be N_e. Therefore $N_e \times N_p$ variables are combined to form N_e points. Each of the N_e points contain a value for each of the N_p parameters. While each of these N_p are uniformly distributed, it was found that the spatial distribution of the N_e points (formed by combination of N_p parameters) was not always uniform.

To obtain consistent spatially uniform ensembles, a grid-like distribution of the points with perturbations was considered. The scatter of the initial ensemble for both cases is shown in 4.7 (The image shows the initial ensemble of the toy problem discussed in Section 4.1.1. It is used for illustrative purposes). Using a grid based distribution can generate a consistently

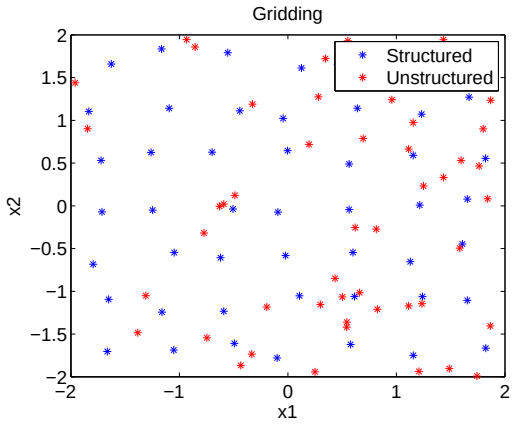

Figure 4.7: Distribution of models in the initial ensemble. The dots are generated by combining the uniform distributions created for each variable. The stars are created by introducing perturbations in a square gridding

even spread across the parameter space while a combined distribution can result in clustering. Figures 4.4-4.5 indicate the impact of this on results.

The latter distribution of the initial ensemble was successful at finding minima that were close to the starting points. This demonstrates that the location of a model in the parameter space does not hinder its ability to converge to an optimum.

4.2 2-D Synthetic reservoir model

Once the implementation of the DGN was verified with the toy problems, a synthetic reservoir model was chosen for testing. The forward modeling was done using MRST [Lie, 2014] while the EnKF [Leeuwenburgh, 2013] package was used to prepare data for use in MRST. The model is a 5-spot incompressible reservoir with one injector in the centre of the field and 4 producers at the corners. The reservoir was discretized uniformly into square grids blocks with 21 cells a side. This results in a total of 441 cells. The reservoir is simulated for 14 years with a time step of half a year. A constant bottom hole pressure of 300 bar is maintained in the producing wells while the injector has a constant water injection rate of 150 m^3/day. The history matched data could be pressure in the injector, flux rates or water cut in the producing wells. The parameters being tuned are the permeability in the grid cells. Figure 4.8 shows the permeability distribution of the 'truth' case in the 5-spot model.

The full form of the objective function defined in Equation 2.1 is used. The cross co-variance matrix C_M is found using the parameter matrix x. C_D is a diagonal matrix containing the variances of the observed data.

This section starts with a sensitivity study of the parameters that could affect results of the DGN. The pseudo-inverse was used in the tests to reduce the ensemble size. This is followed by tests that employ principal component analysis which leads to why different optimization schemes were considered.

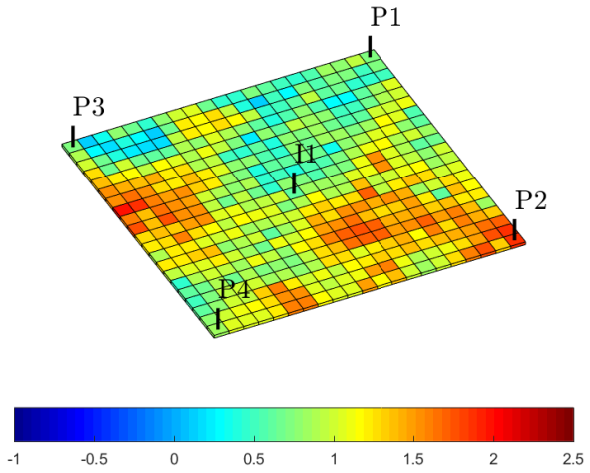

Figure 4.8: Permeability distribution in the truth model. It also shows the reservoir shape and well locations.

4.2.1 Parameter sensitivity

Preliminary testing of the model showed that some parameters used in the work flow may have an impact on the results. To understand which settings provide the best results, a series of tests were performed. Only one parameter was modified per test. To make comparisons, a standard model was defined. Here, the prior is chosen to be the mean of the starting ensemble and the initial trust region size was set to 0.6. Further, data (for history matching) were collected only once (at the end of the history match period, i.e., at 14 years). For these tests, only water cut measurements was used in history matching. Table 4.2 contains a list of documented tests for this part.

- Weights were added to the components of the objective function
- The initial trust region size was modified
- The size of the parameter space was adjusted
- The choice of prior used was changed
- The number of assimilated data was increased
- Perturbations were added to the observations

To expedite the rate at which tests are performed, it was necessary to limit the ensemble size. This can be achieved through the use of methods discussed in Section 2.5. For the sensitivity study, pseudo inverse was adopted to reduce the ensemble size. The aim was to use the optimal parameters found in the study in a full rank scheme, i.e., an ensemble size larger than number of tuned parameters, to determine their effectiveness.

The results obtained from tests during the sensitivity study were such that using objective function plots do not add sufficient information. So, tables are used to quantify improvements in the objective function. The numbers in the first row (improvement) are the improvement

Table 4.2: List of pseudo inverse experiments

Test code	inital trust size	# of data data error	weights- model error, scaling	scaling
		Base case		
t_base	0.6	1	1,1	1
		Effect of weights		
t_simp1	0.6	1	0.2,0.8	1
t_simp2	0.6	1	0.01,0.99	1
t_numb	0.6	1	$1/N_p, 1/N_d$	1
		Effect of prior		
t_prbasecase525	0.6	1	1,1	1
		Movement of ensemble		
t_trust	2	1	1,1	1
t_scale	0.6	1	1,1	4
		Number of observations		
t_obs4	0.6	4	1,1	1
t_obs14	0.6	14	1,1	1
t_obs28	0.6	28	1,1	1
		Perturbation of data		
t_pert	0.6	1	1,1	1
		Full rank tests		
t_full	2	28	1,1	1

(with respect to initial value) in the function being compared in %. The numbers in the rows corresponding to the iterations indicate the number of ensemble members that show an improvement (rounded to upper 10%) corresponding to the respective column. In the interest of brevity only the results at the 20^{th} iteration are shown in these tables. Iteration 20 is used since improvements in the objective function become negligibly small after this. A more detailed table can be found in Appendix B.

Effect of weighting on the objective function

Preliminary tests showed that the reduction in data error was much smaller than in model error. Weights were added to the model error and data error in a effort to increase the effect of the latter. Initial testing was done with simple weights (multiplication factors) followed by ones that used the number of parameters involved, i.e., the model error was weighted with the inverse of the number of parameters tuned while the data error was weighted with the inverse of the number of observations used.

Two sets of simple weights were used. In the first, the model weight was set at 0.1 with the data weight at 0.9. Next, the effect of the model error was further suppressed by decreasing its weight to 0.01 while that for the data error was increased to 0.99. This was followed by weighting with number of parameters/data used.

From Table 4.3, it can be seen that weighting the objective function has an adverse effect on

the results. When the number of data is used as weights, it was observed that increasing the observed data count resulted in better solutions as long as the model weight and the data weight were comparable. This indicates weights do not improve the history matching in the DGN.

Table 4.3: Objective function improvement, iteration 20
Weighted vs Base case

Improvement	10	20	30	40	50	60	70	80	90	100
t_simp1	168	21	9	2	0	0	0	0	0	0
t_simp2	9	20	31	40	45	24	16	8	2	5
t_numb	167	22	9	2	0	0	0	0	0	0
t_base	0	0	0	0	4	13	34	90	43	16

Movement of ensemble

This test deals with movement of the ensemble members (in the parameter space) towards the prior, in other words, it looks at the magnitude of update. This can be accelerated by increasing the size of the trust region (creates opportunities for finding an optimal point farther away from its initial location) or by decreasing the range of the parameter space. Increasing trust region size for a given parameter range and decreasing the parameter range for a given trust region size were explored.

The parameter range was reduced by using a scaling factor. Prior to scaling, the range is $-0.87 \rightarrow 2.62$. With a scaling factor of 4, this is reduced to $-0.21 \rightarrow .65$. The increased value of the initial trust region is 2. From Table 4.4, it can be seen that there is comparable decrease in objective function in both methods with respect to the base case. Thus, the improvement in the data mismatch is now compared. It should be noted that base cases that show an increase in data error are not listed in the table. Table 4.4 indicates that the scaling parameters test actually results in an increase in data error for a majority of the base cases. While decrease in objective function is the primary aim, it should be not be at the cost of increase in data data error. Hence, scaling of parameters is rejected. The test t_trust performs better than t_base in terms of model error decrease. This means increasing initial trust region size allows for determining updates that are farther from the initial position thus accelerating rate of convergence. Since both tests have comparable decrease in data error, increasing trust region size is considered a superior method.

Effect of perturbations on data

Data perturbation has its roots in the formulation [Burgers et al., 1998] of the ensemble Kalman Filter. In the DGN, there are no definitions that require perturbation of the measured data. However, these measurements are not error-free and hence using these directly may not accurately sample the posterior distribution. Using perturbed measurements, on the other hand, allows for generating reservoir models with data that span the range of possible values the measurements can take. So, perturbations were added to the data.

Table 4.4: Objective function and components' improvement, iteration 20
Base case vs Increased trust region size vs Smaller parameter range

Improvement	10	20	30	40	50	60	70	80	90	100
Objective function										
t_base	0	0	0	0	4	13	34	90	43	16
t_scale	0	0	1	1	2	26	53	75	30	12
t_trust	0	0	0	0	1	7	31	96	52	13
Data error										
t_base	46	49	22	30	10	3	2	1	2	3
t_scale	16	2	1	0	0	0	0	0	0	0
t_trust	55	48	22	29	11	4	5	0	0	1
Model error										
t_base	0	0	0	0	0	0	2	10	73	115
t_trust	0	0	0	0	0	0	1	0	59	140

The perturbation had a mean value of zero and a standard deviation that equals its error which is 0.1 for water cut measurements. The size of this modified data was such that each set of perturbed observations correspond to an ensemble member. The results obtained as a result are compared with the base case scenario in Table 4.5.

Table 4.5: Objective function improvement, iteration 20
Perturbed observations vs base case

Improvement	10	20	30	40	50	60	70	80	90	100
t_pert	0	0	0	0	6	8	42	83	44	17
t_base	0	0	0	0	4	13	34	90	43	16

Effect of prior

In the test thus far, the prior used was the mean of the initial ensemble. Reduction of objective function includes reducing model error and this means the ensemble moves towards the prior. It was believed that using a prior that resembles a reservoir model may improve results. Further, if the prior could be chosen as the initial ensemble itself, then the history matched models that are found will necessarily be as diverse as the initial ensemble resulting in a better characterization of uncertainty. Table 4.6 shows the level of history matching when different priors are used. It can be seen that using an ensemble member as prior provides superior results- both model error and data error reductions are higher. When the initial ensemble was used as the prior, model error at the start is zero. To find updates, the model needs to move from its initial position. The resulting increase in model error has to be offset by a decrease in data error for the update to be accepted. Testing showed that this rarely happened. Allowing for a successful first iteration implies a non-zero model error in subsequent iterations and consequently scope for improvement of the data error. So, the objective function was artificially increased by a factor of 10 in the first iteration and allowed to iterate. While the results found in this manner indicated some improvement, they were not as successful as test t_prbasecase525. This can be observed in the table.

Table 4.6: Objective function and data error improvement, iteration 20
Perturbed observations vs base case

Improvement	10	20	30	40	50	60	70	80	90	100
Objective function										
t_prmean	0	0	0	0	4	13	34	90	43	16
t_prbasecase525	0	0	0	0	0	0	0	8	98	94
t_prinit	139	31	21	3	3	2	1	0	0	0
Data error										
t_base	46	49	22	30	10	3	2	1	2	3
t_prbasecase525	38	40	26	29	22	7	3	3	2	2
t_prinit	73	40	25	29	11	3	1	2	0	0

Effect of number of observations

The base case used for testing involved only the use of one observation time, i.e., data from the four wells at the end of the history match period. The main use of history matching in the field is to predict and plan for future production. At this stage, a fair amount of data is usually available. Due to the non-linear relationship between model parameters and the data, an increase in amount of data does not imply an increase in information, i.e., history matching may not be improved. Tests in this section were aimed at understanding the impact of increase in data on the DGN.

Four different data observation times are compared with a gradual increase in frequency of observation. The first test records data at the 14^{th} year (base case) followed by an increase in frequency to every 3.5 years, every year and every half a year. The results are shown in Table 4.7. The reduction in objective function and data error becomes pronounced with increasing observation times. The opposite in true in case of model error. This indicates that the reduction in data error becomes more important with increasing observations. Test t_obs28 breaks from this trend. There is no certain theory for why this takes place.

Summary

Based on the simulations thus far, the following observations are made.

- Weighting does not improve convergence rate
- Scaling parameters increases rate at which ensemble approaches prior and has a significant impact on the model error but does little to improve data error.
- Increasing trust region allows for faster convergence rates. It provides some decrease in data error.
- Introducing perturbations in data results in a decrease in data error.
- Increase in number of observations significantly decreases objective function.

Table 4.7: Objective function and components improvement, iteration 20
Increasing observation times compared

Improvement	10	20	30	40	50	60	70	80	90	100
Objective function										
t_base	0	0	0	0	4	13	34	90	43	16
t_obs4	0	0	0	11	10	38	40	47	41	13
t_obs14	4	18	33	15	4	3	1	1	19	102
t_obs28	35	38	33	8	3	0	0	2	6	75
Data error										
t_base	46	49	22	30	10	3	2	1	2	3
t_obs4	25	29	31	37	29	20	16	7	3	0
t_obs14	27	34	12	2	2	3	0	6	29	84
t_obs28	56	42	15	4	1	0	1	1	7	73
Model error										
t_base	0	0	0	0	0	0	2	10	73	115
t_obs4	0	0	0	0	0	2	9	17	83	89
t_obs14	0	1	2	13	18	22	16	5	11	112
t_obs28	15	20	9	28	25	18	1	1	7	76

Table 4.8: Objective function improvement, iter 20
Full rank test

Improvement	10	20	30	40	50	60	70	80	90	100
t_full	260	154	46	15	12	5	4	2	2	0

4.2.2 Full rank test

The sensitivity study showed that increasing the number of observations and the trust region size can be used to accelerate decrease in objective function. So the next test is to incorporate these parameters into a full rank test, i.e., ensemble size larger than number of tuned parameters. Since the tuned parameter is permeability in grid cells, there are 441 such parameters and so an ensemble of 500 members is used. The results are shown in Table 4.8. It was observed that the results obtained were not as successful as when the pseudo inverse was used. At this stage, there was only one plausible reason- the local sensitivity calculation using the pseudo inverse was flawed. The linear system solving for sensitivity was made under-determined to reduce ensemble size. The least norm solution that was found as a result does not sufficiently resemble the true solution leading to poor update generation. The next step was to apply a different parameter reduction scheme- principal components analysis (PCA). During testing of PCA, a trend in results was observed that indicated that the above hypothesis could be incorrect.

4.2.3 Principal Component Analysis

Another form of decreasing the ensemble size is through the implementation of PCA. This results in transformation of the model parameter space. In this problem, the tuned parameter is permeability in each of the grid blocks (441 grids in the field) and consequently the model

space has 441 dimensions. Using PCA, it is possible to reduce the model space to a single dimension. Singular value decomposition (SVD) was used for this purpose.

PCA finds 441 orthonormal variance maximizing direction vectors along which the model parameters lie. These direction vectors were used to reconstruct the original ensemble through linear combination and hence are referred to as basis vectors. If fewer (< 441) basis vectors are used to generate an ensemble, then the new model space is smaller. However, the effect of un-included directions are lost leading to a loss of information.

An initial permeability data set is available for the reservoir model. The log permeability of this set had a mean value of 1 with a standard deviation of 0.3 and was not conditioned to well data. To have an accurate estimate of the covariance matrix for PCA using SVD, it is necessary to centre the data set. This involves subtracting the mean from it. The basis vectors thus found are stored in u while corresponding the singular values are stored in a (Equation 2.16).

Testing showed that different basis vectors have different effects on the observations, i.e., the first basis vector had a greater effect on bottom hole pressures while the second one influenced the water cut and so on. Hence, the first 5 basis vectors are used during initial testing.

Preliminary test

- The first 5 basis vectors were used to create the ensemble
- A single observation time was used, i.e., at 14 years
- Three types of observations were used for history matching, i.e., total flow rate and water cut at producing wells and bottom hole pressure at injector
- An ensemble size of 20 was used
- Mean of the ensemble was used as prior

Figure 4.9 shows the decrease in normalized objective function with iterations followed by its contributing terms. It can be seen that the contribution from the model error is insignificant in comparison to the data error. The reduction in objective, is thus, primarily a result of decrease in data error. This can also be seen in the production profiles where a clustering of grey lines (ensemble) around the red ('truth') takes place. Figure 4.10 show the prior and posterior water cut profiles in the producing wells. Pressure and flux profiles can be found in Appendix-C. This indicates that history matching is successful. These profiles were taken at the end of 30 DGN iterations. To increase the rate of convergence, the size of the initial trust region was increased from 0.2 to 0.4. The production charts at 15 DGN iterations for the latter case are marginally better than the the former at 30 DGN iterations. Similarly, when the number of data assimilated was increased, the level of history matching (at 15 DGN iterations) improved. The new frequency of data collection was once every half a year. The objective function curves show a significant improvement (Figure 4.12). The posterior water cut profiles in the above two cases are compared in Figure 4.11. The pressure and flux rate comparisons can be found in Appendix-C. This indicates that decreasing data collection intervals and increasing trust region size improve history matching in the DGN, which is consistent with results from the pseudo inverse tests.

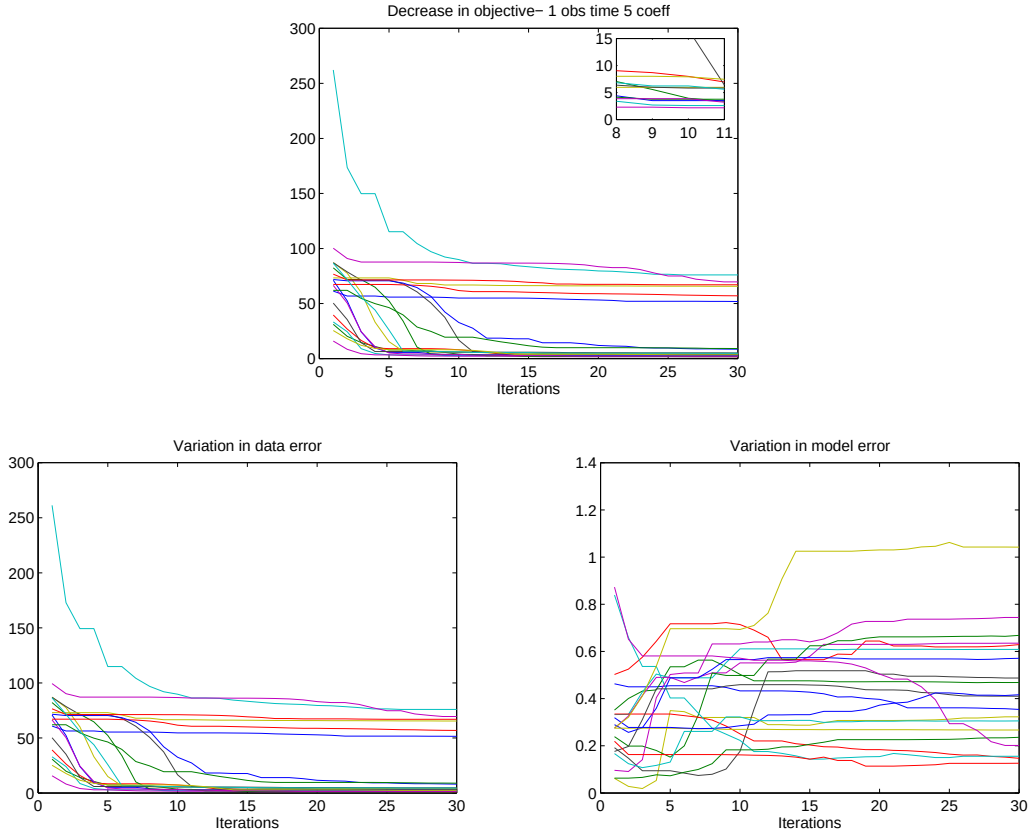

Figure 4.9: Decrease in normalized objective function and its components over iterations when 5 PCA coefficients are used.

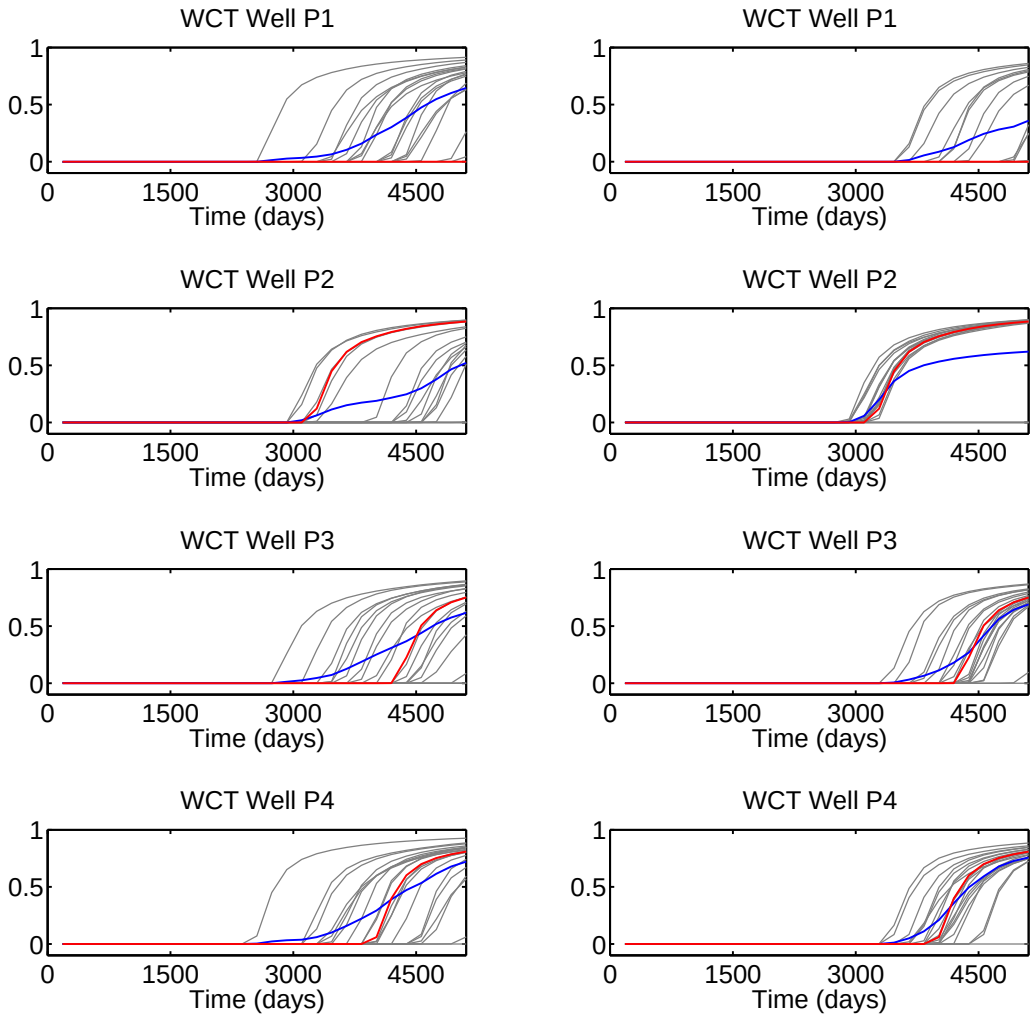

Figure 4.10: Water cut profiles before and after history matching (30 DGN iterations) in the producing wells. The figures on the left are the prior while the right correspond to the posterior.

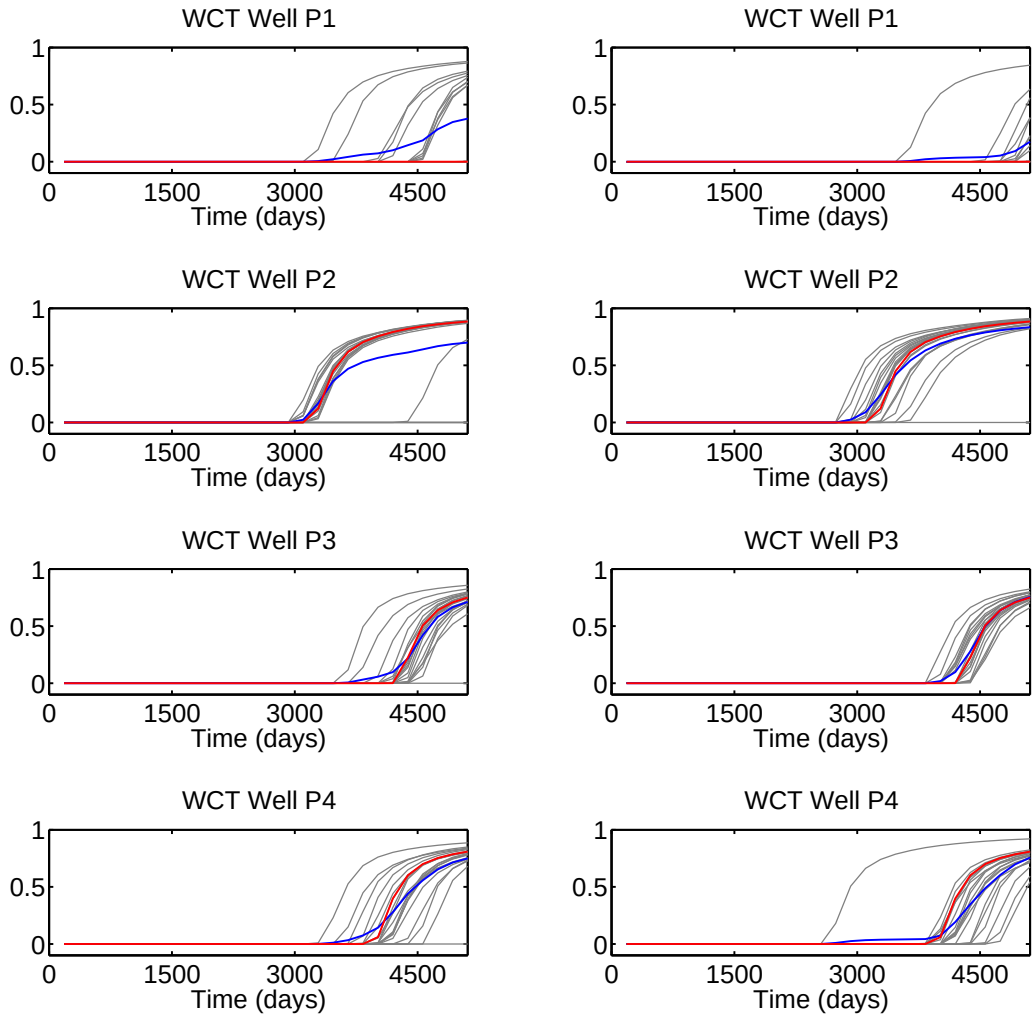

Figure 4.11: Water cut profiles after history matching (15 DGN iterations) in the producing wells. The figures on the left are obtained when using a larger initial trust region while the right correspond to increased (28) observation times.

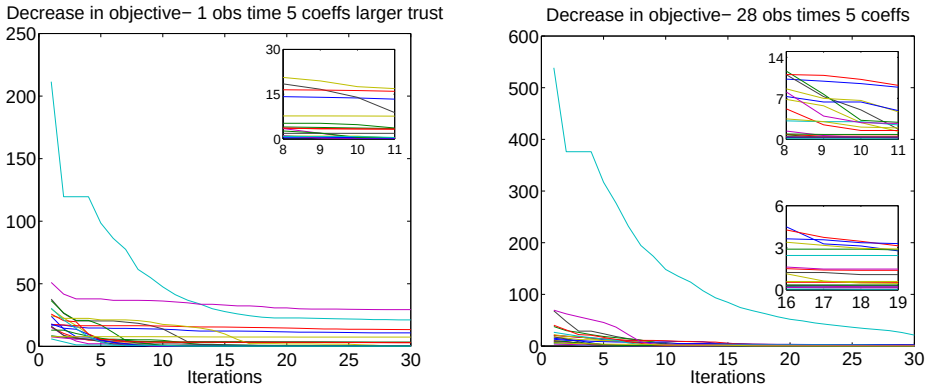

Figure 4.12: Variation in normalized objective function with iterations. The figure to the left results when a larger initial trust region size is used with a single observation time while the one on the right is the result of using 28 observation times. The insets in each figure are magnified slices of corresponding parts.

Table 4.9: Variation in retained variance with size of reduced parameter space

Number of PCA multipliers	Retained variance (in %)
5	47.65
15	73.74
50	87.57
157	95.42
223	97.41
309	98.97

Number of PCA coefficients

In the previous test, only the first 5 basis vectors were used out of the available 441. While PCA allows for parameter space reduction, it is necessary to use sufficient parameters to ensure the reduced space sufficiently describes the original problem. This can be done by calculating the retained variance in the reduced model space with Equation 2.20. For this problem, Table 4.9 shows the correspondence between number of coefficients and retained variance.

[Gao et al., 2016a] used the PCA to reduce the model space in a facies based history matching problem but do not mention the retained variance parameter. This is true in most cases where PCA was used in history matching. They simply state that the reduced space chosen retains characteristics of the original problem. Yadav [Yadav et al., 2006] employs PCA to history match a fluvial depositional environment and uses 99% retained variance. In a patent by Schlumberger where reduced model space methods based on the PCA were implemented, a value of 75% retained variance was used. In other fields such as image processing, where parameter reduction methods are used, the size of the reduced space is determined such that 99% of the variance is retained. In the absence of clear guidelines on the required value of retained variance, multiple tests were performed with increasing number of basis vectors (or PCA coefficients or PCA multipliers). To ensure best results are obtained, the data acquisition was made frequent (to once every half a year) and the initial trust region size was increased to 0.4.

Table 4.10: Objective function and data error improvement, iteration 8
Dog-leg vs Steihaug-Toint

Improvement	10	20	30	40	50	60	70	80	90	100
Objective function- 157 PCA multipliers										
Dog-leg	64	47	34	22	12	6	10	5	0	0
Steihaug-Toint	2	1	1	1	3	2	7	11	25	147
(prior=mean of ensemble)										
Steihaug-Toint	2	0	1	0	1	2	0	4	14	176
(prior=initial ensemble)										

These tests were done with the multipliers listed in Table 4.9. It was seen that increasing the size of the model space has an adverse impact on the ability of the DGN to produce results. This can be observed in the normalized objective function curves (Figure 4.13). From the plots it can be observed that the number of stagnating base cases increases with increasing number of PCA coefficients. This is similar to the results obtained in the full rank test where most of the base cases were stagnating. This indicated that there could be some other underlying factor causing this. To confirm this, a simulation was made within the PCA framework which used the pseudo inverse. The results obtained are comparable with and without its use. This negates the hypothesis in the previous section where it was thought the use of pseudo-inverse could be the reason for poor reductions in objective function when a shift was made from pseudo inverse to full rank tests. Thus, the pseudo inverse remains a viable option to reduce ensemble size.

4.2.4 Choice of algorithm

Thus far, updates in all the tests were obtained using the trust region implementation from the optimization toolbox in Mablab. This was the part of the workflow that remained untouched since the start. So, a shift was made to a different trust region solver. This was the Steihaug-Toint conjugate gradient algorithm referred to in Section 2.6. This was developed by Steihaug and Toint in 1983 where it was described. The theory in Nocedal and Wright provides a background to understand the method while also detailing the algorithm. [Conn et al., 2000] also provides a commentary on the method. The new trust region solver was verified using a Rosenbrock-like minimization problem. Appendix D shows the ability of the solver to find global optima and follows up with a comparison between this solver and the one in Matlab.

With the new trust region solver, the above tests were repeated. Table 4.10 shows the improvement in objective function and data error for the two algorithms. As a comparison, the objective function curve is shown for both algorithms (Figure 4.14) when 157 PCA coefficients were used. DGN parameter sensitivity showed that using the initial ensemble as prior gave poor improvements in objective function. With the new solver, this is no longer the case and using the initial ensemble as prior provides comparable reduction in objective function (to using mean of the initial ensemble as prior). This can be seen in the table below.

Even with the new solver, it was observed that the number of stagnating base cases (although fewer) increased with increasing PCA multipliers. For these stagnating base cases, the trust region size is modified to understand its behavior. It was observed that at larger trust region

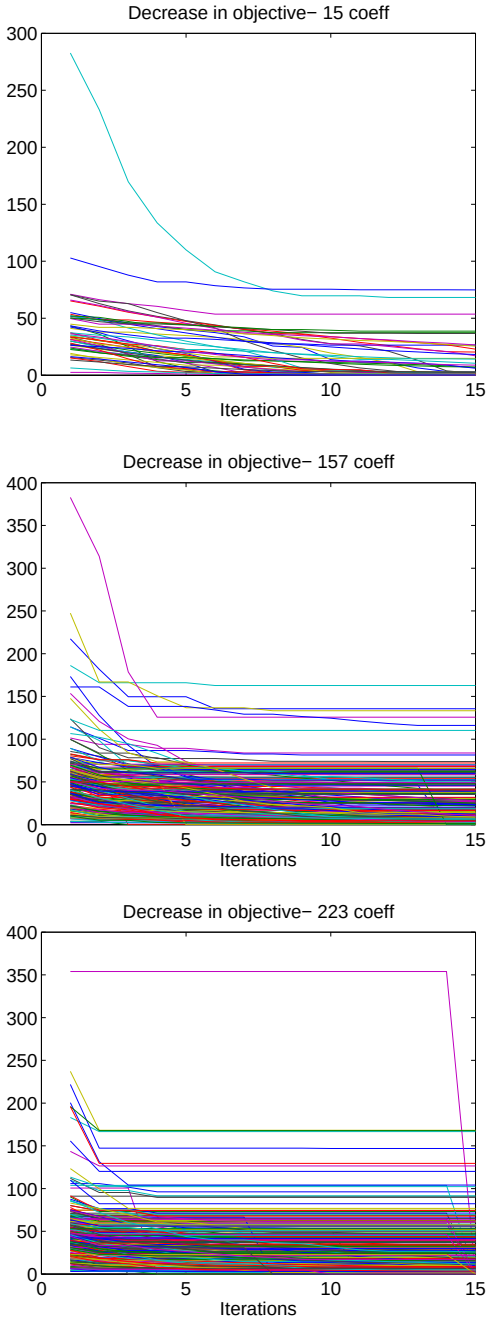

Figure 4.13: Variation in objective function over iterations. The figures, from top, correspond to 15, 157 and 223 PCA co-efficients.

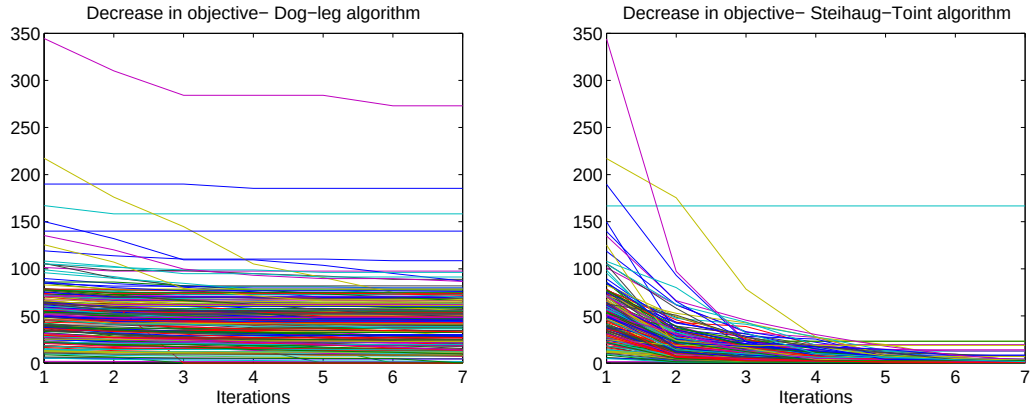

Figure 4.14: Comparing results of the DGN when different trust region methods are used. The image to the left corresponds to the dog-leg algorithm while the one on the right is for the Steihaug-Toint algorithm

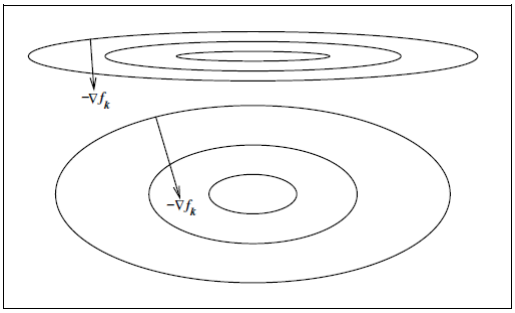

Figure 4.15: The contours with smaller minor axes are poorly scaled in comparison to the ones with larger minor axes. Image taken from Nocedal and Wright

sizes, the updates generated were poor, i.e., the approximated objective function does not sufficiently match the actual objective. As the trust region size decreases, the approximation improves; however, this is not sufficient to generate updates. One possible reason for this is that the model is 'stuck' in a valley of the objective function (Figure 4.15). The later part of Appendix D indicates why this hypothesis could be true.

In optimization, valleys in objective function can be seen in cases where the parameters being updated/tuned are significantly different in magnitude. This is also true if the function is locally more sensitive to changes in value of one parameter over another. These valleys may be visualized as contours with thin ellipsoidal sections (Figure 4.15), i.e., parameter changes in one direction affect the objective more than other directions. In such cases, preconditioners can be used to reparametrize the model space for update generation such that more spherical contours are formed. This improves the quadratic model approximation and hence provides better updates. Some preliminary testing has been done with preconditioners such as the incomplete modified Cholesky factorization and diagonal of the Hessian matrix. It was seen that the magnitude of update becomes very small as a result of the preconditioning resulting in increased number of DGN iterations that are necessary. This may be overcome by selecting a suitably large trust region size. However detailed investigations were not carried out.

Since the solution to the problem appears to lie in the type of solver used to generate updates, a simple steepest descent algorithm was also tested. Preliminary testing indicated that no acceptable updates were generated and hence further testing was stopped.

4.2.5 Comparison with ES-MDA

Now that an implementation of the DGN is obtained, the next step is to compare the results of the DGN with those of ES-MDA. The new trust region solver was used for this purpose. The ES-MDA was simulated with the EnKF package in MRST using the same (reduced space) permeability set as the DGN. The comparison is with respect to number of reservoir simulations required and variability in posterior. While convergence criteria controls the number of simulations used in the DGN, it is fixed for the ES-MDA. Metric space methods are used for posterior variability contrasting.

For the comparison study, the PCA test with 157 coefficients is used. Here, the prior is chosen to be the initial ensemble and data collection takes place at intervals of half a year. An initial trust region size of 3 is used.

Computational comparisons

In the ES-MDA, the number of iterations of the smoother is fixed at the start of history matching. Here, 4 iterations were used along with an ensemble size of 200. Thus, the number of function evaluations (reservoir simulations) used in the ES-MDA is 4×200. Oliver and Reynolds postulate that models that sample the posterior distribution have a maximum value of normalized objective function defined in Equation 4.3. In light of stringent values used for convergence criteria in the DGN, this value is used instead. Thus, a model is considered converged if its normalized objective function value is lower than the limit defined in Equation 4.3. The DGN uses 800 function evaluations during the 5^{th} iteration. At this point, only 115 of the 200 base cases have converged. This value reaches 158 at 8 iterations and a maximum of 167 with the remaining base cases 'stagnating' at high objective function values. For purposes of comparison 8 iterations of the DGN are considered. Thus the DGN uses 1085 function evaluations against the 800 in the ES-MDA. Flux profiles of the posterior are shown in Figure 4.17. Clearly, the ES-MDA provides a better history match compared to the DGN while also requiring fewer function evaluations.

$$O_N \leq 1 + 5\sqrt{\frac{2}{N_d}} \tag{4.3}$$

where,

O_N is the normalized objective function
N_d is the number of observed data

Variability in posterior

A good history matched distribution should match data well in the history match period and have sufficient variability during the forecast period. In this case, this means that the posterior permeability fields should be quite different. This comparison is made through multi

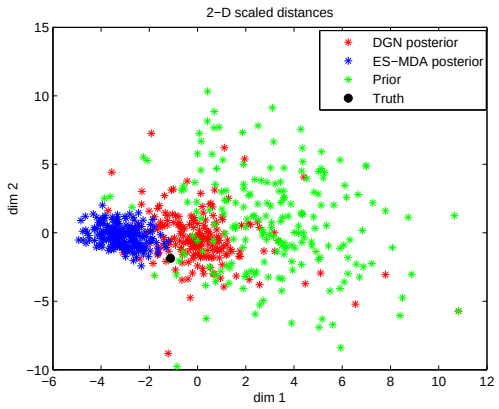

Figure 4.16: Scaled distances of the models

Table 4.11: Error statistics of the tuning parameters

	Permeability (log K)	
	Mean	Standard deviation
Prior	0.9952	0.4232
DGN posterior	1.0149	0.3102
ES-MDA posterior	1.0191	0.2168

dimensional scaling (MDS). Non-metric MDS is employed on the posterior permeability fields to obtain 2D scatter plots that show the relative distances of the models from each other. Figure 4.16 indicates more clustering of the models obtained from the ES-MDA than the DGN. Table 4.11 shows the statistics of the posterior ensembles from both methods. It can be seen that both the DGN and ES-MDA have similar values of mean permeability with the former having a larger standard deviation. This implies that the former has less variability in the posterior and hence a poorer estimate of uncertainty in this case.

As a comparison, the mean permeability field is shown for both methods and compared with the truth (Figure 4.18). It can be observed that ES-MDA mean and DGN mean have high permeability streaks near P2 (producing well 2). In the DGN, a higher than average permeability streak forms along the NE-SW direction near P3 with reduction in permeability at wells I1 and P1. The ES-MDA mean solution has more distinct features compared to that of the DGN. It also resembles the truth to a larger extent. This is indicative of a large part of the ensemble having similar features (to the 'truth'). This in turn could be looked at as a reduction in posterior variability which is in line with inferences from the MDS plot and the statistics table.

The next step is to determine if the posterior distributions from the DGN and ES-MDA are correct. This can be done using comparison with the MCMC or RML, both of which were not done for this test. Since these are computationally intensive, an existing study was chosen where the ES-MDA is studied and the DGN results for that test compared. This is done using the uncertainty quantification study by [Emerick and Reynolds, 2013b]. The next section details the model used in that study and the corresponding results from the DGN.

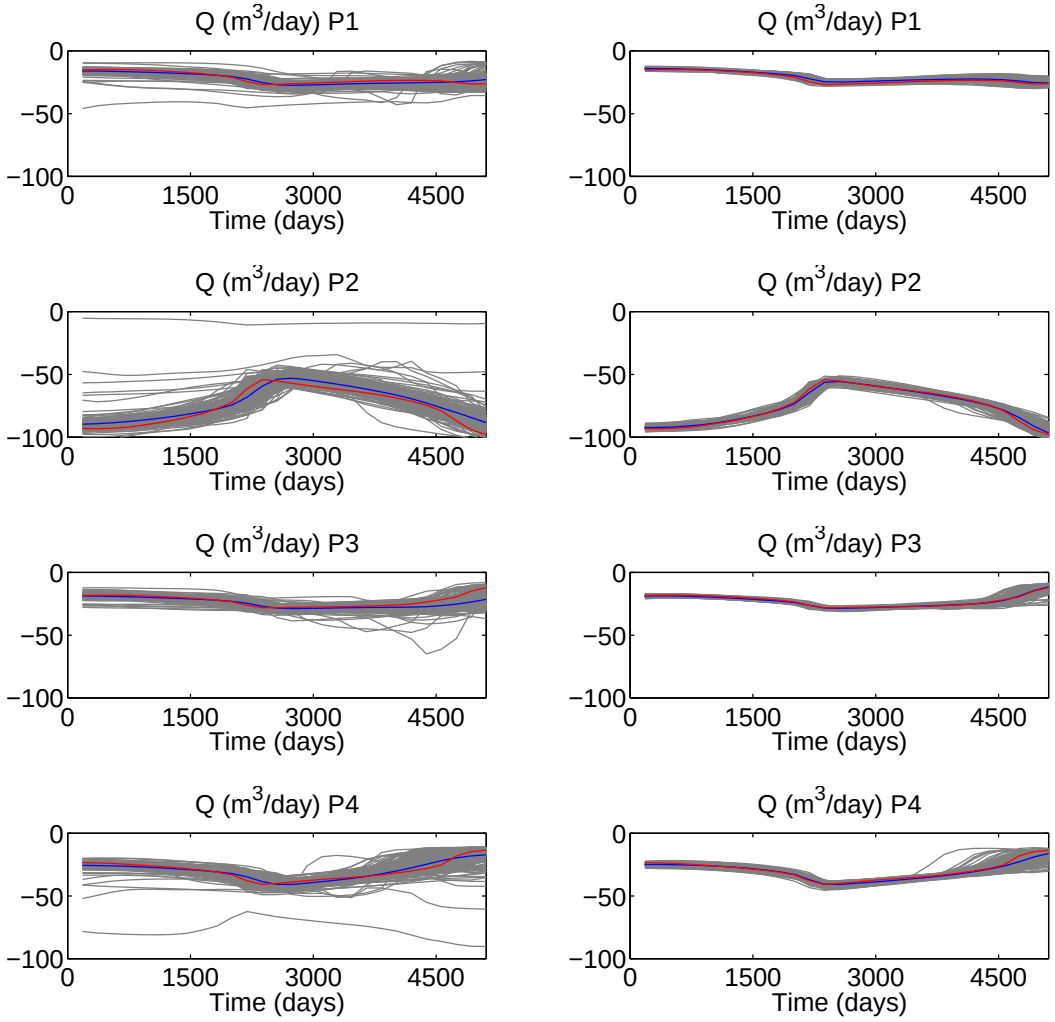

Figure 4.17: Flux profiles after history matching (15 DGN iterations) in the producing wells. The figures on the left are obtained using the DGN while the right corresponds ES-MDA.

4.3 1-D synthetic reservoir model

This model was taken from the uncertainty quantification [Emerick and Reynolds, 2013b] study. In their study, they used a 1-D synthetic compressible reservoir model consisting of 31 grid cells. The data collected for history matching was pressure at the middle grid cell while the parameter being matched was water cut at the producing wells. The reservoir had one producing well at cell 31 and an injector at cell 1. Both wells had specified bottom hole pressures with the injector and producer pressures being 4000 psi and 3000 psi respectively. Pressure data was collected at intervals of 30 days during a history match period of 360 days while the forecast period was 750 days. The observed data was perturbed with noise of zero mean and standard deviation of 1 psi. A total of 10 sets of prior models were available.

The authors made the simulator they used available along with measurements used for history matching and the model covariance matrix. The prior models were split into 10 ensembles of 100 models each. These were incorporated in the DGN to determine the posterior distribution.

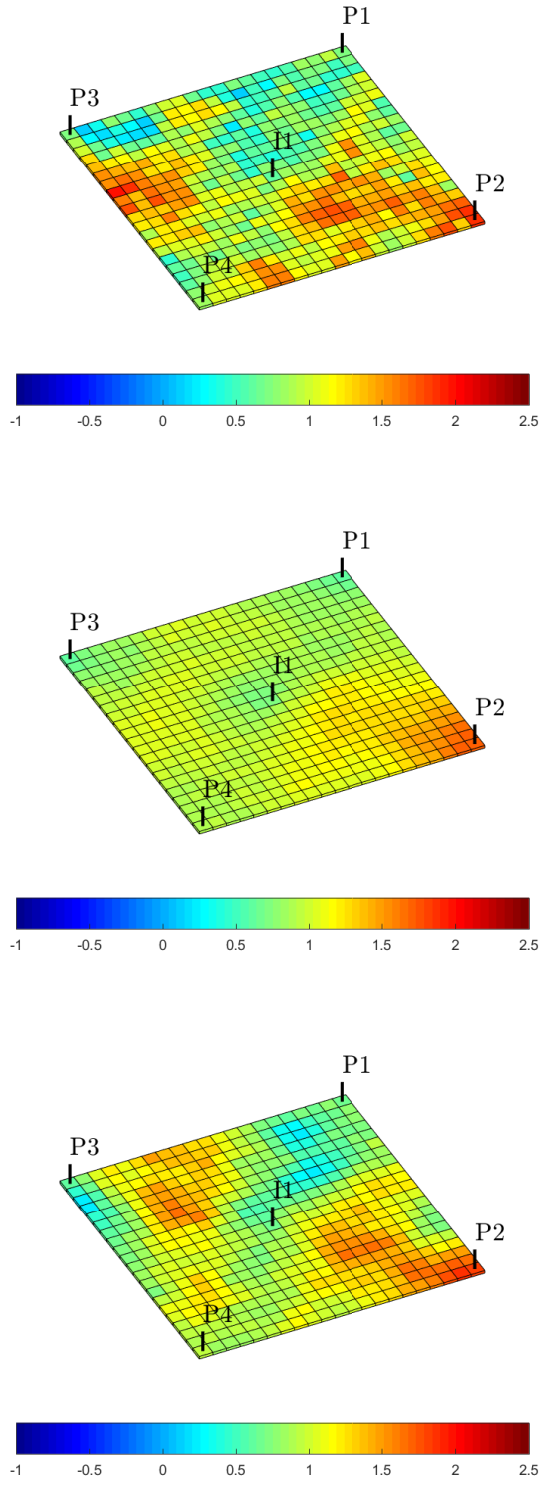

Figure 4.18: Comparing permeability fields. The top one is the truth model while the middle one is the mean value for the DGN posterior while the last corresponds to the mean of the ES-MDA posterior

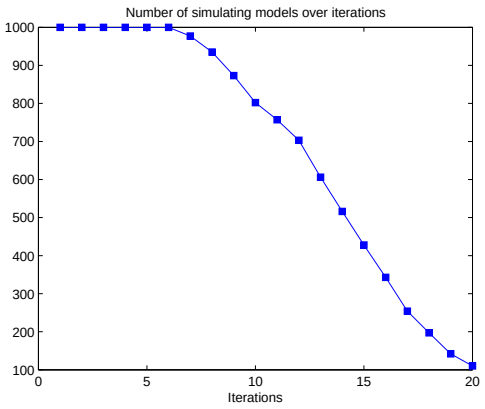

Figure 4.19: Number of simulating models over iterations

In the DGN, the prior was chosen to be the mean of the initial ensemble. The initial trust region size used is 2. The convergence criteria in the DGN were relaxed such that the minimum required objective function improvement for a non-converged base case is 0.01 and the minimum trust region size used is 0.01.

The ES-MDA test for this reservoir used 10 data assimilation steps and hence a total of 10000 function evaluations were needed for all 1000 models. In the DGN, simulation was stopped at 20 iterations. At this point, only a 100 of the original 1000 models were unconverged. Figure 4.19 shows the number of running models over iterations. At the 10^{th} iteration, the DGN has performed 500 fewer function evaluations. Using results from either the 10^{th} or the 11^{th} iteration of the DGN will allow for comparisons at similar computational costs. Further, Figure 4.20 indicates no significant improvement in the normalized objective function after the 11^{th} iteration and so this is used.

The plot shows the base cases with normalized objective function (O_N) in the 25^{th} to 75^{th} percentile in the blue box while the outlier points are shown as a red plus. Outliers are base cases with O_N that is less than 2 percentile or greater than 98 percentile. Significant reduction in value can be seen in the first few iterations for a number of base cases. Neither the higher valued outliers nor the 75^{th} percentile boundary change in value after the 5^{th} iteration. This means that of the 1000 reservoir models being simulated, at least a quarter of them are poorly matched and 'stagnate'.

In their uncertainty estimation study, Emerick and Reynolds compare a number of different history matching techniques (including the ES-MDA) which are bench marked against the MCMC. Their results show that the ES-MDA solution matches the MCMC results quite well. This means that the ES-MDA provides a fairly accurate sampling of the posterior. This thesis uses the graphs in their work and adds the results of the DGN to it to form a comparison. With the choice of results taken from the DGN, differences in posterior can be compared for both methods at roughly the same simulation cost. The posterior distribution is estimated by objective function decrease and permeability plots for the reservoir while uncertainty quantification is done using water production curves.

Figure 4.21 compares the normalized objective function plots for different methods. It can be observed that the RML and the MCMC have bounds that almost overlap at a low objective function value. The ES-MDA (below blue arrow) performs better when minimizing the objec-

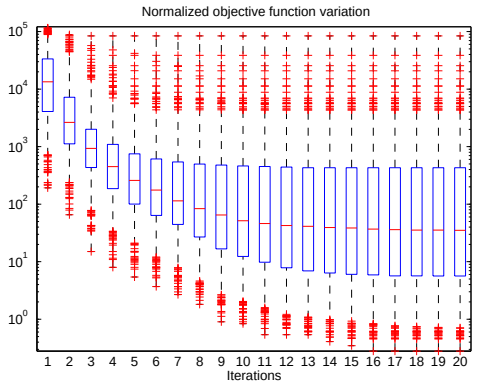

Figure 4.20: Objective function improvement over iterations. The box corresponds to the 25^{th} and 75^{th} percentile while the red line is the median value. The whiskers extend to the extrema. Outlier points are shown in red plus.

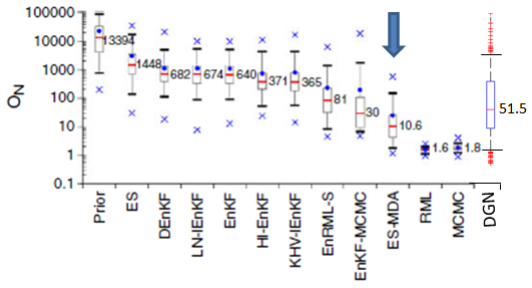

Figure 4.21: Comparison of normalized objective function. The box plot with the arrow corresponds to ES-MDA. Image taken from [Emerick and Reynolds, 2013b].

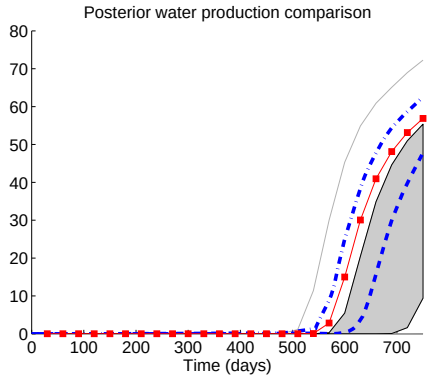

Figure 4.22: Water production forecast for the DGN (grey) is overlaid with the ES-MDA (blue) for comparison. The line in red is the true production. The grey lines correspond to $25^{th}, 75^{th}$ and 98^{th} percentile while the blue are the 2^{nd} and 98^{th} percentile. Image adapted from [Emerick and Reynolds, 2013b].

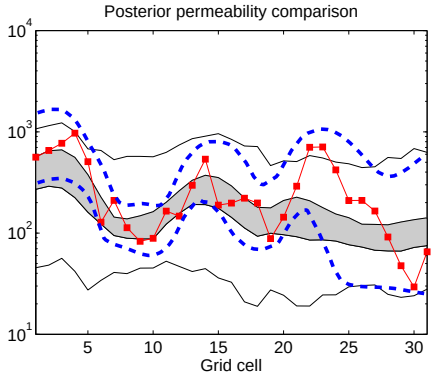

Figure 4.23: Grid cell permeability for the DGN (grey) is overlaid with the ES-MDA (blue) for comparison. The line in red is the true production. The grey lines correspond to $2^{nd}, 25^{th}, 75^{th}$ and 98^{th} percentile while the blue are the 2^{nd} and 98^{th} percentile. Image adapted from [Emerick and Reynolds, 2013b].

tive compared to the DGN-median objective of DGN is 5 times that of ES-MDA. Figure 4.22 compares water production in forecast period. The $2^{nd}, 25^{th}, 75^{th}$and100^{th} percentile of water production is shown. In this case, the first curve lies on the x-axis. The blue lines represent the 2^{nd} and 98^{th} percentile of water production in ES-MDA. The line in red is the true water production. The ES-MDA results are enclosed within the DGN curves. This means the DGN forecasts a larger range of uncertainty; however the ES-MDA is more accurate. Thus, the DGN seems to sample the posterior, albeit with larger variation than is necessary. The same notations are used in plotting the permeability field. Again, the DGN posterior permeability encompasses the ES-MDA permeability distribution (Figure 4.23). However, the DGN predictions are markedly different from the actual posterior. Since a lot of low permeability grids exist in the DGN posterior, this could explain the underestimation of water production in the field (note the low position of the 25^{th} percentile water production curve). Based on these graphs, it can be said that the ES-MDA provides better results than the DGN.

Chapter 5

Conclusions

The objectives of this thesis were twofold, both of which were achieved. First was to develop an implementation of the DGN for use in reservoir history matching. The other was to compare it with an ensemble based method, the ES-MDA.

An attractive feature in the DGN is the ease with which the sensitivity matrix is determined. Since this involves the use of linear regression, it would be necessary to have a minimum ensemble size to find a unique value of local sensitivity. Since this lower limit can become very large in practice, parameter reduction schemes were tested. The pseudo-inverse and PCA were methods of choice; testing showed both methods to be feasible.

In the DGN implementation, the magnitude of update plays in important role in sampling the posterior distribution. If the objective function is considered a hypersurface, the then posterior would be be represented by local minima that have an upper limit, $1+5\sqrt{\frac{2}{N_d}}$. If a large update magnitude is chosen then basin jumping[1] causing some local minima to be missed occurs.

The ability of the DGN to find local minima depends on its update generation scheme for model parameters, i.e., the quadratic model minimization scheme. This thesis predominantly uses trust region methods for this purpose since preliminary testing with the steepest descent algorithm showed little success. It shows that very different results can be obtained when the same algorithm is used; the dog-leg method becomes unsuitable as the size of the model parameter space increases. While the same is true with the conjugate gradient based solver, it performs better than the former.

The DGN and ES-MDA were compared with two different tests. In the first, a 5-spot model was used where it was observed the DGN had a more varied posterior distribution. In the second test, the results of a benchmarking study were used for the ES-MDA. The study showed that the posterior distribution from the ES-MDA approximately matches the MCMC. On the other hand, the posterior from the DGN did not provide as good a match. While the first test showed that the DGN provides a more varied posterior, the second indicates this may not accurately sample the true posterior. The results of the second test should be treated with some caution since I believe there may be scope to improve the trust region scheme currently in use. From a computational perspective, both tests show the ES-MDA require fewer simulation runs to produce comparable if not better results than the DGN.

[1] local minima that are farther away are found rather than closer ones

Chapter 6

Scope for future work

Based on the results of this thesis, the following are possible avenues for future work.

1. The trust region scheme (using the Steihaug-Toint algorithm) performs unconstrained minimization. Implementing bounds on parameter values can allow for more sophisticated tests.
2. This scheme can be improved to yield better results. It is believed that the contours of the objective function are ellipsoidal and hence a ball-shaped trust region would be ill-suited. Preconditioning can transform the parameters such that contours of the objective function become spherical. This should solve the issue of stagnating base cases.
3. The posterior distribution determined will change based on the prior. Different choices of prior were investigated in this work. It is believed that it may not be possible to characterize the posterior using MCMC if the initial ensemble is used as prior. If sufficient validation can be found to show the DGN and MCMC sample the same posterior for a prior mean and variance, then the DGN posterior for the different choice of prior may be deemed correct.
4. [Vink et al., 2015] cite possible pitfalls in posterior determination that arise due to over-estimation of the objective function when a large amount of data is assimilated in history matching. Their recommendations have not been incorporated into this work. This could done to improve posterior sampling by the DGN.
5. The DGN scheme was devised for efficient sampling of the posterior distribution in history matching problems. This is done through minimization of an objective function to find multiple local minima. The DGN, can thus be considered an optimization scheme. Hence, a possible line of research could be benchmarking in a generic optimization context.

Bibliography

[Aanonsen et al., 2009] Aanonsen, S. I., Nævdal, G., Oliver, D. S., Reynolds, A. C., Vallès, B., et al. (2009). The ensemble kalman filter in reservoir engineering–a review. *Spe Journal*, 14(03):393–412.

[Burgers et al., 1998] Burgers, G., Jan van Leeuwen, P., and Evensen, G. (1998). Analysis scheme in the ensemble kalman filter. *Monthly weather review*, 126(6):1719–1724.

[Conn et al., 2000] Conn, A. R., Gould, N. I., and Toint, P. L. (2000). *Trust region methods*. SIAM.

[Emerick and Reynolds, 2012] Emerick, A. A. and Reynolds, A. C. (2012). History matching time-lapse seismic data using the ensemble kalman filter with multiple data assimilations. *Computational Geosciences*, 16(3):639–659.

[Emerick and Reynolds, 2013a] Emerick, A. A. and Reynolds, A. C. (2013a). Ensemble smoother with multiple data assimilation. *Computers & Geosciences*, 55:3–15.

[Emerick and Reynolds, 2013b] Emerick, A. A. and Reynolds, A. C. (2013b). Investigation of the sampling performance of ensemble-based methods with a simple reservoir model. *Computational Geosciences*, 17(2):325.

[Evensen, 2003] Evensen, G. (2003). The ensemble kalman filter: Theoretical formulation and practical implementation. *Ocean dynamics*, 53(4):343–367.

[Fenwick and Batycky, 2011] Fenwick, D. and Batycky, R. (2011). Using metric space methods to analyse reservoir uncertainty. In *Proceedings of the 2011 Gussow Conference*.

[Gao et al., 2016a] Gao, G., Vink, J., Chen, C., El Khamra, Y., and Tarrahi, M. (2016a). Distributed gauss-newton method for history matching problems with multiple best matches. In *ECMOR XV-15th European Conference on the Mathematics of Oil Recovery*.

[Gao et al., 2016b] Gao, G., Vink, J. C., Chen, C., Alpak, F. O., Du, K., et al. (2016b). A parallelized and hybrid data-integration algorithm for history matching of geologically complex reservoirs. *SPE Journal*, 21(06):2–155.

[Gao et al., 2016c] Gao, G., Vink, J. C., Chen, C., Tarrahi, M., El Khamra, Y., et al. (2016c). Uncertainty quantification for history matching problems with multiple best matches using a distributed gauss-newton method. In *SPE Annual Technical Conference and Exhibition*. Society of Petroleum Engineers.

[Leeuwenburgh, 2013] Leeuwenburgh, O. (2013). Enkf module for mrst-isapp. *URL¡ http://www. isapp2. com/data-sharepoint/enkf-module-for-mrst*.

[Lie, 2014] Lie, K.-A. (2014). An introduction to reservoir simulation using matlab: user guide for the matlab reservoir simulation toolbox (mrst). sintef ict.

[Liu and Oliver, 2004] Liu, N. and Oliver, D. S. (2004). Experimental assessment of gradual deformation method. *Mathematical geology*, 36(1):65–77.

[Oliver and Chen, 2011] Oliver, D. S. and Chen, Y. (2011). Recent progress on reservoir history matching: a review. *Computational Geosciences*, 15(1):185–221.

[Oliver et al., 2008] Oliver, D. S., Reynolds, A. C., and Liu, N. (2008). *Inverse theory for petroleum reservoir characterization and history matching*. Cambridge University Press.

[Planitz, 1979] Planitz, M. (1979). Inconsistent systems of linear equations. *The Mathematical Gazette*, 63(425):181–185.

[Roggero et al., 1998] Roggero, F., Hu, L., et al. (1998). Gradual deformation of continuous geostatistical models for history matching. In *SPE annual technical conference and exhibition*. Society of Petroleum Engineers.

[Rommelse, 2009] Rommelse, J. R. (2009). Data assimilation in reservoir management.

[Steihaug, 1983] Steihaug, T. (1983). The conjugate gradient method and trust regions in large scale optimization. *SIAM Journal on Numerical Analysis*, 20(3):626–637.

[Vink et al., 2015] Vink, J. C., Gao, G., Chen, C., et al. (2015). Bayesian style history matching: Another way to under-estimate forecast uncertainty? In *SPE Annual Technical Conference and Exhibition*. Society of Petroleum Engineers.

[Wright and Nocedal, 1999] Wright, S. J. and Nocedal, J. (1999). Numerical optimization. *Springer Science*, 35(67-68):7.

[Yadav et al., 2006] Yadav, S. et al. (2006). History matching using face-recognition technique based on principal component analysis. In *SPE Annual Technical Conference and Exhibition*. Society of Petroleum Engineers.

Appendix A

Deriving the objective function gradients

The derivative for an expression of the form $\nabla(\mathbf{AB})$ is $\nabla(\mathbf{A})\mathbf{B} + \nabla(\mathbf{B}^T)\mathbf{A}^T$. For expressions of the form $\mathbf{x}^T\mathbf{Ax}$ and $\mathbf{y}^t\mathbf{Ay}$ where $\mathbf{y} = function(\mathbf{x})$,

$$\begin{aligned} f &= \mathbf{x}^T\mathbf{Ax} \\ \nabla\mathbf{f} &= \nabla(\mathbf{x}^T)\mathbf{Ax} + \nabla(\mathbf{x}^T\mathbf{A}^T)(\mathbf{x}^T)^T \\ &= \mathbf{Ax} + \mathbf{A}^T\mathbf{x} \\ &= 2\mathbf{Ax} \qquad \text{(if } \mathbf{A} \text{ is symmetric)} \end{aligned} \tag{A.1}$$

$$\begin{aligned} f &= \mathbf{y}^t A\mathbf{y}, \qquad \mathbf{y} = function(\mathbf{x}) \\ \nabla\mathbf{f} &= \nabla(\mathbf{y}^T)\mathbf{Ay} + \nabla(\mathbf{y}^T\mathbf{A}^T)(\mathbf{y}^T)^T \\ &= \mathbf{J}^T\mathbf{Ay} + \mathbf{J}^T\mathbf{A}^T\mathbf{y} \qquad \text{where } \mathbf{J} = \nabla\mathbf{y} \\ &= 2\mathbf{J}^T\mathbf{Ay} \qquad \text{(if } \mathbf{A} \text{ is symmetric)} \end{aligned} \tag{A.2}$$

Equations A.1 and A.2 are applied to determine the first (Equation 2.2) and second (Equation 2.3) order derivatives of the objective function. In these expressions, $\mathbf{J} = \nabla\mathbf{g}$, $\mathbf{x_{prior}}$ and $\mathbf{d}_{obs}$ are constants. Further, $\mathbf{J}$ is known as the local sensitivity or the Jacobian while the second order derivative $\mathbf{H}$ is the Hessian.

$$\begin{aligned} \nabla\mathbf{f} &= \frac{1}{2}\{2(\mathbf{x} - \mathbf{x}_{prior})^T{\mathbf{C_M}}^{-1} + 2[\mathbf{g}(\mathbf{x}) - \mathbf{d}_{obs}]^T{\mathbf{C_D}}^{-1}\} \\ &= (\mathbf{x} - \mathbf{x}_{prior})^T{\mathbf{C_M}}^{-1} + \mathbf{J}^T{\mathbf{C_D}}^{-1}[\mathbf{g}(\mathbf{x}) - \mathbf{d}_{obs}] \end{aligned} \tag{A.3}$$

$$\begin{aligned} \mathbf{H} &= \nabla(\nabla\mathbf{f}) \\ &= \nabla((\mathbf{x} - \mathbf{x}_{prior})^T{\mathbf{C_M}}^{-1} + \mathbf{J}^T{\mathbf{C_D}}^{-1}[\mathbf{g}(\mathbf{x}) - \mathbf{d}_{obs}]) \\ &= {\mathbf{C_M}}^{-1} + \nabla(\mathbf{J}^T){\mathbf{C_D}}^{-1}[\mathbf{g}(\mathbf{x}) - \mathbf{d}_{obs}] + \nabla({\mathbf{C_D}}^{-1}[\mathbf{g}(\mathbf{x}) - \mathbf{d}_{obs}])^T{(\mathbf{J}^T)}^T \\ &= {\mathbf{C_M}}^{-1} + \mathbf{J'}^T{\mathbf{C_D}}^{-1}[\mathbf{g}(\mathbf{x}) - \mathbf{d}_{obs}] + \mathbf{J}^T{\mathbf{C_D}}^{-1}\mathbf{J}, \qquad \mathbf{J'} = \nabla\mathbf{J} \end{aligned} \tag{A.4}$$

Appendix B

Expanded results of the pseudo inverse tests

Table B.1: Objective function improvement
Weighted vs Base case

Improvement	10	20	30	40	50	60	70	80	90	100
Weighted- simple 0.01 model error, 0.99 data error										
Iter 0	200	0	0	0	0	0	0	0	0	0
Iter 1	168	21	9	2	0	0	0	0	0	0
Iter 2	168	21	9	2	0	0	0	0	0	0
Iter 20	168	21	9	2	0	0	0	0	0	0
Weighted- simple 0.2 model error, 0.8 data error										
Iter 0	200	0	0	0	0	0	0	0	0	0
Iter 1	32	105	52	11	0	0	0	0	0	0
Iter 2	9	31	72	59	19	10	0	0	0	0
Iter 8	9	20	31	40	45	24	17	8	4	2
Iter 20	9	20	31	40	45	24	16	8	2	5
Weighted- with number of data (0.002 model error and 0.25 data error)										
Iter 0	200	0	0	0	0	0	0	0	0	0
Iter 1	167	22	9	2	0	0	0	0	0	0
Iter 2	167	22	9	2	0	0	0	0	0	0
Iter 20	167	22	9	2	0	0	0	0	0	0
Base-case										
Iter 0	200	0	0	0	0	0	0	0	0	0
Iter 1	0	0	50	141	9	0	0	0	0	0
Iter 2	0	0	1	20	77	83	5	13	1	0
Iter 8	0	0	0	0	4	13	34	90	45	14
Iter 20	0	0	0	0	4	13	34	90	43	16

Table B.2: Objective function improvement
Smaller parameter range vs Increased trust size vs Base case

Improvement	10	20	30	40	50	60	70	80	90	100
Base-case										
Iter 0	200	0	0	0	0	0	0	0	0	0
Iter 1	0	0	50	141	9	0	0	0	0	0
Iter 2	0	0	1	20	77	83	5	13	1	0
Iter 8	0	0	0	0	4	13	34	90	45	14
Iter 20	0	0	0	0	4	13	34	90	43	16
Scaled parameters										
Iter 0	200	0	0	0	0	0	0	0	0	0
Iter 1	0	0	2	31	60	87	20	0	0	0
Iter 2	0	0	1	6	3	30	54	84	20	2
Iter 8	0	0	1	1	2	26	53	75	30	12
Iter 20	0	0	1	1	2	26	53	75	30	12
Increased trust region size										
Iter 0	200	0	0	0	0	0	0	0	0	0
Iter 1	0	0	0	6	9	15	79	74	17	0
Iter 2	0	0	0	0	2	10	52	93	39	4
Iter 8	0	0	0	0	1	7	31	97	52	12
Iter 20	0	0	0	0	1	7	31	96	52	13

Table B.3: Data error improvement
Increased trust region size vs Smaller parameter range

Improvement	10	20	30	40	50	60	70	80	90	100
Scaled parameters										
Iter 0	200	0	0	0	0	0	0	0	0	0
Iter 1	21	4	0	0	0	0	0	0	0	0
Iter 2	13	0	0	0	0	0	0	0	0	0
Iter 8	16	2	1	0	0	0	0	0	0	0
Iter 20	16	2	1	0	0	0	0	0	0	0
Increased trust region size										
Iter 0	200	0	0	0	0	0	0	0	0	0
Iter 1	82	39	28	18	7	1	2	0	0	0
Iter 2	73	43	24	17	9	3	0	0	0	0
Iter 8	55	49	22	29	11	3	5	0	1	0
Iter 20	55	48	22	29	11	4	5	0	0	1

Table B.4: Objective function improvement
Perturbed observations vs base case

Improvement	10	20	30	40	50	60	70	80	90	100
Perturbed observations										
Iter 0	200	0	0	0	0	0	0	0	0	0
Iter 1	0	0	51	141	8	0	0	0	0	0
Iter 2	0	0	0	20	77	81	7	13	2	0
Iter 8	0	0	0	0	4	13	39	76	56	12
Iter 20	13	0	0	0	4	13	39	75	53	16
Base-case										
Iter 0	200	0	0	0	0	0	0	0	0	0
Iter 1	0	0	50	141	9	0	0	0	0	0
Iter 2	0	0	1	20	77	83	5	13	1	0
Iter 8	0	0	0	0	4	13	34	90	45	14
Iter 20	0	0	0	0	4	13	34	90	43	16

Table B.5: Objective function improvement
Choice of prior

Improvement	10	20	30	40	50	60	70	80	90	100
Prior- mean of initial ensemble (base case)										
Iter 0	200	0	0	0	0	0	0	0	0	0
Iter 1	0	0	50	141	9	0	0	0	0	0
Iter 2	0	0	1	20	77	83	5	13	1	0
Iter 8	0	0	0	0	4	13	34	90	45	14
Iter 20	0	0	0	0	4	13	34	90	43	16
Prior- Ensemble #525										
Iter 0	200	0	0	0	0	0	0	0	0	0
Iter 1	0	0	38	162	0	0	0	0	0	0
Iter 2	0	0	0	0	21	7	160	12	0	0
Iter 8	0	0	0	0	0	0	0	8	99	93
Iter 20	0	0	0	0	0	0	0	8	98	94

Table B.6: Objective function improvement
Increasing observation times compared

Improvement	10	20	30	40	50	60	70	80	90	100
Base-case										
Iter 0	200	0	0	0	0	0	0	0	0	0
Iter 1	0	0	50	141	9	0	0	0	0	0
Iter 2	0	0	1	20	77	83	5	13	1	0
Iter 8	0	0	0	0	4	13	34	90	45	14
Iter 20	0	0	0	0	4	13	34	90	43	16
4-observation times										
Iter 0	200	0	0	0	0	0	0	0	0	0
Iter 1	0	15	135	48	2	0	0	0	0	0
Iter 2	0	0	16	81	71	23	1	8	0	0
Iter 8	0	0	0	11	10	39	46	52	37	5
Iter 20	0	0	0	11	10	38	40	47	41	13
14-observation times										
Iter 0	200	0	0	0	0	0	0	0	0	0
Iter 1	36	130	27	7	0	0	0	0	0	0
Iter 2	4	47	96	34	13	5	0	1	0	0
Iter 8	4	18	33	15	5	21	34	26	33	11
Iter 20	4	18	33	15	4	3	1	1	19	102

Table B.7: Data error improvement
Increasing observation times compared

Improvement	10	20	30	40	50	60	70	80	90	100
Base-case										
Iter 0	200	0	0	0	0	0	0	0	0	0
Iter 1	148	18	2	0	0	0	0	0	0	0
Iter 2	125	22	17	3	1	0	0	0	0	0
Iter 8	44	49	22	35	8	3	2	2	1	0
Iter 20	46	49	22	30	10	3	2	1	2	3
4-observation times										
Iter 0	200	0	0	0	0	0	0	0	0	0
Iter 1	159	36	2	0	0	0	0	0	0	0
Iter 2	99	82	13	3	0	0	0	0	0	0
Iter 8	25	29	34	50	31	22	4	2	0	0
Iter 20	25	29	31	37	29	20	16	7	3	0
14-observation times										
Iter 0	200	0	0	0	0	0	0	0	0	0
Iter 1	149	50	0	0	0	0	0	0	0	0
Iter 2	59	112	24	4	0	0	0	0	0	0
Iter 8	27	34	12	7	17	31	19	26	24	2
Iter 20	27	34	12	2	2	3	0	6	29	84

Table B.8: Model error improvement
Increasing observation times compared

Improvement	10	20	30	40	50	60	70	80	90	100
Base-case										
Iter 0	200	0	0	0	0	0	0	0	0	0
Iter 1	0	0	0	92	108	0	0	0	0	0
Iter 2	0	0	0	0	3	69	113	6	9	0
Iter 8	0	0	0	0	0	0	2	10	74	114
Iter 20	0	0	0	0	0	0	2	10	73	115
4-observation times										
Iter 0	200	0	0	0	0	0	0	0	0	0
Iter 1	0	0	1	160	39	0	0	0	0	0
Iter 2	0	0	0	0	12	142	37	6	3	0
Iter 8	0	0	0	0	0	2	9	17	95	77
Iter 20	0	0	0	0	0	2	9	17	83	89
14-observation times										
Iter 0	200	0	0	0	0	0	0	0	0	0
Iter 1	0	17	146	34	3	0	0	0	0	0
Iter 2	0	1	2	55	107	29	5	1	0	0
Iter 8	0	1	2	13	18	23	17	37	60	29
Iter 20	0	1	2	13	18	22	16	5	11	112

Appendix C

Flux-pressure profiles-PCA tests

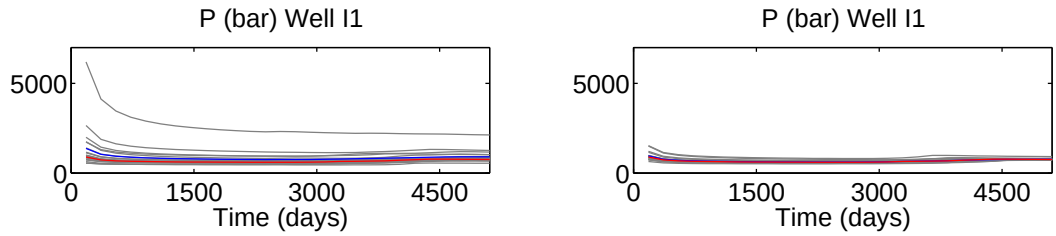

Figure C.1: Pressure profiles before and after history matching (30 DGN iterations) in the injection well. The figures on the left are the prior while the right correspond to the posterior.

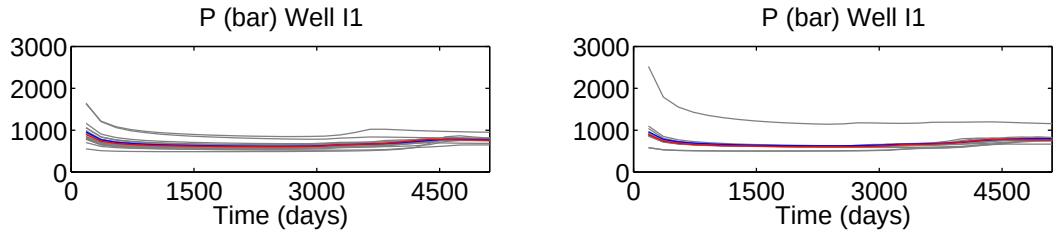

Figure C.2: Pressure profiles after history matching (15 DGN iterations) in the injector. The left insets are obtained when using a larger initial trust region while the right correspond to increased (28) observation times.

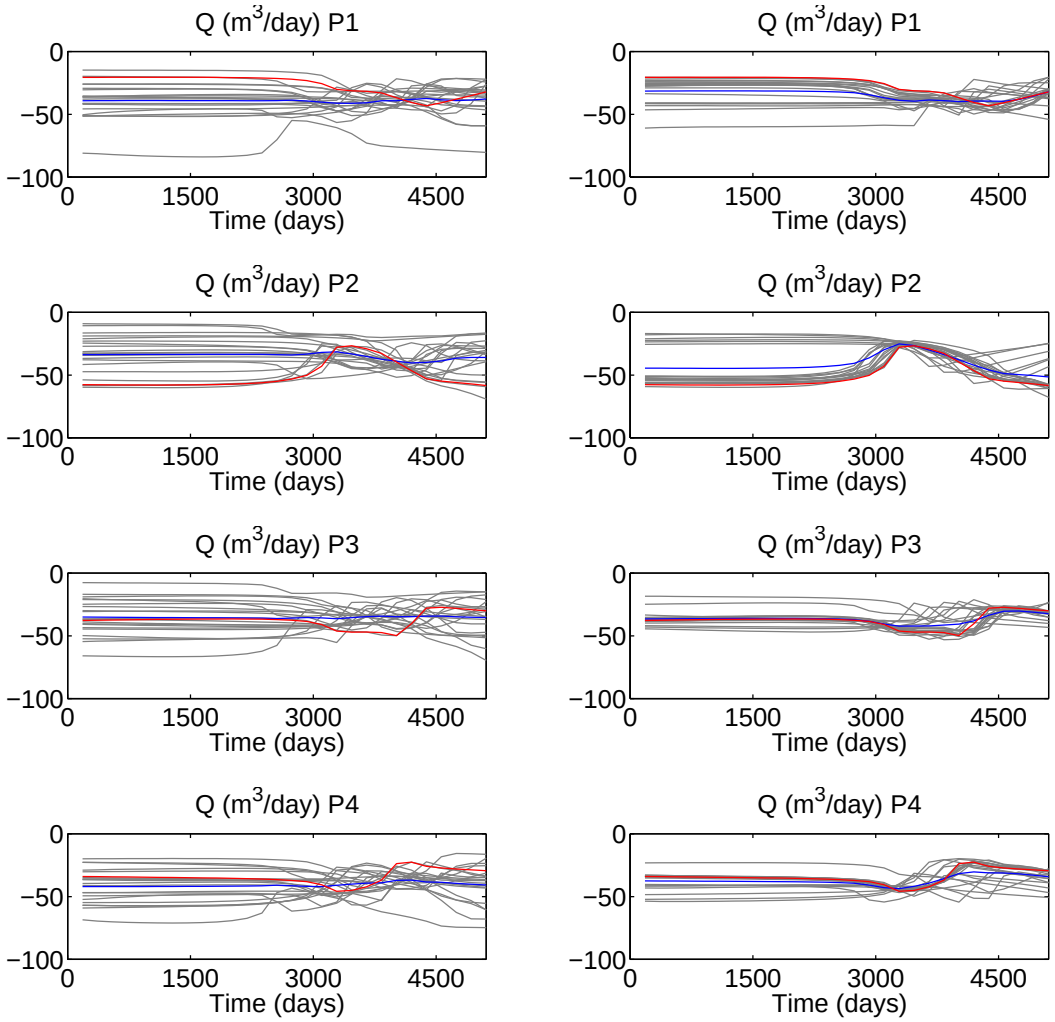

Figure C.3: Flux profiles before and after history matching (30 DGN iterations) in the producing wells. The figures on the left are the prior while the right correspond to the posterior.

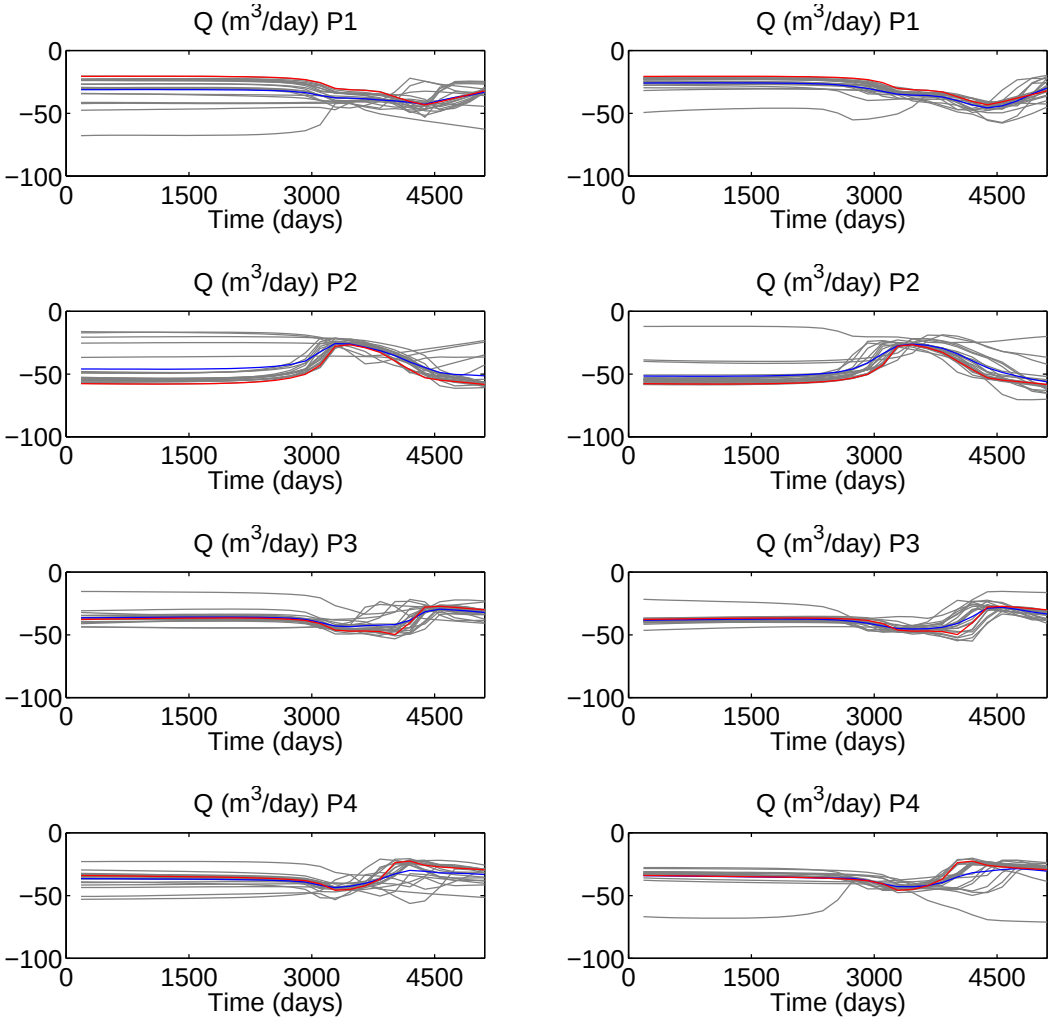

Figure C.4: Flux profiles after history matching (15 DGN iterations) in the producing wells. The figures on the left are obtained when using a larger initial trust region while the right correspond to increased (28) observation times.

Appendix D

Steihaug-Toint algorithm for trust region solving

The implementation of this algorithm is tested on the 2-D Rosenbrock function. This is a popular test for optimization algorithms because of the valley within which the global optimum lies. For the two variable case, the global minimum lies at (1,1).

The contour of the function within the valley (Figure D.1) causes the algorithm to take a number of iterations before finding the global minimum. The starting point is taken at (0,0) with an initial trust region size of 2. Table D.1 shows the movement of the initial point towards the minimum. The algorithm took a total of 21 iterations to converge. In this case, convergence was defined to be the determination of a local minimum. When this occurs, the update ratio, ρ becomes infinite since no new update can be found that improves the objective. As a comparison, the dog leg implementation of Matlab is also used with the same settings. Clearly, the dog leg algorithm fails to determine the global minima. However, if the trust region size updating is done within Matlab, the global minimum is found after 16 iterations. It should also be noted that the toy model results shown earlier were done by pairing the trust region update scheme (introduced in Section 3.4) with Matlab's dog leg solver. This could mean that the trust region size updating scheme used by Matlab is more robust for its dog-leg solver compared to the one used here.

While the new trust region solver is an improvement over the dog-leg algorithm, the issue of stagnating base cases continued to occur when large number of PCA coefficients are used. A stagnating base case is the result of an inability to updates in the immediate neighborhood that minimize the quadratic model.

From a diagnostic perspective, this occurs as non-improvement in objective function and an update ratio that is a small positive number (< 0.01) or negative. This results in a shrinking trust region size. If the update ratio becomes larger, then the model approximation improves. More often than not, this is not the case. One possible reason for this is that the model is present in a valley of the objective function.

A similar trend in update ratio was seen when the DGN was used to determine the global minimum [1] of the Rosenbrock function. The base cases move from their initial locations into

[1] lies in a narrow valley

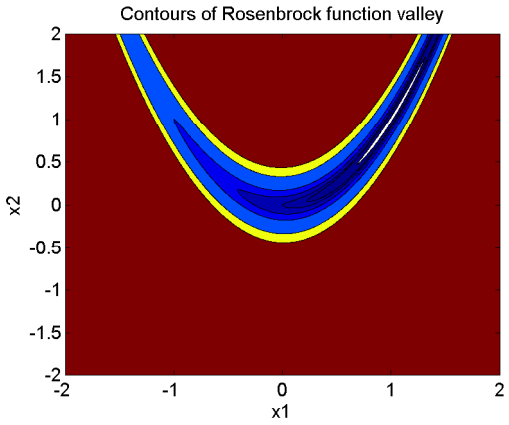

Figure D.1: Contour plot of the valley of the 2D Rosenbrock function

Table D.1: Determining global minimum of the Rosenbrock function

Iteration	Steihaug algorithm			Dog-leg algorithm		
	x_1	x_2	Function value	x_1	x_2	Function value
1	0.0000	0.0000	1.0000	0.0000	0.0000	1.0000
2	0.0000	0.0000	1.0000	0.0000	0.0000	1.0000
3	0.0000	0.0000	1.0000	0.0000	0.0000	1.0000
4	0.0625	0.0000	0.8804	0.0884	0.0000	0.8371
5	0.1803	0.0418	0.6804	0.0884	0.0000	0.8371
6	0.1803	0.0418	0.6804	0.2181	0.0128	0.7319
7	0.2465	0.0489	0.5819	0.2181	0.0128	0.7319
8	0.3532	0.1140	0.4300	0.2181	0.0128	0.7319
9	0.5522	0.2653	0.3574	0.2181	0.0128	0.7319
10	0.6024	0.3604	0.1587	0.2181	0.0128	0.7319
11	0.6024	0.3604	0.1587	0.2181	0.0128	0.7319
12	0.6800	0.4583	0.1041	0.2181	0.0128	0.7319
13	0.8262	0.6611	0.0766	0.2181	0.0128	0.7319
14	0.8590	0.7367	0.0200	0.2181	0.0128	0.7319
15	0.8590	0.7367	0.0200	0.2181	0.0128	0.7319
16	0.9214	0.8450	0.0078	0.2181	0.0128	0.7319
17	0.9648	0.9290	0.0016	0.2181	0.0128	0.7319
18	0.9904	0.9802	0.0001	0.2181	0.0128	0.7319
19	0.9989	0.9977	0.0000	0.2181	0.0128	0.7319
20	1.0000	1.0000	0.0000	0.2181	0.0128	0.7319
21	1.0000	1.0000	0.0000	0.2181	0.0128	0.7319

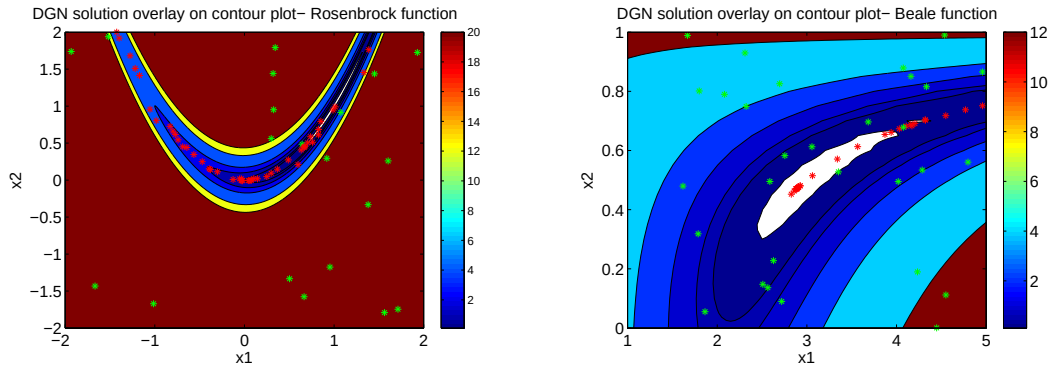

Figure D.2: Countours plots overlain with ensembles. The points in green represent the initial ensemble while those in red are the updated ensemble taken from the DGN after 50 iterations. The contour on the left corresponds to the Rosenbrock function while the one to the right is Beale's function

the valley after which there is little to no further improvement. This causes the update ratio to shrink, becoming smaller till no update is found. When this happens, the update ratio takes a $\frac{0}{0}$ form making it undefined. This has also been observed when the DGN was tested with Beale's function and the eggholder function. Figure D.2 shows the ensemble overlapped on the function curves. Once the ensemble enters the valley, there is little improvement in objective and the points remain within.

Finally, to verify that the update ratio does become undefined in the synthetic reservoir model, a test on the 1-D reservoir model was simulated to over 50 iterations. Over 36 base cases had an undefined update ratio at 50 iterations with the number increasing to 45 at 90 iterations. When the objective function value for the ensemble is compared with the limit $1 + 5\sqrt{\frac{2}{N_d}}$ [Oliver et al., 2008], 55 ensemble members have a lower value. Since the ensemble size is 100 (=45+55), this hypothesis, that models are trapped in regions of local minima could be true.

Appendix E

Erratum

1. Grammatical improvements and minor changes to the text to improve reading experience.
2. Fixed inconsistencies in notation, i.e., Hessian referred to by B and H, partial boldfacing of some vectors and matrices
3. Added references to the ES-MDA theory
4. Reformatting of the iterative form of the ES-MDA and pseudo code for the Steihaug-Toint algorithm
5. Added missing test result for model errors when initial ensemble is used as prior
6. Formatting improvement for Appendix B- C
7. Renamed appendix C to reflect its content.