THE UNIVERSITY OF TULSA
THE GRADUATE SCHOOL

AUTOMATIC HISTORY MATCHING OF PRODUCTION DATA FOR LARGE SCALE PROBLEMS

by
Fengjun Zhang

A dissertation submitted in partial fulfillment of
the requirements for the degree of Doctor of Philosophy
in the Discipline of Petroleum Engineering
The Graduate School
The University of Tulsa

2002

THE UNIVERSITY OF TULSA

THE GRADUATE SCHOOL

AUTOMATIC HISTORY MATCHING OF PRODUCTION DATA FOR LARGE SCALE PROBLEMS

by

Fengjun Zhang

A DISSERTATION

APPROVED FOR THE DISCIPLINE OF

PETROLEUM ENGINEERING

By Dissertation Committee

_____ Chairperson

_____

_____

_____

ABSTRACT

Fengjun Zhang (Doctor of Philosophy in Petroleum Engineering)
AUTOMATIC HISTORY MATCHING OF PRODUCTION DATA FOR LARGE SCALE PROBLEMS
(243 pp.-Chapter VII)
Directed by Dr. Albert C. Reynolds

(319 words)

Within the context of Bayesian statistics, realizations of rock property fields can be generated by automatic history matching of production data using a prior model to provide regularization. Automatic history matching requires the minimization of an objective function which includes the sum of squared production data mismatch as well as a regularization term arising from the prior geostatistical model. For large scale problems, the computational efficiency and robustness of the optimization algorithms used for minimization are of paramount importance.

We consider a variety of optimization algorithms for history matching production data. For history matching problems where tens of thousands of parameters are estimated, preconditioned conjugate gradient methods and quasi-Newton methods appear to be the only viable gradient based methods. Based on several examples considered in this work, a particular implementation of the limited memory Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm is more robust and computationally efficient for large scale problems than the preconditioned conjugate gradient methods that we have tried. It is shown that computational efficiency of the limited memory BFGS can be improved by a proper choice of the scaling factor and the

initial approximation of the inverse Hessian. To the best of our knowledge, the particular implementations of these algorithms presented here are new to the petroleum engineering literature.

An iterative linear solver based on orthomin theory was implemented in this work. For large problems, the iterative solver is orders of magnitude faster than the direct solver which is based on the LU decomposition. The iterative solver was used to solve the adjoint equation system which is a linear system. The solution obtained by the iterative solver is in excellent agreement with the solution obtained by a sparse matrix technique.

The computational algorithms for history matching are applied to condition rock property fields generated from a prior geostatistical model to production data. The procedure allows one to consider the errors in prior means as model parameters.

ACKNOWLEDGMENTS

I wish to express my sincere appreciation to Dr. Albert C. Reynolds, Jr., Professor of Petroleum Engineering and Mathematical Science of the University of Tulsa, and Dr. Dean S. Oliver, Professor of Petroleum Engineering of the University of Tulsa, for their support, guidance, insights, and encouragement during my 4 years' Ph.D. study in the University of Tulsa. I specially thank Dr. Richard A. Redner, Professor of Mathematical Science of the University of Tulsa and Dr. Brenton McLaury, Associate Professor of Mechanical Engineering of the University of Tulsa for serving on my committee and for their helpful suggestions and comments.

I would like to express my thanks and appreciations to Dr. Zhuoxin Bi, one of my best friends, who encouraged me to come to TU and started my study in petroleum engineering. I appreciate all the help in every corner of my life and study he provided generously to me. My sincere appreciations should also be given to Dr. Ruijian Li, one of my closest friends, for his generous help and suggestions. He wrote the initial version of the three-dimensional three-phase automatic history matching code. These programs have greatly contributed to the completion of this study. I appreciate Mrs. Judy Teal and Mrs. Loreta M. Watkins for helping me with many things during my study at TU. I would like to extend my thanks to all the other faculty members and graduate students of the Petroleum Engineering department at TU, and all my friends who helped and encouraged me in the past 4 years.

I gratefully acknowledge financial support from the member companies of TUPREP (the University of Tulsa Petroleum Reservoir Exploitation Projects), Schlumberger Foundation, Inc., the Graduate School and Petroleum Engineering department of the University of Tulsa. The reservoir simulator used in this study was provided by Chevron Petroleum Technology Company.

I am thankful to my wife Youfang Liu and my lovely daughter Chenlu (Sarah) Zhang for their unconditional love, support and patience over the last few years. We share the same dreams and we have gone through good and bad times together.

This work is dedicated to my parents, Bin Zhang and Xiufen Yuan, and all my family members in China for their unconditional love.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Automatic history matching can be used to generate reservoir descriptions that are consistent with both static data and dynamic data. In our application of history matching, we minimize an objective function which consists of a data mismatch part and a model mismatch part. The data mismatch part measures the distance between predicted production data from forward modeling and observed production data. The model mismatch part provides regularization and measures the distance from a prior mean or unconditional realization generated from the prior model. With this approach to automatic history matching, we can construct estimates or realizations of reservoir parameters, e.g., absolute permeability fields and porosity fields. Prior means for model parameters can also be considered as parameters in the history matching procedure. Adjusting prior means is similar to estimating permeability or porosity multipliers. The parameters to be estimated are referred to as the model parameters, or the model.

To minimize the objective function involved in the history matching procedure, we need to apply the optimization algorithms. Basically, there are two categories of minimization algorithms for unconstrained optimization problems. One category consists of gradient based algorithms, e.g., steepest descent, Newton, Gauss-Newton, Levenberg-Marquardt, conjugate gradient and variable metric (or quasi-Newton), and the other category includes non-gradient based algorithms, such as simulated annealing (see, for example, Ouenes et al. (1993) and Vasco et al. (1996)), genetic algorithm (see, for example, Sen et al. (1992)), Monte Carlo methods (see, for example, Hegstad et al. (1994) and Bonet-Cunha et al. (1998)) and neutral networks (see, for example, Ouenes et al. (1994)). Unless one can predict production data for

a given reservoir model by some method which is orders of magnitude faster than running a conventional finite-difference simulator, non-gradient based algorithms are far too slow for practical application (see, for example, Bonet-Cunha et al. (1998)) as they may require hundreds of thousands of iterations for convergence, and one must recalculate the predicted production data at each iteration. In our work, predicted data are always generated by running a reservoir simulator, and only gradient based algorithms are considered.

With this approach, the efficiency of automatic history matching rests on the parameterization of the model and the efficiency of the optimization algorithm. The simplest reparameterization technique is known as the method of zonation, in which the reservoir is divided into a relatively small number of zones over which the parameter is assumed to have uniform values. To the best our knowledge, Jacquard and Jain (1965) and Jahns (1966) used the zonation approach first for history matching purposes. A decade later, Gavalas et al. (1976) and Shah et al. (1978) showed that a Baysian history matching approach gave better estimates of the true permeability and porosity fields than were obtained by zonation in a simulated case of a one-dimensional reservoir. In a recent study, Bissell et al. (1994) proposed a type of zonation in which gridblocks are grouped together into sets that they call gradzones. They provided a procedure to select gradzones based on the high sensitivity of various data with respect to parameters in each of the selected gridblocks. Although it is simple to apply zonation for reservoir inverse problems, it is difficult to obtain a good data match because of the small number of degrees of freedom. More importantly, any coarse zonation method typically yields discontinuous reservoir properties at zonation boundaries.

The pilot point method of parameterization which perturbs reservoir properties only at selected pilot point locations to match the production data was originally proposed by de Marsily et al. (1984) in the groundwater hydrology field. This is a reduced parameterization whose basis vectors are simply the columns of the prior covariance matrix corresponding to the pilot point locations. The pilot point method has been applied to synthetic and field cases by several researchers (for example, Ki-

tanidis (1995), RamaRao et al. (1995) and Gómez-Hernández et al. (1997)) in the groundwater hydrology field. In recent years, some researchers (for example, Xue and Datta-Gupta (1997), Wen et al. (1997), Bissell et al. (1997) and Roggero (1997)) have adapted the same idea to history matching. The drawback of this method is that, it can result in overshoot at the pilot point locations (see, for example, Xue and Datta-Gupta (1997)).

The subspace method (for example, Reynolds et al. (1996), Abacioglu et al. (2000)) is another reparameterization method which can be used for history matching. This method requires applying the adjoint method to compute the subspace vectors and then applying the gradient simulator method to calculate the gradient of the objective function with respect to the subspace vectors. Therefore, even though this method avoids the direct calculation of the sensitivity coefficient G, it is impractical for large scale history matching problems.

Computational efficiency of the optimization process depends on the number of iterations required to converge and the computational cost per iteration. In general, the cost per iteration depends largely on the cost of computing the required sensitivity coefficients, i.e., the derivatives of predicted production data with respect to reservoir model parameters. If the number of model parameters is very small, the finite-difference method can be used to estimate sensitivity coefficients. This process requires $N_m + 1$ simulation runs where N_m is the number of model parameters. If the number of model parameters exceeds a few dozen, it is clear that the finite-difference method will be impractical.

If the number of model parameters is large, the gradient simulator method (Yeh (1986) and Anterion et al. (1989)), which is a popular approach for computing sensitivity coefficients, also becomes impractical. With the gradient simulator method, the sensitivities of all gridblock pressures and gridblock saturations to a particular model parameter are computed at the end of a simulation time step. From these sensitivities, we can easily calculate needed sensitivities, e.g., the sensitivities of water-oil ratio to model parameters. To generate the sensitivities, we must solve a matrix problem which involves the same coefficient matrix as the one that ap-

pears in the finite-difference simulator equations. Only the right-hand side of the matrix problem depends on the particular sensitivities being calculated. Thus, the problem reduces to solving a matrix problem with multiple right-hand side vectors, one right-hand side vector for each model parameter. With the fast iterative solver developed by Killough et al. (1995), it appears that the computational time to compute a single sensitivity coefficient is on the order of 10% of a forward simulation. For the gradient simulator to be practical, the number of model parameters must be small. This means, if the underlying reservoir simulation problem involves thousands of model parameters, e.g., all gridblock porosities and permeabilities, one must reduce the number of parameters estimated directly in the optimization algorithm by some form of reparameterization, e.g., zonation (Jacquard and Jain (1965) or gradzones (Bissell et al. (1994); Bissell (1994), Tan (1995)), pilot points (de Marsily et al. (1984), RamaRao et al. (1995), Bissell et al. (1997)) or subspace methods (Kennett and Williamson (1988), Oldenburg et al. (1993), Reynolds et al. (1996), Abacioglu et al. (2000)). However, these methods have those disadvantages mentioned earlier. Therefore, our research focused on more computationally efficient gradient based optimization algorithms.

The MGPST, which was introduced by Chu et al. (1995) based on ideas of Tang et al. (1989), provides a highly efficient approximation of the gradient simulator method. Unfortunately, the MGPST methods yields highly inaccurate estimates of the sensitivity of pressure data to the porosity field.

The other main alternative for the computation of sensitivity coefficients is the adjoint method (Chavent et al. (1975) and Chen et al. (1974)). For linear problems, the adjoint method is equivalent to a procedure developed for two-dimensional linear single-phase problems by Carter et al. (1974) and extended in an approximate, but highly efficient way, to three-dimensional problems by He et al. (1997). The adjoint method was applied to a water-oil two-phase problem by Wu (1999) and to three-dimensional three-phase problems by Makhlouf et al. (1993) and Li (2001). Using this method to get the sensitivity of one datum with respect to all the model parameters, we need to solve a system of adjoint equations. The system of equation

describing the adjoint problem is similar to the system of finite-difference equations solved in the reservoir simulator, but there are some major differences. First, the adjoint problem is solved backward in time and requires information from the forward solution (reservoir simulation run) to form the matrices involved in the adjoint problem. In our approach, we save pressure and saturation from the forward run so that we can compute the matrices involved in the adjoint problem. Secondly, the system of finite-difference equations for the forward problem is nonlinear and is solved by Newton-Raphson iteration. The adjoint system is linear, consequently, the solution of the adjoint problem over one time step, may require on the order of one half the time or less required to solve the forward problem over one time step, but we have not done any exact timing of the time required. For the purpose of making comparison of the computational effort required by different algorithms, we equate the cost of solving an adjoint system for one time step to the cost of solving the simulator finite-difference equations over one time step. In the adjoint procedure, the number of right-hand sides is no greater than the number of production data and independent of the number of model parameters. If the data are evenly distributed in time, the cost of computing the sensitivities of all data to all model parameters is roughly equal to $N_d/2$ simulation runs backward in time without using the multiple right-hand side technique where N_d is the number of data. If the number of data is large, this method is computationally expensive and is impractical for real problems.

Due to the fact that it exhibits quadratic convergence, Newton's method is a popular method for unconstrained minimization. As the second derivative terms needed for Newton's method are difficult to evaluate, Newton's method is often replaced by the Gauss-Newton method. Near a minimum, the Gauss-Newton method is approximately quadratically convergent and has the advantage that the associated Hessian matrix for Bayesian inverse problems is guaranteed to be positive definite if the prior covariance matrix is positive definite. For a non-quadratic objective function, Newton's method may not provide the optimal downhill direction if the initial guess is far from a model that corresponds to a minimum of the objective function. Moreover, for history matching problems, the Gauss-Newton method occasionally

fails to converge to a model which gives a good match of production data if the initial data mismatch is large; see Wu et al. (1999) and Li et al. (2001). For these reasons, the Levenberg-Marquardt algorithm is often preferable to the Gauss-Newton method.

If the sensitivities are easily obtained, the Gauss-Newton (GN) and Levenberg-Marquardt (LM) methods are good choices for the unconstrained minimization problem, because they have the quadratic convergence property which results from using the curvature information of the objective function. This curvature information is represented by the second order derivatives of the objective function, i.e., the Hessian matrix. Constructing the Hessian matrix requires the computation of sensitivity coefficients. As computation of all sensitivity coefficients is impractical if both the number of data and the number of model parameters are large, we will consider other alternatives for conditioning a model to multi-phase flow production data. The alternatives include those methods that do not require the calculation of all sensitivity coefficients. They require only the gradient of the objective function, which can be calculated from a single adjoint solution. Thus, even if these algorithms require significantly more iterations to converge, they may still require a small fraction of the computer time required to obtain convergence with either the Levenberg-Marquardt algorithm or the Gauss-Newton algorithm.

There are two main classes of effective optimization algorithms which require only the gradient of the objective function. The first class includes the set of conjugate gradient algorithms and the second class includes the set of quasi-Newton or variable metric methods. Each algorithm includes a directional line search, i.e., the solution of a one-dimensional minimization problem, at each iteration. However, the search direction differs from algorithm to algorithm.

Quasi-Newton methods, which are based on generating an approximation to the inverse of the Hessian matrix, require only the gradient of the objective function. The methods differ in how they correct or update the inverse Hessian approximation at each iteration. The rank one correction formula was first suggested by Broyden (1967). Another formula, called the DFP algorithm, was first suggested by Davidon

(1959) and later presented by Fletcher and Powell (1963). The BFGS correction formula (which was suggested by Broyden (1970), Fletcher (1970), Goldfarb (1970) and Shanno (1970) independently), and several variants of the BFGS formula (like the self-scaling variable metric (SSVM) by Oren and Luenberger (1974), limited memory BFGS by Nocedal (1980) and Liu and Nocedal (1989)) have proven useful in many scientific applications.

DFP, BFGS and all the variants of BFGS are members of Broyden's family which is a subset of Huang's family; see Huang (1970) or Appendix B. Dixon's theorem (see Dixon (1972)) shows that for a general continuously differentiable objective function, the set of search directions developed by any two members of Broyden's family differ only by a scalar multiplier and successive iterates are identical, given that the same starting point and the same initial Hessian inverse approximation, $\tilde{H}_0^{-1}$, are used and that all line searches are exact. The importance of an exact line search is underlined by Dixon's theorem, but in practice exact line searches are expensive and lead to less computationally efficient algorithms. Due to an inexact line search and round off errors, different members of the Broyden class usually give in practice different results for the same problem starting with the same initial condition. In practice, BFGS is the best update of the Broyden class (see, e.g., Shanno and Phua (1978), Nocedal (1992) and Kolda et al. (1998)), in the sense that this algorithm exhibits more robust behavior than others in this class.

Another minimization algorithm which only uses the gradient of the objective function is the conjugate gradient method. The conjugate gradient method was originally proposed by Hestenes and Stiefel (1952) for solving linear systems and extended to nonlinear optimization by Fletcher and Reeves (1964) to obtain the Fletcher-Reeves algorithm. Later Polak (1971) proposed a different formula to calculate the coefficient involved in the search direction update equation and the corresponding algorithm is called Polak-Ribière algorithm. Powell (1977) presented some numerical results and gave some theoretical reasons which indicate that the Polak-Ribière algorithm is superior to the Fletcher-Reeves algorithm. Efficiency of the conjugate gradient method depends primarily on the preconditioner used.

When applied to quadratic functions, the conjugate gradient method is equivalent to the standard BFGS algorithm provided that an exact line search is performed; see Buckley (1978b) or Nazareth (1979). However, the BFGS algorithm is more efficient than the conjugate gradient method when applied to nonlinear problems due to the fact that BFGS uses approximate curvature information of the objective function, e.g., see Buckley (1978a) and Kolda et al. (1998). The disadvantage of the BFGS algorithm over the conjugate gradient method is that the BFGS requires much more storage than the standard conjugate gradient method. Here, standard conjugate gradient algorithm refers to the conjugate gradient algorithm without preconditioning. The standard conjugate gradient algorithm requires less memory than the Gauss-Newton, Levenberg-Marquardt and quasi-Newton methods. The limited memory BFGS algorithm requires an intermediate amount of memory which is specified by users and also uses the approximate information of the Hessian. Kolda et al. (1998) shows that limited memory BFGS has the property of termination in finite number of iterations (quadratic termination) when applied to strictly convex quadratic functions. The examples shown in this study also indicate that limited memory is more efficient than the conjugate gradient method.

The limited memory BFGS was designed for the purpose of solving large scale problems which involve thousands of variables. Limited memory BFGS methods originated with the work of Shanno (1978a), and were subsequently developed and analyzed by Buckley (1978a), Nazareth (1979), Nocedal (1980), Shanno (1978b), and Buckley and Lenir (1983). Liu and Nocedal (1989), and Nash and Nocedal (1991) tested LBFGS method on a set of problems. They concluded that LBFGS performs better than conjugate gradient in terms of computational efficiency, except in cases where the function evaluation is inexpensive. Nash and Nocedal also tested the truncated-Newton algorithm in their work. From their comparison, none of the algorithms is clearly superior to the other.

The SSVM (self-scaling variable metric) method was used by Yang and Watson (1988) on hypothetical water floods of both 1D and 2D reservoir models. The 1D reservoir model consisted of 10 gridblocks with an injection well at one end

and a producing well at the other end. Sixty data from each well were used for history matching. Four cases based on this 1D reservoir model were tested. The reservoir was characterized by different parameters in different cases. The number of model parameters varied from 9 to 19. The other two cases were based on a quarter of a five-spot 2D model which consisted of a 10×10 grid. Again sixty data from each well were history matched. The number of model parameters for these two cases were 4 and 11 respectively. In this paper, the authors tested four different algorithms, BFGS, SSVM, conjugate gradient and steepest descent. They concluded that (i) the self-scaling variable metric method is significantly more efficient than the BFGS method; (ii) the SSVM and BFGS methods are more efficient and robust than the conjugate gradient method, except in the case where the objective function is nearly quadratic; and (iii) both SSVM and BFGS methods perform significantly better than the steepest descent method.

Masumoto (2000) applied the SSVM method to a water-oil two-phase flow problem. The author considered a one-dimensional reservoir model with 20 gridblocks. With a fixed porosity field, the author estimated the permeabilities in all gridblocks. Hence, there were 20 unknown model parameters. The objective function he minimized included a pressure mismatch part and the pressure derivative mismatch part. The author did not give any information about how many data he history matched or any assessment of the minimization algorithm. Savioli and Grattoni (1992) tested 4 different algorithms: Davidon-Fletcher-Powell (DFP), Fletcher-Reeves (FR), BFGS and Levenberg-Marquardt (LM). The authors presented two examples. In the first example, they estimated one permeability value and one porosity value by applying these four algorithms. The second example they considered was an oil-water two-phase water flooding problem. They estimated the exponent used to define the relative permeability and capillary pressure curves with a power law function (only one adjustable parameter for each curve). They concluded that among these four algorithms, BFGS performed best in terms of computational efficiency and stability.

Makhlouf et al. (1993) applied the conjugate gradient method to a three-

phase three-dimensional history matching problem. The authors considered a three layer reservoir. A 15×10 grid was used to simulate this problem with only one vertical gridblock per layer. They assumed the porosity and relative permeabilities are known. Absolute permeabilities were the only model parameters estimated (450 model parameters). For the three-phase examples presented, wellbore pressure, water cut, gas-oil ratio as well as phase flow rate at the individual penetrated layers (about two thousand production data) were history matched. The conjugate gradient algorithm presented by Nazareth (1977) was applied to minimize the objective function. The authors presented two examples for the three-phase problem. In the example where free gas was present initially, 222 iterations were required to converge. In the example where no free gas was present initially, 110 iterations were required to converge. The authors did not discuss a preconditioner. If a good preconditioning matrix can be found for nonlinear conjugate gradient methods, it is conceivable that convergence could be considerably accelerated. For the Baysian formulation of the history matching problem considered here, the most straightforward choice of a preconditioning matrix is the inverse of the prior covariance matrix (Kalita and Reynolds (2000)). Although this preconditioner yields some improvement in the examples we have tried, it tends to result in much slower convergence rates than are obtained with a good implementation of the limited memory BFGS method. In this work, we also explore using approximations to the inverse Hessian matrix constructed from a variable metric method as a preconditioning matrix for the nonlinear conjugate gradient method. This method is also shown to be less robust than the limited memory BFGS method.

Deschamps et al. (1998) presented an interesting comparison of the relative efficiency of several optimization methods for automatic history matching of production data. They suggest that the most efficient optimization method will be a hybrid scheme and specifically advocate schemes that combine a Gauss-Newton method with another procedure. They reject the Levenberg-Marquardt scheme as relying too heavily on the steepest descent method and do not present any comparisons based on this method. On the other hand, some results (Li et al. (2001)

and Zhang et al. (2001)) indicate that a Levenberg-Marquardt algorithm is often superior to the Gauss-Newton method. The Levenberg-Marquardt algorithm used in these references, however, is the nonstandard one introduced by Bi et al. (2000) and is based on a regularized objective function. In this procedure, one starts with a large value of the Levenberg-Marquardt parameter λ to avoid making large changes in the model at early iterations, whereas, the standard algorithm starts with a small value of λ. Deschamps et al. history matched only a few model parameters and apparently used no regularization term in their objective function. Instead of the standard Levenberg-Marquardt algorithm, they implemented a trust region method. In the event that neither the Gauss-Newton step nor steepest descent give an acceptable step, then a dogleg strategy is used to interpolate between Gauss-Newton and steepest descent to obtain a search direction.

Deschamps et al. also considered a scheme, due to Law and Fariss (1972), which is referred to as ConReg. This method effectively requires a spectral or singular value decomposition of the Gauss-Newton Hessian matrix. (This is not feasible, when the dimension of these parameters is large.) Based on this decomposition, they calculated parameters corresponding to "small" eigenvalues by a steepest descent method and those corresponding to sufficiently large eigenvalues using a Gauss-Newton search direction. Another algorithm is based on starting the optimization with a quasi-Newton method and then switching to a ConReg when the objective function has been reduced below a preset limit. As the quasi-Newton method requires only the gradient of the objective function, one iteration of a quasi-Newton method requires much less computational effort than one iteration of the Gauss-Newton method. They also consider three other hybrid algorithms which combine Gauss-Newton and quasi-Newton methods.

Deschamps et al. compared the hybrid methods with a pure quasi-Newton method, although it is not clear which quasi-Newton method they used, how they initialized the inverse Hessian approximation, or whether they used scaling. For the two history matching problems they considered, the quasi-Newton method required on the order of three times as many "equivalent simulation runs" to obtain conver-

gence as most of the other methods. For the first example presented, a synthetic case, the quasi-Newton method converged to a much higher value of the objective function than all the other methods.

It is important to note that fewer than 20 parameters were estimated by history matching in the two examples considered by Deschamps et al. For these problems, it is feasible to compute all sensitivity coefficients necessary to form the Hessian matrix for the Gauss-Newton method. In the examples we considered in this study, we estimated a few hundred to tens of thousands of model parameters which include both horizontal and vertical permeabilities in each individual gridblock by history matching up to a thousand production data. As a matter of fact, our code has the capacity of estimating horizonal permeabilities, vertical permeabilities, porosities and well skin factors. For such problems, direct calculation of all sensitivity coefficients is not feasible.

There are 7 chapters and 4 appendices in this dissertation. Chapter 2 briefly describes the theory of automatic history matching. It includes discussions of Bayesian inversion, the construction of the MAP estimate and multiple realizations. A procedure of handling the doubly stochastic model is also presented in this chapter. Chapter 3 discusses the forward simulator and the adjoint method used to calculate the gradient of the objective function. Chapter 4 presents the iterative solver we used to solve the adjoint equation system which is linear. We show examples to compare the accuracy of the iterative solver and the sparse matrix solver. Chapter 5 discusses optimization algorithms. It covers the evaluation of computational efficiency and the memory requirements of different optimization algorithms. Technical details for applying quasi-Newton method are discussed in this chapter. In Chapter 6, we apply optimization algorithms for a set of history matching problems. The problems considered include three-dimensional single-phase flow gas reservoir, two three-phase flow problems and an example modeled on the Oseberg reservoir at North Sea. In Chapter 7, the conclusions and research contributions of this work are summarized. In Appendix A, a detailed theoretical background for the linear equation solver is provided. In Appendix B, some derivations and theoretical results for

quasi-Newton methods are presented. In Appendix C, details on linear and nonlinear conjugate gradient methods are provided. Appendix D discusses the relationship between conjugate gradient and the quasi-Newton methods.

CHAPTER II

THEORY OF AUTOMATIC HISTORY MATCHING

History matching is a process of changing model parameters to find a set of values that will yield a reservoir simulation prediction of data that matches the observed historical production data. As we all know, the reservoir models are built on limited information which may include core data, log data, geological information, lab data (for fluid properties) and 3D seismic data. Note the preceding data sets do not include production data. Before using a model based on this information to make reservoir management decisions, we wish to integrate historical production data. The historical production data may include bottom-hole or static pressures, gas-oil ratios (GOR's), water-oil ratios (WOR's), or phase flow rates. If the current reservoir description, when input into a reservoir simulator, does not predict data that agree with historical "measured" production data, then we need to change the description, i.e., change the model parameters, which may include gridblock permeabilities (horizontal and vertical), gridblock porosities, parameters that define the relative permeability, fault locations or fault transmissibilities. This adjustment procedure is called history matching. History matching can be done manually or automatically. Today, for complex problems, history matching is largely done manually, although automatic history matching tools are starting to have some impact. Our research focused on automatic history matching of production data for the purpose of constructing an estimate or multiple realizations of model parameters, i.e., porosity and permeability fields, well skin factors and parameters defining relative permeability functions. Because we do history matching automatically, we can consider many more parameters than in the manual history matching process. History matching is done in a Bayesian framework so that estimates and realizations are consistent with

a prior geostatistical model formulated from static data. Automatic history matching is accomplished by applying an optimization algorithm to minimize an objective function which includes a sum of data mismatch terms squared. In this approach, one can include time-lapse seismic data or even "hard data" such as gridblock porosity interpreted from a log as well as production data in the data mismatch terms. In the Bayesian framework, generating a suite of realizations is equivalent to sampling the a posteriori probability density function. If sampling is done properly, then the set of realizations will provide a correct assessment of the uncertainty in the model parameters. By simulating future performance under proposed operating conditions using each realization as reservoir simulation input, and constructing statistics for the set of outcomes (e.g., cumulative oil production, WOR's, GOR's and breakthrough time), one can evaluate the uncertainty in predicted performance.

2.1 Prior Model and Data Measurement Errors

Let m be an N_m-dimensional column vector that contains all the model parameters we want to estimate or simulate, where N_m represents the number of model parameters. We write m as

$$m = \begin{bmatrix} m_r \\ m_s \end{bmatrix}, \tag{2.1}$$

where m_s is the vector of the well skin factors and m_r is the vector for the rock property fields. In our work, we use only rock gridblock permeabilities and porosities and well skin factors as model parameters. These parameters are modeled as random variables, so m is a random vector. We approximate the prior reservoir model parameters as multivariate Gaussian with prior mean given by

$$m_{\text{prior}} = \begin{bmatrix} m_{\text{r,prior}} \\ m_{\text{s,prior}} \end{bmatrix}, \tag{2.2}$$

and prior covariance matrix given by

$$C_M = \begin{bmatrix} C_r & 0 \\ 0 & C_s \end{bmatrix}, \tag{2.3}$$

where C_s is the covariance matrix for well skin factors and C_r is the covariance matrix for the rock property fields. The dimension of m is N_m, so C_M is an $N_m \times N_m$ matrix. As the well skin factors are assumed to be independent Gaussian random variables, C_s is a diagonal matrix. C_r is obtained from the geostatistical information. Assuming the prior model is multivariate Gaussian, the probability density function (pdf) for the prior model is given by

$$p(m) = a \exp(-O_m(m)). \tag{2.4}$$

where a is the normalizing constant and

$$O_m(m) = \frac{1}{2}(m - m_{\text{prior}})^T C_M^{-1}(m - m_{\text{prior}}). \tag{2.5}$$

We let d_{obs} be an N_d-dimensional column vector that contains all observed production data that will be history matched, where N_d denotes the number of conditioning production data. Let e be an N_d-dimensional column vector of data measurement errors. Here, we assume that e is a Gaussian random vector with mean equal to zero and covariance matrix given by C_D. Here, we also assume that the data measurement errors are independent although there exist evidence that this may not always be a good assumption, see Aanonsen et al. (2002). Therefore, the data covariance matrix C_D is an $N_d \times N_d$ diagonal matrix. Each entry of C_D corresponds to the variance of a particular measurement error and these variances are not assumed to be identical. For example, we expect the "measurement" of GOR to be much less accurate than the measurement of bottom-hole pressures. The probability density function (pdf) for the data measurement error is given by

$$p(e) = b \exp(-\frac{1}{2} e^T C_D^{-1} e). \tag{2.6}$$

Let d be an N_d-dimensional column vector that contains the predicted data given model m. The equation

$$d = g(m) \tag{2.7}$$

represents the operation of calculating data d corresponding to d_{obs} for a given model m, i.e, the forward simulation run. If m is the true model, then the difference between

d and d_{obs} represents measurement error, i.e.,

$$e = d - d_{\text{obs}}. \tag{2.8}$$

Suppose we are given model m, but have not yet measured d_{obs}, then $d_{\text{obs}} = d + e$ is a random vector. The pdf for d_{obs} given m is

$$\begin{aligned} p(d_{\text{obs}}|m) &= b \exp\big(-\frac{1}{2}(d - d_{\text{obs}})^T C_D^{-1}(d - d_{\text{obs}})\big) \\ &= b \exp\big(-\frac{1}{2}(g(m) - d_{\text{obs}})^T C_D^{-1}(g(m) - d_{\text{obs}})\big). \end{aligned} \tag{2.9}$$

If we treat m as the random variable and assume d_{obs} is given, then Eq. 2.9 gives the likelihood of m given d_{obs} and we write

$$l(m|d_{\text{obs}}) = b \exp\big(-\frac{1}{2}(g(m) - d_{\text{obs}})^T C_D^{-1}(g(m) - d_{\text{obs}})\big). \tag{2.10}$$

The most likely model or the maximum likely estimate is the model that maximizes $l(m|d_{\text{obs}})$, i.e., the model that minimizes

$$O_d(m) = \frac{1}{2}(g(m) - d_{\text{obs}})^T C_D^{-1}(g(m) - d_{\text{obs}}). \tag{2.11}$$

Even though the estimate of the model obtained by minimizing Eq. 2.11 is the same as the least squares estimate, Eq. 2.10 has statistical meaning. If we characterize the measurement error correctly, then Eq. 2.10 represents the likelihood function of the model. By minimizing Eq. 2.11, we get the maximum likelihood estimate of the model.

2.2 Bayesian Estimate

For automatic history matching problems of interest to us, the number of model parameters is usually greater than the number of independent production data and thus the history matching problem does not have a unique solution. If the Gauss-Newton procedure is applied to minimize an objective function consisting of only the sum of squared production data misfit terms, the Hessian matrix will be singular and the optimization algorithm will be unstable. This instability problem can be avoided by adding a regularization term to the objective function to be minimized;

see Tikhonov (1963) and Parker (1994). With a proper regularization, the Hessian matrix in the Gauss-Newton method will be real symmetric positive definite and hence nonsingular. In this work, we use a prior geostatistical model to provide regularization. With this approach, the history matching problem is equivalent to a Bayesian estimation problem, see Gavalas et al. (1976); Tarantola (1987); He et al. (1997); Wu et al. (1999). With the application of Bayes theorem, we can estimate the conditional probability density function (the a posteriori pdf) for the model parameters m given observations d_{obs}, i.e.,

$$p(m|d_{\text{obs}}) = \frac{p(d_{\text{obs}}|m)p(m)}{p(d_{\text{obs}})} = \frac{p(d_{\text{obs}}|m)p(m)}{\int p(d_{\text{obs}}|u)p(u)du} = \frac{p(d_{\text{obs}}|m)p(m)}{\int p(d_{\text{obs}},u)du}, \tag{2.12}$$

where $p(d_{\text{obs}}, u)$ is the joint probability density function. Using 2.4 and Eq. 2.9 in Eq. 2.12, we can write the a posteriori pdf as

$$\pi(m) = c\exp(-O(m)), \tag{2.13}$$

where c is the normalizing constant and

$$\begin{aligned} O(m) &= \frac{1}{2}(m - m_{\text{prior}})^T C_M^{-1}(m - m_{\text{prior}}) + \frac{1}{2}(g(m) - d_{\text{obs}})^T C_D^{-1}(g(m) - d_{\text{obs}}) \\ &= O_m(m) + O_d(m) \end{aligned} \tag{2.14}$$

which is the total objective function we wish to minimize in the history matching procedure. The total objective function contains two parts, the model mismatch part $O_m(m)$ and the data mismatch part $O_d(m)$. The model mismatch part $O_m(m)$ provides the regularization for the objective function to avoid unrealistic changes in the model parameters. Minimizing Eq. 2.14 gives the maximum a posteriori (MAP) estimate of the model which is the most probable model.

2.3 Evaluation of Uncertainty

The MAP estimate of the model gives a very smooth model which does not reflect the heterogeneity that would be typical for a realization generated from the prior model. We are more interested in generating multiple realizations to evaluate

the uncertainty in reservoir description and performance prediction. The general approach we follow here to characterize the uncertainty in reservoir performance prediction is the one that has been advocated by Oliver (1996), He et al. (1997) and Reynolds et al. (1999). This approach is consistent with one promulgated by the Norwegian school of reservoir characterization; see, for example, Omre et al. (1993) and Holden et al. (1995). In our approach, we (i) develop an a posteriori probability density function (pdf) for the model parameters; (ii) sample this pdf to obtain a set of realizations of the model; (iii) predict future reservoir performance for each realization using a reservoir simulator; and (iv) construct statistics (mean, variance, histograms, etc.) for the outcomes (e.g., cumulative oil production, water cut, producing gas-oil ratio or breakthrough times) to evaluate the uncertainty in performance predictions and to evaluate risk in reservoir management decisions.

Sampling the pdf is difficult. The rejection algorithm is theoretically sound, but completely impractical for the general problem of conditioning a reservoir model to production data unless the uncertainty in the production data (noise and modeling error) is very large compared to the uncertainty in the prior model. Liu et al. (2001) recently presented a paper on conditioning a simple one-dimensional model with forty model parameters to pressure data. They tried several different probability density functions to propose realizations, but were unable to generate more than a handful of valid samples using rejection despite proposing millions of candidate realizations.

Markov chain Monte Carlo (MCMC) methods also provide a theoretically sound method for sampling the a posteriori pdf correctly. Unfortunately, even with modifications aimed at improving computational efficiency, the MCMC approach appears to be too computationally inefficient for practical applications; see Oliver et al. (1997) and Bonet-Cunha et al. (1998).

In this work, the randomized maximum likelihood method is used for sampling. The randomized maximum likelihood method refers to the sampling procedure presented within the context of MCMC methods by Oliver et al. (1996), and was also mentioned without discussion by Kitanidis (1995). To generate a realization with

this procedure, we calculate an unconditional realization m_{uc} from

$$m_{\text{uc}} = m_{\text{prior}} + C_M^{1/2} z_M, \tag{2.15}$$

where z_M is an N_m-dimensional column vector of independent standard random normal deviates. The matrix $C_M^{1/2}$ is a square root of C_M and is normally chosen as $C_M^{1/2} = L$ where

$$C_M = LL^T, \tag{2.16}$$

is the Cholesky decomposition of C_M. Similarly a realization of the data is generated from

$$d_{\text{uc}} = d_{\text{obs}} + C_D^{1/2} z_D, \tag{2.17}$$

where z_D is an N_d-dimensional column vector of standard random normal deviates.

If the model is very large, e.g., if the number of model parameters is on the order of ten thousand or larger, then the covariance matrix C_M is so large that Choleskey decomposition of C_M becomes computationally expensive and impractical. Then Gaussian co-simulation provides a practical alternative for generating unconditional realizations of m; see, for example Gómez-Hernández and Journel (1992). The associated conditional realization of m is the model that minimizes

$$O(m) = \frac{1}{2}(m - m_{\text{uc}})^T C_M^{-1}(m - m_{\text{uc}}) + \frac{1}{2}(g(m) - d_{\text{uc}})^T C_D^{-1}(g(m) - d_{\text{uc}}). \tag{2.18}$$

Similar to results given in Tarantola (1987) $O(m)$ can be approximated by a chi-squared distribution with expectation given by $E(O(m)) = N_d$ and standard deviation given approximately by $\sigma(O(m)) \approx \sqrt{2N_d}$. Virtually all samples should be within five standard deviations of the mean. Thus, if applying an optimization algorithm to minimize Eq. 2.18 gives a result m_c, we accept m_c as a legitimate realization if and only if

$$N_d - 5\sqrt{2N_d} \leq O(m_c) \leq N_d + 5\sqrt{2N_d}. \tag{2.19}$$

If Eq. 2.19 is not satisfied, the minimization algorithm has failed. This failure can occur if the algorithm converges to a local minimum or converges so slowly that the decrease in the objective function is so small that the convergence criteria, which is

based on the change in the objective function, is satisfied before we actually reach a minimum. The chi-squared approximation on which Eq. 2.19 is based assumes that $g(m)$ is a linear function of m. In some cases, this chi-squared approximation appears to be valid; see, for example the results on conditioning a stochastic channel to pressure data of Zhang et al. (2002) and the results on history matching pressure data from a single-phase gas reservoir presented in Section 6.1. In other cases, we obtain values of the objective function at convergence much greater than $N_d+5\sqrt{2N_d}$; see, for example the three-phase flow history matching example of Section 6.3. It is unclear whether the failure to satisfy Eq. 2.19 at convergence occurs because $g(m)$ is not well approximated by a linear function of m or for some other reasons.

Ultimately, we wish to be able to history match several hundred to a few thousand production data to generate realizations of tens of thousands of model parameters. Thus, computational efficiency is an extremely important consideration. It is equally important that the algorithm should be robust, i.e., the number of convergence failures should be extremely small. If minimization of $O(m)$ frequently result in values of $O(m_c)$ which do not satisfy Eq. 2.19, the utility of the optimization algorithm is diminished.

2.4 Doubly Stochastic Model

In the history matching procedure, people usually treat the prior mean of the model parameters as known and constant. In practice, however, the prior mean is also obtained from observations and is uncertain. If this situation happens, we need to correct the prior mean of model parameters. Thus, following Reynolds et al. (1999), we introduce a random vector denoted by θ to model the correction to the prior mean where we assume θ can be written as

$$\theta = \begin{bmatrix} \alpha_1 e_1 \\ \alpha_2 e_2 \\ \vdots \\ \alpha_{N_\alpha} e_{N_\alpha} \end{bmatrix} = \begin{bmatrix} e_1 & O & \dots & O \\ O & e_2 & \dots & O \\ \vdots & \vdots & \ddots & \vdots \\ O & O & \dots & e_{N_\alpha} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{N_\alpha} \end{bmatrix} \equiv E\alpha, \tag{2.20}$$

where

$$E = \begin{bmatrix} e_1 & O & \dots & O \\ O & e_2 & \dots & O \\ \vdots & \vdots & \ddots & \vdots \\ O & O & \dots & e_{N_\alpha} \end{bmatrix} \tag{2.21}$$

and

$$e = [1, \cdots, 1]^T \tag{2.22}$$

with appropriate dimensions.

We assume the α_j's are independent random variables in which case C_α is a diagonal, positive definite matrix. The pdf for α is specified as

$$p_\alpha(\alpha) = a\exp\Big(-\frac{1}{2}(\alpha - \alpha_0)^T C_\alpha^{-1}(\alpha - \alpha_0)\Big). \tag{2.23}$$

The conditional pdf of M given α is

$$p_{m|\alpha}(m \mid \alpha) = a\exp\Big(-\frac{1}{2}(m - m_{\text{prior}} - E\alpha)^T C_M^{-1}(m - m_{\text{prior}} - E\alpha)\Big). \tag{2.24}$$

The joint pdf for M and α is given by

$$p_{m,\alpha}(m, \alpha) = p_{M|\alpha}(m \mid \alpha)p_\alpha(\alpha) =$$
$$a\exp\Big(-\frac{1}{2}(m - m_{\text{prior}} - E\alpha)^T C_M^{-1}(m - m_{\text{prior}} - E\alpha) - \frac{1}{2}(\alpha - \alpha_0)^T C_\alpha^{-1}(\alpha - \alpha_0)\Big). \tag{2.25}$$

Eq. 2.25 implies that the expectation of M given α is

$$E[M \mid \alpha] = m_{\text{prior}} + E\alpha, \tag{2.26}$$

where in the preceding equation, E denotes the expectation. Let $\hat{M}$ denote the random vector including the model vector M and vector α, i.e.,

$$\hat{M} = \begin{bmatrix} M \\ \alpha \end{bmatrix}. \tag{2.27}$$

and let $\hat{m}$ denote a realization of $\hat{M}$. Let d represent the data vector which is obtained by running the simulator given the model m. The relationship between the data d

and the model m is specified by Eq. 2.7. Given m, the observed pressure data is treated as a realization of the random vector $d_{\text{obs}} = g(m) + \epsilon$. By the application of Bayes theorem, we can write the a posteriori pdf for (m, α) conditional to d_{obs} as

$$\pi(m,\alpha) = p_{(m,\alpha)}\big(m, \alpha \mid d_{\text{obs}}\big) = a\text{exp}\big(-O(m,\alpha)\big) \tag{2.28}$$

where $O(m,\alpha)$ is given by

$$\begin{aligned} O(m,\alpha) = &\frac{1}{2}(g(m) - d_{\text{obs}})^T C_D^{-1}(g(m) - d_{\text{obs}}) + \\ &\frac{1}{2}(m - m_{\text{prior}} - E\alpha)^T C_M^{-1}(m - m_{\text{prior}} - E\alpha) + \frac{1}{2}(\alpha - \alpha_0)^T C_\alpha^{-1}(\alpha - \alpha_0). \end{aligned} \tag{2.29}$$

Eq. 2.29 is the objective function we should minimize for the doubly stochastic model.

To evaluate the uncertainty, we use the randomized maximum likelihood method (Kitanidis (1995), Oliver et al. (1996)) to sample the a posteriori pdf. There are three steps in this procedure: (i) generate an unconditional realization of the model and correction to the prior mean, respectively, from

$$m_{\text{uc}} = m_{\text{prior}} + C_M^{1/2} z_M, \tag{2.30}$$

where z_M is an N_m-dimensional column vector of independent standard random normal deviates and

$$\alpha_{\text{uc}} = \alpha_0 + C_\alpha^{1/2} z_\alpha, \tag{2.31}$$

where z_α is independent standard random normal deviates with consistent dimensions; (ii) generate a realization of the data from

$$d_{\text{uc}} = d_{\text{obs}} + C_D^{1/2} z_D, \tag{2.32}$$

where z_D is an N_d-dimensional column vector of standard random normal deviates; (iii) replace m_{prior}, d_{obs} and α_0, respectively, by m_{uc}, d_{uc} and α_{uc} in Eq. 2.29 to obtain the modified objective function given by

$$\begin{aligned} O_r(m,\alpha) = &\frac{1}{2}(g(m) - d_{\text{uc}})^T C_D^{-1}(g(m) - d_{\text{uc}}) \\ &+\frac{1}{2}(m - m_{\text{uc}} - E\alpha)^T C_M^{-1}(m - m_{\text{uc}} - E\alpha) + \frac{1}{2}(\alpha - \alpha_{\text{uc}})^T C_\alpha^{-1}(\alpha - \alpha_{\text{uc}}). \end{aligned} \tag{2.33}$$

To obtain a realization conditional to the data, we minimize Eq. 2.33 once. To obtain n independent realizations, we repeat the procedure n times.

CHAPTER III

RESERVOIR SIMULATOR AND ADJOINT METHOD

In the first section, we briefly discuss the forward finite-difference equations that are required to solve in the reservoir simulator. In the second section, we consider the adjoint method which is used to calculate the sensitivity coefficients, which represent derivatives of the data with respect to model parameters. The gradient of the objective function is the derivative of the objective function with respect to the model parameters. Therefore, the whole objective function can be treated like an individual datum so that the gradient of the objective function can be obtained by one application of the adjoint method. This chapter gives the basic formulas of the adjoint method; see Li (2001) for detailed information. We focus on the calculation of the gradient of the objective function.

3.1 The Reservoir Simulator

For simplicity, the reservoir is assumed to be a rectangular parallelepiped which occupies the region

$$\Omega = \{(x, y, z) \mid 0 < x < L_x, 0 < y < L_y, 0 < z < L_z\}. \tag{3.1}$$

The simulator used is based on a fully-implicit, finite-difference formulation of the three-phase flow, black-oil equations expressed in a x–y–z coordinate system which apply on Ω; see Eq. 3.1. Suppose there are N_x, N_y, N_z gridblocks in the $x-$, $y-$ and $z-$ directions respectively. Let N be the total number of gridblocks, i.e., $N = N_x \times N_y \times N_z$. At each of the N gridblocks, three basic finite-difference equations apply. These three equations represent the mass balance for each of the three components, i.e., oil, gas and water. In addition, a constraint is applied at

each of the N_w wells to yield N_w additional equations. At each well at each time step, either an individual phase flow rate, the total flow rate or the wellbore pressure may be specified as a well constraint. In the results considered in this work, capillary pressures are assumed to be negligible. The fully-implicit, black-oil simulator (CLASS-Chevron's Limited Applications Simulation System) used in this work was provided by Chevron.

For gridblock i, the primary variables that are solved for are case dependent. Table 3.1 summarizes the different cases and the primary variables solved for in each case. In the column entitled "Equations", Sum denotes the total mass balance equation (i.e., the summation of the oil, gas and water equations); Oil represents the oil mass balance equation and Gas represents the gas mass balance equation.

Table 3.1: Equations and unknowns solved for in the simulator.

Phases		Equations	Unknowns	Auxiliary equation
O-W-G	$S_g > 0$	Sum, Oil, Gas	p, S_o, S_g	$S_w = 1 - S_o - S_g$; R_s from PVT table
	$S_g = 0$	Sum, Oil, Gas	p, S_o, R_s	$S_g = 0$; $S_w = 1 - S_o - S_g$
O-W		Sum, Oil	p, S_o	$S_w = 1 - S_o$
W-G		Sum, Gas	p, S_g	$S_w = 1 - S_g$
O-G	$S_g > 0$	Sum, Gas	p, S_g	$S_o = 1 - S_g$; R_s from PVT table
	$S_g = 0$	Sum, Gas	p, R_s	$S_g = 0$; $S_o = 1 - S_g$

At each time step, we can output p, S_o, S_g, S_w and R_s of each individual gridblock from CLASS. From these primary variables, we can calculate all the derivatives required for constructing the adjoint system based on the PVT table. In addition to the gridblock variables, the flowing wellbore pressure, $p_{wf,l}$ at the lth well at a specified depth is also a primary variable. We let y^n denote a column vector which contains the set of primary variables (pressures and saturations) at time step n. At gridblock i, the finite-difference equation for component u can be written as

$$f_{u,i}(y^{n+1}, y^n, m) = 0, \tag{3.2}$$

for $u = o, w, g$ and $i = 1, \ldots, N$. The well constraints are represented by

$$f_{wf,l}(y^{n+1}, y^n, m) = 0, \tag{3.3}$$

for $l = 1, 2, \ldots, N_w$. For simplicity, we let

$$f_{u,i}^{n+1} = f_{u,i}(y^{n+1}, y^n, m), \tag{3.4}$$

and

$$f_{wf,l}^{n+1} = f_{wf,l}(y^{n+1}, y^n, m), \tag{3.5}$$

then Eqs. 3.2 and 3.3 can be rewritten as

$$f_{u,i}^{n+1} = 0, \tag{3.6}$$

and

$$f_{wf,l}^{n+1} = 0, \tag{3.7}$$

respectively. If the flowing wellbore pressure at well l at the datum depth at time t^{n+1} is specified to be equal to $p_{wf,l,0}^{n+1}$, then Eq. 3.5 is simplified to

$$f_{wf,l}^{n+1} = p_{wf,l}^{n+1} - p_{wf,l,0}^{n+1} = 0. \tag{3.8}$$

In CLASS, the three equations for three-phase problem that are solved at gridblock i are

$$f_{1,i}^{n+1} = f_{o,i}^{n+1} + f_{w,i}^{n+1} + f_{g,i}^{n+1} = 0 \tag{3.9}$$

$$f_{2,i}^{n+1} = f_{o,i}^{n+1} = 0 \tag{3.10}$$

$$f_{3,i}^{n+1} = f_{g,i}^{n+1} = 0 \tag{3.11}$$

where $i = 1, 2, \cdots, N$. If the following three equations

$$f_{1,i}^{n+1} = f_{o,i} = 0 \tag{3.12}$$

$$f_{2,i}^{n+1} = f_{w,i}^{n+1} = 0 \tag{3.13}$$

$$f_{3,i}^{n+1} = f_{g,i}^{n+1} = 0 \tag{3.14}$$

instead of Eqs. 3.9 through 3.11 are used to construct the Jacobian matrix, then we will have trouble for some situations to do incomplete LU decomposition of the

Jacobian matrix in order to use orthomin which is an iterative solver. From Table 3.1, we can see no matter which case happens, the pressure is always one of the primary variables. Hence, in the Jacobin matrix, the derivative of a certain equation at ith gridblock with respect to the pressure at the ith gridblock is always the diagonal element. The location of this entry depends on how you order the primary variables. If the pressure is ordered as the first primary variable in each gridblock as people usually do, then in the Jacobian matrix, every third diagonal entry will be $df_{1,i}/dp_i$ where i is the gridblock index. If $f_{1,i} = f_{o,i}$, then the derivative $df_{1,i}/dp_i$ is zero whenever oil saturation $S_{o,i}$ is zero. Because if this is the case, then every individual term involved in $df_{1,i}/dp_i$ is related to oil saturation by either relative permeability or S_o itself and becomes zero. The subroutine we used to perform the incomplete LU decomposition will perform the operation of dividing the row of Jacobian matrix by the diagonal element. Therefore, if the situation presented above happens, e.g., in the gas cap area, then this subroutine will be terminated because of the illegal math operation.

Eq. 3.7 and Eqs. 3.9 through 3.11 represent a system of N_e equations where

$$N_e = 3N + N_w. \tag{3.15}$$

These N_e equations are solved to obtain the values of the primary variables at time $t^{n+1} = t^n + \Delta t^n$. For wells at which the flowing bottom-hole pressure is specified, phase flow rates at each well are computed by Peaceman's equation (Peaceman, 1983). The component flow rates from the perforated layer k of well l (at gridblock (i, j, k)) at time step $n + 1$ can be evaluated as

$$q^{n+1}_{o,i,j,k} = WI_{i,j,k} \left(\frac{k_{ro}}{B_o \mu_o} \right)^{n+1}_{i,j,k} (p^{n+1}_{i,j,k} - p^{n+1}_{wf,l,k}), \tag{3.16}$$

$$q^{n+1}_{w,i,j,k} = WI_{i,j,k} \left(\frac{k_{rw}}{B_w \mu_w} \right)^{n+1}_{i,j,k} (p^{n+1}_{i,j,k} - p^{n+1}_{wf,l,k}), \tag{3.17}$$

and

$$\begin{aligned} q^{n+1}_{g,i,j,k} &= WI_{i,j,k} \left(\frac{k_{rg}}{B_g \mu_g} \right)^{n+1}_{i,j,k} (p^{n+1}_{i,j,k} - p^{n+1}_{wf,l,k}) + R^{n+1}_{so,i,j,k} q^{n+1}_{o,i,j,k} \\ &= WI_{i,j,k} \left(\frac{k_{rg}}{B_g \mu_g} + R_s \frac{k_{ro}}{B_o \mu_o} \right)^{n+1}_{i,j,k} (p^{n+1}_{i,j,k} - p^{n+1}_{wf,l,k}). \end{aligned} \tag{3.18}$$

The rates $q^{n+1}_{o,i,j,k}$ and $q^{n+1}_{w,i,j,k}$ are in units of STB/Day, and $q^{n+1}_{g,l,k}$ has units of SCF/Day. Here, layer k means the wellbore gridblock with z-direction gridblock index equal to k. The well index term $WI_{i,j,k}$ is the geometry part of productivity index and it is defined by

$$WI_{i,j,k} = \frac{0.00708 \triangle z_k \sqrt{k_{x,i,j,k} k_{y,i,j,k}}}{\ln(r_{o,l,k}/r_{w,l,k}) + s_{l,k}}, \tag{3.19}$$

and $r_{o,l,k}$ is defined

$$r_{o,l,k} = \frac{0.28073 \triangle x_i \sqrt{1 + \frac{k_{x,i,j,k}}{k_{y,i,j,k}} \left(\frac{\triangle y_j}{\triangle x_i}\right)^2}}{1 + \sqrt{k_{x,i,j,k}/k_{y,i,j,k}}}. \tag{3.20}$$

Here, $r_{w,l,k}$ is the wellbore radius of the well l at layer k and $s_{l,k}$ is the skin factor for well l at layer k.

The complete system of equations can formally be written as

$$f^{n+1} = f(y^{n+1}, y^n, m) = \begin{bmatrix} f^{n+1}_{1,1} \\ f^{n+1}_{o,1} \\ f^{n+1}_{g,1} \\ f^{n+1}_{1,2} \\ \vdots \\ f^{n+1}_{g,N} \\ f^{n+1}_{wf,1} \\ \vdots \\ f^{n+1}_{wf,N_w} \end{bmatrix} = 0, \tag{3.21}$$

where

$$m = [m_1, m_2, \cdots, m_{N_m}]^T, \tag{3.22}$$

and

$$y^{n+1} = [p^{n+1}_1, S^{n+1}_{o,1}, x^{n+1}_1, p^{n+1}_2, \cdots, p^{n+1}_i, S^{n+1}_{o,i}, x^{n+1}_i, \cdots, x^{n+1}_N, p^{n+1}_{wf,1}, \cdots, p^{n+1}_{wf,N_w}]^T, \tag{3.23}$$

where

$$x^{n+1}_i = \begin{cases} S^{n+1}_{g,i} & \text{for} \quad S_{g,i} > 0 \\ R^{n+1}_{s,i} & \text{for} \quad S_{g,i} = 0 \end{cases} \tag{3.24}$$

Eq. 3.21 is solved by the Newton-Raphson method (Aziz and Settari, 1979) which can be written as

$$J^{n+1,k}\delta y^{n+1,k+1} = -f^{n+1,k} \tag{3.25}$$

$$y^{n+1,k+1} = y^{n+1,k} + \delta y^{n+1,k+1}, \tag{3.26}$$

where k is the Newton-Raphson iteration index, n is the time step index and

$$J^{n+1,k} = \left[\nabla_{y^{n+1}}(f^{n+1})^T\right]^T_{y^{n+1,k}}, \tag{3.27}$$

is the Jacobian matrix evaluated at $y^{n+1,k}$, which represents the kth approximation for y^{n+1}. The initial guess for y^{n+1} is chosen as the solution at the previous time step, i.e.,

$$y^{n+1,0} = y^n. \tag{3.28}$$

3.2 Adjoint Equations

We define a general scalar function by

$$\beta = \beta(y^1, ..., y^L, m), \tag{3.29}$$

where L corresponds to the last time step t^L at which one wishes to compute sensitivity coefficients. The objective is to compute the derivatives of β with respect to the model parameters m. We obtain an adjoint functional J by adjoining Eq. 3.21 to the function β:

$$J = \beta + \sum_{n=0}^{L}(\lambda^{n+1})^T f^{n+1}, \tag{3.30}$$

where λ^{n+1} is the vector of adjoint variables at time step $n+1$, and is given by

$$\lambda^{n+1} = \left[\lambda_1^{n+1}, \quad \lambda_2^{n+1}, \quad \ldots, \quad \lambda_{N_e}^{n+1}\right]^T. \tag{3.31}$$

Taking the total differential of Eq. 3.30, and doing some simple rearranging

gives

$$\begin{aligned} dJ &= d\beta + \sum_{n=0}^{L}\Big\{(\lambda^{n+1})^T[\nabla_{y^{n+1}}(f^{n+1})^T]^T dy^{n+1} + [\nabla_m(f^{n+1})^T]^T dm\Big\} \\ &\quad + \sum_{n=0}^{L}(\lambda^{n+1})^T[\nabla_{y^n}(f^{n+1})^T]^T dy^n \\ &= d\beta + BT + \sum_{n=1}^{L}\{[(\lambda^n)^T[\nabla_{y^n}(f^n)^T]^T \\ &\quad + (\lambda^{n+1})^T[\nabla_{y^n}(f^{n+1})^T]^T]dy^n + (\lambda^n)^T[\nabla_m(f^n)^T]^T dm\}, \end{aligned} \tag{3.32}$$

where

$$BT = (\lambda^{L+1})^T\Big\{[\nabla_{y^{L+1}}(f^{L+1})^T]^T dy^{L+1} + [\nabla_m(f^{L+1})^T]^T dm\Big\} + (\lambda^1)^T\big[\nabla_{y^0}(f^1)^T\big]^T dy^0. \tag{3.33}$$

The total differential of β can be written as

$$d\beta = \sum_{n=1}^{L}[\nabla_{y^n}\beta]^T dy^n + [\nabla_m\beta]^T dm. \tag{3.34}$$

The initial conditions are fixed, so

$$dy^0 = 0. \tag{3.35}$$

Choosing

$$\lambda^{L+1} = 0, \tag{3.36}$$

it follows that $BT = 0$. Using this result and Eq. 3.34 in Eq. 3.32 and rearranging the resulting equation gives

$$\begin{aligned} dJ = \sum_{n=1}^{L}\Big[\big\{(\lambda^n)^T[\nabla_{y^n}(f^n)^T]^T + (\lambda^{n+1})^T[\nabla_{y^n}(f^{n+1})^T]^T \\ + [\nabla_{y^n}\beta]^T\big\}dy^n\Big] + \Big\{[\nabla_m\beta]^T + \sum_{n=1}^{N}(\lambda^n)^T[\nabla_m(f^n)^T]^T\Big\}dm. \end{aligned} \tag{3.37}$$

To obtain the adjoint system, the coefficients multiplying dy^n in Eq. 3.37 are set equal to zero; i.e., we require that the adjoint variables satisfy

$$(\lambda^n)^T[\nabla_{y^n}(f^n)^T]^T + (\lambda^{n+1})^T[\nabla_{y^n}(f^{n+1})^T]^T + [\nabla_{y^n}\beta]^T = 0. \tag{3.38}$$

Taking the transpose of Eq. 3.38, gives the adjoint system

$$\boxed{\left[\nabla_{y^n}(f^n)^T\right]\lambda^n = -\left[\nabla_{y^n}(f^{n+1})^T\right]\lambda^{n+1} - \nabla_{y^n}\beta.} \tag{3.39}$$

where

$$\nabla_{y^n}[f^n]^T = \begin{bmatrix} \frac{\partial f^n_{1,1}}{\partial p^n_1} & \frac{\partial f^n_{w,1}}{\partial p^n_1} & \cdots & \frac{\partial f^n_{g,N}}{\partial p^n_1} & \frac{\partial f^n_{wf,1}}{\partial p^n_1} & \cdots & \frac{\partial f^n_{wf,N_w}}{\partial p^n_1} \\ \frac{\partial f^n_{1,1}}{\partial S^n_{w,1}} & \frac{\partial f^n_{w,1}}{\partial S^n_{w,1}} & \cdots & \frac{\partial f^n_{g,N}}{\partial S^n_{w,1}} & \frac{\partial f^n_{wf,1}}{\partial S^n_{w,1}} & \cdots & \frac{\partial f^n_{wf,N_w}}{\partial S^n_{w,1}} \\ \frac{\partial f^n_{1,1}}{\partial S^n_{g,1}} & \frac{\partial f^n_{w,1}}{\partial S^n_{g,1}} & \cdots & \frac{\partial f^n_{g,N}}{\partial S^n_{g,N}} & \frac{\partial f^n_{wf,1}}{\partial S^n_{g,1}} & \cdots & \frac{\partial f^n_{wf,N_w}}{\partial S^n_{g,1}} \\ \frac{\partial f^n_{1,1}}{\partial p^n_2} & \frac{\partial f^n_{w,1}}{\partial p^n_2} & \cdots & \frac{\partial f^n_{g,N}}{\partial p^n_2} & \frac{\partial f^n_{wf,1}}{\partial p^n_2} & \cdots & \frac{\partial f^n_{wf,N_w}}{\partial p^n_2} \\ \vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ \frac{\partial f^n_{1,1}}{\partial S^n_{g,N}} & \frac{\partial f^n_{w,1}}{\partial S^n_{g,N}} & \cdots & \frac{\partial f^n_{g,N}}{\partial S^n_{g,N}} & \frac{\partial f^n_{wf,1}}{\partial S^n_{g,N}} & \cdots & \frac{\partial f^n_{wf,N_w}}{\partial S^n_{g,N}} \\ \frac{\partial f^n_{1,1}}{\partial p^n_{wf,1}} & \frac{\partial f^n_{w,1}}{\partial p^n_{wf,1}} & \cdots & \frac{\partial f^n_{g,N}}{\partial p^n_{wf,1}} & \frac{\partial f^n_{wf,1}}{\partial p^n_{wf,1}} & \cdots & \frac{\partial f^n_{wf,N_w}}{\partial p^n_{wf,1}} \\ \vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ \frac{\partial f^n_{1,1}}{\partial p^n_{wf,N_w}} & \frac{\partial f^n_{w,1}}{\partial p^n_{wf,N_w}} & \cdots & \frac{\partial f^n_{g,N}}{\partial p^n_{wf,N_w}} & \frac{\partial f^n_{wf,1}}{\partial p^n_{wf,N_w}} & \cdots & \frac{\partial f^n_{wf,N_w}}{\partial p^n_{wf,N_w}} \end{bmatrix}, \tag{3.40}$$

where $f_{1,i}$ is given by Eq. 3.9 and

$$\nabla_{y^n}[f^{n+1}]^T = \begin{bmatrix} \frac{\partial f^{n+1}_{1,1}}{\partial p^n_1} & \frac{\partial f^{n+1}_{w,1}}{\partial p^n_1} & \cdots & \frac{\partial f^{n+1}_{g,N}}{\partial p^n_1} & \frac{\partial f^{n+1}_{wf,1}}{\partial p^n_1} & \cdots & \frac{\partial f^{n+1}_{wf,N_w}}{\partial p^n_1} \\ \frac{\partial f^{n+1}_{1,1}}{\partial S^n_{w,1}} & \frac{\partial f^{n+1}_{w,1}}{\partial S^n_{w,1}} & \cdots & \frac{\partial f^{n+1}_{g,N}}{\partial S^n_{w,1}} & \frac{\partial f^{n+1}_{wf,1}}{\partial S^n_{w,1}} & \cdots & \frac{\partial f^{n+1}_{wf,N_w}}{\partial S^n_{w,1}} \\ \frac{\partial f^{n+1}_{1,1}}{\partial S^n_{g,1}} & \frac{\partial f^{n+1}_{w,1}}{\partial S^n_{g,1}} & \cdots & \frac{\partial f^{n+1}_{g,N}}{\partial S^n_{g,N}} & \frac{\partial f^{n+1}_{wf,1}}{\partial S^n_{g,1}} & \cdots & \frac{\partial f^{n+1}_{wf,N_w}}{\partial S^n_{g,1}} \\ \frac{\partial f^{n+1}_{1,1}}{\partial p^n_2} & \frac{\partial f^{n+1}_{w,1}}{\partial p^n_2} & \cdots & \frac{\partial f^{n+1}_{g,N}}{\partial p^n_2} & \frac{\partial f^{n+1}_{wf,1}}{\partial p^n_2} & \cdots & \frac{\partial f^{n+1}_{wf,N_w}}{\partial p^n_2} \\ \vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ \frac{\partial f^{n+1}_{1,1}}{\partial S^n_{g,N}} & \frac{\partial f^{n+1}_{w,1}}{\partial S^n_{g,N}} & \cdots & \frac{\partial f^{n+1}_{g,N}}{\partial S^n_{g,N}} & \frac{\partial f^{n+1}_{wf,1}}{\partial S^n_{g,N}} & \cdots & \frac{\partial f^{n+1}_{wf,N_w}}{\partial S^n_{g,N}} \\ \frac{\partial f^{n+1}_{1,1}}{\partial p^n_{wf,1}} & \frac{\partial f^{n+1}_{w,1}}{\partial p^n_{wf,1}} & \cdots & \frac{\partial f^{n+1}_{g,N}}{\partial p^n_{wf,1}} & \frac{\partial f^{n+1}_{wf,1}}{\partial p^n_{wf,1}} & \cdots & \frac{\partial f^{n+1}_{wf,N_w}}{\partial p^n_{wf,1}} \\ \vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ \frac{\partial f^{n+1}_{1,1}}{\partial p^n_{wf,N_w}} & \frac{\partial f^{n+1}_{w,1}}{\partial p^n_{wf,N_w}} & \cdots & \frac{\partial f^{n+1}_{g,N}}{\partial p^n_{wf,N_w}} & \frac{\partial f^{n+1}_{wf,1}}{\partial p^n_{wf,N_w}} & \cdots & \frac{\partial f^{n+1}_{wf,N_w}}{\partial p^n_{wf,N_w}} \end{bmatrix}, \tag{3.41}$$

and

$$\nabla_{y^n}\beta = \left[\frac{\partial \beta}{\partial p^n_1}, \frac{\partial \beta}{\partial S^n_{w,1}}, \frac{\partial \beta}{\partial S^n_{g,1}}, \frac{\partial \beta}{\partial p^n_2}, \cdots, \frac{\partial \beta}{\partial S^n_{g,N}}, \frac{\partial \beta}{\partial p^n_{wf,1}}, \cdots, \frac{\partial \beta}{\partial p^n_{wf,N_w}}\right]^T. \tag{3.42}$$

Note that when we set up the adjoint system, we use the water equation f_w, instead of the oil equation f_o as in CLASS, as the second equation in order to

use the previous code developed by Ruijian Li without modifying it too much. Our results indicate that using f_w instead of f_o as the second equation does not affect the accuracy of the adjoint solutions. When we construct the adjoint system, the entries of the y vector are always p, S_o, S_g and p_{wf}, i.e.,

$$y = [p_1, S_{o,1}, S_{g,1}, p_2, \cdots, p_i, S_{o,i}, S_{g,i}, \cdots, S_{g,N}, p_{wf,1}, \cdots, p_{wf,N_w}], \tag{3.43}$$

whereas in the forward simulator, Eq. 3.23 is used. Our results indicate that this does not affect the accuracy of the adjoint solutions.

Eq. 3.39 with initial condition 3.36 is solved backwards in time for $n = L, L-1, \ldots, 1$. Note that the forward simulation equation is solved forward in time. Also note that the coefficients in Eq. 3.39 are independent of the adjoint variable λ, which means that the adjoint equation is linear. Therefore, solving the adjoint system is cheaper in terms of the computation cost than solving the forward simulation equation which is nonlinear. In the above equations, $\nabla_{y^n}(f^n)^T$ and $\nabla_{y^n}(f^{n+1})^T$ are $N_e \times N_e$ matrices, and $\nabla_{y^n}\beta$ is an N_e-dimensional column vector.

The matrix given by Eq. 3.41 is a diagonal band matrix which is only related to the accumulation terms in the reservoir simulation equations. Note that the coefficient matrix $(\nabla_{y^n}(f^n)^T)$ (Eq. 3.40) in the adjoint system is simply the transpose of the Jacobian matrix of Eq. 3.27 evaluated at y^n when the equations and primary variables used to construct adjoint system are the same as used in the forward equations. As the adjoint system is solved backwards in time, information needed in these matrices (Eqs. 3.40 and 3.41) must be saved from the forward simulation run. In our code, we write all these primary variables to disk to save memory. For details on these equations for computing the derivatives $\nabla_{y^n}(f^n)^T$, $\nabla_{y^n}(f^{n+1})^T$, and $\nabla_{y^n}\beta$ in the adjoint equation, Eq. 3.39, see Li (2001).

As a summary, the adjoint system has the following properties:

(i) the adjoint system is solved backward in time;

(ii) the adjoint system is linear;

(iii) the coefficient matrix in the adjoint system is the transpose of the Jacobian matrix used for solving the forward equations only if the adjoint system is fully consistent with the forward equation, i.e., in each gridblock the same equations and primary variables are used to construct the adjoint system and the flow equation system.

Considering J as a function of m , we can write its total differential as

$$dJ = (\nabla_m J)^T dm. \tag{3.44}$$

By comparing Eq. 3.37 and Eq. 3.44, it follows that the desired sensitivity coefficients for J, or equivalently, β, are given by

$$\boxed{\nabla_m J = \nabla_m \beta + \sum_{n=1}^{L} [\nabla_m (f^n)^T](\lambda^n),} \tag{3.45}$$

where

$$\nabla_m [f^n]^T = \begin{bmatrix} \frac{\partial f^n_{1,1}}{\partial m_1} & \frac{\partial f^n_{w,1}}{\partial m_1} & \frac{\partial f^n_{g,1}}{\partial m_1} & \frac{\partial f^n_{1,2}}{\partial m_1} & \cdots & \frac{\partial f^n_{g,N}}{\partial m_1} & \frac{\partial f^n_{wf,1}}{\partial m_1} & \cdots & \frac{\partial f^n_{wf,N_w}}{\partial m_1} \\ \frac{\partial f^n_{1,1}}{\partial m_2} & \frac{\partial f^n_{w,1}}{\partial m_2} & \frac{\partial f^n_{g,1}}{\partial m_2} & \frac{\partial f^n_{1,2}}{\partial m_2} & \cdots & \frac{\partial f^n_{g,N}}{\partial m_2} & \frac{\partial f^n_{wf,1}}{\partial m_2} & \cdots & \frac{\partial f^n_{wf,N_w}}{\partial m_2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f^n_{1,1}}{\partial m_{N_m}} & \frac{\partial f^n_{w,1}}{\partial m_{N_m}} & \frac{\partial f^n_{g,1}}{\partial m_{N_m}} & \frac{\partial f^n_{1,2}}{\partial m_{N_m}} & \cdots & \frac{\partial f^n_{g,N}}{\partial m_{N_m}} & \frac{\partial f^n_{wf,1}}{\partial m_{N_m}} & \cdots & \frac{\partial f^n_{wf,N_w}}{\partial m_{N_m}} \end{bmatrix}, \tag{3.46}$$

and

$$\nabla_m \beta = \left[\frac{\partial \beta}{\partial m_1}, \frac{\partial \beta}{\partial m_2}, \cdots, \frac{\partial \beta}{\partial m_{N_m}} \right]^T. \tag{3.47}$$

The matrix $\nabla_m [f^n]^T$ is an $N_m \times N_e$ sparse matrix and $\nabla_m \beta$ is an N_m-dimensional column vector. In Eq. 3.45, the gradient $\nabla_m \beta$ involves the partial derivatives of β with respect to the model parameters. If the jth model parameter does not explicitly appear in the expression for β, then $\partial \beta / \partial m_j = 0$. For example, if $\beta = p^n_{wf}$, then we set $\nabla_m \beta = 0$ in Eq. 3.45.

To apply a conjugate gradient (Makhlouf et al., 1993) or variable metric method (Yang and Watson, 1988), we need only compute the gradient of the objective function and this can be done by setting $\beta = O(m)$ in the adjoint procedure. In this

case, one only needs to solve the adjoint system Eq. 3.39 once and substitute the resulting adjoint solutions to Eq. 3.45 to obtain the gradient.

To apply the adjoint method to calculate the sensitivity of the variable β to model parameters m, one needs to solve the adjoint system equation Eq. 3.39 to obtain the adjoint variable λ, and then use Eq. 3.45 to calculate sensitivity coefficients. If we consider permeabilities (k_x, k_y and k_z) and porosities (ϕ) in each individual gridblock, i.e.,

$$m_{k_x} = k_x = [\ k_{x,1},\quad k_{x,2},\quad \cdots,\quad k_{x,N}\ ]^T, \tag{3.48}$$

$$m_{k_y} = k_y = [\ k_{y,1},\quad k_{y,2},\quad \cdots,\quad k_{y,N}\ ]^T, \tag{3.49}$$

$$m_{k_z} = k_z = [\ k_{z,1},\quad k_{z,2},\quad \cdots,\quad k_{z,N}\ ]^T, \tag{3.50}$$

and

$$m_\phi = \phi = [\ \phi_1,\quad \phi_2,\quad \cdots,\quad \phi_N\ ]^T, \tag{3.51}$$

then from Eq. 3.45, the equations to calculate the derivatives with respect to k_x, k_y, k_z and ϕ are given by

$$\nabla_{k_x} J = \nabla_{k_x}\beta + \sum_{n=1}^{L}[\nabla_{k_x}(f^n)^T](\lambda^n), \tag{3.52}$$

$$\nabla_{k_y} J = \nabla_{k_y}\beta + \sum_{n=1}^{L}[\nabla_{k_y}(f^n)^T](\lambda^n), \tag{3.53}$$

$$\nabla_{k_z} J = \nabla_{k_z}\beta + \sum_{n=1}^{L}[\nabla_{k_z}(f^n)^T](\lambda^n), \tag{3.54}$$

and

$$\nabla_{\phi} J = \nabla_{\phi}\beta + \sum_{n=1}^{L}[\nabla_{\phi}(f^n)^T](\lambda^n), \tag{3.55}$$

where β is p_{wf}, GOR, WOR at some specified time step L, the whole data mismatch part of the objective function $O_d(m)$ or any other terms for which we wish to calculate sensitivities.

In order to calculate the gradient of the objective function, we consider β as the whole data mismatch part of the objective function, i.e.,

$$\beta = O_d(m) = \frac{1}{2}(g(m) - d_{\text{obs}})^T C_D^{-1}(g(m) - d_{\text{obs}}), \tag{3.56}$$

or in the case of stochastic simulation of m,

$$\beta = O_d(m) = \frac{1}{2}(g(m) - d_{\text{uc}})^T C_D^{-1}(g(m) - d_{\text{uc}}). \tag{3.57}$$

Thus, we have

$$\begin{aligned}
\nabla_{y^n}\beta &= \nabla_{y^n}\{\frac{1}{2}(g(m) - d_{\text{obs}})^T C_D^{-1}(g(m) - d_{\text{obs}})\} \\
&= \left[\nabla_{y^n}(g(m) - d_{\text{obs}})^T\right] C_D^{-1}(g(m) - d_{\text{obs}}) \\
&= \nabla_{y^n}[g(m)]^T C_D^{-1}(g(m) - d_{\text{obs}}).
\end{aligned} \tag{3.58}$$

In the case of β given by Eq. 3.57, the d_{obs} in Eq. 3.58 should be replaced by d_{uc}. The matrix $\nabla_{y^n}[g(m)]^T$ is an $N_e \times N_d$ matrix and defined as

$$\nabla_{y^n}[g(m)]^T = \begin{bmatrix}
\frac{\partial g_1}{\partial p_1^n} & \frac{\partial g_2}{\partial p_1^n} & \cdots & \frac{\partial g_{N_d}}{\partial p_1^n} \\
\frac{\partial g_1}{\partial S_{w,1}^n} & \frac{\partial g_2}{\partial S_{w,1}^n} & \cdots & \frac{\partial g_{N_d}}{\partial S_{w,1}^n} \\
\frac{\partial g_1}{\partial S_{g,1}^n} & \frac{\partial g_2}{\partial S_{g,1}^n} & \cdots & \frac{\partial g_{N_d}}{\partial S_{g,N}^n} \\
\frac{\partial g_1}{\partial p_2^n} & \frac{\partial g_2}{\partial p_2^n} & \cdots & \frac{\partial g_{N_d}}{\partial p_2^n} \\
\vdots & \vdots & \cdots & \vdots \\
\frac{\partial g_1}{\partial S_{g,N}^n} & \frac{\partial g_2}{\partial S_{g,N}^n} & \cdots & \frac{\partial g_{N_d}}{\partial S_{g,N}^n} \\
\frac{\partial g_1}{\partial p_{wf,1}^n} & \frac{\partial g_2}{\partial p_{wf,1}^n} & \cdots & \frac{\partial g_{N_d}}{\partial p_{wf,1}^n} \\
\vdots & \vdots & \cdots & \vdots \\
\frac{\partial g_1}{\partial p_{wf,N_w}^n} & \frac{\partial g_2}{\partial p_{wf,N_w}^n} & \cdots & \frac{\partial g_{N_d}}{\partial p_{wf,N_w}^n}
\end{bmatrix}. \tag{3.59}$$

The entries of vector $g(m)$ represent production data. The vector may contain entries like p_{wf}, GOR and WOR or any combination of these three kinds of production data. Details for calculating each entry of matrix $\nabla_{y^n}[g(m)]^T$ can be found in Li (2001). It turns out many columns of this matrix are zero. Only the columns corresponding to data that are measured at time n are nonzero. After we evaluate the matrix $\nabla_{y^n}[g(m)]^T$, we multiply $C_D^{-1}(g(m) - d_{\text{obs}})$ by this matrix to obtain $\nabla_{y^n}\beta$. Once we have $\nabla_{y^n}\beta$, we can apply Eq. 3.39 to compute the adjoint variables.

To apply Eq. 3.45 to compute the derivatives, we need to evaluate $\nabla_m\beta$

first. The vector $\nabla_m \beta$ is given by

$$\begin{aligned}
\nabla_m \beta &= \nabla_m O_d(m) \\
&= \nabla_m \{ \frac{1}{2} (g(m) - d_{\text{obs}})^T C_D^{-1} (g(m) - d_{\text{obs}}) \} \\
&= \left[\nabla_m (g(m) - d_{\text{obs}})^T \right] C_D^{-1} (g(m) - d_{\text{obs}}) \\
&= \nabla_m [g(m)]^T C_D^{-1} (g(m) - d_{\text{obs}}).
\end{aligned} \tag{3.60}$$

In the case of β given by Eq. 3.57, the d_{obs} in Eq. 3.60 should be replaced by d_{uc}. The matrix $\nabla_m [g(m)]^T$ is an $N_m \times N_d$ matrix and defined as

$$\nabla_m [g(m)]^T = \begin{bmatrix} \frac{\partial g_1}{\partial m_1} & \frac{\partial g_2}{\partial m_1} & \cdots & \frac{\partial g_{N_d}}{\partial m_1} \\ \frac{\partial g_1}{\partial m_2} & \frac{\partial g_2}{\partial m_2} & \cdots & \frac{\partial g_{N_d}}{\partial m_2} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial g_1}{\partial m_{N_m}} & \frac{\partial g_2}{\partial m_{N_m}} & \cdots & \frac{\partial g_{N_d}}{\partial m_{N_m}} \end{bmatrix}. \tag{3.61}$$

The vector $g(m)$ is the calculated production data vector. For the history matching problems considered here, an entry of g will correspond to p_{wf}, GOR or WOR. The formulas for calculation of elements in the matrix $\nabla_m [g(m)]^T$ can be found in Li (2001). After we compute $\nabla_m \beta$, we can use Eq. 3.45 to compute the derivatives of the objective function with respect to model parameters, i.e., the gradient of the objective function.

CHAPTER IV

LINEAR EQUATION SOLVERS

In the automatic history matching procedure, we need to repeatedly solve adjoint equations given by Eq. 3.39 either to compute the gradient of the objective function or to form the sensitivity coefficient matrix. Hence, the computational efficiency of solving the adjoint equations plays a dominant role in the computational efficiency of the overall history matching procedure. Solving an adjoint equation problem is equivalent to solving a system of linear finite-difference equations backwards in time. At each time step, we must solve a linear system of equations. The linear equation solver used in the previous code developed by Ruijian Li is the Harwell sparse matrix solver based on a direct method. Direct methods require far too much computer memory to be useful for field scale history matching problems. On the other hand, iterative solvers are far more efficient than direct method for large scale problems. In this work, an iterative solver was implemented in our history matching code. The iterative method we used to solve the adjoint system is effectively the same iterative solver used in the CLASS (Chevron's Limited Application Simulation Systems) simulator. The iterative solver is based on the orthomin technique; see Vinsome (1976). An incomplete LU decomposition (see, for example, Todd Dupont and Rachford (1968), Axelsson and Gustafsson (1980) and Meijerink (1981)) was used to generate the preconditioner for the purpose of accelerating the convergence.

For simplicity, the adjoint equation system at each time step is written as

$$Ax = b, \tag{4.1}$$

where

$$A = \nabla_{y^n}(f^n)^T, \tag{4.2}$$

$$x = \lambda^n, \tag{4.3}$$

and

$$b = -[\nabla_{y^n}(f^{n+1})^T]\lambda^{n+1} - \nabla_{y^n}\beta. \tag{4.4}$$

at any time step n. A theoretical discussion of iterative solvers is given in Appendix A. As noted above, the iterative solver implemented in our code is based on the orthomin technique. The algorithm for applying orthomin to solve Eq. 4.1 is given below.

- ⋆ Choose an initial guess x_0.
- ⋆ Set $r_0 = b - Ax_0$.
- ⋆ Solve $M\delta x_1 = r_0$ for δx_1 where M is an approximation to matrix A which is chosen such that this linear equation can be solved easily and set $p_1 = \delta x_1$ and calculate

$$a_1 = \frac{(r_0, Ap_1)}{(Ap_1, Ap_1)} \tag{4.5}$$

$$x_1 = x_0 + a_1 p_1 \tag{4.6}$$

$$r_1 = r_0 - a_1 Ap_1 \tag{4.7}$$

- ⋆ Iteration loop

  DO $k = 1, 2, \cdots$

$$\delta x_{k+1} = M^{-1} r_k \tag{4.8}$$

$$b_j = -\frac{(A\delta x_{k+1}, Ap_j)}{(Ap_j, Ap_j)}, \quad j = 1, 2, \cdots, k \tag{4.9}$$

$$p_{k+1} = \delta x_{k+1} + \sum_{j=1}^{k} b_j p_j \tag{4.10}$$

$$a_{k+1} = \frac{(r_k, Ap_{k+1})}{(Ap_{k+1}, Ap_{k+1})} \tag{4.11}$$

$$x_{k+1} = x_k + a_{k+1} p_{k+1} \tag{4.12}$$

$$r_{k+1} = r_k - a_{k+1} Ap_{k+1} \tag{4.13}$$

  END DO

We call this algorithm orthomin. In this algorithm, p_{k+1} denotes the search direction vector at iteration $k+1$ and x_{k+1} denotes the $(k+1)$st approximation to the solution of Eq. 4.1. Note that in Eq. 4.10, all the previous search direction vectors are used to construct the current search direction. If we just use a limited number of previous search direction vectors to construct p_{k+1}, we call the corresponding version of the algorithm the truncated orthmin method. If we use l previous vectors, then the only change is that Eq. 4.10 is replaced by

$$p_{k+1} = \delta x_{k+1} + \sum_{j=k+1-l}^{k} b_j p_j. \tag{4.14}$$

(If $l = 1$ and M is the identity matrix, then the algorithm is referred to as orthomin (2); see Appendix A). Note that in Eq. 4.8, the value of δx_{k+1} is obtained by solving

$$M\delta x_{k+1} = r_k, \tag{4.15}$$

instead of forming M^{-1} and the matrix product $M^{-1}r_k$. Recall that M can be considered to be a preconditioning matrix which is an approximation to the coefficient matrix A. The key issue for implementation of the orthomin algorithm is how to choose M such that Eq. 4.15 can be solved very efficiently. In our implementation, an incomplete LU decomposition of A was applied. This entails a decomposition of the form $A = LU - R$ where L is lower triangular, U is upper triangular and R is the residual or error in the decomposition. L and U are typically chosen to be sparse and have a simple structure. If L and U, respectively, are required to have the same nonzero structure as the lower and upper triangular of A, then incomplete LU decomposition is known as ILU(0) or called the level-0 fill-in incomplete LU decomposition. The fill-in refers to nonzero elements of L and U that occurs at locations where the corresponding elements of A are zero. To improve the rate of convergence, more fill-in in L and U are allowed to develop higher level fill-in incomplete decompositions. In general, we expect that more accurate ILU decompositions require fewer iterations to converge, but the preprocessing cost to compute such factors is higher. Meijerink and van der Vorst (1977) provided a theoretical basis for the incomplete decomposition. Let P represent the set of the locations in the matrix A where the

corresponding entries are zero. The algorithm can be written as

DO $k = 1, ..., n-1$

DO $i = k+1, n$ and if $(i,k) \ni P$

$a_{ik} = a_{ik}/a_{kk}$.

DO $j = k+1, \cdots, n$ and if $(i,j) \ni P$

$a_{ij} = a_{ij} - a_{ik} * a_{kj}$.

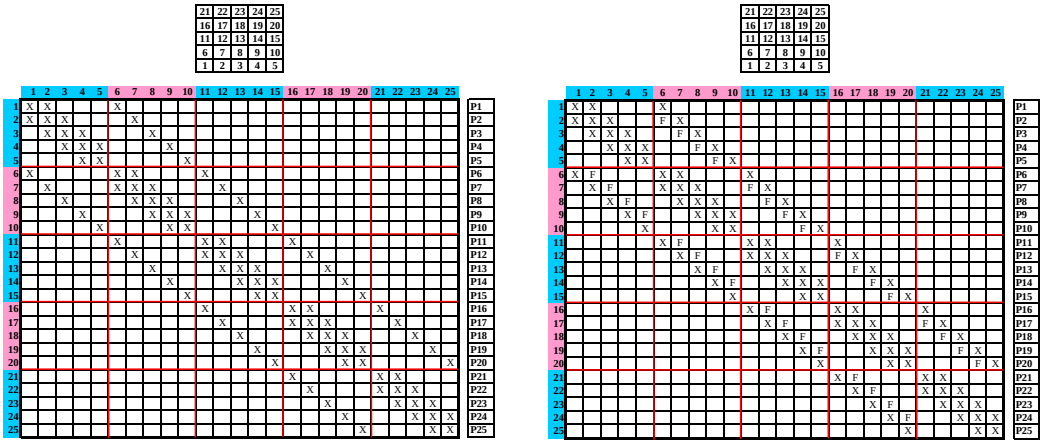
ENDDO

ENDDO

ENDDO

In the above algorithm, $\ni$ means "not belong to".

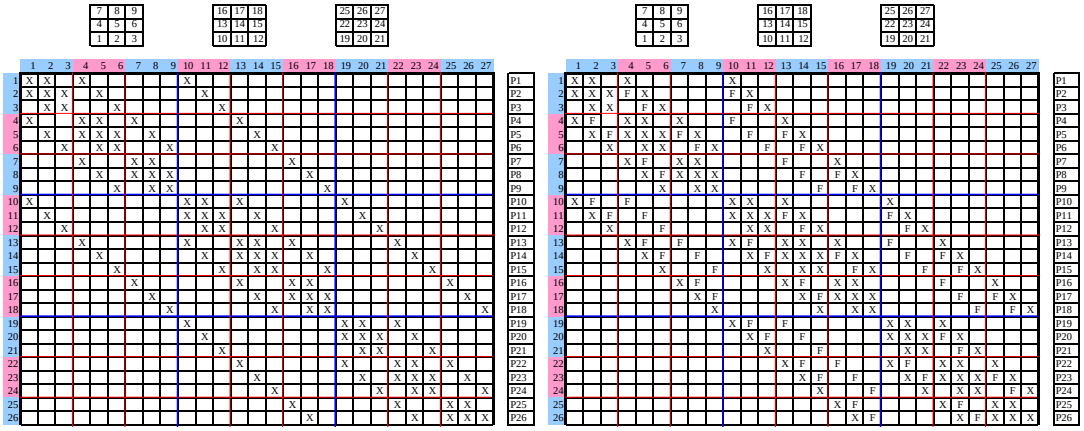
In our application, level-1 fill-in was applied. For example, the standard 5 diagonal matrix (Fig. 4.1(a)) becomes a 7 diagonal matrix (Fig. 4.1(b)) after level-1 fill-in; the standard 7 diagonal matrix (Fig. 4.2(a)) becomes a 13 diagonal matrix (Fig. 4.2(b)) after level-1 fill-in. In all these 4 figures, an "X" in a cell represents an original nonzero entry and an "F" in a cell represents a fill-in element.



(a) Level 0 fill-in

(b) Level 1 fill-in

Fig. 4.1: Matrix structure of 2D (5x5) single-phase flow equation.



(a) Level 0 fill-in

(b) Level 1 fill-in

Fig. 4.2: Matrix structure of 3D (3x3x3) single-phase flow equation.

4.1 Comparison of the Iterative Solver with the Sparse Matrix Solver

A two-dimensional three-phase history matching problem was considered for the purpose of comparison of the iterative solver with the sparse matrix solver. We use a 15×15 grid with $\Delta x = \Delta y = 40$ ft and $\Delta z = 30$ ft. The porosity for the true model is homogeneous and equal to 0.22. Permeability is isotropic and uniform in three different zones; see Fig. 4.3. The value of $\ln(k)$ in the lower left zone, lower right zone and the upper half zone are equal to 4.0, 4.6 and 4.2 respectively for k in md. Four producers and one water injection well are completed in the reservoir. The well locations are indicated by the white squares in Fig. 4.3. All producers start production at time 0 at a constant total flow rate of 200 STB/Day and produce for 300 days. Bottom-hole pressure from all five wells, GOR and WOR from all four producers are used as conditioning data to estimate the gridblock permeabilities only, i.e., the porosity is fixed at its true values. A total of 364 production data are history matched, 28 data of each of the three types at the four producing wells and 28 pressure data at the water injection well. An isotropic spherical variogram with the range equal to 240 ft in all three directions and the variance for $\ln(k)$ equal to 1 was used to construct the prior covariance matrix for log-permeability. Note

the true case does not exhibit the type of permeability heterogeneity that would be typical of an unconditional realization that would be generated from the prior model. The maximum a posteriori (MAP) estimate was obtained by history matching the production data. The Levenberg-Marquardt algorithm was applied to minimize the objective function involved in the history matching procedure. The truncated orthmin algorithm using five previous vectors with level-1 fill-in was applied to solve the adjoint equations involved in the computation of the sensitivity coefficient matrix. The iteration was stopped when the following condition is satisfied

$$\frac{\| r_k \|_\infty}{\| r_0 \|_\infty} \leq \varepsilon = 10^{-4}, \tag{4.16}$$

where $\| \cdot \|_\infty$ denotes the infinity or maximum norm.

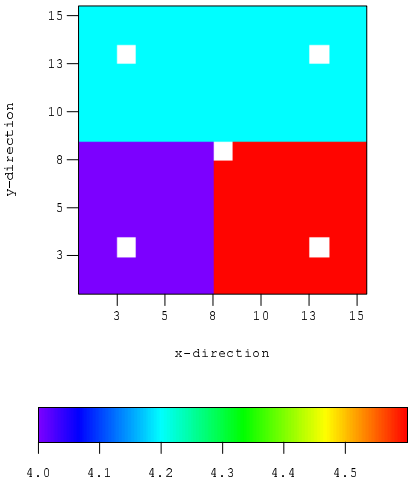


Fig. 4.3: Permeability field for the true model.

To apply the iterative solver, we have to provide an initial guess for the vector of adjoint variables, λ. Often, we did not obtain the same solution obtained with the Harwell sparse matrix solver (a direct solver based on LU decomposition) if we used an arbitrary initial guess. For example, when solving the adjoint equation systems corresponding to the time step at which we have wellbore pressure data as the conditioning data, we did not get the correct solution in 100 iterations when we

used 0, 1, 10 or 100 as the initial guess for all components of λ. Fig. 4.4 shows the Harwell sparse matrix solver solution to the adjoint equation system for the case where we wish to obtain the sensitivity to $p_{\mathrm{wf},1}(t_L)$, the wellbore pressure at well 1 at time t_L. The solution for the λ's represents the results obtained by doing a single time step backward in time to obtain the adjoint variables $\lambda_j(t_L)$ where the subscript $j = 1, 2, \cdots, N_e$. The x-coordinate in Fig. 4.4 represents the adjoint equation index. We have $15 \times 15 = 225$ gridblocks. Each gridblock has three equations corresponding to three phases. In addition, we have 5 equations corresponding to 5 wells. Thus, the total number of adjoint variables is 680. For the 680 equations, only the 676th equation (which corresponds to the first well equation) has a nonzero right-hand side, all other equations have zero right-hand sizes. Most of the λ's are on the order of 10^9 in amplitude. The biggest value corresponds to the 676th λ. The adjoint solutions corresponding to the well gridblock equations are bigger than the adjoint solutions corresponding to the equations for the gridblocks surrounding the well gridblocks. Since, all the λ's are large, setting all λ's equal to a small value provides a poor initial guess.

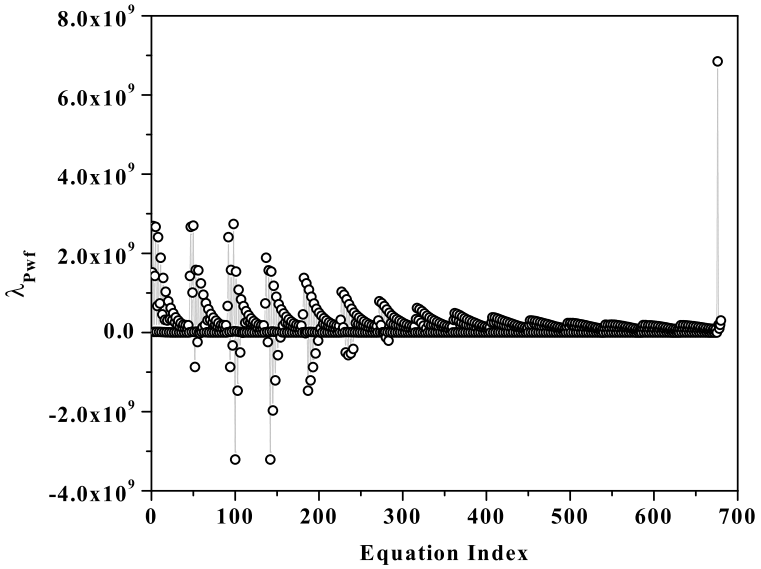


Fig. 4.4: Adjoint solution from Harwell solver, the source term from $p_{\mathrm{wf},1}(t_L)$.

Figs. 4.5 and 4.6 show the Harwell solver solution to the adjoint equation

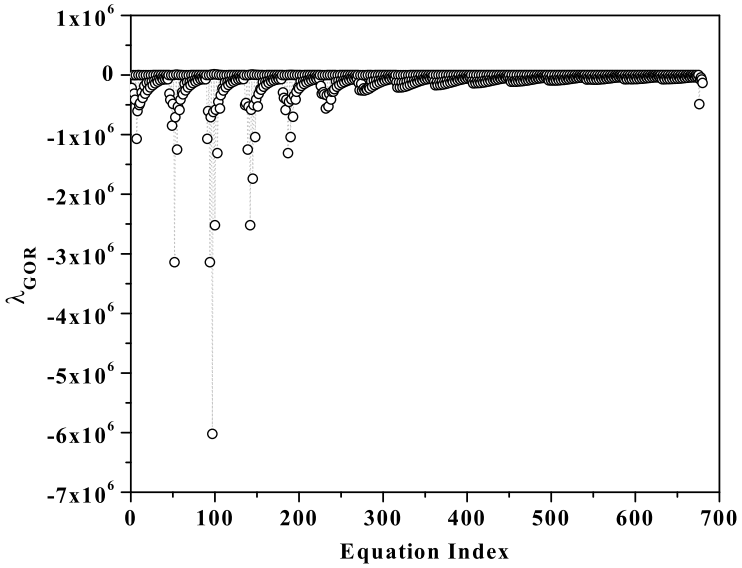


Fig. 4.5: Adjoint solution from Harwell solver, source term from a GOR datum.

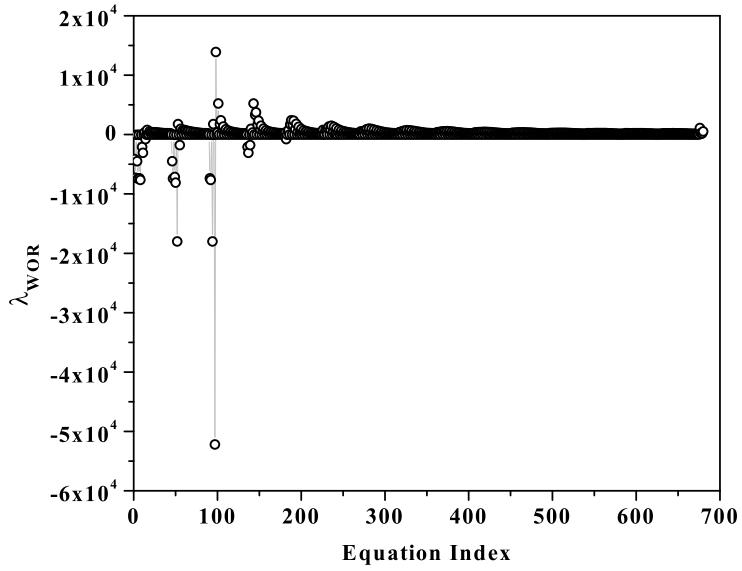


Fig. 4.6: Adjoint solution from Harwell solver, source term from a WOR datum.

systems associated with GOR and WOR as source terms respectively. These adjoint solutions are corresponding to all the forward simulation equations at a single time step where we have measured data. In checking the Harwell solutions carefully, we found that almost all the solutions for the adjoint equation system associated with GOR are negative. We also found that the solutions to the adjoint equation system

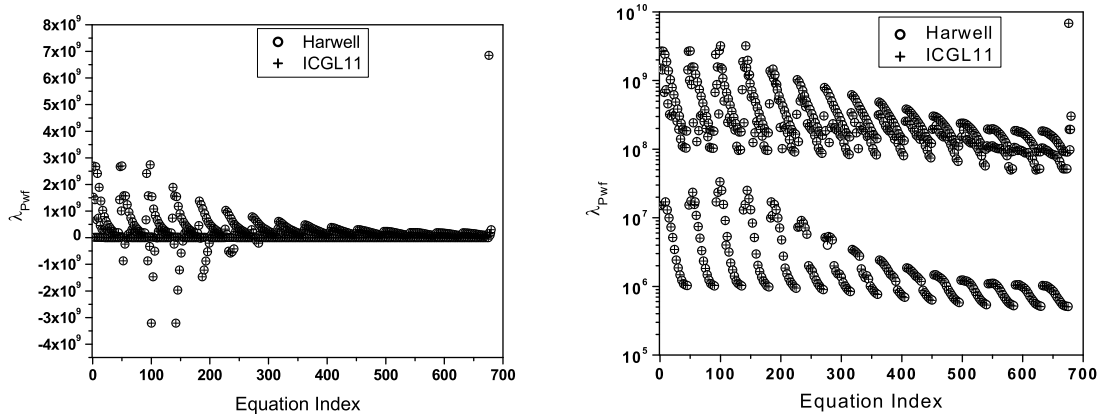
associated with WOR are relatively small in amplitude with some positive and some negative.

Based on the features of the correct adjoint solution (Figs. 4.4 through 4.6), we provide an ad hoc procedure for generating the initial guess. The equation for solving the λ's at the first step backward in time has the form

$$a_{i,i}\lambda_i^L + \sum_{j \neq i} a_{i,j}\lambda_j^L = s_i \tag{4.17}$$

for $1 \leq i \leq N_a$ where N_a is the number of adjoint variables at each time step. Note that $N_a = N_e$. If the source term, s_i, is not zero, we set the initial guess for λ_i^L equal to $s_i/a_{i,i}$. For equations with $s_i = 0$, we set the initial guess for λ_i equal to 1000 if we solve for λ's to compute the sensitivities of p_{wf} to model parameters, to -100 if we compute GOR sensitivities and to 0 if we are computing WOR sensitivities. After the first step of solving the adjoint equation system backward in time, we use the resulting solution as the initial guess when solving the adjoint system at the next time step backward and repeat this procedure for all the subsequent time steps. For simplicity, this iterative solver is called iterative solver with initial guess scheme 1. Fig. 4.7 shows the adjoint solutions for the equation systems when p_{wf} from well 1 was used as data to generate the source term. In this figure and similar figures, the circles represent the Harwell solver solution and the plus signs represent the iterative solver solution. In the semilog plot, we plotted the absolute value of the λ's. We can see that the iterative solver solution matched the Harwell solver solution very well.

In addition, we can see that the semilog plot has a very beautiful pattern. The calculated adjoint variables fall into three big groups. These three groups of adjoint solutions correspond to the three types of equation, i.e., oil equation (top group), gas equation (middle group) and water equation (bottom group). Note that although it is not so obvious, the middle group is underneath the top group. Each big group contains 15 small groups of adjoint solutions. Each small group corresponds to a specific row of gridblocks; recall that the grid system is 15×15. Each small group contains 15 points. Each point represents the adjoint solution corresponding to one equation at a gridblock of a certain row. The last 5 points represent the adjoint



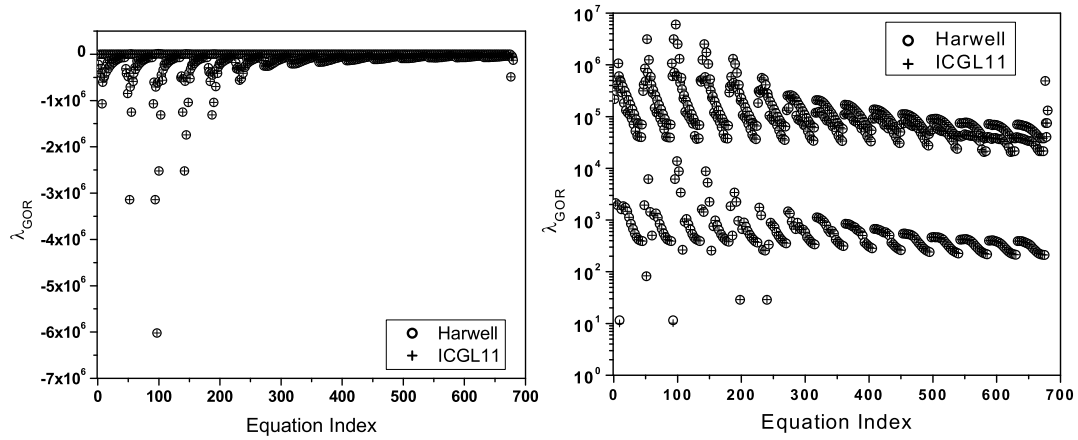
(a) Linear Plot (b) Semilog Plot

Fig. 4.7: Adjoint solution obtained from iterative solver with initial guess scheme 1, source term from a p_{wf} datum.

solutions corresponding to the 5 well equations respectively. We also can see that the adjoint solution corresponding to the well at which the datum was measured has the biggest amplitude. We also can see that an adjoint variable corresponding to a gridblock that is closer to the well gridblock has a bigger amplitude than the adjoint variable corresponding to a gridblock that is further away from the well gridblock.

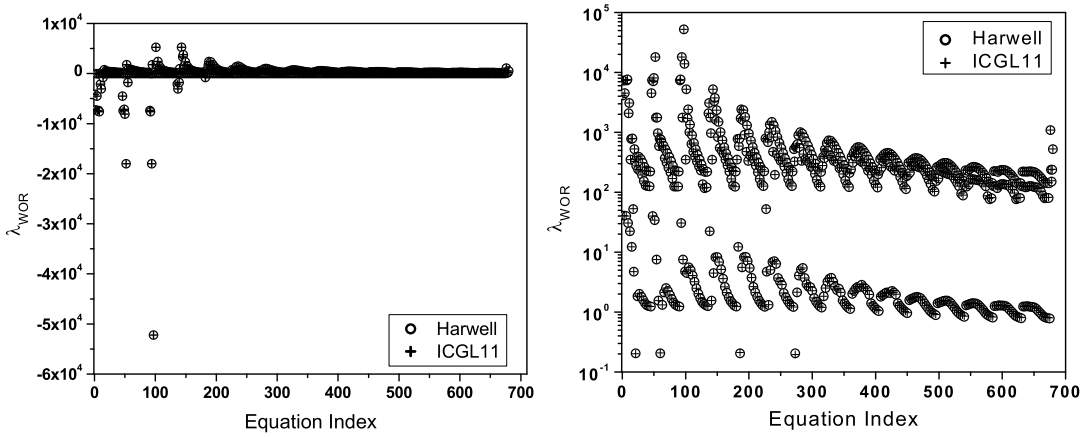
Figs. 4.8 and 4.9, respectively, show the adjoint solutions when GOR and WOR from well 1 was used to generate the source term. For both cases, the iterative solver solution matched the Harwell solver solution very well. The pattern observed in Fig. 4.7 can also be observed in Figs. 4.8 and 4.9.

Above, we provided an ad hoc procedure to generate an initial guess for the adjoint variables for different types of source terms. This procedure is used for the first time step (backward) associated with the generation of the sensitivity of a particular production datum to model parameters. Due to the fact that the adjoint equations are solved backward in time, after we solve the adjoint equation system corresponding to the time at which we have conditioning data, we use these values of λ as the initial guess to solve the adjoint equation at the next time step and repeat this process for all the subsequent time steps. Figs. 4.10 through 4.12 show the plot



(a) Linear Plot (b) Semilog Plot

Fig. 4.8: Adjoint solution obtained from iterative solver with initial guess scheme 1, source term from a GOR datum.



(a) Linear Plot (b) Semilog Plot

Fig. 4.9: Adjoint solution obtained from iterative solver with initial guess scheme 1, source term from WOR.

of the adjoint solution for the λ's corresponding to the first well equation versus time when p_{wf}, GOR and WOR were used as the source terms, respectively. We can see that the changes are not dramatic except at the time steps very close to the time at which we have a source term. Note that in Fig. 4.10 the pressure datum was

measured at the 1st time step backward, in Fig. 4.11 the GOR datum was obtained at the 4th time step backward and in Fig. 4.12 the WOR datum was obtained at the 4th time step backward.

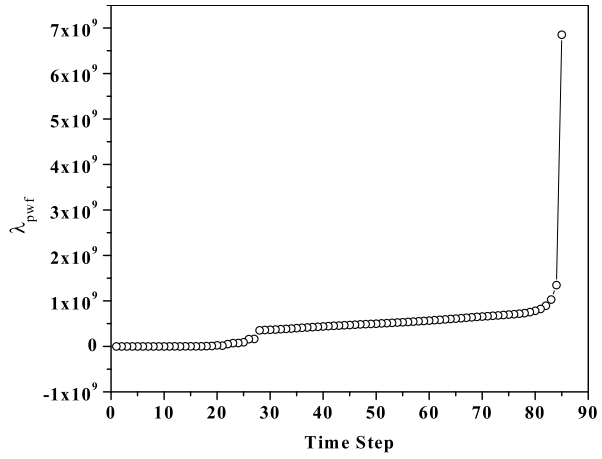


Fig. 4.10: Adjoint variables corresponding to the well equation versus time, source term from p_{wf}.

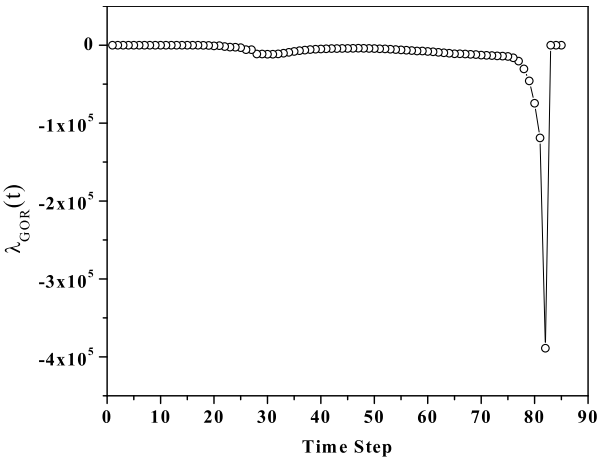


Fig. 4.11: Adjoint variables corresponding to the well equation versus time, source term from GOR.

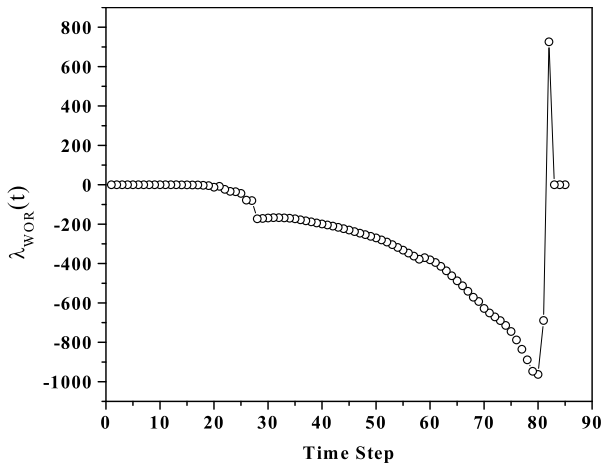
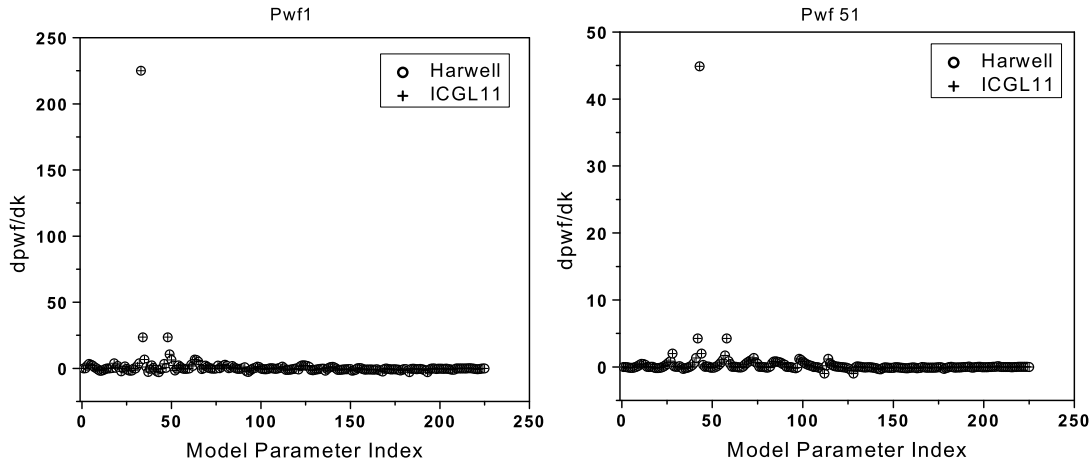


Fig. 4.12: Adjoint variables corresponding to the well equation versus time, source term from WOR.

Fig. 4.13 (a) and (b), respectively, show the sensitivities of the first pressure data (the first pressure data at the first well) and the 51st pressure data (the 23rd pressure data at the second well) to all gridblock log-permeabilities. Fig. 4.14 shows the sensitivities of one GOR data point (1st GOR at well 1) to the gridblock log-permeabilities and Fig. 4.15 shows the sensitivities of one WOR data point (1st WOR at well 1) to the gridblock log-permeabilities. We can see that results obtained by the two solvers are in good agreement in Figs. 4.13 through 4.15.

The results shown in Fig. 4.16 represent MAP estimates of the log-permeability field obtained by history matching the production data. In generating the MAP estimate of log-permeability shown in Fig. 4.16 (a), all adjoint solutions were obtained using the Harwell solver, whereas the result of Fig. 4.16 (b) pertain to the case where the iterative solver was used to solve the adjoint equations. We see that similar results were obtained for both cases and both of them are similar to the true model. As shown in Fig. 4.17, the convergence behavior of the Levenberg-Marquardt algorithm was similar for the two cases, where Iterative 1 referred to using the iterative solver with the initial guess generated by the method discussed above. The results labeled Iterative 2 will be discussed later.

For this example, only pressure data was matched at the injection well. At



(a) p_{wf} 1 (b) p_{wf} 51

Fig. 4.13: Sensitivity of p_{wf} to permeability, iterative solver with initial guess scheme 1.

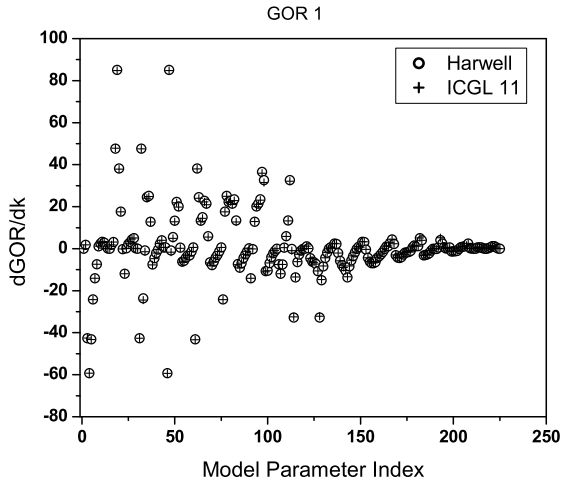


Fig. 4.14: Sensitivity of GOR to permeability, iterative solver with initial guess scheme 1.

the producers, pressure, WOR and GOR data were matched. The observed, initial and pressure data predicted by the MAP estimate for well 2 and well 5 are shown in Fig. 4.18 (a) and (b). In this figure and in similar figures, the circles represent the observed data, the diamonds represent the data calculated from the initial model

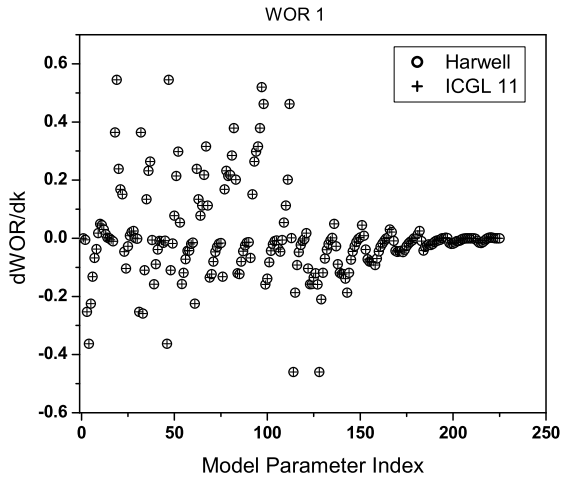
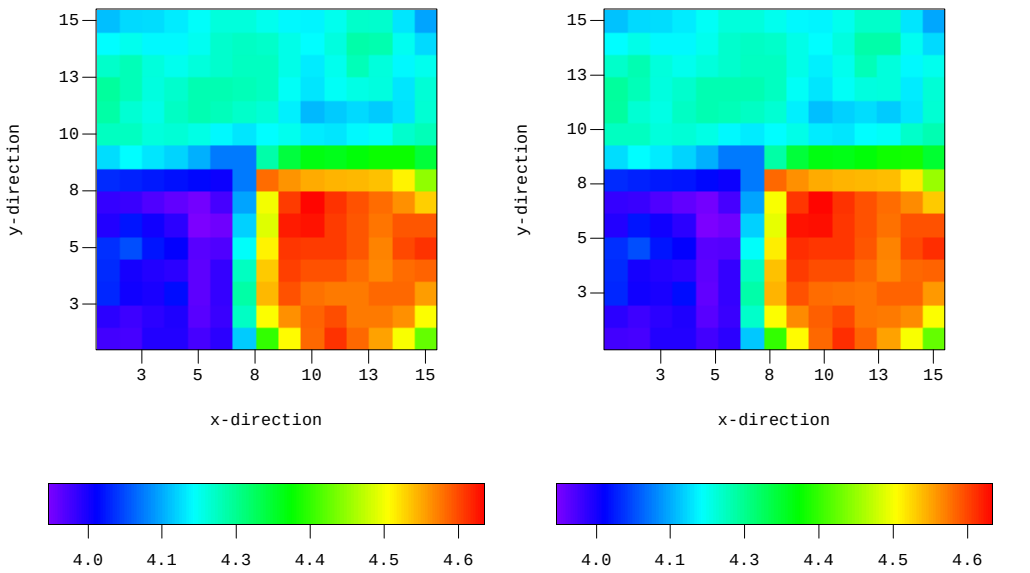


Fig. 4.15: Sensitivity of WOR to permeability, iterative solver with initial guess scheme 1.



(a) Harwell solver (b) Iterative solver with initial guess scheme 1

Fig. 4.16: Final permeability model.

and the plus signs represent the data calculated from the final model. The observed, initial and conditioned gas-oil ratio from well 1 and well 2 are shown in Fig. 4.19 (a)

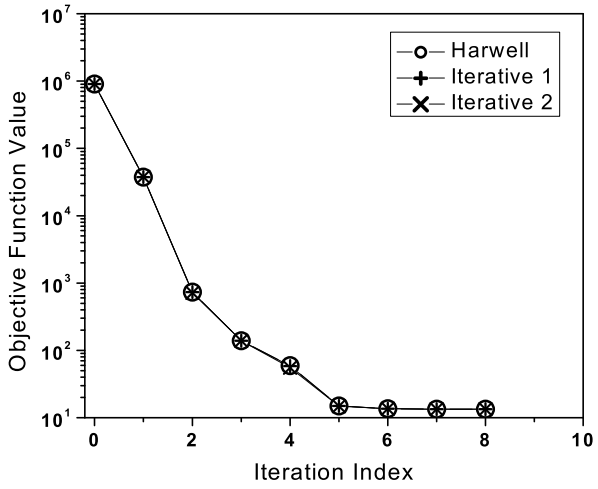
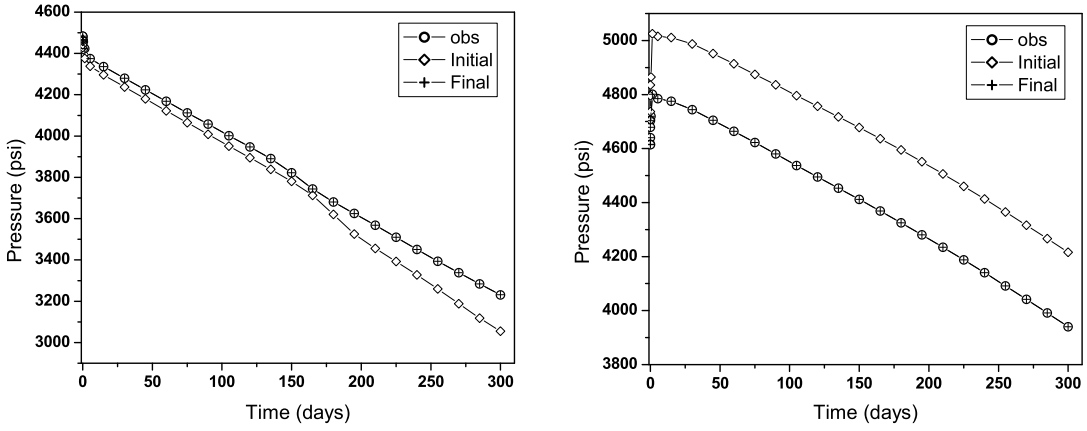


Fig. 4.17: Behavior of the objective function.

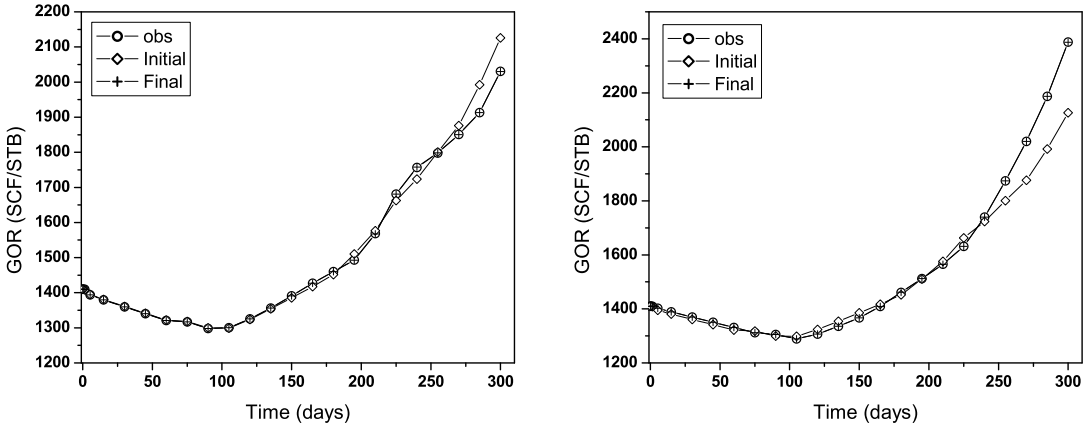
and (b). The observed, initial and water-oil ratio predicted from the MAP estimate at well 1 and well 4 are shown in Fig. 4.20 (a) and (b). From these figures, we can see that we obtain very good matches for all three types of data. Data matches of similar quality were obtained from all wells.



(a) Well 2 (b) Well 5

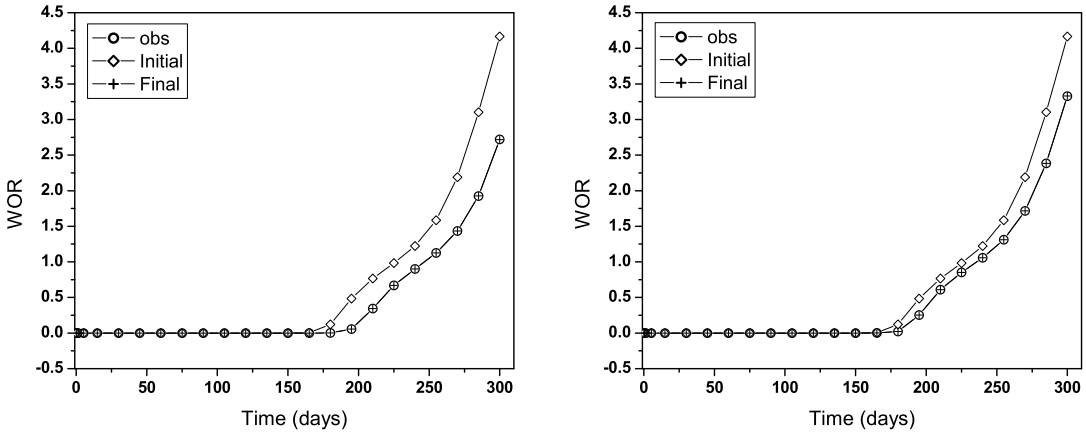
Fig. 4.18: Wellbore pressure match from two wells.

We also considered another scheme to generate the initial guess. In this scheme, we take the summation of all the nonzero coefficients of the source term



(a) Well 1 (b) Well 2

Fig. 4.19: GOR match from two wells.



(a) Well 1 (b) Well 4

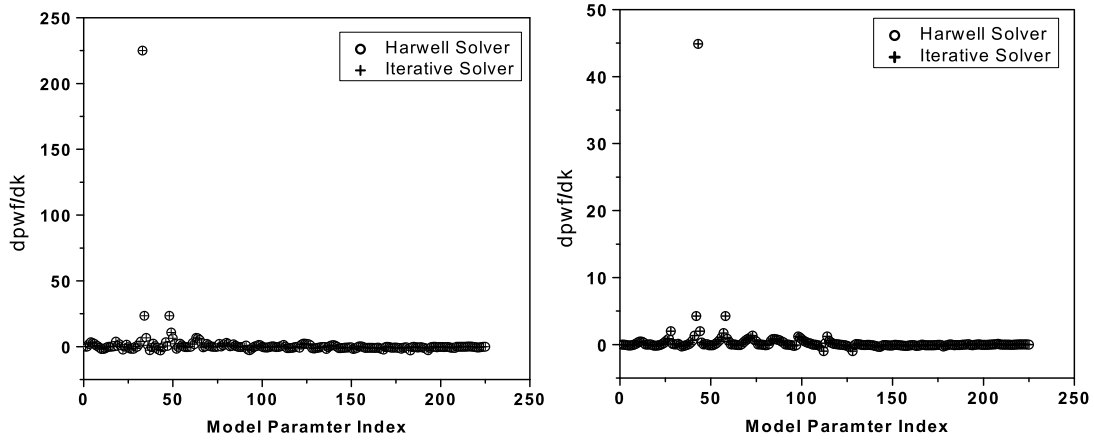
Fig. 4.20: WOR match from two wells.

equation; and divide the source term (right-hand side) by this summation; and then take the quotient as the initial guess for the component of λ corresponding to this equation and use 1000 as the initial guess for all other entries of λ if we solve for λ's to compute the sensitivities of p_{wf} to model parameters. If we calculate the sensitivities of GOR data, we use -100 as the initial guess for all the λ's. If we calculate the sensitivities of WOR, we use 0 as the initial guess for all the λ's. After the first

step of solving the adjoint equation system backward in time, we use the resulting solution as the initial guess when solving the adjoint system at the next time step and repeat this procedure for all the subsequent time steps. The sensitivity of p_{wf} and GOR obtained by this scheme is almost the same as the results obtained by the iterative solver with initial guess scheme 1. But the sensitivities of WOR obtained by this scheme is less accurate than those obtained by the iterative solver with initial scheme 1. Therefore, the scheme for generating initial guess presented previously was used in our code.

The method presented above for generating an initial guess for the adjoint variables is quite ad hoc. If we consider the fact that only a small number of adjoint equation systems are associated with source terms, then we can use the Harwell solver to solve adjoint equations at time steps which have source terms without affecting the overall efficiency very much provided the problem is not so large that computer memory requirements preclude the use of the Harwell solver. Specifically, when we solve the adjoint equations to find the adjoint variables necessary to compute the sensitivity to a particular datum at time t_L, only the adjoint equations for λ_j^L, $1 \leq j \leq N_a$, have nonzero source terms. This linear system corresponds to the first time step backward in time. Initial guess scheme 2 refers to solving for the λ_j^L using the Harwell direct solver then solving subsequent time steps for λ_j^l, $l = L-1, L-2, \cdots, 1$, using the iterative solver. When we solve for λ_j^l, $1 \leq j \leq N_a$, we use the vector of λ_j^{l+1} as the initial guess for λ_j^l, $1 \leq j \leq N_a$. Figs. 4.21 through 4.23 compare the sensitivities of two pressure data, one GOR and one WOR to all the model parameters obtained by the Harwell solver and those obtained by the iterative solver with this initial guess scheme 2. We can see that the iterative solver yields a very good agreement for the sensitivities of all three types of data to model parameters as compared to the corresponding results obtained by using the Harwell solver to solve the adjoint system. The crosses in Fig. 4.17 show the behavior of the objective function when the iterative solver with initial guess scheme 2 was applied. Fig. 4.24 (a) shows the MAP estimate of the model obtained by using the Harwell solver to solve the adjoint system and Fig. 4.24 (b) shows the MAP estimate obtained by using this iterative

solver with initial guess scheme 2 to solve the adjoint system. They agree very well and also are close to the results obtained with initial guess scheme 1. Since initial guess scheme 1 has significantly lower memory requirements, it is preferred.



(a) P_{wf} 1 (b) P_{wf} 51

Fig. 4.21: Sensitivity of p_{wf} to permeability, iterative solver with initial guess scheme 2.

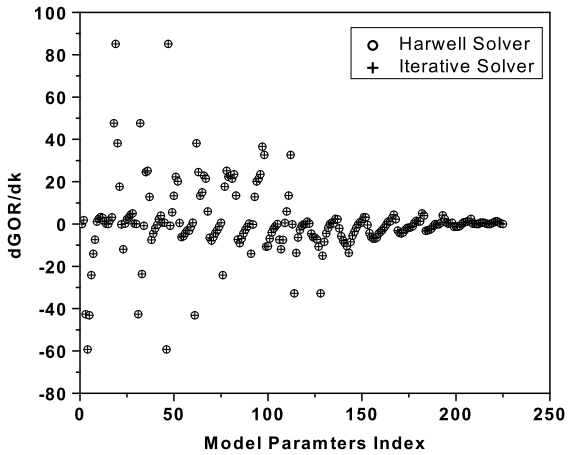


Fig. 4.22: Sensitivity of GOR to permeability, iterative solver with initial guess scheme 2.

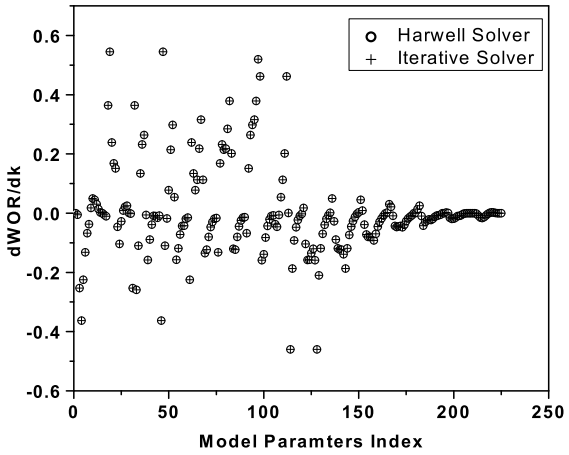
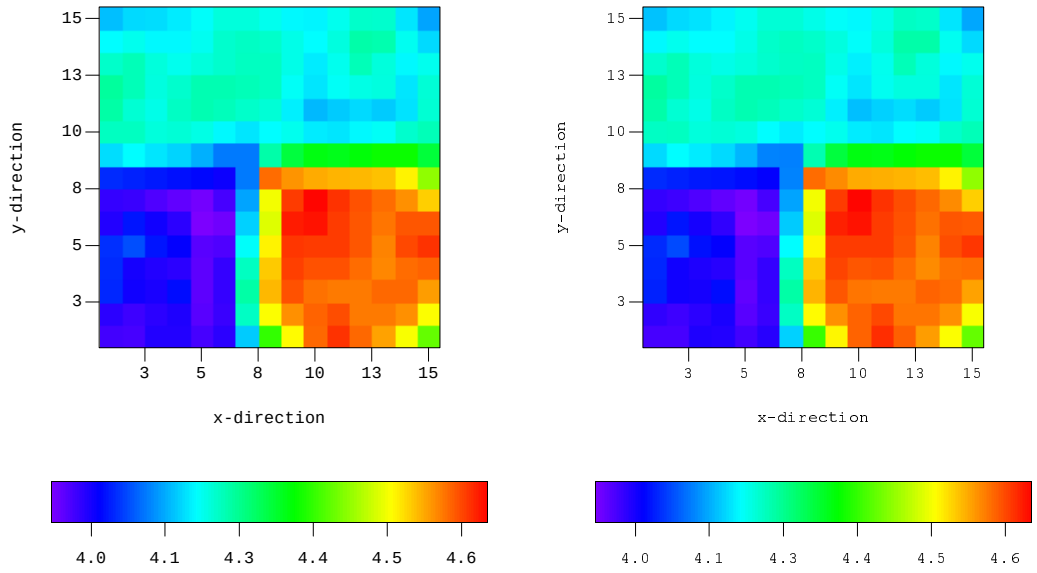


Fig. 4.23: Sensitivity of WOR to permeability, iterative solver with initial guess scheme 2.



(a) Harwell solver

(b) Iterative solver with initial guess scheme 2

Fig. 4.24: Final permeability model.

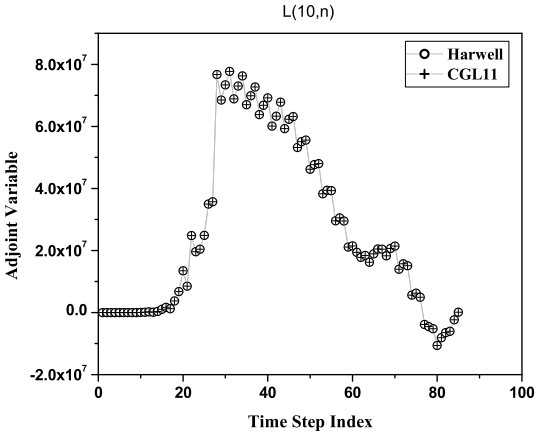
When we calculate the gradient of the objective function, we treat the whole objective function as a single datum. So when we solve the adjoint equation back-

wards, source terms appear at all times corresponding to data mismatch terms in the objective function. In all the applications, we use initial guess scheme 1 to generate the initial guess when calculating the gradient of the objective function. But, here we compare adjoint solutions obtained by this method with results obtained with the Harwell solver. Fig. 4.25 (a) through (c) show the adjoint solution corresponding to the 10th gridblock equation, the first well equation and the last well equation which is an injection well versus time. From Fig. 4.25 (c) we can see clearly that, unlike the case when we only calculate the sensitivity of an individual datum, the adjoint solution changes significantly with time. The observed data set was generated by picking the production data, i.e., p_{wf}, GOR and WOR, every third time step backward when we run the simulator based on the true model given in Fig. 4.3. Fig. 4.25 (c) indicates clearly that the adjoint solutions jump up every third time step backward. Whenever a source term appears on the right-hand side of the adjoint equation system, the adjoint solution jumps up. Note that Fig. 4.25 (c) corresponding to well 5 (the water injection well) is much different from Fig. 4.25 (b) corresponding to well 1 (producing well). Figs. 4.26 through 4.28 show the adjoint solutions at times equal to 300 days, 75 days and 0.001 day. Again, in these figures, circles represent the solution obtained by the Harwell solver and the plus signs represent the solution obtained by the iterative solver. These same results are also shown on the corresponding semilog plot of these figures where we plotted the absolute value of the adjoint variables. We can see that the two sets of results are in very good agreement.

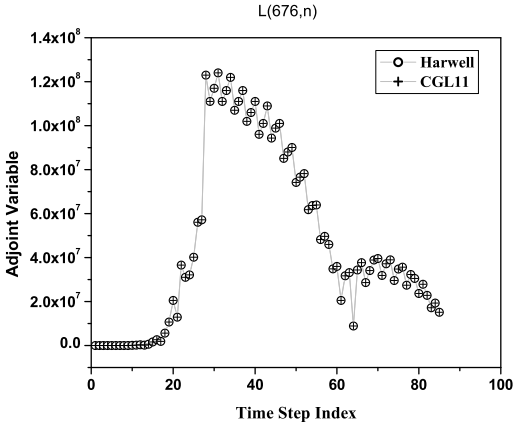
Fig. 4.29 shows the gradient of the objective function constructed by computing the full sensitivity coefficient matrix G, i.e.,

$$\nabla O = C_M^{-1}(m - m_{\text{prior}}) + G^T C_D^{-1}(g(m) - d_{\text{obs}}) \tag{4.18}$$

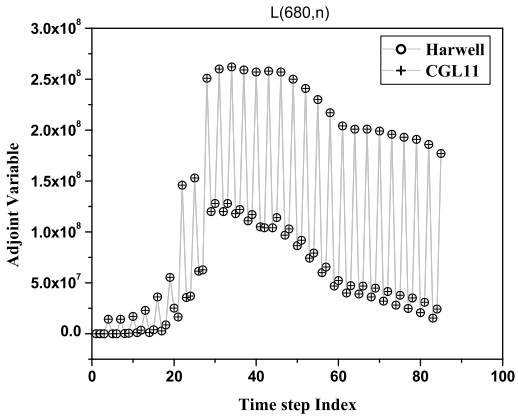
where all individual entries of G were calculated by the adjoint method with the Harwell solver (circles) and the gradient obtained by one application of the adjoint method with adjoint equations solved using the iterative solver (plus signs). We can see a very good agreement is obtained. Fig. 4.30 (a) shows the difference between the two results for elements of ∇O and Fig. 4.30 (b) shows the relative difference,



(a) The 10th gridblock



(b) The first well



(c) The fifth well

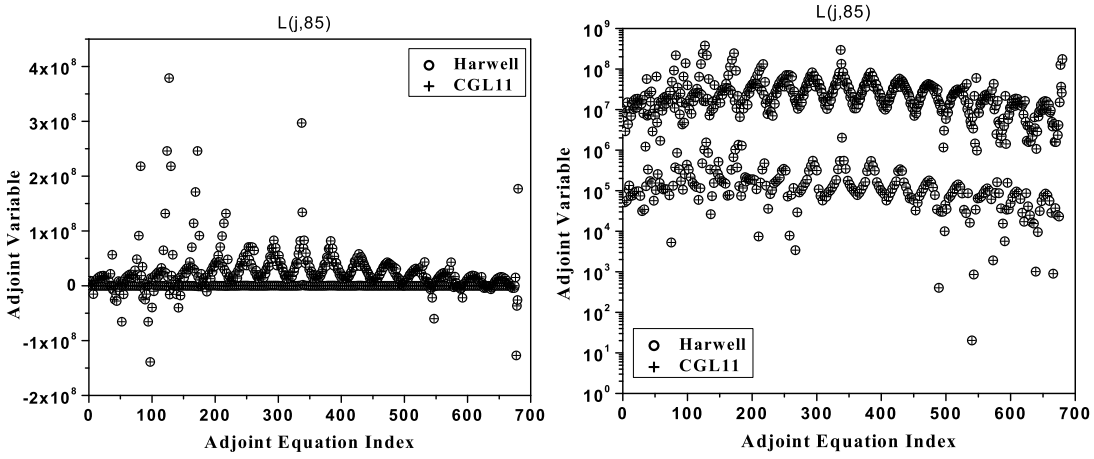
Fig. 4.25: Adjoint solution versus time.

i.e.,

$$\frac{\left(\frac{\partial O}{\partial m_j}\right)_H - \left(\frac{\partial O}{\partial m_j}\right)_{\text{it}}}{\left(\frac{\partial O}{\partial m_j}\right)_H}, \tag{4.19}$$

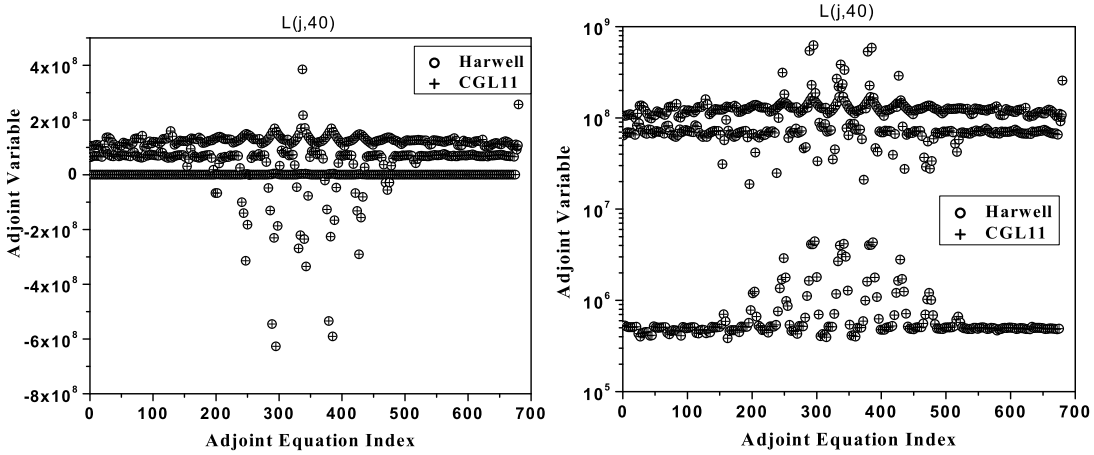
where $\left(\frac{\partial O}{\partial m_j}\right)_H$ and $\left(\frac{\partial O}{\partial m_j}\right)_{\text{it}}$, respectively, represent the derivative of the objective function with respect to the jth model parameters obtained by applying Eq. 4.18 and by one application of the adjoint method.

Fig. 4.31 shows that the gradient of the objective function obtained by the



(a) Linear (b) Semilog

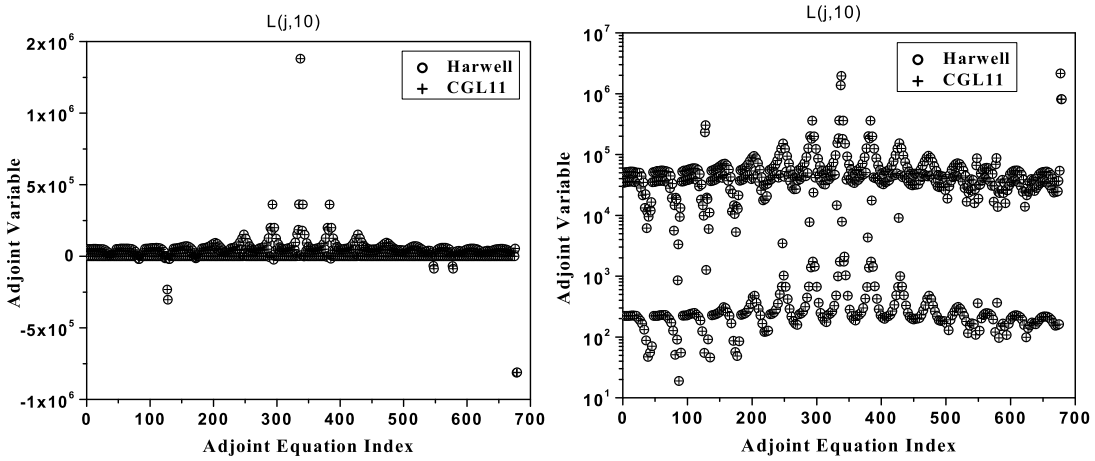
Fig. 4.26: Adjoint solution at 300 days.



(a) Linear (b) Semilog

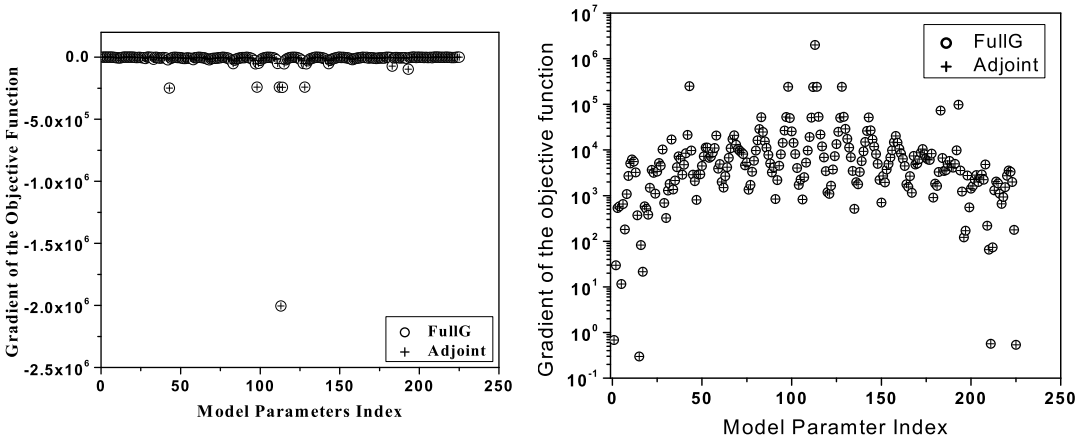
Fig. 4.27: Adjoint solution at 75 days.

finite-difference method (circles) and the gradient obtained by the adjoint method using the iterative solver. The perturbation we used in the finite-difference method was 0.0004 for log-permeability which is 0.01% of the log-permeability value to be perturbed. From the semilog plot, we can see that almost all the derivatives obtained by the adjoint method are in good agreement with the derivatives obtained by the



(a) Linear (b) Semilog

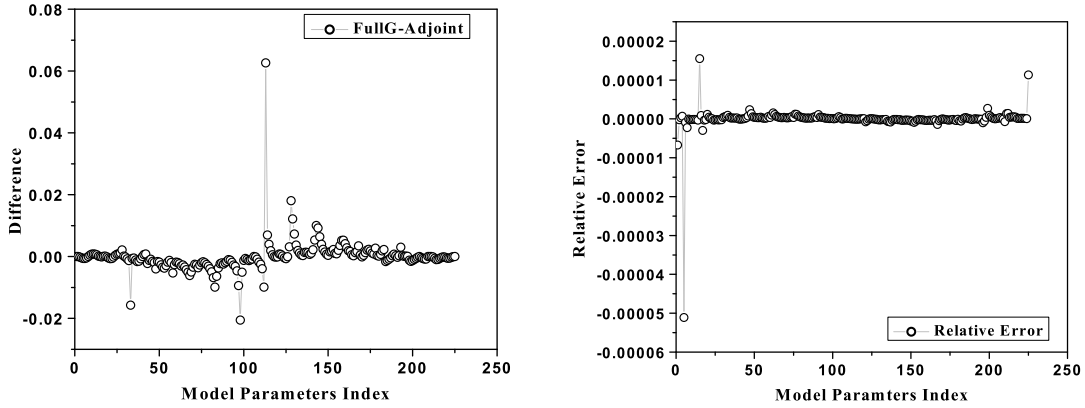
Fig. 4.28: Adjoint solution at 0.001 days.



(a) Linear Plot (b) Semilog Plot

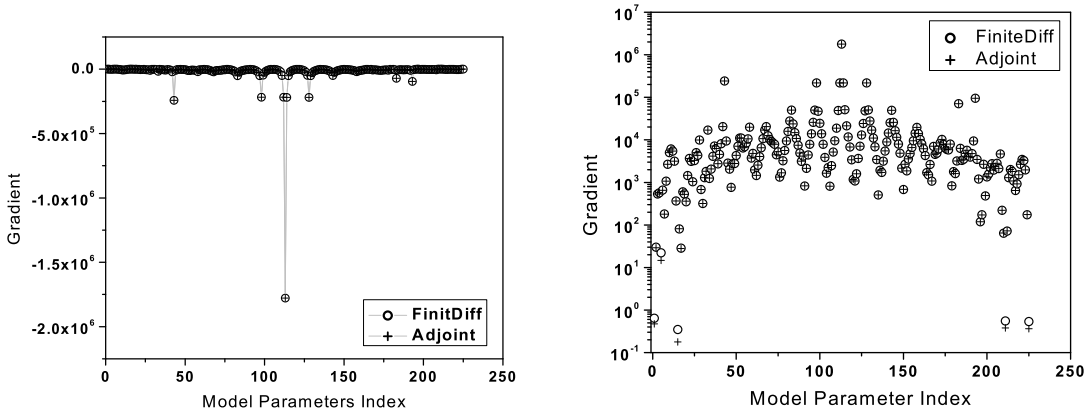
Fig. 4.29: The gradient of the objective function.

finite-difference method. Points where the two methods do not agree well corresponds to derivatives which are relatively small in magnitude. These points are expected to have a very small effect on the history matching process.



(a) Difference (b) Relative difference

Fig. 4.30: Difference between the gradient constructed using the full G with associated adjoint equations solved with the Harwell solver and the gradient obtained by the adjoint method with the adjoint equations solved by the iterative solver.



(a) Linear Plot (b) Semilog Plot

Fig. 4.31: Comparison of the gradient of the objective function obtained by finite difference method and the adjoint method.

CHAPTER V

OPTIMIZATION ALGORITHMS

There are two categories of minimization algorithms for unconstrained optimization problems. One category consists of gradient based algorithms, e.g., steepest descent, Newton, Gauss-Newton, Levenberg-Marquardt, conjugate gradient and variable metric (or quasi-Newton), and the other category includes non-gradient based algorithms, such as simulated annealing (see, for example, Ouenes et al. (1993) and Vasco et al. (1996)), genetic algorithm (see, for example, Sen et al. (1992)), Monte Carlo methods (see, for example, Hegstad et al. (1994) and Bonet-Cunha et al. (1998)) and neutral networks (see, for example, Ouenes et al. (1994)). Unless one can predict production data for a given reservoir model by some method which is orders of magnitude faster than running a conventional finite-difference simulator, non-gradient based algorithms are far too slow for practical application (see, for example, Bonet-Cunha et al. (1998)) as they may require tens of thousands of iterations for convergence, and one must recalculate the predicted production data at each iteration. In our work, predicted data is always generated by running a reservoir simulator, and only gradient based algorithms are considered.

In this chapter, we will discuss gradient based optimization algorithms and some technical details on line search and scaling factors used in quasi-Newton methods. A comparison of the computational efficiency and memory requirement for different algorithms is also provided in this chapter. We consider the problem of minimizing a real-valued function $O(m)$. We assume that O is bounded below and twice continuously differentiable with respect to m. The optimization algorithms for all the gradient based methods are similar and an outline of the methods are described below:

1. Choose an initial guess, m_0, of the model and an initial search direction d_0. Set the iteration index $k = 0$.

2. Calculate the step size α_k along the search direction d_k. Different algorithms use different schemes to find the step size; see Table 5.1.

Table 5.1: Step size calculation scheme used by different algorithms.

Algorithm	Step size calculation scheme
Gauss-Newton	restricted step
Levenberg-Marquardt	damping factor or restricted step
steepest descent	line search
conjugate gradient	line search
quasi-Newton	line search

3. Determine whether or not the stop criterion is satisfied. If it is satisfied then stop. If it is not, then compute the new search direction d_{k+1}, set $k = k + 1$ and then go to 2. Different algorithms employ different search directions.

5.1 Steepest Descent Method

For most problems, this method is relatively inefficient compared to other gradient based methods. Thus, we only provide the basic idea of this method. In this approach, the negative gradient of the objective function is chosen as the search direction. This is the simplest approach. It has been illustrated in the literature that the direction of descent computed with this method is very inefficient; see, for example, Fletcher (1987). Also the procedure has only a linear convergence rate for a quadratic problem; see Greenbaum (1997) or Fletcher (1987).

5.2 Gauss-Newton and Levenberg-Marquardt Algorithms

Let m_k be the most recent estimate of the model m that minimizes $O(m)$. Approximate $O(m)$ by a quadratic expansion about m_k and let

$$\delta m = m - m_k, \tag{5.1}$$

we have

$$O(m) = O(m_k) + \big(\nabla O(m_k)\big)^T \delta m + \frac{1}{2}(\delta m)^T \big[\nabla \cdot (\nabla O(m_k))^T\big]\delta m. \tag{5.2}$$

We define the Hessian H_k by

$$H_k = \nabla \cdot (\nabla O(m_k))^T. \tag{5.3}$$

Taking the gradient of $O(m)$ in Eq. 5.2 with respect to m, we get

$$\begin{aligned} \nabla O(m) &= \nabla(\delta m)^T[\nabla O(m_k)] + \frac{1}{2}\{[\nabla(\delta m)^T](H_k \delta m) + \nabla(\delta m^T H_k^T)\delta m\} \\ &= \nabla O(m_k) + \frac{1}{2}(H_k \delta m + H_k^T \delta m). \end{aligned} \tag{5.4}$$

It is easy to show that H_k is symmetric. So Eq. 5.4 becomes

$$\nabla O(m) = \nabla O(m_k) + H_k \delta m. \tag{5.5}$$

If m minimizes $O(m)$ then

$$\nabla O(m) = 0. \tag{5.6}$$

Setting $\nabla O(m) = 0$ in Eq. 5.5 and rearranging gives

$$H_k \delta m = -\nabla O(m_k). \tag{5.7}$$

Eq. 5.7 is usually written in the iterative form which is given by

$$H_k \delta m_{k+1} = -\nabla O(m_k), \tag{5.8}$$

and

$$m_{k+1} = m_k + \delta m_{k+1}. \tag{5.9}$$

This is Newton's method. Based on the objective function Eq. 2.14, we can easily write down the gradient of the objective function which is given by

$$\nabla O(m_k) = G_k^T C_D^{-1}(g(m_k) - d_{\text{obs}}) + C_M^{-1}(m_k - m_{\text{prior}}), \tag{5.10}$$

and the Hessian matrix which is given by

$$H(m_k) = \left(\nabla G_k^T\right) C_D^{-1}(g(m_k) - d_{\text{obs}}) + G_k^T C_D^{-1} G_k + C_M^{-1}. \tag{5.11}$$

In these equations, G_k denotes the matrix of sensitivity coefficients, i.e., the derivatives of predicted data with respect to model parameters, evaluated at m_k. The individual elements of the sensitivity matrix are

$$G_{i,j} = \frac{\partial g_i}{\partial m_j}, \tag{5.12}$$

for $1 \leq i \leq N_d$ and $1 \leq j \leq N_m$. A sensitivity coefficient gives a measure of how strongly the change in the data, $d_i = g_i(m)$, is affected by the change in the model parameter m_j. The sensitivity coefficients can be obtained by either gradient simulator method or adjoint method. If the number of model parameters and the number of data are both large, computation of the sensitivity coefficient matrix G by either the gradient simulator method or the adjoint method is not feasible.

The term in Eq. 5.11 that involves the gradient of G is small if the residual term, $(g(m_k) - d_{\text{obs}})$, is small, or if the data is linearly related to the model parameters, i.e., the function $g(m)$ is linear. As a matter of fact, calculation the gradient of G is impractical in practice. So we simply drop this term to form the Gauss-Newton method. The approximation to the Hessian at m_k is then given by

$$H_k = H(m_k) = C_M^{-1} + G_k^T C_D^{-1} G_k. \tag{5.13}$$

Substitution Eqs. 5.13 and 5.10 into Eq. 5.8 gives the Gauss-Newton iterative procedure, i.e.,

$$\boxed{(C_M^{-1} + G_k^T C_D^{-1} G_k)\delta m_{k+1} = -C_M^{-1}(m_k - m_{\text{prior}}) - G_k^T C_D^{-1}(g(m_k) - d_{\text{obs}}).} \tag{5.14}$$

This equation is called the Gauss-Newton equation. The model is updated by

$$m_{k+1} = m_k + \mu_k \delta m_{k+1}, \tag{5.15}$$

where μ_k is usually obtained by the restricted-step procedure; see Fletcher (1987). Note δm_{k+1} is the search direction. The model obtained at convergence based on the above derivation is the maximum a posteriori (MAP) estimate of the model which is the most probable model. If one wants to generate the realizations, the iterative equation Eq. 5.14 should be modified slightly by replacing $d_{\rm obs}$ by $d_{\rm uc}$ and $m_{\rm prior}$ by $m_{\rm uc}$. We also can consider the case where the corrections to the prior mean are also variables. The corresponding version of the Gauss-Newton iterative procedure is given later.

Gauss-Newton equation given by Eq. 5.14 requires solving an $N_m \times N_m$ matrix problem where N_m is the number of model parameters which is normally large as well as evaluating C_M^{-1}. By applying the inverse lemma (see Beck and Arnold (1977)), Eq. 5.14 with Eq. 5.15 can be reformulated as

$$\begin{aligned}\delta m_{k+1} =&(m_{\rm prior} - m_k)+ \\ &\left\{C_M G_k^T \left[C_D + G_k C_M G_k^T\right]^{-1}\left[G_k(m_k - m_{\rm prior}) - \left(g(m_k) - d_{\rm obs}\right)\right]\right\}.\end{aligned} \tag{5.16}$$

Note that Eq. 5.16 requires solving an $N_d \times N_d$ matrix problem where N_d is the number of data. This formulation will result in an algorithm which is much more computationally efficient than application of Eq. 5.14 if $N_d << N_m$.

For many examples, the Gauss-Newton procedure with restricted-step converges without difficulty. However, it may be necessary to damp the Gauss-Newton step at early iterations if the initial estimate gives a large data mismatch; see, for example, Wu et al. (1999) and Li et al. (2001). If damping is not done, the effect of regularization provided by the prior model covariance matrix appears to be lost. In this case, the Gauss-Newton method may yield rock property fields which are excessively rough and give an unacceptable match of the pressure data. This problem can be avoided using the ideas due to Levenberg and Marquardt. The modified Levenberg-Marquardt algorithm given in Bi (1999) can be written as

$$\delta m_{k+1} = -\Big[(1+\lambda)C_M^{-1} + G_k^T C_D^{-1} G_k\Big]^{-1}\Big[C_M^{-1}(m_k - m_{\rm prior}) + G_k^T C_D^{-1}(g(m_k) - d_{\rm obs})\Big]. \tag{5.17}$$

As shown by Bi (1999), matrix inverse lemma can be applied to rewrite Eq. 5.17 as

$$\delta m_{k+1} = \frac{m_{\text{prior}} - m_k}{1+\lambda} + C_M G_k^T \Big[(1+\lambda)C_D + G_k C_M G_k^T\Big]^{-1} \Big[\frac{G_k(m_k - m_{\text{prior}})}{1+\lambda} - \big(g(m_k) - d_{\text{obs}}\big)\Big]. \tag{5.18}$$

Note the formula of Eq. 5.17 requires calculation of C_M^{-1} and then solving an $N_m \times N_m$ matrix problem where N_m is the number of model parameters. Applying Eq. 5.18 requires solving an $N_d \times N_d$ matrix problem where N_d is the number of data. When we apply Levenberg-Marquardt, we usually choose a large number as the initial value for λ, for example, 10^5, such that the initial step is relatively small. If the value of λ used results in a decrease in the objective function, we simply decrease λ for the next iteration by a factor of 10. Otherwise, λ is increased by a factor of 10. The model m_k is not updated to m_{k+1} unless the update decreases the objective function. One can also compute the optimal damping factor at each step; see, for example, Abacioglu et al. (2001). However, calculation of the optimal damping factor is quite expensive. Thus, in our study, we use the simple scheme based on multiplying or dividing λ by a factor of 10.

5.3 Truncated Gauss-Newton Method

According to Nash (1985), if the search direction in the Gauss-Newton method (see, for example, Eq. 5.16) is not formed accurately, then the corresponding iterative procedure is referred to as the truncated Gauss-Newton method. Typically, the Gauss-Newton equation could be solved approximately at early iterations due to the fact that the second order approximation of the objective function is not a good approximation at early iterations. When the model approaches the minimum, the Hessian matrix tends to be a constant matrix, then the Gauss-Newton equation should be solved more accurately. The Gauss-Newton version given by Eq. 5.16 requires solving

$$\big(C_D + G_k C_M G_k^T\big)x = G_k(m_k - m_{\text{prior}}) - (g(m_k) - d_{\text{obs}}), \tag{5.19}$$

for

$$x = \left(C_D + G_k C_M G_k^T\right)^{-1}\left(G_k(m_k - m_{\text{prior}}) - (g(m_k) - d_{\text{obs}})\right). \tag{5.20}$$

Eq. 5.19, which is a linear equation, can be solved by any iterative solver, for example, successive-over-relax (SOR), steepest descent or conjugate gradient. Detailed information about these linear iterative solvers can be found in Appendix A. If this matrix problem is solved iteratively by the conjugate gradient method, then one does not need to explicitly compute G; one only needs to be able to calculate Gu and $G^T v$ for vectors u and v at each iteration of the Gauss-Newton or Levenberg-Marquardt algorithm. Chu et al. (2000) suggested solving Eq. 5.19 by a conjugate gradient method and implemented a procedure for computing Gu and $G^T v$ for the single-phase flow of a slightly compressible fluid. A somewhat different and clearer presentation of how one may compute Gu and $G^T v$ is given in Abacioglu (2001). Computation of Gu requires a forward run of the simulation. Computation of $G^T v$ requires one solution of the adjoint system. As the solution of the adjoint system requires roughly the same computational time as one simulation run, each iteration of the conjugate gradient method requires roughly two reservoir simulation runs. Computation of the right-hand side of Eq. 5.19 also requires one simulator run to evaluate $G_k(m_k - m_{\text{prior}})$ but must be done only once for each Gauss-Newton iteration. To apply the conjugate gradient algorithm, we also need to calculate the residual corresponding to the initial estimate which requires one operation of Gu and one operation of $G^T v$. To accomplish one Gauss-Newton iteration, we need one more operation of $G^T v$ outside the inner iteration; see Eq. 5.16. Thus, if the inner iteration (the solution of Eq. 5.19 by the conjugate gradient method) requires on average k_{CG} iterations for convergence and k_{GN} iterations are required to obtain convergence of the Gauss-Newton method, roughly

$$I_{GN} = k_{GN}(2k_{CG} + 4), \tag{5.21}$$

reservoir simulation runs are required to generate each realization. For the overall procedure to be feasible k_{CG} must be quite small. If an extremely good preconditioning matrix could be found for the conjugate gradient step, it is possible that the

method could be effective. However, the matrices G_k and $C_D + G_k C_M G_k^T$ are never explicitly constructed, so it is not clear how to construct a good preconditioner. At this point, we are skeptical that the method will prove to be sufficiently computationally efficient for practical applications, but we have not implemented it in our work.

5.4 Nonlinear Conjugate Gradient Method

Nonlinear conjugate gradient method which is usually used to minimize non-quadratic function can "be evolved" from the linear conjugate gradient method which is normally used to solve a linear equation system. A variety of linear conjugate gradient algorithms and nonlinear conjugate gradient algorithms are given in Appendix C. In this section, we focus on the application of the nonlinear conjugate gradient method to our history matching problem.

In the conjugate gradient method, the search direction is given by

$$d_{k+1} = -M_k^{-1} g_k + \beta_k d_k, \tag{5.22}$$

where k is the iteration index, g_k represents the gradient of the objective function, M_k is called the preconditioning matrix which is an approximation to the Hessian matrix H_k and β_k is obtained by the Polak-Ribière formula given by

$$\beta_k = \frac{r_{k+1}^T (M_{k+1}^{-1} r_{k+1} - M_k^{-1} r_k)}{r_k^T M_k^{-1} r_k}, \tag{5.23}$$

where $r_k = -g_k$. As discussed later, the step size can be obtained by a line search. If we choose the preconditioning matrix M_k to be identity matrix I, then Eq. 5.22 reduces to the standard conjugate gradient method without preconditioning.

It is well known that the nonlinear conjugate gradient method can be applied to minimize non-quadratic objective functions; see, Fletcher and Reeves (1964). Although the method has been applied for the history matching of production data (see, for example, Makhlouf et al. (1993)), its slow rate of convergence has precluded its use in large scale history matching problems. The success of the conjugate gradient method for nonlinear optimization depends on whether we are able to construct a

good preconditioner. A good preconditioning matrix at the kth iteration is a matrix M_k which is a good approximation to the Hessian H_k so that

$$M_k^{-1} H_k \approx I. \tag{5.24}$$

For our problem, the Hessian at the kth iteration is given by

$$H_k = C_M^{-1} + G_k^T C_D^{-1} G_k, \tag{5.25}$$

see the deviation of Eq. 5.13. An optional preconditioner for the conjugate gradient method would be

$$M_k = H_k, \tag{5.26}$$

but the conjugate gradient method requires solving the matrix problem

$$M_k \tilde{d}_k = -g_k, \tag{5.27}$$

to form search direction d_{k+1} using Eq. 5.22. If $M_k = H_k$, Eq. 5.27 requires the same computational effort as the direct application of Gauss-Newton method Eq. 5.14 and does not improve computational efficiency. If we choose $M_k = C_M^{-1}$, however, then Eq. 5.27 becomes

$$\tilde{d}_k = -C_M g_k, \tag{5.28}$$

and the calculation of $\tilde{d}_k$ which is the first term in Eq. 5.22 requires only multiplication of g_k by the prior covariance matrix C_M. Kalita (2000) considered the problem of conditioning a gas reservoir model to well test pressure data by automatic history matching. Both the Gauss-Newton method and the conjugate gradient method with C_M^{-1} as the preconditioner were used to minimize the relevant objective function (Eq. 2.14 or Eq. 2.18). Kalita's results indicate that the conjugate gradient method was not always more efficient than the Gauss-Newton method. Moreover, in most cases, the conjugate gradient method converged to a value of the objective function which was significantly higher than the converged value of the objective function obtained by the Gauss-Newton method.

In the preconditioned conjugate gradient method, the preconditioning matrix M_k is used only in equations like Eq. 5.27. Thus, it is preferable to estimate M_k^{-1}

directly instead of estimating M_k. We would like M_k^{-1} to be an approximation to the inverse Hessian. This suggests that $\tilde{H}_k^{-1}$ constructed from quasi-Newton might be a good candidate for a preconditioner. Quasi-Newton method will be discussed in the next section. The difficulty with this procedure is that we can only approximate the quasi-Newton $\tilde{H}_k^{-1}$ using information in the conjugate gradient algorithm. Our work indicated that the preconditioner constructed by this scheme works better than C_M^{-1} for some cases, for example, in the gas reservoir examples shown by Zhang et al. (2001); and works worse than C_M^{-1} for some cases, for example, in the three-phase example presented later. The reason is that the iterates generated by the quasi-Newton method are different from the iterates generated by the conjugate gradient method. The search direction for the conjugate gradient algorithm is given by Eq. 5.22 whereas it is given by Eq. 5.31 in the quasi-Newton method. Different search directions generate different iterates and in turn different y_k's and s_k's which are used to construct Hessian inverse approximation matrix $\tilde{H}_k^{-1}$. Therefore, the inverse Hessian approximation generated within the conjugate gradient algorithm will not be the same as the one generated in a quasi-Newton method. In particular, the "inverse Hessian approximation" generated with the conjugate gradient procedure may not have the property that the inverse Hessian approximation will be equal to the true inverse Hessian at the Nth iteration for a N-dimensional quadratic function given that the line search is exact; see Oren and Luenberger (1974) and Oren (1974b).

5.5 Quasi-Newton Methods

The search direction in the Newton's method can be written as

$$d_{k+1} = -H_k^{-1} g_k, \tag{5.29}$$

where H_k and g_k, respectively, denote the second derivative (Hessian matrix) and the first derivative (gradient) of the objective function evaluated at m_k and k is the iteration index. With $O(m)$ given by either Eq. 2.14 or Eq. 2.18, the Gauss-Newton

Hessian matrix is estimated by

$$H_k = C_M^{-1} + G_k^T C_D^{-1} G_k, \tag{5.30}$$

where G_k is the sensitivity matrix evaluated at m_k. As noted before, if both the number of model parameters and the number of data are large, the evaluation of G_k is computationally expensive. In quasi-Newton methods, H_k^{-1} is approximated by a symmetric positive definite matrix $\tilde{H}_k^{-1}$ which is corrected or updated from iteration to iteration. With this Hessian inverse approximation matrix, the search direction can be written as

$$d_{k+1} = -\tilde{H}_k^{-1} g_k. \tag{5.31}$$

Because the matrix $\tilde{H}_k^{-1}$ takes the place of H_k^{-1} in Eq. 5.29, the method with search direction given by Eq. 5.31 is called a quasi-Newton method. This method is also called a variable metric method. The reason why this method is called a variable metric method is given below.

Suppose $f(x)$ represents the real functional and p is a nonzero vector. The directional derivative of $f(x)$ at x_0 along the direction p is given by

$$\frac{\partial f(x_0)}{\partial p} = \left[\nabla f(x_0)\right]^T e, \tag{5.32}$$

where e is the unit vector in the direction p. If $\left[\nabla f(x_0)\right]^T e < 0$, then p is a downhill direction, otherwise it is a uphill direction. The rate of change in the function $f(x)$ in the direction p depends on the absolute value of the directional derivative, i.e., $\left|\frac{\partial f(x_0)}{\partial p}\right|$. Applying the Cauchy-Schwarz inequality, we have

$$\left|\frac{\partial f(x_0)}{\partial p}\right| = \left|\left[\nabla f(x_0)\right]^T e\right| \leq \| \nabla f(x_0) \|_2 * \| e \|_2 = \| \nabla f(x_0) \|_2, \tag{5.33}$$

where $\| \cdot \|_2$ represents the l_2 norm or the Euclidean norm. For example, if x is a vector, then $\|x\|_2 = \sqrt{x^T x}$. From Eq. 5.33, we have

$$\left|\left[\nabla f(x_0)\right]^T e\right| \leq \| \nabla f(x_0)] \|_2 \, . \tag{5.34}$$

If we take

$$e = -\frac{\nabla f(x_0)}{\|\nabla f(x_0)\|_2} \equiv e_0, \tag{5.35}$$

then $\left|\left[\nabla f(x_0)\right]^T e\right|$ takes on its maximum value given by the right-hand side of Eq. 5.34. Because

$$\left[\nabla f(x_0)\right]^T e_0 = -\parallel \nabla f(x_0)\right] \parallel_2 < 0, \tag{5.36}$$

$e = e_0$ results in minimizing $\dfrac{\partial f(x_0)}{\partial p}$. Therefore,

$$p = \|\nabla f(x_0)\|_2 e_0 = -\nabla f(x_0) \tag{5.37}$$

is the steepest descent direction of $f(x)$ at x_0.

We introduce a new norm or a new metric $\|\cdot\|_A$ to measure the length of a vector. $\|\cdot\|_A$ defined by

$$\|x\|_A = \sqrt{x^T A x} \tag{5.38}$$

where A is a symmetric positive definite matrix. We can show that

$$|x^T A y| \leq \|x\|_A \|y\|_A. \tag{5.39}$$

The absolute value of the directional derivative in terms of the new introduced norm $\|\cdot\|_A$ can be written as

$$\begin{aligned} \left|\left[\nabla f(x_0)\right]^T e\right| &= \left|\left[\nabla f(x_0)\right]^T A^{-1} A e\right| = \left|\left[A^{-1}\nabla f(x_0)\right]^T A e\right| \\ &\leq \|A^{-1}\nabla f(x_0)\right] \|_A \|e\|_A = \|\left[A^{-1}\nabla f(x_0)\right] \|_A. \end{aligned} \tag{5.40}$$

Note that here e is a unit vector in terms of $\|\cdot\|_A$, i.e., $\|e\|_A = 1$. From Eq. 5.40, we have

$$\left|\left[\nabla f(x_0)\right]^T e\right| \leq \|\left[A^{-1}\nabla f(x_0)\right] \|_A. \tag{5.41}$$

If we take

$$e = -\frac{A^{-1}\nabla f(x_0)}{\|A^{-1}\nabla f(x_0)\|_A} \equiv e_0, \tag{5.42}$$

then

$$\left|\left[\nabla f(x_0)\right]^T e_0\right| = \|A^{-1}\nabla f(x_0)\|_A \tag{5.43}$$

which is the maximum value of the directional derivative in terms of $\|\cdot\|_A$. Therefore,

$$p = \|A^{-1}\nabla f(x_0)\|_A e_0 = -A^{-1}\nabla f(x_0) \tag{5.44}$$

is the steepest descent direction in terms of $\|\cdot\|_A$ of the function $f(x)$ at x_0. If the objective function is a quadratic function, then the second derivative of the objective function, i.e., the Hessian matrix, is a constant matrix, will take the place of matrix A in Eq. 5.44. If we consider a nonlinear problem, like our history matching problem, the Hessian matrix H_k, which can be treated as a metric matrix, changes from iteration to iteration. This provides a motivation for calling this method a variable metric method. We also can treat this method as a "steepest descent" method in terms of the new metric.

In a quasi-Newton method, the key issue is how to generate the approximation to the inverse Hessian matrix. Different quasi-Newton methods use different formulas to calculate $\tilde{H}_{k+1}^{-1}$ from $\tilde{H}_k^{-1}$. All updating formulas satisfy the quasi-Newton condition given by

$$\tilde{H}_{k+1}^{-1} y_k = s_k, \tag{5.45}$$

where

$$y_k = g_{k+1} - g_k, \tag{5.46}$$

and

$$s_k = m_{k+1} - m_k; \tag{5.47}$$

see Appendix B. Various possible updating formulas honor this quasi-Newton condition. The Broyden family equation is given by

$$\tilde{H}_{k+1}^{-1} = \tilde{H}_k^{-1} + \frac{s_k s_k^T}{s_k^T y_k} - \frac{\tilde{H}_k^{-1} y_k y_k^T \tilde{H}_k^{-1}}{y_k^T \tilde{H}_k^{-1} y_k} + \theta_k v_k v_k^T, \tag{5.48}$$

where $\theta_k \in [0, 1]$ and

$$v_k = (y_k^T \tilde{H}_k^{-1} y_k)^{1/2} \Big(\frac{s_k}{s_k^T y_k} - \frac{\tilde{H}_k^{-1} y_k}{y_k^T \tilde{H}_k^{-1} y_k} \Big). \tag{5.49}$$

Given that the line search is exact and the initial Hessian inverse approximation is real symmetric positive definite, the Hessian inverse approximation generated by Eq. 5.48 is guaranteed to be symmetric positive definite; see details in Appendix B. In our procedure, we use the Broyden-Fletcher-Goldfarb-Shanno (BFGS) correction equation proposed by Broyden (1970), Fletcher (1970), Goldfarb (1970) and Shanno

(1970) independently, which is a special case of Broyden family obtained by setting $\theta_k = 1$ in Eq. 5.48. The BFGS update equation is given by

$$\tilde{H}_{k+1}^{-1} = \tilde{H}_k^{-1} + \frac{s_k s_k^T}{s_k^T y_k} - \frac{\tilde{H}_k^{-1} y_k y_k^T \tilde{H}_k^{-1}}{y_k^T \tilde{H}_k^{-1} y_k} + v_k v_k^T. \tag{5.50}$$

The limited memory BFGS (LBFGS), which uses a limited number of previous vectors (y_k's and s_k's) to construct the inverse Hessian approximation at each iteration, is an appropriate method for large scale problems where it is not feasible to explicitly store and compute the full matrix $\tilde{H}_k^{-1}$. In our work, the algorithm proposed by Nocedal (1980) was implemented and applied. In order to derive the limited memory BFGS, the normal BFGS formula Eq. 5.50 can be written as

$$\tilde{H}_{k+1}^{-1} = V_k^T \tilde{H}_k^{-1} V_k + \rho_k s_k s_k^T, \tag{5.51}$$

where $\rho_k = 1/y_k^T s_k$ and $V_k = I - \rho_k y_k s_k^T$. Nocedal (1980) suggested a procedure where only the L previous vectors are used when constructing the new $\tilde{H}_{k+1}^{-1}$. When $k < L$, the update equation is still given by Eq. 5.51 which can be rewritten as

$$\begin{aligned} \tilde{H}_{k+1}^{-1} =& V_k^T V_{k-1}^T \cdots V_0^T \tilde{H}_0^{-1} V_0 \cdots V_{k-1} V_k \\ &+ V_k^T \cdots V_1^T \rho_0 s_0 s_0^T V_1 \cdots V_k \\ &\vdots \\ &+ V_k^T \rho_{k-1} s_{k-1} s_{k-1}^T V_k \\ &+ \rho_k s_k s_k^T. \end{aligned} \tag{5.52}$$

For $k + 1 > L$ the update equation is

$$\begin{aligned} \tilde{H}_{k+1}^{-1} =& V_k^T V_{k-1}^T \cdots V_{k-L+1}^T \tilde{H}_0^{-1} V_{k-L+1} \cdots V_{k-1} V_k \\ &+ V_k^T \cdots V_{k-L+2}^T \rho_{k-L+1} s_{k-L+1} s_{k-L+1}^T V_1 \cdots V_k \\ &\vdots \\ &+ V_k^T \rho_{k-1} s_{k-1} s_{k-1}^T V_k \\ &+ \rho_k s_k s_k^T. \end{aligned} \tag{5.53}$$

Unless the dimension of $\tilde{H}_k^{-1}$ is small, direct application of Eqs. 5.52 and 5.53, which involve matrix products, is inefficient. Instead, we form the product $\tilde{H}_k^{-1} g_k$, which

is used to construct the search direction, directly by using the algorithm proposed by Nocedal (1980). The calculation of $\tilde{H}_k^{-1} g_k$ only involves vector products instead of matrix products. Because only the L most recent vectors from the set of s_k and y_k are used to construct $\tilde{H}_{k+1}^{-1}$, this algorithm is called the limited memory BFGS method. The precise algorithm for this method is given in Appendix B.

The BFGS or LBFGS algorithm we used to minimize $O(m)$ (Eq. 2.14 or Eq. 2.18) is given below.

Step 1 Initialization

(a)Provide an initial guess, m_0, of the model, calculate the objective function corresponding to m_0 and evaluate the gradient of the objective function at m_0, i.e., compute g_0; (b)provide an initial Hessian inverse approximation $\tilde{H}_0^{-1}$ (e.g., C_M in our examples), set the initial iteration index k=0.

Step 2 Calculate the search direction $d_k = -\tilde{H}_k^{-1} g_k$ and check whether it is a downhill direction, i.e., check to see if $d_k^T g_k < 0$. If d_k is not a downhill search direction, set $d_k = -\tilde{H}_0^{-1} g_k$.

Step 3 Calculate the step size α_k by a line search procedure as discussed later.

Step 4 Update the model to $m_c = m_k + \alpha_k d_k$.

Step 5 Calculate the objective function based on m_c.

Step 6 Determine if the Wolfe conditions (discussed later) are satisfied; if they are satisfied, then set $m_{k+1} = m_c$ and go to step 7, otherwise do

(a) fit a quadratic and find a step size by minimizing this quadratic, then go to step 4;

(b) if a quadratic fit has already been done, cut the step size by a specified factor (in our examples we cut the step size by a factor of 10) and go to step 4. All computations we have done suggest this case does not occur very often.

Step 7 Determine if the stopping criteria are satisfied. If satisfied, then stop; otherwise go to step 8.

Step 8 Calculate $s_k = m_{k+1} - m_k = \alpha_k d_k$ and $y_k = g_{k+1} - g_k$. Apply Eq. 5.50 or Eqs. 5.52 and 5.53 to update the inverse Hessian approximation $\tilde{H}_{k+1}^{-1}$. Set $k = k + 1$ and then go to step 2.

5.5.1 Scaling

The scaling is obtained by multiplying the old $\tilde{H}_k^{-1}$ by a factor γ_k and then substituting $\gamma_k \tilde{H}_k^{-1}$ instead of $\tilde{H}_k^{-1}$ itself into the update equation (e.g., Eq. 5.48) to calculate $\tilde{H}_{k+1}^{-1}$; see Oren (1973), Oren and Luenberger (1974), Oren (1974b) and Shanno (1970). If we do so, Eq. 5.48 becomes

$$\tilde{H}_{k+1}^{-1} = \gamma_k \tilde{H}_k^{-1} + \frac{s_k s_k^T}{s_k^T y_k} - \frac{\gamma_k \tilde{H}_k^{-1} y_k y_k^T \gamma_k \tilde{H}_k^{-1}}{y_k^T \gamma_k \tilde{H}_k^{-1} y_k} + \theta_k \gamma_k v_k v_k^T, \tag{5.54}$$

and this equation can be further simplified to

$$\tilde{H}_{k+1}^{-1} = \Big(\tilde{H}_k^{-1} - \frac{\tilde{H}_k^{-1} y_k y_k^T \tilde{H}_k^{-1}}{y_k^T \tilde{H}_k^{-1} y_k} + \theta_k v_k v_k^T \Big) \gamma_k + \frac{s_k s_k^T}{s_k^T y_k}, \tag{5.55}$$

where v_k is given by Eq. 5.49 and is the same one used in Eqs. 5.48 and 5.50.

For BFGS and LBFGS, scaling can have a significant effect on the rate of convergence. The self-scaling variable metric (SSVM) method developed by Oren and Luenberger (1974) and Oren (1974b) is motivated by the desire to choose a scalar γ_{k-1} so that the condition number of $R_k = H_k^{1/2} \tilde{H}_k^{-1} H_k^{1/2}$ is as close to one as possible. If $\tilde{H}_k^{-1}$ is identical to the inverse of the true Hessian, H_k, then this condition number is equal to one. For a quadratic objective function, these authors provide theoretical conditions and a method for computing γ_k that insure that (i) $\lambda_{\min} \leq 1 \leq \lambda_{\max}$ where $\lambda_{\min}$ and $\lambda_{\max}$, respectively, denote the minimum and maximum eigenvalues of R_k; and (ii) the condition number of R_{k+1} is less than or equal to the condition number of R_k. A quasi-Newton method which satisfies these two conditions is referred to as a self-scaling variable metric method.

Let

$$\tau_k = \frac{s_k^T \tilde{H}_k s_k}{s_k^T y_k}, \tag{5.56}$$

and

$$\sigma_k = \frac{s_k^T y_k}{y_k^T \tilde{H}_k^{-1} y_k}. \tag{5.57}$$

By applying the fact that

$$s_k = \alpha_k d_k = -\alpha_k \tilde{H}_k^{-1} g_k, \tag{5.58}$$

and using the fact that $\tilde{H}_k^{-1}$ is real symmetric, Eq. 5.56 can be rewritten as

$$\begin{aligned} \tau_k &= \frac{s_k^T \tilde{H}_k s_k}{s_k^T y_k} && (5.59)\\ &= -\alpha_k \frac{g_k^T s_k}{s_k^T y_k} && (5.60)\\ &= \frac{s_k^T g_k}{g_k^T \tilde{H}_k^{-1} y_k}. && (5.61) \end{aligned}$$

In general the motivation for using the last two formulae to calculate τ_k is to avoid calculating the inverse of $\tilde{H}_k^{-1}$.

There are many options we can choose to perform scaling. Oren and Spedicato (1976) proposed an optimal condition which minimizes the upper bound of the condition number of $\tilde{H}_{k+1}^{-1}$ by proper selection of θ_k and γ_k. This condition is given by

$$\theta_k = \frac{a_k b_k - a_k^2 \gamma_k}{b_k c_k \gamma_k - a_k^2 \gamma_k} \tag{5.62}$$

where θ_k is the parameter used in Broyden family update equation, Eq. 5.48, $a_k = s_k^T y_k$, $b_k = s_k^T \tilde{H}_k s_k$ and $c_k = y_k^T \tilde{H}_k^{-1} y_k$. With these definitions, there is still freedom to select scaling factors. In the same paper, Oren proposed four similar switching rules based on his earlier published paper (Oren (1974a)). One of the switching rules for choosing γ_k and θ_k in Eq. 5.55 is

if $\tau_k \leq 1$, choose $\gamma_k = \tau_k$ and $\theta_k = 0$;

if $\sigma_k \geq 1$, choose $\gamma_k = \sigma_k$ and $\theta_k = 1$;

if $\sigma_k \leq 1 \leq \tau_k$, choose $\gamma_k = 1$ and $\theta_k = \frac{1 - \sigma_k}{\tau_k - \sigma_k}$.

Some examples (see Zhang et al. (2001)) show that when we use $\theta = \theta_k$ not equal to 1 in Eq. 5.55, the convergence rate is almost always slower than the case where we use $\theta_k = 1$. Therefore, we consider only the case where $\theta_k = 1$; this choice corresponds to the BFGS algorithm. Setting $\theta_k = 1$ in Eq. 5.62 and solving for $\gamma = \gamma_k$, we obtain the optimal scaling factor for BFGS which is given by

$$\gamma_k = \frac{a_k}{c_k} = \frac{s_k^T y_k}{y_k^T \tilde{H}_k^{-1} y_k} = \sigma_k, \tag{5.63}$$

where σ_k is given by Eq. 5.57. Shanno and Phua (1978) and Yang and Watson (1988) use this scaling factor and only scale the initial matrix, $\tilde{H}_0^{-1}$, in their work. In our implementation, we modify Oren's switching rule to

$$\begin{cases} \gamma_k = \tau_k, & \text{if} \quad \tau_k \leq 1; \\ \gamma_k = \sigma_k, & \text{otherwise} \end{cases} \tag{5.64}$$

and always use $\theta_k = 1$. Note that using switching rule Eq. 5.64 for choosing the scaling factor still guarantees the condition number of matrix $R_k = H_k^{1/2} \tilde{H}_k^{-1} H_k^{1/2}$ monotonously decreases at least for quadratic functions; see Appendix B for detail.

The scaled version BFGS algorithm is similar to the standard BFGS algorithm given previously. The only difference is in step 8. For the scaled BFGS algorithm, step 8 is replaced by

Step 8 Calculate $s_k = m_{k+1} - m_k = \alpha_k d_k$ and $y_k = g_{k+1} - g_k$. Calculate τ_k by Eq. 5.61 and determine whether τ_k is less than one. If it is, then set $\gamma_k = \tau_k$. Otherwise, calculate σ_k using Eq. 5.57 and set $\gamma_k = \sigma_k$. Apply Eq. 5.55 to update the inverse Hessian approximation $\tilde{H}_{k+1}^{-1}$. Set $k = k + 1$ and go to step 2.

This step 8 is given for the case where the inverse Hessian approximation is scaled at each iteration. For the case where only initial scaling is done, set $\gamma_k = 1$ for $k > 0$ in this step. For the LBFGS algorithm with initial scaling, we just replace $\tilde{H}_0^{-1}$ in Eqs. 5.52 and 5.53 by $\gamma_0 \tilde{H}_0^{-1}$ in computing $\tilde{H}_1^{-1}$ and use $\gamma_k = 1$ at all subsequent iterations. For the LBFGS with all scaling, we replace $\tilde{H}_0^{-1}$ in Eqs. 5.52 and 5.53 by $\gamma_k \tilde{H}_0^{-1}$ in computing $\tilde{H}_{k+1}^{-1}$. The efficient LBFGS method given by

Nocedal (1980) avoid formation of $\tilde{H}_k^{-1}$ for $k \geq 1$, only $\tilde{H}_k^{-1} g_k$ is calculated at each iteration. However, $\tilde{H}_0^{-1}$ must be provided as the initial approximation to the inverse Hessian. Based on these considerations, we tried implementing Eq. 5.57 with $\tilde{H}_k^{-1}$ replaced by $\tilde{H}_0^{-1}$, then we have

$$\tilde{\sigma}_k = \frac{s_k^T y_k}{y_k^T \tilde{H}_0^{-1} y_k}. \tag{5.65}$$

The $\tilde{H}_k^{-1}$ and $\tilde{H}_k$ in Eqs. 5.59 and 5.61 are replaced by $\tilde{H}_0^{-1}$ and $\tilde{H}_0$ respectively. The resulting three equations for τ_k are no longer equivalent except at the first iteration. By using $\tilde{H}_0^{-1}$ in place of $\tilde{H}_k^{-1}$ and $\tilde{H}_0$ in place of $\tilde{H}_k$ in Eqs. 5.59 and 5.61, we obtain

$$\tilde{\tau}_{1k} = \frac{s_k^T \tilde{H}_0 s_k}{s_k^T y_k} \tag{5.66}$$

$$\tilde{\tau}_{3k} = \frac{s_k^T g_k}{g_k^T \tilde{H}_0^{-1} y_k} \tag{5.67}$$

respectively. However, Eq. 5.60 can be applied exactly in all cases. To simplify the notation, we let

$$\tilde{\tau}_{2k} \equiv \tau_k = -\alpha_k \frac{g_k^T s_k}{s_k^T y_k}. \tag{5.68}$$

More details about the scaling schemes we used and how they affect the convergence are given in the example sections.

5.6 Convergence Criteria

In our results, the following stopping criteria are used to terminate the algorithm:

1. $$\frac{\mid O_{k+1} - O_k \mid}{O_k + 10^{-14}} < \varepsilon_1 \tag{5.69}$$

   and

   $$\frac{\parallel m_{k+1} - m_k \parallel_2}{\parallel m_k \parallel_2 + 10^{-14}} < \varepsilon_2 \tag{5.70}$$

   where k denotes the iteration index and $\| \cdot \|_2$ denotes the l_2 norm of a vector. Both conditions must be satisfied to terminate the iteration. If we use only Eq. 5.69 as the convergence criterion, the algorithm may converge prematurely

especially when the objective function decreases very slowly at the early iterations. Because at the early iteration, the objective function is relatively big such that Eq. 5.69 becomes easier to be satisfied.

2. Specify a maximum allowable iteration number. If the number of iterations exceeds the specified number, we force the iteration to stop. In our examples, we usually specify the maximum number of iterations as 100. Note that reaching the maximum number of iterations does not imply that the algorithm has converged.

5.7 Line Search

In our implementation of conjugate gradient and quasi-Newton methods, the line search is performed using one iteration of the Newton-Raphson method followed by a quadratic fit if necessary. We do not do an exact line search, but terminate the line search when the Wolfe conditions are satisfied; see, for example, Fletcher (1987). The Wolfe conditions are used to ensure that step sizes are not too small and that the reduction in the objective function is not negligible. In addition, the Wolfe conditions are side conditions for the exact line search; see Kolda et al. (1998). At each iteration, we perform one Newton-Raphson iteration to find a step size. Then we check whether this step satisfies the Wolfe conditions. If it does, we accept this step. Otherwise we find an optimum step size by fitting a quadratic, as discussed in the next section, and then check whether the new step satisfies the Wolfe conditions. If it does, we accept this new step. Otherwise, we check whether the objective function increases or decreases. If it increases, we cut the step size by a factor of 10. If it decreases, we accept this new step size no matter whether the Wolfe conditions are satisfied or not. Our experience shows that for most of the iterations, the step size generated by one Newton-Raphson iteration satisfies the Wolfe conditions and virtually all the step sizes satisfy the Wolfe conditions after the quadratic fit. One may argue that we should perform a sequence of quadratic fits or a sequence of cubic fits after one quadratic fit instead of cutting the step size after

one quadratic fit. Our limited experience shows that the Wolfe conditions may never be satisfied during the sequence of quadratic or cubic fits. If this situation happens, then we are in a "dead loop" and the iteration never terminates. Because of this, we implemented the simple scheme of reducing the step size by a factor of ten whenever the situation arise that the Wolfe conditions are not satisfied after the quadratic fit. Sometimes, however, this procedure leads to a false convergence. Thus, whenever we obtain convergence as a result of reducing the step size by a factor of 10, we check the objective function value. If the objective function value is still so big, then we restart the algorithm manually.

A line search is used to find the step size α at the kth iteration such that

$$f(\alpha) = O(m_k + \alpha d_k), \tag{5.71}$$

is minimized along the search direction d_k. The minimizer can be found by setting the derivative of the function $f(\alpha)$ equal to zero, i.e.,

$$h(\alpha) \equiv f'(\alpha) = \frac{dO(m_k + \alpha d_k)}{d\alpha} = \big(\nabla O(m_k + \alpha d_k)\big)^T d_k = 0. \tag{5.72}$$

This equation can be solved by using the Newton-Raphson algorithm which is given by

$$\alpha_{j+1} = \alpha_j - \frac{h(\alpha_j)}{h'(\alpha_j)}, \tag{5.73}$$

where j denotes the index of the Newton-Raphson iteration and the first derivative of h can be evaluated by

$$h'(\alpha) = \frac{dh(\alpha)}{d\alpha} = d_k^T \nabla \Big[\big(\nabla O(m_k + \alpha d_k)\big)^T\Big] d_k = d_k^T H(m_k + \alpha d_k) d_k. \tag{5.74}$$

In an exact line search, the Newton-Raphson iteration is stopped when a convergence criterion is satisfied. The exact line search is very expensive due to the evaluation of the term $d_k^T H(m_k + \alpha d_k) d_k$ which requires at least one simulation run. In our procedure, we use an inexact line search. Specifically, we do only one Newton-Raphson iteration as mentioned previously. To perform one Newton-Raphson iteration, we set $\alpha_0 = 0$ and then Eq. 5.73 gives

$$\alpha_1 = -\frac{\big(\nabla O(m_k)\big)^T d_k}{d_k^T H(m_k) d_k}. \tag{5.75}$$

Eq. 5.75 involves the Hessian matrix which can be approximated. The Hessian for the objective function given by Eq. 2.14 is given by

$$H_k = G_k^T C_D^{-1} G_k + C_M^{-1}. \tag{5.76}$$

So

$$\begin{aligned} d_k^T H_k d_k &= d_k^T (G_k^T C_D^{-1} G_k + C_M^{-1}) d_k \\ &= d_k^T (G_k^T C_D^{-1} G_k) d_k + d_k^T C_M^{-1} d_k \\ &= (G_k d_k)^T C_D^{-1} (G_k d_k) + d_k^T C_M^{-1} d_k. \end{aligned} \tag{5.77}$$

In this equation, we do not need to compute the sensitivity coefficient matrix G directly. We only need to calculate Gd_k which can be done by using a finite-difference approximation as shown next. It could also be calculated using one run of the gradient simulator method. The method given below was originally implemented by Kalita (2000). The elements of the sensitivity coefficient matrix can be written as

$$G_{i,j} = \frac{\partial g_i}{\partial m_j}, \tag{5.78}$$

where $i = 1, \cdots, N_d$ and $j = 1, \cdots, N_m$. The directional derivative is

$$\Big(\frac{dg}{d\alpha}\Big)_{\alpha=0} = \Big(\frac{dg(m+\alpha d_k)}{d\alpha}\Big)_{\alpha=0}. \tag{5.79}$$

Let $u = d_k / \parallel d_k \parallel$. So we have

$$\begin{aligned} \Big(\frac{dg_i}{d\alpha}\Big)_{\alpha=0} &= [\nabla g_i(m)]^T u \\ &= \frac{1}{\parallel d_k \parallel} [\nabla g_i(m)]^T d_k. \end{aligned} \tag{5.80}$$

The ith component of Gd_k is given by

$$\begin{aligned} \big[Gd_k\big]_i &= \sum_{j=1}^{N_m} \frac{\partial g_i}{\partial m_j} d_{k,j} \\ &= [\nabla g_i(m)]^T d_k, \end{aligned} \tag{5.81}$$

where $d_{k,j}$ denotes the jth component of the vector d_k. Substituting Eq. 5.80 into

Eq. 5.81, we obtain

$$\begin{aligned} Gd_k =& \parallel d_k \parallel \left(\frac{dg}{d\alpha}\right)_{\alpha=0} \\ \approx& \parallel d_k \parallel \frac{g(m+\epsilon d_k) - g(m)}{\epsilon \parallel d_k \parallel} \\ =& \frac{g(m+\epsilon d_k) - g(m)}{\epsilon}, \end{aligned} \tag{5.82}$$

where ϵ is a small number. We choose ϵ based on the infinity norm of d_k such that ϵ satisfies $\epsilon \parallel d_k \parallel_\infty = 10^{-3}$. Note that calculating Gd_k needs one additional simulation run. Once we have Gd_k, it is straight forward to calculate $d_k^T H_k d_k$ using Eq. 5.77 and then Eq. 5.75 can be applied to calculate the step size. Application of Eq. 5.77 requires evaluating $C_M^{-1} d_k$. In our code, we provide two ways to calculate this term. One way is to solve a matrix problem

$$C_M x = d_k \tag{5.83}$$

for $x = C_M^{-1} d_k$ using either LU decomposition or preconditioned conjugate gradient method (both of them are available in our code). The other way is to approximate C_M^{-1} by using stencil method; see Skjervheim (2002) or Oliver (1998).

Fig. 5.1 presents plots of $O(m_k + \alpha d_k)$ (circles) and $f'(\alpha) = [\nabla O(m_k + \alpha d_k)]^T d_k$ (diamonds) versus α. The cross shows $f'(\alpha_1)$ where α_1 was computed by Eq. 5.75. Note $f'(\alpha_1)$ is close to zero and α_1 is close to the point ($\alpha = 3.765$) where $O(m_k + \alpha d_k)$ is minimum. This result illustrates our observation that one iteration of the Newton-Raphson algorithm usually yields a sufficient accurate line search so that the Wolfe conditions are satisfied.

The above procedure for computing $H(m_k)d_k$ was based on the particular form of the Hessian given by Eq. 5.76. For other cases, for example, the case where the prior means are considered as model parameters, the above procedure must be modified. A more general derivation based on Taylor's series is given below. The resulting formula can be applied to estimate directly the Hessian-vector product. For any vector valued function $f(m)$ (here, f is an N-dimensional column vector where each component is a function of m,) we can write a Taylor series approximation as

$$f(m + \epsilon v) = f(m) + \left(\nabla(f(m)^T)\right)^T \epsilon v, \tag{5.84}$$

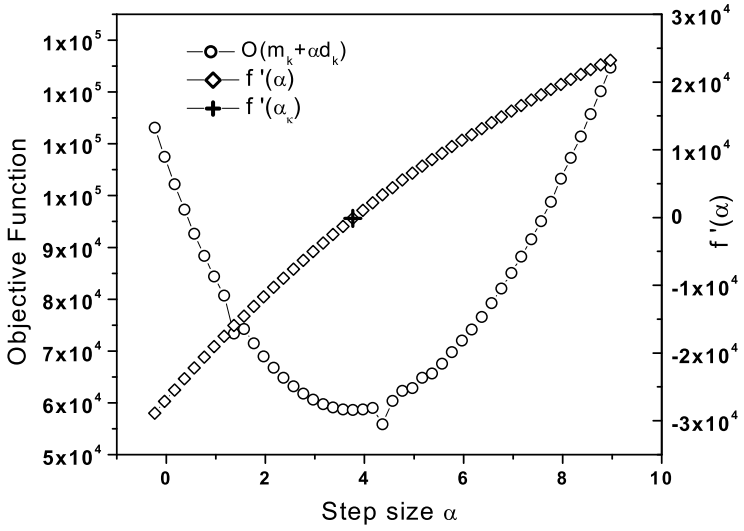


Fig. 5.1: Illustration of Newton-Raphson.

where ϵ is a real scalar and v and m are fixed N_m-dimensional vectors. If $f = g(m)$ represents calculated data, then

$$\left(\nabla(f(m)^T)\right)^T \epsilon v = \left(\nabla(g(m)^T)\right)^T \epsilon v = \epsilon G v \tag{5.85}$$

so

$$Gv = \frac{g(m+\epsilon v) - g(m)}{\epsilon}, \tag{5.86}$$

which gives a finite-difference method for calculating Gv. Another procedure for doing so is to use the gradient simulator method. Now suppose $f = \nabla O$, then Eq. 5.84 gives

$$\begin{aligned} \nabla O(m+\epsilon v) &= \nabla O(m) + \left(\nabla \cdot (\nabla O(m))^T\right)^T \epsilon v \\ &= \nabla O(m) + H^T \epsilon v \\ &= \nabla O(m) + H \epsilon v. \end{aligned} \tag{5.87}$$

The last equality follows from the fact that the Hessian is symmetric. Rearranging the last equation gives

$$Hv = \frac{\nabla O(m+\epsilon v) - \nabla O(m)}{\epsilon}. \tag{5.88}$$

The advantage of the last equation is that the H represents the Hessian for whatever objective function O we use. Application of Eq. 5.88 implies that the Hessian matrix used is the true Hessian given by Eq. 5.11 instead of the Hessian approximation given by Eq. 5.13. However, Eq. 5.77 makes use of the approximation form of the Hessian matrix, i.e., Eq. 5.13.

We choose ϵ by either

$$\epsilon = (1 + \|m\|_\infty)10^{-8} \tag{5.89}$$

or

$$\epsilon(1 + \|v\|_\infty) = (1 + \|m\|_\infty)10^{-k}, \tag{5.90}$$

where $3 \leq k \leq 5$. In our code, Eq. 5.89 was used to calculate ϵ. The application of Eq. 5.88 requires one forward simulation run to calculate the primary variables that are required to form the adjoint system and one adjoint solution to form the gradient evaluated at $m + \epsilon v$, whereas application Eq. 5.82 requires only a forward simulation run. Thus, we use Eq. 5.82 whenever the objective function is given by Eq. 2.14 or Eq. 2.18 and use Eq. 5.88 only for the case where we correct the prior mean. This case is discussed later.

As mentioned above, instead of doing an exact line search, we generate a sequence of iterate approximations to α that minimizes $f(\alpha)$ and terminate the iterations when the Wolfe conditions are satisfied. As discussed below, the Wolfe conditions were formulated to ensure that step sizes are not too small and that there is a non-negligible reduction in the objective function at each iteration. In addition, the Wolfe conditions are the side conditions for quadratic termination for linear problems in practice; see Kolda et al. (1998).

5.8 Wolfe Conditions

Following Fletcher (1987), let $\bar{\alpha}_k$ denote the smallest positive value of α for which $O(m_k + \alpha d_k) = O(m_k)$. Negligible reductions in the value of the objective function can occur either if $\alpha_k \to \bar{\alpha}_k$ or $\alpha_k \to 0$; see Fig. 5.2. Goldstein (1965)

conditions can be used to avoid the occurrence of these negligible reductions. Again we let $f(\alpha)$ denote $O(m_k + \alpha d_k)$, so $f(0) = O(m_k)$. The Goldstein conditions are

$$f(\alpha) \leq f(0) + \alpha\rho f'(0), \tag{5.91}$$

to exclude the right-hand side extreme of $[0, \bar{\alpha}_k]$, and

$$f(\alpha) \geq f(0) + \alpha(1-\rho)f'(0) \tag{5.92}$$

to exclude the left-hand side extreme of $[0, \bar{\alpha}_k]$, where $\rho \in (0, \frac{1}{2})$ is a fixed parameter. In our examples, we choose $\rho = 0.0001$. Eq. 5.91 is often also referred to as a Wolfe condition. In Figs. 5.2 and 5.3, the line labeled $\rho f'(0)$ goes through $f(0)$ and its slope is equal to $\rho f'(0)$; the line labeled $f'(0)$ goes through $f(0)$ and has slope equal to $f'(0)$. Eq. 5.91 can be rewritten as

$$\frac{f(\alpha) - f(0)}{\alpha} \leq \rho f'(0). \tag{5.93}$$

From Eq. 5.93, we can see that if the step size satisfies this condition then the line through $(0, f(0))$ and $(\alpha, f(\alpha))$ in Fig. 5.2 must be below the line with slope of $\rho f'(0)$. Similarly, Eq. 5.92 can be rewritten as

$$\frac{f(\alpha) - f(0)}{\alpha} \geq (1-\rho)f'(0). \tag{5.94}$$

If the step size satisfies this condition then the line through $(0, f(0))$ and $(\alpha, f(\alpha))$ must be above the line labeled $(1-\rho)f'(0)$ in Fig. 5.3. Hence, if the step size satisfies both Eq. 5.91 and Eq. 5.92, then the line through $(0, f(0))$ and $(\alpha, f(\alpha))$ must fall between the line with slope of $\rho f'(0)$ and the line with slope of $(1-\rho)f'(0)$.

Applying the fact that $f'(0) = g_k^T d_k$ where $g_k = \nabla O(m_k)$, Eq. 5.91 can be rewritten as

$$f_k - f_{k+1} \geq -\alpha\rho g_k^T d_k. \tag{5.95}$$

When $f(\alpha)$ is non-quadratic, the second Goldstein condition (Eq. 5.92) may exclude the minimum point of $f(\alpha)$; see Fig. 5.3. Wolfe (1969) replaced Eq. 5.92 by a new condition which is given by

$$f'(\alpha) \geq \eta f'(0), \tag{5.96}$$

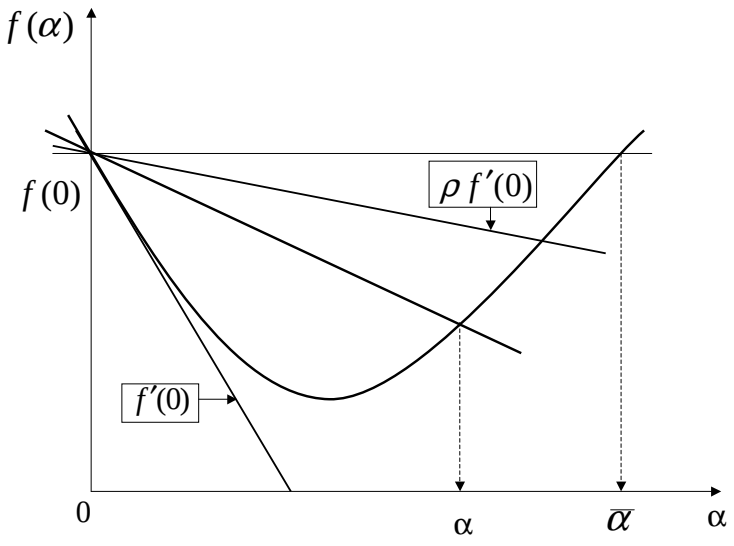


Fig. 5.2: Illustration of the Goldstein or the first Wolfe condition.

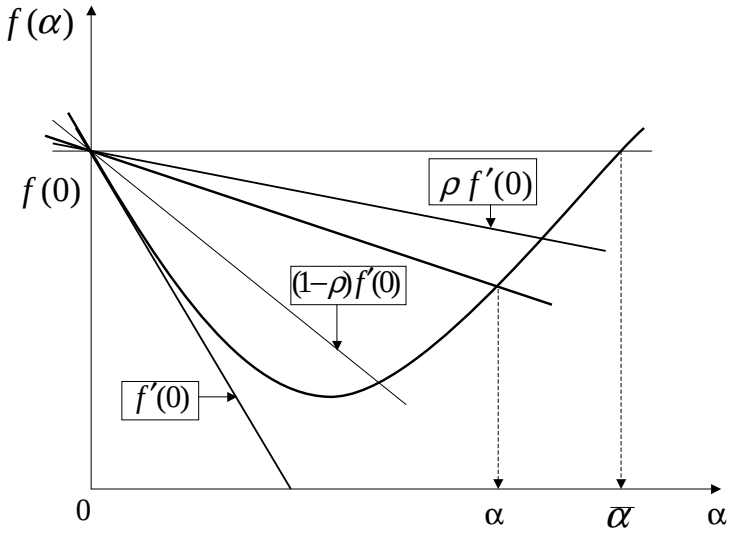


Fig. 5.3: Illustration of the second Goldstein condition.

where $\eta < 1$. Eq. 5.96 can be rewritten as

$$g_{k+1}^T d_k \geq \eta g_k^T d_k, \tag{5.97}$$

which is called Wolfe's condition. In practice, Eq. 5.97 is often replaced by

$$\mid g_{k+1}^T d_k \mid \leq \eta \mid g_k^T d_k \mid . \tag{5.98}$$

which is called the strong Wolfe condition. The reason for using Eq. 5.98 instead of Eq. 5.97 is given below, also see Fletcher (1987) for details. In our examples we use $\eta = 0.25$. In Fig. 5.4, the top dashed line shows a situation where Eq. 5.97 is satisfied but the strong Wolfe condition Eq. 5.98 is not satisfied. This dashed line, which intersects the objective function curve at the point $(\tilde{\alpha}, f(\tilde{\alpha}))$, falls below the line with slope of $\rho f'(0)$. At the point $(\tilde{\alpha}, f(\tilde{\alpha}))$ where this dashed line intersects the objective function, the slope of the objective function which is $g_{k+1}^T d_k$ is greater than $-\eta f'(0)$. With $m_{k+1} = m_k + \tilde{\alpha} d_k$, Eq. 5.97 is satisfied, due to the fact that $f'(\tilde{\alpha})$ is positive whereas $f'(0)$ is negative. However, the strong Wolfe condition (Eq. 5.98) is not satisfied at $\alpha = \tilde{\alpha}$, because the value of $f'(\tilde{\alpha}) > \eta f'(0)$. To satisfy the strong Wolfe condition, we have to move the dashed line toward the minimum until it falls below the dot dashed line which intersects the objective function curve at the point at which the derivative is equal to $-\eta f'(0)$. We use $\eta = 0.25$ in our work. So the strong Wolfe condition is more restrictive. The first condition (Eq. 5.95) ensures that the objective function is reduced sufficiently, and the second condition (Eq. 5.98) prevents the steps from being too small. As is standard, we simply refer to Eqs. 5.95 and 5.98 as the Wolfe conditions.

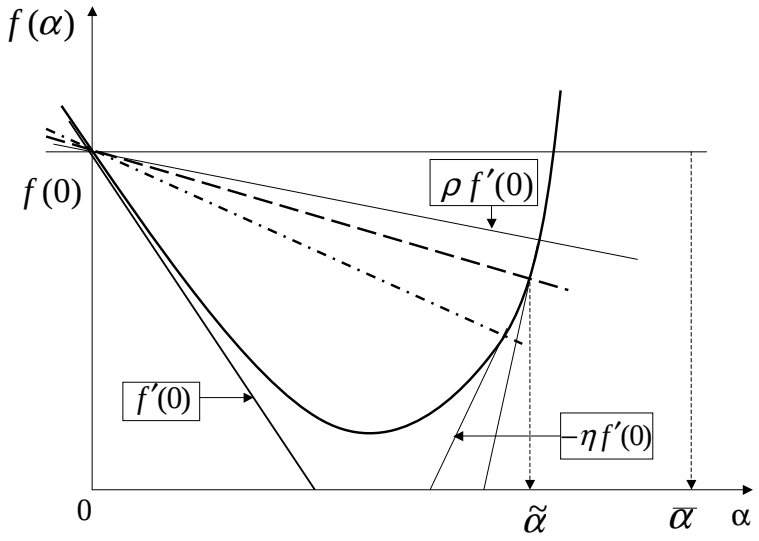


Fig. 5.4: Illustration of the strong Wolfe condition.

5.9 Quadratic Fit

Suppose at iteration k, we perform one Newton-Raphson iteration and find a step size $\hat{\alpha}_k$ for the current search direction d_k. With this step size, we calculate a new objective function value $O(m_k + \hat{\alpha}_k d_k) = f(\hat{m}_{k+1})$ and check the Wolfe conditions. If the Wolfe conditions are not satisfied, we fit the function $f(\alpha) = O(m_k + \alpha d_k)$ with a quadratic function given by

$$q(\alpha) = a\alpha^2 + b\alpha + c. \tag{5.99}$$

With the known values of $q(0) = f(m_k)$, $q'(0) = f'(m_k)$ and $q(\hat{\alpha}_k) = f(m_k + \hat{\alpha}_k d_k)$, we find

$$\begin{aligned} a &= \frac{f(m_k + \hat{\alpha}_k d_k) - f'(m_k)\hat{\alpha}_k - f(m_k)}{\hat{\alpha}_k^2}, \\ b &= f'(m_k), \\ c &= f(m_k). \end{aligned}$$

Minimizing q gives

$$\alpha_k = -\frac{f'(m_k)\hat{\alpha}_k^2}{2[f(m_k + \hat{\alpha}_k d_k) - f'(m_k)\hat{\alpha}_k - f(m_k)]}, \tag{5.100}$$

which is used as the new step size. Based on our experience, the quadratic fit almost always results in a decrease in the objective function. Even though it rarely happens, however, a quadratic fit may yield a model at which the objective function value is bigger than the value of the objective function corresponding to the model obtained by one Newton-Raphson iteration. Such a situation is depicted in Fig. 5.5 where the quadratic approximation to the true objective function is inaccurate near the minimum. In this figure, point A corresponds to m_k; point B corresponds to $m_k + \tilde{\alpha} d_k$ obtained after the Newton-Raphson iteration and point C corresponds to $m_k + \alpha_k d_k$ which is obtained after quadratic fit. We see that the value of the objective function at point C obtained from the quadratic fit is larger than the value of the objective function at point B obtained by one Newton-Raphson iteration.

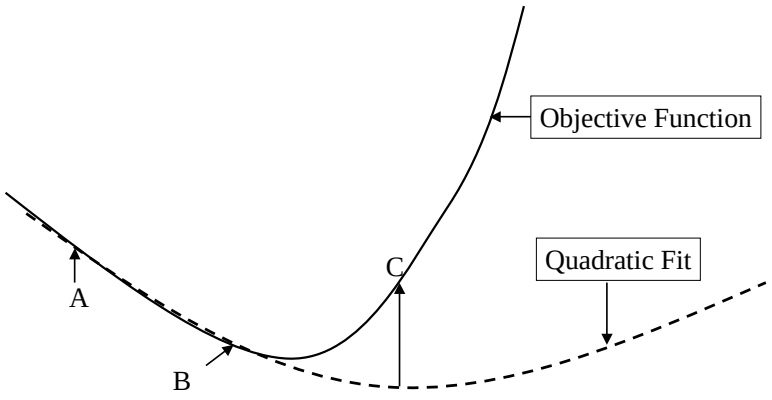


Fig. 5.5: Illustration of quadratic fit.

5.10 Evaluation of Computational Efficiency

Here, we assess the computational efficiency of GN (Gauss-Newton), LM (Levenberg-Marquardt), PCG (preconditioned conjugate gradient), BFGS and LBFGS. In the evaluation of computational efficiency, we count only the number of adjoint solutions and the number of reservoir simulation runs required by each method. Moreover, we count one adjoint solution over the total time interval of a simulation run as one equivalent simulation run although in our examples, one adjoint solution typically takes less than one half of the time of a simulation run. We do not keep track of the computational effort incurred when a proposed model update is rejected because it results in a violation of the Wolfe conditions.

In GN and LM, if the data are evenly distributed in the time domain, the computational cost of calculating sensitivity of all data to all model parameters requires $N_d/2$ adjoint solutions which we count as being equivalent to $N_d/2$ simulation runs. GN and LM require one additional simulation run to calculate the new value of the objective function. So a total of $N_d/2 + 1$ simulation runs are needed to accomplish one GN or LM iteration.

In LBFGS and PCG, the total computational cost of implementing one iteration is equivalent to 3 simulation runs, which include one equivalent simulation run for calculating the gradient of the objective function by using the adjoint method, one simulation run for calculating the step size when using Newton-Raphson iteration (only one iteration of Newton-Raphson is done in our implementation) and another simulation run for calculating the objective function. Thus, LBFGS and PCG are $(N_d/2+1)/3$ times faster than GN and LM for each iteration. For example, if we have 1000 data, LBFGS and PCG will be roughly 167 times faster than GN or LM per iteration. In terms of the total time, if GN or LM require n_{GN} iterations to converge on average, while LBFGS or PCG need n_{BFGS} iterations to converge on average, then LBFGS or PCG will be l times faster than GN or LM where

$$l = \frac{n_{GN}}{\frac{n_{BFGS}}{(N_d/2+1)/3}} = \frac{n_{GN}}{n_{BFGS}} \times \frac{N_d/2+1}{3}. \tag{5.101}$$

Although BFGS requires more time than LBFGS and PCG to perform the matrix operations involved in the update equation, it is the memory requirement that makes the standard BFGS method inferior to LBFGS and PCG for large scale problems. Hence, the "standard" BFGS refers to using Eq. 5.50 and storing $\tilde{H}_k^{-1}$. We could of course implement BFGS in exactly the same way as LBFGS in which case we do not explicitly compute or store $\tilde{H}_k^{-1}$.

5.11 Comparison of Memory Requirements

For large scale problems, the memory required by an optimization algorithm is a key issue that needs to be considered. Because we are only concerned with the difference between algorithms, we only consider the memory used by the optimization algorithm itself. Table 5.2 gives a rough estimate of the number of double precision real numbers used by each algorithm when applied to minimize the objective function of Eq. 2.14 or Eq. 2.18. Recall that N_d is the number of production data, N_m is the number of model parameters, and L is the number of previous vectors used in the LBFGS algorithm. For convenience, we use one memory unit to stand for the memory occupied by one double precision real number. Recall the dimension of m_k, m_{k+1},

δm_{k+1}, s, d_k, g_k, g_{k+1}, m_{prior} and $\nabla_m O$ is N_m, the dimension of a sensitivity coefficient matrix G is $N_d \times N_m$ and the dimension of C_M is $N_m \times N_m$. In the GN or LM method, $(4 + 2N_d) \times N_m$ (m_k, m_{k+1}, δm, m_{prior}, sensitivity coefficient matrix G and $C_M G^T$) memory units are used. In CG, $8 \times N_m$ ($8 \times N_m$: m_k, m_{k+1}, δm_{k+1}, s_k, d_k, g_{k+1}, m_{prior}, $\nabla_m O$) memory units are used. For PCG, in addition to the memory required for the standard conjugate gradient, memory is required to store the preconditioner. In BFGS $(10+N_m) \times N_m$ ($10 \times N_m$: m_k, m_{k+1}, δm_{k+1}, d_k, g_k, g_{k+1}, m_{prior}, v_k, $\tilde{H}_k^{-1} y_k$, y_k; $N_m \times N_m$: $\tilde{H}_k^{-1}$) memory units are used. In LBFGS, $(7 + 2 \times L) \times N_m$ ($7 \times N_m$: m_k, m_{k+1}, δm_{k+1}, g_k, d_k, m_{prior}, diagonal inverse Hessian approximation; $2 \times L \times N_m$: y_k and s_k for $k = 1, 2, \cdots, L$) memory units are used. From the results of Table 5.2, we see that the full-memory version of BFGS uses the most memory which is on the order of N_m^2, the standard conjugate gradient method uses the least memory which is on the order of N_m and Gauss-Newton or Levenberg-Marquardt and limited memory BFGS have intermediate memory requirements. For large scale problems in which the number of data and the number of model parameters are both large, the memory used by limited memory BFGS depends on the number of previous vectors (denoted by L in Table 5.2) used to construct the Hessian inverse approximation update; L must be specified by the user. Fig. 5.6 shows a snapshot of the panel which monitors the memory usage history and the CPU usage history when the LBFGS algorithm was applied to do a history match for a 2D problem presented in Chapter VI. In the right bottom black window, the curve shows the memory usage. The period of high memory usage corresponds to a simulation run and the lower memory period corresponds to solving an adjoint system.

5.12 Optimization for Doubly Stochastic Model

In this section, we will consider the optimization algorithms for the doubly stochastic model specifically where the objective function is given by Eq. 2.29 which

Table 5.2: Memory used by each algorithm.

	No. of DP real numbers
GN/LM	$(4 + 2 \times N_d) \times N_m$
CG	$8 \times N_m$
PCG	$8 \times N_m$ + memory for preconditioner
BFGS	$(10 + N_m) \times N_m$
LBFGS	$(7 + 2L) \times N_m$

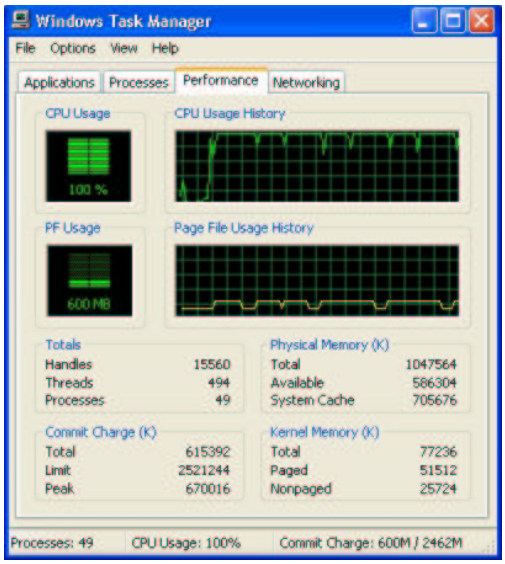


Fig. 5.6: Memory and CPU usage history of the computer.

is repeated as

$$O(m,\alpha) = \frac{1}{2}(g(m) - d_{\text{obs}})^T C_D^{-1}(g(m) - d_{\text{obs}}) + \frac{1}{2}(m - m_{\text{prior}} - E\alpha)^T C_M^{-1}(m - m_{\text{prior}} - E\alpha) + \frac{1}{2}(\alpha - \alpha_0)^T C_\alpha^{-1}(\alpha - \alpha_0). \quad (5.102)$$

Recall that $\hat{M}$ denotes the random vector including the model vector M and vector α, i.e.,

$$\hat{M} = \begin{bmatrix} M \\ \alpha \end{bmatrix}, \quad (5.103)$$

and $\hat{m}$ denotes a realization of $\hat{M}$. We can apply the methods that require both the Hessian and gradient, such as Gauss-Newton or Levenberg-Marquardt, or the methods that require only the gradient, such as quasi-Newton or conjugate gradient.

5.12.1 Application of Gauss-Newton/Levenberg-Marquardt Methods

The gradient of the objective function can be partitioned as

$$\nabla O(\hat{m}) = \begin{bmatrix} \nabla_m O(\hat{m}) \\ \nabla_\alpha O(\hat{m}) \end{bmatrix}. \tag{5.104}$$

The gradient of the objective function with respect to m and α are given by

$$\nabla_m O(\hat{m}) = G^T C_D^{-1}(g(m) - d_{\text{obs}}) + C_M^{-1}(m - m_{\text{prior}} - E\alpha), \tag{5.105}$$

and

$$\nabla_\alpha O(\hat{m}) = -E^T C_M^{-1}(m - m_{\text{prior}} - E\alpha) + C_\alpha^{-1}(\alpha - \alpha_0), \tag{5.106}$$

respectively. The first term in the right-hand side of Eq. 5.105 can be obtained by the adjoint method. The second term in the right-hand side of the same equation can be obtained by solving the matrix problem using the conjugate gradient method. The two terms in the right-hand side of Eq. 5.106 can be obtained by solving two matrix problems respectively. One is an $N_m \times N_m$ matrix problem. The other one is $N_\alpha \times N_\alpha$ matrix problem. Note that $N_\alpha \ll N_m$.

The Hessian matrix is given by

$$H = \begin{bmatrix} \nabla_m\big(\nabla_m O(\hat{m})\big)^T & \nabla_m\big(\nabla_\alpha O(\hat{m})\big)^T \\ \nabla_\alpha\big(\nabla_m O(\hat{m})\big)^T & \nabla_\alpha\big(\nabla_\alpha O(\hat{m})\big)^T \end{bmatrix} = \begin{bmatrix} G^T C_D^{-1} G + C_M^{-1} & -C_M^{-1} E \\ -E^T C_M^{-1} & E^T C_M^{-1} E + C_\alpha^{-1} \end{bmatrix}, \tag{5.107}$$

where the last equality is actually approximation because we have used the Gauss-Newton approximation of the Hessian. The Gauss-Newton method is given by

$$H_k \delta\hat{m}_{k+1} = -\nabla_{\hat{m}} O(\hat{m}) \tag{5.108}$$

where k denotes the iterative index. If we delete the off diagonal entries of Hessian matrix, then we obtain the approximation

$$\hat{H}_k = \begin{bmatrix} G^T C_D^{-1} G + C_M^{-1} & O \\ O & E^T C_M^{-1} E + C_\alpha^{-1} \end{bmatrix}. \tag{5.109}$$

When $\hat{H}_k$ is used as the modified Hessian in the Gauss-Newton iteration procedure, the overall iteration can be decomposed as follows:

$$(G_k^T C_D^{-1} G_k + C_M^{-1})\delta m_{k+1} = -G_k^T C_D^{-1}\big(g(m) - d_{\text{obs}}\big) - C_M^{-1}(m_k - m_{\text{prior}} - E\alpha_k) \quad (5.110)$$

$$(E^T C_M^{-1} E + C_\alpha^{-1})\delta\alpha_{k+1} = E^T C_M^{-1}(m_k - m_{\text{prior}} - E\alpha_k) - C_\alpha^{-1}(\alpha - \alpha_0) \quad (5.111)$$

$$m_{k+1} = m_k + \mu_k \delta m_{k+1} \quad (5.112)$$

$$\alpha_{k+1} = \alpha_k + \mu_k \delta\alpha_{k+1} \quad (5.113)$$

where μ_k is calculated by the restricted step scheme; see Fletcher (1987). Note in the spirit of the restricted step, it is important to use the same value of μ_k in both Eqs. 5.112 and 5.113, otherwise we effectively change the search direction. Note by replacing H by $\hat{H}$, we avoid inversion of H, i.e., we have "decoupled" the iteration on the model m from the iteration on the correction α to the prior mean. Eq. 5.110 is essentially the same equation we used when the prior mean is fixed and we can apply any of the techniques mentioned earlier to solve it, including applying matrix inversion lemmas. In Eq. 5.111, the dimension of α is $N_\alpha \times N_\alpha$ and if porosity and all log-permeability fields are modeled as stationary random functions throughout the domain, then the dimension of α is at most 4 so the matrix on the left side of Eq. 5.111 is of lower order. However, to form the matrix on the left-hand side of Eq. 5.111 appears to require calculation of C_M^{-1} and then doing the matrix multiplication. The product of $C_M^{-1}E$ is calculated by computing $C_M^{-1}e_l$ where e_l is the lth column of E. This is done by solving $C_M w_l = e_l$ using a preconditioned conjugate gradient method.

5.12.2 Application of Quasi-Newton Method

In this method, the gradient required is given by Eqs. 5.105 and 5.106. The step size can be obtained by applying Newton-Raphson iterative procedure using the basic formula of Eq. 5.88. The initial Hessian inverse approximation used for this case is given by

$$\tilde{\hat{H}}_0^{-1} = \begin{bmatrix} C_M & O \\ O & C_\alpha \end{bmatrix}. \quad (5.114)$$

Once we have the gradient of the objective function, the step size and the initial Hessian inverse approximation, we can use the same LBFGS algorithm as used for the case without correcting the prior mean to minimize Eq. 2.29.

CHAPTER VI

EXAMPLES

In this chapter, several history matching examples are presented. The first example is a 3D single-phase gas problem. Five optimization algorithms Gauss-Newton (GN), Levenberg-Marquardt (LM), preconditioned conjugate gradient (PCG), Broyden-Fletcher-Goadfarb-Shanno (BFGS) and limited memory Broyden-Fletcher-Goadfarb-Shanno (LBFGS) are tested on this example. The convergence behavior and the computational cost are compared for the five algorithms. Extensive options for choosing the initial inverse Hessian approximation and for choosing scaling schemes are explored and analyzed for the BFGS and LBFGS algorithms. Among of all these options, we recognized the best options for choosing the scaling factor and initial Hessian inverse approximation for both BFGS and LBFGS for this problem.

The second example is a 2D three-phase synthetic example. The purpose of this example is to further test and verify the points that we observed from the single-phase gas example on the three-phase problem. The same optimization algorithms are tested and analyzed on this example. Basically, the same conclusions can be made except that the LBFGS-PCG (preconditioned conjugate gradient using a preconditioner which is an approximation to $(\tilde{H}_k^{-1})^{-1}$ where $\tilde{H}_k^{-1}$ is the inverse Hessian approximation) does not work as well as in the single-phase gas example. For both examples, LBFGS with the "optimum" scaling option works very well. Then the LBFGS was applied to a 3D three-phase history matching problem. The results are encouraging. Based on all these examples we have done, we believe that LBFGS is the best minimization algorithm for large scale problems.

Finally, we applied LBFGS to a pseudo-field example. The example is a synthetic example which was constructed using data and information from the Os-

eberg reservoir which is located in the Norwegian sector of the North Sea. There is a gas cap at the top and an aquifer at the bottom. Two gas injection wells are located in the gas cap. Five producing wells are located above the aquifer. Wellbore pressure from all seven wells and the GOR from the five producing wells are history matched. We obtained a very good match for both types of data.

6.1 Three-Dimensional Single-Phase Gas Synthetic Example

This example pertains to flow in a 3D single-phase gas reservoir. Reservoir simulation was done on a $20 \times 20 \times 4$ grid. The dimension of the reservoir is 2000 ft $\times$ 2000 ft $\times$ 40 ft. The gridblock size is actually uniform with $\Delta x = \Delta y = 100$ ft and $\Delta z = 10$ ft. A spherical variogram was used to generate the prior covariance matrix. The correlated lengths in the $x-$, $y-$ and $z-$ direction are 400 ft, 200 ft and 10 ft, respectively. Horizontal permeability, vertical permeability, porosity and skin factor are the model parameters for this example. We assume that the porosity field is correlated with the horizonal log-permeability field and the correlation coefficient is 0.7 and that the vertical log-permeability field is uncorrelated with the porosity and horizontal permeability fields. The prior information for the model parameters are given in Table 6.1 where s denotes the skin factors at all wells. Because of the small variance on s, the skin factor is almost fixed.

Table 6.1: Prior information on model parameters.

	Mean	Variance
$\ln(k_x)$	4.0	0.5
$\ln(k_z)$	-2.9	0.5
ϕ	0.25	0.002
s_{skin}	4.0	0.0001

The initial pressure is 3230 psi. All six boundaries are assumed to be no-flow boundaries. The reservoir is produced by two completely penetrating wells. Well 1 is

located in areal gridblock (5, 5) and well 2 is located in areal gridblock (15, 15). Well 1 was shut in for two days and then was produced at the rate of 4×10^4 Mscf/day for another two days. Well 2 produced at the rate of 3.5×10^4 Mscf/day for the first two days and was then shut in for the following two days. Fig. 6.1 shows the pressure response of the two wells. We used 22 measured data from each well as conditioning data. Thus, the total number of data to be history matched is 44. The observed data are obtained by adding random noise to the simulated pressure data predicted from the true reservoir. The variance for the observed pressure data is specified as 1 psi^2. The total number of reservoir variables is 4808. Note that each layer has a skin factor at each well.

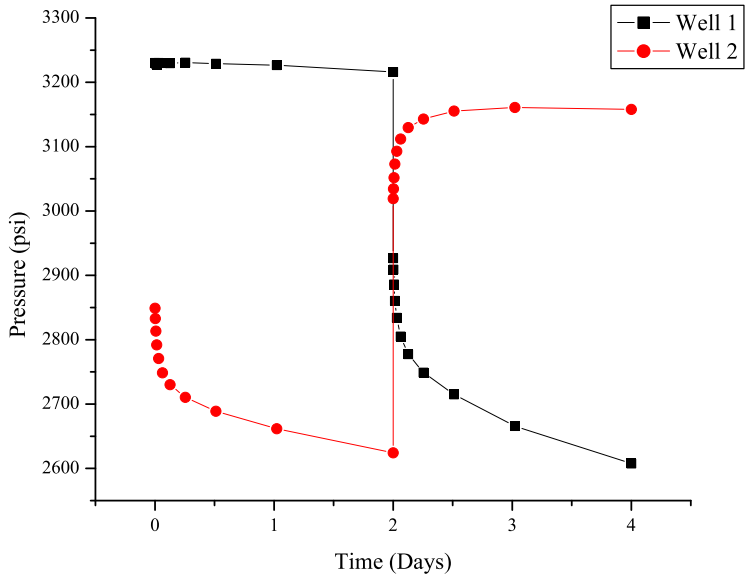


Fig. 6.1: Pressure response from the true model.

Stochastic simulation was done using randomized maximum likelihood method. Ten realizations were generated using five different optimization algorithms: (i) Gauss-Newton (GN) with restricted step, (ii) Levenberg-Marquardt (LM), (iii) preconditioned conjugate gradient (PCG), (iv) BFGS and (v) limited memory BFGS (LBFGS). In LM, we simply use the value of 1000 for the initial damping factor and multiply by 10 when the objective function increases and divide by 10 when the objective function decreases. In PCG, the full matrix C_M^{-1} was chosen as the precon-

ditioner and fixed for each iteration. In BFGS, the full prior covariance matrix C_M was used as the initial inverse Hessian approximation. In BFGS, we only scale the initial inverse Hessian approximation and it is scaled by a factor of $\gamma_0 = s_0^T y_0 / y_0^T \tilde{H}_0^{-1} y_0$ where s_0 and y_0 are obtained at the first iteration. The LBFGS we used is the algorithm proposed by Nocedal (1980). In the LBFGS method, at most 30 previous vectors were used to construct the inverse Hessian approximation $\tilde{H}_0^{-1}$, and at each iteration, $\tilde{H}_0^{-1}$, which is an identity matrix, is scaled by a factor of $s_k^T y_k / (y_k^T y_k)$ where k represents the kth iteration. For the GN and LM methods, only Eq. 5.69 with $\varepsilon_1 = 10^{-3}$ was used as a stopping criterion to terminate the iteration. When the same stopping criterion was used for the other algorithms, most of the "realizations" obtained at convergence gave very high values of the objective function; see Table 6.2. In order to obtain a smaller value of the objective function at convergence, we used $\varepsilon_1 = 10^{-7}$ in Eq. 5.69 as the convergence criterion for PCG, BFGS and LBFGS. All five algorithms were applied to the same 10 unconditional realizations of data and the model, when doing history matching, i.e., the same m_{uc} and d_{uc} were used in the objective function of Eq. 2.18. The observed and unconditional data are the same for each algorithm.

Fig. 6.2(a) through Fig. 6.2(e) show the behavior of the objective function iteration by iteration for each algorithm based on using $\varepsilon_1 = 10^{-3}$ for GN and LM algorithms and $\varepsilon_1 = 10^{-7}$ for the other algorithms. Each curve on each figure corresponds to one realization. Table 6.3 shows the objective function values at convergence and the number of iterations required to obtain convergence. In terms of the number of iterations, the GN and LM methods are the best algorithms and the objective function converges to a small value (approximately 33 or so) for each of the 10 realizations. Both the GN and LM methods, however, require considerable computational work at each iteration due to evaluating the sensitivity of each data to all model parameters. From the results shown in Table 6.3, we can make the following observations:

1. In terms of total machine time, algorithms which only require the gradient of the objective function (especially LBFGS and PCG) are much faster than GN

Table 6.2: Comparison of BFGS, LBFGS and PCG, $\varepsilon_1 = 10^{-3}$.

	BFGS		LBFGS		PCG	
Real. No.	Obj.	No. Iter.	Obj.	No. Iter.	Obj.	No. Iter.
R1	65	17	69	25	153	12
R2	372	10	F	F	146	18
R3	56	15	63	17	70	11
R4	86	11	171	25	263	10
R5	58	11	138	25	347	13
R6	77	15	68	25	112	25
R7	64	25	66	10	184	5
R8	113	19	245	22	213	17
R9	445	7	112	20	230	19
R10	47	25	60	22	45	23
Average	138	15.5	99	21.2	176	15.3

and LM methods. As discussed previously, theoretically, LBFGS and PCG should be roughly $[N_d/2+1]/3$ times faster than GN and LM per iteration. In this example, we history matched 44 data to generate each realization. Thus, LBFGS and PCG algorithms should be $[N_d/2+1]/3 \approx 8$ times faster than GN and LM per iteration.

2. PCG converges to higher values of the objective function than does the LBFGS method under the same stopping criterion but requires fewer iterations for convergence.

3. Based on our previous discussions, we believe that the average value of the objective function at the minima should be around $N_d = 44$. However the average objective function value at convergence is higher than this value for all methods except GN and LM. Moreover, the average objective function value for

the preconditioned conjugate gradient method exceeds $N_d + 5\sqrt{2N_d} = 91$; see the discussion in section 2.3. Given that significantly higher average value of the objective function were obtained for BFGS, LBFGS and PCG than for GN and LM, it is not clear that these four methods converge to appropriate minima of $O(m)$. Thus we seek modified algorithms that will have better convergence properties.

Table 6.3: Comparison between algorithms.

		R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	Average
GN	Obj.	37	31	21	33	43	28	27	38	40	33	33.1
	No. Iter.	7	13	6	8	12	8	12	12	6	6	9
LM	Obj.	38	30	21	33	43	28	27	38	40	34	33.2
	No. Iter.	8	14	12	8	13	13	21	8	14	10	12
BFGS	Obj.	77	43	33	86	52	33	39	114	103	41	62.1
	No. Iter.	13	47	25	12	22	76	41	19	22	38	31.5
LBFGS	Obj.	53	47	63	131	138	44	66	174	87	42	84.5
	No. Iter.	36	41	17	42	27	50	10	39	34	43	33.9
PCG	Obj.	153	146	70	94	347	42	184	213	230	45	152
	No. Iter.	12	20	11	23	19	64	5	17	19	23	21.3

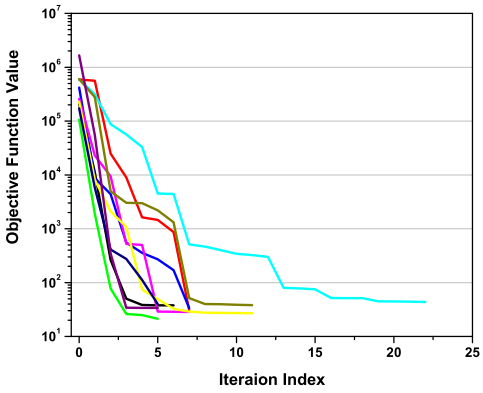
6.1.1 Scaling Effects on BFGS

The following three different scaling options were tested for the BFGS. The full matrix C_M was used as the initial Hessian approximation $\tilde{H}_0^{-1}$ in all options.

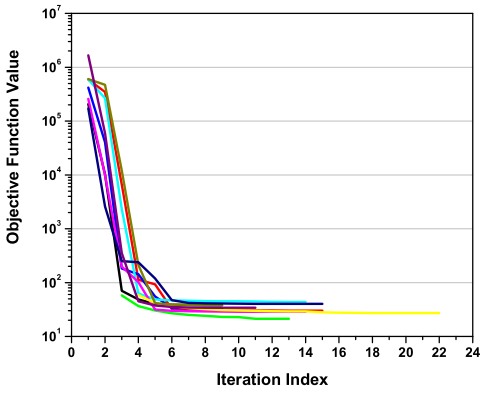
BFGS-Opt1: Use only initial scaling and scale $\tilde{H}_0^{-1}$ by the factor of

$$\gamma_0 = \sigma_0 = s_0^T y_0 / (y_0^T \tilde{H}_0^{-1} y_0). \tag{6.1}$$

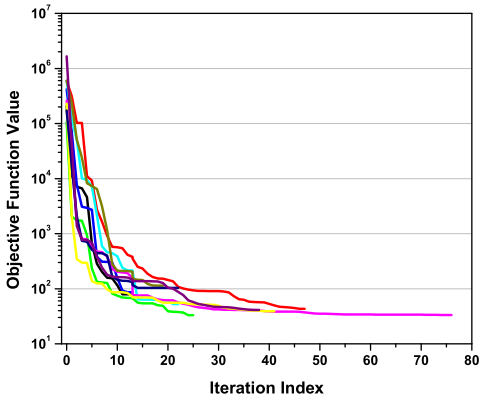
This option was suggested by Shanno and Phua (1978) and used by Yang and Watson (1988) in their work. This option is the one we used previously (see Table 6.3).



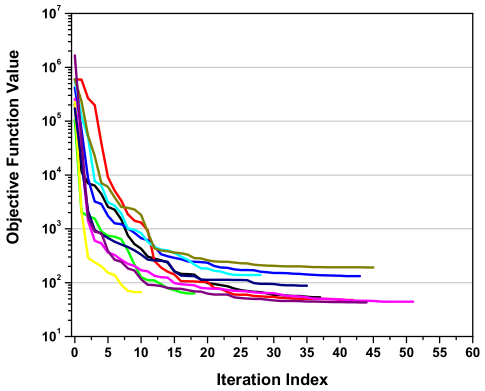
(a) Gauss-Newton



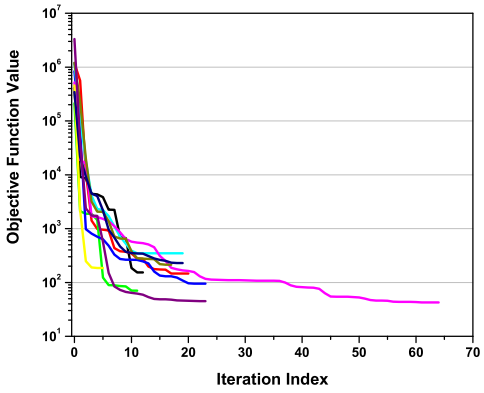
(b) Levenberg-Marquardt



(c) BFGS



(d) LBFGS



(e) PCG

Fig. 6.2: Behavior of objective function for different optimization methods.

BFGS-Opt2: Scale at each iteration with the scaling factor chosen by the following scheme:

$$\gamma_k = \begin{cases} \sigma_k \quad \text{and} \quad \theta_k = 1 & \text{for} \quad \sigma_k > 1 \\ \tau_k \quad \text{and} \quad \theta_k = 0 & \text{for} \quad \tau_k < 1 \\ 1 \quad \text{and} \quad \theta_k = \dfrac{1-\sigma_k}{\tau_k - \sigma_k} & \text{for} \quad \sigma_k \leq 1 \leq \tau_k \end{cases} \tag{6.2}$$

where σ_k and τ_k are given by Eqs. 5.57 and 5.56, respectively. This option was suggested by Oren (1974a).

BFGS-Opt3: Use only initial scaling and choose the scaling factor by the scheme:

$$\gamma_0 = \begin{cases} \tau_0 & \text{for} \quad \tau_0 < 1 \\ \sigma_0 & \text{otherwise} \end{cases} \tag{6.3}$$

This option is a modification of BFGS-Opt2.

BFGS-Opt4: No scaling

In BFGS-Opt2, when $\sigma_k \leq 1 \leq \tau_k$ is satisfied, both γ_k and θ_k are calculated. This is the only case where θ_k is computed by an equation. This equation forces θ_k to be in the interval between 0 and 1. We know when $\theta_k = 0$, Eq. 5.48 reduces to the DFP formula and when $\theta_k = 1$, it reduces to the BFGS formula. As we discussed in the Appendix B, the DFP algorithm is inferior to BFGS. It is not clear whether or not the algorithm with $\theta_k < 1$ is inferior to the BFGS algorithm in which $\theta_k = 1$. However, our results show that the BFGS-Opt3 is as good as BFGS-Opt2. The results are summarized in Table 6.4. Note that BFGS-Opt3 is slightly superior to BFGS-Opt1. On average BFGS-Opt2 requires 4 fewer iterations than BFGS-Opt3 to obtain convergence, but it results in a higher value of the objective function at convergence. Thus, at this point, we believe that BFGS-Opt3 is the better choice for scaling in the BFGS algorithm. Comparing BFGS-Opt3 with BFGS-Opt4, we can see that scaling does not have apparent improvement on the convergence behavior for this particular single-phase gas problem. At least in terms of the value of the objective function at convergence, however, Option 2 seems to perform the worst.

Table 6.4: Comparison of different scaling options in BFGS algorithm.

		R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	Average
BFGS-Opt1	Obj.	77	43	33	86	52	33	39	114	103	41	62.1
	No. Iter.	13	47	25	12	22	76	41	19	22	38	31.5
BFGS-Opt2	Obj.	89	78	53	86	85	38	70	104	62	41	70.6
	No. Iter.	15	22	20	15	24	48	21	18	46	24	25.3
BFGS-Opt3	Obj.	77	50	33	43	56	36	43	105	64	38	54.5
	No. Iter.	13	51	25	38	17	41	29	14	32	31	29.1
BFGS-Opt4	Obj.	59	52	35	41	70	41	39	53	54	87	53
	No. Iter.	33	17	18	19	33	38	18	42	35	52	30.5

6.1.2 Scaling Effects on LBFGS

The following options are used to test the effect of the different initial inverse Hessian approximation and different scaling schemes on the behavior of LBFGS.

LBFGS-Opt1: Use the identity matrix as the initial inverse Hessian approximation; scale $\tilde{H}_0^{-1}$ for each iteration by a factor γ_k which is given by

$$\gamma_k = \tilde{\sigma}_k, \tag{6.4}$$

where $\tilde{\sigma}_k$ is given by Eq. 5.65 with $\tilde{H}_0^{-1} = I$.

LBFGS-Opt2: Use the identity matrix as the initial inverse Hessian approximation; scale $\tilde{H}_0^{-1}$ for each iteration by a factor γ_k which is determined by the following scheme:

$$\gamma_k = \begin{cases} \tilde{\tau}_{3k} & \text{for} \quad \tilde{\tau}_{3k} < 1 \\ \tilde{\sigma}_k & \text{otherwise} \end{cases} \tag{6.5}$$

where $\tilde{\tau}_{3k}$ is given by Eq. 5.67 with $\tilde{H}_0^{-1} = I$.

LBFGS-Opt3: Use the identity matrix as the initial inverse Hessian approximation; scale $\tilde{H}_0^{-1}$ for each iteration by a factor γ_k which is

determined by the following scheme:

$$\gamma_k = \begin{cases} \tilde{\tau}_{2k} & \text{for} \quad \tilde{\tau}_{2k} < 1 \\ \tilde{\sigma}_k & \text{otherwise} \end{cases} \tag{6.6}$$

where $\tilde{\tau}_{2k}$ is given by Eq. 5.68 with $\tilde{H}_0^{-1} = I$.

LBFGS-Opt4: Use the identity matrix as the initial inverse Hessian approximation; scale $\tilde{H}_0^{-1}$ for each iteration by a factor γ_k which is determined by the following scheme:

$$\gamma_k = \begin{cases} \tilde{\tau}_{1k} & \text{for} \quad \tilde{\tau}_{1k} < 1 \\ \tilde{\sigma}_k & \text{otherwise} \end{cases} \tag{6.7}$$

where $\tilde{\tau}_{1k}$ is given by Eq. 5.66 with $\tilde{H}_0^{-1} = I$.

LBFGS-Opt5: Use the diagonal of C_M instead of the identity matrix as the initial inverse Hessian approximation and only scale the initial matrix at the first iteration by a factor γ_0 given by the following scheme:

$$\gamma_0 = \begin{cases} \tilde{\tau}_{10} & \text{for} \quad \tilde{\tau}_{10} < 1 \\ \tilde{\sigma}_0 & \text{otherwise} \end{cases} \tag{6.8}$$

Note that for the initial scaling, i.e., $k = 0$, $\tilde{\tau}_{10} = \tilde{\tau}_{20} = \tilde{\tau}_{30}$ holds, i.e., Eqs. 5.66 through 5.68 give the same value of $\tilde{\tau}_k$.

LBFGS-Opt6: Use the diagonal of C_M as the initial inverse Hessian approximation; scale $\tilde{H}_0^{-1}$ for each iteration by a factor γ_k which is determined by the following scheme:

$$\gamma_k = \begin{cases} \tilde{\tau}_{3k} & \text{for} \quad \tilde{\tau}_{3k} < 1 \\ \tilde{\sigma}_k & \text{otherwise} \end{cases} \tag{6.9}$$

where $\tilde{\tau}_{3k}$ is given by Eq. 5.67 with $\tilde{H}_0^{-1} = \text{Diag}[C_M]$.

LBFGS-Opt7: Use the diagonal of C_M as the initial inverse Hessian approximation; scale $\tilde{H}_0^{-1}$ for each iteration by a factor γ_k which is

determined by the following scheme:

$$\gamma_k = \begin{cases} \tilde{\tau}_{2k} & \text{for} \quad \tilde{\tau}_{2k} < 1 \\ \tilde{\sigma}_k & \text{otherwise} \end{cases} \tag{6.10}$$

where $\tilde{\tau}_{2k}$ is given by Eq. 5.68 with $\tilde{H}_0^{-1} = \text{Diag}[C_M]$.

LBFGS-Opt8: Use the diagonal of C_M as the initial inverse Hessian approximation; scale $\tilde{H}_0^{-1}$ at each iteration by a factor γ_k which is determined by the following scheme:

$$\gamma_k = \begin{cases} \tilde{\tau}_{1k} & \text{for} \quad \tilde{\tau}_{1k} < 1 \\ \tilde{\sigma}_k & \text{otherwise} \end{cases} \tag{6.11}$$

where $\tilde{\tau}_{1k}$ is given by Eq. 5.66 with $\tilde{H}_0^{-1} = \text{Diag}[C_M]$.

LBFGS-Opt1 was previously considered by Liu and Nocedal (1989), to the best of our knowledge, none of the other options have been considered previously. All the results corresponding to the above options are summarized in Table 6.5. Comparing the first four options, LBFGS-Opt1 is the worst one in terms of the value of the objective function at convergence. As shown in Table 6.5 both the average value of objective function at convergence and the number of iterations required to obtain convergence are higher for LBFGS-Opt4 than for LBFGS-Opt3. Note that for realization R4, the objective function is equal to 309 at convergence for option 4 which significantly increases the average value of the objective function at convergence. If we ignore the result for R4 of LBFGS-Opt4, the average value of the objective function at convergence and the number of iterations required are 49 and 40.5 respectively, these results are better than those obtained with LBFGS-Opt3. If we compare these two options based on the convergence behavior for every conditional realization, we would say that these two options are similar. In this table, an F entry indicates that the algorithm converged to a very large value. In our examples, F corresponds to a value greater than or equal to 700. Note that LBFGS-Opt1, in which a "fixed" scaling factor was used, converged in slightly fewer

iterations than options 2, 3 and 4, but it converged to a higher average objective function value (84.5). In LBFGS-Opt5, the diagonal of C_M was used as the initial inverse Hessian approximation and we only scale the initial $\tilde{H}_0^{-1}$. For initial scaling, all the three formula which are used to calculate $\tilde{\tau}_k$ (Eqs. 5.66 through 5.68) are identical. Thus, in the LBFGS-Opt5 case, it does not matter which formula is used to calculate $\tilde{\tau}_k$. In option 6, 7, and 8, the diagonal of C_M was used as the initial inverse Hessian approximation and we scale the initial $\tilde{H}_0^{-1}$ at each iteration. The difference between them is that different formulas were used to calculate $\tilde{\tau}_k$. Comparing these results with those from LBFGS-Opt5, we can conclude that scaling $\tilde{H}_0^{-1}$ at each iteration is better than just scaling at only the initial iteration. (We obtained the same conclusion for the case where the identity matrix was used as $\tilde{H}_0^{-1}$ even though we did not show these results in Table 6.5.) Comparing the results of option 6 through 8, we also can conclude that using $(s_k^T \tilde{H}_0 s_k)/(s_k^T y_k)$ (LBFGS-Opt8) to calculate $\tilde{\tau}_k$ provides the best results.

Table 6.5: Comparison of LBFGS algorithm with different options.

		R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	Average
LBFGS-Opt1	Obj.	53	47	63	131	138	44	66	174	87	42	84.5
	No. Iter.	36	41	17	42	27	50	10	39	34	43	33.9
LBFGS-Opt2	Obj.	53	51	41	70	114	42	65	F	95	42	63.7
	No. Iter.	41	33	40	52	40	53	11	F	24	43	37.4
LBFGS-Opt3	Obj.	47	51	41	62	115	43	66	F	54	40	57.6
	No. Iter.	41	34	25	51	39	41	10	F	41	40	35.7
LBFGS-Opt4	Obj.	47	55	30	309	94	36	36	F	58	38	78
	No. Iter.	38	50	36	12	43	44	35	F	39	39	37
LBFGS-Opt5	Obj.	90	F	60	100	70	58	59	110	156	60	84.8
	No. Iter.	40	F	14	39	25	39	16	50	11	46	31.1
LBFGS-Opt6	Obj.	55	55	78	66	F	41	60	163	61	41	69
	No. Iter.	21	26	8	46	F	46	16	36	50	32	31.2
LBFGS-Opt7	Obj.	59	F	30	38	83	42	36	67	50	36	49
	No. Iter.	10	F	20	32	16	24	23	16	34	34	23.2
LBFGS-Opt8	Obj.	43	41	31	38	55	33	35	54	54	36	42
	No. Iter.	34	22	18	38	21	35	26	24	21	34	27.3

6.1.3 Preconditioning Effects on the Conjugate Gradient Method

In the results shown previously, we found that the conjugate gradient method with C_M^{-1} as the preconditioner does not work well. Here we denote this method by CM-PCG which means the preconditioner is given by the full matrix C_M^{-1}. For the 10 realizations tested, the conjugate gradient algorithm converged to higher objective function values than the BFGS algorithm with C_M as the initial inverse Hessian approximation and the LBFGS algorithm; see Table 6.3. As discussed previously, an approximation to the inverse Hessian calculated from the quasi-Newton equation can be incorporated into the conjugate gradient algorithm as a preconditioner. Here we tested two preconditioners. One is generated from BFGS with the full matrix C_M as the initial inverse Hessian approximation. For simplicity, we call this algorithm BFGS-PCG which means the preconditioner is generated from BFGS. Note that when we form the inverse Hessian approximation we apply Eq. 5.50 with s_k and y_k obtained based on conjugate gradient method. The other preconditioner is generated from LBFGS using LBFGS-Opt8, i.e., the diagonal of C_M was used as $\tilde{H}_0^{-1}$ and the scaling of option 8 was used at each iteration. We refer to this algorithm as LBFGS-PCG which means the preconditioner is generated from LBFGS. Again, in this algorithm when we calculate $\tilde{H}_k^{-1} g_k$, the s_k's and y_k's are obtained based on the conjugate gradient method.

The final objective function value and the number of iterations required to converge for both algorithms are shown in Table 6.6. For the purpose of comparison, we also include the results from BFGS with scaling option of BFGS-Opt3, LBFGS with scaling option of BFGS-Opt8 and CM-PCG in this table. The convergence behavior of LBFGS and LBFGS-PCG are similar. Compared to the BFGS algorithm, we can see that on average the BFGS-PCG converged to slightly lower objective function values in fewer iterations although BFGS-PCG failed for one realization. Both BFGS-PCG and LBFGS-PCG have much better convergence properties than CM-PCG.

From this particular example, we can see that LBFGS and LBFGS-PCG

Table 6.6: Results for BFGS-PCG and LBFGS-PCG.

		R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	Average
BFGS-PCG	Obj.	90	F	28	40	54	41	39	46	53	47	48.7
	No. Iter.	35	F	25	31	10	33	25	19	24	26	25.3
LBFGS-PCG	Obj.	44	51	29	38	57	33	36	47	54	36	42.5
	No. Iter.	23	17	23	31	23	38	21	35	20	30	26.1
CM-PCG	Obj.	153	146	70	94	347	42	184	213	230	45	152
	No. Iter.	12	20	11	23	19	64	5	17	19	23	21.3
BFGS	Obj.	77	50	33	43	56	36	43	105	64	38	54.5
	No. Iter.	13	51	25	38	17	41	29	14	32	31	29.1
LBFGS	Obj.	43	41	31	38	55	33	35	54	54	36	42
	No. Iter.	34	22	18	38	21	35	26	24	21	34	27.3

are the most effective and efficient methods among all the algorithms. So these two algorithms were used to generate 50 different realizations. For the LBFGS algorithm, the average value of the objective function at the convergence and the average number of iterations required to converge are 44.7 and 27.2 respectively. For the LBFGS-PCG algorithm, these two average values are 44.3 and 26.3, respectively.

To further confirm the effectiveness of the preconditioner generated from the LBFGS, we used LBFGS-PCG algorithm in the restricted-entry example of Kalita (2000) where CM-PCG worked very poorly. The restricted-entry example is a slight modification of the gas reservoir problem just considered. In the restricted-entry case, only the top layer is open to flow. For this case we also generated 10 realizations using the CM-PCG and the LBFGS-PCG algorithms. The value of the objective function at convergence and the number of iterations required to converge for both methods are summarized in Table 6.7. When CM-PCG was used, all the 10 realizations converged to a very high value. On average, 35.3 iterations were required to reduce the average objective function value to 447. For the same 10 realizations, LBFGS-PCG performed very well, the objective function converged to an average value of 43.1 in 29.3 iterations.

Table 6.7: Results for restricted-entry case.

		R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	Average
CM-PCG	Obj.	237	578	159	372	252	119	776	1381	504	93	447
	No. Iter.	46	63	30	12	17	52	33	39	19	42	35.3
LBFGS-PCG	Obj.	55	37	32	41	52	38	38	51	50	37	43.1
	No. Iter.	35	34	23	32	21	30	34	23	31	30	29.3

6.2 Two-Dimensional Three-Phase Synthetic Example

A 2D three-phase history matching problem was considered in this example. We use a 15×15 grid with $\Delta x = \Delta y = 40$ ft and $\Delta z = 30$ ft. We consider a very small problem so we can easily apply all optimization algorithms. The porosity for the true model is homogeneous and equal to 0.22. Permeability is isotropic and uniform in three different zones; see Fig. 6.3. The values of $\ln(k)$ in the lower left zone, lower right zone and the upper half zone are equal to 4.0, 4.6 and 4.2 with k in md, respectively. Four producers and one water injection well are completed in the reservoir. The well locations are indicated by the white squares in Fig. 6.3. All producers start producing at time zero at a constant total flow rate of 200 STB/Day and produce for 300 days. The production constraint is the minimum bottom-hole pressure which is set to 50 psi and the economic limit is the maximum WOR which is set to 49 STB/STB. When the bottom-hole pressure of a well decreases below 50 psi, then the well will be produced at a constant bottom-hole pressure equal to 50 psi. If the WOR exceeds 49, then the well will be shut in. For all examples considered in this section, the production constraint and economic limit are never reached. Bottom-hole pressure data from all five wells, GOR and WOR from all four producers are used as the conditioning data to estimate the gridblock permeabilities, i.e., the porosity is fixed at its true values. A total of 364 data (28 for each type of data at each producing well and 28 pressure data at the water injection well) are history matched. We assume pressure measurement errors to be independent, identically-distributed, normal random variables with mean equal to zero and variance equal to 1 psi^2. GOR measurement errors were modeled similarly except the variance was set equal to 25 (SCF/STB)^2. Following Wu (1999), the variance of WOR measurement errors was specified by

$$\mathrm{Var}[e_{\mathrm{WOR}}] = \mathrm{WOR}^2_{\mathrm{obs}}\epsilon_o^2 + \frac{1}{q^2_{o,\mathrm{obs}}}\sigma^2_{q_{w,\mathrm{obs}}}, \tag{6.12}$$

where

$$\epsilon_m^2 = \frac{\mathrm{Var}[q_{m,\mathrm{obs}} - q_m]}{q^2_{m,\mathrm{obs}}}, \tag{6.13}$$

for $m = o, w$ and

$$\sigma_{q_w,\text{obs}} = \max[\epsilon_w q_{w,\text{obs}}, \sigma^{\min}_{q_w,\text{obs}}]. \tag{6.14}$$

Here, $q_{m,\text{obs}}$, $m = o, w$ denotes the observed rate of phase m. In this example, we choose $\sigma^{\min}_{q_w,\text{ obs}}$=2.0 STB/Day, ϵ_o=0.001 and ϵ_w=0.02. The variances for different data are used to form the data covariance matrix C_D. The isotropic spherical variogram with the range equal to 240 ft in all three directions and the variance for $\ln(k)$ equal to 1 was used to construct the model covariance matrix, C_M. The objective function given in Eq. 2.14 was minimized; i.e., we generated the maximum a posteriori (MAP) estimate by history matching the production data.

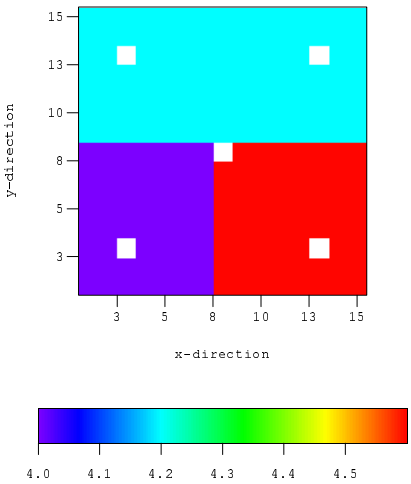


Fig. 6.3: Permeability field for the true model.

6.2.1 Comparison of the Optimization Algorithms

For the examples considered in this subsection, we did not add any noise to the true data generated by running the simulator with the true model as input. Our focus is on the investigation of the computational efficiency of different optimization algorithms.

The iterative solver (see Appendix A) was applied to solve the adjoint equations involved in the computation of the sensitivity coefficient matrix and the com-

putation of the gradient of the objective function. For the comparison purpose, the Levenberg-Marquardt (LM), Broyden-Fletcher-Goldfarb-Shanno (BFGS), limited memory BFGS (LBFGS), conjugate gradient with C_M as the preconditioner (CM-PCG) and conjugate gradient with preconditioner generated from limited memory BFGS (LBFGS-PCG) were used to minimize the objective function involved in the history matching procedure. In some cases, GN fails to converge to a legitimate model; see Li (2001). Thus, GN is not compared with the other algorithms. For these algorithms, a uniform value of 4 for $\ln(k)$ was used as the initial guess.

In Levenberg-Marquardt, the initial damping factor was chosen as 10^5. When the objective function increases, the damping factor was simply multiplied by a factor of 10; whereas, when the objective function decreases, the damping factor was simply divided by a factor of 10. Levenberg-Marquardt (LM) converged to 13.343 in 9 iterations. The curve through the circles in Fig. 6.4 shows the behavior of the objective function during the LM iterations. Fig. 6.5(b) shows the final model (i.e., the permeability field) obtained by Levenberg-Marquardt, which is very similar to the true model which is reproduced in Fig. 6.5(a).

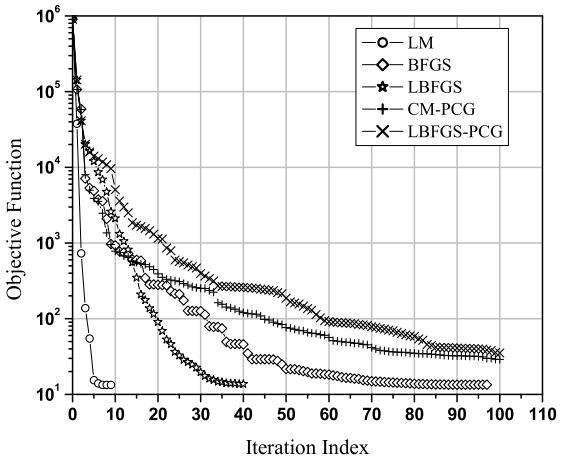


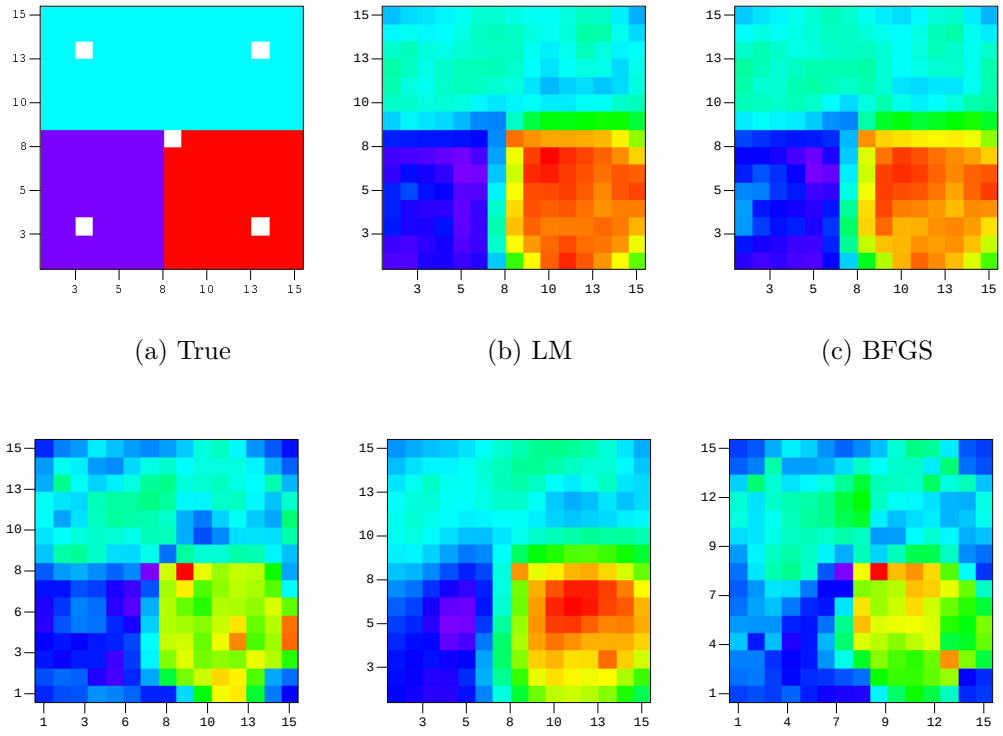
Fig. 6.4: Behavior of the objective function.

The BFGS method we used is the standard Broyden-Fletcher-Goldfarb-Shanno method with initial scaling. In other words, $\gamma_k = 1$ for all $k > 0$ was

used when applying Eq. 5.55 to update the inverse Hessian approximation. The initial scaling factor γ_0 was obtained by

$$\gamma_0 = \sigma_0 = \frac{s_0^T y_0}{y_0^T \tilde{H}_0^{-1} y_0^T}, \tag{6.15}$$

where $\tilde{H}_0^{-1}$ is equal to C_M. BFGS converged to 13.448 in 97 iterations. The curve through the diamonds in Fig. 6.4 shows the behavior of the objective function. Fig. 6.5(c) shows the final model obtained by the BFGS. We can see that the final model obtained by BFGS is very similar to the final model obtained by the Levenberg-Marquardt method (compare Figs. 6.5(c) and 6.5(b)).



(a) True (b) LM (c) BFGS

(d) LBFGS (e) CM-PCG (f) LBFGS-PCG

Fig. 6.5: Final model obtained by different optimization algorithms.

In limited memory BFGS, we scaled the inverse Hessian approximation at each iteration. Following Oren (1974a), and the results of our comparative studies,

the scaling factors were chosen by the following scheme

$$\gamma_k = \tilde{\tau}_{1k} \quad \text{if} \quad \tilde{\tau}_{1k} < 1.0, \tag{6.16}$$

$$\gamma_k = \tilde{\sigma}_k \quad \text{otherwise.} \tag{6.17}$$

The diagonal of C_M was chosen as the initial inverse Hessian approximation. The 30 most recent vectors (i.e., s_k's and y_k's) were used to construct the inverse Hessian approximation at each iteration, i.e, $L = 30$. Limited memory BFGS converged to 13.685 in 40 iterations. The curve through the stars in Fig. 6.4 shows the behavior of the objective function. Fig. 6.5(d) shows the final model obtained by the limited memory BFGS, which captures the main characteristics of the true model, but is somewhat rougher than the true model and the MAP estimate obtained with LM.

The two preconditioned conjugate gradient methods discussed earlier were implemented and applied to this history matching problem. In one method, the full C_M^{-1} was used as the preconditioner (we call this method CM-PCG); whereas in the other method, we used an estimated quasi-Newton preconditioner (we call this method LBFGS-PCG). Both algorithms were terminated at 100 iterations, because this was the maximum number of iterations allowed. However, the convergence criteria of Eqs. 5.69 and 5.70 were not satisfied at the 100th iteration of either method. CM-PCG "converged" to a model corresponding to an objective function value equal to 28.851 in 100 iterations. The curve through the plus signs in Fig. 6.4 shows the behavior of the objective function. Fig. 6.5(e) shows the final model obtained by the CM-PCG. LBFGS-PCG "converged" to a model corresponding to an objective function value equal to 35.187 in 100 iterations. The curve through the crosses in Fig. 6.4 shows the behavior of the objective function. Fig. 6.5(f) shows the final model obtained by the LBFGS-PCG. Note the MAP estimates obtained with the preconditioned CG method are inferior to those obtained by LM and BFGS.

In the quasi-Newton methods, the Hessian inverse approximation is constructed based on the quasi-Newton search direction. The theory guarantees that for a positive definite quadratic function the Hessian inverse approximation becomes the true Hessian inverse at the nth iteration where n is the number of model param-

eters for the quadratic problem with exact line search. However, in LBFGS-PCG the Hessian inverse approximation is constructed based on the preconditioned conjugate gradient search direction, which is no longer the quasi-Newton Hessian inverse approximation. When we use the conjugate gradient search direction to find a new model, and based on this new model, construct the Hessian inverse approximation as a preconditioner, it is not clear how good this preconditioner is. For the previous single-phase gas example, this preconditioner results in faster convergence than is obtained by simply using C_M^{-1} as a preconditioner; in the current example, however, convergence is slower.

The final objective function value at convergence and the number of iterations required to converge for different algorithms are summarized in Table 6.8. In the "Scaling Scheme" column, "Initial Scaling" means that only the initial Hessian inverse approximation is scaled by a factor γ_0. In other words, we choose a sequence γ_k such that $\gamma_k = 1$ for all $k > 0$. "All Scaling" means that the inverse Hessian approximation $\tilde{H}_k^{-1}$ was scaled by γ_k at each iteration for the case when BFGS was applied and the initial inverse Hessian approximation $\tilde{H}_0^{-1}$ was scaled by γ_k at each iteration for the case where limited memory BFGS was applied. "N/A" means not available. FCM-LBFGS stands for the LBFGS algorithm in which the initial Hessian inverse approximation is chosen as the full C_M matrix instead of just the diagonal of C_M; see the discussion presented later.

Algorithms	Scaling Scheme	Objective Function	No. of Iterations
LM	N/A	13.343	9
BFGS	Initial Scaling	13.448	99
LBFGS	All Scaling	13.685	40
CM-PCG	N/A	28.851	100
LBFGS-PCG	All Scaling	35.187	100
FCM-LBFGS	All Scaling	13.992	33

Table 6.8: Comparison of different minimization algorithms.

Based on Eq. 5.101 and the number of iterations required (see Table 6.8), LBFGS and FCM-LBFGS, respectively, are 13.7 and 16.6 times faster than Levenberg-Marquardt overall. Table 6.9 shows the CPU time in seconds used by different algorithms. Based on the real CPU time, LBFGS and FCM-LBFGS, respectively, are 10.5 and 11.1 times faster than Levenberg-Marquardt overall. The column labeled "Scaling Scheme" has the same meaning as in Table 6.8.

Algorithms	Scaling Scheme	CPU time (seconds)
LM	N/A	2930
BFGS	Initial Scaling	923
LBFGS	All Scaling	279
CM-PCG	N/A	887
LBFGS-PCG	All Scaling	904
FCM-LBFGS	All Scaling	263

Table 6.9: Comparison of the CPU time used by different minimization algorithms.

6.2.2 Effect of Preconditioning Matrix on Conjugate Gradient Methods

Fig. 6.6 shows the behavior of the objective function obtained by the conjugate gradient without preconditioning (triangles) and the preconditioned conjugate gradient with C_M^{-1} as the preconditioner (circles). We can see clearly that the preconditioned conjugate gradient is slightly better than conjugate gradient without preconditioning. Fig. 6.7 shows the final model (a) obtained by the conjugate gradient method without preconditioning compared to the true model (c) and the model (b) obtained using C_M^{-1} as a preconditioner. We can see that the MAP estimate obtained from CG without preconditioning is far rougher than the model obtained by the CM-PCG. We can conclude that the prior covariance matrix C_M not only acts as a preconditioning matrix but also provides smoothness for the model.

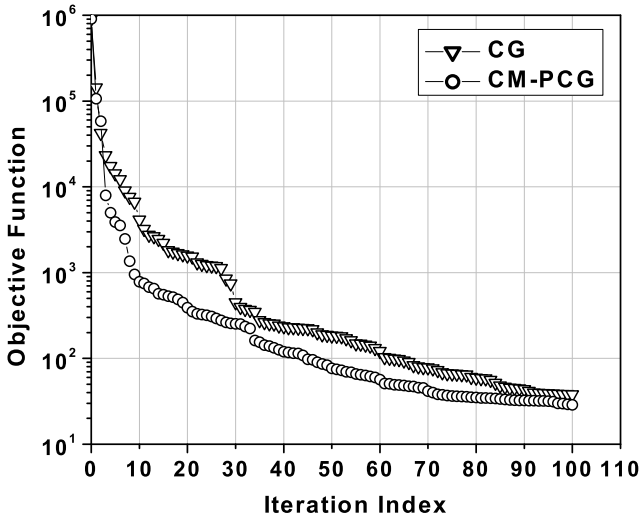
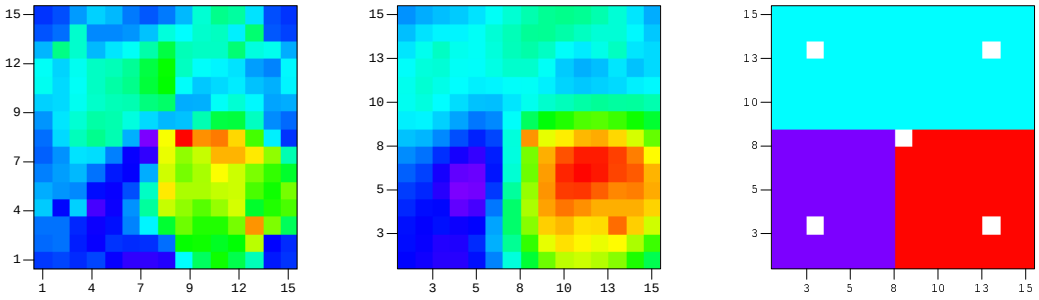


Fig. 6.6: Behavior of the objective function obtained by CG and CM-PCG.



(a) CG (b) CM-PCG (c) True

Fig. 6.7: Final model obtained by conjugate gradient without preconditioning.

6.2.3 BFGS Scaling Scheme

The BFGS result of the preceding section was obtained by only scaling the initial Hessian inverse approximation $\tilde{H}_0^{-1}$. If we consider the fact that the Hessian matrix changes iteration by iteration for the nonlinear problem, it seems that we should scale the matrix at every iteration based on the new information; see Oren and Luenberger (1974) or Oren (1974b). According to Oren and Spedicato (1976), the optimal scaling factor which minimizes the upper bound of the condition number

of $\tilde{H}_{k+1}^{-1}$ at the kth iteration should be

$$\gamma_k = \sigma_k \equiv \frac{s_k^T y_k}{y_k^T \tilde{H}_k^{-1} y_k}. \tag{6.18}$$

The new Hessian inverse approximation $\tilde{H}_{k+1}^{-1}$ was calculated by using Eq. 5.55 with γ_k given by Eq. 6.18. With this all-scaling scheme, BFGS converged to 16.1989 in 66 iterations; see the curve through circles in Fig. 6.8. The curve through the plus signs in Fig. 6.8 shows the behavior of objective function iteration by iteration for BFGS with initial scaling scheme. This case was presented previously. Although it appears that the all-scaling scheme is better than just initial scaling in terms of the number of iterations required to converge, at the 66th iteration, the objective function value for the two schemes are very close. Yet, the convergence criteria of Eqs. 5.69 and 5.70 were satisfied for all-scaling BFGS but not for the initial scaling BFGS at the 66th iteration. The diamonds shown in Fig. 6.8 represent the objective function values obtained by BFGS without any scaling at each iteration. Clearly, BFGS without scaling is much worse than initial scaling and all-scaling. As presented previously, scaling does help to improve the convergence rate of BFGS for this three-phase problem. Note this result is somewhat different than was obtained for the gas example of section 6.1 where scaling did not have a great effect. The MAP estimate obtained with the all scaling BFGS is shown in Fig. 6.9 and is very similar to the MAP estimate obtained by applying BFGS with initial scaling (see Fig. 6.5(c)).

Based on Oren and Spedicato (1976), σ_k is the optimal scaling factor for γ_k if the BFGS algorithm ($\theta_k = 1$) or one of its variants is applied; see Eq. 5.63. Our experiments discussed later, however, indicate that $\gamma_k = \tau_k$ (Eq. 5.56) is superior to $\gamma_k = \sigma_k$ (Eq. 5.57) for LBFGS; also see Zhang et al. (2001). Here, we investigate the convergence behavior of BFGS with the scaling factor equal to τ_k. The behavior of the objective function obtained by BFGS with $\gamma_k = \tau_k$ is shown by the curve through stars in Fig. 6.10. The curve through circles in the same figure is obtained by the BFGS with $\gamma_k = \sigma_k$. Even though $\gamma_k = \sigma_k$ is the optimal scaling factor based on Oren's theory, it turns out $\gamma_k = \tau_k$ is not worse than $\gamma_k = \sigma_k$. The curve through plus signs in this same figure is obtained by the BFGS with the modified

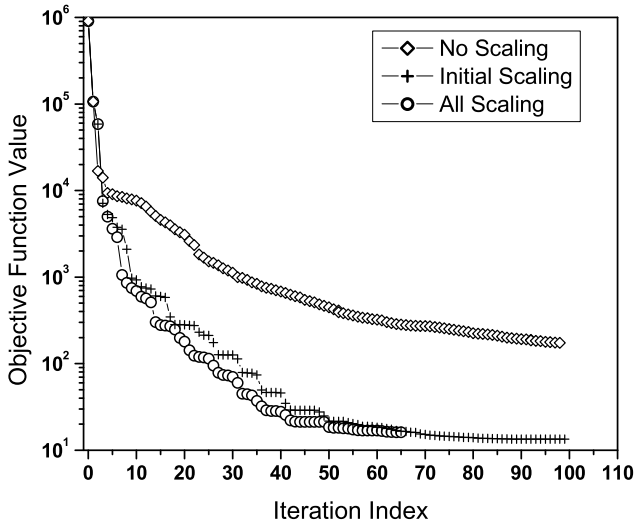


Fig. 6.8: Behavior of the objective function for BFGS with different scaling schemes.

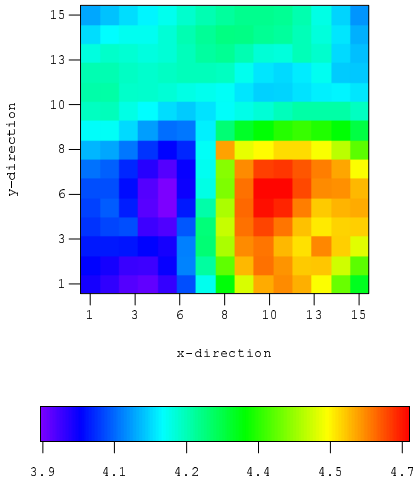


Fig. 6.9: Final model obtained by BFGS with scaling at all iterations; $\gamma_k = \sigma_k$.

Oren scaling scheme, i.e., $\gamma_k = \tau_k$ if $\tau_k < 1.0$; otherwise $\gamma_k = \sigma_k$. There is not much difference between these three scaling schemes in terms of the objective function at convergence.

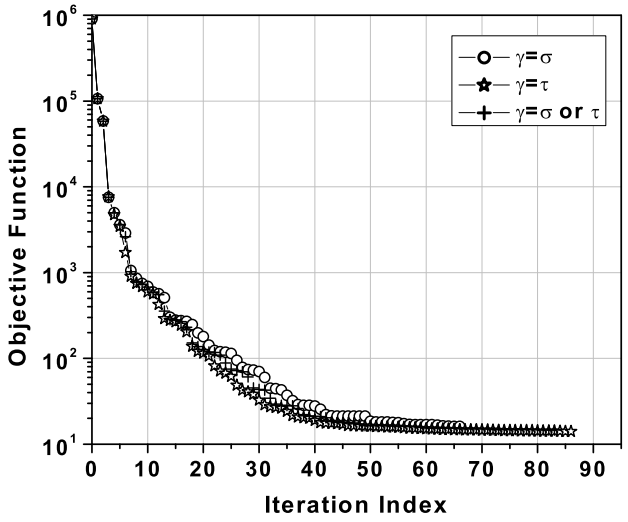


Fig. 6.10: Behavior of the objective function for BFGS with different schemes for scaling at all iterations.

6.2.4 LBFGS Scaling Scheme

As shown in Oren (1974b), $\tilde{\tau}_k$ calculated by Eq. 5.66 or Eq. 5.67 is an approximation to the correct τ_k which is given by Eq. 5.56. In our implementation of LBFGS, we only form the product of $\tilde{H}_k^{-1} g_k$ which is used to generate the search direction. We never form $\tilde{H}_k^{-1}$ or $\tilde{H}_k$ explicitly. Therefore, we cannot obtain τ_k using the exact form of Eqs. 5.59 and 5.61 and cannot obtain σ_k using the exact form of Eq. 5.57. Thus, as we described in chapter V, we have to use Eq. 5.66 or Eq. 5.67 to approximate τ_k and use Eq. 5.65 to approximate σ_k. The scaling factor was chosen as follows:

$$\gamma_k = \tilde{\tau}_k \qquad \text{if} \quad \tilde{\tau}_k < 1, \tag{6.19}$$

$$\gamma_k = \tilde{\sigma}_k \qquad \text{otherwise}, \tag{6.20}$$

where $\tilde{\tau}_k$ can be obtained by either Eq. 5.66, Eq. 5.67 or Eq. 5.68 and $\tilde{\sigma}_k$ is obtained by Eq. 5.65.

Fig. 6.11 shows the behavior of the objective functions during the LBFGS iterations when different formulas are applied to calculate $\tilde{\tau}_k$. The diamonds repre-

sent the objective function obtained by using Eq. 5.67 to calculate $\tilde{\tau}_k$. The circles represent the objective function obtained by using Eq. 5.68 to calculate $\tilde{\tau}_k$. The curve through stars represents the objective function obtained by using Eq. 5.66 to calculate $\tilde{\tau}_k$. The triangles represent the objective function obtained by setting $\gamma_k = \tilde{\sigma}_k$ at every iteration where $\tilde{\sigma}_k$ is calculated from Eq. 5.65. This scaling option was used by Liu and Nocedal (1989). Shanno and Phua (1978) suggested using $\tilde{\sigma}_k$ and only scaling the initial matrix. From these curves, we can see that the curve through the stars in which the $\tilde{\tau}_k$ is calculated by the Eq. 5.66 is the best. The three other scaling factors give roughly the same convergence results as each other. Note that the curve through the circles in which $\tilde{\tau}_k = \tau_k$ is calculated by Eq. 5.68 requires the most iterations to converge. Note that Eq. 5.68 gives the correct value for τ_k. This is somewhat surprising in that it indicates that using an approximation for τ_k, which is defined by Eq. 5.56, gives better convergence results than using the correct value of τ_k. It turns out that when we use Eq. 5.66 and Eq. 5.67 to calculate $\tilde{\tau}_k$, the value of $\tilde{\tau}_k$ is always less than 1 which implies that the scaling factor always takes the value of $\tilde{\tau}_k$. When we use Eq. 5.68 to calculate $\tilde{\tau}_k$, which is the correct τ_k, it is less than 1 at some iterations and is bigger than 1 at other iterations. So whenever $\tilde{\tau}_k > 1$, γ_k takes the value of $\tilde{\sigma}_k$ which is also an approximation. So these four options are all approximations. Based on the example just presented and the gas reservoir example presented earlier, Eq. 5.66 is the best approximation. Based on the paper published by Oren and Spedicato (1976), the optimal scaling factor γ_k should always take the value of σ_k for the BFGS method. It is not clear how to form σ_k accurately at every iteration without forming $\tilde{H}_k^{-1}$ explicitly. Moreover, our result presented previously (see Fig. 6.10) indicates that σ_k might not be the optimal scaling factor. For almost all examples that we have considered using either BFGS or LBFGS, $\gamma_k = \tilde{\tau}_k$ gives as good or better convergence results than are obtained using $\gamma_k = \tilde{\sigma}_k$.

Fig. 6.12 shows the behavior of the objective function when LBFGS was applied without scaling, with initial scaling and with all-scaling based on Eqs. 6.19 and 6.20 where $\tilde{\tau}_k$ is calculated by Eq. 5.66. In this figure, the circles represent the objective function values obtained by LBFGS with all-scaling, the plus signs

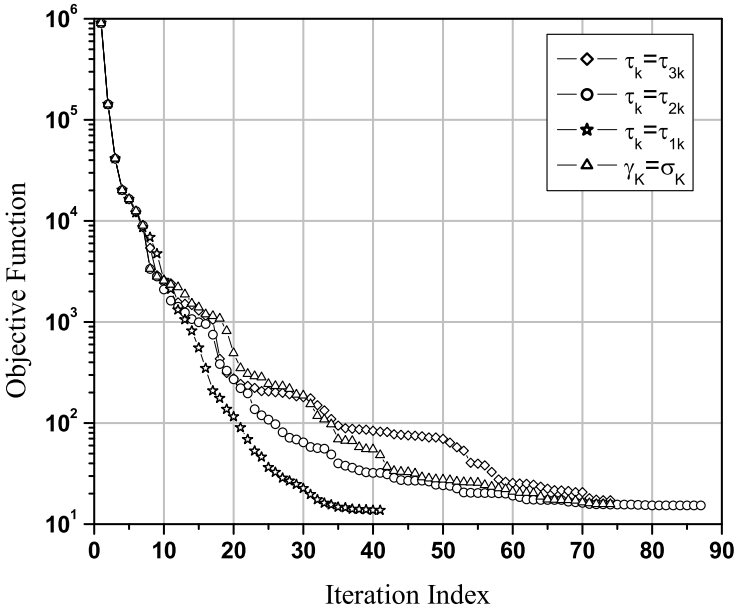


Fig. 6.11: Behavior of the objective function for LBFGS with different scaling factors.

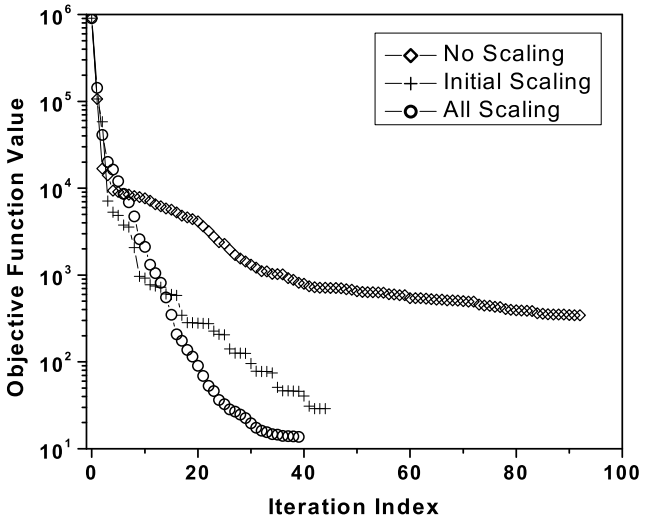


Fig. 6.12: Comparison of the behavior of the objective function for different scaling schemes.

represent the objective function value obtained by LBFGS with initial scaling and diamonds represent the objective function obtained by LBFGS without any scaling.

As in BFGS, we can see clearly that LBFGS with all-scaling is better than just initial scaling and initial scaling is better than no scaling.

6.2.5 Sensitivity to the Number of Previous Vectors

Here we investigate how the convergence rate is affected by the number of previous vectors used in constructing the Hessian update in the LBFGS algorithm. Fig. 6.13 shows the behavior of the objective function obtained by using a different number of previous vectors to construct the new approximate inverse Hessian. In all cases, the all-scaling scheme with scaling factor given by Eq. 5.66 was applied. Table 6.10 lists the number of iterations required for convergence and the value of the objective function at convergence when using a different number of previous vectors to construct the Hessian inverse updates. L denotes the number of previous vectors. We can see that when we use too few (10 in this example) previous vectors to construct Hessian inverse updates, more iterations are required for convergence and the value of the objective function is higher than when $L = 20, 30$ or 50. When L equals 20, 30 or 50, there is not too much difference in terms of the number of iterations required to obtain convergence, but $L = 20$ gives a higher value of the objective function at convergence.

L	Objective Function	No. of Iterations
10	20.922	90
20	18.555	43
30	13.685	40
50	12.512	46

Table 6.10: Sensitivity to the number of previous vectors.

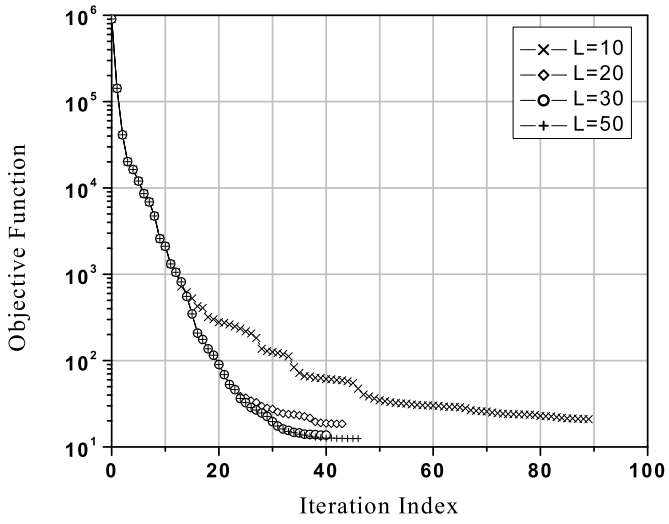
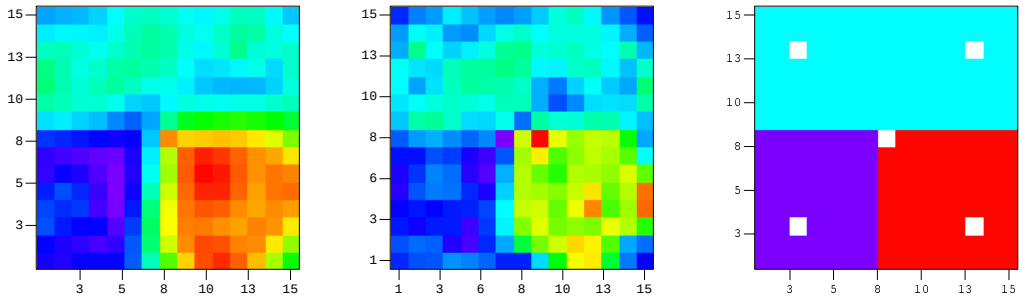


Fig. 6.13: The effect of the number of previous vectors used to construct the new Hessian inverse approximation on the performance of LBFGS.

6.2.6 Improvement of the Smoothness of the Model

From the results of Figs. 6.5(c) and 6.5(d), we see that the final model obtained by LBFGS is somewhat rougher than the true model and the MAP estimate obtained with BFGS. It turns out this is caused by the fact that only the diagonal of C_M was used as the initial Hessian inverse approximation. When we form the search direction, i.e., the product of the $\tilde{H}_k^{-1}$ and the gradient of the objective function g_k, we need to perform the operation of multiplying a vector by the initial Hessian inverse approximation. If we use the full C_M instead of just the diagonal elements of C_M as $\tilde{H}_0^{-1}$, we obtain a smoother result. Fig. 6.14(a) shows the MAP estimate obtained by LBFGS using the full C_M as the initial Hessian inverse approximation compared with using only the diagonal of C_M, DCM-LBFGS. Using $\tilde{H}_0^{-1} = C_M$ gives improved results, i.e., gives a MAP estimate much closer to the truth than is obtained with $\tilde{H}_0^{-1}$ =diag$[C_M]$. With $\tilde{H}_0^{-1} = C_M$, the LBFGS converged in 33 iterations and the value of the objective function at convergence was equal to 13.992. For this case, Fig. 6.15 indicates again that convergence is fastest when Eq. 5.66 is

used to calculate the scaling factor. This method is labeled as FCM-LBFGS in Table 6.8.



(a) FCM-LBFGS (b) DCM-LBFGS (c) True

Fig. 6.14: Final model obtained by LBFGS with full C_M (a) and diagonal C_M (b) as the initial Hessian inverse approximation, compared with the true model (c).

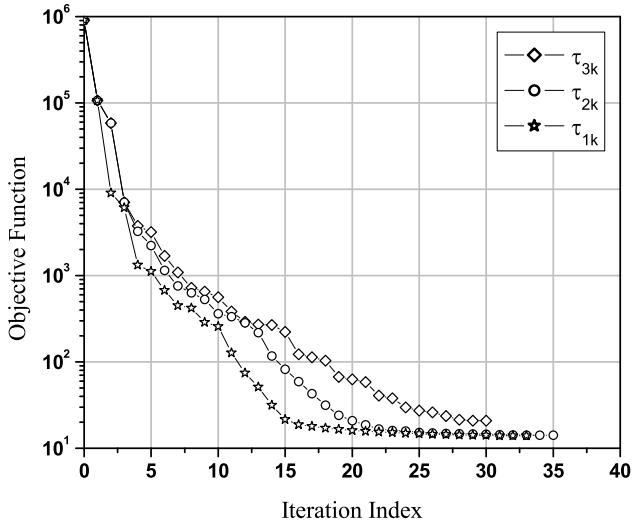


Fig. 6.15: Behavior of the objective function; full C_M as the initial Hessian inverse approximation; different scaling factor options.

6.2.7 Effect of Data Noise

In this subsection we consider the case where the observed data were obtained by adding noise to the true data. With this data set, we repeated the example in the previous subsection using LBFGS with the full C_M as the initial Hessian inverse approximation. The objective function evaluated based on Eq. 2.14 is reduced from 900728 to 309 in 35 iterations. For the case where data are noisy, the approximate results of Tarantola suggest that the expected value of the objective function is $N_d/2 = 182$ with standard deviation equal to $\sqrt{N_d/2} = 13.5$. Thus, the expectation plus five standard deviations equals 249 which is somewhat smaller than the 309. Although the value of the objective function does not satisfy the criterion given in section 2.3, one should recall that the criterion assumed data is a linear function of the model, which is not the case. Recall that when using the true data without noise the objective function converged to 13.992 in 33 iterations. The behavior of the objective function for both cases (solid circles for the true data case and the circles for the case where the data with noise) are shown in Fig. 6.16. The model obtained by history matching data with noise (observed data) is shown in Fig. 6.17(a). We can see that even though the model obtained by history matching data with noise captures the main structure of the true model, it is worse than the model obtained by history matching the true data.

6.3 Three-Dimensional Three-Phase Example

Here, we consider a three-dimensional three-phase history matching problem. We use a $40 \times 40 \times 6$ grid with $\Delta x = \Delta y = 100$ ft and $\Delta z = 30$ ft. The porosity for the true model is homogeneous and equal to 0.22. The true permeability field is an unconditional realization generated by Gaussian co-simulation. An isotropic spherical variogram with the range in all directions equal to 600 ft was used to generate unconditional realizations. The variance for $\ln(k)$ is 1 and the mean for $\ln(k)$ is 4.5. One layer of the true permeability field is shown in Fig. 6.18. Fig. 6.18 (b) is the interpolation plot of (a). The interpolation plots make it easier to compare the

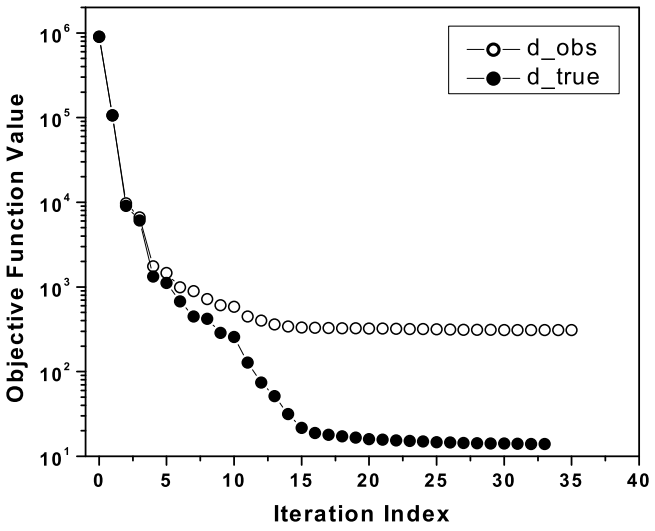
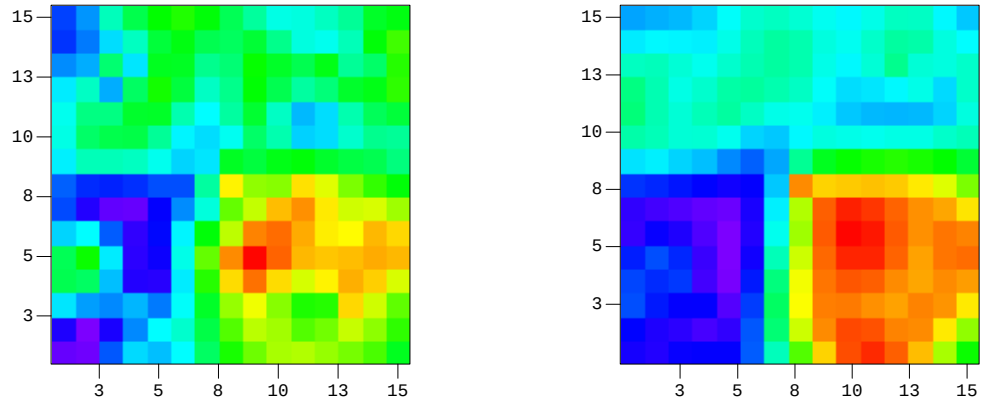


Fig. 6.16: Behavior of the objective function; full C_M as the initial Hessian inverse approximation; data with noise and without noise.



(a) data with noise

(b) true data

Fig. 6.17: Final model obtained by LBFGS with full C_M as the initial Hessian inverse approximation by history matching data with noise (a) and true data (b).

permeability structure. The initial pressure is 4500 psi and the bubble point pressure is 4417 psi. The formation volume factor (FVF) and the viscosity for oil, water and gas at the bubble point pressure are given in Table 6.11. The capillary pressure is assumed to be negligible. The water-oil and oil-gas two-phase relative permeability

are given in Fig. 6.19 (a) and (b), respectively. Stone's model II is used to calculate three-phase oil relative permeability; see Aziz and Settari (1979). Six producers and four water injection wells are completed in the reservoir. The producers and the injectors, respectively, are indicated by black squares and white squares, respectively, in Fig. 6.18 (a). All producers start producing at time 0. We will history match synthetic data generated from running the CLASS simulator up to 500 days using the truth case as input. The injectors start to inject water at 30 days and stop at 500 days. The well operating conditions are summarized in Table 6.12. The wells are operating with the target first. Whenever the constraint is violated at a particular well, then the constraint will be switched to be the target for the corresponding well. When the economic limits are violated at a particular well, then the corresponding well will shut in. In this table, the keyword MAXVOL means maximum total rate in STB/Day; MAXWATINJ means the maximum water injection rate in STB/Day; MINBHP means the minimum bottom-hole pressure in psi and MAXWOR means the maximum water-oil ratio in STB/STB.

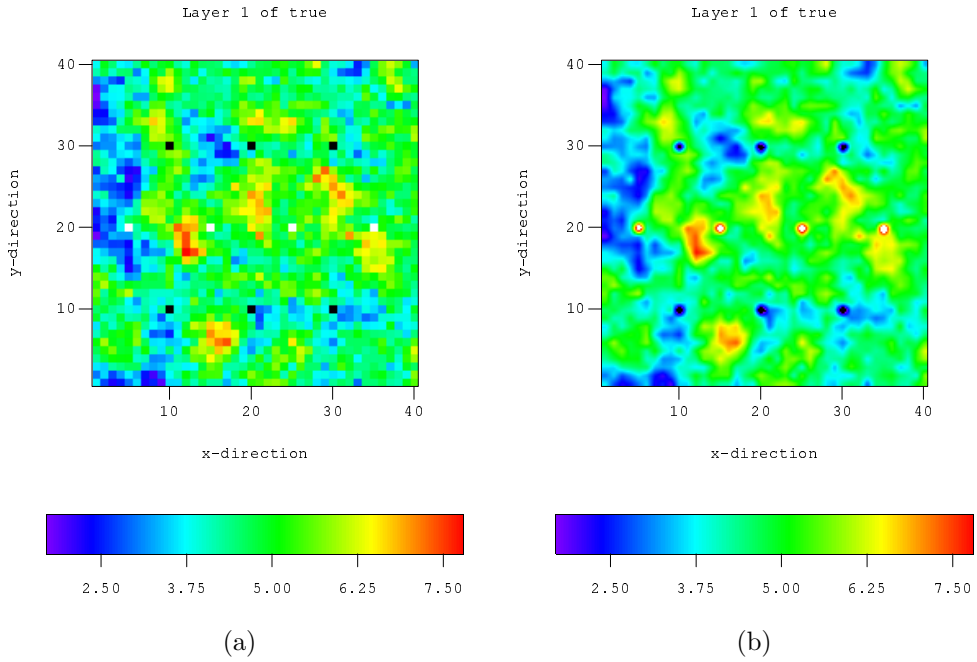
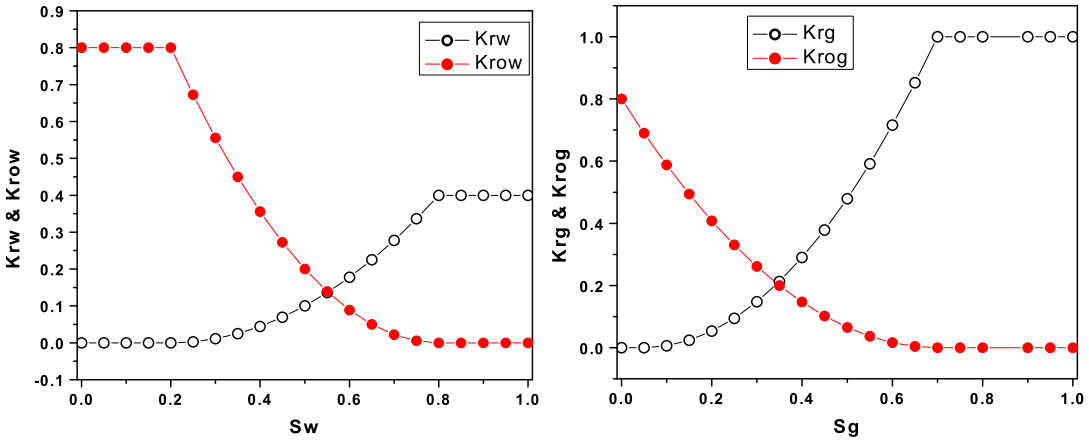


Fig. 6.18: The layer 1 log-permeability field of the true model.

B_o (RB/STB)	1.748
μ_o (cp)	0.486
B_g (RB/MSCF)	0.75
μ_g (cp)	0.0284
B_w (RB/STB)	1.006
μ_w (cp)	0.012

Table 6.11: Fluid properties at bubble point pressure.



(a) Water-Oil

(b) Oil-Gas

Fig. 6.19: The relative permeability curve used in this example.

As in the example presented in the previous section, the production target is a constant total flow rate for the producers and constant water injection rate for injection wells. The production constraints for the producers are the minimum bottom-hole pressure which is set to 50 psi and the maximum WOR which is set to 49 STB/STB. If the bottom-hole pressure of a well decreases below 50 psi, then thereafter, the well will be produced at a constant bottom-hole pressure equal to 50 psi. When the WOR is bigger than 49 STB/STB, then the corresponding well will be shut in. For this example, the production constraint and economic limit are never reached, so all wells are produced at their target rate.

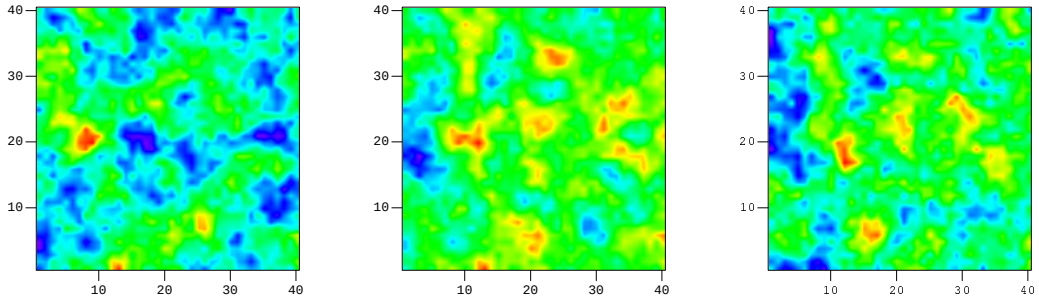
Well No.	Targets		Constraints	Economic Limits
	MAXVOL (STB/Day)	MAXWATINJ (STB/Day)	MINBHP (psi)	MAXWOR (STB/STB)
1	4000	-	50	49
2	10000	-	50	49
3	4000	-	50	49
4	15000	-	50	49
5	8000	-	50	49
6	8000	-	50	49
7	-	6000	-	-
8	-	10000	-	-
9	-	8000	-	-
10	-	8000	-	-

Table 6.12: Well operating targets, constraints and economic limits.

6.3.1 Conditioning to True Data

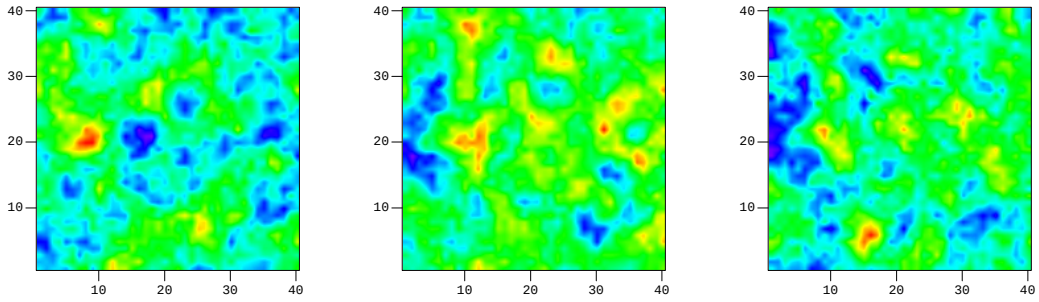
In this subsection, we consider the case where the true data are history matched. Bottom-hole pressure from all ten wells, GOR and WOR from all six producers are used as the conditioning data to estimate the gridblock permeabilities only, i.e., the porosity is fixed at its true values. There are a total of 880 conditioning production data which are history matched. (40 for each type at each producing well and 40 pressure data at the water injection wells.) The variance used for all the measurement errors are the same as used in the 2D three-phase flow example. The variances for different data are used to form the diagonal data covariance matrix C_D. The objective function given in Eq. 2.18 was minimized, except we used d_{obs} instead of d_{uc} and d_{obs} is equal to the true data with no noise added. Although we did not add noise to the data, we refer to the resulting realization as a conditional realization.

The unconditional realization was generated by Gaussian co-simulation. Two layers of the unconditional realization of the log-permeability fields are shown in Fig. 6.20 (a) and Fig. 6.21 (a), respectively. Optimization was done with the LBFGS algorithm using scaling at all iterations. Eq. 5.66 was used to generate the kth scaling factor. The objective function was reduced from 312,164,623 to 649 in 70 iterations. The objective function value was calculated based on the Eq. 2.18 with d_{uc} replaced by the true data without noise. The behavior of the objective function is shown in Fig. 6.22.



(a) Unconditional (b) Conditional (c) True

Fig. 6.20: The log-permeability field for layer 1 generated by Gaussian co-simulation (a), by history matching production data (b) and the true model(c).



(a) Unconditional (b) Conditional (c) True

Fig. 6.21: The log-permeability field for layer 4 generated by Gaussian co-simulation (a), by history matching production data (b) and the true model(c).

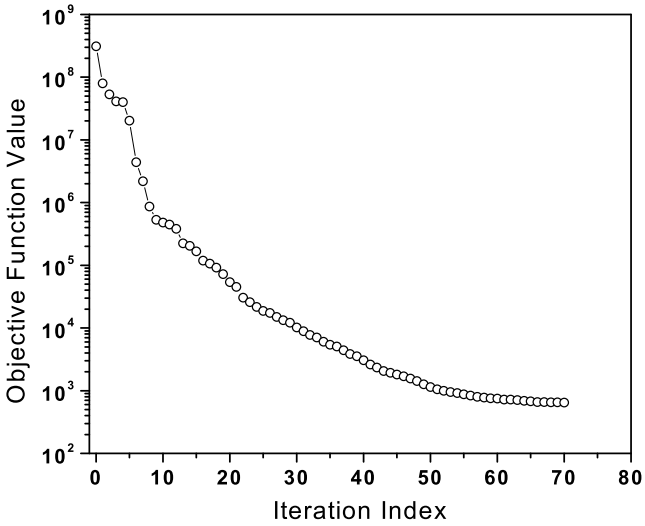
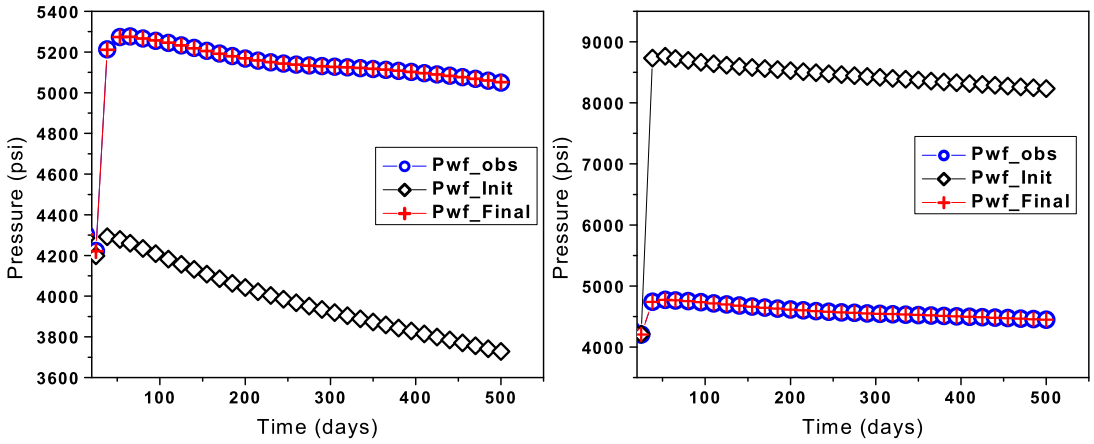


Fig. 6.22: Behavior of the objective function for the big model.

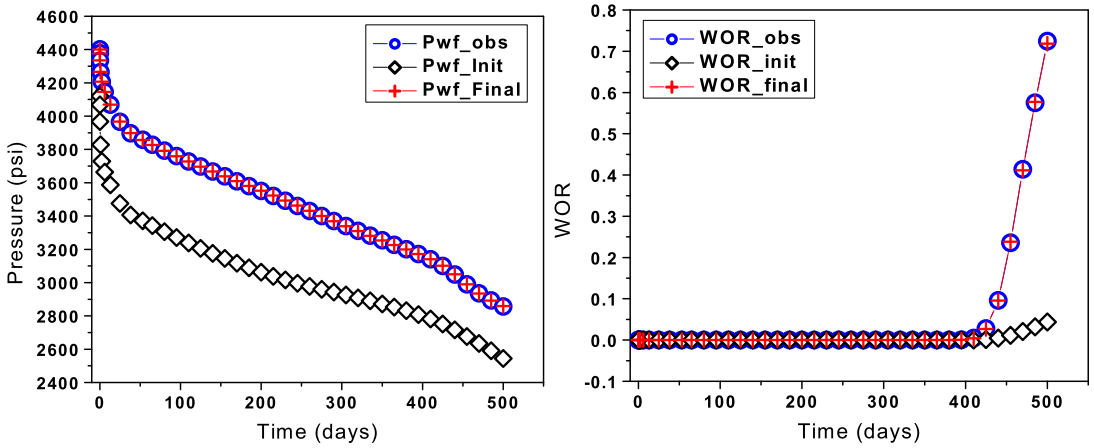
Fig. 6.23 shows the pressure match for two injectors (well 7 and well 8). In this figure and in similar figures, the line through circles represents the observed data; the line through the plus signs represents the calculated data based on the conditional realization obtained by history matching the production data and the line through the diamonds represents the calculated data based on the initial model, m_{uc}, before history matching. In Fig. 6.23 (a), the pressure data generated from the initial model falls below the observed data during the injection period, whereas in Fig. 6.23 (b), the pressure data generated using the initial model is greater than the observed pressure data. For both wells, the pressure data are matched very well. Fig. 6.24 (a) and (b) show the pressure data match for well 4 and the WOR match at the same well which is the only well at which water has broken through. We can see that at this well, both the pressure and the WOR are matched very well. Fig. 6.25 shows the gas-oil ratio data match for two producers (well 3 and well 4). We can see that we obtained a very good GOR match for both wells. We obtained matches of comparable quality to these shown in Figs. 6.23 through 6.25 at all wells.

Two layers of the final permeability fields are shown in Fig. 6.20 (b) and



(a) well 7, injector (b) well 8, injector

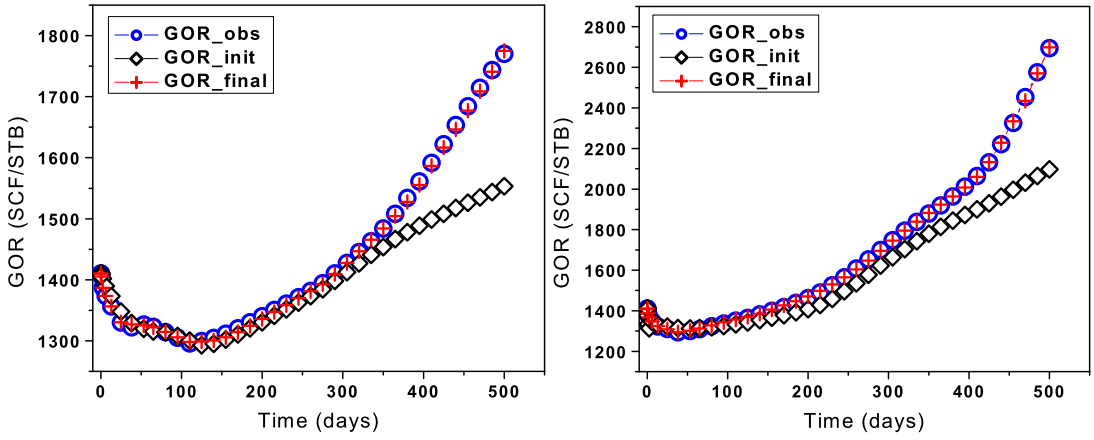
Fig. 6.23: Pressure match at two water injection wells.



(a) Pressure (b) WOR

Fig. 6.24: Pressure and WOR match at well 4.

Fig. 6.21 (b) and the corresponding two layers from the true model are shown in Fig. 6.20 (c) and Fig. 6.21 (c), respectively. The corresponding unconditional permeability distribution is shown in (a) in these two figures. The conditional realization is close to the truth, in fact much closer than would normally be expected. This occurs because of the long correlation length in the vertical direction.



(a) well 3 (b) well 4

Fig. 6.25: GOR match from two wells.

6.3.2 Conditioning to Observed Data with Noise Added

In this section, the objective function given by Eq. 2.18 is minimized. So the unconditional data were generated using Eq. 2.17. The objective function value is reduced from 313,023,514 to 5471 in 45 iterations. The squares in Fig. 6.26 show the behavior of the objective function when Eq. 2.18 is minimized. The circles in this figure show the behavior of the objective function when the true data without noise were used in Eq. 2.18. We can see that the objective function value at convergence is much bigger when unconditional data were used than when true data were used. Fig. 6.27 (a) shows the first layer of the model obtained by minimizing the objective function given by Eq. 2.18. Figs. 6.28 and 6.29 show results comparable to those shown in Figs. 6.24 and 6.25. Note although the match of data is not as close as in the case without noise in the data, we still obtain a reasonable match.

6.4 Doubly Stochastic Model Example

A 2D three-phase history matching problem was considered in this section. We use a 15×15 grid with $\Delta x = \Delta y = 40$ ft and $\Delta z = 30$ ft. The porosity for the

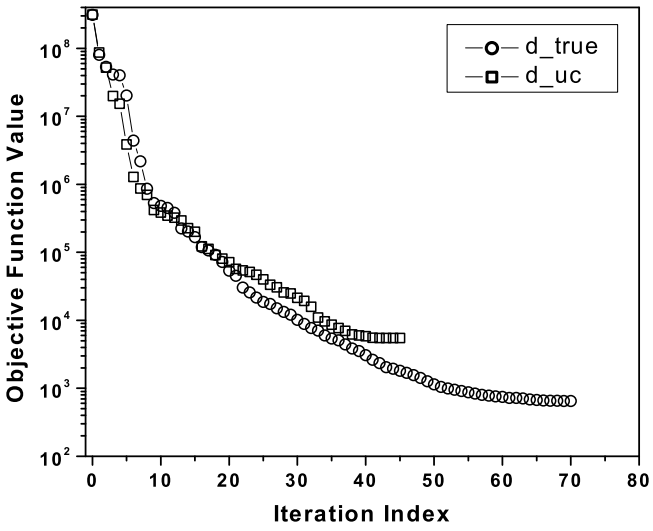
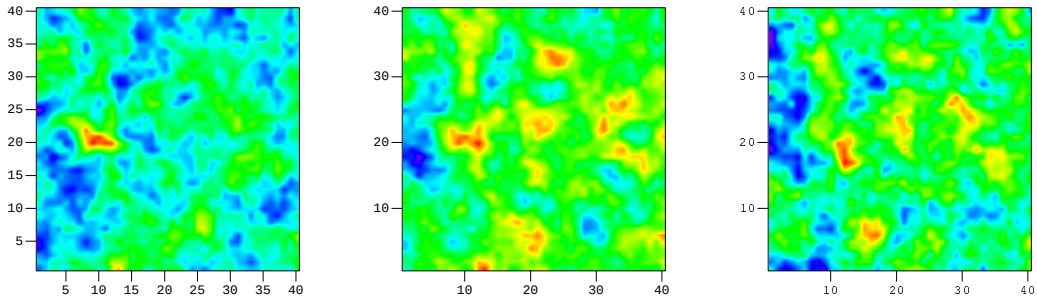


Fig. 6.26: Behavior of the objective function when unconditional data were used.

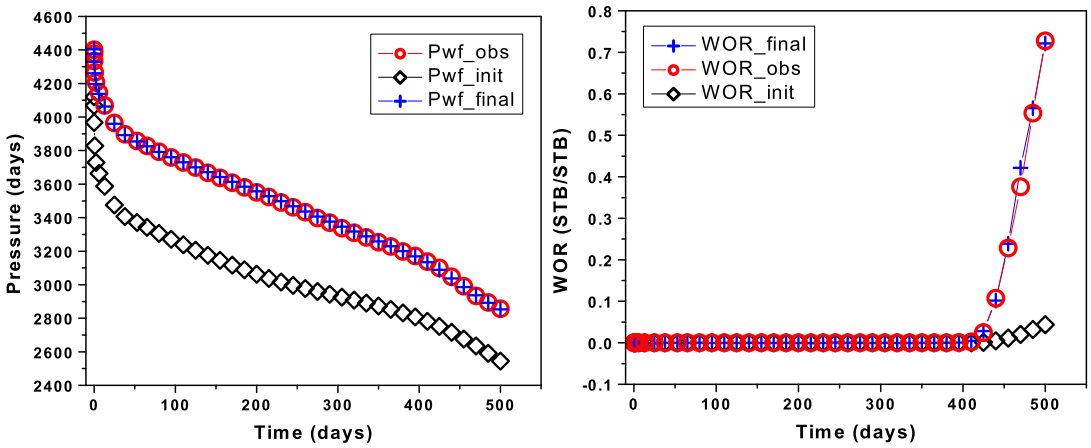


(a) Unconditional Data (b) True Data (c) True Model

Fig. 6.27: The log-permeability field for layers 1 generated by history matching d_{uc} (a), by history matching true data (b) and the true model(c).

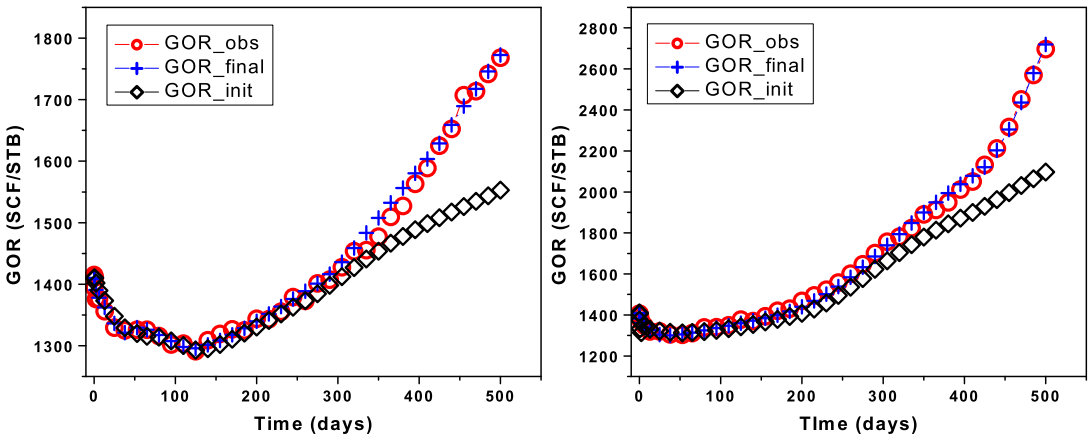
true model is homogeneous and equal to 0.22. Permeability for the true model is an unconditional realization which is generated using Gaussian co-simulation. The variogram used is an isotropic spherical variogram with the mean and variance for $\ln(k)$ equal to 4.0 and 1.0 respectively. The range in x-, $y-$ and z-directions are 240 ft, 120 ft and 30 ft respectively. The true permeability field is shown in Fig. 6.30.

Four producers and one water injection well are completed in the reservoir. The four producers are located in areal gridblock $(3,3)$, (13,3), (13,13) and (3,13)



(a) Pressure (b) WOR

Fig. 6.28: Pressure and WOR match at well 4, d_{uc}.



(a) well 3 (b) well 4

Fig. 6.29: GOR match from two wells, d_{uc}.

respectively and the water injection well is located in areal gridblock (8,8). All producers start producing at time zero at a constant total flow rate of 200 STB/Day and produce for 300 days. The production constraint is the minimum bottom-hole pressure which is set to 50 psi and the economic limit is maximum WOR which is set to 49 STB/STB. When the bottom-hole pressure of a well decreases below 50 psi,

then the well will be produced at a constant bottom-hole pressure equal to 50 psi. If the WOR exceeds 49, then the well will be shut in. For this example, the production constraint and economic limit are never reached. The injectors start injecting water at time zero at a constant flow rate of 700 STB/Day. The initial pressure is 4500 psi and the bubble point pressure is 4417 psi. Bottom-hole pressure data from all five wells, GOR and WOR from all four producers are used as the conditioning data. A total of 364 data (28 for each type of data at each producing well and 28 pressure data at the water injection well) are history matched. Noise was not added to the true data. The prior mean for the log-permeability field is adjusted also during the history matching procedure. The LBFGS was applied to minimizing the objective function given by Eq. 2.29 with d_{uc} replaced by d_{true}.

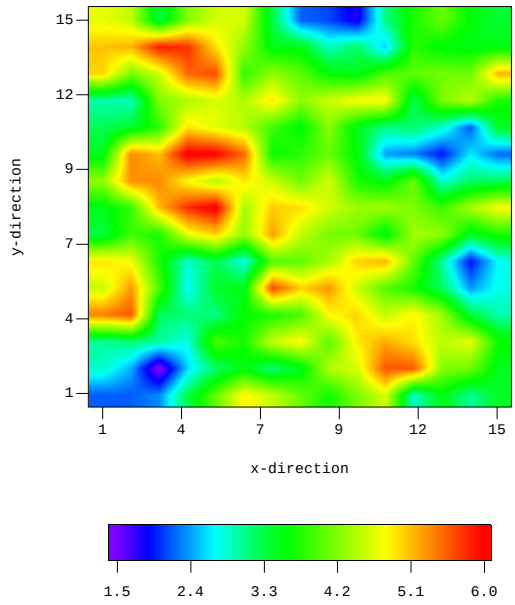


Fig. 6.30: Permeability field for the true model.

In the history matching process, we assume the prior mean for $\ln(k)$ is 3.0. Thus, when we generate an unconditional realization, m_{uc}, of the log-permeability field to use in the randomized maximum likelihood method, it is based on a lower mean for $\ln(k)$ than was used to generate the truth case. The correction to the prior

mean for $\ln(k)$ is assumed to be a Gaussian random variable with mean equal to 0 and variance equal to 0.5, respectively, when we generate unconditional realizations of the correction to the prior mean; see Eq. 2.31. We expected that by history matching the production data, we should obtain the value of the correction to the prior mean for $\ln(k)$ be around 1. Fig. 6.31 shows the 30 unconditional realizations (the triangles) and 30 conditional realizations (the solid circles) of the correction to the prior mean. We can see as expected the correction to the prior mean for $\ln(k)$ is around 1.

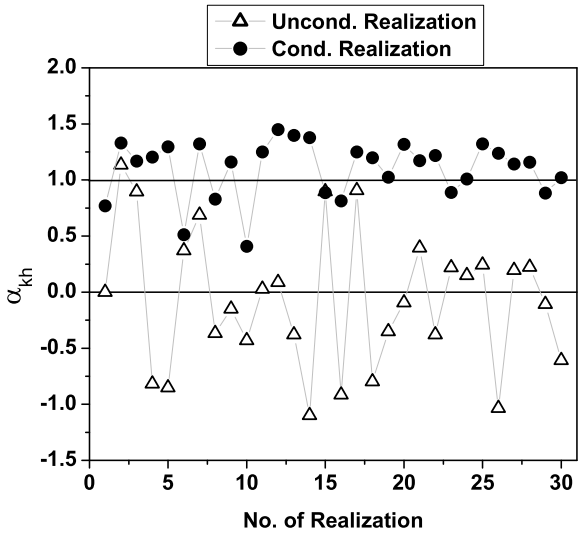


Fig. 6.31: Correction to mean of horizontal log-permeability.

In order to check whether the method with correction to the prior mean helps to improve the convergence or not, we performed a history matching for one realization without using correction to the prior mean. Without using correction to the prior mean, the objective function is reduced from 47,310,977 to 540 in 100 iterations and the corresponding behavior of the objective function is shown by plus signs in Fig. 6.32. When using correction to the prior mean in the history matching procedure, for the same realization the objective function is reduced from 47,310,993 to 440 in 66 iterations and the corresponding behavior of the objective function is shown by circles in Fig. 6.32. The final model obtained when the objective function

given by Eq. 2.29 was minimized in the history matching procedure is shown in Fig. 6.33 (a). The final model obtained when the objective function given by Eq. 2.18 was minimized is shown in Fig. 6.33 (b). We can see that the model shown in Fig. 6.33 (b) has bigger variation than the model shown in Fig. 6.33 (a) compared with the true model which is shown in Fig. 6.33 (d). Fig. 6.33 (c) shows the corresponding unconditional realization used as the initial guess in the history matching process to generate Fig. 6.33 (a) and (b).

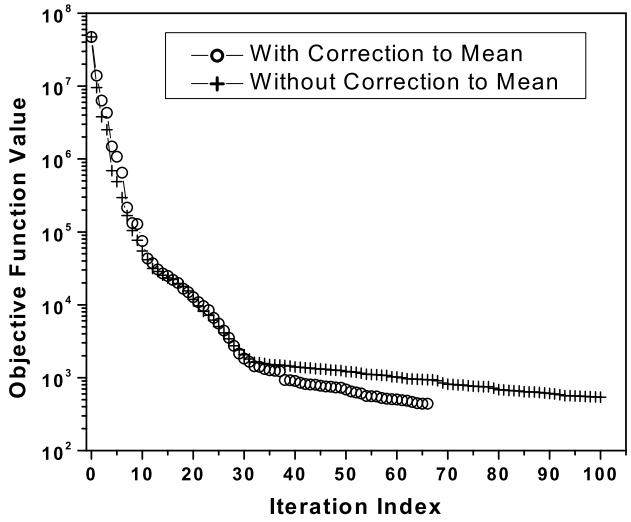
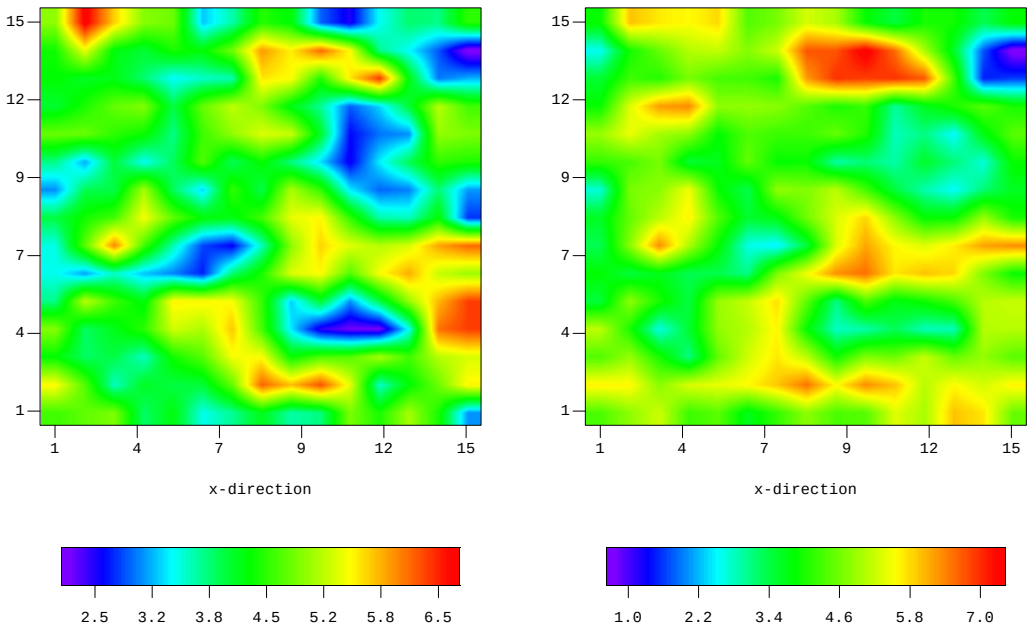
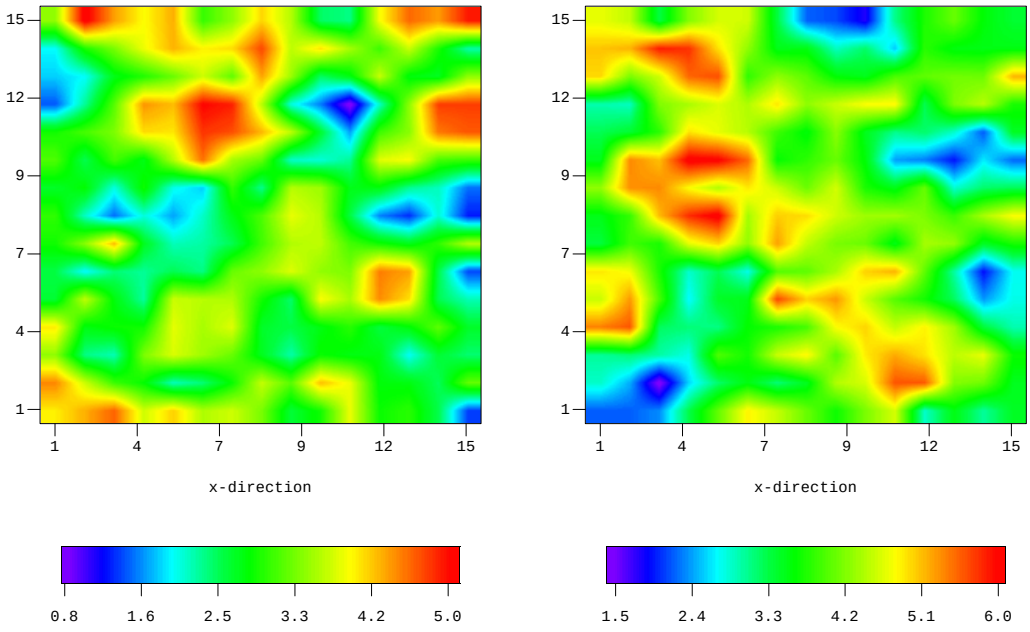


Fig. 6.32: Behavior of the objective function for the case with correction to the prior mean (circles) and the case without correction to the prior mean (plus signs).



(a) With Correction to Mean

(b) Without Correction to Mean



(c) Unconditional Model

(d) True Model

Fig. 6.33: The log-permeability field obtained by history matching with correction to the prior mean(a) and without correction to the mean (b); Unconditional realization of the model (c) and the true model (d).

6.5 Field Example – Oseberg Reservoir from North Sea

We consider a synthetic problem which is based on the Oseberg reservoir in the Norwegian sector of the North Sea. The reservoir consists of three distinct geological zones, Etive, Rannoch and Oseberg. Etive is the top zone and Oseberg is the bottom zone. These two zones are separated by Rannoch which is a relatively tight layer. There is vertical communication between the three zones. In our simulation study, we simulate only one half of the reservoir using a $39 \times 25 \times 10$ grid. Only one vertical gridblock is used in the Etive and Rannoch layer. So layer 3 through 10 are used to model the Oseberg zone. The gridblock size in the x-direction is equal to 328 ft in the central part of the reservoir, and the gridblock sizes expand gradually towards the ends from 328 ft to 2624 ft. The gridblock sizes in the y-direction are uniform and equal to 656 ft. Gridblock sizes in the z-direction are non-uniform with values equal to 23.0 ft in Etive, 16.5 ft in Rannoch and 11.5 ft in Oseberg. Initial reservoir pressure is 4071 psi at the depth of 8192 ft subsea, and the initial bubble point pressure is 3771 psi. The reservoir has a gas cap at the top and an aquifer at the bottom. The initial gas-oil contact is at 8192 ft subsea and the water-oil contact is at 8918 ft subsea. Fig. 6.34 (a) shows the surface plot of the reservoir top and (b) shows the overview of the reservoir top with well locations indicated by black squares for producers and by white squares for injectors. Note that the reservoir has a significant dip. The oil column is separated from the aquifer by a tar mat. The initial permeability and porosity field are based on a geostatistical model, and the true synthetic model is generated by using a Gaussian co-simulation algorithm.

Two gas injection wells, which are indicated by the white squares in Fig. 6.34 (b), are located in the gas cap and five producing wells, which are indicated by the black squares in Fig. 6.34 (b), are located in the oil zone. The producing wells are named PROD1, PROD2, $\cdots$, PROD5 and the two gas injection wells are named INJT1 and INJT2 respectively. All these wells are fully-penetrated, i.e., all layers are open to flow. The areal locations of the five producing wells are in gridblocks (32,3), (32,8), (32,13), (32,18) and (32,23) respectively. Fig. 6.35 (a) through (e)

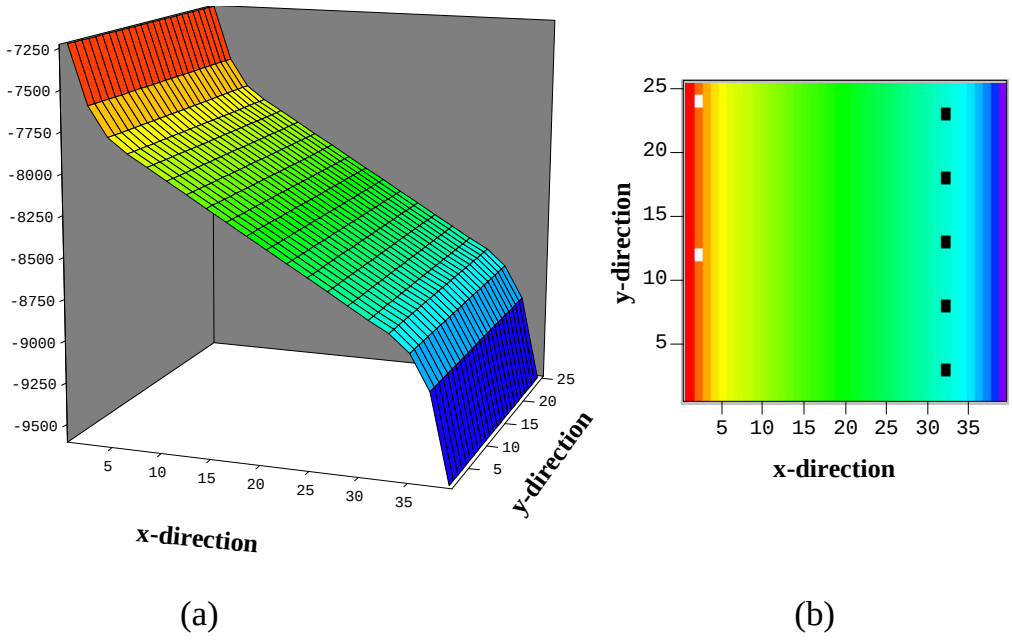


Fig. 6.34: Top depth of Oseberg reservoir and well locations.

show the production rate history for each well respectively. The production rate history for all producing wells are plotted together in Fig. 6.35 (f). We can see that these five wells are open to flow with a high flow rate (15000 STB/Day) in a sequential order (at 30, 90, 180, 270 and 360 days in turn). The oil rate shown in Fig. 6.35 are the rates which were specified as the target rates in the simulation runs. The minimum bottom-hole pressure (MINBHP) which is fixed at 2000 psi is used as the producing constraint. When the bottom-hole pressure falls below 2000 psi, this constraint will be switched to be the producing target. The maximum water-oil ratio (MAXWOR) which is specified to be 50 STB/STB and the maximum gas-oil ratio (MAXGOR) which is specified to be 561000 MSCF/STB are used as the producing economic limits. When these economic limits are violated at a certain well, then the corresponding well will be shut in. INJT1 and INJT2 are gas injection wells which are located in gridblocks (2,12) and (2,24) respectively. The gas injection rate history for the two gas injectors are plotted in Fig. 6.36. INJ1 starts to inject gas at 900 days and INJ2 starts to inject gas at 990 days. All six boundaries are assumed to be no-flow boundaries, and the capillary pressure is assumed to be negligible. In

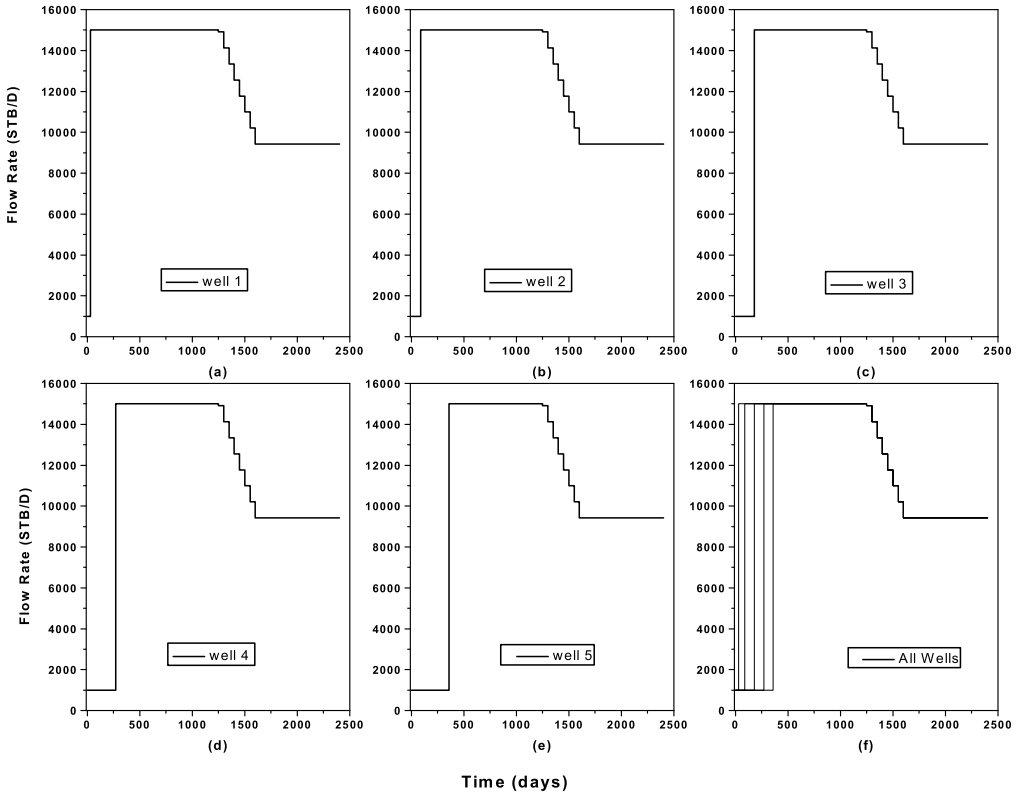


Fig. 6.35: Production rate history for the five producing wells.

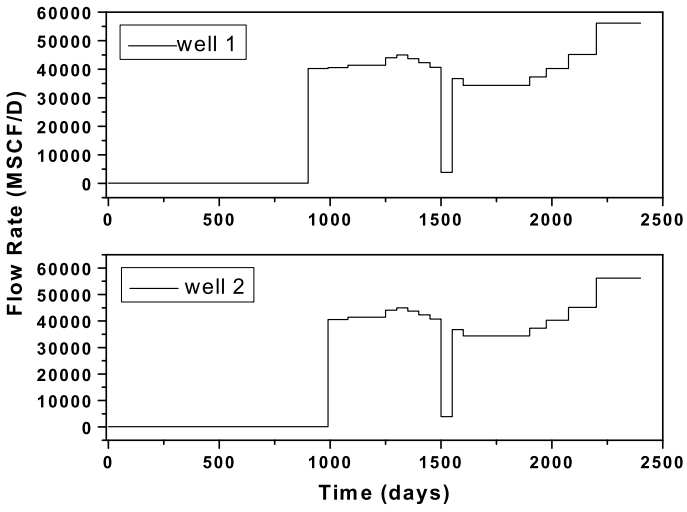
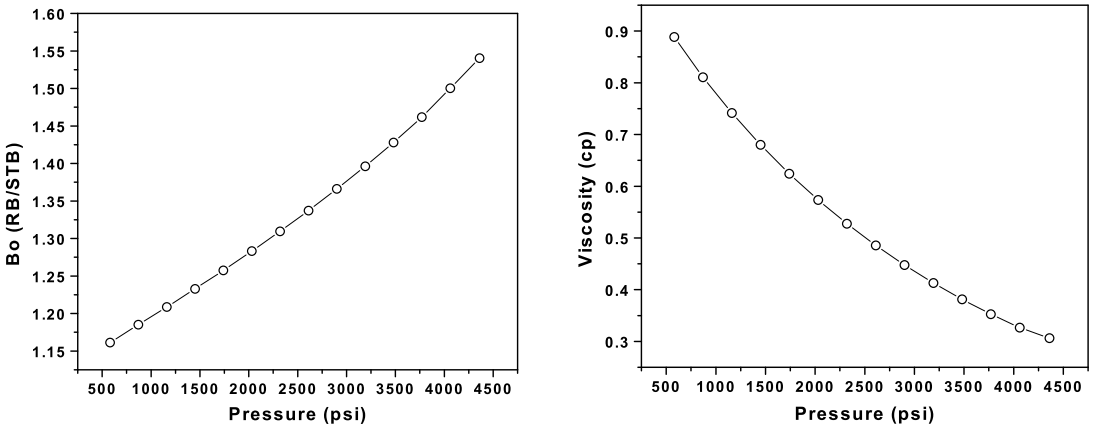


Fig. 6.36: Injection rate history for the gas injection wells.

the oil column, initial oil saturation is 0.885 and the initial water saturation is equal to the irreducible water saturation which is equal to 0.115. In the gas cap, the initial gas saturation is 0.885. The water saturation in the gas cap is the irreducible water saturation and there is no oil in the gas cap initially.

The oil formation volume factor (FVF) and the oil viscosity are shown in Fig. 6.37. Gas FVF and viscosity are shown in Fig. 6.38. Water FVF and viscosity are 1.03 RB/STB and 0.34 cp, respectively, at the reference pressure of 4219.5 psi. The water-oil two-phase relative permeability is shown in Fig. 6.39 (a) and oil-gas two-phase relative permeability is shown in Fig. 6.39 (b) respectively. Stone's model II is used to generate three-phase oil relative permeability; see Aziz and Settari (1979).

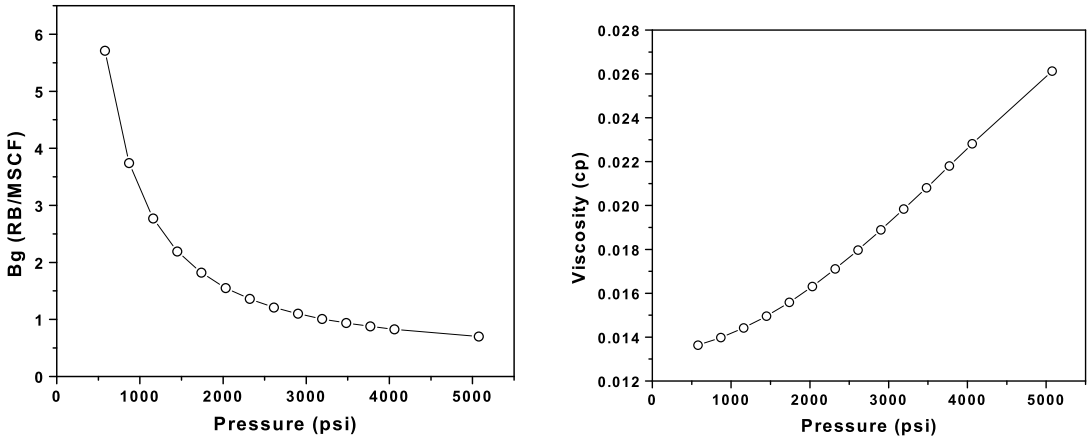


(a) FVF (b) Viscosity

Fig. 6.37: Oil FVF and viscosity.

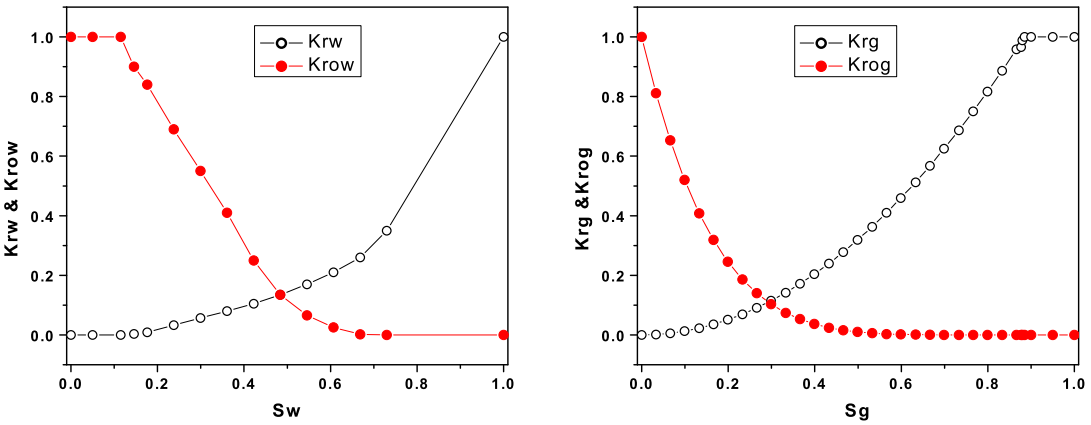
6.5.1 Reservoir Model

A non-isotropic exponential covariance structure, with ranges $a_x = 1968$ ft, $a_y = 6555$ ft and $a_z = 20$ ft was used. The statistical descriptions of the prior models for Etive and Rannoch, and Oseberg are given in Tables 6.13 and 6.14 respectively. Note that horizontal and vertical log-permeability in Oseberg decrease linearly from top to bottom, i.e., from layer 3 to layer 10. In Table 6.14, only the means for



(a) FVF (b) Viscosity

Fig. 6.38: Gas FVF and viscosity.



(a) Water-oil (b) Oil-gas

Fig. 6.39: Relative permeability.

model layer 3 and 10 are specified. Means for the intermediate model layers are obtained by linear interpolation. The unconditional realization of $\ln(k)$ and $\ln(k_z)$ for Oseberg layers was generated with fixed means. The mean value of 7.48 for $\ln(k)$ and 6.47 for $\ln(k_z)$ were used for generating the unconditional realization. Therefore, the unconditional realization for $\ln(k)$ and $\ln(k_z)$ do not have any trend vertically. The mean used for generating the unconditional realization for $\ln(k)$ and $\ln(k_z)$ for Etive layer are 6.02 and 4.61, respectively. The same means as used in generating the true Rannoch layer were used to generate the unconditional realization for $\ln(k)$ and

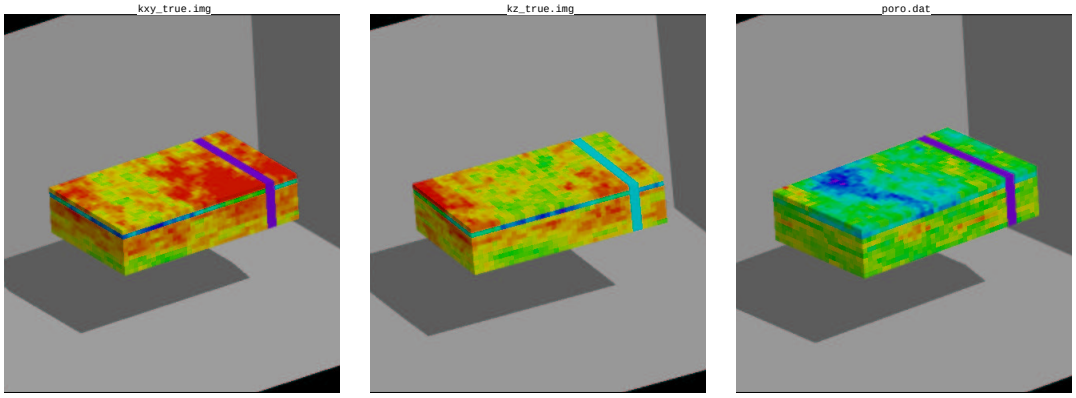
$\ln(k_z)$ for the Rannoch layer. The correlation between horizontal and vertical log-permeability, ρ_{k,k_z}, was equal to 0.8, and the correlations between log-permeability and porosity, $\rho_{k,\phi}$, $\rho_{k_z,\phi}$, was equal to 0.3. In the history matching process, the porosity field was fixed and equal to the true porosity. The tar-zone is located in gridblocks centered at (x_i, y_j, z_k), $i = 33, 34$, $1 \leq j \leq 25$, $1 \leq k \leq 10$. In the tar-zone, we set the horizontal and vertical permeability equal to 1 md in the true case. When history matching to generate a realization by the randomized maximum likelihood method, we first generate an unconditional realization m_{uc} from the prior model and then modify m_{uc} by setting the entries corresponding to $\ln(k)$ and $\ln(k_z)$ for the tar-zone to zero. As noted previously, the tar-zone prevents water conning from the aquifer. Fig. 6.40 (a) through (c) show the 3D cube of the horizontal log-permeability, $\ln(k)$, vertical log-permeability, $\ln(k_z)$, and porosity ϕ, respectively, for the true model. Fig. 6.41 (a) through (c) show the middle x-z cross-section corresponding to true $\ln(k)$, $\ln(k_z)$ and ϕ respectively. These figures indicate that Rannoch has very low permeability values compared to Etive and Oseberg, and acts as a flow restriction between the Etive and Oseberg formations.

	Etive		Rannoch	
	Mean	Variance	Mean	Variance
$\ln(k)$	7.5	1.2	2.1	1.8
$\ln(k_z)$	6.3	1.8	0.15	2.2
ϕ	0.14	0.002	0.10	0.001

Table 6.13: Prior model of Etive and Rannoch.

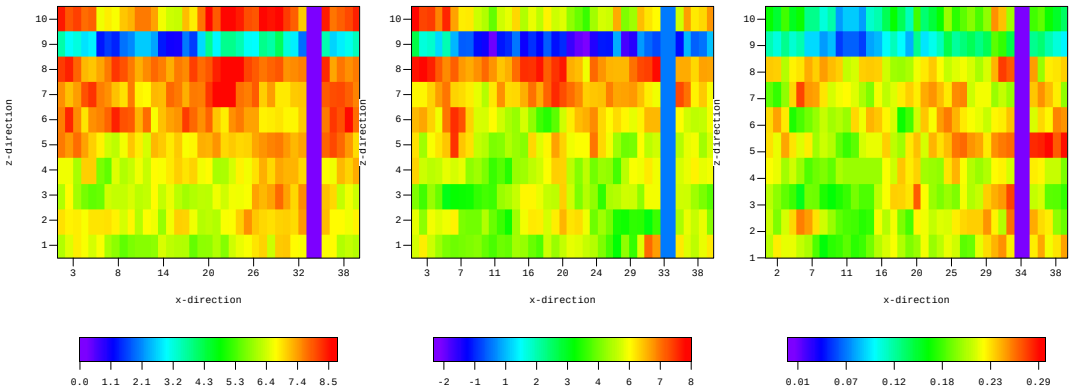
	Mean top	Mean bottom	Variance
$\ln(k)$	7.8	6.3	0.4
$\ln(k_z)$	6.4	4.4	0.8
ϕ	0.22	0.22	0.001

Table 6.14: Prior model of Oseberg.



(a) Horizontal log-perm. (b) Vertical log-perm. (c) Porosity

Fig. 6.40: Permeability and porosity for the true model.

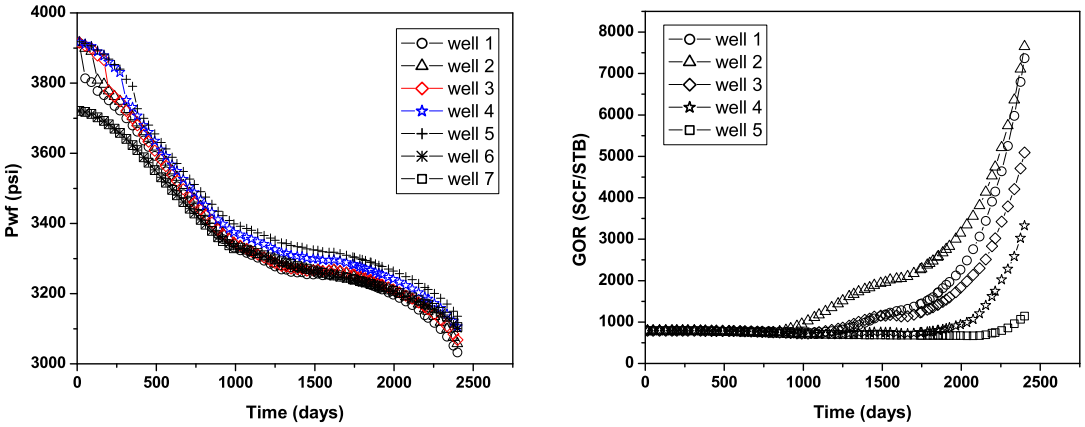


(a) Horizontal log-perm. (b) Vertical log-perm. (c) Porosity

Fig. 6.41: Permeability and porosity for the true model.

Fig. 6.42(a) shows the observed wellbore pressures, which are generated based on the true model, for all seven wells including both producers and injectors. Fig. 6.42(b) shows the GOR data from the five producing wells. From this figure, we can see that the gas breaks through at the five producing wells in a sequential order due to the fact that the producers start to produce in a time-delayed scheme. the exception is that the PROD2 experiences gas breakthrough earlier than PROD1. This occurs because PROD2 is much closer to the injection well than

PROD1. Fig. 6.43(a) shows wellbore pressures obtained from the unconditional realization which is used as initial model (initial guess) in the history matching procedure. Comparing Fig. 6.43(a) with Fig. 6.42(a), we can see that the behavior of the wellbore pressure for the initial model is different from the behavior of the wellbore pressure for the true model. Fig. 6.43(b) shows the GOR data for the initial model. We can see that, for the initial model, only PROD1 and PROD2 had gas breakthrough and the breakthrough happened very late compared with the true model. Breakthrough did not happen for PROD3 to PROD5.

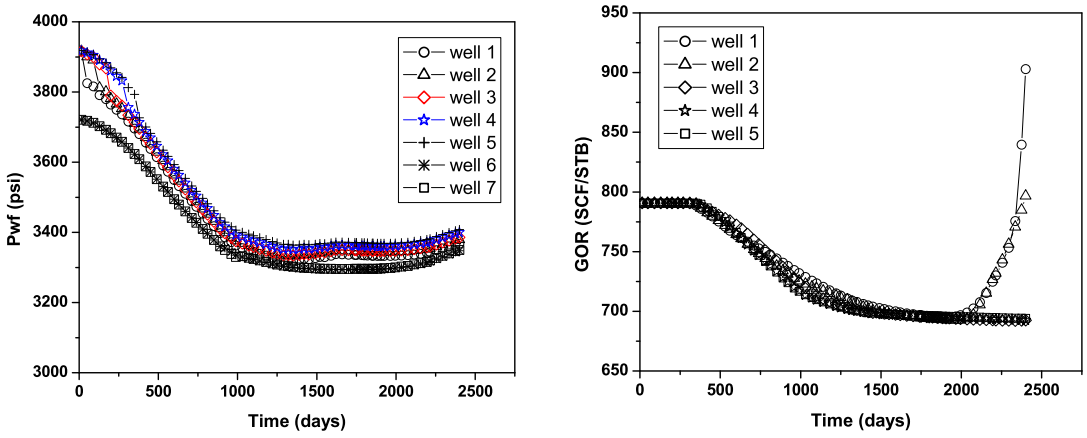


(a) Pressure (b) GOR

Fig. 6.42: Observed production data.

6.5.2 History Matching

The observed data used for history matching are constructed by running the simulator with the true model for a total time of 2400 days. Several data sets are used for history matching. Data set 1 contains the wellbore pressures from both the producing wells and injection wells and the GOR data from the producers. There are 71 measurements for each type of data at each well. So the total number of data history matched is 852 for the first data set. Data set 2 contains only the wellbore pressure from the five producing wells and the two gas injection wells. The total number of data history matched for the second data set is 497. Data set 3 contains



(a) Pressure (b) GOR

Fig. 6.43: Calculated data for the initial model.

only the GOR data from the five producing wells. So for this case, the total number of data used for history matching is 375. No noise was added to the data, i.e., we history matched the objective function given by Eq. 2.18 with d_{uc} replaced by d_{true}. We assumed wellbore pressure measurement errors to be independent Gaussian random variable with mean zero and variance equal to 1 psi^2 and gas-oil ratio measurement errors to be independent Gaussian error with mean zero and variance equal to 25.0. The limited memory BFGS algorithm was used for the minimization.

With data set 1, even though no noise was added to the true data, the objective function value is only reduced from 1.4×10^7 to 3×10^4 in 31 iterations which is much larger than N_d. As shown below, however, we obtained decent matches of the data. We restarted the algorithm once at the 19th iteration. The behavior of the objective function is shown by the triangles in Fig. 6.44. Pressure data matches for three producing wells, PROD1, PROD2, PROD4 and one injection well, INJ1, are shown in Fig. 6.45. In these four figures and in similar figures, the circles represent the observed data, the diamonds represent the calculated data based on the initial model and the plus signs represent the calculated data based on the final model after history matching the production data. From these four figures, we can see that we get a very good match for pressure data at both producing wells and the injection

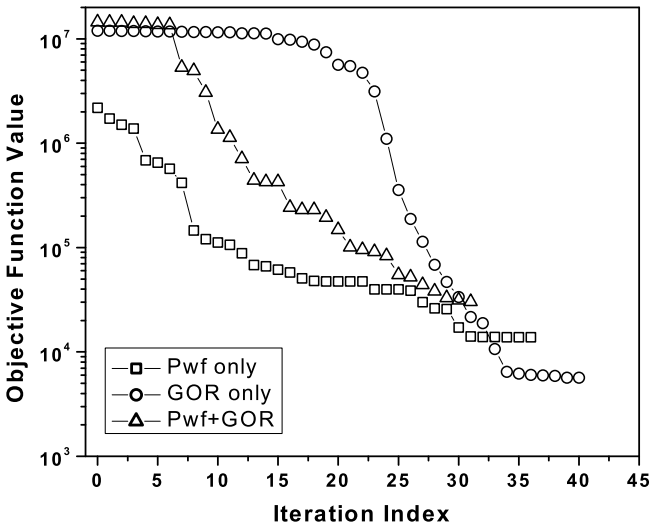


Fig. 6.44: Behavior of the objective function when the both pressure and GOR were history matched, data set 1.

well. Matches of comparable quality were obtained at the other three wells. Fig. 6.46 shows the GOR data match for PROD1, PROD2, PROD3 and PROD4. Again a very good match for GOR data at these four wells is obtained.

Fig. 6.47 shows the gas saturation at 2400 days for the first layer which corresponds to Etive (top row) and the third layer which is the first layer of the Oseberg (bottom row) corresponding to initial, true and final model which is obtained by history matching both pressure and GOR data. Fig. 6.48 shows the gas saturation at 2400 days for the 13th x-z cross-section (top row) and the 20th x-z cross-section (bottom row) corresponding to the initial, true and final models. From these figures, we can see that the gas distribution for the final model obtained by history matching the production data is similar to the gas distribution for the true model. For the initial model (see Fig. 6.47 (a) and (d) and Fig. 6.48 (a) and (d)), the gas moves more slowly towards the producing wells. The gas breaks through only at PROD1 and PROD2 and the breakthrough time is much later than for the true model. For the model obtained by history matching the production data, the gas breaks through at all wells at about the same time as for the true model. Fig. 6.49 shows 4 layers of

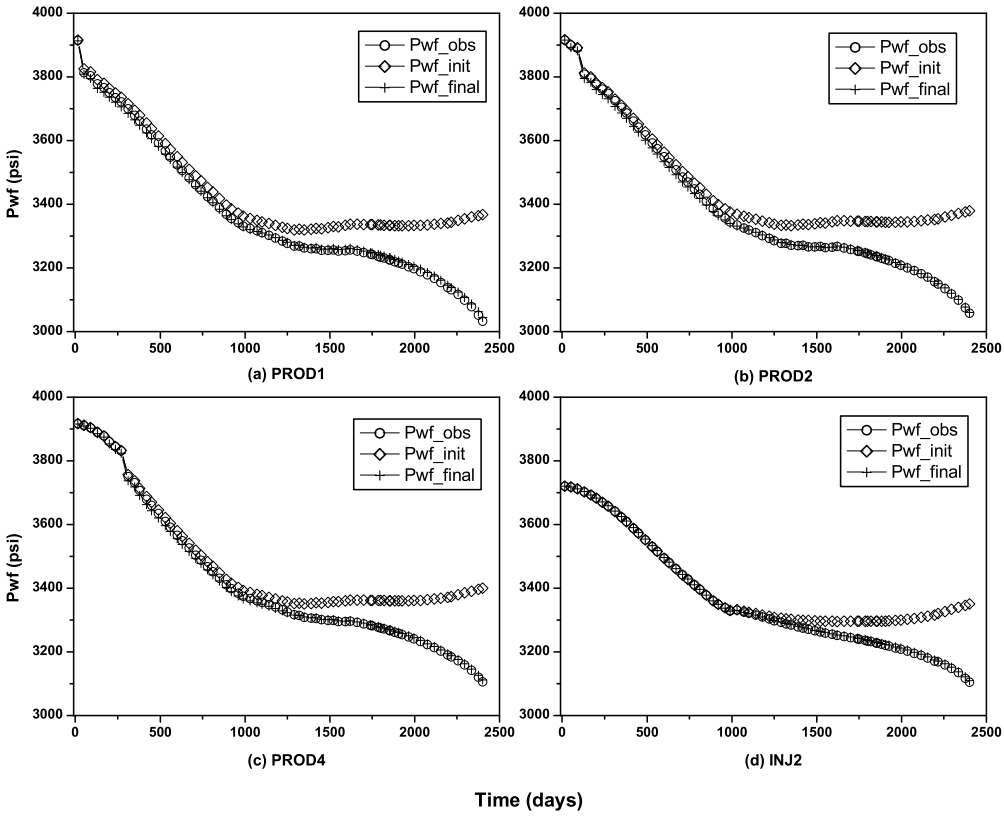


Fig. 6.45: Wellbore pressure data match at four wells, data set 1.

the horizontal log-permeability field corresponding to the initial, true and final model obtained by history matching data set 1. In this figure, the log-permeability fields for the same layer are plotted with the same scale. From these figures, it is not easy to see how the log-permeabilities change by history matching the production data. In order to see how the history matching changes the permeability field from the initial guess, we plotted in Fig. 6.50 the log-permeability change, i.e., the difference between the final model and the initial model, for six layers. From this figure, we can see clearly that for the first three layers, the permeabilities around some wells (especially at the PROD1 and PROD2) for the final model are larger than the permeabilities in the same area for the initial model which is an unconditional realization. Recall that the mean of log-permeability for the Etive layer and the top two layers of the Oseberg zone for the initial model is smaller than the true model. Therefore the

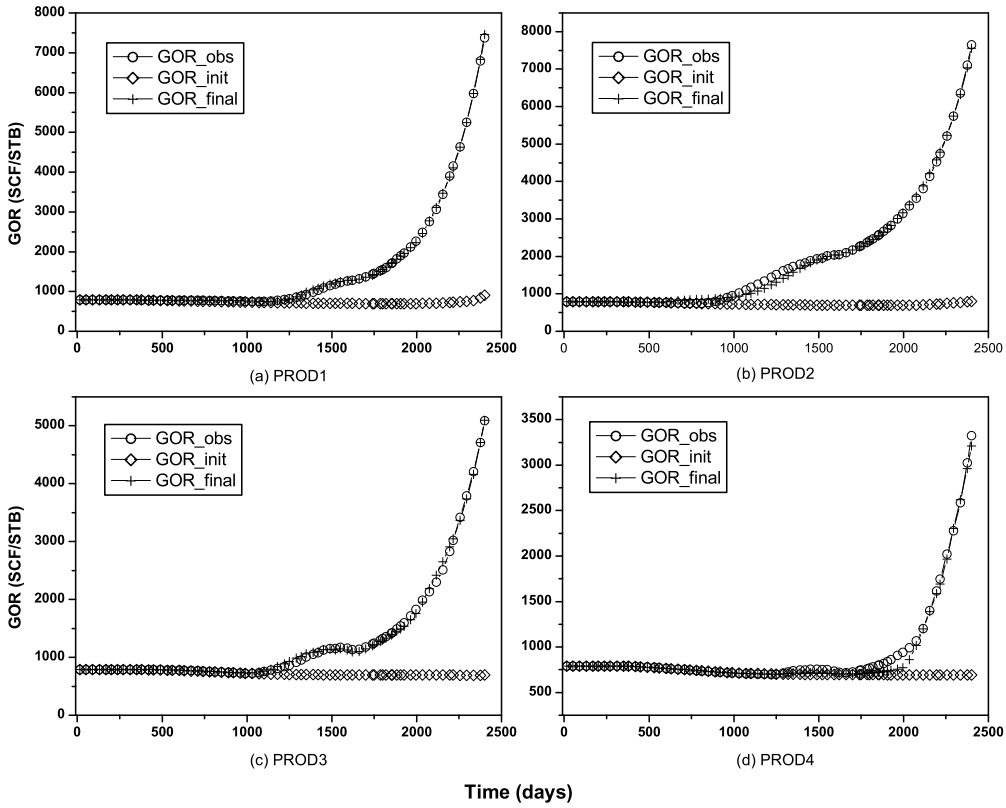


Fig. 6.46: GOR data match at four wells, data set 1.

permeabilities are increased for these layers in order to match the GOR data. We also can see that the permeabilities for the layers 7 through 9 are decreased. Even though we did not show it here, the permeabilities are decreased also for the bottom layer. Recall that the permeability for the true model has the decreasing trend vertically and the permeability for the layer 6 through 10 for the true model are smaller than the initial model. The vertical log-permeabilities were only changed slightly by history matching (less than 0.06 for the change in log-permeability).

When data set 2, which contains only the wellbore pressure data, is history matched, the objective function value is reduced from 2.2×10^6 to 1.4×10^4 in 36 iterations. The behavior of the objective function is shown by the squares in Fig. 6.44. Pressure data match for PROD1, PROD2, PROD4 and INJ1 are shown in Fig. 6.51. From these four figures, we can see that we obtain a reasonable match

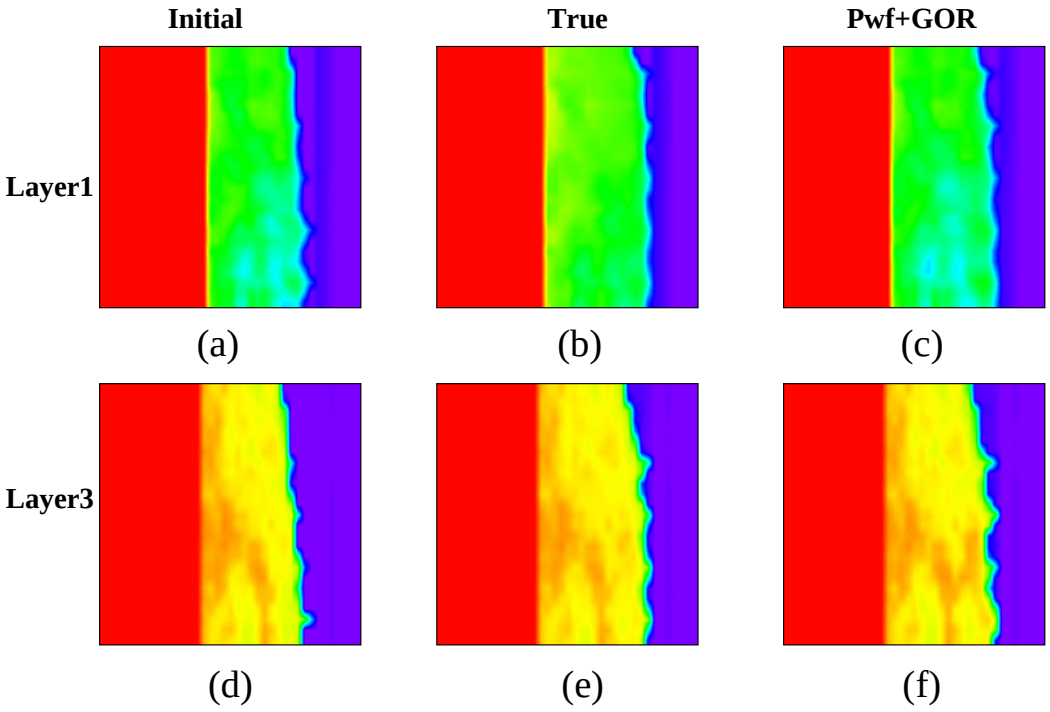


Fig. 6.47: Gas saturation at 2400 days in the first layer (top row) and the third layer (bottom row) corresponding to initial, true and final model from history matching wellbore pressure and GOR, data set 1.

for pressure data at both producing wells and the injection well. Based on the model obtained by history matching only the wellbore pressure data, we ran the simulator to calculate the GOR data at the five producing wells in order to compare with the observed GOR data. Fig. 6.52 shows the GOR data based on different reservoir models. The circles in these figures represent the observed GOR data obtained based on the true model; the diamonds represent the calculated GOR data based on the initial reservoir model and the stars represent the calculated GOR data based on the model obtained by history matching only wellbore pressure data. We can see that GOR data obtained from the model obtained by history matching wellbore pressure data moves towards the observations but does not give as nearly as good a match as when we actually matched GOR and pressure data (see Fig. 6.46) although calculated GOR data at PROD1 and PROD2 are fairly close to the observed GOR

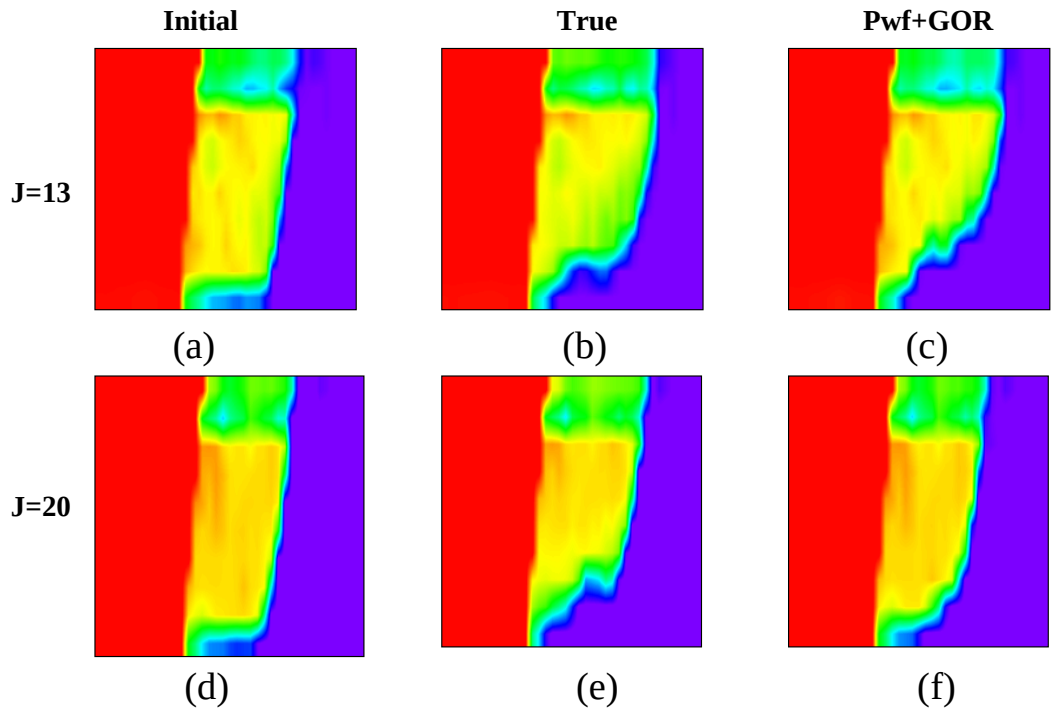


Fig. 6.48: Gas saturation at 2400 days for the 13th cross-section (top row) and the 20th cross-section (bottom row) corresponding to initial, true and final model from history matching wellbore pressure and GOR, data set 1.

data. These figures indicate that history matching wellbore pressure data improves the model to some extent, but if only one type of data are history matched then the other type data predicted based on this model do not match the observations as well as if they are explicitly included in the history matched data.

The final data set we history matched contains only GOR data from all five producing wells. Each producing well has 71 GOR data. So a total of 355 GOR data are history matched. The behavior of the objective function is shown in Fig. 6.44 by the circles. The value of the objective function is reduced from 1.2×10^6 to 5.7×10^3 in 40 iterations. The algorithm was restarted at the 10th iteration. The GOR data matches for PROD1 through PROD4 are shown in Fig. 6.53. Again, the circles represent the observed data; the plus signs represent the data calculated based on the model obtained after history matching the GOR data and the diamonds

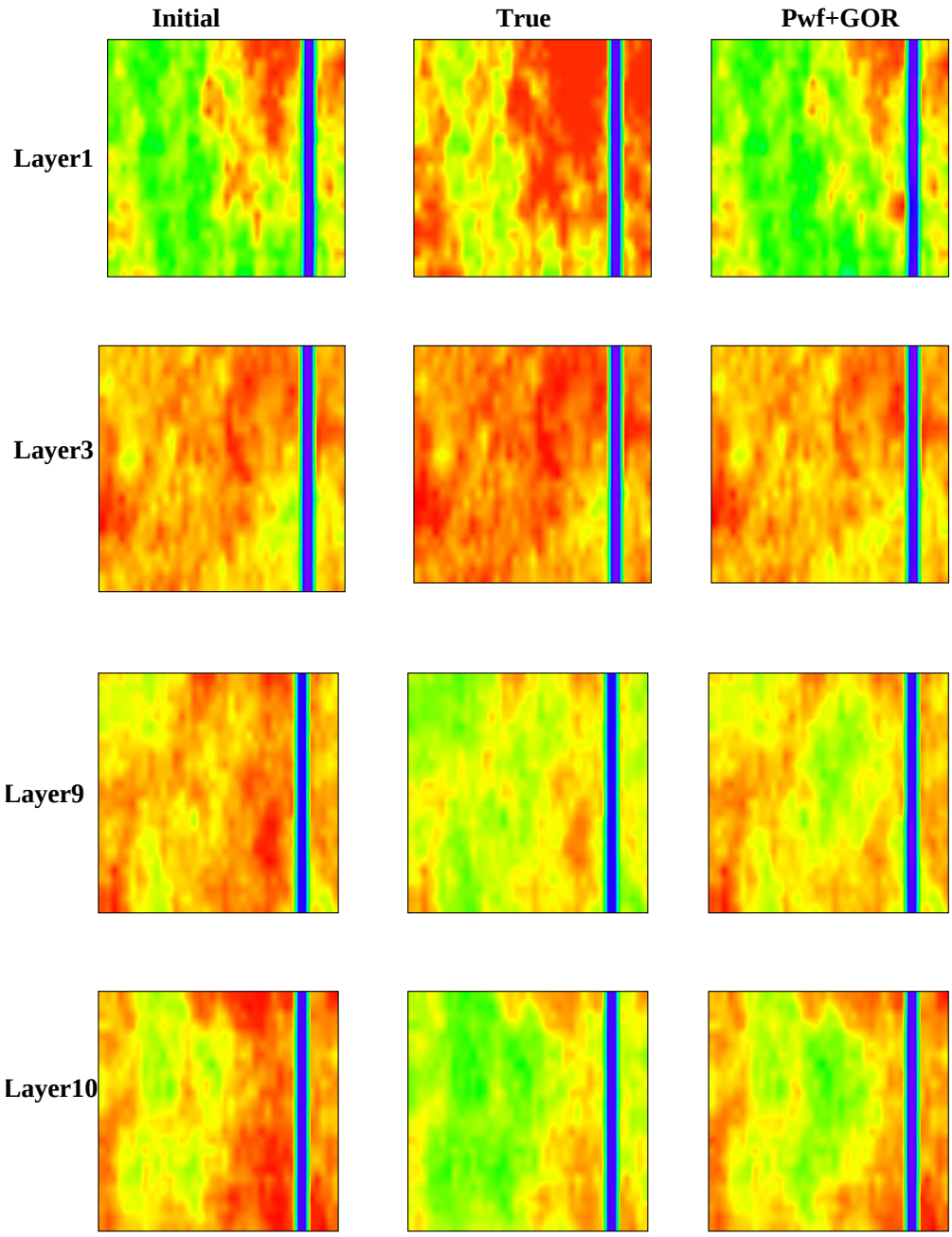


Fig. 6.49: Horizontal permeability field for 4 different layers corresponding to initial, true and final model obtained from history matching wellbore pressure and GOR, data set 1.

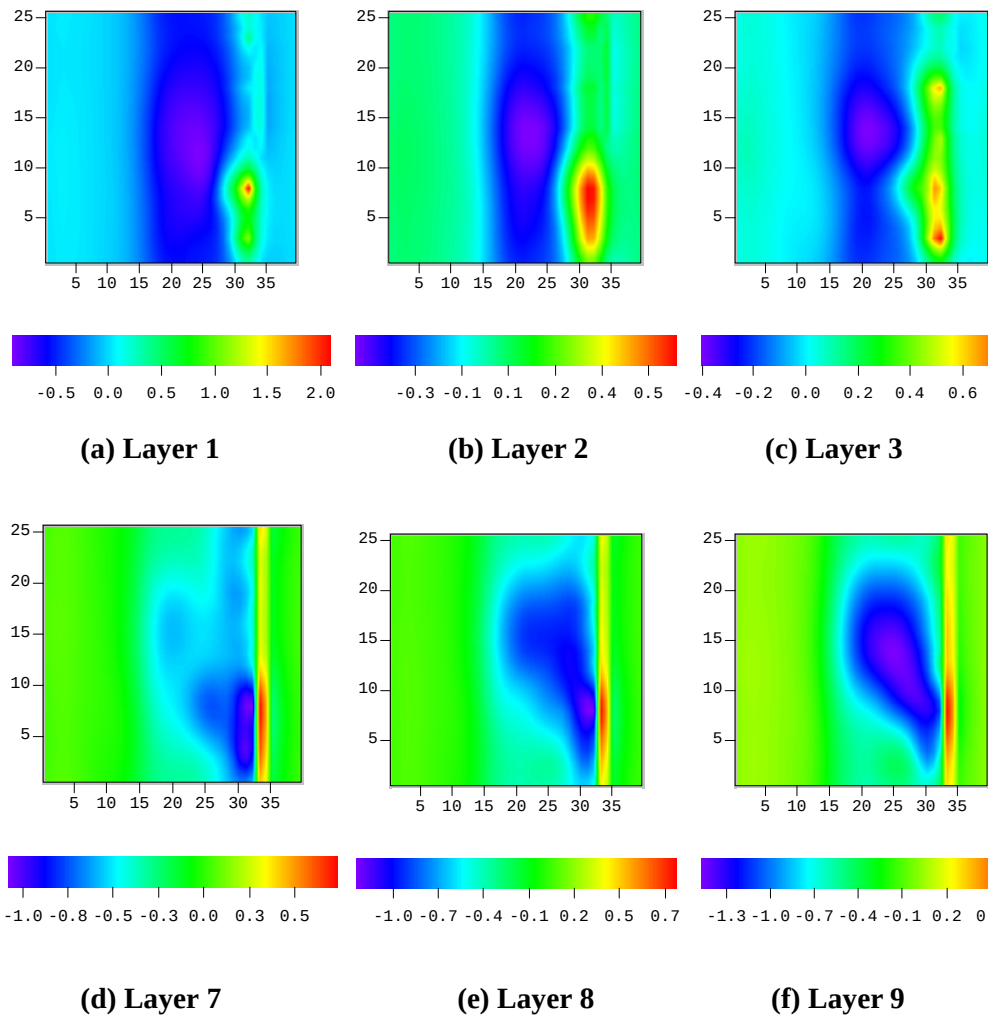


Fig. 6.50: Change in log-permeability for six layers.

represent the calculated data based on the initial model. We can see that we obtain a slightly better match of GOR data compared to the GOR match obtained by history matching both pressure and GOR data; see Fig. 6.53 and Fig. 6.46.

The stars in Fig. 6.54 shows the bottom-hole pressure data which are calculated based on the model obtained after history matching only GOR data. Again, in these figures, circles represent the observed data and diamonds represent the calculated data corresponding to the initial model. We can see that even though pressures

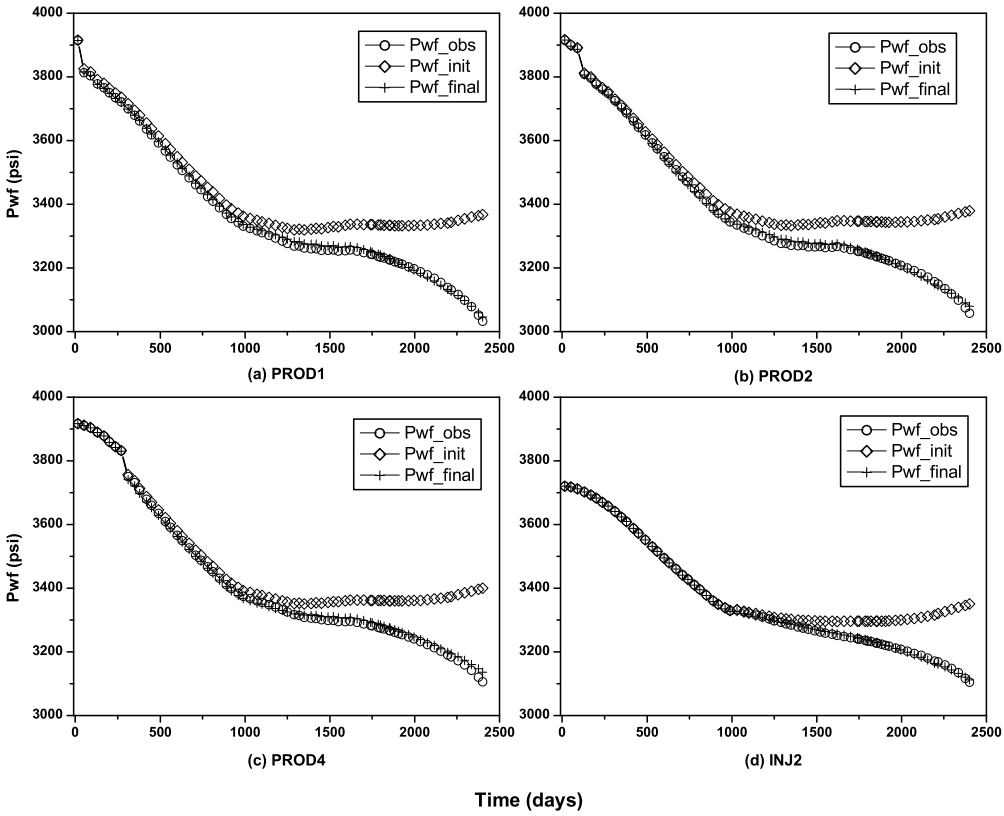


Fig. 6.51: Wellbore pressure data match at four wells, data set 2.

are not matched as well as the case where the pressure data were used as the conditioning data, the pressures predicted from the matched model are much closer to the observed data than those predicted from the initial model. Although we did not match any data from the injection wells, the observed pressure data are matched very well at the injection wells which implies that the GOR data at the producing wells can resolve the permeability around the injection wells very well. Wu (1999) also observed this phenomenon. Comparing the results given in Figs. 6.52 and 6.54 suggests that GOR data are more useful in resolving the permeability field than the pressure data.

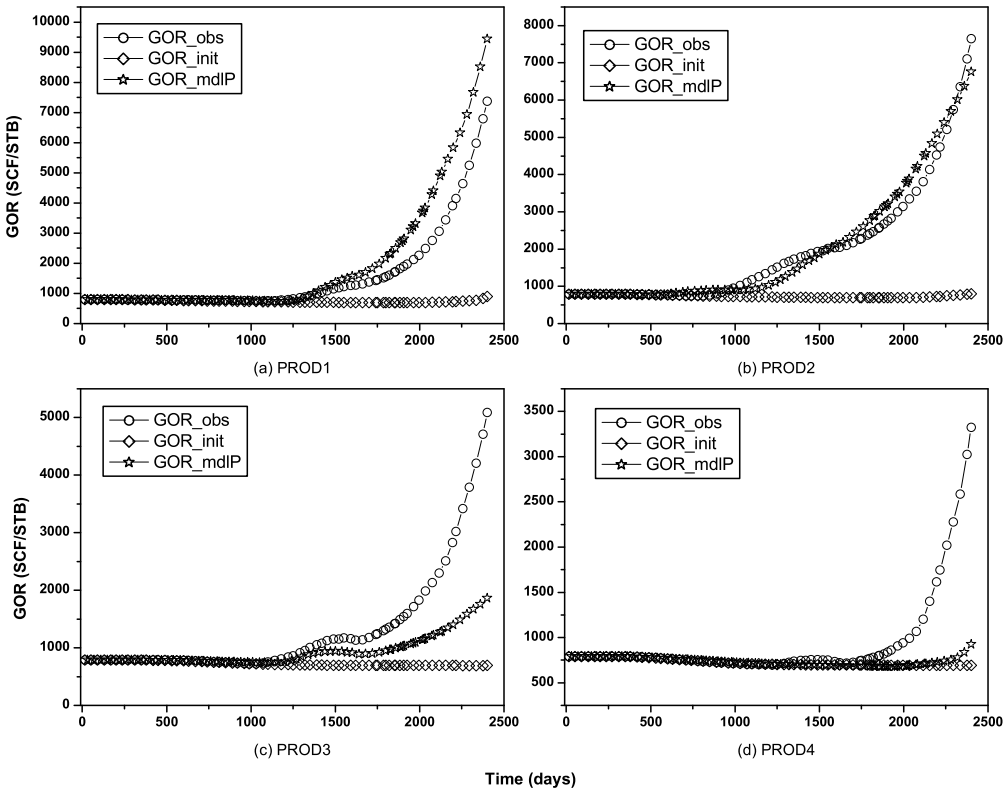


Fig. 6.52: GOR data obtained from initial model (diamonds), true model (circles) and the model obtained by history matching only wellbore pressure data (stars) at four wells, data set 2.

6.5.3 Future Performance Prediction

In making a future performance prediction, after the 2400 days production history, we let the five producing wells keep producing at the constant rate of 9000 STB/Day with the minimum bottom-hole pressure of 2000 psi as the constraint for another 400 days, i.e., from 2400 days to 2800 days. At 2800 days, we let the five producing wells produce at a fixed constant bottom-hole pressure of 2000 psi for 1100 days, i.e., from 2800 days to 3900 days. After the first 2400 days history, the two injection wells were shut in for 200 days from 2400 days to 2600 days and then we started injecting gas again at the rate of 18715 MSCF/Day, which is one third of

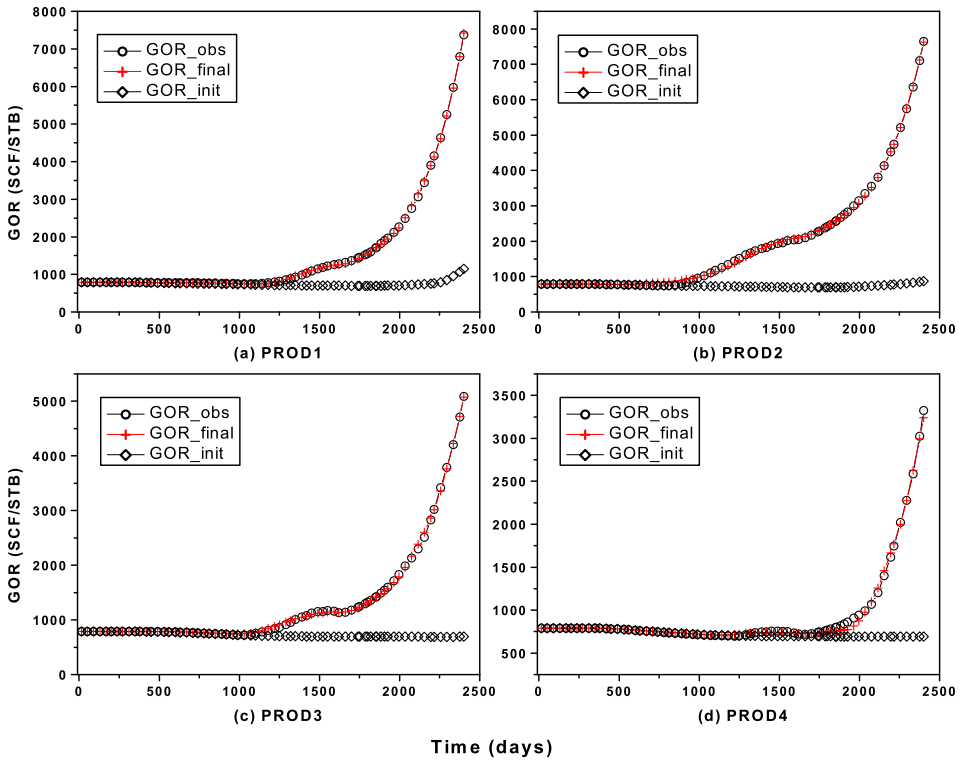


Fig. 6.53: GOR data obtained from initial model (diamonds), true model (circles) and final model obtained by history matching only GOR data (plus signs) at four wells, data set 3.

the injection rate at the end of the observed history, for another 200 days, i.e., from 2600 days to 2800 days. Then the injection rates were changed to 56145 MSCF/Day, which is equal to the rate at the end of the history, for the rest of the time in the prediction period in order to provide enough pressure support. So the total time span for the future performance prediction is 1500 days, i.e., from 2400 days to 3900 days. The simulation run were performed for the initial model, true model and the models obtained by history matching the three different data sets described in the previous section. The total cumulative oil production obtained for the five models are shown in Fig. 6.55. Fig. 6.55 (b) shows the total cumulative oil production only for the period of time for future prediction. In this figure, the solid line, dashed line, short dashed line, dot-dash line and the dotted line, respectively, represent the total cumulative oil production from the true model, the model obtained by history matching both

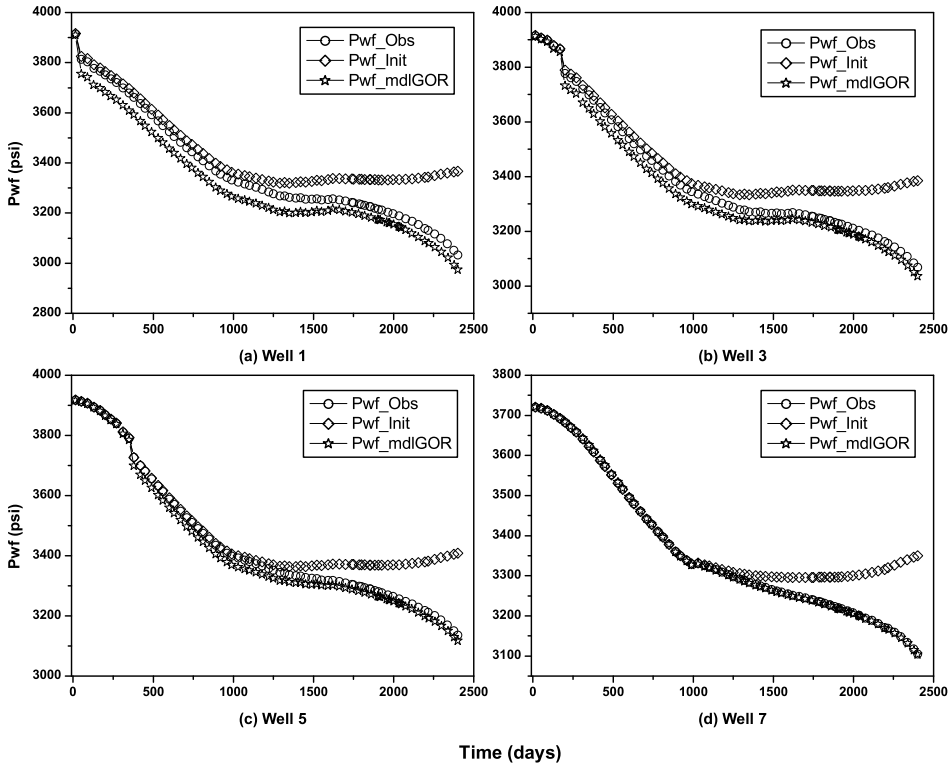
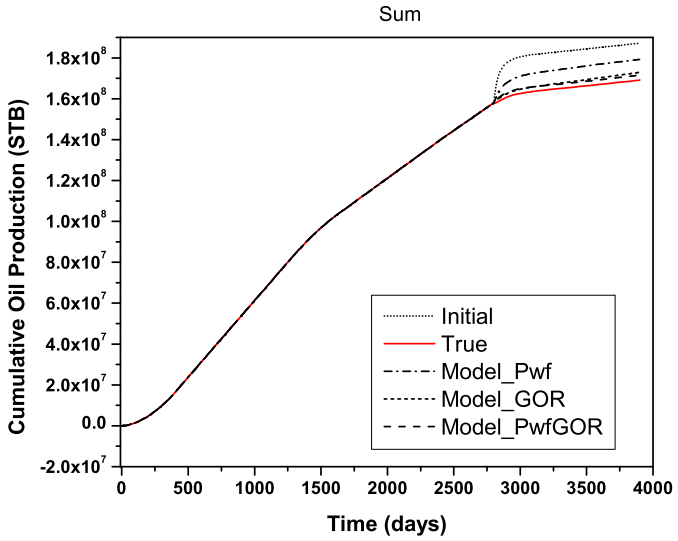
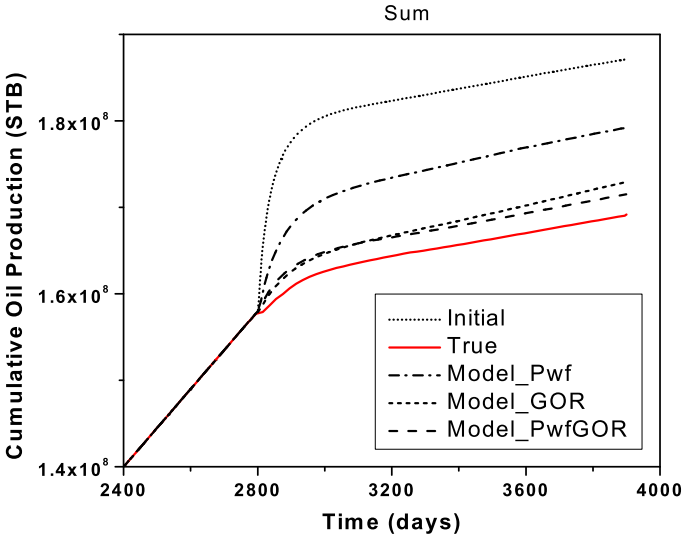


Fig. 6.54: data obtained from initial model (diamonds), true model (circles) and the model obtained by history matching only GOR data (stars) at four wells, data set 3.

pressure and GOR data, the model obtained by history matching only GOR data, the model obtained by history matching only pressure and the initial model. From this figure, we can see that the cumulative oil productions based on the model obtained by conditioning the initial model to both pressure data and GOR data and the model obtained by conditioning only to the GOR data are very close to the cumulative oil production corresponding to the true model. The total cumulative oil production obtained for the initial model is far away from that obtained for the true model. From the future performance prediction, we also can see that the GOR data are more useful than pressure data. Fig. 6.56 (a) through (e) show the predictions of GOR obtained based on the five different models for PROD1 through PROD5.

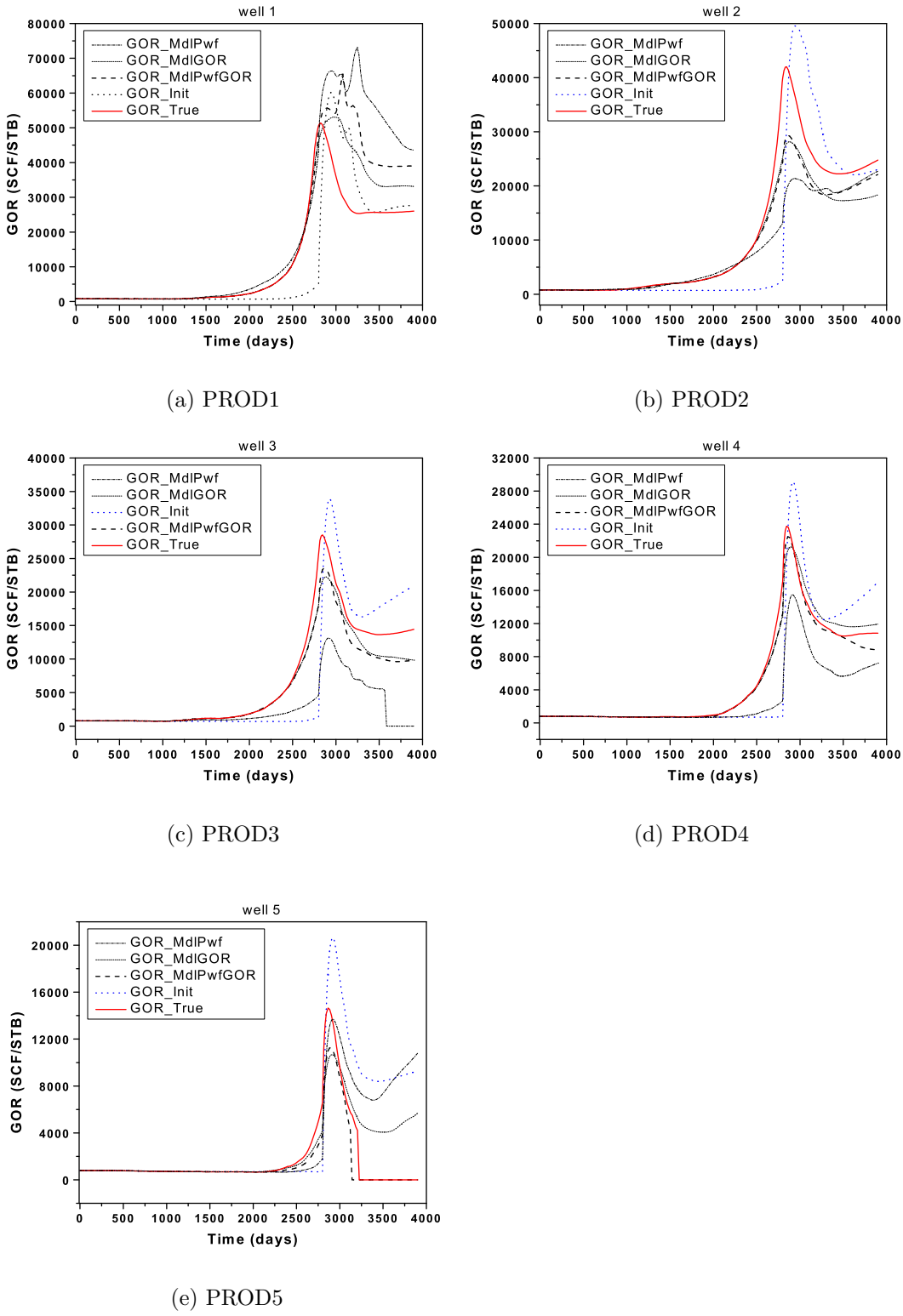


(a) 0-3900 days



(b) 2400-3900 days

Fig. 6.55: Total field cumulative oil production.



(a) PROD1

(b) PROD2

(c) PROD3

(d) PROD4

(e) PROD5

Fig. 6.56: GOR data for all producing wells for the production history and the future performance predictions.

CHAPTER VII

CONCLUSIONS

The main objective of this work was to develop and implement an automatic history matching procedure that is computationally efficient enough to be applicable to large scale problems. For large scale problems where the number of data and the number of model parameters are both large, it is too expensive to apply optimization algorithms which require the Hessian of the objective function, such as the Gauss-Newton and Levenberg-Marquardt methods, to minimize the objective function involved in history matching. In this work, we compared the convergence performance of a set of gradient based nonlinear optimization algorithms including modified Levenberg-Marquardt, preconditioned conjugate gradient, BFGS and limited memory BFGS on a set of history matching problems. The implementation of BFGS used was based on explicitly computing and storing the approximate inverse Hessian at each iteration; although computationally inefficient, this implementation allows one to apply all scaling procedures that have been suggested in the literature. Our results indicate that for large scale history matching problems, the limited memory BFGS algorithm requires significantly less time and less memory than the modified Levenberg-Marquardt and BFGS algorithms, but yields results of comparable quality based on the value of the objective function obtained at convergence and the model obtained at convergence. Scaling has a significant effect on the performance of the LBFGS and BFGS algorithms. The scaling factors used here result in significant improvement in the convergence properties of the algorithm as compared to the no scaling cases. For all examples considered, our implementations of preconditioned conjugate gradient algorithms were less robust than the scaled BFGS and LBFGS algorithms. Based on our results, we conclude that the limited

memory BFGS is the only viable algorithm for large scale 3D three-phase automatic history matching problems where the number of model parameters and the number of data are both large. Conceptually, a preconditioned nonlinear conjugate gradient method could be competitive with the limited memory BFGS method, but we have not identified an effective preconditioner to achieve this result.

An iterative linear solver based on orthomin theory was implemented in this work. Essentially, this iterative solver is the same as the solver used in CLASS (Chevron Limited Application of Simulation System) simulator for solving the finite-difference flow equation system. For large problems, the iterative solver is orders of magnitude faster than the direct solver which is based on the LU decomposition. The iterative solver was used to solve the adjoint equation system which is a linear system. In this application, it is important to obtain a reasonable initial guess for the adjoint variables in order to avoid convergence failure. We have presented an ad hoc procedure for generating a sufficiently good initial guess. With this procedure, the solution obtained by the iterative solver is in excellent agreement with the solution obtained by a sparse matrix technique.

The iterative solver and limited memory BFGS developed in this work were applied to several problems including a synthetic example based on a North Sea reservoir. In this example, we history matched only GOR data, then matched the wellbore pressure only and then matched both pressure and GOR. The limited results suggest that the information content of GOR data is as good as or superior to wellbore pressure data at least in terms of reducing the uncertainty in future performance prediction.

NOMENCLATURE

$C =$ covariance matrix.

$C_D =$ data covariance matrix.

$C_M =$ model covariance matrix.

$C_r =$ covariance matrix for rock property field.

$C_s =$ covariance matrix for well skin factors.

$d =$ vector of data (units depend on data type).

$d_{\text{obs}} =$ vector of observed data.

$d_{\text{uc}} =$ observed data with random noise added.

$G =$ matrix of sensitivity coefficients.

$g(m) =$ calculated data based on m.

$H =$ Hessian matrix.

$\tilde{H} =$ approximation to the Hessian matrix.

$I =$ Identity matrix

$J =$ Jacobian matrix

$k =$ permeability, mD.

$m =$ vector of model parameters.

$m_{\text{prior}} =$ vector of prior mean for model parameters.

$m_{\text{uc}} =$ vector of unconditional realization of model parameters.

$N =$ total number of gridblocks.

$N_d =$ number of data.

$N_e =$ total number of flow equations.

$N_m =$ number of model parameters.

$N_w =$ number of wells.

$N_x =$ number of gridblocks in x-direction.

$N_y =$ number of gridblocks in y-direction.

$N_z =$ number of gridblocks in z-direction.

$O() =$ objective function.

$P() =$ probability density function.

$p_{wf} =$ wellbore pressure.

$p =$ gridblock pressure.

$q =$ flow rate.

$r =$ residual.

$R_s =$ solution gas oil ratio.

$S =$ saturation.

$Z =$ vector of standard random normal deviates

$\alpha =$ vector of corrections to the prior mean, or step size.

$\epsilon =$ random noise in pressure measurements.

$\lambda =$ adjoint variable.

$\mu =$ step size in Gauss-Newton method.

$\gamma =$ scaling factor.

$\phi =$ porosity.

Subscripts

$M =$ model.

obs $=$ observed.

prior $=$ a priori value.

uc $=$ unconditional.

Superscripts

T $=$ transpose.

$-1 =$ inverse.

Bibliography

Sigurd I. Aanonsen, Alberto Cominelli, Olivier Gosselin, Ivar Aavatsmark, and Tor Barkve. Integration of 4D data in the history match loop by investigating scale dependent correlations in the acoustic impedance cube. In *European Conference on the Mathematics of Oil Recovery, VIII*, pages 1–8, 2002.

Y. Abacioglu, D. S. Oliver, and A. C. Reynolds. Efficient history-matching using subspace vectors. In *TUPREP Research Report 17 (May 22, 2000)*, pages 69–90, 2000.

Y. Abacioglu, D. S. Oliver, and A. C. Reynolds. Efficient reservoir history matching using subspace vectors. *Computational Geosciences*, 5, 151–172, 2001.

Yafes Abacioglu. *The Use of Subspace Methods for Efficient Conditioning of Reservoir Models to Production Data*. Ph.D. thesis, University of Tulsa, Tulsa, Oklahoma, 2001.

F. Anterion, B. Karcher, and R. Eymard. Use of parameter gradients for reservoir history matching, SPE-18433. In *10th SPE Reservoir Simulation Symp.*, pages 339–354, 1989.

O. Axelsson and I. Gustafsson. On the use of preconditioned conjugate gradient method for red-black ordered five-point different scheme. *Journal of Computational Physics*, 35, 284–289, 1980.

Owe Axelsson. *Iterative Solution Methods*. Cambridge University Press, 40 West 20th Street, New York, NY 10011-4211,USA, 1996.

Khalid Aziz and A. Settari. *Petroleum Reservoir Simulation*. Elsevier Applied Science Publishers, London, 1979.

James V. Beck and Kenneth J. Arnold. *Parameter Estimation in Engineering and Science*. John Wiley & Sons, Chichester, England, 1977.

Zhuoxin Bi. *Conditioning 3D Stochastic Channels to Well-Test Pressure Data*. Ph.D. thesis, University of Tulsa, Tulsa, Oklahoma, 1999.

Zhuoxin Bi, Dean S. Oliver, and Albert C. Reynolds. Conditioning 3D stochastic channels to pressure data. *SPE Journal*, 5(4), 474–484, 2000.

Robert Bissell. Calculating optimal parameters for history matching. In *4th European Conference on the Mathematics of Oil Recovery*, 1994.

Robert Bissell, O. Dubrule, P. Lamy, P. Swaby, and O. Lepine. Combining geostatistical modelling with gradient information for history matching: The pilot point method, SPE 38730. In *Proceedings of the 1997 SPE Annual SPE Technical Conference and Exhibition*, pages 139–154, 1997.

Robert Bissell, Yogeshwar Sharma, and J. E. Killough. History matching using the method of gradients: Two case studies. *SPE 69th Annual Technical Conference and Exhibition*, SPE 28590, 275–289, 1994.

Luciane Bonet-Cunha, D. S. Oliver, R. A. Rednar, and A. C. Reynolds. A hybrid Markov chain Monte Carlo method for generating permeability fields conditioned to multiwell pressure data and prior information. *SPE Journal*, 3(3), 261–271, 1998.

C. G. Broyden. Quasi-Newton methods and their application to function minimization. *Math. Comp.*, 21(99), 368–381, 1967.

C. G. Broyden. The convergence of a class of double rank minimization algorithm parts I and II. *J. of Institute of Mathematics and its Applications*, 6, 76–90 and 222–231, 1970.

A.G. Buckley. A combined conjugate-gradient quasi-Newton minimization algorithm. *Mathematical Programming*, 15(2), 200–210, 1978a.

A.G. Buckley. Extending the relationship between the conjugate gradient and BFGS algorithms. *Mathematical Programming*, 15(3), 343–348, 1978b.

A.G. Buckley and A. Lenir. QN-like variable storage conjugate gradients. *Mathematical Programming*, 27(2), 155–175, 1983.

R. D. Carter, L. F. Kemp, A. C. Pierce, and D. L. Williams. Performance matching with constraints. *Soc. Petrol. Eng. J.*, 14(4), 187–196, 1974.

Guy M. Chavent, M. Dupuy, and P. Lemonnier. History matching by use of optimal control theory. *Soc. Petrol. Eng. J.*, 15(1), 74–86, 1975.

W. H. Chen, G. R. Gavalas, John H. Seinfeld, and Mel L. Wasserman. A new algorithm for automatic history matching. *Soc. Petrol. Eng. J.*, 14(6), 593–608, 1974.

Lifu Chu, M. Komara, and R. A. Schatzinger. An efficient technique for inversion of reservoir properties using iteration method. *SPE Journal*, 5(1), 71–81, 2000.

Lifu Chu, Albert C. Reynolds, and Dean S. Oliver. Computation of sensitivity coefficients for conditioning the permeability field to well-test data. *In Situ*, 19(2), 179–223, 1995.

G. de Marsily, G. Lavedan, M. Boucher, and G. Fasanino. Interpretation of interference tests in a well field using geostatistical techniques to fit the permeability distribution in a reservoir model. In *Geostatistics for Natural Resources Characterization, Part 2*, pages 831–849. D. Reidel, 1984.

T. Deschamps, T. Grussaute, D. Mayers, and R. Bissel. The results of tesing six different gradient optimisers on two history matching problems. In *European Conference on the Mathematics of Oil Recovery, VI*, pages 1–8, 1998.

L.C.W. Dixon. Quasi-Newton algorithms generate identical points. *Mathematical Programming*, 2, 383–387, 1972.

R. Fletcher. A new approach to variable metric algorithms. *Computer Journal*, 13, 317–322, 1970.

R. Fletcher and M. J. D. Powell. A rapidly convergent descent method for minimization. *Computer Journal*, 6, 163–168, 1963.

R. Fletcher and C. M. Reeves. Function minimization by conjugate gradient. *Computer Journal*, 7, 149–154, 1964.

Roger Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, New York, second edition, 1987.

G. R. Gavalas, P. C. Shah, and John H. Seinfeld. Reservoir history matching by Bayesian estimation. *Soc. Petrol. Eng. J.*, 16(6), 337–350, 1976.

D. Goldfarb. A family of variable-metric methods derived by variational means. *Maths. Comp.*, 24(109), 23–26, 1970.

A. A. Goldstein. On steepest descent. *SIAM J. control*, 3(1), 147–151, 1965.

J. Jaime Gómez-Hernández and André G. Journel. Joint sequential simulation of multigaussian fields. In A. Soares, editor, *Geostatistic Troia 92*, pages 133–144, 1992.

J. Jaime Gómez-Hernández, Andrés Sahuquillo, and José E. Capilla. Stochastic simulation of transmissivity fields conditional to both transmissivity and piezometric data. 1. Theory. *Journal of Hydrology*, (203), 162–174, 1997.

Anne Greenbaum. *Iterative Methods for solving linear systems*. Society for Industrial and Applied Mathematics, University City Science Center, Philadelphia, PA, 1997.

N. He, A. C. Reynolds, and D. S. Oliver. Three-dimensional reservoir description from multiwell pressure data and prior information. *Soc. Pet. Eng. J.*, 2(3), 312–327, 1997.

Bjørn Kåre Hegstad, Henning Omre, Håkon Tjelmeland, and Kelly Tyler. Stochastic simulation and conditioning by annealing in reservoir description. In M. Armstrong and P. A. Dowd, editors, *Geostatistical Simulation*, Kluwer Acad., pages 43–55, 1994.

M. R. Hestenes and E. S. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. Nat. Bur. Stand*, 46, 409–536, 1952.

Lars Holden, R. Madsen, A. Skorstad, K. A. Jakobsen, C. B. Tjølsen, and Sven Vik. Use of well test data in stochastic reservoir modelling, SPE 30591. In *Proceedings of the SPE annual Technical Conference and Exhibition*, 1995.

H. Y. Huang. Unified approach to quadratically convergent algorithms for function minimization. *Journal of Optimization Theory and Application*, 5(6), 405–423, 1970.

P. Jacquard and C. Jain. Permeability distribution from field pressure data. *Soc. Petrol. Eng. J.*, 5(4), 281–294, 1965.

Hans O. Jahns. A rapid method for obtaining a two-dimensional reservoir description from well pressure response data. *Soc. Petrol. Eng. J.*, 6(12), 315–327, 1966.

R. Kalita and A. C. Reynolds. Application of the conjugate gradient method to conditioning a gas reservoir model to pressure data. In *TUPREP Research Report 17 (May 22, 2000)*, pages 107–126, 2000.

Rintu Kalita. *Conditioning a Three Dimensional Reservoir Model to Gas Production Data*. M.S. thesis, University of Tulsa, Tulsa, Oklahoma, 2000.

B. L. N. Kennett and P. R. Williamson. Subspace methods for large-scale nonlinear inversion. In *Mathematical Geophysics*, pages 139–154. D. Reidel, 1988.

J. E. Killough, Yogeshwar Sharma, Alain Dupuy, Robert Bissell, and John Wallis. A multiple right hand side iterative solver for history matching SPE 29119. In

Proceedings of the 13th SPE Symposium on Reservoir Simulation, pages 249–255, 1995.

Peter K. Kitanidis. Quasi-linear geostatistical theory for inversing. *Water Resour. Res.*, 31(10), 2411–2419, 1995.

Tamara K. Kolda, Dianne P. O'Leary, and Larry Nazareth. BFGS with update skipping and varying memory. *SIAM J. Optim*, 8(4), 1060–1083, 1998.

Law and Fariss. Transformational discrimination for unconstrained optimisation. *Ind. Eng. Chem. Funcdam*, 12(2), 1972.

Ruijian Li. *Conditioning Geostatistical Models to Three-Dimensional Three-Phase Flow Production Data by Automatic History Matching.* Ph.D. thesis, University of Tulsa, Tulsa, Oklahoma, 2001.

Ruijian Li, A. C. Reynolds, and D. S. Oliver. History matching of three-phase flow production data, SPE 66351. In *Proceedings of the 2001 SPE Reservoir Simulation Symposium*, 2001.

D. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Math. Programming*, 45(3), 503–528, 1989.

Ning Liu, Soraya Betancourt, and Dean S. Oliver. Assessment of uncertainty assessment methods. *Proceedings of the 2001 SPE Annual Technical Conference and Exhibition*, pages 1–15, 2001.

Eliana M. Makhlouf, Wen H. Chen, Mel L. Wasserman, and John H. Seinfeld. A general history matching algorithm for three-phase, three-dimensional petroleum reservoirs. *SPE Advanced Technology Series*, 1(2), 83–91, 1993.

Kiyoshi Masumoto. Pressure derivative matching method for two phase fluid flow in heterogeneous reservoir. *SPE-59462*, 2000.

J. A. Meijerink. Guidelines for the usage of incomplete decompositions in solving sets of linear equations as they occur in practical problems. *Journal of Computational Physics*, 44, 134–155, 1981.

J. A. Meijerink and H. A. van der Vorst. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Mathematics of Computation*, 31(137), 148–162, 1977.

W. Murray. *Numerical Methods for Unconstrained Optimization.* Academic Press, New York-London, 1972.

Stephen G. Nash. Preconditioning of truncated-Newton methods. *SIAM. J. Sci. Stat. Comput*, 6(3), 599–616, 1985.

Stephen G. Nash and Jorge Nocedal. A numerical study of the limited memory BFGS method and the truncated-Newton method for large scale optimization. *SIAM. J. Optm.*, 1(3), 358–372, 1991.

L. Nazareth. A conjugate gradient algorithm without line searches. *J. Optimization Theory and Application*, 23(3), 373–387, 1977.

L. Nazareth. A relationship between the BFGS and conjugate gradient algorithms and its implications for new algorithms. *SIAM. J. Numer. Anal.*, 16(5), 794–800, 1979.

Jorge Nocedal. Updating quasi-Newton matrices with limited storage. *Math. Comp.*, 35(151), 773–782, 1980.

Jorge Nocedal. Theory of algorithms for unconstrained optimization. In *Acta Numerica (1991), Cambridge University Press, London*, pages 199–244, 1992.

D. W. Oldenburg, P. R. McGillivray, and R. G. Ellis. Generalized subspace methods for large-scale inverse problems. *Geophys. J. Int.*, 114(1), 12–20, 1993.

Dean S. Oliver. Multiple realizations of the permeability field from well-test data. *Soc. Petrol. Eng. J.*, 1(2), 145–154, 1996.

Dean S. Oliver. Calculation of the inverse of the covariance. *Mathematical Geology*, 30(7), 911–933, 1998.

Dean S. Oliver, Luciane B. Cunha, and Albert C. Reynolds. Markov chain Monte Carlo methods for conditioning a permeability field to pressure data. *Mathematical Geology*, 29(1), 61–91, 1997.

Dean S. Oliver, Nanqun He, and Albert C. Reynolds. Conditioning permeability fields to pressure data. In *European Conference for the Mathematics of Oil Recovery, V*, pages 1–11, 1996.

Henning Omre, Håkon Tjelmeland, Yuanchang Qi, and Leif Hinderaker. Assessment of uncertainty in the production characteristics of a sand stone reservoir. In *Reservoir Characterization III*, pages 556–603. PennWell Books, Tulsa, OK, 1993.

S. S. Oren. Self-scaling variable metric algorithm without line-search for unconstrained minimization. *Mathematics of Computation*, 27(124), 873–885, 1973.

S. S. Oren. On the selection of parameters in self-scaling variable metric algorithms. *Mathematical Programming*, 7(3), 351–367, 1974a.

S. S. Oren. Self-scaling variable metric (SSVM) algorithms II: Implementation and experiments. *Management Science*, 20(5), 863–874, 1974b.

S. S. Oren and D.G. Luenberger. Self-scaling variable metric (SSVM) algorithms I: Criteria and sufficient conditions for scaling a class of algorithms. *Management Science*, 20(5), 845–862, 1974.

S. S. Oren and E. Spedicato. Optimal conditioning of self-scaling variable metric algorithms. *Mathematical Programming*, 10(1), 70–90, 1976.

Ahmed Ouenes, B. Bréfort, G. Meunier, and Dupéré. A new algorithm for automatic history matching: Application of simulated annealing method (SAM) to reservoir inverse modeling. *SPE 26297*, 1993.

Ahmed Ouenes, Rao S. Doddi, Yinghua Lin, George Cunningham, and Naji Saad. A new approach combining neural networks and simulated annealing for solving petroleum inverse problems. In *4th European Conference on the Mathematics of Oil Recovery*, 1994.

Robert L. Parker. *Geophysical Inverse Theory*. Princeton University Press, Princeton, New Jersey, 1994.

D. W. Peaceman. Interpretation of well-block pressures in numerical reservoir simulation with non-square grid blocks and anisotropic permeability. *Soc. Pet. Eng. J.*, 23(6), 531–543, 1983.

E. Polak. *Computational Methods in Optimization: a Unified Approach*. Academic Press, London, 1971.

M.J.D. Powell. Restart procedures for the conjugate gradient method. *Mathematical Programming*, 12(2), 241–254, 1977.

William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in FORTRAN*. Cambridge University Press, 1992.

Banda S. RamaRao, A. Marsh LaVenue, Ghislain de Marsily, and Melvin G. Marietta. Pilot point methodology for automated calibration of an ensemble of conditionally simulated transmissivity fields, 1. Theory and computational experiments. *Water Resour. Res.*, 31(3), 475–493, 1995.

Albert C. Reynolds, Nanqun He, Lifu Chu, and Dean S. Oliver. Reparameterization techniques for generating reservoir descriptions conditioned to variograms and well-test pressure data. *Soc. Petrol. Eng. J.*, 1(4), 413–426, 1996.

Albert C. Reynolds, Nanqun He, and Dean S. Oliver. Reducing uncertainty in geostatistical description with well testing pressure data. In Richard A. Schatzinger and John F. Jordan, editors, *Reservoir Characterization—Recent Advances*, pages 149–162. American Association of Petroleum Geologists, 1999.

Frédéric Roggero. Direct selection of stochastic model realizations constrained to historical data. *SPE 38731, Annual Technical Conference*, pages 155–165, 1997.

G. B. Savioli and C. A. Grattoni. On the inverse problem application to reservoir characterization. *SPE-025522*, 1992.

M. K. Sen, Akhil Datta Gupta, P. L. Stoffa, L. W. Lake, and G. A. Pope. Stochastic reservoir modeling using simulated annealing and genetic algorithm. In *Proceedings from the 67th Annual Technical Conference and Exhibition of the Society of Petroleum Engineers*, pages 939–950, October 1992.

P. C. Shah, G. R. Gavalas, and J. H. Seinfeld. Error analysis in history matching: The optimum level of parameterization. *Soc. Petrol. Eng. J.*, 18(6), 219–228, 1978.

D. F. Shanno. Conditioning of quasi-Newton method for function minimization. *Math. Comp.*, 24(111), 647–656, 1970.

D. F. Shanno. Conjugate gradient methods with inexact searches. *Mathematics of Operation Research*, 3(3), 244–256, 1978a.

D. F. Shanno. On the convergence of a new conjugate gradient algorithm. *SIAM Journal on Numerical Analysis*, 15(6), 1247–1257, 1978b.

D. F. Shanno and Kang-Hoh Phua. Matrix conditioning and nonlinear optimization. *Mathematical Programming*, 14(2), 149–160, 1978.

Jan Arild Skjervheim. *Automatic history matching.* M.S. thesis, University of Tulsa and Norweigian University of Science and Technology, Tulsa, Oklahoma, 2002.

Thomas B. Tan. A computationally efficient Gauss-Newton method for automatic history matching, SPE-29100. *Proceedings of the 13th SPE Symposium on Reservoir Simulation*, pages 61–70, 1995.

Y. N. Tang, Y. M. Chen, W. H. Chen, and Mel L. Wasserman. Generalized pulse-spectrum technique for 2-D and 2-phase history matching. *Applied Numerical Mathematics*, 5(6), 529–539, 1989.

Albert Tarantola. *Inverse Problem Theory: Methods for Data Fitting and Model Parameter Estimation*. Elsevier, Amsterdam, The Netherlands, 1987.

A. N. Tikhonov. Regularization of incorrectly posed problems. *Soviet Math. Dokl.*, 4, 1624–1627, 1963.

Richard P. Kendall Todd Dupont and H. H. Rachford. An approximation factorization procedure for solving self-adjoint elliptic difference equations. *SIAM Journal on Numerical Analysis*, 5(3), 559–573, 1968.

D. W. Vasco, A. Datta-Gupta, and J. C. S. Long. Integrating field production history in stochastic reservoir characterization, SPE-36567. In *Proccedings of the 1996 Annual Technical Conference of the SPE*, pages 829–840, 1996.

P. K. W. Vinsome. Orthomin, an iterative method for solving sparse sets of simultaneous linear equations. *SPE-5729*, 1976.

Xian-Hun Wen, Clayton V. Deutsch, and A. S. Cullick. High resolution reservoir models integrating multiple-well production data, SPE-38728. In *Proccedings of the 1997 Annual Technical Conference of the SPE*, pages 115–129, 1997.

P. Wolfe. Convergence conditions for ascent methods. *SIAM Rev.*, 11(2), 226–235, 1969.

Zhan Wu. *Conditioning Geostatistical Models to Two-Phase Flow Production Data*. Ph.D. thesis, University of Tulsa, 1999.

Zhan Wu, A. C. Reynolds, and D. S. Oliver. Conditioning geostatistical models to two-phase production data. *Soc. Petrol. Eng. J.*, 3(2), 142–155, 1999.

Guoping Xue and Akhil Datta-Gupta. Structure preserving inversion: An efficient approach to conditioning stochastic reservoir models to dynamic data. *1997 SPE Annual Technical Conference*, SPE 35412, 1997.

Pin-Huel Yang and A. Ted Watson. Automatic history matching with variable-metric methods. *SPE Reservoir Engineering*, 3(3), 995–1001, 1988.

William W-G Yeh. Review of parameter identification in groundwater hydrology: The inverse problem. *Water Resour. Res.*, 22(2), 95–108, 1986.

F. Zhang, A. C. Reynolds, and D. S. Oliver. Optimization algorithms for history matching. In *TUPREP Research Report 18*, pages 144–171. The University of Tulsa, 2001.

F. Zhang, A. C. Reynolds, and D. S. Oliver. Evaluation of the reduction in uncertainty obtained by conditioning a 3D stochastic channel to multiwell pressure data. *Mathematical Geology*, 34(6), 715–742, 2002.

APPENDIX A

THEORY OF LINEAR EQUATION SOLVERS

In this appendix, we summarize some theoretical results and algorithms for iterative solvers. Some material presented here is based on material that can be found in Greenbaum (1997). From the derivations, we can understand the relationship between the different methods. We provide some description of the computational efforts required by a direct solver in the first section for the purpose of comparison and to provide a more complete overview of linear solvers. All other sections of this appendix are focused on the iterative solvers.

A.1 Direct Solvers

Direct methods include Gauss-Jordan elimination, Gaussian elimination, LU decomposition and nested factorization often in conjunction with special procedures for sparse matrices. Here, we only compare the arithmetic operations (additions/subtractions and multiplications) used by different methods for the purpose of comparing the computational efficiency; see Press et al. (1992) for details. Assume N is the number of equations and M is the number of right hand sides. Gauss-Jordan elimination requires $\frac{1}{2}N^3 + N^2M$ additions/subtractions and the same number of multiplications. Gaussian elimination requires $\frac{1}{3}N^3 + N^2M$ additions/subtractions and the same number of multiplications. (For both methods, divisions by pivot elements are not counted.) For $M \ll N$ (a few right hand sides), Gaussian elimination is faster than Gauss-Jordan elimination. One disadvantage of both elimination methods is that the right hand sides must be known in advance to perform the elimination assuming we do not wish to store all the multipliers used in elimination. The LU decomposition, which is another direct solver, does not depend on the right hand

sides. The LU decomposition requires about $\frac{1}{3}N^3$ additions and $\frac{1}{3}N^3$ multiplications. To solve one linear equation system with one right hand side, LU decomposition requires $\frac{1}{3}N^3 + N^2$ additions and the same number of multiplications. If we have N right hand sides, LU decomposition requires $\frac{1}{3}N^3 + N \times N^2$ additions and the same number of multiplications.

A.2 Iterative Solvers

Iterative solvers that were frequently used in the past, e.g., Jacobi, Gauss-Seidel and SOR, can be treated as preconditioned iterative solvers. Assume the system of linear equations to be solved is given by

$$Ax = b, \tag{A–1}$$

where A is an $N \times N$ nonsingular matrix and x and b are N-dimensional column vectors. Let M denote a preconditioning matrix. Since a preconditioner is designed so that $M^{-1}A$ in some sense approximates the identity, $M^{-1}(b - Ax_k)$ can be expected to approximate the error $A^{-1}b - x_k$ in the approximate solution x_k. Intuitively, we expect a better solution can be obtained using the following formula:

$$x_{k+1} = x_k + M^{-1}(b - Ax_k). \tag{A–2}$$

Note that if $M^{-1} = A^{-1}$, then Eq. A–2 becomes

$$\begin{aligned} x_{k+1} &= x_k + A^{-1}(b - Ax_k) \\ &= x_k + A^{-1}b - x_k \\ &= A^{-1}b, \end{aligned} \tag{A–3}$$

which is the exact solution of Eq. A–1. Different choices of M in Eq. A–2 gives different iterative procedures. Let

$$A = D - L - U, \tag{A–4}$$

where D is a diagonal matrix with its diagonal entries equal to the diagonal entries of A, L is strictly lower triangular and U is strictly upper triangular. M equal to D

(the diagonal of A) gives the Jacobi method. If M is equal to $D - L$, Eq. A–2 gives the Gauss-Seidel procedure, i.e.

$$x_{k+1} = x_k + (D - L)^{-1}(b - Ax_k). \tag{A–5}$$

Left multiplying by $(D - L)$ on both sides of Eq. A–5, we find

$$\begin{aligned}(D - L)x_{k+1} &= (D - L)x_k + (b - Ax_k) \\ &= (D - L)x_k + b - (D - L - U)x_k \\ &= Ux_k + b.\end{aligned} \tag{A–6}$$

Rearranging Eq. A–6 gives

$$x_{k+1} = (D - L)^{-1}Ux_k + (D - L)^{-1}b, \tag{A–7}$$

which is equivalent to Eq. A–5. For $M = \omega^{-1}D - L$, where ω is a relaxation parameter, the resulting iterative method obtained from Eq. A–2 is called the successive over relaxation or SOR method.

Define the error vector at the kth iteration as

$$e_k \equiv A^{-1}b - x_k. \tag{A–8}$$

Applying Eq. A–2, Eq. A–8 can be rewritten as

$$\begin{aligned}e_k &= A^{-1}b - x_{k-1} - M^{-1}(b - Ax_{k-1}) \\ &= e_{k-1} - M^{-1}(AA^{-1}b - Ax_{k-1}) \\ &= e_{k-1} - M^{-1}A(A^{-1}b - x_{k-1}) \\ &= e_{k-1} - M^{-1}Ae_{k-1} \\ &= (I - M^{-1}A)e_{k-1}.\end{aligned} \tag{A–9}$$

The following theorem is given in Greenbaum (1997).

THEOREM: The iteration scheme of Eq. A–2 converges to $A^{-1}b$ for every initial error e_0 if and only if $\rho(I - M^{-1}A) < 1$ where $\rho(A) = \max\{|\lambda| : \lambda \text{ is an eigenvalue of A}\}$ is the spectral radius of the matrix A.

By introducing dynamically computed parameters a_k into the iterative scheme of Eq. A–2 and assuming b and A are preconditioned already (which implies $M^{-1}b$

can be treated as the new right hand side vector and $M^{-1}A$ can be treated as the new coefficient matrix so we do not need to include M^{-1}), Eq. A–2 can be rewritten as

$$x_{k+1} = x_k + a_k(b - Ax_k). \tag{A–10}$$

The residual at the kth iteration is defined by

$$r_k = b - Ax_k. \tag{A–11}$$

So Eq. A–10 can be rewritten as

$$x_{k+1} = x_k + a_k r_k \tag{A–12}$$

where r_k can be considered to be a search direction. From Eqs. A–10 and A–11, we have

$$\begin{aligned} r_{k+1} &= b - Ax_{k+1} \\ &= b - Ax_k - a_k A(b - Ax_k) \\ &= r_k - a_k A r_k, \end{aligned} \tag{A–13}$$

which gives the relationship between the new residual r_{k+1} and the previous residual r_k. Using Eqs. A–8 and A–12, it follows that

$$\begin{aligned} e_{k+1} &= A^{-1}b - x_{k+1} \\ &= A^{-1}b - x_k - a_k r_k \\ &= e_k - a_k r_k, \end{aligned} \tag{A–14}$$

which gives a relationship between the new error vector e_{k+1}, the old error vector e_k and the old residual r_k. Left multiplying by A on both side of Eq. A–8, we find that

$$\begin{aligned} Ae_k &= AA^{-1}b - Ax_k \\ &= b - Ax_k \\ &= r_k, \end{aligned} \tag{A–15}$$

which relates the error vector e_k to the residual r_k. Using Eq. A–15 in Eq. A–14, we obtain

$$Ae_k = e_k - a_k Ae_k \tag{A–16}$$

which gives a relationship between the new error vector e_{k+1} and the old error vector e_k.

The coefficient a_k in Eq. A–12 can be determined either by minimizing the residual or by minimizing the error vector. The consequence of these choices are discussed later.

A.2.1 Orthomin (1)

In this method, a_k in Eq. A–12 is obtained by minimizing the l_2 norm of the residual. The square of the l_2 norm of r_{k+1} is given by

$$\begin{aligned}(r_{k+1}, r_{k+1}) &= (r_k - a_k A r_k, r_k - a_k A r_k) \\ &= (r_k, r_k) - a_k(A r_k, r_k) - a_k(r_k, A r_k) + a_k^2(A r_k, A r_k) \\ &= (r_k, r_k) - 2a_k(A r_k, r_k)) + a_k^2(A r_k, A r_k).\end{aligned} \tag{A–17}$$

Here and throughout $(\cdot, \cdot)$ denotes the l_2 inner product. Note that in Eq. A–17

$$(A r_k, r_k) = r_k^T A r_k = (A r_k)^T r_k = (r_k, A r_k), \tag{A–18}$$

i.e., we have not assumed that A is a symmetric matrix. To minimize the l_2 norm of the residual, we set the derivative of Eq. A–17 with respect to a_k equal to zero. This gives

$$a_k = \frac{(r_k, A r_k)}{(A r_k, A r_k)}. \tag{A–19}$$

This a_k minimizes (r_{k+1}, r_{k+1}) if and only if the second derivative of (r_{k+1}, r_{k+1}) with respect to a_k , which is given by $2(A r_k, A r_k) = 2 r_k^T A^T A r_k = 2\|A r_k\|^2$, is positive. This is true if the matrix A is nonsingular and $r_k \neq 0$. A is assumed to be nonsingular, so Eq. A–1 has a unique solution. Therefore, $r_k^T A^T A r_k$ is positive if r_k is not equal to zero, but $r_k = 0$ means $A x_k = b$ so x_k is the desired solution. Eq. A–12 with a_k given by Eq. A–19 is called the orthomin (1) method. In this method, A can be any nonsingular square matrix.

A.2.2 Steepest Descent

Note that the coefficient a_k in orthomin (1) is obtained by minimizing the l_2 norm of the residual. The coefficient a_k can also be obtained by minimizing the

$A-$norm of the error vector which is defined by

$$\|e_{k+1}\|_{\mathrm{A}}^2 = (e_{k+1}, Ae_{k+1}). \tag{A–20}$$

We now assume that A is a real symmetric matrix, then using Eq. A–14, Eq. A–20 can be written as

$$\begin{aligned} \|e_{k+1}\|_{\mathrm{A}}^2 &= (e_{k+1}, Ae_{k+1}) \\ &= (e_k - a_k r_k, A(e_k - a_k r_k)) \\ &= (e_k, Ae_k) - a_k(e_k, Ar_k) - a_k(r_k, Ae_k) + a_k^2(r_k, Ar_k) \\ &= (e_k, Ae_k) - 2a_k(e_k, Ar_k) + a_k^2(r_k, Ar_k), \end{aligned} \tag{A–21}$$

By setting the first derivative of $\|e_{k+1}\|_{\mathrm{A}}^2$ with respect to a_k equal to zero, we obtain

$$a_k = \frac{(e_k, Ar_k)}{(r_k, Ar_k)} = \frac{(r_k, r_k)}{(r_k, Ar_k)}. \tag{A–22}$$

The second equality in Eq. A–22 is obtained from Eq. A–15 ($Ae_k = r_k$) and the fact that A is real symmetric, which implies $(e_k, Ar_k) = (Ae_k, r_k) = (r_k, r_k)$. To determine whether a_k corresponds to a maximum, minimum or saddle point, we need the second derivative of $\|e_{k+1}\|_{\mathrm{A}}^2$ which is given by

$$\frac{d^2(\|e_{k+1}\|_{\mathrm{A}}^2)}{da_k^2} = 2(r_k, Ar_k) = 2r_k^T Ar_k. \tag{A–23}$$

To guarantee that a_k given by Eq. A–22 minimizes $\|e_{k+1}\|_{\mathrm{A}}^2$, the second derivative of $\|e_{k+1}\|_{\mathrm{A}}^2$ given in Eq. A–23 must be positive. This is guaranteed if the matrix A is positive definite and r_k is nonzero. Eq. A–12 with a_k given by Eq. A–22 is called steepest descent. Note that applying this method requires the matrix A to be real symmetric positive definite, whereas, the matrix A in orthomin (1) can be any real nonsingular square matrix.

Eq. A–12 can be written in the more general form

$$x_{k+1} = x_k + a_k p_k, \tag{A–24}$$

where p_k is the search direction at the $(k+1)$st iteration which can be formed in many different ways. In orthomin (1) and steepest descent, the search direction p_k

is equal to the residual r_k. The residual and error vectors based on this new form can be rewritten as

$$\begin{aligned} r_{k+1} &= b - Ax_{k+1} \\ &= b - Ax_k - a_k Ap_k \\ &= r_k - a_k Ap_k, \end{aligned} \tag{A–25}$$

and

$$\begin{aligned} e_{k+1} &= A^{-1}b - x_{k+1} \\ &= A^{-1}b - x_k - a_k p_k \\ &= e_k - a_k p_k. \end{aligned} \tag{A–26}$$

Note that the relation

$$\begin{aligned} Ae_{k+1} &= A(A^{-1}b - x_{k+1}) \\ &= A\big(A^{-1}(b - Ax_{k+1})\big) \\ &= AA^{-1}r_{k+1} \\ &= r_{k+1}, \end{aligned} \tag{A–27}$$

always holds.

If we choose p_k as a linear combination of r_k and p_{k-1}, i.e.,

$$p_k = r_k + b_k p_{k-1}, \tag{A–28}$$

then we have two parameters, a_k and b_k, to adjust. Different choices for a_k in Eq. A–24 and b_k in Eq. A–28 result in different algorithms. Reasonable ways to choose a_k and b_k are presented below.

A.2.3 Orthomin (2)

We can effectively determine b_k by requiring the following orthogonality relation holds:

$$(Ap_k, Ap_{k-1}) = 0. \tag{A–29}$$

Using Eq. A–28 in Eq. A–29, we find

$$\begin{aligned}(Ap_k, Ap_{k-1}) &= (Ar_k + b_k Ap_{k-1}, Ap_{k-1}) \\ &= (Ar_k, Ap_{k-1}) + b_k(Ap_{k-1}, Ap_{k-1}) \\ &= 0.\end{aligned} \tag{A–30}$$

Thus,

$$b_k = -\frac{(Ar_k, Ap_{k-1})}{(Ap_{k-1}, Ap_{k-1})}. \tag{A–31}$$

So Eq. A–28 can be rewritten as

$$p_k = r_k - \frac{(Ar_k, Ap_{k-1})}{(Ap_{k-1}, Ap_{k-1})} p_{k-1}. \tag{A–32}$$

As in orthomin (1), we minimize the l_2 norm of the residual, i.e., we minimize (r_{k+1}, r_{k+1}), to determine the coefficient a_k in Eq. A–24. This procedure gives

$$a_k = \frac{(r_k, Ap_k)}{(Ap_k, Ap_k)}. \tag{A–33}$$

The formula for a_k can also be obtained by setting $(r_{k+1}, Ap_k) = 0$. Eqs. A–24, A–32 and A–33 represent the orthomin (2) method. As in orthomin (1), the matrix A can be any real nonsingular square matrix.

A.2.4 Conjugate Gradient

Instead of choosing b_k so that Eq. A–29 holds, we can choose b_k in Eq. A–28 so that the following orthogonality relation holds:

$$(p_k, Ap_{k-1}) = 0, \tag{A–34}$$

i.e., the search directions p_k and p_{k-1} are $A-$orthogonal. Using Eq. A–28 in Eq. A–34 gives

$$\begin{aligned}(p_k, Ap_{k-1}) &= (r_k + b_k p_{k-1}, Ap_{k-1}) \\ &= (r_k, Ap_{k-1}) + b_k(p_{k-1}, Ap_{k-1}) \\ &= 0.\end{aligned} \tag{A–35}$$

Thus,

$$b_k = -\frac{(r_k, Ap_{k-1})}{(p_{k-1}, Ap_{k-1})}. \tag{A–36}$$

With this b_k, Eq. A–28 can be rewritten as

$$p_k = r_k - \frac{(r_k, Ap_{k-1})}{(p_{k-1}, Ap_{k-1})} p_{k-1}. \tag{A–37}$$

As in steepest descent, the coefficient a_k in Eq. A–24 is chosen by minimizing

$$\begin{aligned} \| e_{k+1} \|_A^2 &= (Ae_{k+1}, e_{k+1}) \\ &= (Ae_k - a_k Ap_k, e_k - a_k p_k) \\ &= (Ae_k, e_k) - 2a_k(Ae_k, p_k) + a_k^2(p_k, Ap_k). \end{aligned} \tag{A–38}$$

Note that we assumed A is real symmetric in the above equation. Setting the derivative of $\| e_{k+1} \|_A^2$ with respect to a_k be zero, we obtain

$$a_k = \frac{(Ae_k, p_k)}{(p_k, Ap_k)} = \frac{(r_k, p_k)}{(p_k, Ap_k)}. \tag{A–39}$$

The second equality in Eq. A–39 was obtained by using Eq. A–27. As in steepest descent, to guarantee that a_k given by Eq. A–39 minimizes $\|e_{k+1}\|_A$, we assume A is real symmetric positive definite. Eq. A–39 can also be obtained by setting $(e_{k+1}, Ap_k) = 0$ given that A is real symmetric. Eq. A–24 with Eq. A–37 and Eq. A–39 is called the conjugate gradient method.

Using Eq. A–28, the inner product (r_k, p_k) can be written as

$$\begin{aligned} (r_k, p_k) &= (r_k, r_k + b_k p_{k-1}) \\ &= (r_k, r_k) + b_k(r_k, p_{k-1}). \end{aligned} \tag{A–40}$$

Using Eqs. A–25 and A–39, we obtain

$$\begin{aligned} (r_k, p_{k-1}) &= (r_{k-1} - a_{k-1} Ap_{k-1}, p_{k-1}) \\ &= (r_{k-1}, p_{k-1}) - a_{k-1}(Ap_{k-1}, p_{k-1}) \\ &= (r_{k-1}, p_{k-1}) - (r_{k-1}, p_{k-1}) \\ &= 0, \end{aligned} \tag{A–41}$$

and using this equation in Eq. A–40, it follows that

$$(r_k, p_k) = (r_k, r_k). \tag{A–42}$$

Using Eq. A–42 in Eq. A–39 gives

$$a_k = \frac{(r_k, p_k)}{(p_k, Ap_k)} = \frac{(r_k, r_k)}{(p_k, Ap_k)}. \tag{A–43}$$

Using Eqs. A–25, A–28, A–34 and A–43, we can show that $(r_k, r_{k-1}) = 0$ as follows:

$$\begin{aligned}
(r_k, r_{k-1}) &= (r_{k-1} - a_{k-1}Ap_{k-1}, r_{k-1}) \\
&= (r_{k-1}, r_{k-1}) - a_{k-1}(Ap_{k-1}, r_{k-1}) \\
&= (r_{k-1}, r_{k-1}) - a_{k-1}(Ap_{k-1}, p_{k-1} - b_{k-1}p_{k-2}) \\
&= (r_{k-1}, r_{k-1}) - a_{k-1}(Ap_{k-1}, p_{k-1}) \\
&= (r_{k-1}, r_{k-1}) - \frac{(r_{k-1}, r_{k-1})}{(Ap_{k-1}, p_{k-1})}(Ap_{k-1}, p_{k-1}) \\
&= 0.
\end{aligned} \tag{A–44}$$

Using this result and Eq. A–25, we see that

$$\begin{aligned}
(r_k, Ap_{k-1}) &= (r_k, \frac{1}{a_{k-1}}(r_{k-1} - r_k)) \\
&= \frac{1}{a_{k-1}}(r_k, r_{k-1}) - \frac{1}{a_{k-1}}(r_k, r_k) \\
&= -\frac{1}{a_{k-1}}(r_k, r_k).
\end{aligned} \tag{A–45}$$

Using this result and Eqs. A–36 and A–43, b_k can be written as

$$\begin{aligned}
b_k &= -\frac{(r_k, Ap_{k-1})}{(p_{k-1}, Ap_{k-1})} \\
&= -\frac{-\frac{1}{a_{k-1}}(r_k, r_k)}{(p_{k-1}, Ap_{k-1})} \\
&= -\frac{-\frac{1}{a_{k-1}}(r_k, r_k)}{\frac{1}{a_{k-1}}(r_{k-1}, r_{k-1})} \\
&= \frac{(r_k, r_k)}{(r_{k-1}, r_{k-1})},
\end{aligned} \tag{A–46}$$

so

$$b_k = -\frac{(r_k, Ap_{k-1})}{(p_{k-1}, Ap_{k-1})} = \frac{(r_k, r_k)}{(r_{k-1}, r_{k-1})}. \tag{A–47}$$

The standard way to apply the conjugate gradient algorithm is to use the second equality in Eq. A–43 to calculate a_k and use the second equality in Eq. A–47 to calculate b_k. This avoids calculation of (r_k, p_k) and (r_k, Ap_{k-1}).

A.2.5 Orthomin

To obtain a more general form of orthomin, we require the current search direction be constructed as a linear combination of all the previous search direction vectors and the current shift vector $\delta x_{k+1} = x_{k+1} - x_k$, i.e.,

$$p_{k+1} = \delta x_{k+1} - \sum_{j=1}^{k} b_j p_j. \tag{A–48}$$

We require that the p_j's satisfy the following orthogonality relation:

$$(Ap_j, Ap_i) = 0, \qquad i \neq j. \tag{A–49}$$

Thus, we wish p_{k+1} to be such that

$$(Ap_{k+1}, Ap_i) = 0, \qquad i = 1, 2, \cdots, k. \tag{A–50}$$

From Eq. A–48, we see that Eq. A–50 holds if and only if

$$b_i = \frac{(A\delta x_{k+1}, Ap_i)}{(Ap_i, Ap_i)}, \qquad i = 1, 2, \cdots, k. \tag{A–51}$$

Now we apply the following iterative procedure to compute the new updated approximation of x,

$$x_{k+1} = x_k + a_k p_k, \tag{A–52}$$

where a_k is chosen by minimizing the l_2 norm of the residual. Similar to Eq. A–33, we obtain

$$a_k = \frac{(r_k, Ap_k)}{(Ap_k, Ap_k)}. \tag{A–53}$$

Since this gives the minimum of $f(a_k) = \|r_{k+1}\|^2 = (r_{k+1}, r_{k+1})$ where

$$r_{k+1} = r_k - a_k A p_k, \tag{A–54}$$

and $f(0) = (r_k, r_k) = \|r_k\|^2$, we always have

$$\|r_{k+1}\|^2 \leq \|r_k\|^2, \tag{A–55}$$

which implies that the method can not "diverge". Define

$$\delta r_k = r_{k+1} - r_k. \tag{A–56}$$

Applying Eq. A–54, Eq. A–56 can be written as

$$\delta r_k = -a_k A p_k. \tag{A–57}$$

Applying Eqs. A–54 and A–53, we see that

$$\begin{aligned}(r_{k+1}, \delta r_k) &= (r_k - a_k A p_k, -a_k A p_k) \\ &= (r_k, -a_k A p_k) + a_k^2 (A p_k, A p_k) \\ &= a_k \big[(r_k, -A p_k) + a_k (A p_k, A p_k) \big] \\ &= a_k \Big[-(r_k, A p_k) + \frac{(r_k, A p_k)}{(A p_k, A p_k)} (A p_k, A p_k) \Big] \\ &= 0.\end{aligned} \tag{A–58}$$

This equation indicates that r_{k+1} is orthogonal to δr_k. Fig. A–1 illustrates the relationship between r_{k+1} and r_k. As $\|r_k\|$ is the "length" of the hypothenuse of a right triangle and $\|r_{k+1}\|$ is the length of one of its sides we see that $\|r_{k+1}\|$ is strictly less than $\|r_k\|$ unless $r_{k+1} = \pm r_k$. We would say that the length measured in a certain norm (e.g., l_2 norm) of the new residual is always less than the length of the old residual.

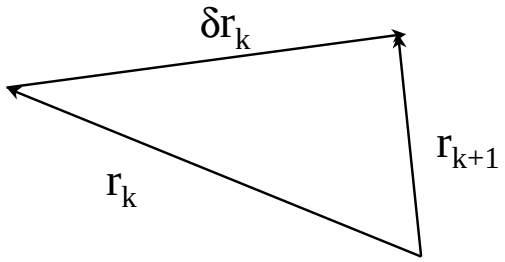


Fig. A–1: Illustration of the relationship of the two adjacent residuals in orthomin method.

Eqs. A–48, A–51, A–52 and A–53, represent the more general orthomin algorithm. The overall algorithm for the more general orthomin method for solving $Ax = b$ is given below; see Vinsome (1976) for details.

- ⋆ Choose an initial guess x_0.

- ⋆ Set $r_0 = b - Ax_0$.

- ⋆ Solve $M\delta x_1 = r_0$ for δx_1 where M is an approximation to matrix A which is chosen such that this linear equation can be solved easily and set $p_1 = \delta x_1$ and calculate

$$a_1 = \frac{(r_0, Ap_1)}{(Ap_1, Ap_1)} \tag{A–59}$$

$$x_1 = x_0 + a_1 p_1 \tag{A–60}$$

$$r_1 = r_0 - a_1 Ap_1 \tag{A–61}$$

- ⋆ Iteration loop

  DO $k = 1, 2, \cdots$

$$\delta x_{k+1} = M^{-1} r_k \tag{A–62}$$

$$b_j = -\frac{(A\delta x_{k+1}, Ap_j)}{(Ap_j, Ap_j)}, \quad j = 1, 2, \cdots, k \tag{A–63}$$

$$p_{k+1} = \delta x_{k+1} + \sum_{j=1}^{k} b_j p_j \tag{A–64}$$

$$a_{k+1} = \frac{(r_k, Ap_{k+1})}{(Ap_{k+1}, Ap_{k+1})} \tag{A–65}$$

$$x_{k+1} = x_k + a_{k+1} p_{k+1} \tag{A–66}$$

$$r_{k+1} = r_k - a_{k+1} Ap_{k+1} \tag{A–67}$$

  END DO

We call this algorithm orthomin. The orthomin algorithm was stopped when the following condition is satisfied

$$\frac{\| r_k \|_\infty}{\| r_0 \|_\infty} \leq \varepsilon, \tag{A–68}$$

where $\| \cdot \|_\infty$ denotes the infinity or maximum norm.

Note that in Eq. A–64, all the previous search direction vectors are used to construct the current search direction. If we just use a limited number of previous

search direction vectors to construct p_{k+1}, we call the corresponding version of algorithm the truncated orthmin method. If we use l previous vectors, then the only change is that Eq. A–64 is replaced by

$$p_{k+1} = \delta x_{k+1} + \sum_{j=k+1-l}^{k} b_j p_j. \tag{A–69}$$

If $l = 1$ and M is the identity matrix, then the general algorithm reduces to orthomin (2) discussed previously. In Eq. A–62, the value of δx_{k+1} is obtained by solving

$$M\delta x_{k+1} = r_k, \tag{A–70}$$

instead of forming the matrix product $M^{-1}r_k$. Recall that M can be considered to be a preconditioning matrix. Note that the first term in Eq. A–64, i.e., $\delta x_{k+1} = M^{-1}r_k$, is equivalent to the second term in Eq. A–2. The preconditioner should be chosen such that $M \approx A$, so that

$$M^{-1}A \approx I, \tag{A–71}$$

where I is the appropriate identity matrix. Combining Eq. A–71 with Eq. A–70, we have

$$A\delta x_{k+1} \approx r_{k+1}, \tag{A–72}$$

which represents the relationship between the shift vector δx_{k+1} and the residual r_{k+1} (analogue to Eq. A–27). The key issue for implementation of the general orthomin algorithm is how to choose M such that Eq. A–70 can be solved very efficiently. In our implementation, the incomplete LU decomposition was applied; see Chapter IV.

APPENDIX B

QUASI-NEWTON METHODS

In this appendix, we give the derivation of the updating equation for the inverse Hessian approximation and the principle of the self-scaling variable metric. Scaling can have a significant effect on the convergence rate of a variable metric (quasi-Newton) algorithm. Proofs of the propositions, theorems and corollaries presented here can be found in Oren and Luenberger (1974), Oren (1974b), Oren (1974a) or Oren and Spedicato (1976). Throughout we assume real vectors and matrices and all vectors being n-dimensional.

The iteration equation of Newton's method for minimizing the objective function $f(x)$ can be written as:

$$x_{k+1} = x_k - \alpha_k H_k^{-1} g_k, \tag{B–1}$$

where g_k and H_k, respectively, represent the gradient and the second derivative (Hessian) of $f(x)$ evaluated at x_k. The term α_k denotes the step size in the direction d_k, where

$$d_k = -H_k^{-1} g_k. \tag{B–2}$$

Therefore, in Newton's method, we need to solve $H_k d_k = -g_k$ for d_k which may be computationally expensive for large scale problems, especially if the evaluation of the Hessian requires considerable effort. In problems of interest to us, the Hessian matrix involves the sensitivity coefficients. For the history matching problems of interest to us, the calculation of an individual sensitivity coefficient requires a significant fraction of the time required to make a reservoir simulation run regardless of the method used to compute the sensitivity coefficient. Thus, for large scale problems where the number of model parameters and the number of data are both large,

computation of all individual sensitivity coefficients is impractical. In quasi-Newton methods, calculation of the Hessian matrix is avoided; instead, we directly generate an approximation to the inverse Hessian, $\tilde{H}_k^{-1} = \tilde{H}^{-1}(x_k)$, at the kth iteration; i.e., we construct a sequence of matrices $\{\tilde{H}_k^{-1}\}$ to approximate the sequence of the inverse Hessian matrices $\{H_k^{-1}\}$. Then Eqs. B–2 and B–1, respectively, are approximated by

$$d_k = -\tilde{H}_k^{-1} g_k \tag{B–3}$$

and

$$x_{k+1} = x_k + \alpha_k d_k. \tag{B–4}$$

According to standard theory (see Murray (1972)), the approximation $\tilde{H}_k^{-1}$ is required to satisfy the conditions listed below:

1. Each $\tilde{H}_k^{-1}$ must be symmetric positive definite to guarantee the iterative equation has the descent property.

   To guarantee that the search direction $d_k = -\tilde{H}_k^{-1} g_k$ is a descent direction, the condition $g_k^T d_k = -g_k^T \tilde{H}_k^{-1} g_k < 0$ must be satisfied. This is equivalent to $g_k^T \tilde{H}_k^{-1} g_k > 0$ which holds if $\tilde{H}_k^{-1}$ is positive definite.

2. $\{\tilde{H}_k^{-1}\}$ must satisfy the quasi-Newton condition:

   $$\tilde{H}_{k+1}^{-1}(g_{k+1} - g_k) = x_{k+1} - x_k. \tag{B–5}$$

   A motivation for quasi-Newton condition is given later.

3. One should be able to calculate $\tilde{H}_{k+1}^{-1}$ from $\tilde{H}_k^{-1}$ by a simple calculational formula such as

   $$\tilde{H}_{k+1}^{-1} = \tilde{H}_k^{-1} + E_k, \tag{B–6}$$

   where E_k, called the correction matrix, can be easily calculated.

We now provide a motivation for the quasi-Newton condition. Assume the objective function $f(x)$ is twice continuously differentiable. Applying a Taylor expansion, we obtain

$$g_k \approx g_{k+1} + H_{k+1}(x_k - x_{k+1}). \tag{B–7}$$

If H_{k+1} is nonsingular, it follows that

$$H_{k+1}^{-1}(g_{k+1} - g_k) \approx x_{k+1} - x_k. \tag{B–8}$$

For quadratic functions, the preceding two equations are exact, so

$$H_{k+1}^{-1}(g_{k+1} - g_k) = x_{k+1} - x_k. \tag{B–9}$$

For non-quadratic functions, it is desirable to require that $\tilde{H}_{k+1}^{-1}$ satisfies Eq. B–9, i.e.,

$$\tilde{H}_{k+1}^{-1}(g_{k+1} - g_k) = x_{k+1} - x_k. \tag{B–10}$$

We define

$$y_k = g_{k+1} - g_k, \tag{B–11}$$

and

$$s_k = x_{k+1} - x_k. \tag{B–12}$$

Then, the quasi-Newton condition, Eq. B–10, can be written as

$$\tilde{H}_{k+1}^{-1} y_k = s_k, \tag{B–13}$$

which is another form of the quasi-Newton condition.

There are many update equations that satisfy the three conditions enumerated above. Here, we give the derivation of a general equation from which all others can be obtained. Methods obtained from this equation are referred to as Huang family of equations (see Huang (1970)). In Huang's original equation, the inverse Hessian approximation $\tilde{H}^{-1}$ was allowed to be nonsymmetric. However, the Hessian matrices that occur in our history matching problems are always symmetric. Thus, we present the Huang equation only for the cases where all the Hessian matrices are symmetric. Part of the derivation assumes a quadratic objective function. From the series $\tilde{H}_0^{-1}, \tilde{H}_1^{-1}, \ldots, \tilde{H}_{k+1}^{-1}$ we can determine the set of search directions

$$d_j = -\tilde{H}_j^{-1} g_j, \quad j = 0, 1, \ldots, k+1. \tag{B–14}$$

If we want the search directions to be H-conjugate in the quadratic case, where H is the Hessian matrix, so that the algorithm has the property of quadratic termination

(Fletcher (1987)), the following equation must hold:

$$d_{k+1}^T H d_j = 0, \quad j = 0, 1, \ldots, k. \tag{B–15}$$

Substituting Eq. B–14 into Eq. B–15 gives

$$g_{k+1}^T \tilde{H}_{k+1}^{-1} H d_j = 0, \quad j = 0, 1, \ldots, k. \tag{B–16}$$

It can be shown that the current gradient is orthogonal to all the previous search directions for the quadratic function case provided that the line search is exact. So

$$g_{k+1}^T d_j = 0, \quad j = 0, 1, \ldots, k. \tag{B–17}$$

We provide a proof of Eq. B–17, which is important for deriving the Huang family equations. Applying Eq. B–11, we have

$$\begin{aligned} g_{k+1}^T d_j &= g_{j+1}^T d_j + (g_{j+2}^T d_j - g_{j+1}^T d_j) + (g_{j+3}^T d_j - g_{j+2}^T d_j) +, \cdots, +(g_{k+1}^T d_j - g_k^T d_j) \\ &= g_{j+1}^T d_j + (y_{j+1} + y_{j+2} +, \cdots, + y_k)^T d_j. \end{aligned} \tag{B–18}$$

From Eqs. B–4 and B–12,

$$s_j = x_{j+1} - x_j = \alpha_j d_j. \tag{B–19}$$

For a quadratic objective function, the quasi-Newton condition (Eq. B–13) holds exactly. From Eqs. B–11, B–9, B–12 and B–19, it follows that

$$y_j = g_{j+1} - g_j = H(x_{j+1} - x_j) = H s_j = \alpha_j H d_j, \tag{B–20}$$

Substituting Eq. B–20 into Eq. B–18 gives

$$\begin{aligned} g_{k+1}^T d_j &= g_{j+1}^T d_j + (\alpha_{j+1} H d_{j+1} + \alpha_{j+2} H d_{j+2} +, \cdots, + \alpha_k H d_k)^T d_j \\ &= g_{j+1}^T d_j + \alpha_{j+1} d_{j+1}^T H d_j + \alpha_{j+2} d_{j+2}^T H d_j +, \cdots, + \alpha_k d_k^T H d_j. \end{aligned} \tag{B–21}$$

An exact line search gives

$$g_{j+1}^T d_j = 0. \tag{B–22}$$

Using Eqs. B–22 and B–15 in Eq. B–21 gives Eq. B–17. Subtracting Eq. B–17 multiplied by w, which can be any nonzero scalar, from Eq. B–16 gives

$$g_{k+1}^T(\tilde{H}_{k+1}^{-1}Hd_j - wd_j) = 0, \quad j = 0, 1, \ldots, k. \tag{B–23}$$

Eq. B–23 will be satisfied if we choose

$$\tilde{H}_{k+1}^{-1}Hd_j = wd_j, \quad j = 0, 1, \ldots, k. \tag{B–24}$$

Multiplying Eq. B–24 by α_j and applying Eqs. B–20 and B–19, we find

$$\tilde{H}_{k+1}^{-1}y_j = ws_j, \quad j = 0, 1, \ldots, k. \tag{B–25}$$

Since

$$\tilde{H}_{k+1}^{-1} = \tilde{H}_k^{-1} + E_k, \tag{B–26}$$

Eq. B–25 can be rewritten as

$$E_k y_j = ws_j - \tilde{H}_k^{-1}y_j, \quad j = 0, 1, \ldots, k. \tag{B–27}$$

From Eqs. B–25 and B–27, we obtain

$$E_k y_j = 0, \quad \text{for} \quad j = 0, 1, \ldots, k-1, \tag{B–28}$$

$$E_k y_k = ws_k - \tilde{H}_k^{-1}y_k. \tag{B–29}$$

Hence, E_k should be chosen such that Eq. B–28 and Eq. B–29 are both satisfied. Suppose

$$E_k = A_k + B_k, \tag{B–30}$$

and let

$$A_k = \xi_k c_k u_k^T, \tag{B–31}$$

$$B_k = \eta_k p_k v_k^T, \tag{B–32}$$

where ξ_k and η_k are scalars, and c_k, u_k, p_k and v_k are column vectors. Substituting $E_k = A_k + B_k$ into Eq. B–29 gives

$$A_k y_k + B_k y_k = ws_k - \tilde{H}_k^{-1}y_k. \tag{B–33}$$

Eq. B–33 will be satisfied if we choose the vectors such that

$$A_k y_k \equiv \xi_k c_k u_k^T y_k = w s_k, \tag{B–34}$$

and

$$B_k y_k \equiv \eta_k p_k v_k^T y_k = -\tilde{H}_k^{-1} y_k. \tag{B–35}$$

If we choose

$$c_k = s_k, \tag{B–36}$$

and

$$p_k = -\tilde{H}_k^{-1} y_k, \tag{B–37}$$

we obtain

$$\xi_k = \frac{w}{u_k^T y_k}, \tag{B–38}$$

and

$$\eta_k = \frac{1}{v_k^T y_k}. \tag{B–39}$$

From Eqs. B–36, B–37, B–38 and B–39, it follows that

$$E_k = A_k + B_k = w\frac{s_k u_k^T}{u_k^T y_k} - \frac{\tilde{H}_k^{-1} y_k v_k^T}{v_k^T y_k}. \tag{B–40}$$

Our derivation indicates that Eq. B–29 will be satisfied when E_k is given by Eq. B–40. If u_k and v_k are chosen to be vectors such that $u_k^T y_j = 0$ and $v_k^T y_j = 0$ for all $j = 0, 1, \ldots, k-1$, then Eq. B–28 will be satisfied. Since $H s_j = y_j$ (see Eq. B–9) and $s_l = \alpha_l d_l$ for all l (Eq. B–19),

$$\begin{aligned} s_k^T y_j &= s_k^T H s_j \\ &= \alpha_k \alpha_j d_k^T H d_j \\ &= 0, \end{aligned} \tag{B–41}$$

for $j \neq k$ and the last equality follows from the orthogonality condition of Eq. B–15. Moreover, Eqs. B–25 and B–41 can be used to show that

$$\begin{aligned} y_k^T \tilde{H}_k^{-1} y_j &= y_k^T w s_j \\ &= w s_k^T H s_j \\ &= 0, \end{aligned} \tag{B–42}$$

for $j = 0, 1, \ldots, k-1$. So if we choose u_k to be either s_k or $\tilde{H}_k^{-1} y_k$ and choose v_k to be either s_k or $\tilde{H}_k^{-1} y_k$, then Eq. B–28 will be satisfied. More generally, we can choose u_k and v_k to be a linear combination of s_k and $\tilde{H}_k^{-1} y_k$, i.e.,

$$u_k = a_1 s_k + a_2 \tilde{H}_k^{-1} y_k, \tag{B–43}$$

and

$$v_k = b_1 s_k + b_2 \tilde{H}_k^{-1} y_k, \tag{B–44}$$

where a_1, a_2, b_1 and b_2 are all scalars. Substituting u_k and v_k as given by Eqs. B–43 and B–44 into Eq. B–40, we obtain a general expression for E_k. Inserting the resulting equation for E_k into the update equation (Eq. B–26), we obtain

$$\tilde{H}_{k+1}^{-1} = \tilde{H}_k^{-1} + \omega \frac{s_k (a_1 s_k + a_2 \tilde{H}_k^{-1^T} y_k)^T}{(a_1 s_k + a_2 \tilde{H}_k^{-1^T} y_k)^T y_k} - \frac{\tilde{H}_k^{-1} y_k (b_1 s_k + b_2 \tilde{H}_k^{-1^T} y_k)^T}{(b_1 s_k + b_2 \tilde{H}_k^{-1^T} y_k)^T y_k}. \tag{B–45}$$

This equation is referred to as the Huang family equation. Choosing different values for ω, a_1, a_2, b_1, b_2, we obtain different update procedures. With $\omega = 1$, $a_1 = 1 + \beta y_k^T \tilde{H}_k^{-1} y_k$, $a_2 = -\beta s_k^T y_k$, $b_1 = \beta y_k^T \tilde{H}_k^{-1} y_k$, $b_2 = 1 - \beta s_k^T y_k$ where β is a scalar, we obtain the Broyden family equation which is given by

$$\tilde{H}_{k+1}^{-1} = \tilde{H}_k^{-1} + \frac{s_k s_k^T}{s_k^T y_k} - \frac{\tilde{H}_k^{-1} y_k y_k^T \tilde{H}_k^{-1}}{y_k^T \tilde{H}_k^{-1} y_k} + \beta (y_k^T s_k)(y_k^T \tilde{H}_k^{-1} y_k) w_k w_k^T, \tag{B–46}$$

where

$$w_k = \frac{s_k}{y_k^T s_k} - \frac{\tilde{H}_k^{-1} y_k}{y_k^T \tilde{H}_k^{-1} y_k}. \tag{B–47}$$

Eq. B–46 is also called Broyden-β class update equation. Taking $\omega = 1$, $a_1 = b_2 = 1$, $b_1 = a_2 = 0$ we obtain the DFP update equation given by

$$\tilde{H}_{k+1}^{-1} = \tilde{H}_k^{-1} + \frac{s_k s_k^T}{s_k^T y_k} - \frac{\tilde{H}_k^{-1} y_k y_k^T \tilde{H}_k^{-1}}{y_k^T \tilde{H}_k^{-1} y_k}. \tag{B–48}$$

Taking $\omega = 1$, $a_1 = 1 + (y_k^T \tilde{H}_k^{-1} y_k)/(s_k^T y_k)$, $a_2 = -1$, $b_1 = (y_k^T \tilde{H}_k^{-1} y_k)/(s_k^T y_k)$, $b_2 = 0$ we get the BFGS update equation given by

$$\tilde{H}_{k+1}^{-1} = \tilde{H}_k^{-1} + \frac{s_k s_k^T}{s_k^T y_k} - \frac{\tilde{H}_k^{-1} y_k y_k^T \tilde{H}_k^{-1}}{y_k^T \tilde{H}_k^{-1} y_k} + v_k v_k^T, \tag{B–49}$$

where

$$v_k = (y_k^T \tilde{H}_k^{-1} y_k)^{1/2} \Big(\frac{s_k}{s_k^T y_k} - \frac{\tilde{H}_k^{-1} y_k}{y_k^T \tilde{H}_k^{-1} y_k} \Big) = (y_k^T \tilde{H}_k^{-1} y_k)^{1/2} w_k. \tag{B–50}$$

Note from Eqs. B–48 and B–49 it is clear that if $\tilde{H}_k^{-1}$ is real symmetric then $\tilde{H}_{k+1}^{-1}$ is also real symmetric. Also note that both BFGS and DFP are members of the Broyden family. In the DFP algorithm, round off errors or inaccurate line search may cause the inverse Hessian approximation to become singular. The BFGS algorithm is more numerically stable than the DFP algorithm; see Murray (1972). Among the Broyden family, the BFGS algorithm appears to work best in practice based on the numerical stability; see Shanno and Phua (1978)and Kolda et al. (1998). Thus, the BFGS algorithm is chosen as the minimization algorithm in our history matching procedure.

We can show that once $\tilde{H}_k^{-1}$ becomes singular, the correct solution of the minimization problem may be unattainable during the subsequent iterations. All the Broyden family equations can be written as (see, Murray (1972))

$$\tilde{H}_{k+1}^{-1} = \tilde{H}_k^{-1} M_k, \tag{B–51}$$

where M_k is a matrix specific to a particular update. Thus, by induction,

$$\tilde{H}_{k+r}^{-1} = \tilde{H}_k^{-1} M_k M_{k+1} \cdots M_{k+r-1}, \qquad \text{for all} \quad r \geq 1. \tag{B–52}$$

Since

$$s_{k+r} = -\alpha_{k+r} \tilde{H}_{k+r}^{-1} g_{k+r}, \tag{B–53}$$

we can write

$$s_{k+r} = -\tilde{H}_k^{-1} u, \qquad r \geq 1 \tag{B–54}$$

where

$$u = \alpha_{k+r} M_k M_{k+1} \cdots M_{k+r-1} g_{k+r}. \tag{B–55}$$

Suppose $\tilde{H}_k^{-1}$ is singular, so that for some nonzero vector q, we have $q^T \tilde{H}_k^{-1} = 0$. From Eq. B–54, we have $q^T s_{k+r} = 0$ for all $r \geq 1$. Therefore, once a particular $\tilde{H}_k^{-1}$ becomes singular, all subsequent steps are orthogonal to some fixed vector and hence are restricted to lie in a subspace of the real n-dimensional Euclidean space. The solution will be completely unattainable subsequent to the occurrence of a singular $\tilde{H}_k^{-1}$ unless the solution also lies in this subspace (in general it will not).

If the initial Hessian inverse approximation is real symmetric positive definite and an exact line search is performed, then we can show that the BFGS update guarantees the Hessian inverse approximation at each iteration is a real symmetric positive definite matrix. To do so, we only need to show that $\tilde{H}_{k+1}^{-1}$ is positive definite if $\tilde{H}_k^{-1}$ is positive definite. In other words, we need to show $x^T \tilde{H}_{k+1}^{-1} x > 0$ given any nonzero vector x. Eq. B–49 can be rewritten as

$$\tilde{H}_{k+1}^{-1} = \bar{H}_{k+1}^{-1} + v_k v_k^T, \tag{B–56}$$

where

$$\bar{H}_{k+1}^{-1} = \tilde{H}_k^{-1} + \frac{s_k s_k^T}{s_k^T y_k} - \frac{\tilde{H}_k^{-1} y_k y_k^T \tilde{H}_k^{-1}}{y_k^T \tilde{H}_k^{-1} y_k}, \tag{B–57}$$

which is the DFP update formula given by Eq. B–48. Note that

$$x^T v_k v_k^T x_k = (x^T v_k)^2 \geq 0. \tag{B–58}$$

So if we can show $\bar{H}_{k+1}^{-1}$ is positive definite, then $\tilde{H}_{k+1}^{-1}$ is positive definite. We can write

$$\begin{aligned} x^T \bar{H}_{k+1}^{-1} x &= x^T \tilde{H}_k^{-1} x + \frac{x^T s_k s_k^T x}{s_k^T y_k} - \frac{x^T \tilde{H}_k^{-1} y_k y_k^T \tilde{H}_k^{-1} x}{y_k^T \tilde{H}_k^{-1} y_k} \\ &= \frac{x^T \tilde{H}_k^{-1} x y_k^T \tilde{H}_k^{-1} y_k - x^T \tilde{H}_k^{-1} y_k y_k^T \tilde{H}_k^{-1} x}{y_k^T \tilde{H}_k^{-1} y_k} + \frac{x^T s_k s_k^T x}{s_k^T y_k}. \end{aligned} \tag{B–59}$$

Let $p_k = \tilde{H}_k^{-1/2} x$ and $q_k = \tilde{H}_k^{-1/2} y_k$ so that Eq. B–59 can be written as

$$\begin{aligned} x^T \bar{H}_{k+1} x &= \frac{(p_k, p_k)(q_k, q_k) - (p_k, q_k)(q_k, p_k)}{(q_k, q_k)} + \frac{x^T s_k s_k^T x}{s_k^T y_k} \\ &= \frac{(p_k, p_k)(q_k, q_k) - (p_k, q_k)(q_k, p_k)}{(q_k, q_k)} + \frac{(s_k^T x)^2}{s_k^T y_k}. \end{aligned} \tag{B–60}$$

Applying Schwartz's inequality, we have

$$\mid (p_k, q_k) \mid \leq \parallel p_k \parallel \parallel q_k \parallel = (p_k, p_k)^{1/2} (q_k, q_k)^{1/2}. \tag{B–61}$$

where equality holds if and only if p_k and q_k are linearly dependent, i.e, $p_k = \lambda q_k$ for some $\lambda \neq 0$. So when p_k and q_k are linearly independent, i.e., $p_k \neq \lambda q_k$, the first term in Eq. B–60 is greater than zero and the second term is greater than or equal

to zero as long as $s_k^T y_k > 0$ which is shown below. Hence, the whole term is greater than zero which means $\tilde{H}_{k+1}^{-1}$ is positive definite for this case. When $p_k = \lambda q_k$, the first term in Eq. B–60 is equal to zero. Hence we need to show the second term in Eq. B–60 is greater than zero to guarantee that the $\tilde{H}_{k+1}^{-1}$ is positive definite. In this case, applying the fact that $p_k = \lambda q_k$, we find $x = \lambda y_k$. Thus,

$$\frac{(s_k^T x)^2}{s_k^T y_k} = \frac{\lambda^2 (s_k^T y_k)^2}{s_k^T y_k} = \lambda^2 s_k^T y_k. \tag{B–62}$$

Because $\lambda \neq 0$ we need to show that $s_k^T y_k > 0$ to complete the proof. When an exact line search is done, we have

$$s_k^T g_{k+1} = 0. \tag{B–63}$$

Using this result and Eqs. B–11, B–14 and B–19, we find that

$$\begin{aligned} s_k^T y_k &= s_k^T (g_{k+1} - g_k) \\ &= s_k^T g_{k+1} - s_k^T g_k \\ &= -s_k^T g_k \\ &= -\alpha_k d_k^T g_k \\ &= \alpha_k g_k^T \tilde{H}_k^{-1} g_k > 0. \end{aligned} \tag{B–64}$$

Note the last equality of Eq. B–64 assumes g_k is not the zero vector. But if $g_k = 0$, then x_k is the solution and there is no need to form $\tilde{H}_{k+1}^{-1}$. Thus, the exact line search guarantees that $s_k^T y_k > 0$ and hence guarantees that $\tilde{H}_{k+1}^{-1}$ is positive definite given that $\tilde{H}_k^{-1}$ is positive definite. Using Eq. B–57, the Broyden family equation (Eq. B–46) can be written as

$$\tilde{H}_{k+1}^{-1} = \bar{H}_{k+1}^{-1} + \beta (s_k^T y_k) v_k v_k^T, \tag{B–65}$$

where $\bar{H}_{k+1}^{-1}$ is given by Eq. B–57. We have already shown that $\bar{H}_{k+1}^{-1}$ is positive definite and that an exact line search guarantees $s_k^T y_k > 0$. Therefore, if we choose $\beta > 0$, all the update equations in Broyden family are such that $\tilde{H}_{k+1}^{-1}$ is positive definite provided that $\tilde{H}_k^{-1}$ is positive definite. The fact above can be summarized in the following proposition.

PROPOSITION 1: *Let* $\tilde{H}_{k+1}^{-1}$ *be defined by Eq. B–46. If* $\tilde{H}_k^{-1}$ *is positive definite,* $\beta > 0$ *and* $s_k^T y_k > 0$ *then* $\tilde{H}_{k+1}^{-1}$ *is positive definite.*

For a quadratic function with a real symmetric positive definite Hessian H, $s_k^T y_k = s_k^T H s_k > 0$ always holds. For a general objective function, $s_k^T y_k = y_k^T s_k = g_{k+1}^T s_k - g_k^T s_k > 0$, because $g_k^T s_k < 0$ holds by the descent property and if an exact line search is used then $g_{k+1}^T s_k = \alpha_k g_{k+1}^T d_k = 0$. Thus, Proposition 1 indicates that if we choose the initial inverse Hessian approximation to be any real symmetric positive definite matrix (for example the identity matrix) and choose $\beta > 0$, all the subsequent updated inverse Hessian approximations will be symmetric positive definite, which guarantees that $d_k = -\tilde{H}_k^{-1} g_k$ is a descent direction. This will always be true for quadratic functions and will be true for a non-quadratic function given that the line search is exact.

Early applications of quasi-Newton methods routinely used an exact line search arguing that this is necessary to achieve quadratic termination and is also desirable for stability. In practice, inexact line searches that satisfy side conditions such as Wolfe's conditions are substituted for exact line searches. As noted in Chapter V, one of Wolfe's conditions is given by

$$g_{k+1}^T d_k \geq \eta g_k^T d_k, \tag{B–66}$$

where $\eta < 1$. Subtracting $g_k^T d_k$ from both sides of Eq. B–66 gives

$$g_{k+1}^T d_k - g_k^T d_k \geq \eta g_k^T d_k - g_k^T d_k. \tag{B–67}$$

Using Eq. B–11 in Eq. B–67 gives

$$y_k^T d_k \geq (\eta - 1) g_k^T d_k. \tag{B–68}$$

If $\tilde{H}_k^{-1}$is positive definite, then d_k is a downhill direction and we have

$$g_k^T d_k = -g_k^T \tilde{H}_k^{-1} g_k < 0. \tag{B–69}$$

Using the fact $\eta < 1$, it follows from Eqs. B–68 and B–69 that

$$y_k^T d_k > 0, \tag{B–70}$$

which implies $s_k^T y_k > 0$ which guarantees that $\tilde{H}_{k+1}^{-1}$ is positive definite according to Proposition 1.

Next, we consider the scaling issue for quasi-Newton methods. Consider the following equation which is a subset of the Huang family

$$\tilde{H}_{k+1}^{-1} = \big(\tilde{H}_k^{-1} - \frac{\tilde{H}_k^{-1} y_k y_k^T \tilde{H}_k^{-1}}{y_k^T \tilde{H}_k^{-1} y_k} + \theta_k v_k^T v_k\big) + \frac{s_k s_k^T}{s_k^T y_k}, \tag{B–71}$$

where

$$v_k = (y_k^T \tilde{H}_k^{-1} y_k)^{1/2}\Big(\frac{s_k}{s_k^T y_k} - \frac{\tilde{H}_k^{-1} y_k}{y_k^T \tilde{H}_k^{-1} y_k}\Big). \tag{B–72}$$

Note that if we choose $\theta_k = \beta y_k^T s_k$, then Eq. B–71 is exactly the same as Eq. B–46 which is the Broyden-β class. The BFGS method corresponds to choosing $\theta_k = 1$ for all k in Eq. B–71. If we replace $\tilde{H}_k^{-1}$ by $\hat{H}_k^{-1} = \gamma_k \tilde{H}_k^{-1}$, where γ_k is a scalar, in Eq. B–71 and Eq. B–72, we will obtain the scaled version of the update equation given by

$$\tilde{H}_{k+1}^{-1} = \big(\tilde{H}_k^{-1} - \frac{\tilde{H}_k^{-1} y_k y_k^T \tilde{H}_k^{-1}}{y_k^T \tilde{H}_k^{-1} y_k} + \theta_k v_k^T v_k\big)\gamma_k + \frac{s_k s_k^T}{s_k^T y_k}, \tag{B–73}$$

where γ_k is a scaling factor which may be adjusted to try to improve the condition number of $H^{1/2}\tilde{H}_k^{-1}H^{1/2}$ and the v_k in Eq. B–73 is still given by Eq. B–72.

Before proceeding further, we record a simple algorithm for a quasi-Newton method based on an exact line search.

Algorithm 1:

Step 1 Initialization:

Provide an initial guess x_0, calculate the objective function corresponding to x_0, evaluate g_0 (the gradient of the objective function at x_0), provide an initial Hessian inverse approximation $\tilde{H}_0^{-1}$ and set the initial iteration index to k=0.

Step 2 Calculate the search direction $d_k = -\tilde{H}_k^{-1} g_k$.

Step 3 Calculate the step size α_k by an exact line search procedure.

Step 4 Update: $x_{k+1} = x_k + \alpha_k d_k$.

Step 5 Calculate the objective function based on x_{k+1} and calculate the new gradient g_{k+1}.

Step 6 Determine if the stopping criteria are satisfied or not. If satisfied, then stop; otherwise go to step 7.

Step 7 Update $\tilde{H}_k^{-1}$ to obtain $\tilde{H}_{k+1}^{-1}$ based on Eq. B–73 and go to step 2.

From Oren and Luenberger (1974), we have the following theorem.

THEOREM 1: *Let H be a real symmetric positive definite matrix. For a positive definite quadratic objective function*

$$f(x) = \frac{1}{2}(x - x^*)^T H(x - x^*) + f(x^*) \tag{B–74}$$

and for any starting point x_0, the Algorithm 1 converges to the unique x^* which minimizes f. Furthermore, for all k

$$f(x_{k+1}) - f(x^*) \leq \left[\frac{\kappa(R_k) - 1}{\kappa(R_k) + 1}\right]^2 [f(x_k) - f(x^*)] \tag{B–75}$$

where $\kappa(R_k)$ denotes the condition number of the matrix $R_k = H^{1/2}\tilde{H}_k^{-1}H^{1/2}$.

Note that this theorem implies exact line search. This theorem suggests that the rate of convergence can be improved by decreasing $\left[\frac{\kappa(R_k)-1}{\kappa(R_k)+1}\right]^2$ and therefore one should strive to make $\kappa(R_k)$ as close to unity as possible, i.e., we wish to minimize the condition number of the matrix R_k at each iteration. Furthermore, notice that steepest descent corresponds to $\tilde{H}_k^{-1} = I$ where I is an identity matrix, and in this case $R_k = H$. Thus, if $\kappa(R_k) > \kappa(H)$ for some k, then the convergence rate of quasi-Newton at these steps may be worse than steepest descent.

We introduce the notation

$$\tilde{H}_{k+1}^{-1} = \Gamma^{\theta_k}(\tilde{H}_k^{-1}, \gamma_k, s_k, y_k) = \Big(\tilde{H}_k^{-1} - \frac{\tilde{H}_k^{-1} y_k y_k^T \tilde{H}_k^{-1}}{y_k^T \tilde{H}_k^{-1} y_k} + \theta_k v_k v_k^T\Big)\gamma_k + \frac{s_k s_k^T}{s_k^T y_k}. \tag{B–76}$$

Several general theorems about how to select the scaling factors will be given below; see Oren and Luenberger (1974) and Oren (1974b) for additional details.

THEOREM 1: *Let $\tilde{H}_k^{-1}$ be the kth positive definite approximation to the inverse of the fixed positive definite matrix H and let s_k be an arbitrary nonzero vector*

in E^n and let $y_k = Hs_k$. Also let $R_k = H^{1/2}\tilde{H}_k^{-1}H^{1/2}$ and assume the eigenvalues of R_k satisfy $0 \leq \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. Then the condition

$$\lambda_1 \leq 1/\gamma_k \leq \lambda_n \tag{B–77}$$

is a sufficient condition for

$$\kappa(H^{1/2}\tilde{H}_{k+1}^{-1}H^{1/2}) = \kappa(R_{k+1}) \leq \kappa(R_k) = \kappa(H^{1/2}\tilde{H}_k^{-1}H^{1/2}) \tag{B–78}$$

and for $\tilde{H}_{k+1}^{-1}$ to be positive definite if and only if $\theta_k \in [0, 1]$.

This theorem is general. We can translate this theorem into the problem of interest to us as follows. Suppose $\tilde{H}_k^{-1}$ is the positive definite approximation to the inverse of the fixed positive definite matrix H of a quadratic function f; see Eq. B–74. $\tilde{H}_{k+1}^{-1}$ is obtained by Eq. B–73 where $s_k = x_{k+1} - x_k$ and $y_k = g_{k+1} - g_k$. The equation $y_k = Hs_k$ always holds for the quadratic function case. Assume the eigenvalues of the matrix $R_k = H^{1/2}\tilde{H}_k^{-1}H^{1/2}$ satisfy $0 \leq \lambda_1 \leq \lambda_2, \cdots, \leq \lambda_n$, then the condition

$$\lambda_1 \leq 1/\gamma_k \leq \lambda_n, \tag{B–79}$$

is a sufficient condition for

$$\kappa(H^{1/2}\tilde{H}_{k+1}^{-1}H^{1/2}) = \kappa(R_{k+1}) \leq \kappa(R_k) = \kappa(H^{1/2}\tilde{H}_k^{-1}H^{1/2}). \tag{B–80}$$

This theorem suggests that we should choose the scaling factor γ_k between $1/\lambda_n$ and $1/\lambda_1$ to make the condition number of R decrease from iteration k to iteration $k + 1$. If we chose γ_k this way, at least in the positive definite quadratic case, the condition number of R_k is monotonically decreasing. We do not know λ_1 and λ_n in general, but the following theorem indicates how to select the scaling factor γ_k so that $\lambda_1 \leq 1/\gamma_k \leq \lambda_n$ without explicit knowledge of λ_1 and λ_n.

THEOREM 2: *Let s_k, y_k be nonzero vectors in E^n such that $s_k^T y_k > 0$. Assume that $\tilde{H}_k^{-1}$, H and R are real symmetric positive definite matrices such that $y_k = Hs_k$ and $R_k = H^{1/2}\tilde{H}_k^{-1}H^{1/2}$. Define*

$$\gamma_k^{\varphi}(\tilde{H}_k^{-1}, s_k, y_k) = (1 - \varphi)\frac{s_k^T y_k}{y_k^T \tilde{H}_k^{-1} y_k} + \varphi\frac{s_k^T \tilde{H}_k s_k}{s_k^T y_k}. \tag{B–81}$$

Then for any $\varphi \in [0,1]$,

$$\frac{1}{\lambda_n} \leq \gamma_k^{\varphi}(\tilde{H}_k^{-1}, s_k, y_k) \leq \frac{1}{\lambda_1},$$

where λ_1 *and* λ_n *are the smallest and the largest eigenvalue of* $R_k = H^{1/2}\tilde{H}_k^{-1}H^{1/2}$.

We can apply this theorem to the problem of interest to us by defining $s_k = x_{k+1} - x_k$ and $y_k = g_{k+1} - g_k$. Theorem 2 in combination with Theorem 1 indicates that if we choose the scaling factor according to Eq. B–81, the condition number of the matrix R_{k+1} will be less than the condition number of R_k, at least for the quadratic objective function case. Eq. B–81 involves $\tilde{H}_k$, the inverse of $\tilde{H}_k^{-1}$, which is not easy to evaluate (note that we form $\tilde{H}_k^{-1}$ instead of $\tilde{H}_k$ at each iteration). If we substitute $s_k = -\alpha_k \tilde{H}_k^{-1} g_k$, we obtain

$$\gamma^{(1)}(\tilde{H}_k^{-1}, s_k, y_k) = \frac{s_k^T \tilde{H}_k s_k}{s_k^T y_k} = -\frac{\alpha_k g_k^T s_k}{s_k^T y_k} = \frac{g_k^T s_k}{g_k^T \tilde{H}_k^{-1} y_k}. \tag{B–82}$$

We can use either $-\alpha_k g_k^T s_k/(s_k^T y_k)$ or $g_k^T s_k/(g_k^T \tilde{H}_k^{-1} y_k)$ to evaluate the second term in the right hand side of Eq. B–81 when calculating γ_k. If we set $\varphi = 0$, Eq. B–81, gives $\gamma_k = (s_k^T y_k)/(y_k^T \tilde{H}_k^{-1} y)$ which is the scaling factor used by several authors (see Shanno and Phua (1978), Yang and Watson (1988)). Some authors (Shanno and Phua (1978)) suggest initial scaling in which only the initial inverse Hessian approximation $\tilde{H}_0^{-1}$ is scaled by a factor γ_0 and in subsequent iterations we set all $\gamma_k = 1$. The initial scaling strategy is superior to the traditional way of choosing $\gamma_k = 1$ for all k since initial scaling not only has "property 1" but also benefits from the freedom of choosing γ_0. Property 1 means that for a quadratic function, the nth inverse Hessian approximation $\tilde{H}_n^{-1}$ will be exactly equal to H^{-1}, where n is the dimension of the problem, so the algorithm will satisfy quadratic termination. Unfortunately, for general objective functions, scaling only at the first iteration may be insufficient. When considering non-quadratic functions, one may expect that as the algorithm proceeds changes in the Hessian may cause the eigenvalues of the matrix $H_k^{1/2}\tilde{H}_k^{-1}H_k^{1/2}$ to drift farther from unity unless the problem is rescaled. It should be possible to improve convergence by occasionally rescaling. Unfortunately, by allowing the scaling factor γ to vary, we lose the property that $\tilde{H}_n^{-1} = H^{-1}$ in the

quadratic case (where n denotes the dimension of the problem), i.e, we lose "property 1". On the other hand, we ensure monotonic improvement in the convergence rate if the γ_k is calculated by Eq. B–81, which is implied by monotonic decrease in $\kappa(R_k)$ which is called "property 2".

From Eq. B–81, we see that $\gamma_k = \gamma_k^{\varphi}$ can vary from $s_k^T y_k / y_k^T \tilde{H}_k^{-1} y_k$ to $s_k^T \tilde{H}_k s_k / s_k^T y_k$. The following proposition gives bounds on γ_k which can help us choose the scaling factor γ_k. This proposition is a part of a proposition presented by Oren (1973).

PROPOSITION 2: *Let $f(x)$ be a real-valued twice continuously differentiable objective function, $g_k = \nabla f(x_k)$, $s_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$, $\tilde{H}_k^{-1}$ be the inverse Hessian approximation at the kth iteration given by Eq. B–73 and let*

$$\tau_k = g_k^T s_k / (g_k^T \tilde{H}_k^{-1} y_k) \tag{B–83}$$

and

$$\sigma_k = s_k^T y_k / (y_k^T \tilde{H}_k^{-1} y_k) \tag{B–84}$$

with γ_k^{φ} given by Eq. B–81, then

(a) $\sigma_k \leq \gamma_k \leq \tau_k$

(b)$\lambda_1 < 1/\tau_k \leq 1/\sigma_k \leq \lambda_n$ where λ_1 and λ_n are the smallest and largest eigenvalue of R_k respectively.

From the previous discussion, we know that by allowing the scaling factor γ_k to vary, we lose the property $\tilde{H}_n^{-1} = H^{-1}$ in the quadratic case. Therefore, it seems desirable to choose γ_k such that γ_k falls between $1/\lambda_n$ and $1/\lambda_1$ on the one hand and such that γ_k varies as little as possible on the other hand. Because R_k is real symmetric, λ_1, λ_n, τ_k and σ_k are positive numbers. Thus, part (b) of proposition 2 is equivalent to

$$\frac{1}{\lambda_n} \leq \sigma_k \leq \gamma_k \leq \tau_k \leq \frac{1}{\lambda_1}. \tag{B–85}$$

To reduce oscillation in γ_k and avoid unnecessary scaling, it is desirable to use the value of γ_k closest to unity and this leads to the following procedure:

(1) If $\sigma_k > 1$ choose $\gamma_k = \sigma_k$.

(2) If $\tau_k < 1$ choose $\gamma_k = \tau_k$.

(3) If $\tau_k \geq 1 \geq \sigma_k$ choose $\gamma_k = 1$.

Oren and Spedicato (1976) proposed an additional property based on minimizing the sharp upper bound on the condition number of the inverse Hessian approximation matrix at each iteration, i.e, $\tilde{H}_{k+1}^{-1}$. Reducing this condition number can be important for decreasing the round off error. The following theorem gives the sharp upper bound on the condition number of the inverse Hessian approximation.

THEOREM 3: *Let $\tilde{H}_{k+1}^{-1}$ be defined by Eq. B–73, assume $\tilde{H}_k^{-1}$ is positive definite, that $s_k^T y_k > 0$,*

$$\frac{s_k^T y_k}{y_k^T \tilde{H}_k^{-1} y_k} \leq \gamma_k \leq \frac{s_k^T \tilde{H}_k s_k}{s_k^T y_k}, \tag{B–86}$$

and $\theta_k > 0$. Then

$$\kappa(\tilde{H}_{k+1}^{-1}) \leq \kappa(\tilde{H}_k^{-1})[\zeta + (\zeta^2 - 1)^{1/2}]^2, \tag{B–87}$$

where

$$\zeta = \frac{\nu}{\mu^{1/2}}, \tag{B–88}$$

$$\nu = \frac{s_k \tilde{H}_k s_k + \mu y_k^T \tilde{H}_k^{-1} y_k}{2 s_k^T y_k}, \tag{B–89}$$

and

$$\mu = \frac{\gamma_k \left[(s_k^T y_k)^2 + \theta_k \left((s_k^T \tilde{H}_k s_k)(y_k^T \tilde{H}_k^{-1} y_k) - (s_k^T y_k)^2 \right) \right]}{(y_k^T \tilde{H}_k^{-1} y_k)(s_k^T y_k)}. \tag{B–90}$$

Furthermore, Eq. B–87 becomes equality if $\tilde{H}_k^{-1} = I$ or $\tilde{H}_k^{-1} y_k = s_k$.

From the numerical stability standpoint, it is desirable to use an update that will minimize the condition number of $\tilde{H}_{k+1}^{-1}$ at each iteration. As we do not have a single procedure to do so, a reasonable alternative is to minimize the upper bound of the condition number of $\tilde{H}_{k+1}^{-1}$ by proper selection of θ_k and γ_k. Based on Oren's definition, if $\tilde{H}_k^{-1}$ is positive definite and $s_k^T y_k > 0$, then $\tilde{H}_{k+1}^{-1}$ obtained by Eq. B–73 is said to be optimally conditioned if γ_k and θ_k are such that $\tilde{H}_{k+1}^{-1}$ is positive definite and the right hand side of Eq. B–87 is minimized. Oren and Spedicato (1976) showed that if the relation given by

$$\theta_k = \frac{a_k b_k - a_k^2 \gamma_k}{b_k c_k \gamma_k - a_k^2 \gamma_k}, \tag{B–91}$$

where $a_k = s_k^T y_k$, $b_k = s_k^T \tilde{H}_k s_k$ and $c_k = y_k^T \tilde{H}_k^{-1} y_k$ is satisfied, then the matrix $\tilde{H}_{k+1}^{-1}$ is optimally conditioned. For BFGS, $\theta_k = 1$ for all k and solving Eq. B–91 for the optimal scaling factor γ_k gives

$$\gamma_k = \frac{a_k}{c_k} = \frac{s_k^T y_k}{y_k^T \tilde{H}_k^{-1} y_k}. \tag{B–92}$$

If the full matrix of $\tilde{H}_k^{-1}$ is used to implement the BFGS (or the scaled version of the BFGS algorithm), we refer to the algorithm as the standard BFGS (or standard scaled BFGS). Nocedal (1980) developed an algorithm called limited memory BFGS, denoted by LBFGS, in which $\tilde{H}_0^{-1}$ is diagonal and it is not necessary to store the full $\tilde{H}_k^{-1}$ matrix at any iteration.

To develop the LBFGS method, the BFGS update equation Eq. B–49 is rewritten as

$$\tilde{H}_{k+1}^{-1} = V_k^T \tilde{H}_k^{-1} V_k + \rho_k s_k s_k^T, \tag{B–93}$$

where $\rho_k = 1/y_k^T s_k$ and $V_k = I - \rho_k y_k s_k^T$. If only L previous vectors are used to construct $\tilde{H}_{k+1}^{-1}$, memory will be saved. When $k < L$ the update equation is as usual, which is given by

$$\begin{aligned} \tilde{H}_{k+1}^{-1} =& V_k^T V_{k-1}^T \cdots V_0^T \tilde{H}_0^{-1} V_0 \cdots V_{k-1} V_k \\ &+ V_k^T \cdots V_1^T \rho_0 s_0 s_0^T V_1 \cdots V_k \\ &\vdots \\ &+ V_k^T \rho_{k-1} s_{k-1} s_{k-1}^T V_k \\ &+ \rho_k s_k s_k^T. \end{aligned} \tag{B–94}$$

For $k + 1 > L$ the update equation is

$$\begin{aligned} \tilde{H}_{k+1}^{-1} =& V_k^T V_{k-1}^T \cdots V_{k-L+1}^T \tilde{H}_0^{-1} V_{k-L+1} \cdots V_{k-1} V_k \\ &+ V_k^T \cdots V_{k-L+2}^T \rho_{k-L+1} s_{k-L+1} s_{k-L+1}^T V_{k-L+2} \cdots V_k \\ &\vdots \\ &+ V_k^T \rho_{k-1} s_{k-1} s_{k-1}^T V_k \\ &+ \rho_k s_k s_k^T. \end{aligned} \tag{B–95}$$

This method can be implemented by using a very efficient recursive formula. This recursive form is used to compute $\tilde{H}^{-1} \times g$ which is required to construct the search direction in quasi-Newton method. The following is the algorithm proposed by Nocedal (1980).

1. If $k \leq L$, set **incr**=0 and **bound**=k; else set **incr**=$k - L$ and **bound**=L

2. $q_{\mathbf{bound}} = g_k$.

3. For i=**bound**-1, ..., 0

$$\begin{cases} j = i + \mathbf{incr} \\ \alpha_i = \rho_j s_j^T q_{i+1} \\ q_i = q_{i+1} - \alpha_i y_j \end{cases}$$

$r_0 = \tilde{H}_0^{-1} \times q_0$

For i=0, 1, ..., **bound-1**

$$\begin{cases} j = i + \mathbf{incr} \\ \beta_j = \rho_j y_j^T r_i \\ r_{i+1} = r_i + s_j(\alpha_i - \beta_i) \end{cases}$$

where k is the iteration number; L is the user specified number of previous vectors used in the algorithm;. In this procedure, r_i is equal to the search direction vector given by $-\tilde{H}_i^{-1} \times g_i$.

We can see that this recursive form only involves a small number of vector operations and only requires enough memory to store a few vectors if $\tilde{H}_0^{-1}$ is a diagonal matrix. Scaling can also be introduced into this algorithm (see Liu and Nocedal (1989)). In their paper, they conclude that scaling $\tilde{H}_0^{-1}$ by a factor of γ_k at each iteration, i.e. replacing $\tilde{H}_0^{-1}$ by $\gamma_k \tilde{H}_0^{-1}$ in step 3 of the recursion algorithm, is very effective. But this scaling is different from the scaling scheme discussed above where we scale $\tilde{H}_k^{-1}$ by γ_k at each iteration. If we scale $\tilde{H}_k^{-1}$ by γ_k, Eqs. B–94 and B–95, respectively,

should be written as shown below.

When $k < L$ the update equation is given by

$$\begin{aligned}\tilde{H}_{k+1}^{-1} =&\gamma_k^{1/2} V_k^T \gamma_{k-1}^{1/2} V_{k-1}^T \cdots \gamma_0^{1/2} V_0^T \tilde{H}_0^{-1} \gamma_0^{1/2} V_0 \cdots \gamma_{k-1}^{1/2} V_{k-1} \gamma_k^{1/2} V_k \\ &+ \gamma_k^{1/2} V_k^T \cdots \gamma_1^{1/2} V_1^T \rho_0 s_0 s_0^T \gamma_1 1/2 V_1 \cdots \gamma_k^{1/2} V_k \\ &\vdots \\ &+ \gamma_k^{1/2} V_k^T \rho_{k-1} s_{k-1} s_{k-1}^T \gamma_k^{1/2} V_k \\ &+ \rho_k s_k s_k^T.\end{aligned} \tag{B–96}$$

For $k + 1 > L$ the update equation is

$$\begin{aligned}\tilde{H}_{k+1}^{-1} =&\gamma_k^{1/2} V_k^T \gamma_{k-1}^{1/2} V_{k-1}^T \cdots \gamma_{k-L+1}^{1/2} V_{k-L+1}^T \tilde{H}_0^{-1} \gamma_{k-L+1}^{1/2} V_{k-L+1} \cdots \gamma_{k-1}^{1/2} V_{k-1} \gamma_k^{1/2} V_k \\ &+ \gamma_k^{1/2} V_k^T \cdots \gamma_{k-L+2}^{1/2} V_{k-L+2}^T \rho_{k-L+1} s_{k-L+1} s_{k-L+1}^T \gamma_{k-L+2}^{1/2} V_{k-L+2} \cdots \gamma_k^{1/2} V_k \\ &\vdots \\ &+ \gamma_k^{1/2} V_k^T \rho_{k-1} s_{k-1} s_{k-1}^T \gamma_k^{1/2} V_k \\ &+ \rho_k s_k s_k^T.\end{aligned} \tag{B–97}$$

APPENDIX C

CONJUGATE GRADIENT ALGORITHMS

The conjugate gradient method is a minimization procedure which only requires the first derivative, i.e., the gradient, of the objective function. Although our main interest is nonlinear problems, we first discuss the methods for solving linear systems of equations.

C.1 Conjugate Gradient Methods for Linear Problems

In this section, several conjugate gradient algorithms including preconditioned conjugate gradient algorithms for solving linear problems are given.

C.1.1 Standard Conjugate Gradient Methods

Consider the problem of solving a linear equation system given by

$$Ax = b, \tag{C–1}$$

where A is an $n \times n$ real symmetric positive definite matrix and b is a real $n-$dimensional column vector.

LEMMA: If A is real symmetric positive definite, solving $Ax = b$ is equivalent to minimizing the quadratic form:

$$\hat{f}(x) = \frac{1}{2}x^T Ax - b^T x = \frac{1}{2}(x, Ax) - (x, b), \tag{C–2}$$

where $(\cdot, \cdot)$ represents the standard inner product (e.g. $(u, v) = v^T u$.)

Taking the gradient of the function $\hat{f}(x)$ gives

$$\nabla \hat{f}(x) = Ax - b \equiv -r, \tag{C–3}$$

where r is the residual. Eq. C–3 indicates that the negative gradient of $\hat{f}(x)$ is the residual of the linear system $Ax = b$. Note $\nabla \hat{f}(x) = 0$ is equivalent to $Ax = b$ which is the linear system we wish to solve. Moreover, since A is positive definite, it can be shown that $\hat{f}(x)$ is minimum at x^* if and only if $Ax^* = b$. Thus, solving Eq. C–1 is equivalent to an optimization problem. To minimize the objective function given by Eq. C–2, we can apply different algorithms. If we choose the negative gradient as the search direction and do a line search in this direction at each iteration, then the resulting algorithm is the steepest descent algorithm. In a conjugate direction method, search directions are required to satisfy orthogonality conditions. The conjugate gradient method is a type of conjugate direction method. The conjugate gradient method has the property of quadratic termination which means that the method will locate the minimizing point of the quadratic function of Eq. C–2 in at most n iterations. We show that minimizing Eq. C–2 is equivalent to minimizing the following function

$$f(x) = \frac{1}{2}\big(r, A^{-1}r\big). \tag{C–4}$$

Applying the definition of the inner product in Eq. C–4 and the definition of the residual gives

$$\begin{aligned} f(x) &= \frac{1}{2}(r, A^{-1}r) \\ &= \frac{1}{2}(Ax - b, A^{-1}(Ax - b)) \\ &= \frac{1}{2}(Ax - b, x - A^{-1}b) \\ &= \frac{1}{2}\big[(Ax, x) - (Ax, A^{-1}b) - (b, x) + (b, A^{-1}b)\big] \\ &= \frac{1}{2}\big[(Ax, x) - (x, b) - (b, x) + (b, A^{-1}b)\big] \\ &= \frac{1}{2}x^T Ax - b^T x + \frac{1}{2}b^T A^{-1}b \\ &= \hat{f}(x) + \frac{1}{2}b^T A^{-1}b. \end{aligned} \tag{C–5}$$

Note that the last term is constant so minimizing $\hat{f}(x)$ and $f(x)$ are equivalent. The basic iterative step for minimizing the objective function $f(x)$ takes the form

$$x_{k+1} = x_k + \alpha_k d_k, \tag{C–6}$$

where k denotes the iteration index, α_k represents the step size and d_k denotes the search direction. The step size is determined by performing a line search along the search direction. To do a line search, we try to find $\alpha = \alpha_k$ which minimizes $f(x_k + \alpha d_k)$, i.e., we minimize f along the search direction d_k. Applying the inner product, we can write $f(x_k + \alpha d_k)$ as

$$\begin{aligned} f(x_k + \alpha d_k) &= \frac{1}{2}(A(x_k + \alpha d_k) - b, A^{-1}(A(x_k + \alpha d_k) - b)) \\ &= \frac{1}{2}(Ax_k + \alpha Ad_k - b, A^{-1}(Ax_k + \alpha Ad_k - b)) \\ &= \frac{1}{2}\big[(Ax_k - b, A^{-1}(Ax_k - b)) + (Ax_k - b, \alpha A^{-1}Ad_k) \\ &\quad + (\alpha Ad_k, A^{-1}(Ax_k - b)) + (\alpha Ad_k, \alpha A^{-1}Ad_k)\big] \\ &= f(x_k) + \frac{1}{2}\big[\alpha(Ax_k - b, d_k) + \alpha(d_k, Ax_k - b) + \alpha^2(d_k, Ad_k)\big] \\ &= f(x_k) + \alpha(Ax_k - b, d_k) + \frac{1}{2}\alpha^2(d_k, Ad_k) \\ &= f(x_k) + h(\alpha), \end{aligned} \tag{C–7}$$

where

$$h(\alpha) = \alpha(Ax_k - b, d_k) + \frac{1}{2}\alpha^2(d_k, Ad_k) = -\alpha(r_k, d_k) + \frac{1}{2}\alpha^2(d_k, Ad_k), \tag{C–8}$$

and $r_k = b - Ax_k$ is the residual. Taking

$$\frac{dh(\alpha)}{d\alpha} = 0, \tag{C–9}$$

we find the optimal step size given by

$$\alpha = \alpha_k = \frac{(r_k, d_k)}{(d_k, Ad_k)}. \tag{C–10}$$

At each iteration, we construct a search direction d_k such that the search directions are from an orthogonal set $\{d_0, d_1, d_2, \cdots, d_k\}$. The new direction d_{k+1} is chosen such that

$$(d_{k+1}, d_j)_A \equiv (d_{k+1}, Ad_j) = 0, \qquad 0 \leqslant j \leqslant k. \tag{C–11}$$

There are many methods can be used to generate conjugate directions. In the conjugate gradient method, the new search direction is constructed by using the gradient at the current iteration and previous search direction, i.e.

$$d_{k+1} = r_{k+1} + \beta_k d_k, \tag{C–12}$$

where $r_{k+1} = b - Ax_{k+1} = -\nabla f(x_{k+1})$ and β_k is obtained by solving

$$(d^{k+1}, Ad^k) = 0 \tag{C–13}$$

which gives

$$\beta_k = -\frac{(r_{k+1}, Ad_k)}{(d_k, Ad_k)}. \tag{C–14}$$

From Eq. C–6, it follows that

$$r_{k+1} = b - Ax_{k+1} = b - Ax_k - \alpha_k Ad_k, \tag{C–15}$$

or

$$r_{k+1} = r_k - \alpha_k Ad_k. \tag{C–16}$$

Given the search direction and the step size along that direction, we can now write down the conjugate gradient algorithm for solving Eq. C–1 where A is an $n \times n$ real symmetric positive definite matrix.

Algorithm 1:

(1) Select initial guess x_0.

(2) Calculate

$$r_0 = b - Ax_0, \tag{C–17}$$

$$d_0 = r_0. \tag{C–18}$$

(3) Iteration loop

DO $k = 0, 1, \cdots$

$$\alpha_k = \frac{(r_k, d_k)}{(d_k, Ad_k)} = \frac{(d_k)^T r_k}{(d_k)^T Ad_k}, \tag{C–19}$$

$$r_{k+1} = r_k - \alpha_k Ad_k, \tag{C–20}$$

$$x_{k+1} = x_k + \alpha_k d_k, \tag{C–21}$$

$$\beta_k = -\frac{(r_{k+1}, Ad_k)}{(d_k, Ad_k)} = -\frac{(d_k)^T Ar_{k+1}}{(d_k)^T Ad_k}, \tag{C–22}$$

$$d_{k+1} = r_{k+1} + \beta_k d_k. \tag{C–23}$$

END DO

We can prove that the gradient or residual at the current step is orthogonal to all the previous search directions. Applying Eqs. C–16 and C–10, we obtain

$$\begin{aligned}(r_{k+1}, d_k) &= (r_k - \alpha_k A d_k, d_k) \\ &= (r_k, d_k) - \alpha_k (A d_k, d_k) \\ &= (r_k, d_k) - \frac{(r_k, d_k)}{(d_k, A d_k)} (A d_k, d_k) \\ &= 0.\end{aligned} \tag{C–24}$$

Applying Eq. C–24 and the fact that the search directions are $A-$orthogonal for conjugate gradient method, we have

$$\begin{aligned}(r_{k+1}, d_{k-1}) &= (r_k - \alpha_k A d_k, d_{k-1}) \\ &= (r_k, d_{k-1}) - \alpha_k (A d_k, d_{k-1}) \\ &= 0.\end{aligned} \tag{C–25}$$

By mathematical induction, we have

$$\boxed{(r_k, d_j) = 0, \qquad \text{for} \quad j < k.} \tag{C–26}$$

Applying this equation, we find that

$$\begin{aligned}(r_k, d_k) &= (r_k, r_k + \beta_{k-1} d_{k-1}) \\ &= (r_k, r_k) + \beta_{k-1} (r_k, d_{k-1}) \\ &= (r_k, r_k).\end{aligned} \tag{C–27}$$

Applying Eqs. C–26, C–23 and C–12 gives

$$\begin{aligned}0 &= (r_{k+1}, d_k) \\ &= (r_{k+1}, r_k + \beta_{k-1} d_{k-1}) \\ &= (r_{k+1}, r_k) + \beta_{k-1} (r_{k+1}, d_{k-1}) \\ &= (r_{k+1}, r_k).\end{aligned} \tag{C–28}$$

Using this equation and Eq. C–26, we obtain

$$\begin{aligned}0 &= (r_{k+1}, d_{k-1}) \\ &= (r_{k+1}, r_{k-1} + \beta_{k-2} d_{k-2}) \\ &= (r_{k+1}, r_{k-1}) + \beta_{k-2} (r_{k+1}, d_{k-2}) \\ &= (r_{k+1}, r_{k-1}).\end{aligned} \tag{C–29}$$

By mathematical induction, we have

$$\boxed{(r_k, r_j) = 0, \qquad \text{for} \quad j < k.} \tag{C–30}$$

As a summary, we can write

$$(d_k, Ad_j) = 0 \qquad \text{for} \quad j \neq k, \tag{C–31}$$

$$(r_k, d_j) = 0 \qquad \text{for} \quad j \leq k, \tag{C–32}$$

$$(r_k, r_j) = 0 \qquad \text{for} \quad j \neq k. \tag{C–33}$$

Eqs. C–32 and C–33 hold when an exact line search is done. Let g be the gradient of the quadratic objective function given by Eq. C–4. Using the fact that the residual is equal to the negative gradient of the objective function, Eqs. C–32 and C–33 are, respectively, equivalent to

$$(g_k, d_j) = 0 \qquad \text{for} \quad j \leq k, \tag{C–34}$$

$$(g_k, g_j) = 0 \qquad \text{for} \quad j \neq k, \tag{C–35}$$

Eqs. C–31 through C–35 indicate that when applied to quadratic functions the conjugate gradient method has the following properties:

(i) the search directions are A-orthogonal;

(ii) the residuals are orthogonal;

(iii) the gradients are orthogonal;

(iv) the residual at the current iteration is orthogonal to all previous search directions;

(v) the gradient at the current iteration is orthogonal to all previous search directions.

Note that the last four properties require exact line searches.

From Eqs. C–16 and C–10, it follows that

$$(r_{k+1} - r_k, d_k) = -\alpha_k (Ad_k, d_k) = -(r_k, d_k). \tag{C–36}$$

From Eq. C–16

$$Ad_k = -\frac{1}{\alpha_k}(r_{k+1} - r_k). \tag{C–37}$$

Thus, using this result and Eq. C–30, we find

$$\begin{aligned}(r_{k+1}, Ad_k) &= (r_{k+1}, -\frac{1}{\alpha_k}(r_{k+1} - r_k)) \\ &= -\frac{1}{\alpha_k}(r_{k+1}, r_{k+1}).\end{aligned} \tag{C–38}$$

Using Eqs. C–23, C–37, C–26 and C–30, the inner product (d_k, Ad_k) can be written as

$$\begin{aligned}(d_k, Ad_k) &= (r_k + \beta_{k-1} d_{k-1}, -\frac{1}{\alpha_k}(r_{k+1} - r_k)) \\ &= \frac{1}{\alpha_k}(r_k, r_k).\end{aligned} \tag{C–39}$$

Substituting Eqs. C–38 and C–39 into the pertinent equations given in Algorithm 1, we obtain the following equivalent algorithm:

Algorithm 2:

(1) Select initial guess x_0.

(2) Calculate

$$r_0 = b - Ax_0, \tag{C–40}$$

$$d_0 = r_0. \tag{C–41}$$

(3) Iteration loop

DO $k = 0, 1, \cdots$

$$\alpha_k = \frac{(r_k, r_k)}{(d_k, Ad_k)} = \frac{(r_k)^T r_k}{(d_k)^T A d_k}, \tag{C–42}$$

$$r_{k+1} = r_k - \alpha_k A d_k, \tag{C–43}$$

$$x_{k+1} = x_k + \alpha_k d_k, \tag{C–44}$$

$$\beta_k = \frac{(r_{k+1}, r_{k+1})}{(r_k, r_k)} = \frac{(r_{k+1})^T r_{k+1}}{(r_k)^T r_k}, \tag{C–45}$$

$$d_{k+1} = r_{k+1} + \beta_k d_k. \tag{C–46}$$

END DO

C.1.2 Preconditioned Conjugate Gradient

Multiplying both sides of Eq. C–1 by C^{-1}, we get

$$C^{-1}Ax = C^{-1}b, \tag{C–47}$$

where C is chosen such that it is an approximation of A. C is called a preconditioning matrix. Note if C is exactly the same as A, then Eq. C–47 becomes

$$x = C^{-1}b = A^{-1}b, \tag{C–48}$$

i.e., the solution of Eq. C–1 is obtained. It is well known (see, for example, Greenbaum (1997)) that the error at the kth iteration of the conjugate gradient satisfies

$$\|e_k\|_A \leq 2\Big(\frac{\kappa(A)-1}{\kappa(A)+1}\Big)^k \|e_0\|_A, \tag{C–49}$$

where

$$e_k = A^{-1}b - x_k, \tag{C–50}$$

$\kappa(A)$ is the condition number of A and for any real n-dimensional column vector y,

$$\|y\|_A = (y,y)_A^{1/2} = \sqrt{y^T A y}. \tag{C–51}$$

Eq. C–49 suggests that the closer the condition number is to unity, the more rapid the convergence. The idea of preconditioning is to choose a matrix C close to A so that $\kappa(C^{-1}A) \leq \kappa(A)$. Then, if we apply the conjugate gradient method to the transformed problem given by Eq. C–47, the conjugate gradient method should converge faster than if we apply it to the original problem. Define the inner product:

$$(u,v)_C = u^T C v,$$

and the pseudo residual:

$$h_k = C^{-1}(b - Ax_k) = C^{-1}r_k.$$

In terms of the transformed problem, it can be shown (see Axelsson (1996)) that the Algorithm 2 becomes

Algorithm 3:

(1) Select initial guess x_0.

(2) Calculate

$$h_0 = C^{-1}r_0 = C^{-1}(b - Ax_0), \tag{C–52}$$

$$d_0 = h_0. \tag{C–53}$$

(3) Iteration loop

DO $k = 0, 1, \cdots$

$$\alpha_k = \frac{(h_k, h_k)_C}{(d_k, C^{-1}Ad_k)_C} = \frac{(h_k)^T C h_k}{(d_k)^T A d_k}, \tag{C–54}$$

$$h_{k+1} = h_k - \alpha_k C^{-1} A d_k, \tag{C–55}$$

$$x_{k+1} = x_k + \alpha_k d_k, \tag{C–56}$$

$$\beta_k = \frac{(h_{k+1}, h_{k+1})_C}{(h_k, h_k)_C} = \frac{(h_{k+1})^T C h_{k+1}}{(h_k)^T C h_k}, \tag{C–57}$$

$$d_{k+1} = h_{k+1} + \beta_k d_k. \tag{C–58}$$

END DO

The algorithm for the preconditioned conjugate gradient is analog to the algorithm for the standard conjugate gradient. Note that if we replace the residual by the pseudo residual and the standard inner product $(\cdot, \cdot)$ by the inner product $(\cdot, \cdot)_C$ in Algorithm 2, we obtain the algorithm for preconditioned conjugate gradient, i.e., Algorithm 3. However, when we implement the preconditioned conjugate gradient method, we will use the residual instead of the pseudo residual and the corresponding algorithm given below.

Algorithm 4:

(1) Select initial guess x_0.

(2) Calculate

$$r_0 = b - Ax_0, \tag{C–59}$$

$$d_0 = C^{-1}r_0. \tag{C–60}$$

(3) Iteration loop

DO $k = 0, 1, \cdots$

$$\alpha_k = \frac{(r_k, r_k)_{C^{-1}}}{(d_k, C^{-1}Ad_k)_{C^{-1}}} = \frac{(r_k)^T C^{-1} r_k}{(d_k)^T A d_k}, \tag{C–61}$$

$$r_{k+1} = r_k - \alpha_k A d_k, \tag{C–62}$$

$$x_{k+1} = x_k + \alpha_k d_k, \tag{C–63}$$

$$\beta_k = \frac{(r_{k+1}, r_{k+1})_{C^{-1}}}{(r_k, r_k)_{C^{-1}}} = \frac{(r_{k+1})^T C^{-1} r_{k+1}}{(r_k)^T C^{-1} r_k}, \tag{C–64}$$

$$d_{k+1} = C^{-1} r_{k+1} + \beta_k d_k. \tag{C–65}$$

END DO

This algorithm is called the Fletcher-Reeves algorithm; see Fletcher and Reeves (1964).

C.2 Conjugate Gradient Method for Nonlinear Problems

When minimizing a non-quadratic objective function, the meaning of the term residual is not so clear. By analogy, the residual for a linear problem will be replaced by the negative gradient of the objective function for a nonlinear problem.

The preconditioner has an important impact on the convergence rate of the conjugate gradient algorithm. For the linear quadratic problem, preconditioned conjugate gradient method solves $C^{-1}Ax = C^{-1}b$ where C is called the preconditioner, instead of $Ax = b$. In the history matching problem of interest to us, the Hessian matrix which is the second derivative of the objective function, is given by

$$H = C_M^{-1} + G^T C_D^{-1} G. \tag{C–66}$$

At an iteration of the Gauss-Newton method, we need to solve a problem of the form

$$Hx = (C_M^{-1} + G^T C_D^{-1} G)x = b. \tag{C–67}$$

For large problems, calculation of the whole sensitivity coefficient matrix G is too computationally expensive to be feasible. If we do not compute G directly, it is difficult to use information in G to obtain a good preconditioner. Thus, the obvious

choice for a preconditioning matrix is C_M^{-1}. Using C_M^{-1} as the preconditioner can significantly improve the convergence rate compared to applying the conjugate gradient algorithm without using any preconditioner (see Kalita and Reynolds (2000)). But Kalita found that this preconditioned conjugate gradient method often converges to a very high objective function value. Thus, a better preconditioner is needed. As the quasi-Newton method provides an approximation of the inverse Hessian, we attempted to generate an approximation to the inverse Hessian within the conjugate gradient procedure and use this approximate inverse Hessian as a preconditioner. If we applied the quasi-Newton method to a quadratic function, given that the exact line search is done, then the Hessian inverse approximation would become the true inverse Hessian at the nth iteration where n is the dimension of the problem. Even though the preconditioner is an approximation of the coefficient matrix (for example, it is Hessian when solving Eq. C–67), the inverse of the preconditioning matrix will be used in our implementation of the conjugate gradient algorithm. So we can use the inverse Hessian approximation calculated from the quasi-Newton update equation as the preconditioner inverse to implement the conjugate gradient algorithm. The algorithm follows the description given in Kalita and Reynolds (2000) and is given below.

(1) Select initial guess m_0 and initialize the iteration index k and restart counter i by setting $k = 0$ and $i = 0$ respectively.

(2) Calculate

$$r_0 = -\nabla_m O(m_0). \tag{C–68}$$

Select the initial preconditioning matrix M_0 (C_M^{-1} or $\tilde{H}_0^{-1}$), calculate "pseudo" residual

$$s_0 = M_0^{-1} r_0, \tag{C–69}$$

and select the initial search direction equal to the "pseudo" residual, i.e.,

$$d_0 = s_0. \tag{C–70}$$

Calculate

$$\delta_o = \delta_{\text{new}} = r_0^T d_0 = r_0^T M_0^{-1} r_0. \tag{C–71}$$

(3) Iteration loop

$k = 0, \ldots, k_{\max}$ (CG iteration loop)

Calculate the step size α_k in the search direction d_k by using the approach discussed later in this appendix. Then calculate

$$\begin{aligned} m_{k+1} &= m_k + \alpha_k d_k, & \text{(C–72)} \\ r_{k+1} &= -\nabla_m O(m_{k+1}), & \text{(C–73)} \\ \delta_{\text{mid}} &= r_{k+1}^T s_k = r_{k+1}^T M_k^{-1} r_k, & \text{(C–74)} \\ s_{k+1} &= M_{k+1}^{-1} r_{k+1}, & \text{(C–75)} \\ \delta_{\text{new}} &= r_{k+1}^T s_{k+1} = r_{k+1}^T M_{k+1}^{-1} r_{k+1}. & \text{(C–76)} \end{aligned}$$

(Note if C_M^{-1} is used as the preconditioning matrix then the preconditioning matrix will be fixed for all iterations, i.e., $M_k^{-1} = C_M$ for all k's. If the preconditioning matrix is generated from quasi-Newton method, then $M_k^{-1} = \tilde{H}_k^{-1}$ at the kth iteration.)

Calculate β_{k+1}:

$$\begin{aligned} \beta^{PR} &= \frac{r_{k+1}^T (s_{k+1} - s_k)}{r_k^T s_k} = \frac{\delta_{\text{new}} - \delta_{\text{mid}}}{\delta_o}, & \text{(C–77)} \\ \beta_{k+1} &= \max\left\{\beta^{PR}, 0\right\}. & \text{(C–78)} \end{aligned}$$

If $i = n$ (n is the maximum number of iteration allowed before restart) or $r_k^T d_k \leq 0$

set

$$d_{k+1} = s_{k+1}; \quad \beta_{k+1} = 0; \quad i = 0.$$

else

$$d_{k+1} = s_{k+1} + \beta_{k+1} d_k \tag{C–79}$$

Endif

$$i = i + 1 \tag{C–80}$$

End k loop

For our history matching problem, the whole objective function can be written as

$$O(m) = O_m + O_d. \tag{C–81}$$

If we are interested in constructing the MAP estimate then

$$O_m = (m - m_{\text{prior}})^T C_M^{-1} (m - m_{\text{prior}}), \tag{C–82}$$

and

$$O_d = (g(m) - d_{\text{obs}})^T C_D^{-1} (g(m) - d_{\text{obs}}). \tag{C–83}$$

If we are interested in calculating a realization of the model m by the randomized maximum likelihood method, then

$$O_m = (m - m_{\text{uc}})^T C_M^{-1} (m - m_{\text{uc}}), \tag{C–84}$$

and

$$O_d = (g(m) - d_{\text{uc}})^T C_D^{-1} (g(m) - d_{\text{uc}}). \tag{C–85}$$

The preceding equations assume we do not incorporate a correction to the prior mean. If we do so, then the objective function is given by Eq. 2.29. In the remainder of this appendix, we give the equations for calculating the MAP estimate, but it is quite simple to modify the equations to obtain the relevant equation for generating a realization with or without a correction to the prior mean.

Some technical details about preconditioned conjugate gradient algorithm are given below.

1. Calculation of the gradient of the objective function.

$$\nabla_m O(m_k) = \nabla_m \big[\frac{1}{2}(g(m_k) - d_{\text{obs}})^T C_D^{-1} (g(m_k) - d_{\text{obs}})\big] + C_M^{-1}(m_k - m_{\text{prior}}) \tag{C–86}$$

The first term $\nabla_m O_d(m_k)$ can be calculated by using the adjoint method. The computational cost of computing this term is roughly equivalent to one

simulation run. The second term in Eq. C–86 can be obtained by solving the following linear equation system for x.

$$C_M x = m_k - m_{\text{prior}}. \tag{C–87}$$

Because the coefficient matrix C_M is fixed, we just do an LU decomposition of C_M once. At subsequent iterations, the solution of Eq. C–87 can be obtained by just forward and backward substitution. This assumes that C_M is not so large that we cannot afford to store its LU decomposition. If this is the case, then Eq. C–87 will be solved by preconditioned linear conjugate gradient method using incomplete LU decomposition to generate a preconditioning matrix.

2. Step size calculation.

   The step size can be obtained by minimizing $f(\alpha) = O(m_k + \alpha d_k)$ with respect to α. The value of α at its minimum can be found by solving $h(\alpha) = f'(\alpha) = 0$ for α, i.e., we solve

$$h(\alpha) \equiv f'(\alpha) = \frac{dO(m_k + \alpha d_k)}{d\alpha} = \big(\nabla O(m_k + \alpha d_k)\big)^T d_k = 0, \tag{C–88}$$

   for α. This equation can be solved by using the Newton-Raphson algorithm which is given by

$$\alpha_{j+1} = \alpha_j - \frac{h(\alpha_j)}{h'(\alpha_j)} \tag{C–89}$$

   where j denotes the index of Newton-Raphson iteration and the first derivative of f can be evaluated by

$$h'(\alpha) = \frac{dh(\alpha)}{d\alpha} = d_k^T \nabla \Big[\big(\nabla (m_k + \alpha d_k)\big)^T \Big] d_k = d_k^T H(m_k + \alpha d_k) d_k. \tag{C–90}$$

   The Newton-Raphson iteration could be stopped when a suitable convergence criterion is satisfied. However, an exact line search based on the Newton-Raphson iteration is very expensive due to the fact that the evaluation of the term $d_k^T H(m_k + \alpha d_k) d_k$ requires one simulation run. In our procedure, we do only one Newton-Raphson iteration with $\alpha_0 = 0$. As discussed Chapter V, if α_1 is such that the Wolfe conditions are not satisfied, we apply a quadratic fit

followed by cuts in the step size to find a suitable α_k and $m_{k+1} = m_k + \alpha_k d_k$. To perform one Newton-Raphson iteration, we set $\alpha_0 = 0$ and then Eq. C–89 gives

$$\alpha_1 = -\frac{\big(\nabla O(m_k)\big)^T d_k}{d_k^T H(m_k) d_k}. \tag{C–91}$$

Because the problem is nonlinear and an inexact line search is used, the search directions may not be orthogonal or the calculated search direction may be uphill which is indicated by $r_k^T d_k < 0$. So we restart the CG iteration using search direction given by $d_0 = -M_0^{-1}\nabla O(m_k)$ whenever a user specified number of iteration for restart is reached or $r_k^T d_k < 0$. The denominator in Eq. C–91 can be calculated without explicit calculation of the Hessian matrix. If the Hessian matrix is in the form of

$$H_k = G_k^T C_D^{-1} G_k + C_M^{-1}, \tag{C–92}$$

then

$$\begin{aligned} d_k^T H_k d_k &= d_k^T (G_k^T C_D^{-1} G_k + C_M^{-1}) d_k \\ &= d_k^T (G_k^T C_D^{-1} G_k) d_k + d_k^T C_M^{-1} d_k \\ &= (G_k d_k)^T C_D^{-1} (G_k d_k) + d_k^T C_M^{-1} d_k. \end{aligned} \tag{C–93}$$

To evaluate this last equation, we do not need to compute the sensitivity coefficient coefficient matrix G directly. We only need to calculate Gd_k which can be done by using a finite-difference method or gradient simulator method. The elements of the sensitivity coefficient matrix can be written as

$$g_{i,j} = \frac{\partial g_i}{\partial m_j}, \tag{C–94}$$

where $i = 1, \cdots, N_d$ and $j = 1, \cdots, N_m$ and $g(m)$ represents the vector of calculated data for the model m with g_i representing the ith component of g. The directional derivative is

$$\Big(\frac{dg}{d\alpha}\Big)_{\alpha=0} = \Big(\frac{dg(m + \alpha d_k)}{d\alpha}\Big)_{\alpha=0}. \tag{C–95}$$

Let $u = d_k / \parallel d_k \parallel$, so we have

$$\begin{aligned}\Big(\frac{dg_i}{d\alpha}\Big)_{\alpha=0} &= [\nabla g_i(m)]^T u \\ &= \frac{1}{\parallel d_k \parallel}[\nabla g_i(m)]^T d_k.\end{aligned} \tag{C–96}$$

The ith component of Gd_k is given by

$$\begin{aligned}\big[Gd_k\big]_i &= \sum_{j=1}^{M} \frac{\partial g_i}{\partial m_j} d_{k,j} \\ &= [\nabla g_i(m)]^T d_k.\end{aligned} \tag{C–97}$$

Substituting Eq. C–96 into Eq. C–97, we obtain

$$\begin{aligned}[Gd_k]_i =& \parallel d_k \parallel \Big(\frac{dg}{d\alpha}\Big)_{\alpha=0} \\ \approx& \parallel d_k \parallel \frac{g_i(m+\epsilon d_k) - g_i(m)}{\epsilon \parallel d_k \parallel} \\ =& \frac{g_i(m+\epsilon d_k) - g_i(m)}{\epsilon},\end{aligned} \tag{C–98}$$

where ϵ is a small number. As this holds for all components of Gd_k, it follows that

$$Gd_k \approx \frac{g(m+\epsilon d_k) - g(m)}{\epsilon}. \tag{C–99}$$

The preceding derivation can be found in Kalita and Reynolds (2000). We choose ϵ based on the infinity norm of d_k such that ϵ satisfies $\epsilon \parallel d_k \parallel_\infty = 10^{-3}$. Note that calculating Gd_k needs one additional simulation run to evaluate $g(m+\epsilon d_k)$. Once we obtain Gd_k, we can use Eqs. C–93 and C–91 to calculate the step size.

If the Hessian matrix does not have the particular form given in Eq. C–92 we can use the finite-difference method to calculate $H_k d_k$ directly, i.e.,

$$H_k d_k = \frac{\nabla O(m_k + \epsilon d_k) - \nabla O(m_k)}{\epsilon}. \tag{C–100}$$

Applying Eq. C–100 requires one forward simulation run to calculate the primary variables that are required to form the adjoint system and one adjoint solution to form the gradient evaluated at $m_k + \epsilon d_k$. Whereas applying Eq. C–98 requires only one forward simulation run.

3. Calculation of preconditioner.

   There are two ways that can be used to obtain the preconditioning matrix. The first choice is to use $M_k = C_M^{-1}$ at all iterations. In the second choice, the preconditioner M_k can be obtained by using quasi-Newton update equation. In our implementation, we use the limited memory BFGS (LBFGS) proposed by Nocedal (1980) to generate $M \times r$ where r is the residual. Implementation of LBFGS only needs vector operations and only needs to store vectors. The details about calculating $M \times r$ are given in Appendix B. The difficulty with this procedure is that we can only approximate the quasi-Newton $\tilde{H}_k^{-1}$ using information in the conjugate gradient algorithm.

4. Calculation of β.

   In the algorithm, the procedure we use to calculate β_{k+1} is based on the Polak-Ribiere equation for β^{PR} (Eq. C–77); see Polak (1971). By selecting $\beta_{k+1} = max\{\beta^{PR}, 0\}$, we will restart the algorithm when $\beta^{PR} \leq 0$.

5. Restart CG.

   The conjugate gradient algorithm is restarted if any the following conditions holds:

   (i) $\beta_{k+1} \leq 0$;

   (ii) reach the maximum allowable number of iterations without restarting;

   (iii) $r_k^T d_k < 0$, i.e., the search direction is uphill.

APPENDIX D

RELATIONSHIP BETWEEN CONJUGATE GRADIENT AND QUASI-NEWTON METHODS

Much of the discussion in this appendix can be found in Murray (1972), Nazareth (1979), Buckley (1978b) and Buckley and Lenir (1983). Suppose we wish to minimize an objective function $f(x)$ by using an iterative scheme of the form

$$x_{k+1} = x_k + \alpha_k d_k \quad k = 0, 1, \cdots , \tag{D–1}$$

where k is the iteration index, x_0 is the initial guess, d_k is the search direction at the kth iteration and α_k is the step size. If an exact line search is used, then α_k minimizes $h(\alpha) = f(x_k + \alpha d_k)$. The theoretical results are often obtained under the restrictive condition that $f(x)$ is a quadratic function of the form

$$f(x) = \frac{1}{2} x^T A x - b^T x + c \tag{D–2}$$

where A is an $n \times n$ real symmetric positive definite matrix, b is a fixed $n-$dimensional column vector and c is a constant. It is easy to show that there is an unique x^* which minimizes $f(x)$ and x^* is the unique solution of

$$Ax = b. \tag{D–3}$$

Using a Taylor series expansion, it is easy to show that Eq. D–2 can also be written as

$$f(x) = f(x^*) + \frac{1}{2}(x - x^*)^T A (x - x^*). \tag{D–4}$$

Definition: A set of vectors $d_0, d_1, \cdots , d_{n-1}$ where $d_i \neq 0$ are conjugate with respect to a given real symmetric positive definite matrix A, if

$$d_i^T A d_j = 0 \qquad \text{for all} \quad i \neq j. \tag{D–5}$$

If A is equal to the $n \times n$ identity matrix, then Eq. D–5 reduces to the usual definition of orthogonality. It is easy to show that if $\{d_j\}_{j=0}^{n-1}$ are $A-$orthogonal, then $\{d_j\}_{j=0}^{n-1}$ are linearly independent.

Theorem: If the iterative scheme of Eq. D–1 with an exact line search is applied to minimize the quadratic objective function of Eq. D–2 starting with an initial guess x_0 and the set $\{d_j\}_{j=0}^{n-1}$ are $A-$orthogonal, then convergence to the x^* that minimizes $f(x)$ is obtained in at most n iterations. Moreover,

$$x_{i+1} = \text{argmin}\{f(x) \mid x = x_0 + \sum_{j=0}^{i} \beta_j d_j\}, \tag{D–6}$$

i.e., x_{i+1} minimizes $f(x)$ over the subspace consisting of all vectors of the form $x_0 + \sum_{j=0}^{i} \beta_j d_j$

The proof of this theorem can be found in Fletcher (1987) and Murray (1972). This theorem essentially says that for a quadratic function

$$\text{conjugacy} + \text{exact line search} = \text{quadratic termination.}$$

Let g_i denote the gradient of the quadratic objective function $f(x)$ given by Eq. D–2 evaluated at the x_i. Let $\{d_j\}$ denote search directions and $(\cdot, \cdot)$ denote the standard inner product on the set of real n-dimensional column vectors. We can show that

$$(Ad_i, d_j) = 0 \qquad \text{for} \quad i \neq j \tag{D–7}$$

$$(g_i, d_j) = 0 \qquad \text{for} \quad i > j \tag{D–8}$$

$$(g_i, g_j) = 0 \qquad \text{for} \quad i \neq j \tag{D–9}$$

hold for all quadratic functions (linear problems) when applying the conjugate gradient method (see Appendix C) to minimize the quadratic objective function of Eq. D–2. Due to round off errors, search directions may lose the conjugacy, so restarts are required. As derived in Appendix C, Eqs. D–8 and D–9 require an exact line search.

For nonlinear problems, Eqs. D–8 and D–9 do not hold. However, we can show that $(g_{j+1}, d_j) = 0$ holds if an exact line search is performed. Let

$$h(\alpha) = f(x_j + \alpha d_j), \tag{D–10}$$

where f is now a non-quadratic objective function. An exact line search will find the value of α ($\alpha = \alpha_j$) such that

$$\begin{aligned} 0 &= \frac{dh(\alpha)}{d\alpha}|_{\alpha=\alpha_j} \\ &= \left(\nabla f(x_j + \alpha d_j)|_{\alpha=\alpha_j}\right)^T d_j \\ &= \left(\nabla f(x_{j+1})\right)^T d_j \\ &= g_{j+1}^T d_j. \end{aligned} \tag{D–11}$$

Thus, the current gradient is orthogonal to the previous search direction, but is not necessarily orthogonal to all the previous search directions for a non-quadratic objective function.

Before we consider the relationship between the conjugate gradient and the quasi-Newton method, we write down the algorithms for both methods. For a quadratic function of the form given by Eq. D–2, the preconditioned conjugate gradient algorithm with preconditioner $\tilde{H}^{-1}$ is given below.

(1) Choose initial guess x_0 and a preconditioner $\tilde{H}$, set $d_0 = -\tilde{H}^{-1} g_0$ and iteration index j equal to zero.

(2) Find the step size α_j in the search direction d_j by an exact line search.

(3) Update the model and the search directions by applying

$$x_{j+1} = x_j + \alpha_j d_j, \tag{D–12}$$

$$\beta_{j+1} = \frac{g_{j+1}^T \tilde{H}^{-1} g_{j+1}}{g_j^T \tilde{H}^{-1} g_j}, \tag{D–13}$$

$$d_{j+1} = -\tilde{H}^{-1} g_{j+1} + \beta_{j+1} d_j = -\tilde{H}^{-1} g_{j+1} + \frac{g_{j+1}^T \tilde{H}^{-1} g_{j+1}}{g_j^T \tilde{H}^{-1} g_j} d_j. \tag{D–14}$$

(4) Determine whether the stopping criteria are satisfied. If satisfied, then stop; otherwise replace j by $j + 1$ and go to step 2.

In the above algorithm, $\tilde{H}^{-1}$ denotes the preconditioner. Note that $\tilde{H}^{-1} = I$ corresponds to the standard conjugate gradient method without preconditioning.

The algorithm for BFGS with $\tilde{H}_0^{-1} = \tilde{H}^{-1}$ is given by

(1) Choose an initial guess x_0 and the initial Hessian approximation matrix $\tilde{H}_0^{-1} = \tilde{H}^{-1}$. Set $d_0 = -\tilde{H}^{-1}g_0$ and set the iteration index j equal to zero.

(2) Find the step size α_j in the search direction d_j by an exact line search.

(3) Update the model and the search directions by applying

$$\begin{aligned} x_{j+1} &= x_j + \alpha_j d_j, && \text{(D–15)} \\ s_j &= x_{j+1} - x_j, && \text{(D–16)} \\ y_j &= g_{j+1} - g_j, && \text{(D–17)} \\ \tilde{H}_{j+1}^{-1} &= \tilde{H}_j^{-1} + \frac{1}{s_j^T y_j}\Big(1 + \frac{y_j^T \tilde{H}_j^{-1} y_j}{s_j^T y_j}\Big) s_j s_j^T \\ &\quad - \frac{1}{s_j^T y_j}\big(\tilde{H}_j^{-1} y_j s_j^T + s_j y_j^T \tilde{H}_j^{-1}\big), && \text{(D–18)} \\ d_{j+1} &= -\tilde{H}_{j+1}^{-1} g_{j+1}. && \text{(D–19)} \end{aligned}$$

(4) Determine whether the stopping criteria are satisfied. If satisfied, then stop; otherwise replace j by $j + 1$ and go to step 2.

For the quadratic function of Eq. D–2, applying a Taylor series expansion we have

$$g_{j+1} = g_j + A(x_{j+1} - x_j) \qquad \text{(D–20)}$$

where the Hessian matrix is equal to A. Let

$$y_j = g_{j+1} - g_j, \qquad \text{(D–21)}$$

$$s_j = x_{j+1} - x_j. \qquad \text{(D–22)}$$

From the algorithm, we note that

$$s_j = \alpha_j d_j. \qquad \text{(D–23)}$$

Using Eq. D–21 and D–22, Eq. D–20 can be rewritten as

$$y_j = A s_j \qquad \text{(D–24)}$$

which is called Newton condition. Using Eq. D–23 we can also write Eq. D–24 as

$$y_j = As_j = \alpha_j A d_j. \tag{D–25}$$

Therefore, the conjugate condition of Eq. D–5 can be replaced by

$$d_i^T y_j = 0 \qquad \text{for} \quad i \neq j. \tag{D–26}$$

From Eq. D–23, Eq. D–26 implies that

$$s_i^T y_j = \alpha_i d_i^T y_j = 0, \qquad \text{for} \quad i \neq j. \tag{D–27}$$

It is well known that both the conjugate gradient methods and the quasi-Newton methods satisfy the conjugate condition given by Eq. D–5; see Murray (1972).

For any $i < j$, Eq. D–27 and Eq. D–11 imply that, for a quadratic function,

$$\begin{aligned} s_i^T g_j &= s_i^T g_{i+1} + s_i^T (g_{i+2} - g_{i+1}) + s_i^T (g_{i+3} - g_{i+2}) +, \cdots, + s_i^T (g_j - g_{j-1}) \\ &= s_i^T g_{i+1} + s_i^T (y_{i+1} + y_{i+2} + \cdots + y_{j-1}) = 0 \qquad \text{for} \quad i < j. \end{aligned} \tag{D–28}$$

Note that the number of terms in the parentheses of the second term in Eq. D–28 is equal to $j - i - 1$. For example, if $i = j - 1$ then the second term in Eq. D–28 will disappear. Note that Eq. D–28 is equivalent to

$$\boxed{g_i^T s_j = 0 \qquad \text{for} \quad i > j.} \tag{D–29}$$

Eq. D–29 indicates that the current gradient is orthogonal to all the previous search directions when both conjugate gradient and quasi-Newton methods are applied to quadratic functions, given that an exact line search is performed.

For the BFGS algorithm, Eq. D–28 gives

$$\begin{aligned} s_i^T g_j &= \alpha_i d_i^T g_j \\ &= -\alpha_i (\tilde{H}_i^{-1} g_i)^T g_j \\ &= 0. \end{aligned} \tag{D–30}$$

Applying Eq. D–28 and the Hessian inverse approximation update equation used in

the BFGS algorithm, we have

$$\begin{aligned}
0 &= (\tilde{H}_i^{-1} g_i)^T g_j \\
&= g_i^T \tilde{H}_i^{-1} g_j \\
&= g_i^T \Big(\tilde{H}_{i-1}^{-1} + \frac{s_{i-1} y_{i-1}^T}{s_{i-1}^T y_{i-1}} \tilde{H}_{i-1}^{-1} \Big) g_j \\
&= g_i^T \tilde{H}_{i-1}^{-1} g_j + \frac{g_i^T s_{i-1}}{s_{i-1}^T y_{i-1}} y_{i-1}^T \tilde{H}_{i-1}^{-1} g_j \\
&= g_i^T \tilde{H}_{i-1}^{-1} g_j \\
&= g_i^T \Big(\tilde{H}_{i-2}^{-1} + \frac{s_{i-2} y_{i-2}^T}{s_{i-2}^T y_{i-2}} \tilde{H}_{i-2}^{-1} \Big) g_j \\
&= g_i^T \tilde{H}_{i-2}^{-1} g_j + \frac{g_i^T s_{i-2}}{s_{i-2}^T y_{i-2}} y_{i-2}^T \tilde{H}_{i-2}^{-1} g_j \\
&= g_i^T \tilde{H}_{i-2}^{-1} g_j \\
&\;\;\vdots \\
&= g_i^T \tilde{H}^{-1} g_j
\end{aligned} \tag{D–31}$$

which is valid for all $i < j$. Thus,

$$g_i^T \tilde{H}_i^{-1} g_j = 0 \quad \text{for} \quad i < j. \tag{D–32}$$

We can prove that

$$\tilde{H}_i^{-1} g_j = \tilde{H}^{-1} g_j, \tag{D–33}$$

holds for all $i < j$ by using mathematical induction on i. Now assume Eq. D–33 holds for some $\tilde{H}_{i-1}^{-1}$, i.e.,

$$\tilde{H}_{i-1}^{-1} g_j = \tilde{H}^{-1} g_j. \tag{D–34}$$

For $\tilde{H}_i^{-1}$, using Eqs. D–31, D–34 and D–21 we have

$$\begin{aligned}
\tilde{H}_i^{-1} g_j &= \tilde{H}_{i-1}^{-1} g_j + \frac{s_{i-1}}{s_{i-1}^T y_{i-1}} y_{i-1}^T \tilde{H}_{i-1}^{-1} g_j \\
&= \tilde{H}^{-1} g_j + \frac{s_{i-1}}{s_{i-1}^T y_{i-1}} (g_i - g_{i-1})^T \tilde{H}^{-1} g_j \\
&= \tilde{H}^{-1} g_j.
\end{aligned} \tag{D–35}$$

So using Eq. D–28, the search direction in BFGS (see Eq. D–18) can be written as

$$\begin{aligned} d_{j+1}^{BFGS} &= -\tilde{H}_{j+1}^{-1} g_{j+1} \\ &= -\tilde{H}_j^{-1} g_{j+1} + \frac{1}{s_j^T y_j}\left(y_j^T \tilde{H}_j^{-1} g_{j+1}\right) s_j \\ &= -\tilde{H}_j^{-1} g_{j+1} + \frac{(g_{j+1}-g_j)^T \tilde{H}_j^{-1} g_{j+1}}{\alpha_j d_j^T (g_{j+1}-g_j)} \alpha_j d_j \\ &= -\tilde{H}_j^{-1} g_{j+1} + \frac{g_{j+1}^T \tilde{H}_j^{-1} g_{j+1}}{-(\tilde{H}_j^{-1} g_j)^T (g_{j+1}-g_j)} d_j \\ &= -\tilde{H}^{-1} g_{j+1} + \frac{g_{j+1}^T \tilde{H}^{-1} g_{j+1}}{g_j^T \tilde{H}^{-1} g_j} d_j \\ &= d_{j+1}^{CG}. \end{aligned} \tag{D–36}$$

So the BFGS and conjugate gradient are equivalent no matter how we choose the initial Hessian inverse approximation in the BFGS algorithm. If the initial Hessian inverse approximation is the identity matrix, then BFGS is equivalent to the standard conjugate gradient method without preconditioning. If the initial Hessian inverse approximation is the an arbitrary matrix $\tilde{H}^{-1}$, then BFGS is equivalent to the preconditioned conjugate gradient method with preconditioner equal to $\tilde{H}^{-1}$. This result was derived assuming that the algorithm are applied to a quadratic objective function and the search direction is exact.

We summarize the well known properties of conjugate gradient and variable metric algorithms in this paragraph. When applied to the minimization of a quadratic function and using the same initial metric defined by $\tilde{H}^{-1}$, we have

(1) termination in at most n steps (quadratic termination);

(2) search direction vectors are conjugate;

(3) $g_i^T \tilde{H}^{-1} g_j = 0$ for $i < j$;

(4) the jth direction d_j lies in the subspace spanned by $\tilde{H}^{-1}g_0, \tilde{H}^{-1}g_1, \cdots, \tilde{H}^{-1}g_{j-1}$.

One of the more general conjugate gradient algorithms due to see Shanno and Phua (1978) is given below.

(1) Choose an initial guess x_0, set $d_0 = -g_0$ and set $j = 0$.

(2) Find the step size α_j in the direction d_j by an exact line search.

(3) Update the model and the search directions by applying

$$\begin{aligned} x_{j+1} &= x_j + \alpha_j d_j, && \text{(D–37)} \\ y_j &= g_{j+1} - g_j, && \text{(D–38)} \\ \beta_j &= \frac{y_j^T g_{j+1}}{y_j^T d_j}, && \text{(D–39)} \\ d_{j+1} &= -g_{j+1} + \beta_j d_j. && \text{(D–40)} \end{aligned}$$

(4) Determine whether the stopping criteria are satisfied. If satisfied, then stop; otherwise replace j by $j + 1$ and go to step 2.

If we do an exact line search, then we have $d_j^T g_{j+1} = 0$. If the objective function is quadratic, we have $g_j^T g_{j+1} = 0$. For quadratic functions, if we do an exact line search, the above algorithm becomes the Fletcher and Reeves (1964) algorithm where β_j is given by

$$\begin{aligned} \beta_j &= \frac{y_j^T g_{j+1}}{y_j^T d_j} \\ &= \frac{(g_{j+1} - g_j)^T g_{j+1}}{(g_{j+1} - g_j)^T d_j} \\ &= \frac{g_{j+1}^T g_{j+1}}{-g_j^T d_j} \\ &= \frac{g_{j+1}^T g_{j+1}}{-g_j^T(-g_j + \beta_{j-1} d_{j-1})} \\ &= \frac{g_{j+1}^T g_{j+1}}{g_j^T g_j}. \end{aligned} \qquad \text{(D–41)}$$

If we do an exact line search but relax the condition that the function is quadratic, we have

$$\beta_j = \frac{y_j^T g_{j+1}}{g_j^T g_j}, \qquad \text{(D–42)}$$

which is the Polak (1971) algorithm with an exact line search.

Eq. D–40 can be rewritten as

$$\begin{aligned} d_{j+1} &= -g_{j+1} + d_j\beta_j \\ &= -g_{j+1} + \frac{d_j(y_j^T g_{j+1})}{y_j^T d_j} \\ &= -g_{j+1} + \frac{(d_j y_j^T)g_{j+1}}{y_j^T d_j} \\ &= -\Big(I - \frac{d_j y_j^T}{y_j^T d_j}\Big)g_{j+1}. \end{aligned} \tag{D–43}$$

Perry proposed

$$d_{j+1} = -\Big(I - \frac{s_j y_j^T}{y_j^T s_j} + \frac{s_j s_j^T}{s_j^T y_j}\Big)g_{j+1} \equiv -Q_{j+1}g_{j+1} \tag{D–44}$$

where $s_j = \alpha_j d_j = x_{j+1} - x_j$, and I is the $n \times n$ identity matrix and

$$Q_{j+1} = I - \frac{s_j y_j^T}{y_j^T s_j} + \frac{s_j s_j^T}{s_j^T y_j}. \tag{D–45}$$

Note Q_{j+1} satisfies

$$\begin{aligned} y_j^T Q_{j+1} &= y_j^T \Big(I - \frac{s_j y_j^T}{y_j^T s_j} + \frac{s_j s_j^T}{s_j^T y_j}\Big) \\ &= y_j^T - \frac{(y_j^T s_j) y_j^T}{y_j^T s_j} + \frac{(y_j^T s_j) s_j^T}{s_j^T y_j} \\ &= y_j^T - y_j^T + s_j^T \\ &= \alpha_j d_j^T. \end{aligned} \tag{D–46}$$

Note that Q_{j+1} is not symmetric and also note that with an exact line search Eq. D–

44 can be rewritten as

$$\begin{aligned}
d_{j+1} &= -\Big(I - \frac{s_j y_j^T}{y_j^T s_j} + \frac{s_j s_j^T}{s_j^T y_j}\Big) g_{j+1} \\
&= -\Big(I - \frac{\alpha_j d_j y_j^T}{y_j^T \alpha_j d_j} + \frac{\alpha_j d_j \alpha_j d_j^T}{s_j^T y_j}\Big) g_{j+1} \\
&= -g_{j+1} + \frac{d_j y_j^T}{y_j^T d_j} g_{j+1} - \frac{\alpha_j d_j \alpha_j (d_j^T g_{j+1})}{s_j^T y_j} \\
&= -g_{j+1} + \frac{d_j y_j^T}{y_j^T d_j} g_{j+1} \\
&= -g_{j+1} + \frac{y_j^T g_{j+1}}{y_j^T d_j} d_j \\
&= -g_{j+1} + \frac{y_j^T g_{j+1}}{(g_{j+1} - g_j)^T d_j} d_j \\
&= -g_{j+1} + \frac{y_j^T g_{j+1}}{g_{j+1}^T d_j - g_j^T d_j} d_j \\
&= -g_{j+1} + \frac{y_j^T g_{j+1}}{-g_j^T d_j} d_j \\
&= -g_{j+1} + \frac{y_j^T g_{j+1}}{-g_j^T (-g_j + \beta_j d_{j-1})} d_j \\
&= -g_{j+1} + \frac{y_j^T g_{j+1}}{g_j^T g_j} d_j,
\end{aligned} \tag{D–47}$$

which is Polak-Ribiere algorithm; see Eq. D–42.

Shanno (1978a) proposed

$$Q_{j+1} = I - \frac{s_j y_j^T}{y_j^T s_j} - \frac{y_j s_j^T}{y_j^T s_j} + \frac{s_j s_j^T}{s_j^T y_j} \tag{D–48}$$

which is symmetric. Right multiplying Eq. D–48 by y_j gives

$$\begin{aligned}
Q_{j+1} y_j &= y_j - \frac{s_j y_j^T y_j}{y_j^T s_j} - y_j + s_j \\
&= s_j - \frac{s_j y_j^T y_j}{y_j^T s_j}.
\end{aligned} \tag{D–49}$$

which does not satisfy the quasi-Newton condition, i.e., $Q_{j+1} y_j \neq s_j$. If we define

$\bar{Q}_{j+1}$ by

$$\begin{aligned}\bar{Q}_{j+1} &= Q_{j+1} + \frac{y_j^T y_j s_j s_j^T}{s_j^T y_j s_j^T y_j} \\ &= I - \frac{s_j y_j^T + y_j s_j^T}{y_j^T s_j} + \Big(1 + \frac{y_j^T y_j}{s_j^T y_j}\Big)\frac{s_j s_j^T}{s_j^T y_j},\end{aligned} \tag{D–50}$$

then $\bar{Q}_{j+1}$ satisfies the quasi-Newton condition, i.e.,

$$\begin{aligned}\bar{Q}_{j+1} y_j &= y_j - \frac{s_j y_j^T y_j + y_j s_j^T y_j}{y_j^T s_j} + \Big(1 + \frac{y_j^T y_j}{s_j^T y_j}\Big)\frac{s_j s_j^T y_j}{s_j^T y_j} \\ &= y_j - \frac{s_j y_j^T y_j}{y_j^T s_j} - \frac{y_j s_j^T y_j}{y_j^T s_j} + \Big(1 + \frac{y_j^T y_j}{s_j^T y_j}\Big) s_j \\ &= y_j - \frac{s_j y_j^T y_j}{y_j^T s_j} - y_j + s_j + \frac{y_j^T y_j}{s_j^T y_j} s_j \\ &= y_j - \frac{y_j^T y_j}{y_j^T s_j} s_j - y_j + s_j + \frac{y_j^T y_j}{s_j^T y_j} s_j \\ &= s_j.\end{aligned} \tag{D–51}$$

If we replace the $\tilde{H}_j^{-1}$ in BFGS Eq. D–18 by an identity matrix I, then we will obtain Eq. D–50. So the conjugate gradient method with Eqs. D–50 is exactly the BFGS method when the approximation to the inverse Hessian is restarted as the identity matrix at every step. Shanno also called this algorithm the "memoryless" BFGS method.