UNIVERSITY OF TULSA
THE GRADUATE SCHOOL

CONDITIONING A THREE DIMENSIONAL RESERVOIR MODEL TO GAS PRODUCTION DATA

by
Rintu Kalita

A thesis submitted in partial fulfillment of
the requirements for the degree of Master of Science
in the Discipline of Petroleum Engineering
The Graduate School
The University of Tulsa
2000

UNIVERSITY OF TULSA

THE GRADUATE SCHOOL

CONDITIONING A THREE DIMENSIONAL RESERVOIR MODEL TO GAS PRODUCTION DATA

by

Rintu Kalita

A THESIS

APPROVED FOR THE DISCIPLINE OF

PETROLEUM ENGINEERING

By Thesis Committee

_____ Chairperson

_____

_____

ABSTRACT

Rintu Kalita (Master of Science in Petroleum Engineering)
CONDITIONING A THREE DIMENSIONAL RESERVOIR MODEL TO GAS PRODUCTION DATA
(134 pp.-Chapter V)
Directed by Dr. A. C. Reynolds

(249 words)

In this work, we apply inverse theory techniques to generate either an estimate or multiple realizations of rock property fields by conditioning a geostatistical model to well-test pressure data obtained for the single-phase flow of gas. In effect, the procedure for generating estimates and realizations is an automatic history matching procedure with the prior geostatistical model used as an regularization term. Much of the previous work done within TUPREP was based on conditioning reservoir models to pressure data assuming the single-phase flow of a slightly compressible liquid of constant compressibility and viscosity. The objectives of the current work are (i) to extend the methodology to condition a geostatistical model to pressure data obtained from a gas reservoir under single-phase flow conditions and (ii) to explore the applicability of the conjugate gradient (CG) method as an optimization technique.

The conjugate gradient method is applied to synthetic cases to generate realizations of porosity and permeability fields and its performance is compared with that of Gauss-Newton method. Both the methods result in similar final conditioned models. The convergence rate of the method CG is much slower than the Gauss-Newton method. However the computational cost per iteration is much less in CG

than in Gauss-Newton. The results suggest that the conjugate gradient method can be applied as the optimization algorithm for the problem of interest here, but further gain in efficiency may be necessary in order to routinely apply the method to large scale field problems.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would first like to express my sincere appreciation and gratitude to Dr. Albert C. Reynolds, Jr., Professor of Petroleum Engineering and Mathematical Sciences of the University of Tulsa, and Dr. Dean S. Oliver, Associate Professor of Petroleum Engineering of the University of Tulsa for their valuable assistance and guidance during the study. I also thank Dr. William A. Coberly, Chairman of Mathematical and Computer Science of the University of Tulsa for participating in my thesis committee and for his comments and suggestions. I would like to extend my thanks and appreciation to all other faculty members in the Petroleum Engineering for their guidance throughout my course of study as a TU graduate student.

Appreciation is also extended to all my friends and my graduate student colleagues for their valuable suggestion and encouragement. I gratefully acknowledge the financial support from TUPREP (Tulsa University Petroleum Reservoir Exploitation Projects) during my study.

CHAPTER I

INTRODUCTION

1.1 Background and Literature Review

In this work, we apply inverse theory techniques to generate realizations or estimates of rock property fields (reservoir simulator gridblock porosities and directional permeabilities) and well skin factors conditioned to geostatistical information and well-test pressure data for single-phase gas flow.

Our approach to the inverse problem is the probabilistic one that has been consistently used by TUPREP researchers and is grounded in Bayesian statistics. In this work, the prior model for the rock property fields is assumed to have a multivariate Gaussian probability density function (pdf). The a posteriori pdf is the conditional pdf for the model given the observed data and is the one that we use to construct either a conditional realization or an estimate of the rock property fields. The maximum a posteriori (MAP) estimate is the model which maximizes the a posteriori pdf and for this reason is sometimes referred to as the "most probable model." Generation of the MAP estimate requires the minimization of an objective function which includes a weighted sum of squares of data mismatch terms plus a regularization term which essentially represents the distance between a particular model and the mean of the prior model. Realizations are generated by minimizing an objective function which has a similar structure.

The procedure we use to generate realizations was introduced independently by Kitanidis [15] and Oliver et al. [19] and is referred to here as the randomized max-

imum likelihood method. Strictly speaking, the method yields a suite of realizations which represent a correct sampling of the a posteriori pdf if and only if the mapping from the model space to the data space represented by the forward model is linear; see Oliver [18] and Reynolds et al. [21].

In past work, the Gauss-Newton method combined with a restricted-step procedure has normally been used by TUPREP researchers to minimize the objective functions of interest; see, for example, Chu et al. [8] or He et al. [11]. Sometimes, the Gauss-Newton method converges to a local minimum which gives an unacceptable match of the pressure data unless the change in model parameters are damped at early iterations. In some cases, these local minima correspond to models which are much rougher than should be expected from the statistics of the prior model. To avoid these convergence problems, one may apply either an artificial damping of model changes at early times (Wu et al. [24]) or use the Levenberg-Marquardt algorithm (see, Bi et al. [2]).

For single-phase liquid flow, a highly efficient procedure developed by TUPREP researchers (He at al. [11]) has been used to calculate sensitivity coefficients. This procedure is effectively a three-dimensional extension of a method developed by Carter et al. [4]. For linear problems, the Carter method and the adjoint (or optimal control) method (Chen at al. [6] and Chavent et al. [5]) for computing sensitivity coefficients are identical; see Carter et al. [3]. However, for nonlinear problems Carter's technique is not applicable. Thus, we derived and implemented an adjoint procedure to calculate sensitivity coefficients. We have compared these results with corresponding sensitivities calculated from the finite-difference method and found that results from the two procedures are in excellent agreement.

For nonlinear problems, computation of sensitivity coefficients requires one adjoint solution for each individual observed data that will be used to condition the model. In the event, N_d observed data are used, we solve the adjoint system with N_d right hand sides. Roughly speaking, we expect the computational time required

in this procedure to be equivalent to $1 + (N_d/10)$ or more reservoir simulation runs; see Killough et al. [14]. If the number of data is large, the computational work required becomes significant. Although, it is clear that solving the adjoint system with multiple right-hand side would be a more efficient computational algorithm, our current code actually solves N_d right-hand sides to generate sensitivity coefficients for N_d data.

We have also implemented the adjoint method to calculate the sensitivity of the objective function to the rock property fields. This sensitivity coefficient is nothing more than the gradient of the objective function and its computation requires a single adjoint solution. If the conjugate gradient method is applied for optimization of the objective function, individual sensitivity coefficients are not needed; only the gradient of the objective function is required. Based on the work of Killough et al. [14], we assume the work required to compute sensitivities for N_d data is equivalent to $1 + (N_d/10)$ simulation runs. The computational effort required to compute the gradient of the objective function is roughly equivalent to one simulation run. It follows that if the Gauss-Newton method requires K_c iterations to obtain convergence, the conjugate gradient method must converge in fewer that $[1 + (N_d/10)] \times K_c$ iterations to be more efficient. If $N_d = 100$ and $K_c = 7$, this means the conjugate gradient method must converge in fewer than 77 iterations to be more efficient than the Gauss-Newton method. These, of course, are only rough comparisons.

Our initial investigation of the conjugate gradient method indicates that the method is promising especially if we can generate a better preconditioner than the one currently used. We hasten to add that the application of the conjugate gradient method to history matching by optimization is not new. For example, Makhlouf et al. [16] history matched two-phase (water-oil) data from a 450 cell reservoir model using a conjugate gradient method. The algorithm they applied converged relatively slowly, and for one problem, required 6400 CPU seconds on a CRAY X-MP/48 to obtain a history match.

CHAPTER II

MODEL DESCRIPTION AND INVERSE SOLUTION

2.1 The Prior Model

We assume that principal permeability directions coincide with the axes of the Cartesian coordinate system used so the permeability tensor is diagonal and involves only three directional permeabilities, k_x, k_y and k_z, which we assume have log normal probability distributions with known means and variances given by $\sigma^2_{k_x}, \sigma^2_{k_y}$ and $\sigma^2_{k_z}$ respectively. Porosity is assumed to be normal with known mean and variance given by σ^2_ϕ. Each rock property attribute is modeled as stationary Gaussian random function so that covariance functions are directly related to the variograms. Vertical permeability k_z may be either treated as a Gaussian random field or we may simply specify that $k_z = a\sqrt{k_x k_y}$, where the multiplier a is independent of position and is modeled as a Gaussian random variable with specified mean and variance. In the later case, a is a model parameter. The well skin factors are modeled as uncorrelated Gaussian random variables with specified means and variances available. Here, we specify only one skin factor per well, i.e, assume the skin factor at a well does not vary from layer to layer. The code we use, however, has the option of incorporating a different skin factor in each layer.

If m denotes the vector of model parameters, then in the most general three-dimensional anisotropic case, m is given by the M dimensional column vector

$$m = \begin{bmatrix} m_\phi \\ m_{k_x} \\ m_{k_y} \\ m_{k_z} \\ m_s \end{bmatrix}, \tag{2.1}$$

where m_ϕ is the column vector representing all the grid block porosities, m_{k_x} is the column vector containing all gridblock x-direction permeabilities and so on. In the most general case, the vector m_s includes the skin factors for all gridblocks penetrated by the wells. For the examples considered here, m_s includes a single skin factor for each well. Throughout, m_{prior} is the vector containing the prior estimates of these parameters.

The prior model is assumed to be described by a multivariate Gaussian pdf with prior covariance matrix C_M. In the anisotropic case, the prior covariance matrix is given by

$$C_M = \begin{bmatrix} C_\phi & C_{\phi k_x} & C_{\phi k_y} & C_{\phi k_z} & O \\ C_{k_x\phi} & C_{k_x} & C_{k_x k_y} & C_{k_x k_z} & O \\ C_{k_y\phi} & C_{k_y k_x} & C_{k_y} & C_{k_y k_z} & O \\ C_{k_z\phi} & C_{k_z k_x} & C_{k_z k_y} & C_{k_z} & O \\ O & O & O & O & C_s \end{bmatrix} \tag{2.2}$$

In Eq. 2.2, C_ϕ is the covariance matrix for gridblock porosities (derived from the porosity variogram); C_{k_x} is the covariance matrix for gridblock $\ln(k_x)$'s; C_{k_y} is the covariance matrix for gridblock $\ln(k_y)$'s and so on; $C_{\phi k_x}$ is the cross covariance matrix between porosity and $\ln(k_x)$ at the set of gridblocks; $C_{k_x\phi}$ is equal to the transpose of $C_{\phi k_x}$ and the other similar notations bear the similar meanings. Throughout, submatrices indicated with an O denote null matrices, i.e. matrices with all entries equal to zero. The cross covariance is obtained using the screening hypothesis of Xu et al.([25]).

2.2 A Posteriori Probability Density Function

The relationship between the vector d of calculated production (pressure or rate) data and the vector m of reservoir model parameters is written as:

$$d = g(m), \tag{2.3}$$

which represents generating d from the reservoir simulator for a given m. If the true reservoir could be described by discretization into gridblocks, and the entries of m were exactly equal to the true values of gridblock rock properties, then Eq. 2.3 would predict the observed production data provided the solution of the finite difference equations was not effected by truncation or round-off errors and data were measured exactly. However, the observed data will be corrupted by measurement errors. We let d_{obs} denote the vector of observed production data that will be used as conditioning data in the inverse problem. Then, assuming that the prior model is multivariate Gaussian and that data measurement errors are Gaussian with the data covariance matrix given by C_D, application of Bayes' theorem (see Tarantola [23]) indicates that the posteriori probability function is given by:

$$f(m) = \widehat{a} \exp\left(-\frac{1}{2}\Big[(m-m_{prior})^T C_M^{-1}(m-m_{prior}) + (g(m)-d_{obs})^T C_D^{-1}(g(m)-d_{obs})\Big]\right), \tag{2.4}$$

where $\widehat{a}$ is the normalizing constant.

If the pressure measurement errors are independent Gaussian random variables with zero mean and the variance of the jth measurement error is given by $\sigma_{d,j}^2$, then the data covariance matrix C_D is a $N_d \times N_d$ diagonal matrix with its j^{th} diagonal element equal to $\sigma_{d,j}^2$.

2.3 The Maximum A Posteriori Estimate

The maximum a posteriori (MAP) estimate refers to the model which maximizes the a posteriori probability density function (pdf) given by Eq. 2.4. For the

obvious reason, this m is sometimes referred to as the most probable model. The MAP estimate can be obtained by minimizing the following objective function:

$$O(m) = \frac{1}{2}\left[(m - m_{prior})^T C_M^{-1}(m - m_{prior}) + (g(m) - d_{obs})^T C_D^{-1}(g(m) - d_{obs})\right]. \tag{2.5}$$

If the predicted data are linearly related to the model, i.e.

$$d = Gm, \tag{2.6}$$

where G is the $N_d \times M$ sensitivity matrix, then Eq. 2.5 has a global minimum which can be constructed from the following analytical formula:

$$m_\infty = m_{prior} - H^{-1}\left[G^T C_D^{-1}(Gm_{prior} - d_{obs})\right], \tag{2.7}$$

where H is the $M \times M$ Hessian matrix given by

$$H = C_M^{-1} + G^T C_D^{-1} G. \tag{2.8}$$

In this work, the MAP is generated by minimizing the objective function of Eq. 2.5 using either the Gauss-Newton method with restricted step or the conjugate gradient method.

2.4 Sampling the Posteriori Probability Density Function

We need to generate a set of realizations of the model parameters by correctly sampling the a posteriori probability density function in order to characterize the uncertainty in reservoir performance. Here, we use the procedure suggested independently by Kitandis [15] and Oliver [19]. We refer to this procedure as the randomized maximum likelihood method. It can be shown that this method samples the a posteriori pdf correctly when the data is linearly related to the model. A proof of this fact is given in Oliver [18]. An alternate and more general proof which applies to a partially stochastic model is given in Reynolds [21]. This sampling procedure is described below.

First we generate an unconditional realization m_{uc} of the vector of the model parameters by sampling the prior pdf. We also generate an unconditional realization of the data d_{uc}. The m_{uc} can be generated by using Cholesky or square root decomposition of the prior covariance matrix or by sequential Gaussian cosimulation (see Gomez et al. [10]). The unconditional simulation of the data can be obtained from

$$d_{uc} = d_{obs} + C_D^{1/2} Z_D, \tag{2.9}$$

where Z_D is a vector of the independent standard normal deviates. A conditional realization m_c of the model parameters can be generated by minimizing

$$O_r(m) = -\frac{1}{2}\left[(m - m_{uc})^T C_M^{-1}(m - m_{uc}) + (g(m) - d_{uc})^T C_D^{-1}(g(m) - d_{uc})\right]. \tag{2.10}$$

In order to generate N_r realizations the procedure is repeated N_r times.

2.5 The Gauss-Newton Method

The Gauss-Newton method with restricted step has frequently been applied to generate the MAP estimate or realizations by minimizing the appropriate objective function; see, for example Chu and Reynolds [7], He at al. [13] and Reynolds et al. [20]. The gradient of the objective function $O(m)$ with respect to the model parameters is given by:

$$\nabla_m O(m) = G^T C_D^{-1}(g(m) - d_{obs}) + C_M^{-1}(m - m_{prior}), \tag{2.11}$$

and the Hessian for the Gauss-Newton is given by:

$$H = G^T C_D^{-1} G + C_M^{-1}, \tag{2.12}$$

where G is the sensitivity matrix. In this work, sensitivity coefficients are calculated using the adjoint method.

In the Gauss-Newton procedure, at the l^{th} iteration, we solve for δm^{l+1} by applying the following equation

$$H_l \delta m^{l+1} = -\nabla_l O, \tag{2.13}$$

where δm^{l+1} can be calculated from

$$\left(G_l^T C_D^{-1} G_l + C_M^{-1}\right) \delta m^{l+1} = -G_l^T C_D^{-1} \left(g(m^l) - d_{obs}\right) - C_M^{-1}(m^l - m_{prior}) \qquad (2.14)$$

and then compute the updated estimate of the model parameters from

$$m^{l+1} = m^l + \mu_l \delta m^{l+1}. \qquad (2.15)$$

where μ_l is calculated by the same restricted step procedure used in Chu et al. [7] and Reynolds et al. [20]. Fletcher et al. [9] provides a detailed discussion of the restricted step procedure. In the preceding equations l (used either as a subscript or a superscript) denotes the iteration index. The preceding two equations can be combined to obtain:

$$\begin{aligned} m^{l+1} &= m^l - \mu_l [G_l^T C_D^{-1} G_l + C_M^{-1}]^{-1} \times \\ &\quad [G_l^T C_D^{-1} (g(m_l) - d_{obs}) + C_M^{-1}(m^l - m_{prior})]. \end{aligned} \qquad (2.16)$$

Using matrix inverse lemmas on Eq. 2.14 (see Tarantola [23] or Chu and Reynolds [7]), the following form of the Gauss- Newton method can be obtained.

$$\begin{aligned} m^{l+1} &= \mu_l m_{prior} + (1 - \mu_l) m^l - \mu_l [C_M G_l^T (C_D + G_l C_M G_l^T)^{-1} \times \\ &\quad (g(m) - d_{obs} - G_l (m^l - m_{prior})]. \end{aligned} \qquad (2.17)$$

Eq. 2.16 and 2.17 are mathematically equivalent, but the computational time for the two schemes may be quite different. The inverse matrix on the right side of Eq. 2.16 is $M \times M$ where M is the number of model parameters. The inverse matrix on the right side of Eq. 2.17 is $N_d \times N_d$ where N_d is the number of conditioning data. Since we normally have $N_d < M$, Eq. 2.17 will normally be more efficient. For all problems considered in this work, we apply the form of the Gauss-Newton method given by 2.17. Also we are calculating the full C_M^{-1} while evaluating the objective function, instead of approximating it with a diagonal matrix whose elements are equal to the inverse of the diagonal terms of the C_M matrix.

2.6 The Conjugate Gradient Method

To generate the MAP estimate or a realization, the conjugate gradient method can be applied to minimize the appropriate objective function. Here, we use the preconditioned conjugate gradient (CG) algorithm for nonlinear problems with the preconditioning matrix at each iteration given by C_M^{-1}. The CG algorithm described below was taken directly from Schewchuk [22].

2.6.1 Notation

The following notations are used throughout:

k = CG iteration index.

i = Number of CG iterations since last restart. (Restarting means we ignore all the previous search directions and set the search direction to the steepest descent direction.)

ϵ = CG error tolerance for convergence.

r_k = residual at the kth iteration of the CG algorithm.

d_k = search direction at iteration k.

M_k = preconditioning matrix at the k iteration of the CG algorithm.

Initialization:

Let m^0 be the initial guess.

Set $i = 0, k = 0$.

Set $r^0 = -\nabla_m O(m^0)$, where the gradient of the objective function evaluated at m^0 is given by

$$\nabla_m O(m^0) = \nabla_m \left[\frac{1}{2} (g(m) - d_{obs})^T C_D^{-1} (g(m) - d_{obs}) \right] + C_M^{-1}(m - m_{prior}). \quad (2.18)$$

The gradient of the term within square brackets can be obtained by the adjoint method and the second part, $C_M^{-1}(m - m_{prior})$, is calculated directly and added to the first part.

Calculate the preconditioning matrix M_0 as an approximation to the Hessian. In this work M_0 is approximated by the inverse of the covariance matrix. So at all iterations, we set $M_0 = C_M^{-1}$ so that $M_0^{-1} = C_M$.

Set $s_0 = M_0^{-1} r_0 = C_M r_0$.

Set $d_0 = s_0$.

Set $\delta_{new} = r_0^T d_0 = r_0^T M_0^{-1} r_0$.

Set $\delta_0 = \delta_{new}$.

Start CG iteration

Start loop for k.

For $k = 0$ to $k_{\max}$ the following steps are repeated.

Check for convergence:

If $\delta_{new} < \epsilon^2 \delta_0$, STOP since the CG has converged.

Step size calculation:

Determine step size α by minimizing $g(\alpha) = O\left(m^k + \alpha d_k\right)$. This minimization can be done by using the Newton-Raphson algorithm to find a zero of $f(\alpha)$, where

$$f(\alpha) = \frac{dO(m^k + \alpha d_k)}{d\alpha} = \left(\nabla O(m^k + \alpha d_k)\right)^T d_k. \tag{2.19}$$

Also

$$f'(\alpha) = \frac{df(\alpha)}{d\alpha} = d_k^T \nabla \left[\left(\nabla O(m^k + \alpha d_k)\right)^T\right] d_k = d_k^T H(m^k + \alpha d_k) d_k. \tag{2.20}$$

Note since the Hessian is positive definite, $f(\alpha)$ is an increasing function of α and can have only one zero. Moreover this zero must be a minimum of $g(\alpha)$ since the second derivative of $g(\alpha)$ is positive. The Newton-Raphson iteration for finding a zero of $f(\alpha)$ is

$$\begin{aligned} \alpha_{j+1} &= \alpha_j - \frac{f(\alpha_j)}{f'(\alpha_j)} \\ &= \alpha_j - \frac{\left(\nabla O(m^k + \alpha d_k)\right)^T d_k}{d_k^T H(m^k + \alpha d_k) d_k.} \end{aligned} \tag{2.21}$$

The Newton-Raphson iteration is stopped when a suitable convergence criteria is reached. In our procedure we use the convergence criteria $(\Delta\alpha_j)^2 d_k^T d_k < \gamma^2$, where $\gamma = 10^{-6}$; or, as discussed later, we simply do one iteration.

The exact line search method given by Eq. 2.21 can be expensive because the evaluation of the term $d_k^T H(m^k + \alpha d_k) d_k$ requires one simulation run (shown later in this section). We attempt to gain efficiency by doing only one Newton-Raphson iteration. In that case, taking $\alpha_0 = 0$, we find,

$$\alpha = -\frac{\left(\nabla O(m^k)\right)^T d_k}{d_k^T H(m^k) d_k}. \tag{2.22}$$

If an inexact line search leads to the construction of a search direction that is not a descent direction (see Schewchuk [22]), we simply restart the CG iteration using steepest descent. To check if d_k is a descent direction, we evaluate $r_k^T d_k$ and restart if $r_k^T d_k \leq 0$.

Update parameters:

Set $m^{k+1} = m^k + \alpha_k d_k$.

Set $r^{k+1} = -\nabla_m O(m^{k+1})$.

Set $\delta_0 = \delta_{new} = r_k^T M_k^{-1} r_k$.

Set $\delta_{mid} = r_{k+1}^T s_k = r_{k+1}^T M_k^{-1} r_k$.

Calculate the new preconditioner M_{k+1}. (In our application, $M_{k+1} = C_M^{-1}$ at all iterations.)

Set $s_{k+1} = M_{k+1}^{-1} r_{k+1}$.

Set $\delta_{new} = r_{k+1}^T s_{k+1}$.

Calculation of β:

The parameter β can be calculated either by the Fletcher-Reeves formula or Polak-Ribiere method.

Fletcher-Reeves gives:

$$\beta_{k+1} = \frac{r_{k+1}^T s_{k+1}}{r_k^T s_k} = \frac{r_{k+1}^T M_{k+1}^{-1} r_{k+1}}{r_k^T M_k^{-1} r_k} = \frac{\delta_{new}}{\delta_0}. \tag{2.23}$$

Polak-Ribiere gives:

$$\beta_{k+1} = \frac{r_{k+1}^T(s_{k+1} - s_k)}{r_k^T s_k} = \frac{r_{k+1}^T M_{k+1}^{-1} r_{k+1} - r_{k+1}^T M_k^{-1} r_k}{r_k^T M_k^{-1} r_k} = \frac{\delta_{new} - \delta_{mid}}{\delta_0}. \tag{2.24}$$

We use the later formula since the convergence of Polak-Ribiere can be guaranteed by selecting $\beta = \max\{\beta^{PR}, 0\}$ (Schewchuk [22]). Using this value is equivalent to restarting CG when $\beta^{PR} < 0$. To restart the CG means we ignore the previous search directions, and begin CG iteration again where the first conjugate vector is in the direction of steepest descent.

Restart:

If $i = n$ (n is the maximum number of iteration allowed before restart) or $\beta_{k+1} \leq 0$,

Set $\beta_{k+1} = 0$

Set $d_{k+1} = s_{k+1}$ and set $i = 0$

Else

$d_{k+1} = s_{k+1} + \beta_{k+1} d_k$

Endif

Set $i = i + 1$

Finish CG iteration (End of loop on k)

Calculation of $d_k^T H(m^k) d_k$ in step size determination:

The above term can be approximated without actually calculating the Hessian. We know that at the kth iteration, the Hessian is given by

$$H_k = G_k^T C_D^{-1} G_k + C_M^{-1}, \tag{2.25}$$

so,

$$\begin{aligned} d_k^T H_k d_k &= d_k^T \left(G_k^T C_D^{-1} G_k + C_M^{-1} \right) d_k \\ &= d_k^T \left(G_k^T C_D^{-1} G_k \right) d_k + d_k^T C_M^{-1} d_k \\ &= \left(G_k d_k \right)^T C_D^{-1} (G_k d_k) + d_k^T C_M^{-1} d_k. \end{aligned} \tag{2.26}$$

Thus, we do not need to compute the sensitivity coefficient matrix G directly; we only need to compute Gd_k. Note that here d_k is the search direction at the kth conjugate gradient iteration. Also we know the relationship between data d and model parameters m given by $d = g(m)$. Thus, the sensitivity coefficient matrix G can be represented by

$$G = \{g_{i,j}\} \text{ where } g_{i,j} = \frac{\partial d_i}{\partial m_j} = \frac{\partial g_i}{\partial m_j},\ i = 1, ..., N_d \text{ and } j = 1, ..., M. \tag{2.27}$$

Now consider the directional derivative of g, i.e,

$$\left(\frac{dg}{d\alpha}\right)_{\alpha=0} = \left(\frac{dg(m + \alpha d_k)}{d\alpha}\right)_{\alpha=0}, \text{ where } d_k \text{ is the search direction.}$$

Letting u be the unit vector defined by $u = d_k/\|d_k\|$, it is well known that the ith component of this directional derivative is

$$\begin{aligned} \left(\frac{dg_i}{d\alpha}\right)_{\alpha=0} &= [\nabla g_i(m)]^T u \\ &= \frac{1}{\|d_k\|} [\nabla g_i(m)]^T d_k. \end{aligned} \tag{2.28}$$

The ith component of Gd_k is given by,

$$\begin{aligned} \Big[Gd_k\Big]_i &= \sum_{j=1}^{M} \frac{\partial g_i}{\partial m_j} d_{k,j} \\ &= [\nabla g_i(m)]^T d_k, \end{aligned} \tag{2.29}$$

where $d_{k,j}$ is the jth component of d_k. From Eqs. 2.28 and 2.29, it follows that

$$\begin{aligned} Gd_k &= \|d_k\| \Big(\frac{dg}{d\alpha}\Big)_{\alpha=0} \\ &\approx \|d_k\| \frac{g(m + \epsilon d_k) - g(m)}{\epsilon \|d_k\|} \\ &= \frac{g(m + \epsilon d_k) - g(m)}{\epsilon}, \end{aligned} \tag{2.30}$$

where ϵ is a small number. In our calculation, we choose ϵ based on the infinity norm of d_k so that ϵ satisfies $\epsilon\|d_k\|_\infty = 10^{-3}$. Once Gd_k is approximated as shown above, $d_k^T H(m^k) d_k$ can be determined from Eq. 2.26.

We can calculate $\left[\nabla O(m^k + \alpha d_k)\right]^T d_k$ in a similar fashion (or otherwise calculate it by adjoint method). Let $\widehat{m}^k = m^k + \alpha_j d_k$, we have

$$\left[\nabla O(\widehat{m}^k)\right]^T d_k = \sum_{j=1}^{M} \frac{\partial O(\widehat{m}^k)}{\partial m_j} d_{k,j}. \tag{2.31}$$

Again,

$$\left(\frac{dO(\widehat{m}^k)}{d\alpha}\right)_{\alpha=0} = \left[\nabla O(\widehat{m}^k)\right]^T \frac{d_k}{\|d_k\|}. \tag{2.32}$$

So

$$\begin{aligned} \left[\nabla O(\widehat{m}^k)\right]^T d_k &= \|d_k\| \left(\frac{dO(\widehat{m}^k)}{d\alpha}\right)_{\alpha=0} \\ &\approx \|d_k\| \frac{O(\widehat{m}^k + \epsilon d_k) - O(\widehat{m}^k)}{\epsilon \|d_k\|} \\ &= \frac{O(\widehat{m}^k + \epsilon d_k) - O(\widehat{m}^k)}{\epsilon}. \end{aligned} \tag{2.33}$$

The term ϵ is calculated in the same way as described previously. Note that if we do one inner iteration with the Newton-Raphson method, i.e, calculate α by Eq. 2.22 we do not have to calculate $\left[\nabla O(m^k)\right]^T d_k$ by Eq. 2.33, because $\left[\nabla O(m^k)\right]$ is already evaluated while constructing the residual r^k for the kth iteration of the conjugate gradient iteration. Eq 2.33 comes into play only when we do multiple inner iterations in the Newton-Raphson method.

Important Remarks:

1. As discussed previously, if we apply an exact line search method in the step size calculation, we must run the simulator once for each Newton-Raphson iteration (see Eq. 2.21). As this is computationally expensive, we conducted some experiments to investigate whether an exact line search is needed. We found that in the initial stages of the Newton-Raphson iteration, one Newton-Raphson iteration is sufficient to obtain a step size which reduces the objective function in the direction of d_k. For the cases we have considered, this did not result in additional restarts of the algorithm.

We also found that when the objective function is relatively flat, the zero of $f(\alpha) = dg(\alpha)/d\alpha$ computed by the Newton-Raphson iteration does not necessarily correspond to the minimum of $g(\alpha)$. At this point, we are not sure whether this discrepancy is due to inaccuracy in the computation of the gradient of the objective function using the adjoint method or other numerical errors. In any case, to gain efficiency, we do only one iteration to determine the step size. In the rare case that the new model $(m_k + \alpha d_k)$ gives an increase in the objective function (which can happen if $g(\alpha)$ is relatively flat in the neighborhood of the desired minimum), we continually cut the step size by a factor of 2 until we obtain a decrease in the objective function. It some cases this procedure cuts the step size so much that we make only a very small change in our estimate of m. However, this occurs only when the only current estimate of m approximately minimizes the objective function in the direction d_k.

2. In the basic CG algorithm, we restart the procedure with a steepest descent vector whenever $\beta < 0$ or the maximum allowable number of iteration without a restart is reached. However, if as in our case, an exact line search is not done, then at the next iteration, we may end up with a conjugate direction which is not a descent direction. The standard procedure for dealing with this problem is to restart conjugate gradient whenever $r_k^T d_k \leq 0$. This control and the control, $\beta < 0$, can be referred to as implicit control. In addition, we can explicitly specify the maximum number of iterations allowed before restarting the CG in an attempt to ensure the orthogonality of the search directions. This can be referred to as explicit control. We have experimented with the number of explicit controls and the results are discussed in Chapter IV.

3. We have also tried CG without preconditioning. Our results indicate that this procedure results in very localized changes in the model, i.e., only the rock properties at gridblocks containing wells are changed (see the results in Chapter IV, MAP estimation). Because such results are unreasonable, we abandoned the non-preconditioned

conjugate gradient methods without extensive testing. Using C_M^{-1} as a preconditioner effectively uses the correlation between gridblock parameters inherent in the covariance functions to cause a modification in the model parameters near the well. Also we have seen the change in the convergence rate using CG with preconditioning and without preconditioning (see results in Chapter IV). Preconditioning definitely improves the convergence rate, which perhaps can be further improved by choosing a more appropriate preconditioning matrix.

CHAPTER III

GENERATION OF SENSITIVITY COEFFICIENTS BY ADJOINT METHOD

In this section, we present the derivation of the adjoint method used to generate the sensitivity coefficients for wellbore pressure (in the case of constant rate production) and sensitivity of flow rate (in the case of constant pressure production) with respect to the rock property fields and skin factors. The single phase oil simulator developed by He et. al. [12] is adapted to take account of the gas phase and has been used to solve the forward problem in this work.

We present the derivation of the discrete adjoint equations. The adjoint variables obtained by solving this system of equations are used to calculate the sensitivity coefficients related to production data.

3.1 Flow Equation and Reservoir Simulator

The flow equation is written as,

$$C_1 \nabla.(\frac{1}{\mu_g B_g}[k]\,\nabla p(x,y,z,t)) = \frac{\phi c_t}{C_2 B_g}\frac{\partial p}{\partial t} + \widehat{q}(x,y,z,t). \tag{3.1}$$

The constants C_1 and C_2 are defined as:

$$C_1 = 1.127 \times 10^{-3}, \tag{3.2}$$

and

$$C_2 = 5.615. \tag{3.3}$$

The term $[k]$ denote the permeability tensor, B_g is the gas formation volume factor in RB/scf and μ_g is the gas viscosity in cp. B_g and μ_g are calculated based on standard

correlations for a particular pressure (see Appendix A). The term $\widehat{q}(x,y,z,t)$ is a source/sink term with units scf/ft^3-day, which is positive for a producing well and negative for an injection well. This term is nonzero only if the point (x,y,z) is intersected by the well where the rate is nonzero. We assume that the coordinate directions are aligned with the principal permeability directions so that

$$[k] = \begin{bmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & k_z \end{bmatrix}. \tag{3.4}$$

We assume a rectangular parallelopoid reservoir, i.e. Eq. 3.1 applies for all $t > 0$ on

$$\Omega = \{(x,y,z)|0 < x < L_x, 0 < y < L_y, 0 < z < L_z\}. \tag{3.5}$$

The boundary of Ω is denoted by $\partial\Omega$. We assume no flow boundary conditions. The initial conditions are given by

$$p(x,y,z,t) = p_0(x,y,z). \tag{3.6}$$

We discretize the region Ω into gridblocks using a standard block-centered grid and let the gridblock center be denoted by (x_i, y_j, z_k) where $i = 1, 2, ..., N_x$, $j = 1, 2, ..., N_y, k = 1, 2, ..., N_z$. Considering Eq. 3.1 at (x_i, y_j, z_k), we use a standard finite difference procedure to approximate the spatial derivatives and multiply the resulting equation by $\Delta x_i \Delta y_j \Delta z_k$ to obtain the following equation:

$$\begin{aligned} & T_{x,i+1/2,j,k}(t)(p_{i+1,j,k}(t) - p_{i,j,k}(t)) - T_{x,i-1/2,j,k}(t)(p_{i,j,k}(t) - p_{i-1,j,k}(t)) + \\ & T_{y,i,j+1/2,k}(t)(p_{i,j+1,k}(t) - p_{i,j,k}(t)) - T_{y,i,j-1/2,k}(t)(p_{i,j,k}(t) - p_{i,j-1/2,k}(t)) + \\ & T_{z,i,j,k+1/2}(t)(p_{i,j,k+1}(t) - p_{i,j,k}(t)) - T_{y,i,j,k-1/2}(t)(p_{i,j,k}(t) - p_{i,j,k-1/2}(t)) \\ = \; & (\frac{\Delta x_i \Delta y_j \Delta z_k \phi_{i,j,k} c_{t,i,j,k}}{C_2 B_{g,i,j,k}})(\frac{\partial p_{i,j,k}}{\partial t}) + q_{i,j,k}(t). \end{aligned} \tag{3.7}$$

The modified source/sink term is given by

$$q_{i,j,k}(t) = \Delta x_i \Delta y_j \Delta z_k \widehat{q}(x_i, y_j, z_k, t), \tag{3.8}$$

and has the units of scf/day. The terms $\Delta x_i, \Delta y_j, \Delta z_k$ are the dimensions of the grid block centered at (x_i, y_j, z_k). The x-direction boundaries of the gridblock are $x_{i+1/2,j,k}$ and $x_{i-1/2,j,k}$ so that $\Delta x_i = x_{i+1/2,j,k} - x_{i-1/2,j,k}$. Similar notation applies at the gridblock boundaries in the other directions. Transmissibilities are defined at the gridblock interfaces and are given below: The x-direction transmissibilities are

$$T_{x,i+1/2,j,k}(t) = \frac{C_1 \Delta y_j \Delta z_k k_{x,i+1/2,j,k}}{x_{i+1} - x_i} \left(\frac{1}{\mu_g(t) B_g(t)} \right)_{i+1/2,j,k}, \tag{3.9}$$

for all $i = 1, 2, ..., N_x - 1$. To incorporate no flow boundaries, we set

$$T_{x,1/2,j,k}(t) = T_{x,N_x+1/2,j,k}(t) = 0. \tag{3.10}$$

Similarly,

$$T_{y,i,j+1/2,k}(t) = \frac{C_1 \Delta x_i \Delta z_k k_{y,i,j+1/2,k}}{y_{j+1} - y_j} \left(\frac{1}{\mu_g(t) B_g(t)} \right)_{i,j+1/2,k}, \tag{3.11}$$

for all $j = 1, 2, ..., N_y - 1$. To incorporate no flow boundaries, we set

$$T_{y,i,1/2,k}(t) = T_{x,i,N_y+1/2,k}(t) = 0. \tag{3.12}$$

Finally,

$$T_{z,i,j,k+1/2}(t) = \frac{C_1 \Delta x_i \Delta y_j k_{z,i,j,k+1/2}}{z_{k+1} - z_k} \left(\frac{1}{\mu_g(t) B_g(t)} \right)_{i,j,k+1/2}, \tag{3.13}$$

for all $k = 1, 2, ..., N_z - 1$. To incorporate no flow boundaries, we set

$$T_{z,i,j,1/2}(t) = T_{z,i,j,N_z+1/2}(t) = 0. \tag{3.14}$$

Note that the transmissibilities are not independent of time since the terms contain pressure dependent terms. The pressure dependent parameters in the transmissibility term are calculated at the block interface corresponding to the interpolated pressure at the interface. The pressure at the block interface is evaluated by linearly interpolating the corresponding gridblock pressures.

Permeabilities at the grid block interfaces are calculated as the harmonic average of the grid block permeabilities with suitable modification for the boundary cases. So we have

$$k_{x,i+1/2,j,k} = \frac{(\Delta x_i + \Delta x_{i+1})k_{x,i,j,k}k_{x,i+1,j,k}}{\Delta x_i k_{x,i+1,j,k} + \Delta x_{i+1}k_{x,i,j,k}}, \tag{3.15}$$

for $i = 1, 2, ..., N_x - 1$, $j = 1, 2, ..., N_y$, $k = 1, 2, ..., N_z$,

$$k_{x,1/2,j,k} = k_{x,1,j,k} \text{ and } k_{x,N_x+1/2,j,k} = k_{x,N_x,j,k}. \tag{3.16}$$

Similarly,

$$k_{y,i,j+1/2,k} = \frac{(\Delta y_j + \Delta y_{j+1})k_{y,i,j,k}k_{y,i,j+1,k}}{\Delta y_j k_{y,i,j+1,k} + \Delta y_{j+1}k_{y,i,j,k}}, \tag{3.17}$$

for $j = 1, 2, ..., N_y - 1$, $i = 1, 2, ..., N_x$, $k = 1, 2, ..., N_z$,

$$k_{y,i,1/2,k} = k_{y,i,1,k} \text{ and } k_{y,i,N_y+1/2,k} = k_{y,i,N_y,k}. \tag{3.18}$$

Similarly,

$$k_{z,i,j,k+1/2} = \frac{(\Delta z_k + \Delta z_{k+1})k_{z,i,j,k}k_{z,i,j,k+1}}{\Delta z_k k_{z,i,j,k+1} + \Delta z_{k+1}k_{z,i,j,k}}, \tag{3.19}$$

for $k = 1, 2, ..., N_z - 1$, $i = 1, 2, ..., N_x$, $j = 1, 2, ..., N_y$,

$$k_{z,i,j,1/2} = k_{z,i,j,1} \text{ and } k_{z,i,j,N_z+1/2} = k_{z,i,jN_z}. \tag{3.20}$$

For a two-dimensional $x - y$ problem, we simply use one gridblock in the z direction and replace Δz_k by the reservoir thickness and set all the z direction permeabilities to zero.

3.2 Well Constraints

The simulator and the subsequent code for sensitivity coefficient generation, can handle single or multiple producing wells. We can also consider completely penetrating or partially penetrating wells. The wellbore constraints can be either constant flow rate, a sequence of constant flow rates or a constant flowing bottom hole pressure.

In the case of flow rate, we specify the total flow rate of the well. The relationship between a gridblock source or sink term, the gridblock pressure and the flowing bottom hole pressure is specified by Peaceman's equation. Specific source/sink terms in a grid block penetrated by a well must be computed from the specified well constraints. Assume a well is located in a grid block centered at (x_i, y_j, z_k). At time t, the flow rate $(q_{i,j,k}(t))$ for a well is related to grid block pressure, $p_{i,j,k}(t)$ and the flowing bottom hole pressure $p_{wf,i,j}(t)$ by

$$q_{i,j,k}(t) = WI_{i,j,k}(t)(p_{i,j,k}(t) - p_{wf,i,j}(t)). \tag{3.21}$$

The well index terms $WI_{i,j,k}(t)$ is defined as,

$$WI_{i,j,k}(t) = \frac{C_3 h\sqrt{k_{x,i,j,k}k_{y,i,j,k}}}{\ln(r_{o,i,j,k}/r_{w,i,j}) + s_{i,j,k}} \left(\frac{1}{\mu_{i,j,k}(t)B_{g,i,j,k}(t)}\right) \tag{3.22}$$

where

$$C_3 = (2\pi)1.127 \times 10^{-3}, \tag{3.23}$$

and

$$r_{o,i,j,k} = \frac{0.28073\Delta x_i\sqrt{1 + \frac{k_{x,i,j,k}\Delta y^2}{k_{y,i,j,k}\Delta x^2}}}{1 + \sqrt{\frac{k_{x,i,j,k}}{k_{y,i,j,k}}}}. \tag{3.24}$$

Here $r_{w,i,j}$ is the wellbore radius and $s_{i,j,k}$ is the skin factor of the well at model layer k. Note that the $WI_{i,j,k}(t)$, the well index term for the gridblock (x_i, y_j, z_k), is time dependent by its dependence on pressure, since it involves gas viscosity (μ_g) and formation volume factor (B_g). The pressure dependent terms are evaluated at the corresponding gridblock pressure.

3.3 Generation of Sensitivity Coefficients by Adjoint Method

Define the term

$$V_{i,j,k} = \frac{\Delta x_{i,j,k}\Delta y_{i,j,k}\Delta z_{i,j,k}\phi_{i,j,k}}{C_2}. \tag{3.25}$$

Using the above term and incorporating Peaceman's relation to account for the source/sink term, the semidiscrete flow equation can be written as

$$\begin{aligned} &T_{x,i+1/2,j,k}(t)(p_{i+1,j,k}(t)-p_{i,j,k}(t))-T_{x,i-1/2,j,k}(t)(p_{i,j,k}(t)-p_{i-1,j,k}(t))+\\ &T_{y,i,j+1/2,k}(t)(p_{i,j+1,k}(t)-p_{i,j,k}(t))-T_{y,i,j-1/2,k}(t)(p_{i,j,k}(t)-p_{i,j-1/2,k}(t))+\\ &T_{z,i,j,k+1/2}(t)(p_{i,j,k+1}(t)-p_{i,j,k}(t))-T_{y,i,j,k-1/2}(t)(p_{i,j,k}(t)-p_{i,j,k-1/2}(t))\\ &-WI_{i,j,k}(t)(p_{i,j,k}(t)-p_{wf,i,j}(t))=V_{i,j,k}\left(\frac{c_t}{B_g}\right)_{p(t)_{i,j,k}}\frac{\partial p(t)_{i,j,k}}{\partial t} \end{aligned} \tag{3.26}$$

which is equivalent to Eq. 3.7. For each gridblock (i,j,k) the total transmissibility can be defined as

$$\begin{aligned} T_{t,i,j,k}(t) &= T_{x,i+1/2,j,k}(t)+T_{x,i-1/2,j,k}(t)+T_{y,i,j+1/2,k}(t)+T_{y,i,j-1/2,k}(t)+\\ &\quad T_{z,i,j,k+1/2}(t)+T_{z,i,j,k-1/2}(t). \end{aligned} \tag{3.27}$$

Now Eq. 3.26 can be written as

$$\begin{aligned} &T_{z,i,j,k+1/2}(t)p_{i,j,k+1}(t)+T_{y,i,j+1/2,k}(t)p_{i,j+1,k}(t)+T_{x,i+1/2,j,k}(t)p_{i+1,j,k}(t)\\ &-\left[T_{t,i,j,k}(t)+WI_{i,j,k}(t)\right]p_{i,j,k}(t)+T_{x,i-1/2,j,k}(t)p_{i-1,j,k}(t)+T_{y,i,j-1/2,k}(t)p_{i,j-1,k}(t)\\ &+T_{z,i,j,k-1/2}(t)p_{i,j,k-1}(t)+WI_{i,j,k}(t)p_{wf,i,j}(t)=V_{i,j,k}\left(\frac{c_t}{B_g}\right)_{p(t)_{i,j,k}}\frac{\partial p(t)_{i,j,k}}{\partial t}. \end{aligned} \tag{3.28}$$

Assuming that the total flow rate is specified at each well as function of time, the rate at a well located at (x_i,y_j) and penetrating layers $k=l_1,....,l_2$, is given by

$$q_{i,j}(t)=\sum_{k=l_1}^{l_2}WI_{i,j,k}(t)\left[p_{i,j,k}(t)-p_{wf,i,j}(t)\right], \tag{3.29}$$

where the sum is all over the gridblocks penetrated by the well. In the simulator we approximate the time derivative to get the final system of finite difference equations. We let $\Delta t^n, n=1,...$ denote the time steps with $\Delta t^n=t^{n+1}-t^n$ and denote $t^0=0$. Also let

$$\widehat{V}^n_{i,j,k}=V_{i,j,k}\frac{1}{\Delta t^n}. \tag{3.30}$$

So the implicit scheme for the simulator equations can be written as

$$\begin{aligned} &T_{z,i,j,k+1/2}(t)p_{i,j,k+1}(t) + T_{y,i,j+1/2,k}(t)p_{i,j+1,k}(t) + T_{x,i+1/2,j,k}(t)p_{i+1,j,k}(t) \\ &-(T_{t,i,j,k}(t) + WI_{i,j,k}(t))p_{i,j,k}(t) + T_{x,i-1/2,j,k}(t)p_{i-1,j,k}(t) + T_{y,i,j-1/2,k}(t)p_{i,j-1,k}(t) \\ &+T_{z,i,j,k-1/2}(t)p_{i,j,k-1}(t) + WI_{i,j,k}(t)p_{wf,i,j}(t) = \widehat{V}^n_{i,j,k}\left(\frac{c_t}{B_g}\right)_{p_{i,j,k}}(p^n_{i,j,k} - p^{n-1}_{i,j,k}), \end{aligned} \tag{3.31}$$

and

$$q^n_{i,j} = \sum_{k=l_1}^{l_2} WI^n_{i,j,k}(p^n_{i,j,k} - p^n_{wf,i,j}). \tag{3.32}$$

Substituting $B^n_{i,j,k}$ for $\left(\frac{c_t}{B_g}\right)^n_{i,j,k}$ in Eq. 3.31, we can write the simulator equations as

$$A^n_{1,1}p^n + A^n_{1,2}p^n_{wf} = \widehat{V}^n\left[B^n\left(p^n - p^{n-1}\right)\right], \tag{3.33}$$

and

$$A^n_{2,1}p^n + A^n_{2,2}p^n_{wf} = Q^n. \tag{3.34}$$

where $A^n_{1,1}$ is the $N \times N$ matrix with diagonal elements given by terms $-(T_{t,i,j,k}(t) + WI_{i,j,k}(t))$ and the other 6 nonzero diagonals are given by the gridblock transmisibilities. Throughout, N_w denotes the number of wells and $N = N_x \times N_y \times N_z$ is the total number of gridblocks. The matrix $A^n_{1,2}$ is the $N \times N_w$ matrix whose entries are either zero or equal to the well index term. The matrix V^n is $N \times N$ diagonal matrix with the diagonal elements given by the $\widetilde{V}^n_{i,j,k}$ terms. The matrix $A^n_{2,1}$ is $N_w \times N$ and $A^n_{2,1} = (A^n_{1,2})^T$. The matrix $A^n_{2,2}$ is the $N_w \times N_w$ diagonal matrix with a particular diagonal entry corresponding to the sum of well index terms corresponding to a particular well. In Eq. 3.33, $B^n\left(p^n - p^{n-1}\right)$ is a $N \times 1$ dimensional matrix defined in Eq. 3.35 given below. The matrix B^n is a diagonal matrix of size $N \times N$ with each

diagonal element being equal to $\left(\frac{c_t^n}{B_g^n}\right)_{i,j,k}$. So we have

$$
\begin{aligned}
B^n(p^n - p^{n-1}) &= \begin{bmatrix} B^n(p^n - p^{n-1})_1 \\ B^n(p^n - p^{n-1})_2 \\ \vdots \\ B^n(p^n - p^{n-1})_N \end{bmatrix} \\
&= \begin{bmatrix} \left(\frac{c_t^n}{B_g^n}\right)_1 & & & \\ & \left(\frac{c_t^n}{B_g^n}\right)_2 & & \\ & & \ddots & \\ & & & \left(\frac{c_t^n}{B_g^n}\right)_N \end{bmatrix} \begin{bmatrix} (p^n - p^{n-1})_1 \\ (p^n - p^{n-1})_2 \\ \vdots \\ (p^n - p^{n-1})_N \end{bmatrix} . \qquad (3.35)
\end{aligned}
$$

3.4 The Adjoint Functional

We formulate the adjoint solution based on the Eqs. 3.33 and 3.34. We define vectors of gridblock porosities and permeabilities as follows.

$$k_x = \begin{bmatrix} k_{x1,} & k_{x1,} & \cdots & k_{xM} \end{bmatrix}^T , \qquad (3.36)$$

$$k_y = \begin{bmatrix} k_{y1,} & k_{y1,} & \cdots & k_{yM} \end{bmatrix}^T , \qquad (3.37)$$

$$k_z = \begin{bmatrix} k_{z1,} & k_{z1,} & \cdots & k_{zM} \end{bmatrix}^T , \qquad (3.38)$$

and

$$\phi = \begin{bmatrix} \phi_{1,} & \phi_{2,} & \cdots & \phi_M \end{bmatrix}^T . \qquad (3.39)$$

We can also define the skin variable vector by

$$s_{skin} = \begin{bmatrix} s_{skin1,} & s_{skin2,} & \cdots & s_{s_{skinM'}} \end{bmatrix}^T , \qquad (3.40)$$

where $M' = N_w \times N_z$ if we specify skin factors layerwise for each well or $M' = N_z$ if we specify one skin factor for each well. In the later case, the skin factors for each layer in a particular well are assumed to be equal.

We denote the differential of these variables by

$$dk_x = \left[\begin{array}{cccc} dk_{x1,} & dk_{x1,} & \cdots & dk_{xM} \end{array} \right]^T, \tag{3.41}$$

$$dk_y = \left[\begin{array}{cccc} dk_{y1,} & dk_{y1,} & \cdots & dk_{yM} \end{array} \right]^T, \tag{3.42}$$

$$dk_z = \left[\begin{array}{cccc} dk_{z1,} & dk_{z1,} & \cdots & dk_{zM} \end{array} \right]^T, \tag{3.43}$$

$$d\phi = \left[\begin{array}{cccc} d\phi_{1,} & d\phi_{2,} & \cdots & d\phi_{M} \end{array} \right]^T, \tag{3.44}$$

and

$$ds_{skin} = \left[\begin{array}{cccc} ds_{skin1,} & ds_{skin2,} & \cdots & ds_{skinM'} \end{array} \right]^T. \tag{3.45}$$

Assume we wish to compute the sensitivity of some real valued function g with respect to the vector m of model parameters. We also assume that the functional form of g is such that it depends only on p, p_{wf}, k_x, k_y, k_z and s_{skin}. So g can be defined as

$$g = g(p^1, ..., p^L, p^1_{wf}, ..., p^L_{wf}, k_x, k_y, k_z, \phi, s_{skin}). \tag{3.46}$$

The function is arbitrary, but the choice is dictated by the sensitivity coefficient we wish to compute. In fact, the adjoint method is formulated so that we can calculate the sensitivity of g. Our choice of g is dictated by our desire to find out sensitivity of $p_{wf}(t)$ and $q(t)$. For the case where the flow rate is specified, we have two types of adjoint equations; see Eqs. 3.33 and 3.34. So we define two sets of adjoint variables, one M dimensional and the other N_w dimensional, as follows:

$$\lambda^l_f = \left[\begin{array}{cccc} \lambda^l_{f1} & \lambda^l_{f2} & \cdots & \lambda^l_{fM} \end{array} \right]^T, \tag{3.47}$$

and

$$\lambda_s^l = \left[\begin{array}{cccc} \lambda_{s1}^l & \lambda_{s2}^l & \cdots & \lambda_{sM}^l \end{array} \right]^T . \tag{3.48}$$

We now adjoin Eqs. 3.33 and 3.34 to the function g to obtain the functional J given by

$$J = g + \sum_{n=1}^{L} \left[\left(\lambda_f^n\right)^T \left(A_{1,1}^n p^n + A_{1,2}^n p_{wf}^n - \widehat{V}^n \left(B^n p^n - B^n p^{n-1} \right) \right) + \left(\lambda_s^n\right)^T \left(A_{2,1}^n p^n + A_{2,2}^n p_{wf}^n - Q^n \right) \right]. \tag{3.49}$$

Given gridblock and wellbore pressures that satisfy finite difference equations, all the terms in the sum are zero for any values of the adjoint variables. For the sake of simplicity in calculating the total differential, we use the notation,

$$F_{1,1}^n = A_{1,1}^n p^n \quad \text{and} \quad F_{1,2}^n = A_{1,2}^n p_{wf}^n, \tag{3.50}$$

where $F_{1,1}^n$ and $F_{1,2}^n$ are both $(N \times 1)$ matrices. Similarly, let

$$F_{2,1}^n = A_{2,1}^n p^n \quad \text{and} \quad F_{2,2}^n = A_{2,2}^n p_{wf}^n, \tag{3.51}$$

where $F_{2,1}^n$ and $F_{2,2}^n$ are both $(N_w \times 1)$ matrices. So J can be rewritten as

$$J = g + \sum_{n=1}^{L} \left[\left(\lambda_f^n\right)^T \left(F_{1,1}^n + F_{1,2}^n \right) - \widehat{V}^n \left(B^n p^n - B^n p^{n-1} \right) + \left(\lambda_s^n\right)^T \left(F_{2,1}^n + F_{2,2}^n - Q^n \right) \right]. \tag{3.52}$$

From the point of view of computing the total differential, we consider the $F_{1,1}^n, F_{2,1}^n, F_{2,1}^n$, $F_{2,2}^n$ as functions of permeability, gridblock pressure, flowing wellbore pressure and skin factors. So the total differential of the i^{th} component of $F_{1,1}^n$ is given by

$$dF_{1,1,i}^n = \sum_{j=1}^{N} \left(\frac{\partial F_{1,1,i}^n}{\partial p_j^n} dp_j^n + \frac{\partial F_{1,1,i}^n}{\partial k_{x,j}} dk_{x,j} + \frac{\partial F_{1,1,i}^n}{\partial k_{y,j}} dk_{y,j} + \frac{\partial F_{1,1,i}^n}{\partial k_{z,j}} dk_{z,j} + \frac{\partial F_{1,1,i}^n}{\partial s_{skin,j}} ds_{skin,j} \right). \tag{3.53}$$

Defining the gradient operator in the standard way, we have

$$\nabla_{p^n} F^n_{1,1,i} = \begin{bmatrix} \frac{\partial F^n_{1,1,i}}{\partial p^n_1} \\ \frac{\partial F^n_{1,1,i}}{\partial p^n_2} \\ \vdots \\ \frac{\partial F^n_{1,1,i}}{\partial p^n_N} \end{bmatrix}. \tag{3.54}$$

Similarly,

$$\nabla_{k_x} F^n_{1,1,i} = \begin{bmatrix} \frac{\partial F^n_{1,1,i}}{\partial k_{x,1}} \\ \frac{\partial F^n_{1,1,i}}{\partial k_{x,2}} \\ \vdots \\ \frac{\partial F^n_{1,1,i}}{\partial k_{x,N}} \end{bmatrix}. \tag{3.55}$$

Similar notation can be used with the N-dimensional vectors k_y, k_z, and the M'-dimensional vector s_{skin}. So we can write Eq. 3.53 as

$$\begin{aligned} dF^n_{1,1,i} &= (\nabla_{p^n} F^n_{1,1,i})^T dp^n + (\nabla_{k_x} F^n_{1,1,i})^T dk_x + (\nabla_{k_y} F^n_{1,1,i})^T dk_y + \\ &\quad (\nabla_{k_z} F^n_{1,1,i})^T dk_z + (\nabla_{s_{skin}} F^n_{1,1,i})^T ds_{skin}, \end{aligned} \tag{3.56}$$

where dk_x, dk_y, dk_z and ds_{skin} are given by Eq. 3.41, 3.42, 3.43 and 3.45 respectively and

$$dp^n = \begin{bmatrix} dp^n_1 & dp^n_2 & \cdots & dp^n_N \end{bmatrix}. \tag{3.57}$$

Using gradient notation, we can also see that

$$
\begin{aligned}
\nabla_{k_x}[(F^n_{1,1})^T] &= \begin{bmatrix} \frac{\partial}{\partial k_{x,1}} \\ \frac{\partial}{\partial k_{x,2}} \\ \vdots \\ \frac{\partial}{\partial k_{x,N}} \end{bmatrix} [(F^n_{1,1})^T] = \begin{bmatrix} \frac{\partial}{\partial k_{x,1}} \\ \frac{\partial}{\partial k_{x,2}} \\ \vdots \\ \frac{\partial}{\partial k_{x,N}} \end{bmatrix} \begin{bmatrix} (F^n_{1,1,1}) & (F^n_{1,1,2}) & \dots & (F^n_{1,1,N}) \end{bmatrix}^T \\
&= \begin{bmatrix} \nabla_{k_x} F^n_{1,1,1} & \nabla_{k_x} F^n_{1,1,2} & \cdots & \nabla_{k_x} F^n_{1,1,N} \end{bmatrix} \\
&= \begin{bmatrix} \frac{\partial F^n_{1,1,1}}{\partial k_{x,1}} & \frac{\partial F^n_{1,1,2}}{\partial k_{x,1}} & \dots & \frac{\partial F^n_{1,1,N}}{\partial k_{x,1}} \\ \frac{\partial F^n_{1,1,1}}{\partial k_{x,2}} & \frac{\partial F^n_{1,1,2}}{\partial k_{x,2}} & \dots & \frac{\partial F^n_{1,1,N}}{\partial k_{x,2}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F^n_{1,1,1}}{\partial k_{x,N}} & \frac{\partial F^n_{1,1,2}}{\partial k_{x,N}} & \dots & \frac{\partial F^n_{1,1,N}}{\partial k_{x,N}} \end{bmatrix}. \qquad (3.58)
\end{aligned}
$$

A similar expression can be derived for the gradient of $F^n_{1,1}$ with respect to other variables. Then the total differential of $F^n_{1,1}$can be written as

$$dF^n_{1,1} = \begin{bmatrix} dF^n_{1,1,1} \\ dF^n_{1,1,2} \\ \vdots \\ dF^n_{1,1,N} \end{bmatrix}$$

$$= \begin{bmatrix} (\nabla_{p^n} F^n_{1,1,1})^T dp^n + (\nabla_{k_x} F^n_{1,1,1})^T dk_x + (\nabla_{k_y} F^n_{1,1,1})^T dk_y \\ (\nabla_{p^n} F^n_{1,1,2})^T dp^n + (\nabla_{k_x} F^n_{1,1,2})^T dk_x + (\nabla_{k_y} F^n_{1,1,2})^T dk_y \\ \vdots \\ (\nabla_{p^n} F^n_{1,1,N})^T dp^n + (\nabla_{k_x} F^n_{1,1,N})^T dk_x + (\nabla_{k_y} F^n_{1,1,N})^T dk_y \end{bmatrix} +$$

$$\begin{bmatrix} (\nabla_{k_z} F^n_{1,1,1})^T dk_z + (\nabla_{s_{skin}} F^n_{1,1,1})^T ds_{skin} \\ (\nabla_{k_z} F^n_{1,1,2})^T dk_z + (\nabla_{s_{skin}} F^n_{1,1,2})^T ds_{skin} \\ \vdots \\ (\nabla_{k_z} F^n_{1,1,N})^T dk_z + (\nabla_{s_{skin}} F^n_{1,1,N})^T ds_{skin} \end{bmatrix}$$

$$= \begin{bmatrix} (\nabla_{p^n} F^n_{1,1,1})^T \\ (\nabla_{p^n} F^n_{1,1,2})^T \\ \vdots \\ (\nabla_{p^n} F^n_{1,1,N})^T \end{bmatrix} dp^n + \begin{bmatrix} (\nabla_{k_x} F^n_{1,1,1})^T \\ (\nabla_{k_x} F^n_{1,1,2})^T \\ \vdots \\ (\nabla_{k_x} F^n_{1,1,N})^T \end{bmatrix} dk_x + \begin{bmatrix} (\nabla_{k_y} F^n_{1,1,1})^T \\ (\nabla_{k_y} F^n_{1,1,2})^T \\ \vdots \\ (\nabla_{k_y} F^n_{1,1,N})^T \end{bmatrix} dk_y +$$

$$\begin{bmatrix} (\nabla_{k_z} F^n_{1,1,1})^T \\ (\nabla_{k_z} F^n_{1,1,2})^T \\ \vdots \\ (\nabla_{k_z} F^n_{1,1,N})^T \end{bmatrix} dk_z + \begin{bmatrix} (\nabla_{s_{skin}} F^n_{1,1,1})^T \\ (\nabla_{s_{skin}} F^n_{1,1,2})^T \\ \vdots \\ (\nabla_{s_{skin}} F^n_{1,1,N})^T \end{bmatrix} ds_{skin}$$

$$= \begin{bmatrix} (\nabla_{p^n} F^n_{1,1,1})^T & \cdots & (\nabla_{p^n} F^n_{1,1,N})^T \end{bmatrix}^T dp^n + \begin{bmatrix} (\nabla_{k_x} F^n_{1,1,1})^T & \cdots & (\nabla_{k_x} F^n_{1,1,N})^T \end{bmatrix}^T dk_x +$$

$$\begin{bmatrix} (\nabla_{k_y} F^n_{1,1,1})^T & \cdots & (\nabla_{k_y} F^n_{1,1,N})^T \end{bmatrix}^T dk_y + \begin{bmatrix} (\nabla_{k_z} F^n_{1,1,1})^T & \cdots & (\nabla_{k_z} F^n_{1,1,N})^T \end{bmatrix}^T dk_z +$$

$$\begin{bmatrix} (\nabla_{s_{skin}} F^n_{1,1,1})^T & \cdots & (\nabla_{s_{skin}} F^n_{1,1,N})^T \end{bmatrix}^T ds_{skin},$$

(3.59)

or,

$$dF^n_{1,1} = (\nabla_{p^n}(F^n_{1,1})^T)^T dp^n + (\nabla_{k_x}(F^n_{1,1})^T)^T dk_x + (\nabla_{k_y}(F^n_{1,1})^T)^T dk_y + (\nabla_{k_z}(F^n_{1,1})^T)^T dk_z + (\nabla_{s_{skin}}(F^n_{1,1})^T)^T ds_{skin}. \quad (3.60)$$

Similarly $F^n_{1,2}$ is a function of $p^n, p^n_{wf}, k_x, k_y, s_{skin}$ and hence we can find the total derivative as

$$dF^n_{1,2} = (\nabla_{p^n}(F^n_{1,2})^T)^T dp^n + (\nabla_{p^n_{wf}}(F^n_{1,2})^T)^T dp^n_{wf} + (\nabla_{k_x}(F^n_{1,2})^T)^T dk_x + (\nabla_{k_y}(F^n_{1,2})^T)^T dk_y + (\nabla_{s_{skin}}(F^n_{1,2})^T)^T ds_{skin}. \quad (3.61)$$

The vector $F^n_{2,1}$ is a function only of p^n, k_x, k_y and s_{skin} and the total differential can be written as

$$dF^n_{2,1} = (\nabla_{p^n}(F^n_{2,1})^T)^T dp^n + (\nabla_{k_x}(F^n_{2,1})^T)^T dk_x + (\nabla_{k_y}(F^n_{2,1})^T)^T dk_y + (\nabla_{s_{skin}}(F^n_{2,1})^T)^T ds_{skin}. \quad (3.62)$$

The vector $F^n_{2,2}$ is a function only of p^n, p^n_{wf}, k_x, k_y and s_{skin} and the total derivative can be written as

$$dF^n_{2,2} = (\nabla_{p^n}(F^n_{2,2})^T)^T dp^n + (\nabla_{p^n_{wf}}(F^n_{2,2})^T)^T dp^n_{wf} + (\nabla_{k_x}(F^n_{2,2})^T)^T dk_x + (\nabla_{k_y}(F^n_{2,2})^T)^T dk_y + (\nabla_{s_{skin}}(F^n_{2,2})^T)^T ds_{skin}. \quad (3.63)$$

Using the above notation, the total differential of J given by 3.52 can be written as,

$$dJ = dg + \sum_{n=1}^{L} \Big\{ (\lambda_f^n)^T \Big[(\nabla_{p^n} (F_{1,1}^n)^T)^T dp^n + (\nabla_{k_x} (F_{1,1}^n)^T)^T dk_x + (\nabla_{k_y} (F_{1,1}^n)^T)^T dk_y +$$
$$(\nabla_{k_z} (F_{1,1}^n)^T)^T dk_z + (\nabla_{s_{skin}} (F_{1,1}^n)^T)^T ds_{skin} +$$
$$(\nabla_{p^n} (F_{1,2}^n)^T)^T dp^n + (\nabla_{p_{wf}^n} (F_{1,2}^n)^T)^T dp_{wf}^n + (\nabla_{k_x} (F_{1,2}^n)^T)^T dk_x +$$
$$(\nabla_{k_y} (F_{1,2}^n)^T)^T dk_y + (\nabla_{s_{skin}} (F_{1,2}^n)^T)^T ds_{skin} -$$
$$\left(\nabla_{p^n} \left[\widehat{V}^n (B^n \left[p^n - p^{n-1} \right] \right]^T \right)^T dp^n + \left(\nabla_{p^{n-1}} \left[\widehat{V}^n \left(B^n p^{n-1} \right) \right]^T \right)^T dp^{n-1} -$$
$$\left(\nabla_{\phi} \left[\widehat{V}^n \left(B^n p^n - B^n p^{n-1} \right) \right]^T \right)^T d\phi \Big]$$
$$+ (\lambda_s^n)^T \Big[(\nabla_{p^n} (F_{2,1}^n)^T)^T dp^n + (\nabla_{k_x} (F_{2,1}^n)^T)^T dk_x + (\nabla_{k_y} (F_{2,1}^n)^T)^T dk_y +$$
$$(\nabla_{s_{skin}} (F_{2,1}^n)^T)^T ds_{skin} + (\nabla_{p^n} (F_{2,2}^n)^T)^T dp^n + (\nabla_{p_{wf}^n} (F_{2,2}^n)^T)^T dp_{wf}^n +$$
$$(\nabla_{k_x} (F_{2,2}^n)^T)^T dk_x + (\nabla_{k_y} (F_{2,2}^n)^T)^T dk_y + (\nabla_{s_{skin}} (F_{2,2}^n)^T)^T ds_{skin} \Big] \Big\} \tag{3.64}$$

In this case $dQ = 0$ since we are considering constant rate production.

After rearranging the terms, we get

$$dJ = dg + \sum_{n=1}^{L} \Bigg\{ (\lambda_f^n)^T \Big[(\nabla_{p^n} (F_{1,1}^n)^T)^T dp^n + (\nabla_{p^n} (F_{1,2}^n)^T)^T dp^n -$$
$$\left(\nabla_{p^n} \left(\widehat{V}^n \left(B^n (p^n - p^{n-1}) \right) \right)^T \right)^T dp^n + \left(\nabla_{p^{n-1}} \left(\widehat{V}^n \left(B^n p^n \right) \right)^T \right)^T dp^{n-1} \Big] +$$
$$(\lambda_s^n)^T \left[(\nabla_{p^n} (F_{2,1}^n)^T)^T dp^n + (\nabla_{p^n} (F_{2,2}^n)^T)^T dp^n \right] +$$
$$(\lambda_f^n)^T \left[(\nabla_{p_{wf}^n} (F_{1,2}^n)^T)^T dp_{wf}^n \right] + (\lambda_s^n)^T \left[(\nabla_{p_{wf}^n} (F_{2,2}^n)^T)^T dp_{wf}^n \right] +$$
$$(\lambda_f^n)^T \left[(\nabla_{k_x} (F_{1,1}^n)^T)^T dk_x + (\nabla_{k_x} (F_{1,2}^n)^T)^T dk_x \right] +$$
$$(\lambda_s^n)^T \left[(\nabla_{k_x} (F_{2,1}^n)^T)^T dk_x + (\nabla_{k_x} (F_{2,2}^n)^T)^T dk_x \right] +$$
$$(\lambda_f^n)^T \left[(\nabla_{k_y} (F_{1,1}^n)^T)^T dk_y + (\nabla_{k_y} (F_{1,2}^n)^T)^T dk_y \right] +$$
$$(\lambda_s^n)^T \left[(\nabla_{k_y} (F_{2,1}^n)^T)^T dk_y + (\nabla_{k_y} (F_{2,2}^n)^T)^T dk_y \right] +$$
$$(\lambda_f^n)^T \left[(\nabla_{s_{skin}} (F_{1,1}^n)^T)^T ds_{skin} + (\nabla_{s_{skin}} (F_{1,2}^n)^T)^T ds_{skin} \right] +$$
$$(\lambda_s^n)^T \left[(\nabla_{s_{skin}} (F_{2,1}^n)^T)^T ds_{skin} + (\nabla s_{skin} (F_{2,2}^n)^T)^T ds_{skin} \right] +$$
$$(\lambda_f^n)^T \left[(\nabla_{k_z} (F_{1,1}^n)^T)^T dk_z \right] - (\lambda_f^n)^T \left[\left(\nabla_\phi \left(\widehat{V}^n \left(B^n p^n - B^n p^{n-1} \right) \right)^T \right)^T d\phi \right] \Bigg\} \quad (3.65)$$

Also in all cases of interest to us, the function g will be chosen so that it depends explicitly on one or more of the p^n and p_{wf}^n vectors, $n = 0, 1, ..., L$ and possibly k_x, k_y, k_z, ϕ and s_{skin}. Hence the total differential of g can be written as,

$$dg = \sum_{n=1}^{L} [(\nabla_{p^n} g)^T dp^n + (\nabla_{p_{wf}^n} g)^T dp_{wf}^n] + [\nabla_{k_x} g]^T dk_x + [\nabla_{k_y} g]^T dk_y +$$
$$[\nabla_{k_z} g]^T dk_z + [\nabla_\phi g]^T d\phi + [\nabla s_{skin} g]^T ds_{skin}. \quad (3.66)$$

Using Eq. 3.66 in Eq. 3.65 and rearranging the terms, we get

$$
\begin{aligned}
dJ = \sum_{n=1}^{L} \Bigg[\Big\{ (\lambda_f^n)^T [(\nabla_{p^n}(F_{1,1}^n)^T)^T + (\nabla_{p^n}(F_{1,2}^n)^T)^T - \left(\nabla_{p^n} \left[\widehat{V}^n \left(B^n (p^n - p^{n-1}) \right) \right]^T \right)^T] + \\
(\lambda_s^n)^T [(\nabla_{p^n}(F_{2,1}^n)^T)^T + (\nabla_{p^n}(F_{2,2}^n)^T)^T] + (\nabla_{p^n} g)^T \Big\} dp^n + \\
(\lambda_f^n)^T \left(\nabla_{p^{n-1}} \left[\widehat{V}^n \left(B^n p^{n-1} \right) \right]^T \right)^T dp^{n-1} \Bigg] + \\
\sum_{n=1}^{L} \Bigg[\left[(\lambda_f^n)^T \left[(\nabla_{p_{wf}^n}(F_{1,2}^n)^T)^T \right] + (\lambda_s^n)^T \left[(\nabla_{p_{wf}^n}(F_{2,2}^n)^T)^T \right] + (\nabla_{p_{wf}^n} g)^T \right] dp_{wf}^n \Bigg] + \\
\sum_{n=1}^{L} \Bigg[\Big[(\lambda_f^n)^T \left[(\nabla_{k_x}(F_{1,1}^n)^T)^T + (\nabla_{k_x}(F_{1,2}^n)^T)^T \right] + \\
(\lambda_s^n)^T \left[(\nabla_{k_x}(F_{2,1}^n)^T)^T + (\nabla_{k_x}(F_{2,2}^n)^T)^T \right] + (\nabla_{k_x} g)^T \Big] dk_x \Bigg] + \\
\sum_{n=1}^{L} \Bigg[\Big[(\lambda_f^n)^T \left[(\nabla_{k_y}(F_{1,1}^n)^T)^T + (\nabla_{k_y}(F_{1,2}^n)^T)^T \right] + \\
(\lambda_s^n)^T \left[(\nabla_{k_y}(F_{2,1}^n)^T)^T + (\nabla_{k_y}(F_{2,2}^n)^T)^T \right] + (\nabla_{k_y} g)^T \Big] dk_y \Bigg] + \\
\sum_{n=1}^{L} \Big[(\lambda_f^n)^T \left[(\nabla_{k_z}(F_{1,1}^n)^T)^T \right] + [\nabla_{k_z} g]^T \Big] dk_z - \\
\sum_{n=1}^{L} \Bigg[\left\{ (\lambda_f^n)^T \left[\left(\nabla_\phi \left[\widehat{V}^n \left(B^n p^n - B^n p^{n-1} \right) \right]^T \right)^T \right] + [\nabla_\phi g]^T \right\} d\phi \Bigg] \\
+ \sum_{n=1}^{L} \Bigg[\Big[(\lambda_f^n)^T \left[(\nabla_{s_{skin}}(F_{1,1}^n)^T)^T + (\nabla_{s_{skin}}(F_{1,2}^n)^T)^T \right] + \\
(\lambda_s^n)^T \left[(\nabla_{s_{skin}}(F_{2,1}^n)^T)^T + (\nabla_{s_{skin}}(F_{2,2}^n)^T)^T \right] + (\nabla_{s_{skin}} g)^T \Big] ds_{skin} \Bigg]. \qquad (3.67)
\end{aligned}
$$

3.5 Discrete Adjoint Equations

Next we choose the adjoint variables to ensure that the coefficients multiplying by dp^n and dp_{wf}^n in Eq. 3.67 vanishes for $n = 1, 2, ..., L$. Considering only the dp and dp_{wf} part of dJ we have,

$$
\begin{aligned}
d\widehat{J} \;=\; & \sum_{n=1}^{L}\Big[\Big\{(\lambda_f^n)^T[(\nabla_{p^n}(F_{1,1}^n)^T)^T+(\nabla_{p^n}(F_{1,2}^n)^T)^T-\left(\nabla_{p^n}\left[\widehat{V}^n\left(B^n(p^n-p^{n-1})\right)\right]^T\right)^T]+\\
& (\lambda_s^n)^T[(\nabla_{p^n}(F_{2,1}^n)^T)^T+(\nabla_{p^n}(F_{2,2}^n)^T)^T]+(\nabla_{p^n}g)^T\Big\}dp^n+\\
& (\lambda_f^n)^T\left(\nabla_{p^{n-1}}\left[\widehat{V}^n\left(B^np^{n-1}\right)\right]^T\right)^T dp^{n-1}\Big]\\
& +\sum_{n=1}^{L}\Big[\left[(\lambda_f^n)^T[(\nabla_{p_{wf}^n}(F_{1,2}^n)^T)^T]+(\lambda_s^n)^T[(\nabla_{p_{wf}^n}(F_{2,2}^n)^T)^T]+(\nabla_{p_{wf}^n}g)^T\right]dp_{wf}^n\Big]. \quad (3.68)
\end{aligned}
$$

Since the initial conditions are independent of model parameters $dp^0=0$. Thus,

$$
\begin{aligned}
& \sum_{n=1}^{L}\left[(\lambda_f^n)^T\left(\nabla_{p^{n-1}}\left[\widehat{V}^n\left(B^np^{n-1}\right)\right]^T\right)^T dp^{n-1}\right]\\
& =\sum_{n=0}^{L-1}\left[(\lambda_f^{n+1})^T\left(\nabla_{p^n}\left[\widehat{V}^{n+1}\left(B^{n+1}p^n\right)\right]^T\right)^T dp^n\right]\\
=\sum_{n=1}^{L-1}\Big[(\lambda_f^{n+1})^T & \left(\nabla_{pn}\left[\widehat{V}^{n+1}\left(B^{n+1}p^n\right)\right]^T\right)^T dp^n\Big]+\left[(\lambda_f^1)^T\left(\nabla_{p^0}\left[\widehat{V}^1\left(B^1p^0\right)\right]^T\right)^T dp^0\right]\\
& =\sum_{n=1}^{L-1}\left[(\lambda_f^{n+1})^T\left(\nabla_{p^n}\left[\widehat{V}^{n+1}\left(B^{n+1}p^n\right)\right]^T\right)^T dp^n\right]. \quad (3.69)
\end{aligned}
$$

Substituting Eq. 3.69, into Eq. 3.68 and rearranging we obtain,

$$
\begin{aligned}
d\widehat{J} \;=\; & \sum_{n=1}^{L}\Bigg[\Big\{(\lambda_f^n)^T\left[(\nabla_{p^n}(F_{1,1}^n)^T)^T+(\nabla_{p^n}(F_{1,2}^n)^T)^T-\left(\nabla_{p^n}\left(\widehat{V}^n\left(B^n(p^n-p^{n-1})\right)\right)^T\right)^T\right]+\\
& (\lambda_s^n)^T\left[(\nabla_{p^n}(F_{2,1}^n)^T)^T+(\nabla_{p^n}(F_{2,2}^n)^T)^T\right]+(\nabla_{p^n}g)^T\Big\}dp^n\Bigg]+\\
& \sum_{n=1}^{L-1}\left[(\lambda_f^{n+1})^T\left(\nabla_{p^n}\left(\widehat{V}^{n+1}\left(B^{n+1}p^n\right)\right)^T\right)^T dp^n\right]+\\
& \sum_{n=1}^{L}\left[(\lambda_f^n)^T[(\nabla_{p_{wf}^n}(F_{1,2}^n)^T)^T]+(\lambda_s^n)^T[(\nabla_{p_{wf}^n}(F_{2,2}^n)^T)^T]+(\nabla_{p_{wf}^n}g)^T\right]dp_{wf}^n. \quad (3.70)
\end{aligned}
$$

If we define $\lambda_f^{L+1} = 0$, then the above equation can be written as,

$$\begin{aligned} d\widehat{J} \;=\; & \sum_{n=1}^{L} \Bigg[\Big\{ (\lambda_f^n)^T \left[(\nabla_{p^n}(F_{1,1}^n)^T)^T + (\nabla_{p^n}(F_{1,2}^n)^T)^T - \left(\nabla_{p^n} \left(\widehat{V}^n \left(B^n p^n \right) \right)^T \right)^T \right] + \\ & (\lambda_s^n)^T \left[(\nabla_{p^n}(F_{2,1}^n)^T)^T + (\nabla_{p^n}(F_{2,2}^n)^T)^T \right] + (\nabla_{p^n} g)^T \Big\} dp^n \Bigg] + \\ & \sum_{n=1}^{L} \left[(\lambda_f^{n+1})^T \left(\nabla_{p^n} \left(\widehat{V}^{n+1} \left(B^{n+1} p^l \right) \right)^T \right)^T dp^n \right] + \\ & \sum_{n=1}^{L} \Bigg[\Big[(\lambda_f^n)^T [(\nabla_{p_{wf}^n}(F_{1,2}^n)^T)^T] + (\lambda_s^n)^T [(\nabla_{p_{wf}^n}(F_{2,2}^n)^T)^T] + \\ & (\nabla_{p_{wf}^n} g)^T \Big] dp_{wf}^n \Bigg]. \end{aligned} \tag{3.71}$$

The adjoint equations are obtained by setting the coefficients of dp and dp_{wf} to zero. Thus we must have,

$$\begin{aligned} (\lambda_f^n)^T \left[(\nabla_{p^n}(F_{1,1}^n)^T)^T + (\nabla_{p^n}(F_{1,2}^n)^T)^T - (\nabla_{p^n}(\widehat{V}^n(B^n(p^n - p^{n-1})))^T)^T \right] + \\ (\lambda_f^{n+1})^T \left[(\nabla_{p^n}(\widehat{V}^{n+1}(B^{n+1}))^T)^T \right] + (\lambda_s^n)^T \left[(\nabla_{p^n}(F_{2,1}^n)^T)^T + (\nabla_{p^n}(F_{2,2}^n)^T)^T \right] + \\ (\nabla_{p^n} g)^T = 0, \end{aligned} \tag{3.72}$$

and

$$(\lambda_f^n)^T \left[(\nabla_{p_{wf}^n}(F_{1,2}^n)^T)^T \right] + (\lambda_s^n)^T \left[(\nabla_{p_{wf}^n}(F_{2,2}^n)^T)^T \right] + (\nabla_{p_{wf}^n} g)^T = 0, \tag{3.73}$$

for $n = L, L-1, ...1.$ with the additional constraints given by,

$$\lambda_f^{L+1} = 0 \;\text{ and }\; \lambda_s^{L+1} = 0. \tag{3.74}$$

Taking transpose of Eqs. 3.72 and 3.73 we get,

$$\begin{aligned} \left[(\nabla_{p^n}(F_{1,1}^n)^T) + (\nabla_{p^n}(F_{1,2}^n)^T) - (\nabla_{p^n}(\widehat{V}^n(B^n(p^n - p^{n-1})))^T) \right] (\lambda_f^n) + \\ \left[(\nabla_{p^n}(\widehat{V}^{n+1}(B^{n+1}))^T) \right] (\lambda_f^{n+1}) + \left[(\nabla_{p^n}(F_{2,1}^n)^T) + (\nabla_{p^n}(F_{2,2}^n)^T) \right] (\lambda_s^n) + \\ (\nabla_{p^n} g) = 0, \end{aligned} \tag{3.75}$$

and

$$\left[\left(\nabla_{p^n_{wf}}(F^n_{1,2})^T\right)\right](\lambda^n_f) + \left[\left(\nabla_{p^n_{wf}}(F^n_{2,2})^T\right)\right](\lambda^n_s) + (\nabla_{p^n_{wf}} g) = 0. \tag{3.76}$$

Eqs. 3.75 and 3.76 are the adjoint equations need to be solved.

With the selection of the proper source function, we can evaluate the set of adjoint variables and solve the set of equations backward in time starting from $n = L$.

3.6 Matrix Structure Involved in the Adjoint Equations

The structure of the matrices involved in the adjoint equations are the same as the simulator equations and hence the same sparse matrix solver can be used to solve the adjoint system of equations. Consider the following matrix appearing in Eq. 3.75,

$$\left(\nabla_{p^n}(F^n_{1,1})^T\right) = \begin{bmatrix} \frac{\partial F^n_{1,1,1}}{\partial p^n_1} & \frac{\partial F^n_{1,1,2}}{\partial p^n_1} & \cdots & \frac{\partial F^n_{1,1N}}{\partial p^n_1} \\ \frac{\partial F^n_{1,1,1}}{\partial p^n_2} & \frac{\partial F^n_{1,1,2}}{\partial p^n_2} & \cdots & \frac{\partial F^n_{1,1N}}{\partial p^n_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F^n_{1,1,1}}{\partial p^n_N} & \frac{\partial F^n_{1,1,2}}{\partial p^n_N} & \cdots & \frac{\partial F^n_{1,1,N}}{\partial p^n_N} \end{bmatrix} \tag{3.77}$$

where $F^n_{1,1,1}$ is the first element of the column matrix $F^n_{1,1}$, $F^n_{1,1,2}$ is the second element and so on. The element p^n_1 denotes the first term of the column vector p^n i.e. p^n_1 is the gridblock pressure of the grid block $(1,1,1)$, p^n_2 is the gridblock pressure of the grid block $(2,1,1)$ and so on. If m represents the (i,j,k) cell, then m^{th} element of the column matrix $F^n_{1,1}$ will be

$$\begin{aligned} F^n_{1,1,m} = {} & T^n_{z,i,j,k-1/2} p^n_{i,j,k-1} + T^n_{y,i,j-1/2,k} p^n_{i,j-1,k} + T^n_{x,i-1/2,j,k} p^n_{i-1,j,k} + \\ & -(T^n_{t,i,j,k} + WI^n_{i,j,k}) p^n_{i,j,k} + T^n_{x,i+1/2,j,k} p^n_{i+1,j,k} + \\ & T^n_{y,i,j+1/2,k} p^n_{i,j+1,k} + T^n_{z,i,j,k+1/2} p^n_{i,j,k+1}, \end{aligned} \tag{3.78}$$

where the terms bear the usual meaning. The term $WI^n_{i,j,k}$ appears if the cell m is intersected by a well and if not, then this term disappears. The element p^n_m or $p^n_{i,j,k}$ is associated with the transmissibility terms $T^n_{z,i,j,k-1/2}$, $T^n_{y,i,j-1/2,k}$, $T^n_{x,i-1/2,j,k}$, $T^n_{x,i+1/2,j,k}$,

$T^n_{y,i,j+1/2,k}$ and $T^n_{z,i,j,k+1/2}$ since the transmissibility terms contain the pressure dependent gas properties μ_g and B_g. So the derivative of $F^n_{1,1,l}$ for $l = 1, 2, ..., m, ..., N$ with respect to $p^n_{i,j,k}$ will assume a nonzero value only when $F^n_{1,1,l}$ contains either $p^n_{i,j,k}$ explicitly or one of the above six transmissibility terms and otherwise zero. Considering the m^{th} row of the $\left(\nabla_{p^n}(F^n_{1,1})^T\right)$ matrix, the nonzero terms will be

$$\left(\nabla_{p^n}(F^n_{1,1})^T\right)_{m^{th}\ row} = \quad \cdots \quad \frac{\partial F_{1,1,m-N_x\times N_y}}{\partial p^n_m} \quad \cdots \quad \frac{\partial F_{1,1,m-N_x}}{\partial p^n_m} \quad \cdots \quad \frac{\partial F_{1,1,m-1}}{\partial p^n_m} \quad \frac{\partial F_{1,1,m}}{\partial p^n_m} \quad \frac{\partial F_{1,1,m+1}}{\partial p^n_m} \quad \cdots \quad \frac{\partial F_{1,1,m+N_x}}{\partial p^n_m} \quad \cdots \quad \frac{\partial F_{1,1,m+N_x\times N_y}}{\partial p^n_m} \quad \cdots \tag{3.79}$$

and the rest of the terms will be zero. Similarly, considering the m^{th} element of the column matrix $F^n_{1,2}$,

$$F^n_{1,2,m} = \begin{cases} WI^n_{i,j,k}p^n_{wf,i,j} & \text{if } m \text{ is intersected by a well} \\ 0 & \text{if } m \text{ is not intersected by a well} \end{cases} \tag{3.80}$$

So the derivative of $F^n_{1,2,l}$ for $l = 1, 2, ..., m, ..., N$ with respect to p^n_m will assume a nonzero value only when $l = m$ and otherwise zero. So considering the whole matrix,

$$\left(\nabla_{p^n}(F^n_{1,2})^T\right) = \begin{bmatrix} \frac{\partial F^n_{1,2,1}}{\partial p^n_1} & 0 & \cdots & 0 \\ 0 & \frac{\partial F^n_{1,2,2}}{\partial p^n_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\partial F^n_{1,2,N}}{\partial p^n_N} \end{bmatrix}, \tag{3.81}$$

is a diagonal matrix where the diagonal elements will be nonzero only if the corresponding cell is intersected by a well. The matrix $\left(\nabla_{p^n}(\widehat{V}^n(B^n(p^n - p^{n-1})))^T\right)$ is also a diagonal matrix. Hence $\left[(\nabla p^n(F^n_{1,1})^T) + (\nabla p^n(F^n_{1,2})^T) - (\nabla_{p^n}(\widehat{V}^n(B^n(p^n - p^{n-1})))^T)\right]$ will have the same structure as $A_{1,1}$ defined earlier in the simulator equation.

Now in Eq. 3.75, $(F^n_{2,1}) = A_{2,1}p_n$ is an $N_w \times 1$ matrix. It assumes the form

$$(F^n_{2,1}) = \begin{bmatrix} \sum \left(WI^n_{i',j',k'} p^n_{i',j',k'} \right)_1 \\ \sum \left(WI^n_{i',j',k'} p^n_{i',j',k'} \right)_2 \\ \vdots \\ \sum \left(WI^n_{i',j',k'} p^n_{i',j',k'} \right)_{N_w} \end{bmatrix}, \tag{3.82}$$

where

$$\sum (WI^n_{i',j',k'} p^n_{i',j',k'})_i = \sum_{k'=l_i}^{r_i} (WI^n_{i',j',k'} p^n_{i',j',k'}) \text{ for well number } i,\ i = 1, ..., N_w \tag{3.83}$$

denote the appropriate terms involved with the well constraints equation given by Eq. 3.32 and r_i,..,l_i denote the perforated intervals for well i. The matrix $\left(\nabla_{p^n}(F^n_{2,1})^T\right)$ can be expanded as

$$\left(\nabla_{p^n}(F^n_{2,1})^T\right) = \begin{bmatrix} \frac{\partial \sum \left(WI^n_{i',j',k'} p^n_{i',j',k'} \right)_1}{\partial p^n_1} & \frac{\partial \sum \left(WI^n_{i',j',k'} p^n_{i',j',k'} \right)_2}{\partial p^n_1} & \cdots & \frac{\partial \sum \left(WI^n_{i',j',k'} p^n_{i',j',k'} \right)_{N_w}}{\partial p^n_1} \\ \frac{\partial \sum \left(WI^n_{i',j',k'} p^n_{i',j',k'} \right)_1}{\partial p^n_2} & \frac{\partial \sum \left(WI^n_{i',j',k'} p^n_{i',j',k'} \right)_2}{\partial p^n_2} & \cdots & \frac{\partial \sum \left(WI^n_{i',j',k'} p^n_{i',j',k'} \right)_{N_w}}{\partial p^n_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \sum \left(WI^n_{i',j',k'} p^n_{i',j',k'} \right)_1}{\partial p^n_N} & \frac{\partial \sum \left(WI^n_{i',j',k'} p^n_{i',j',k'} \right)_2}{\partial p^n_N} & \cdots & \frac{\partial \sum \left(WI^n_{i',j',k'} p^n_{i',j',k'} \right)_{N_w}}{\partial p^n_N} \end{bmatrix} \tag{3.84}$$

Considering row m of the above matrix, the derivatives are nonzero only if the $(i^{'}, j^{'}, k^{'})$ within the summation term coincides with m, i.e., (i, j, k). This will give N_z nonzero terms in each column of the above matrix if all wells are completely penetrating. Similarly,

$$\left(\nabla_{p^n}(F^n_{2,2})^T\right) = \begin{bmatrix} \frac{\partial \sum\left(\left(WI^n_{i',j',k'}\right)p^n_{wf}\right)_1}{\partial p^n_1} & \frac{\partial \sum\left(\left(WI^n_{i',j',k'}\right)p^n_{wf}\right)_2}{\partial p^n_1} & \cdots & \frac{\partial \sum\left(\left(WI^n_{i',j',k'}\right)p^n_{wf}\right)_{N_w}}{\partial p^n_1} \\ \frac{\partial \sum\left(\left(WI^n_{i',j',k'}\right)p^n_{wf}\right)_1}{\partial p^n_2} & \frac{\partial \sum\left(\left(WI^n_{i',j',k'}\right)p^n_{wf}\right)_2}{\partial p^n_2} & \cdots & \frac{\partial \sum\left(\left(WI^n_{i',j',k'}\right)p^n_{wf}\right)_{N_w}}{\partial p^n_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \sum\left(\left(WI^n_{i',j',k'}\right)p^n_{wf}\right)_1}{\partial p^n_N} & \frac{\partial \sum\left(\left(WI^n_{i',j',k'}\right)p^n_{wf}\right)_2}{\partial p^n_N} & \cdots & \frac{\partial \sum\left(\left(WI^n_{i',j',k'}\right)p^n_{wf}\right)_{N_w}}{\partial p^n_N} \end{bmatrix} \tag{3.85}$$

and considering the m^{th} row, the derivative term will be nonzero only if $(i^{'}, j^{'}, k^{'})$ within the summation term coincides with m. So $\left(\nabla_{p^n}(F^n_{2,1})^T\right)$ and $\left(\nabla_{p^n}(F^n_{2,1})^T\right)$ have the same structure and their nonzero structure is the same as that of $A_{1,2}$ defined in the simulator equation. The $\left(\nabla_{p^n_w}(F^n_{1,2})^T\right)$ appearing in Eq. 3.76 can be expanded as

$$\left(\nabla_{p^n_w}(F^n_{1,2})^T\right) = \begin{bmatrix} \frac{\partial F^n_{1,2,1}}{\partial p^n_{wf,1}} & \frac{\partial F^n_{1,2,2}}{\partial p^n_{wf,1}} & \cdots & \frac{\partial F^n_{1,2,N}}{\partial p^n_{wf,1}} \\ \frac{\partial F^n_{1,2,1}}{\partial p^n_{wf,2}} & \frac{\partial F^n_{1,2,2}}{\partial p^n_{wf,2}} & \cdots & \frac{\partial F^n_{1,2,N}}{\partial p^n_{wf,2}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F^n_{1,2,1}}{\partial p^n_{wf,N_w}} & \frac{\partial F^n_{1,2,2}}{\partial p^n_{wf,N_w}} & \cdots & \frac{\partial F^n_{1,2,N}}{\partial p^n_{wf,Nw}} \end{bmatrix} \tag{3.86}$$

The derivatives $\frac{\partial F^n_{1,2,m}}{\partial p^n_{wf,w}}$ for $m = 1, 2, ..., N$ and $w = 1, ..., N_w$ will be nonzero if $F^n_{1,2,m}$ contains the term $p_{wf,w}$ i.e. the areal location of the grid block m coincides with the areal location of the well w. So, we will get exactly N_z nonzero elements in each row of the above matrix if all the wells are completely penetrating. The structure of this matrix is same as the structure of $A_{2,1}$ defined in the simulator equation.

The matrix $\left(\nabla_{p^n_{wf}}(F^n_{2,2})^T\right)$ can be expanded as

$$\left(\nabla_{p^n_{wf}}(F^n_{2,2})^T\right) = \begin{bmatrix} \frac{\partial \sum\left(\left(WI^n_{i,j,k}\right)p_{wf}\right)_1}{\partial p^n_{wf,1}} & \frac{\partial \sum\left(\left(WI^n_{i,j,k}\right)p_{wf}\right)_2}{\partial p^n_{wf,1}} & \cdots & \frac{\partial \sum\left(\left(WI^n_{i,j,k}\right)p_{wf}\right)_{N_w}}{\partial p^n_{wf,1}} \\ \frac{\partial \sum\left(\left(WI^n_{i,j,k}\right)p_{wf}\right)_1}{\partial p^n_{wf,2}} & \frac{\partial \sum\left(\left(WI^n_{i,j,k}\right)p_{wf}\right)_2}{\partial p^n_{wf,2}} & \cdots & \frac{\partial \sum\left(\left(WI^n_{i,j,k}\right)p_{wf}\right)_{N_w}}{\partial p^n_{wf,2}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \sum\left(\left(WI^n_{i,j,k}\right)p_{wf}\right)_1}{\partial p_{wf,N_w}} & \frac{\partial \sum\left(\left(WI^n_{i,j,k}\right)p_{wf}\right)_2}{\partial p^n_{wf,N_w}} & \cdots & \frac{\partial \sum\left(\left(WI^n_{i,j,k}\right)p_{wf}\right)_{N_w}}{\partial p^n_{wf,N_w}} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\partial \sum\left(\left(WI^n_{i,j,k}\right)p_{wf}\right)_1}{\partial p^n_{wf,1}} & 0 & \cdots & 0 \\ 0 & \frac{\partial \sum\left(\left(WI^n_{i,j,k}\right)p_{wf}\right)_2}{\partial p^n_{wf,2}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\partial \sum\left(\left(WI^n_{i,j,k}\right)p_{wf}\right)_{N_w}}{\partial p^n_{wf,N_w}} \end{bmatrix} \tag{3.87}$$

So we get a diagonal matrix whose structure is same as that of $A_{2,2}$ of the simulator equation. Thus in simple matrix notations, the adjoint equations can be represented as

$$M_{1,1}(\lambda^n_f) + M_{1,2}(\lambda^n_s) = -(\nabla_{p^n} g) - \left[(\nabla_{p^n}(\widehat{V}^{n+1}(B^{n+1}))^T)\right](\lambda^{n+1}_f), \tag{3.88}$$

and

$$M_{2,1}(\lambda^n_f) + M_{2,2}(\lambda^n_s) = -(\nabla_{p^n_{wf}} g), \tag{3.89}$$

where

$$M_{1,1} = \left[(\nabla p^n (F^n_{1,1})^T) + (\nabla p^n (F^n_{1,2})^T) - (\nabla_{p^n}(\widehat{V}^n(B^n(p^n - p^{n-1})))^T)\right], \tag{3.90}$$

$$M_{1,2} = \left[(\nabla p^n (F^n_{2,1})^T) + (\nabla p^n (F^n_{2,2})^T)\right], \tag{3.91}$$

$$M_{2,1} = \left[(\nabla p^n_{wf}(F^n_{1,2})^T)\right], \tag{3.92}$$

and

$$M_{2,2} = \left[(\nabla p^n_{wf}(F^n_{2,2})^T)\right]. \tag{3.93}$$

The structure of $M_{1,1}$, $M_{1,2}$, $M_{2,1}$ and $M_{2,2}$ in Eqs. 3.88 and 3.89 are same as $A_{1,1}$, $A_{1,2}$, $A_{2,1}$ and $A_{2,2}$ in Eqs. 3.33 and 3.34 respectively.

3.7 General Equations for Calculating the Sensitivity Coefficients

Considering J as a function of ϕ, k_x, k_y, k_z and s_{skin}, we can write its total differential as

$$dJ = (\nabla_{k_x} J)^T dk_x + (\nabla_{k_y} J)^T dk_y + (\nabla_{k_z} J)^T dk_z + (\nabla_{\phi} J)^T d\phi + (\nabla_{s_{skin}} J)^T ds_{skin}. \tag{3.94}$$

So by comparing Eq. 3.67 and Eq. 3.94 we see that,

$$(\nabla_{k_x} J)^T = \sum_{n=1}^{L} \Big[(\lambda_f^n)^T [(\nabla_{k_x} (F_{1,1}^n)^T)^T + (\nabla_{k_x} (F_{1,2}^n)^T)^T] + (\lambda_s^n)^T [(\nabla_{k_x} (F_{2,1}^n)^T)^T + (\nabla_{k_x} (F_{2,2}^n)^T)^T] + [\nabla_{k_x} g]^T \Big], \tag{3.95}$$

$$(\nabla_{k_y} J)^T = \sum_{n=1}^{L} \Big[(\lambda_f^n)^T [(\nabla_{k_y} (F_{1,1}^n)^T)^T + (\nabla_{k_y} (F_{1,2}^n)^T)^T] + (\lambda_s^n)^T [(\nabla_{k_y} (F_{2,1}^n)^T)^T + (\nabla_{k_y} (F_{2,2}^n)^T)^T] + [\nabla_{k_y} g]^T \Big], \tag{3.96}$$

$$(\nabla_{k_z} J)^T = \sum_{n=1}^{L} \Big[(\lambda_f^n)^T [(\nabla_{k_z} (F_{1,1}^n)^T)^T + [\nabla_{k_z} g]^T \Big], \tag{3.97}$$

$$(\nabla_{\phi} J)^T = \sum_{n=1}^{L} \Big[(\lambda_f^n)^T [(\nabla_{\phi} \left[\widehat{V}^n (B^n p^n - B^n p^{n-1})^T \right]^T + [\nabla_{\phi} g]^T \Big], \tag{3.98}$$

and

$$(\nabla_{s_{skin}} J)^T = \sum_{n=1}^{L} \Big[(\lambda_f^n)^T \left[(\nabla_{s_{skin}} (F_{1,1}^n)^T)^T + (\nabla_{s_{skin}} (F_{1,2}^n)^T)^T \right] + (\lambda_s^n)^T \left[(\nabla_{s_{skin}} (F_{2,1}^n)^T)^T + (\nabla_{s_{skin}} (F_{2,2}^n)^T)^T \right] + [\nabla_{s_{skin}} g]^T \Big]. \tag{3.99}$$

3.8 Differentiation Involved in the Process

We have to find out many derivatives at each time step to evaluate the gradient of the terms that appear in the flow equation. Expressions for the derivatives needed to formulate the adjoint equations are given in this section.

3.8.1 Differentiation of Pressure Dependent Terms

$$\begin{aligned}
\frac{\partial}{\partial p_{i,j,k}}\left(\frac{1}{\mu B_g}\right)_{i+1/2,j,k} &= \frac{\partial}{\partial p_{i+1/2,j,k}}\left(\frac{1}{\mu B_g}\right)_{i+1/2,j,k}\frac{\partial p_{i+1/2,j,k}}{\partial p_{i,j,k}} \\
&= \left[\left(\frac{1}{B_g}\right)_{i+1/2,j,k}\frac{\partial}{\partial p_{i+1/2,j,k}}\left(\frac{1}{\mu}\right)_{i+1/2,j,k} + \right. \\
&\quad \left.\left(\frac{1}{\mu}\right)_{i+1/2,j,k}\frac{\partial}{\partial p_{i+1/2,j,k}}\left(\frac{1}{B_g}\right)_{i+1/2,j,k}\right]\frac{\partial p_{i+1/2,j,k}}{\partial p_{i,j,k}} \\
&= \left[\frac{1}{B_g}\frac{\left(-\frac{\partial \mu}{\partial p}\right)}{\mu^2} + \frac{1}{\mu}\frac{\left(-\frac{\partial B_g}{\partial p}\right)}{B_g^2}\right]_{p_{i+1/2,j,k}}\frac{\partial p_{i+1/2,j,k}}{\partial p_{i,j,k}}. \quad (3.100)
\end{aligned}$$

Based on linear interpolation, we have

$$p_{i+1/2,j,k} = \left[p_{i,j,k} + \frac{(p_{i+1,j,k} - p_{i,j,k})\Delta X_i}{\Delta X_i + \Delta X_{i+1}}\right]. \quad (3.101)$$

Hence,

$$\frac{\partial p_{i+1/2,j,k}}{\partial p_{i,j,k}} = \left(1 - \frac{\Delta X_i}{\Delta X_i + \Delta X_{i+1}}\right). \quad (3.102)$$

Substituting Eq. 3.102 in 3.100 we have,

$$\frac{\partial}{\partial p_{i,j,k}}\left(\frac{1}{\mu B_g}\right)_{i+1/2,j,k} = \left[\frac{1}{B_g}\frac{\left(-\frac{\partial \mu}{\partial p}\right)}{\mu^2} + \frac{1}{\mu}\frac{\left(-\frac{\partial B_g}{\partial p}\right)}{B_g^2}\right]_{p_{i+1/2,j,k}}\left(1 - \frac{\Delta X_i}{\Delta X_i + \Delta X_{i+1}}\right). \quad (3.103)$$

Similarly,

$$\frac{\partial}{\partial p_{i,j,k}}\left(\frac{1}{\mu B_g}\right)_{i-1/2,j,k} = \left[\frac{1}{B_g}\frac{\left(-\frac{\partial \mu}{\partial p}\right)}{\mu^2} + \frac{1}{\mu}\frac{\left(-\frac{\partial B_g}{\partial p}\right)}{B_g^2}\right]_{p_{i-1/2,j,k}}\frac{\partial p_{i-1/2,j,k}}{\partial p_{i,j,k}} \quad (3.104)$$

The term $\frac{\partial p_{i-1/2,j,k}}{\partial p_{i,j,k}}$ can be evaluated in the similar way as shown above.

3.8.2 Differentiation of Transmissibility Terms

We need to evaluate,

$$\begin{aligned}\frac{\partial T_{x,i+1/2,,j,k}}{\partial k_{x,i,j,k}} &= \frac{\partial}{\partial k_{x,i,j,k}}\left(\frac{1.127\times10^{-3}\Delta y_j\Delta z_k k_{x+1/2,j,k}}{B_{g,i+1/2,j,k}\ \mu_{i+1/2,j,k}(x_{i+1}-x_i)}\right)\\ &= \left(\frac{1.127\times10^{-3}\Delta y_j\Delta z_k}{B_{g,i+1/2,j,k}\ \mu_{i+1/2,j,k}(x_{i+1}-x_i)}\right)\left(\frac{\partial k_{x+1/2,j,k}}{\partial k_{x,i,j,k}}\right). \quad (3.105)\end{aligned}$$

The term $k_{x+1/2,j,k}$ is given by

$$k_{x+1/2,j,k} = \frac{(\Delta x_i+\Delta x_{i+1})k_{x,i,j,k}k_{x,i+1,j,k}}{\Delta x_i k_{x,i+1,j,k}+\Delta x_{i+1}k_{x,i,j,k}}. \quad (3.106)$$

Hence,

$$\begin{aligned}\frac{\partial k_{x+1/2,j,k}}{\partial k_{x,i,j,k}} &= \frac{1}{(\Delta x_{i+1}k_{x,i,j,k}+\Delta x_i k_{x,i+1,j,k})^2}\\ &\Big[(\Delta x_{i+1}k_{x,i,j,k}+\Delta x_i k_{x,i+1,j,k})(\Delta x_{i+1}+\Delta x_i)k_{x,i+1,j,k}-\\ &(\Delta x_{i+1}+\Delta x_i)k_{x,i,j,k}k_{x,i+1,j,k}\Delta x_{i+1}\Big]. \quad (3.107)\end{aligned}$$

Substituting Eq. 3.107 in Eq. 3.105 we have,

$$\begin{aligned}\frac{\partial T_{x,i+1/2,,j,k}}{\partial k_{x,i,j,k}} &= \left(\frac{1.127\times10^{-3}\Delta y_j\Delta z_k}{B_{g,i+1/2,j,k}\mu_{i+1/2,j,k}(x_{i+1}-x_i)}\right)\times\frac{1}{(\Delta x_{i+1}k_{x,i,j,k}+\Delta x_i k_{x,i+1,j,k})^2}\\ &\Big[(\Delta x_{i+1}k_{x,i,j,k}+\Delta x_i k_{x,i+1,j,k})(\Delta x_{i+1}+\Delta x_i)k_{x,i+1,j,k}-\\ &(\Delta x_{i+1}+\Delta x_i)k_{x,i,j,k}k_{x,i+1,j,k}\Delta x_{i+1}\Big]. \quad (3.108)\end{aligned}$$

Similarly, we can show that

$$\begin{aligned}\frac{\partial T_{x,i-1/2,,j,k}}{\partial k_{x,i,j,k}} &= \frac{\partial}{\partial k_{x,i,j,k}}\left(\frac{1.127\times10^{-3}\Delta y_j\Delta z_k k_{x-1/2,j,k}}{B_{g,i-1/2,j,k}\ \mu_{i-1/2,j,k}(x_i-x_{i-1})}\right)\\ &= \left(\frac{1.127\times10^{-3}\Delta y_j\Delta z_k}{B_{g,i-1/2,j,k}\ \mu_{i-1/2,j,k}(x_i-x_{i-1})}\right)\left(\frac{\partial k_{x-1/2,j,k}}{\partial k_{x,i,j,k}}\right)\\ &= \left(\frac{1.127\times10^{-3}\Delta y_j\Delta z_k}{B_{g,i-1/2,j,k}\mu_{i-1/2,j,k}(x_i-x_{i-1})}\right)\times\frac{1}{(\Delta x_i k_{x,i-1,j,k}+\Delta x_{i-1}k_{x,i,j,k})^2}\\ &\Big[(\Delta x_i k_{x,i-1,j,k}+\Delta x_{i-1}k_{x,i,j,k})(\Delta x_i+\Delta x_{i-1})k_{x,i-1,j,k}+\\ &(\Delta x_i+\Delta x_{i-1})k_{x,i-1,j,k}k_{x,i,j,k}\Delta x_{i-1}\Big]. \quad (3.109)\end{aligned}$$

It is easy to see that similar expression can be derived for the differential of $T_{y,i,j-1/2,k}$, $T_{y,i,j+1/2,k}$, $T_{z,i,j,k-1/2}$, $T_{z,i,j,k+1/2}$ with respect to $k_{y,i,j,k}$ and $k_{z,i,j,k}$.

3.8.3 Differentiation of Well-Index Terms

We need to find out the differential of $WI_{i,j,k}(t)$ with respect to $k_{x,i,j,k}$ and $k_{y,i,j,k}$. These relevant derivatives are

$$\frac{\partial WI_{i,j,k}(t)}{\partial k_{x,i,j,k}} = \left(\frac{1}{\mu_{i,j,k}(t)B_{g,i,j,k}(t)}\right)\left(\frac{C_3 h}{2[\ln(r_{o,i,j,k}/r_{w,i,j}) + s_{i,j,k}]}\right)\times \left[\sqrt{\frac{k_{y,i,j,k}}{k_{x,i,j,k}}} - \left(\frac{1}{\ln(r_{o,i,j,k}/r_{w,i,j}) + s_{i,j,k}}\right)\times \left(\frac{\Delta y_{i,j,k}^2\sqrt{k_{x,i,j,k}k_{y,i,j,k}}}{\Delta x_{i,j,k}^2 k_{y,i,j,k} + \Delta y_{i,j,k}^2 k_{x,i,j,k}} - \frac{\sqrt{k_{y,i,j,k}}}{\sqrt{k_{x,i,j,k}} + \sqrt{k_{y,i,j,k}}}\right)\right], \tag{3.110}$$

$$\frac{\partial WI_{i,j,k}(t)}{\partial k_{y,i,j,k}} = \left(\frac{1}{\mu_{i,j,k}(t)B_{g,i,j,k}(t)}\right)\left(\frac{C_3 h}{2[\ln(r_{o,i,j,k}/r_{w,i,j}) + s_{i,j,k}]}\right)\times \left[\sqrt{\frac{k_{x,i,j,k}}{k_{y,i,j,k}}} - \left(\frac{1}{\ln(r_{o,i,j,k}/r_{w,i,j}) + s_{i,j,k}}\right)\times \left(\frac{\Delta y_{i,j,k}^2 k_{x,i,j,k}\sqrt{k_{x,i,j,k}}}{\Delta x_{i,j,k}^2 k_{y,i,j,k}\sqrt{k_{y,i,j,k}} + \Delta y_{i,j,k}^2 k_{x,i,j,k}\sqrt{k_{y,i,j,k}}} - \frac{\sqrt{k_{y,i,j,k}}}{\sqrt{k_{x,i,j,k}} + \sqrt{k_{x,i,j,k}k_{y,i,j,k}}}\right)\right], \tag{3.111}$$

and,

$$\frac{\partial WI_{i,j,k}(t)}{\partial s_{skin,i,j,k}} = -\left[\frac{WI_{i,j,k}(t)}{\ln(r_{o,i,j,k}/r_{w,i,j}) + s_{i,j,k}}\right]. \tag{3.112}$$

The above equations have been taken from previous work done within TUPREP.

3.9 Sensitivity of Flowing Bottomhole Pressure

We wish to calculate the sensitivity of the flowing bottomhole pressure when the well is producing at a specified flow rate i.e., the well constraint is given by $Q(t)$ =constant.

Let $p^{obs}_{wf,i,j}(t^r)$ denote the measured wellbore pressure at the time t^r and for the well completed in cells (i,j,k) $k = l_1, ..., l_2$. Now let us call this cell m and assume we wish to compute the sensitivity coefficients related to the calculated pressure $p_{wf,m}(t^r)$ that will be predicted by the simulator for a given set of porosities, permeabilities and skin factors. So we choose

$$g = p^r_{wf,m}. \tag{3.113}$$

Then for all l,

$$\nabla_{p^l}(g) = 0. \tag{3.114}$$

For all $l \neq r$,

$$\nabla_{p^l_{wf}}(g) = 0, \tag{3.115}$$

and for $l = r$,

$$\nabla_{p^l_{wf}}(g) = \nabla_{p^l_{wf}}(p^r_{wf,m}) = \nabla_{p^l_{wf}}(p^l_{wf,m}) = e^m, \tag{3.116}$$

where the i^{th} component of e^m is given by $e^m_i = \delta_{im}$ where δ_{im} is the delta function given by

$$\delta_{im} = \begin{cases} 1 & \text{if } i = m, \\ 0 & \text{if } i \neq m. \end{cases} \tag{3.117}$$

Eqs. 3.114-3.116 give the source functions to be used in Eqs. 3.75 and 3.76 while solving for the adjoint variables. As shown by Eq. 3.114 and 3.116 we put a unit source in the m^{th} component of the right hand side of Eq. 3.76 at the time $t^l = t^r$. So to put the source time at the exact time step, we put the adjoint variables equal to zero at time step $(l+1) = (r+1)$ and then solve the adjoint system for $l = r, r-1, ..., 1, 0$. Similarly, if we want to compute the sensitivity coefficients of $p^s_{wf,n}$ (i.e. flowing

pressure of a well at different location, say in gridblock n, and different time step t^s) then we choose,

$$g = p^s_{wf,n}. \tag{3.118}$$

Then for all l,

$$\nabla_{p^l}(g) = 0. \tag{3.119}$$

For all $l \neq s$,

$$\nabla_{p^l_{wf}}(g) = 0, \tag{3.120}$$

and for $l = s$,

$$\nabla_{p^l_{wf}}(g) = \nabla_{p^l_{wf}}(p^s_{wf,n}) = \nabla_{p^l_{wf}}(p^l_{wf,n}) = e^n, \tag{3.121}$$

where the i^{th} component of e^n is given by

$$e^n_i = \delta_{in}. \tag{3.122}$$

Using these source terms, we solve the adjoint equations backward in time for $l = s, s-1, ..., 1, 0$. This shows that we have to solve one adjoint system of equations for each data point. This is the procedure currently being used in the code, but we could make it more efficient. In particular, only the right hand side of the adjoint equations depends on the data. So effectively we have to solve one matrix with N_d right hand sides where N_d is the number of conditioning data.

Once the adjoint variables are calculated, the sensitivity of well bore pressure to $k_{x,i,j,k}$, $k_{y,i,j,k}$, $k_{z,i,j,k}$, $\phi_{i,j,k}$ and $s_{skin,i,j,k}$ can be computed using Eq. 3.95, 3.96, 3.97, 3.98 and 3.99 respectively.

3.10 Sensitivity of Layer Flow Rate or Total Flow Rate

If the wellbore pressure is specified at a well, we would like to condition the rock property field to rate data. So we are interested in the sensitivity of the rate data to porosity, permeability and skin factor.

If the wellbore pressure is specified, it is no longer an unknown. So in the simulator, we solve only for $p_{i,j,k}(t)$. After solving for gridblock pressures, the layer flow rate can be calculated using Eq. 3.21 and the total flow at the well can be calculated by using Eq. 3.29 for $t = t^n$. So the simulator equation becomes only

$$A^n_{1,1}p^n + A^n_{1,2}p^n_{wf} = \widehat{V}^n \left[B^n \left(p^n - p^{n-1}\right)\right]. \tag{3.123}$$

Since in this case p_{wf} is specified, $d(p_{wf}(t)) = 0$. So instead of solving two adjoint systems, we have only one adjoint system and the adjoint equations are given by

$$(\lambda^n_f)^T \left[(\nabla p^n (F^n_{1,1})^T)^T + (\nabla p^n (F^n_{1,2})^T)^T - (\nabla_{p^n} \left[\widehat{V}^n (B^n (p^n - p^{n-1}))\right]^T)^T\right] + (\lambda^{n+1}_f)^T \left[(\nabla_{p^n} \left[\widehat{V}^{n+1} (B^{n+1} p^n)\right]^T)^T\right] + (\nabla_{p^n} g)^T = 0, \tag{3.124}$$

where $F^n_{1,1}$ and similar terms are still defined by Eqs. 3.50 and 3.51.

Similarly, the sensitivity equations will take the form

$$(\nabla_{k_x} J)^T = \sum_{n=1}^{L} \left[(\lambda^n_f)^T [(\nabla_{k_x} (F^n_{1,1})^T)^T + (\nabla_{k_x} (F^n_{1,2})^T)^T] + [\nabla_{k_x} g]^T\right], \tag{3.125}$$

$$(\nabla_{k_y} J)^T = \sum_{n=1}^{L} \left[(\lambda^n_f)^T [(\nabla_{k_y} (F^n_{1,1})^T)^T + (\nabla_{k_y} (F^n_{1,2})^T)^T] + [\nabla_{k_y} g]^T\right], \tag{3.126}$$

$$(\nabla_{k_z} J)^T = \sum_{n=1}^{L} \left[(\lambda^n_f)^T [(\nabla_{k_z} (F^n_{1,1})^T)^T + [\nabla_{k_z} g]^T\right], \tag{3.127}$$

$$(\nabla_{\phi} J)^T = \sum_{n=1}^{L} \left[(\lambda^n_f)^T [(\nabla_{\phi} (\widehat{V}^n (B^n p^n - B^n p^{n-1})^T)^T + [\nabla_{\phi} g]^T\right], \tag{3.128}$$

and

$$(\nabla_{S_{skin}} J)^T = \sum_{n=1}^{L} \left[(\lambda^n_f)^T [(\nabla_{s_{skin}} (F^n_{1,1})^T)^T + (\nabla_{s_{skin}} (F^n_{1,2})^T)^T] + [\nabla_{s_{skin}} g]^T\right]. \tag{3.129}$$

The flow rate sensitivity can be calculated by two methods.

3.10.1 Method 1

In this method, we first calculate the sensitivity of the gridblock pressure and then translate it to the sensitivity of flowrate. Let $q^n_{i,j,k}$ denote the layer flow rate at $t = t^n$ for a well completed in the gridblock centered at (x_i, y_j, z_k) and let α denote a reservoir parameter, i.e., a gridblock porosity or permeability. We wish to calculate $\frac{\partial}{\partial\alpha}(q^n_{i,j,k})$.

We know that

$$q_{i,j,k}(t) = WI_{i,j,k}(t)(p_{i,j,k}(t) - p_{wf,i,j}(t)). \tag{3.130}$$

Let us denote

$$\widehat{WI}_{i,j,k} = \frac{C_3 h\sqrt{k_{x,i,j,k}k_{y,i,j,k}}}{\ln(r_{o,i,j,k}/r_{w,i,j}) + s_{i,j,k}}. \tag{3.131}$$

So $WI_{i,j,k}(t)$ can be denoted by

$$WI_{i,j,k}(t) = \widehat{WI}_{i,j,k}\left(\frac{1}{\mu_{i,j,k}(t)B_{g,i,j,k}(t)}\right).$$

Then we can write Eq. 3.130 as

$$q_{i,j,k}(t) = \widehat{WI}_{i,j,k}(t)\frac{1}{\mu_{i,j,k}(t)B_{g,i,j,k}(t)}(p_{i,j,k}(t) - p_{wf,i,j}(t)). \tag{3.132}$$

Taking the derivative of the last equation, we find

$$\begin{aligned}
\frac{\partial}{\partial\alpha}q_{i,j,k}(t) &= \left[\frac{\partial}{\partial\alpha}WI_{i,j,k}(t)\right][p_{i,j,k}(t) - p_{wf,i,j}(t)] + WI_{i,j,k}(t)\frac{\partial}{\partial\alpha}[p_{i,j,k}(t) - p_{wf,i,j}(t)] \\
&= \left[\widehat{WI}_{i,j,k}(t)\frac{\partial}{\partial\alpha}\left(\frac{1}{\mu_{i,j,k}(t)B_{g,i,j,k}(t)}\right) + \frac{1}{\mu_{i,j,k}(t)B_{g,i,j,k}(t)}\frac{\partial\widehat{WI}_{i,j,k}(t)}{\partial\alpha}\right]\times \\
&\qquad \Big[p_{i,j,k}(t) - p_{wf,i,j}(t)\Big] + WI_{i,j,k}(t)\frac{\partial}{\partial\alpha}[p_{i,j,k}(t) - p_{wf,i,j}(t)] \\
&= \left[\widehat{WI}_{i,j,k}(t)\frac{d}{dp_{i,j,k}(t)}\left(\frac{1}{\mu_{i,j,k}(t)B_{g,i,j,k}(t)}\right)\frac{\partial p_{i,j,k}(t)}{\partial\alpha} + \frac{1}{\mu_{i,j,k}(t)B_{g,i,j,k}(t)}\frac{\partial\widehat{WI}_{i,j,k}(t)}{\partial\alpha}\right]\times \\
&\qquad \Big[p_{i,j,k}(t) - p_{wf,i,j}(t)\Big] + WI_{i,j,k}(t)\frac{\partial}{\partial\alpha}(p_{i,j,k}(t)).
\end{aligned} \tag{3.133}$$

So if we calculate $\frac{\partial}{\partial\alpha}(p_{i,j,k}(t))$, we can evaluate $\frac{\partial}{\partial\alpha}q_{i,j,k}(t)$ since the other terms involved can be computed easily. So we choose the source term accordingly. To compute

sensitivity of $p_{i,j,k}(t)$ (when $t = t^n$) to the parameters, assume cell (i, j, k) is the m^{th} cell and choose

$$g = p^n_{i,j,k} = p^n_m. \tag{3.134}$$

Then for $l \neq n$,

$$\nabla_{p^l}(g) = 0, \tag{3.135}$$

and for $l = n$,

$$\nabla_{p^l}(g) = \nabla_{p^l}(p^n_m) = e^m, \tag{3.136}$$

where e^m denotes the vector whose m^{th} component is equal to unity and all other terms are equal to zero. Thus the source term takes the value of 1 only at the position corresponding to the gridblock m. Now we can solve the adjoint equation given by Eq. 3.124 for $l = n, n-1, ..., 1, 0$ to obtain the adjoint variables. Then it is straightforward to calculate the desired sensitivity coefficients by Eq. 3.125, 3.126, 3.127, 3.128 and 3.129.

However this method suffers from the drawback that to calculate the sensitivity of the total flow rate at a well i.e., $\sum_{k=l_1}^{l_2} q_{i,j,k}(t)$, we have to solve the adjoint system with a source term corresponding to each $p_{i,j,k}(t)$, $k = l_1, ..., l_2$. So even for a single well this leads to the computation of $(l_2 - l_1 + 1)$ adjoint solutions. This is not computationally efficient unless we wish to compute the sensitivity of individual $q_{i,j,k}$. Hence, we seek an alternative procedure.

3.10.2 Method 2

To overcome the shortcoming of the procedure described above, we treat the flow rate term directly as the source term and solve the adjoint equations. Let us consider the total rate at a well. Then we choose

$$g = q^n_{i,,j} = \sum_{k=l_1}^{l_2} WI^n_{i,j,k}(p^n_{i,j,k} - p^n_{wf,i,j}). \tag{3.137}$$

So

$$\nabla_{p^n}(g) = \nabla_{p^n}\left[\sum_{k=l_1}^{l_2} WI^n_{i,j,k}(p^n_{i,j,k} - p^n_{wf,i,j})\right]. \tag{3.138}$$

Now unless $(i^{'}, j^{'}) = (i, j)$ and $k^{'} = k$ for some k with $l_1 \leq k \leq l_2$, we have

$$\frac{\partial}{\partial p^n_{i^{'},j^{'},k^{'}}}\left[\sum_{k=l_1}^{l_2} WI^n_{i,j,k}(p^n_{i,j,k} - p^n_{wf,i,j})\right] = 0. \tag{3.139}$$

If $i^{'} = i$, $j^{'} = j$ and $k^{'} = k$ for some k with $l_1 \leq k \leq l_2$,

$$\begin{aligned}
&\frac{\partial}{\partial p^n_{i^{'},j^{'},k^{'}}}\left[\sum_{k=l_1}^{l_2} WI^n_{i,j,k}(p^n_{i,j,k} - p^n_{wf,i,j})\right] \\
= \;& \frac{\partial}{\partial p^n_{i,j,k}}\left[WI^n_{i,j,k}(p^n_{i,j,k} - p^n_{wf,i,j})\right] \\
= \;& \frac{\partial}{\partial p^n_{i,j,k}}(WI^n_{i,j,k}) + WI^n_{i,j,k}\frac{\partial}{\partial p^n_{i,j,k}}(p^n_{i,j,k} - p^n_{wf,i,j}) \\
= \;& \frac{\partial}{\partial p^n_{i,j,k}}(WI^n_{i,j,k}) + WI^n_{i,j,k}.
\end{aligned} \tag{3.140}$$

Note that the source term will assume a nonzero value only at cell positions where the well is completed and otherwise is zero. Using this modified source function we can solve for the adjoint variables and compute the sensitivity coefficients.

Note that if g is specified by Eq. 3.137, then g involves $k_{x,i,j,k}$, $k_{y,i,j,k}$ and $s_{skin,i,j,k}$. So $(\nabla_{k_x} g)$, $(\nabla_{k_y} g)$, $(\nabla_{s_{skin}} g)$ are not equal to the zero vector. These gradient terms must be included in the corresponding sensitivity Eqs. 3.125, 3.126, 3.129. Note, however $(\nabla_{k_z} g)$and $(\nabla_{\phi} g)$ are identically zero. When g is given by Eq. 3.137,

$$(\nabla_{k_x} g) = \nabla_{k_x}\left[\sum_{k=l_1}^{l_2} WI^n_{i,j,k}(p^n_{i,j,k} - p^n_{wf,i,j})\right]. \tag{3.141}$$

Unless $(i^{'}, j^{'}) = (i, j)$ and $k^{'} = k$ for some k with $l_1 \leq k \leq l_2$, we have

$$\frac{\partial}{\partial k_{x,i^{'},j^{'},k^{'}}}\sum_{k=l_1}^{l_2} WI^n_{i,j,k}(p^n_{i,j,k} - p^n_{wf,i,j}) = 0. \tag{3.142}$$

If $i^{'} = i$, $j^{'} = j$ and $k^{'} = k$ for some k with $l_1 \leq k \leq l_2$,

$$\begin{aligned}
&\frac{\partial}{\partial k_{x,i^{'},j^{'},k^{'}}} \sum_{k=l_1}^{l_2} WI^n_{i,j,k}(p^n_{i,j,k} - p^n_{wf,i,j}) \\
&= \frac{\partial}{\partial k_{x,i,j,k}} \left[WI^n_{i,j,k}(p^n_{i,j,k} - p^n_{wf,i,j}) \right] \\
&= (p^n_{i,j,k} - p^n_{wf,i,j}) \frac{\partial}{\partial k_{x,i,j,k}} (WI^n_{i,j,k}),
\end{aligned} \tag{3.143}$$

where $\frac{\partial}{\partial k_{x,i,j,k}}(WI_{i,j,k})$ can be calculated using Eq. 3.110.
Similarly, if $i^{'} = i$, $j^{'} = j$ and $k^{'} = k$ for some k with $l_1 \leq k \leq l_2$

$$\begin{aligned}
&\frac{\partial}{\partial s_{skin,i^{'},j^{'},k^{'}}} \sum_{k=l_1}^{l_2} WI^n_{i,j,k}(p^n_{i,j,k} - p^n_{wf,i,j}) \\
&= (p^n_{i,j,k} - p^n_{wf,i,j}) \frac{\partial}{\partial_{s_{skin,i,j,k}}} (WI_{i,j,k})
\end{aligned} \tag{3.144}$$

where $\frac{\partial}{\partial_{s_{skin,i,j,k}}}(WI_{i,j,k})$ can be calculated using Eq. 3.112.

3.11 Sensitivity Examples

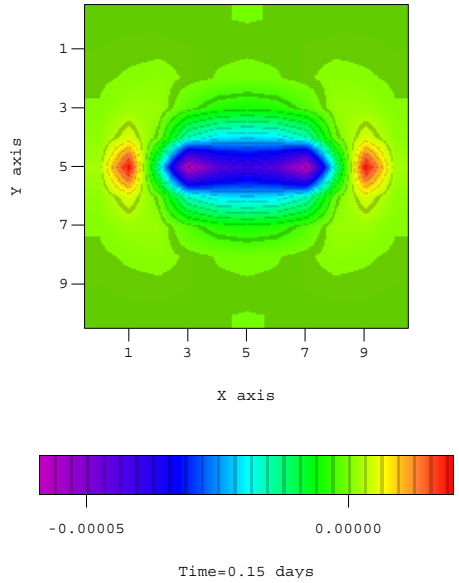
In this section, we compare the sensitivity of wellbore pressure to the rock property fields for a number of cases. We consider both 2D and 3D cases for homogeneous and heterogeneous rock property fields and compare the results obtained by the adjoint method to those obtained by the direct (finite-difference) method.

3.11.1 Case one:

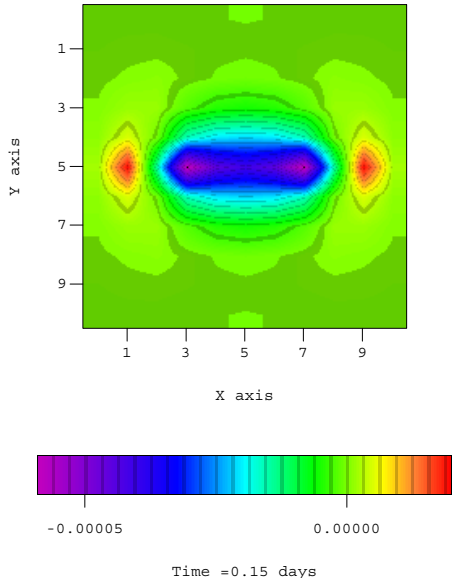
We consider a two-dimensional case with a 11×11 grid. An active well located at $(3, 6)$ is produced at a constant rate of 940,000 scf/day. The second well is an observation well located at $(9, 6)$. The areal dimensions of the reservoir are 1100 ft $\times$ 1100 ft and we assume uniform thickness with $h = 10$ ft. Since sensitivities are more difficult to interpret physically when the reservoir properties are non-uniform, we consider uniform isotropic permeability and uniform porosity. The mean value of ln(k) is 4.0 (k=54.5) and mean porosity is 0.25. The initial reservoir pressure is assumed to be 3,230 psi and specific gravity of gas is 0.75.

3.11.2 Sensitivity to Permeability Field:

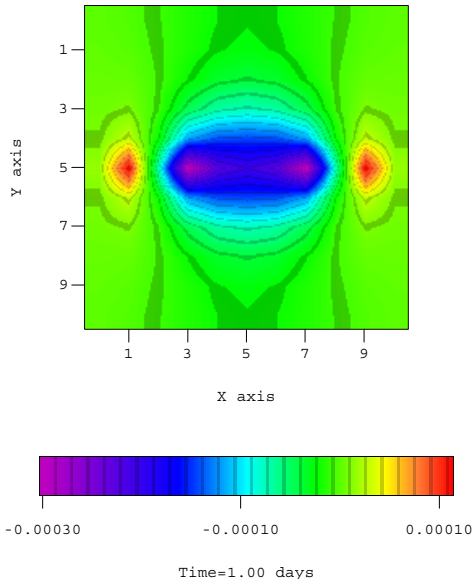
We compare the sensitivity of pressure at the observation well with respect to permeability k at two different times. The first time corresponds to $t = 0.15$ days and the second time to $t = 1.00$ days. Fig. 3.1 shows the sensitivity coefficients obtained by the two methods. We see the results from adjoint method are in excellent agreement with those obtained from the direct method. Since the reservoir is homogeneous in this case, the sensitivity coefficients are quite symmetric around the wells. We can also see that as time increases the wellbore pressure becomes more sensitive to permeabilities at a greater distance from the well. Between the two wells, the sensitivity coefficients are negative. This means that an increase in the permeability at one of these gridblocks will cause a decrease in the pressure at the observation well.



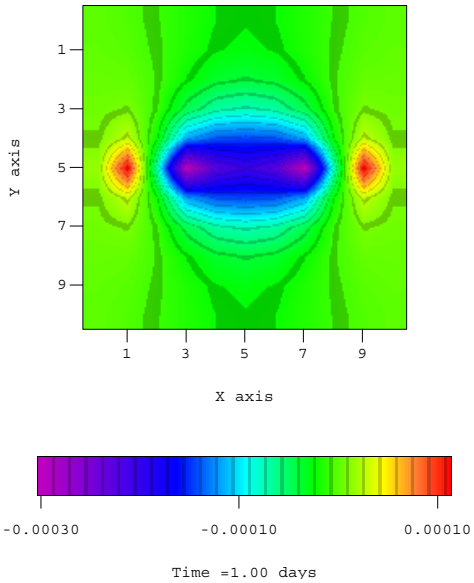
(a) Direct Method



(b) Adjoint Method



(c) Direct Method



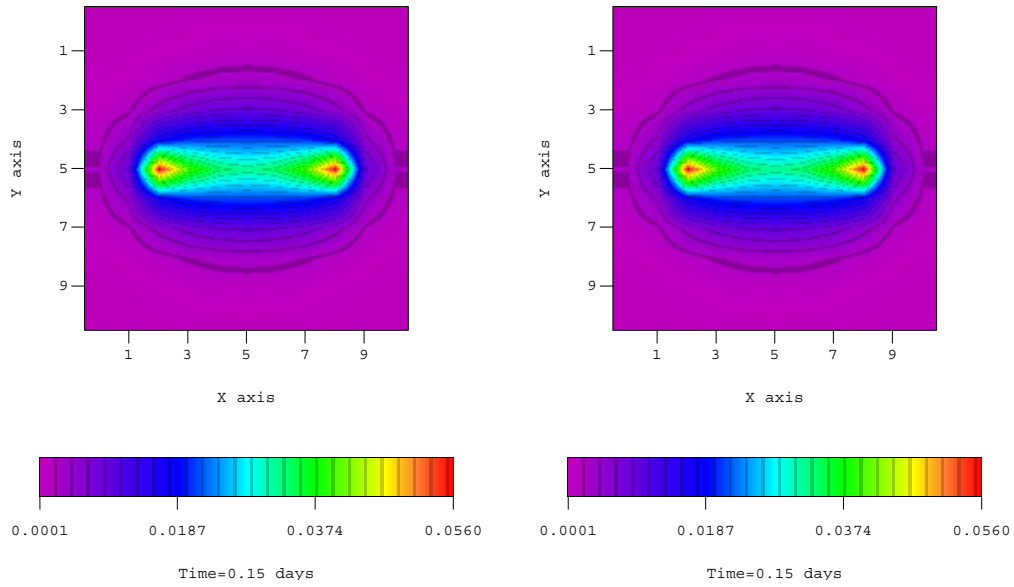
(d) Adjoint Method

Figure 3.1: Comparison of observation well pressure sensitivity to homogeneous permeability field.

Physically, a higher permeability in the interwell region causes the pressure to drop sooner at the observation well, thus resulting in a lower pressure at a given time.

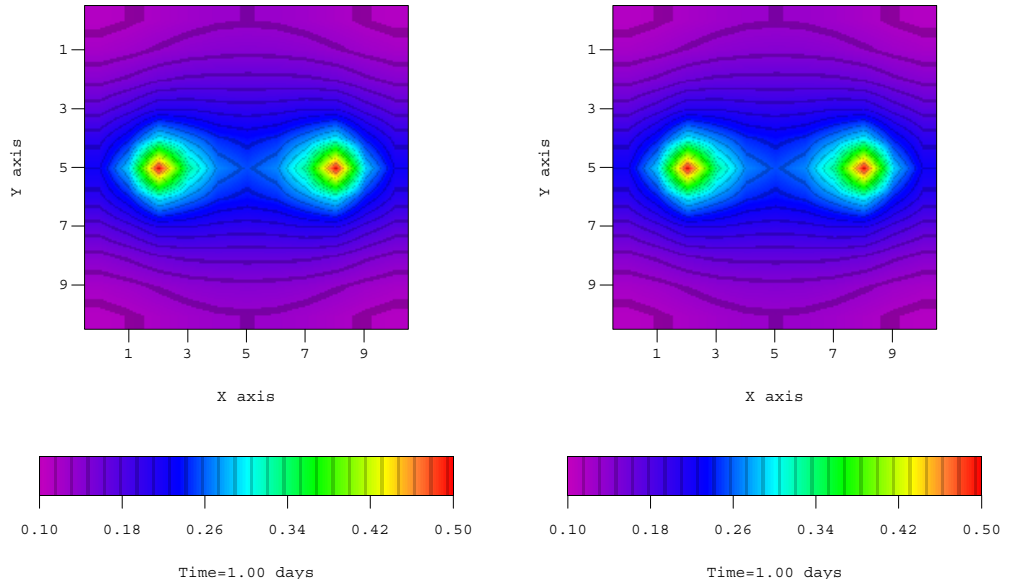
3.11.3 Sensitivity to Porosity Field:

In this we compare the sensitivity of the observation pressure well obtained by adjoint method to those obtained by the direct method; Fig. 3.2 shows the comparison. We can see that the two results are in excellent agreement. Also, as expected the sensitivities are quite symmetric around the wells. Note that all porosity sensitivity coefficients are positive. Increasing the porosity at any gridblock means there is more fluid in the system for pressure support which results in an increase in pressure.



(a) Direct Method

(b) Adjoint Method



(c) Direct Method

(d) Adjoint Method

Figure 3.2: Comparison of observation well pressure sensitivity to homogeneous porosity field.

3.11.4 Case Two:

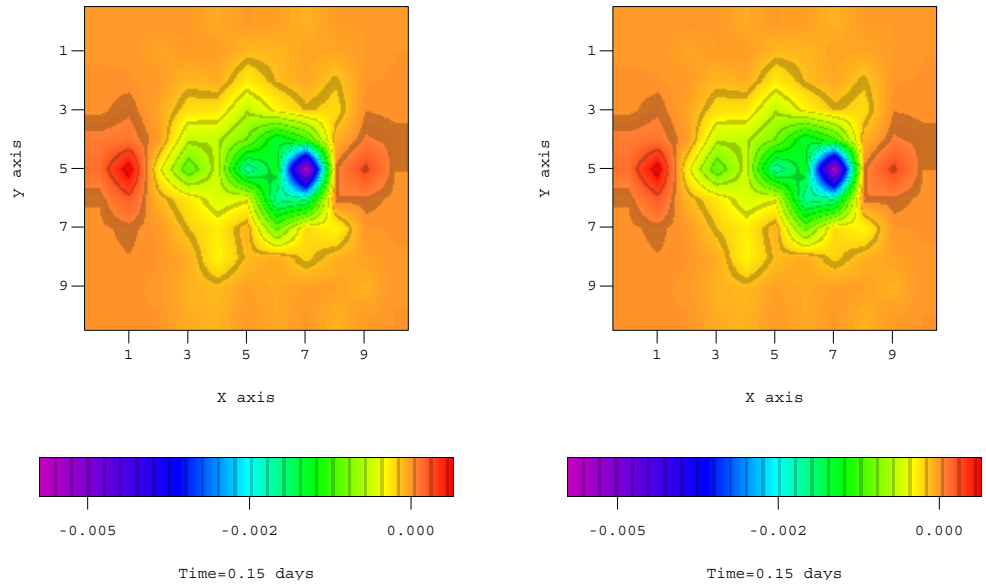
This case pertains to a 3D heterogeneous reservoir with a $11 \times 11 \times 4$ grid. The areal grid sizes are 100 ft $\times$ 100 ft and each Δz is equal to 10 ft. We consider an anisotropic heterogeneous permeability field and non-uniform porosity field. The mean value of ln(k) is 4.0 and mean porosity value is 0.25. We assume a variogram range (correlation length) of six gridblocks in the x direction, 4 gridblocks in the y direction and 2 gridblocks in the z direction. The "true" permeability and porosity fields are obtained by generating an unconditional realization from an exponential anisotropic variogram. For the simplicity of interpreting the results, we consider two wells. An active well is located at areal gridblock $(3, 6)$ and an observation well is located at $(9, 6)$.

3.11.5 Sensitivity to Horizontal Permeability Field:

In this case, all the layers are perforated to produce. We compare two cases one at $t = 0.25$ days and the other at t =1.00 days. Fig. 3.3 shows the comparison of observation wellbore pressure sensitivity to the heterogeneous permeability field obtained by adjoint method and the direct (finite-difference) method. The sensitivity coefficients follow the same trend as in the homogeneous case, the only difference being that they are no longer symmetric around the well. The results plotted are for the first layer of a four layer model. In this case, the two sets of result are also in excellent agreement.

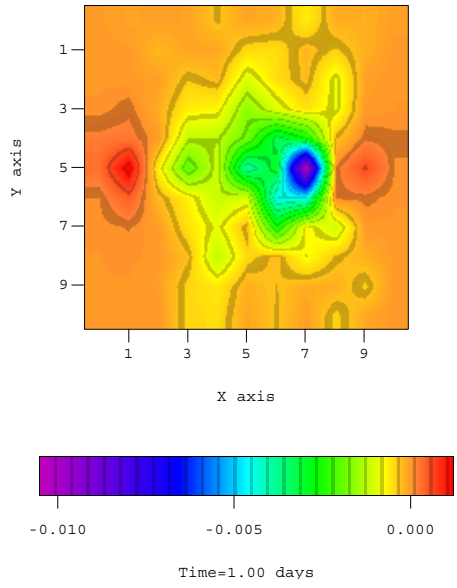
3.11.6 Sensitivity to Porosity Field:

Fig. 3.4 shows the comparison of observation wellbore pressure sensitivity to the heterogeneous porosity field obtained by adjoint method and the direct (finite-difference) method. The sensitivity coefficients are positive as in the homogeneous case for the same reason. The results plotted are for a the first layer of a four layer

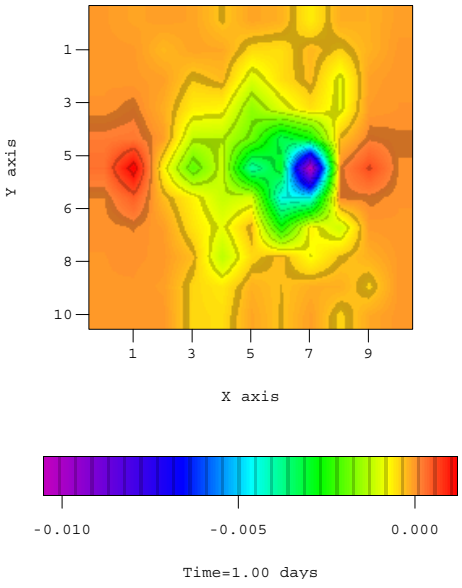


(a) Direct Method

(b) Adjoint Method

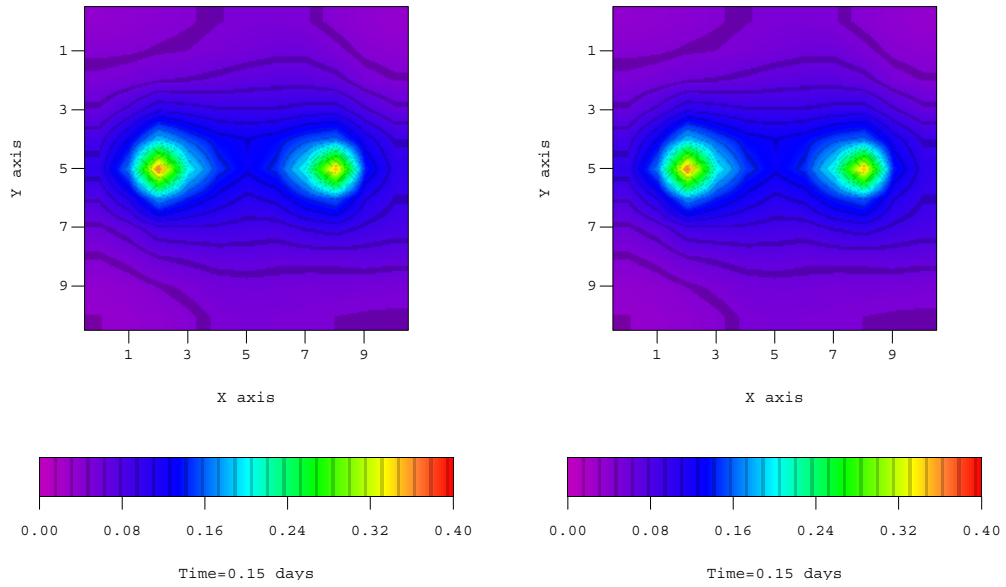


(c) Direct Method



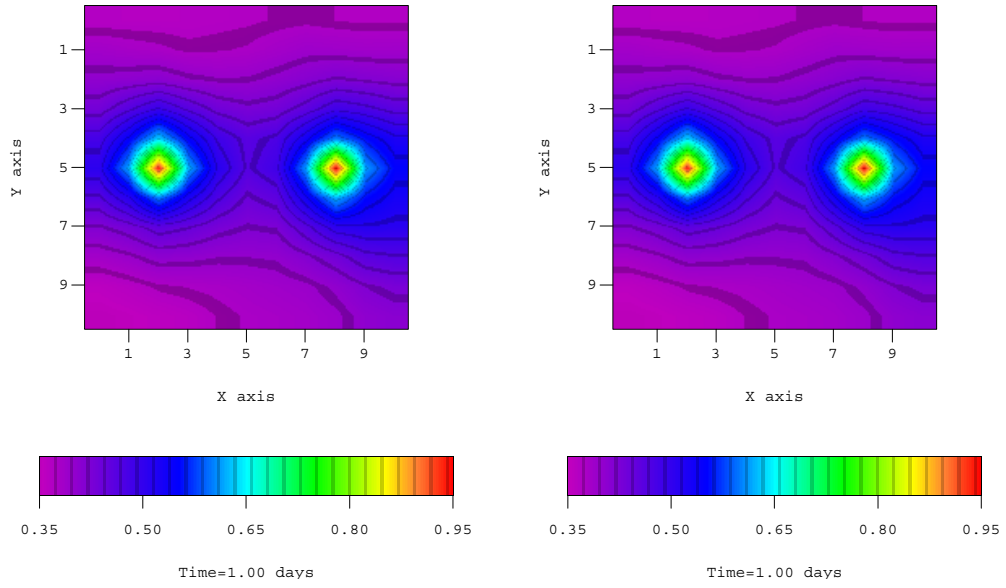
(d) Adjoint Method

Figure 3.3: Comparison of observation well pressure sensitivity to heterogeneous permeability field.



(a) Direct Method

(b) Adjoint Method



(c) Direct Method

(d) Adjoint Method

Figure 3.4: Comparison of observation well pressure sensitivity to heterogeneous porosity field.

model. In this case, the two sets of result are also in excellent agreement.

3.11.7 Sensitivity to Vertical Permeability Field:

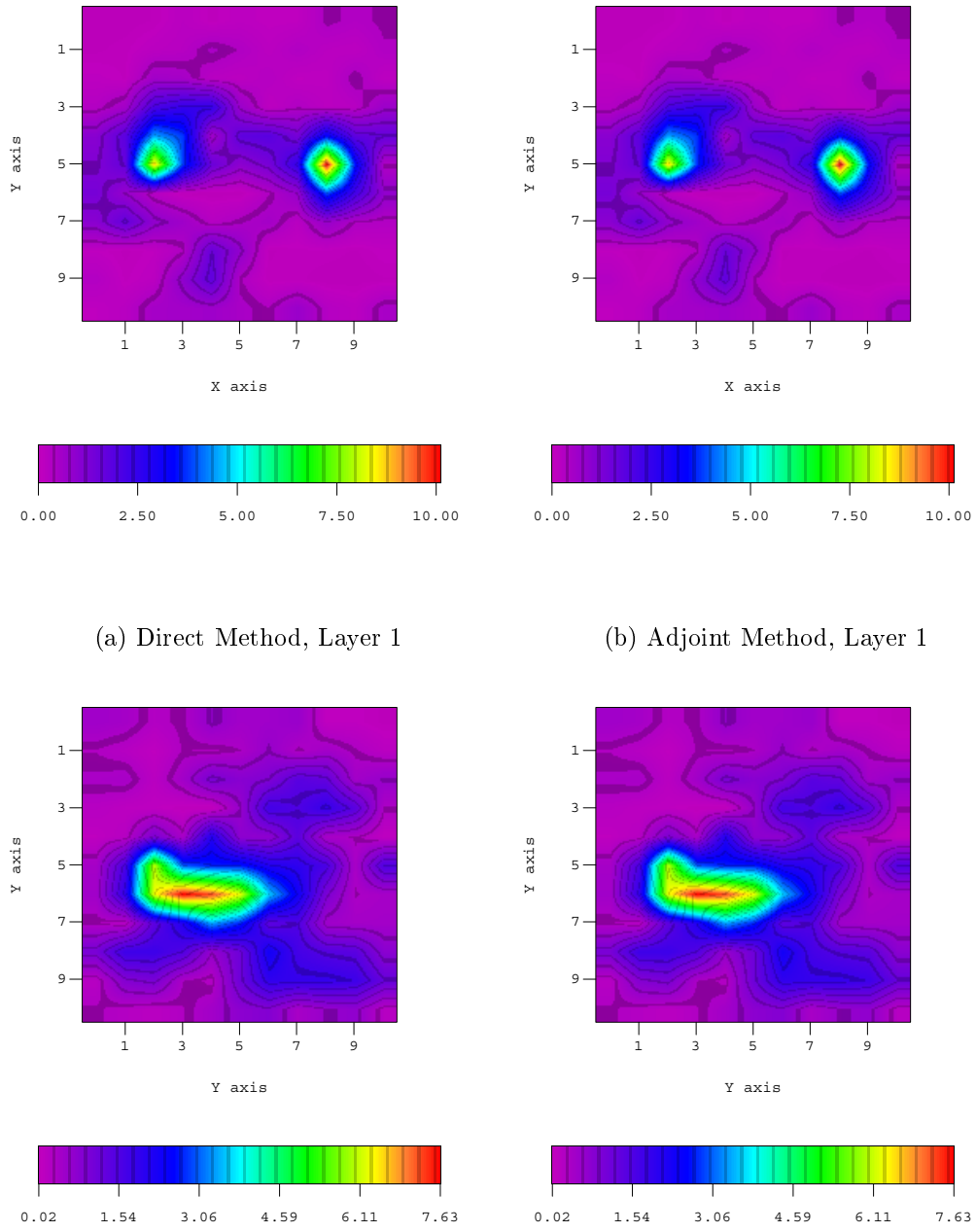
In this case, we compare the sensitivity of wellbore pressure to the vertical permeability field. We consider the same model as in the previous case. The sensitivity of the wellbore pressure to the vertical permeability field is nonzero only if there is vertical flow in the reservoir. So to ensure vertical flow in the reservoir the producing well is completed as a restricted-entry well with only the top layer open to flow and the other three layers closed. The observation well is also completed only with the top layer open.

The results shown in Fig. 3.5 are for the time $t = 1.00$ days for the top layer and the second layer. The vertical flow is controlled by the vertical transmissibilty between the layers. The transmissibility term contains the harmonic average of individual layer k_z and this makes the pressure field sensitive to individual layer k_z in the reservoir. We see that the observation well pressure is much more sensitive to the k_z field around the wells in layer 1. Also note that the sensitivities are positive. This means that if vertical permeability increases, there will be more fluid flow to the top layer thereby increasing the pressure support. In layer 2, the observation well pressure is more sensitive to a region between the two wells. In this case, the results from adjoint method are also in excellent agreement with those from the direct method.

Fig. 3.6 shows the sensitivity of the active well pressure to the vertical field for layer 1 and layer 2. In this case also the results are in very good agreement.

3.11.8 Sensitivity to Skin Factor:

We also compare the sensitivity of the wellbore pressures to the layer skin factors for the above case. Since only the top layer is perforated, the active wellbore pressure should be sensitive only to the skin value of layer 1 and insensitive to the



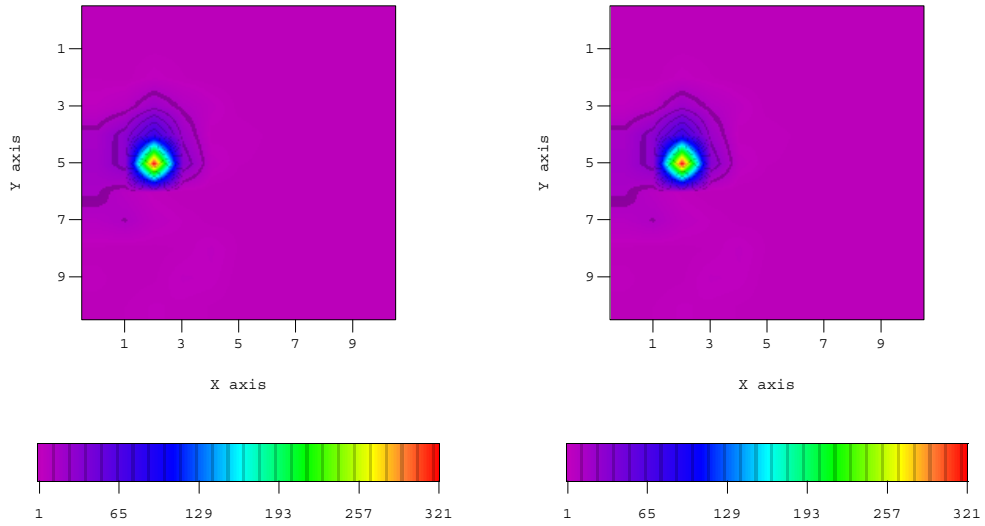
(a) Direct Method, Layer 1

(b) Adjoint Method, Layer 1

(c) Direct Method, Layer 2

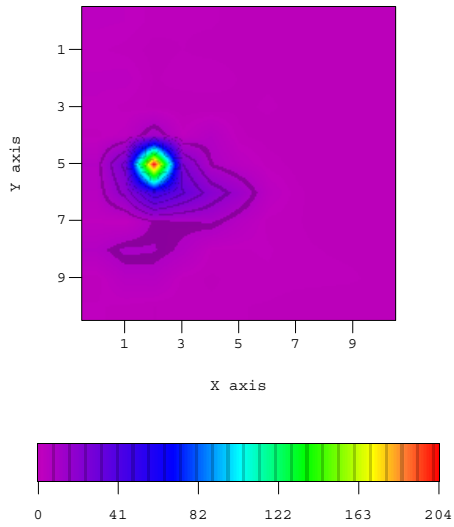
(d) Adjoint Method, Layer 2

Figure 3.5: Comparison of observation well pressure sensitivity to vertical permeability field.

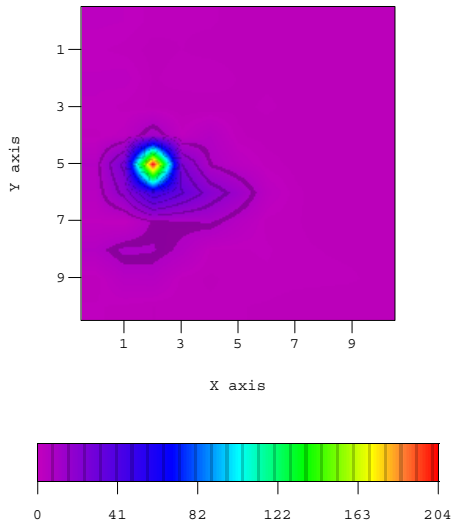


(a) Direct Method, Layer 1

(b) Adjoint Method, Layer 1



(c) Direct Method, Layer 2



(d) Adjoint Method, Layer 2

Figure 3.6: Comparison of active well pressure sensitivity to vertical permeability field.

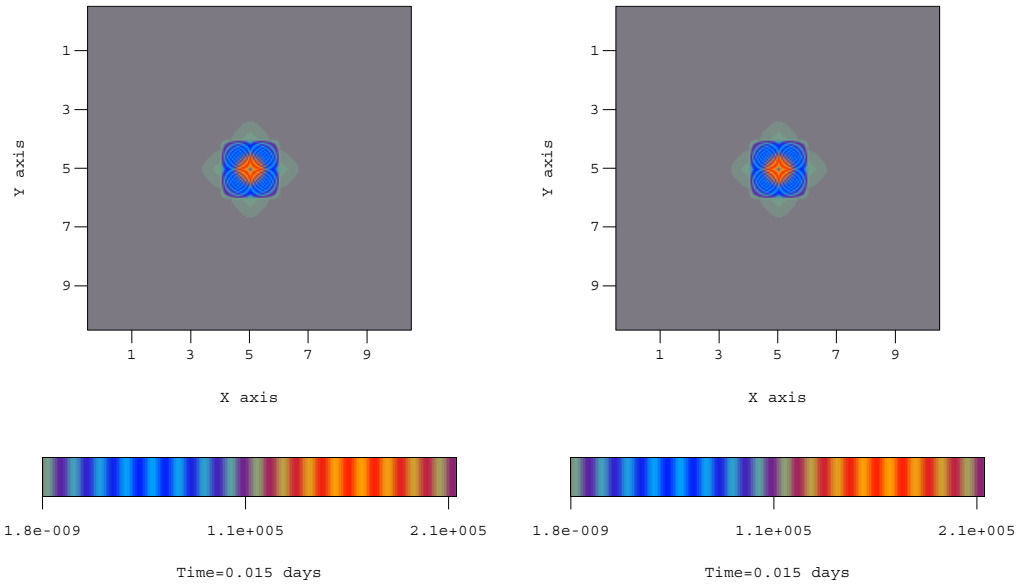
other layer skin values, because in reality, there is no skin value when a layer is not perforated. The sensitivity of the active well pressure at 1.00 days to skin of layer 1, well 1 is -142.73 by direct method and the corresponding value by adjoint method is -143.75. The observation wellbore pressure sensitivity to the layer 1, well 1 skin value is essentially zero by both the methods. The negative sensitivity means that increase in the skin factor value will decrease the wellbore pressure which is physically true because greater pressure drop will occur to overcome the bigger skin.

3.11.9 Case Three:

This case models a 2D homogeneous reservoir with 11×11 areal grid with a single producing well. We compare the sensitivity of flow rate to the permeability field. The permeability field is considered as isotropic and homogeneous with the mean ln(k) equal to 4.00 and porosity field is considered as uniform with mean value equal to 0.25.

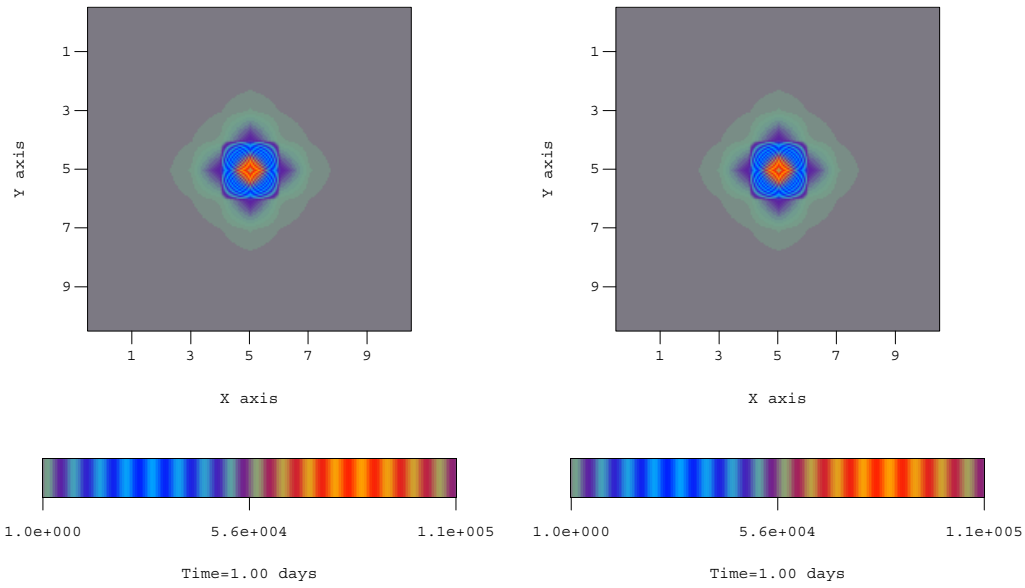
We put the well in constant pressure production at $p_{wf} = 2500$ psi. The sensitivity of flow rate is calculated at two different times, $t = 0.015$ days and $t = 1.00$ days. Fig. 3.7 compares the sensitivity of the well rate to the uniform k_x field. As shown, the two results are in excellent agreement. In the case of the adjoint method, the sensitivity has been calculated by both method 1 and method 2. The two methods give identical results.

Fig. 3.8 shows the corresponding figures for the uniform porosity field.



(a) Direct Method

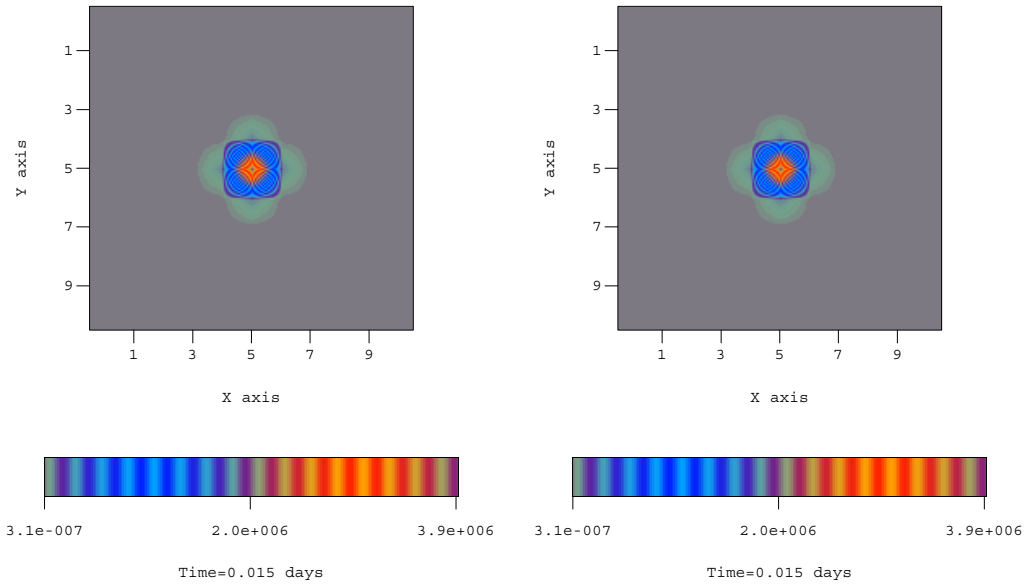
(b) Adjoint Method



(c) Direct Method

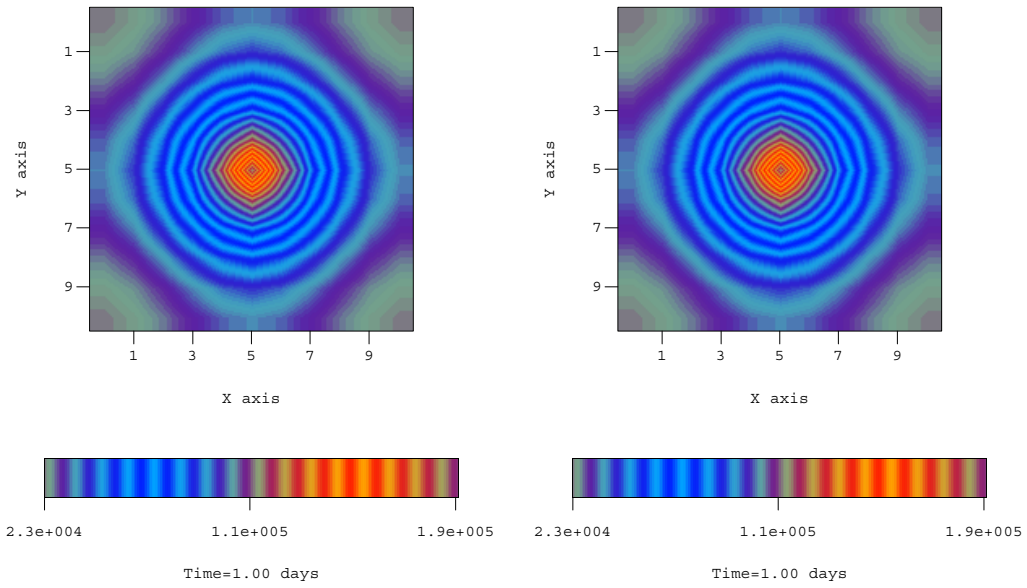
(d) Adjoint Method

Figure 3.7: Comparison of flow rate sensitivity to 2D homogeneous permeability field.



(a) Direct Method

(b) Adjoint Method



(c) Direct Method

(d) Adjoint Method

Figure 3.8: Comparison of flow rate sensitivity to 2D homogeneous porosity field.

CHAPTER IV

CONDITIONING ROCK PROPERTY FIELDS TO PRODUCTION DATA: COMPUTATIONAL EXAMPLES

Here, the inverse method discussed in Chapter II is applied to generate the maximum a posteriori (MAP) reservoir model and to generate realizations of porosity and permeability fields that are conditional to the well-test pressure data for a single-phase flow gas reservoir. In this chapter, we present computational examples for generating the MAP estimate for a two-dimensional reservoir model and for generating realizations of rock property fields for a three-dimensional reservoir model conditional to production data. We apply both the Gauss-Newton and the conjugate gradient optimization technique in the two examples and analyze the performance of the two methods as optimization algorithms.

4.1 Generating the MAP Estimate

The maximum a posteriori estimate is the model, which is the "most probable", although it is generally too smooth to be a "plausible model". The MAP model looks like a smoothed version of the true reservoir model and is found by minimizing the objective function of Eq. 2.5 which is repeated here as

$$O(m) = -\frac{1}{2}\left[(m - m_{prior})^T C_M^{-1}(m - m_{prior}) + (g(m) - d_{obs})^T C_D^{-1}(g(m) - d_{obs})\right], \tag{4.1}$$

where m is the vector of model parameters (gridblock porosities and log-permeabilities), d_{obs} is the vector of observed pressure data, and $g(m)$ is the vector of data that are calculated using m in the reservoir simulator. As discussed in Chapter II, both the

Gauss-Newton and conjugate gradient methods are used to minimize the above objective function. In the case of the Gauss-Newton method, since the number of unknowns is much larger than the number of conditioning data, we use the form given by

$$\begin{aligned} \delta m^{l+1} &= -m^l + m_{prior} - C_M G_l^T (C_D + G_l C_M G_l^T)^{-1} \\ &\quad \times \left[g(m) - d_{obs} - G_l \left(m^l - m_{prior} \right) \right] \end{aligned} \tag{4.2}$$

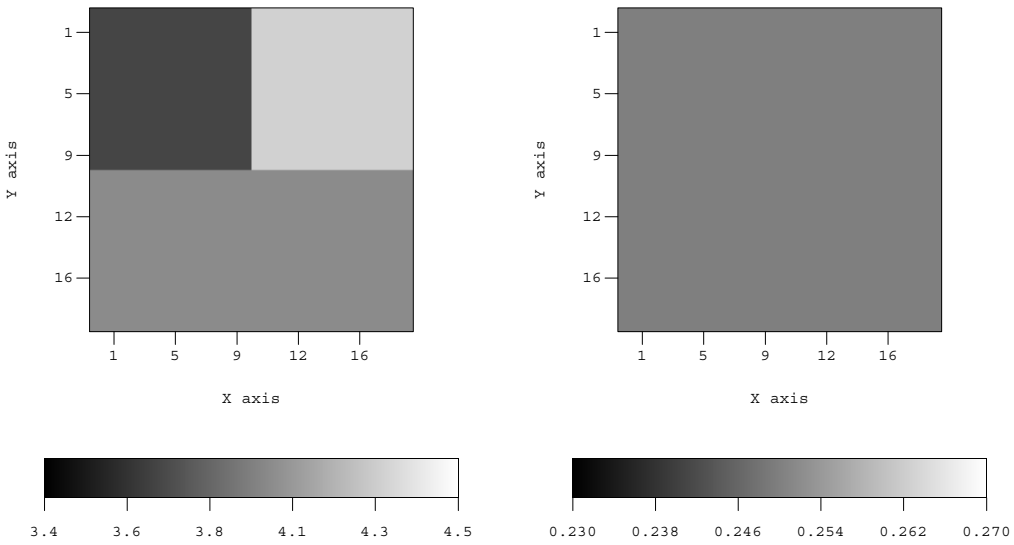
and

$$m^{l+1} = m^l + \mu_l \delta m^{l+1}. \tag{4.3}$$

In the case of the conjugate gradient algorithm, we use both the preconditioned algorithm (with inverse of the prior covariance matrix as the preconditioning matrix) and the method without preconditioning. The objective is to compare the final model estimate obtained by these two methods. Also the effect of restarting on convergence of the conjugate gradient method is studied and presented in the context of this example.

4.1.1 Three-zone Reservoir

The true model in this example, from which the wellbore pressure data is generated is composed of a reservoir of three constant permeability zones constructed on a $20 \times 20 \times 1$ grid. We consider a two-dimensional case for the simplicity of interpreting the results of the a posteriori model in terms of capturing the true rock property field attributes. The true permeability and porosity fields are shown in Fig. 4.1. The log permeability is 3.7 (40.44 md) in the upper left quadrant, 4.3 (73.7 md) in the upper right quadrant and 4.0 (54.59 md) in the lower two quadrants. The porosity field is uniform with $\phi = 0.25$. We consider three wells located at (5,5), (15,5) and (10,15) in the reservoir. Each of the wells is produced at 6,000,000 scf/day for a period of three days to generate the true pressure data. These pressure data are used as the conditioning data after adding measurement errors with mean zero



(a) True permeability field.

(b) True porosity field.

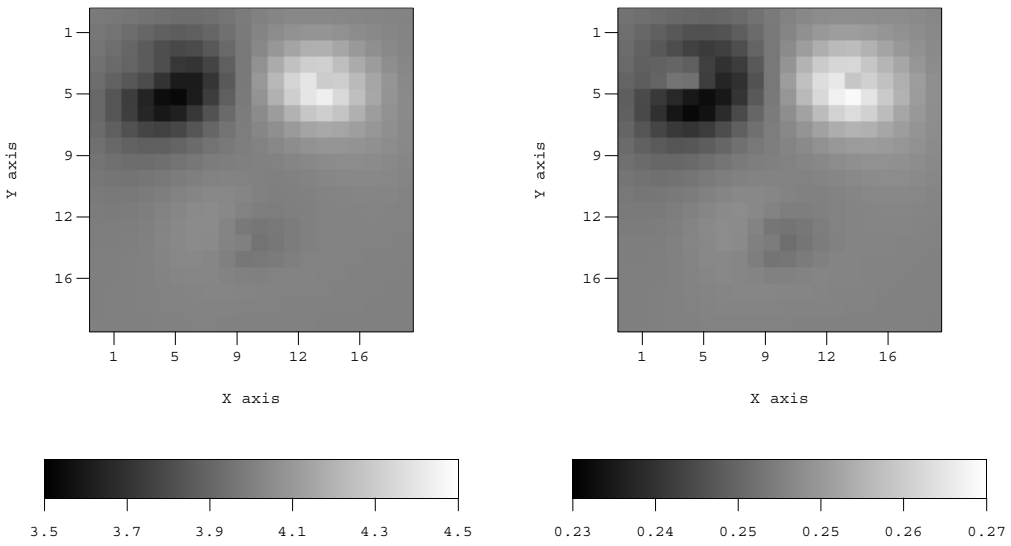
Figure 4.1: The true log permeability(left) and porosity (right) for the three-zone problem.

and variance equal to 1.00 psi^2. The prior mean of $\ln(k)$ is 4.0 and the prior mean of porosity is 0.25. The prior covariance matrix is generated using the variogram information. An anisotropic exponential variogram for $\ln(k)$ is used with the range in the x-direction equal to 600 ft and the range in the y-direction equal to 400 ft. The variance of $\ln(k)$ (sill of the variogram) is specified as $\sigma_k^2 = 0.5$. The anisotropic variogram for porosity is identical to the one for $\ln(k)$ except the variance for porosity is specified as $\sigma_\phi^2 = 0.0025$. We assume a strong prior correlation between porosity and permeability with the correlation coefficient ($\rho_{k,\phi}$) equal to 0.70. The skin factors are assumed to have very small prior variance equal to 0.001, since at this time we are not aiming at resolving the skin factors. (But skin factors can also be estimated.) The reservoir parameters and well specifications used in this example are summarized in table Table 4.1. Starting with this prior model we estimate the most probable model (MAP) conditioned to the observed pressure data. Features in the MAP estimate from the prior model result from the process of conditioning the model to the observed pressure data.

Fig. 4.2 shows the MAP estimates for the ln(k) and porosity fields obtained by the CG and Gauss-Newton optimization algorithms. In all the cases, the maximum a posteriori estimates give smooth estimates of the model parameters. Also we can see that around the well locations in all quadrants, the estimates shows the value of the model parameters closer to the true value. At the exact well locations, the estimated values of porosity and permeability obtained by both the CG and Gauss-Newton method are essentially identical to the true values. Since in the third and fourth quadrant, the prior mean value is equal to the true value of ln(k) and porosity, we do not see much change made to the prior model in these areas during the conditioning process. But significant correction is made in the first (upper left) and second (upper right) quadrant during the conditioning process. Also because of the strong correlation assumed between the porosity and permeability in the gridblocks, we can see correlated changes in the grid block values of porosity and permeability.

Mean Porosity	0.25
Mean Permeability	4.0
Porosity Variance	0.0025
Permeability Variance	0.5
Variogram Range in x-direction	600 ft
Variogram Range in x-direction	400 ft
Correlation coefficient between ln(k) and ϕ	0.70
Well 1, Location (5,5)	6 MMscf/day
Well 2, Location (15,5)	6 MMscf/day
Well 3, Location (10,15)	6 MMscf/day
Initial Reservoir Pressure	3230 psia
Specific Gravity of Gas	0.75
Reservoir temperature	700 ^0R

Table 4.1: Rock and fluid properties for three-zone reservoir.



(a) MAP Estimate of ln(k) by CG.

(b) MAP estimate of porosity by CG.

(c) MAP Estimate of ln(k) by Gauss-Newton.

(d) MAP estimate of porosity by Gauss-Newton.

Figure 4.2: Comparison of MAP estimates obtained by CG and Gauss-Newton.

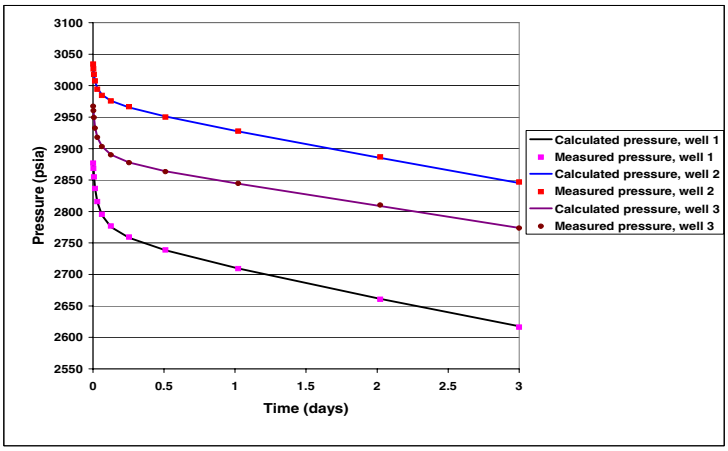
By comparing the MAP estimates we can see that both optimization algorithms give similar MAP models. However the correction to the prior model seems to be slightly more spread out in the case of Gauss-Newton method. However the slight difference in two models does not seem to affect the pressure match obtained. Fig. 4.3 shows the pressure match obtained with MAP estimation obtained from the two algorithms. It can be seen that the pressure match obtained is quite satisfactory. The data mismatch, defined by

$$e_d = \Big(\frac{1}{N_d}\sum_{i=1}^{N_d}(d_i - d_{uc,i})\Big)^{1/2}, \tag{4.4}$$

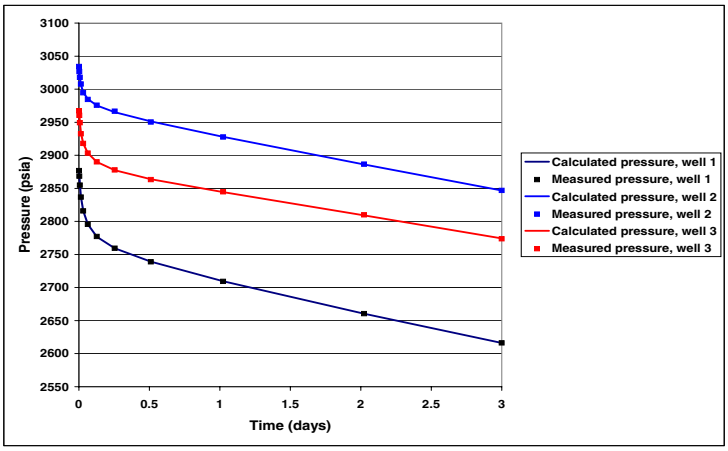
is 0.5899 psi based on the Gauss-Newton results and 0.5504 psi based on the CG results.

4.1.2 Convergence Comparison Between Gauss-Newton and CG

In this section the convergence rate between the two algorithms is compared. The convergence criteria in both the algorithms is defined as $\frac{O(m^l)-O(m^{l+1})}{O(m^l)+10^{-14}} \leq 10^{-5}$, where $O(m^l)$ and $O(m^{l+1})$ are the values of the objective function at the previous and current iterations, respectively. Fig. 4.4 compares the convergence rate of the Gauss-Newton and the conjugate gradient algorithms. Note that Gauss-Newton method converged rapidly, whereas conjugate gradient converged much more slowly. Precise comparisons are given in Table 4.2. Initially the conjugate gradient was restarted after every five iterations. In this case, a detailed examination of the results showed that CG was restarted by this explicit control (see section 2.6, Important remarks) 20 times. So in the next exercise, we set the maximum number of iteration allowed before restarting to 50 and this resulted in a significant improvement in the convergence rate; see Table 4.2. Restarting with a fewer number of number of iterations makes the conjugate gradient method behave more like the steepest descent algorithm and hence the convergence becomes slower. We further increased the maximum number of iterations allowed before restarting to 100 and it yielded almost the same result as in



(a) Pressure match obtained by CG.



(b) Pressure match obtained by Gauss-Newton.

Figure 4.3: Pressure match obtained by CG and Gauss-Newton algorithms.

Optimization Method	Number of Iterations required for convergence	Minimum value of the Objective Function
Gauss Newton	7	10.9080
CG with restarting after 5 iterations	149	7.5853
CG with restarting after 50 iterations	97	7.1912
CG with restarting after 100 iterations	95	7.1936

Table 4.2: Performance Evaluation of Gauss-Newton and CG.

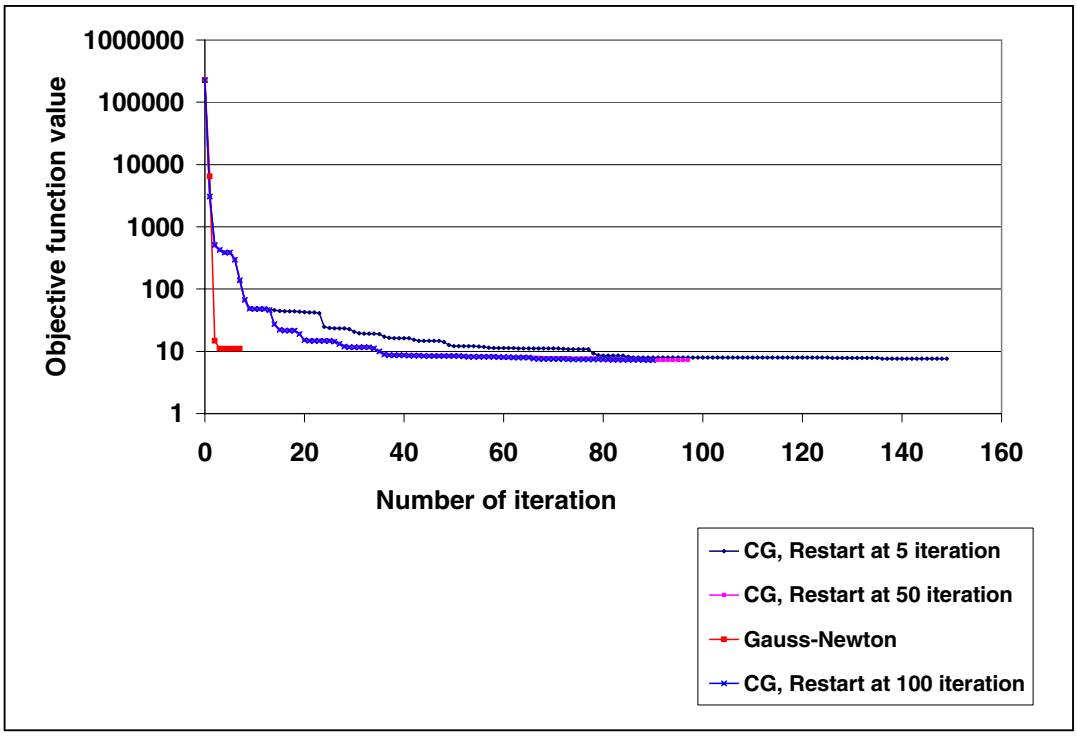


Figure 4.4: Convergence comparison by Gauss-Newton and conjugate gradient.

case of 50 maximum iterations. The convergence summary is given in Table 4.2. The convergence properties are not highly sensitive to the number of iterations allowed before restarting. Also note that in the last trial (where maximum iteration allowed is 100), the CG did not get restarted by the explicit control at all, i.e., it converged before reaching the maximum iteration limit.

For this particular example, the Gauss Newton converges in far fewer iterations than the conjugate gradient method. This behavior is typical although, as we will see later, there exist cases where the conjugate gradient method also converges quite rapidly. Although Gauss-Newton converges faster, the computational work per iteration is much greater than in the conjugate gradient method. This is because of the fact that in the Gauss-Newton algorithm we have to find the sensitivity of each conditioning data to each model parameter. So we have to solve N_d adjoint systems where N_d is the number of conditioning data being used. In this case N_d is equal to 36. But in case of the conjugate gradient method, we have to find only the gradient of the whole objective function, which requires the solution of only one adjoint system of equations.

4.1.3 Computational Comparison Between Gauss-Newton and CG

Here we present a comparison of the computational expense involved for the two methods in this exercise. We assume that since the size of the matrix system involved in the adjoint system is same as the simulator equations, so solving one adjoint system of equations is roughly equivalent to one simulation run. (Because we are actually solving the adjoint system with N_d right hand sides.) Gauss-Newton requires one simulation run to solve for the pressure field, N_d adjoint solutions to construct the sensitivity matrix and one simulation run to evaluate the objective function for each iteration and extra simulation runs for the reduced step size (if it occurs in any of the iterations). In the conjugate gradient method we need one simulation run to solve for the pressure field, one simulation run and one adjoint

solution to get the step size and one simulation run to calculate the objective function for each iteration and then extra simulation runs for reduced step size whenever applicable.

In this example the Gauss-Newton took 7 iterations with 4 step reductions in the final iteration. So the equivalent number of simulation run will be $(1 + 36 + 1) \times 7 + 4 = 270$. The conjugate gradient method took 95 iterations without any step cut (we consider the best performance by the CG). So the equivalent number of simulation run in this case will be $(1 + 1 + 1 + 1) \times 95 + 0 = 380$. This comparison shows more computational effort involved in case of the CG method.

4.1.4 Comparison of CG With and Without Preconditioning

Preconditioning is the technique for improving the convergence of the CG. The nonlinear CG method can be preconditioned by choosing a preconditioner M that approximates the Hessian matrix and has the property that $M^{-1}r$ is easy to calculate (see the detailed algorithm discussed in chapter II). As mentioned earlier in our work, we use the inverse of the prior covariance matrix as the preconditioning matrix in all the iterations. Fig 4.5 shows the convergence rate for the same example considered above by CG with preconditioning and without preconditioning. The restarting option is set at 50 maximum iterations. It is clear that CG with preconditioning has a faster convergence rate. It is likely that the convergence rate could be further enhanced if we could easily generate a preconditioning matrix which approximates the Hessian more accurately. Here, CG without preconditioning did not converge even after 300 iterations. Fig 4.6 shows the MAP estimate obtained by the CG without preconditioning. Comparing it with the results of CG with preconditioning Fig 4.2, it can be seen that the corrections are more localized around the well in case when no preconditioning is used (more clear in the ln(k) estimate). Preconditioning with the inverse of the C_M makes the correction more smoother.

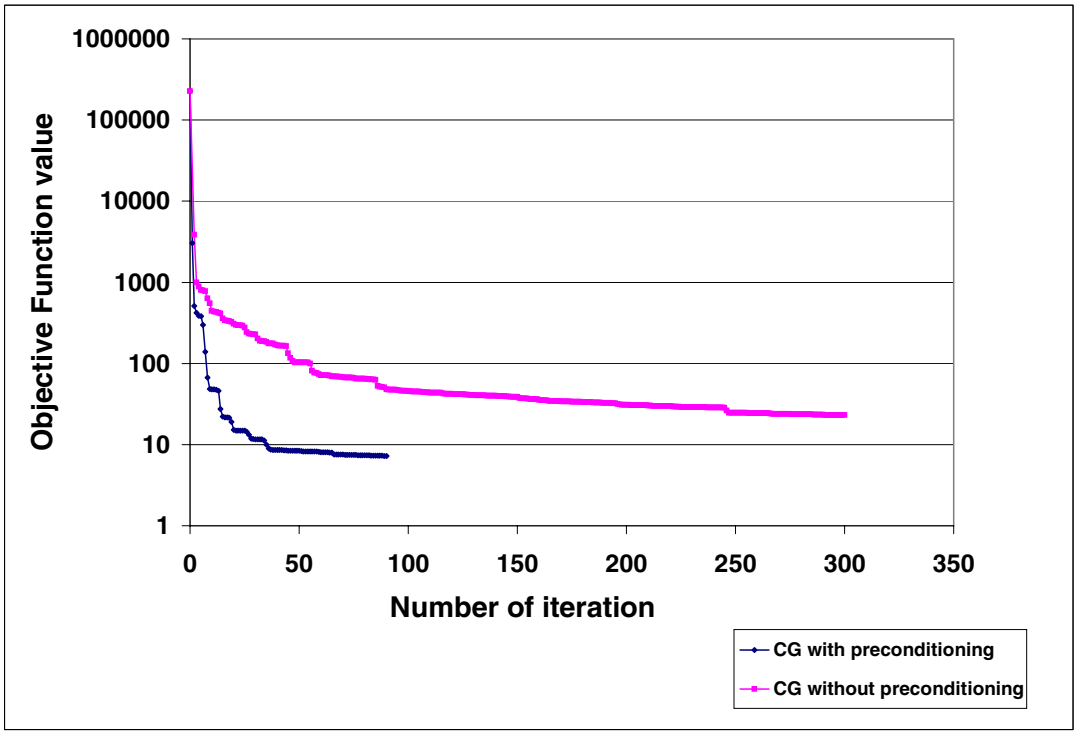
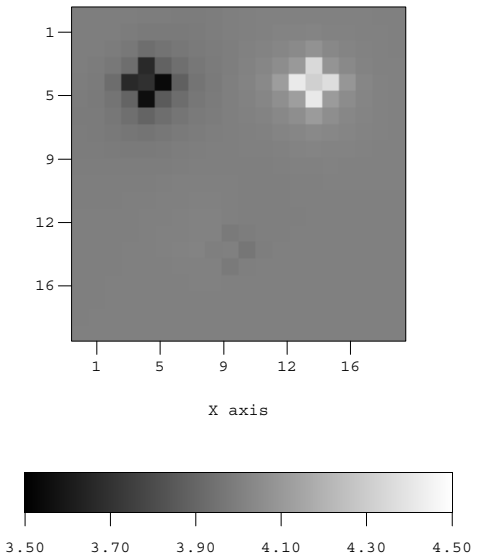
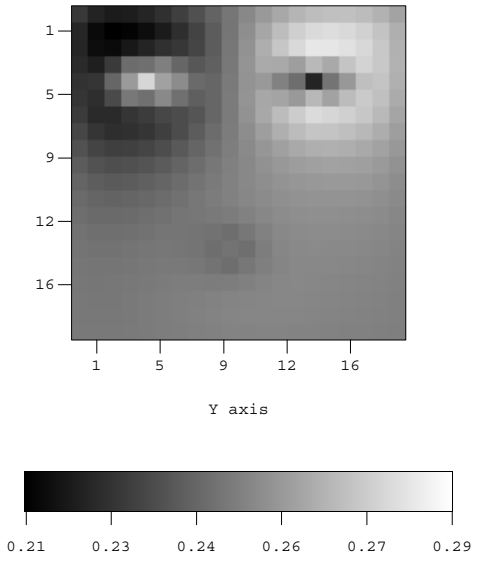


Figure 4.5: Convergence comparison of CG with and without preconditioning.



(a) MAP estimate of ln(k).



(b) MAP estimate of porosity.

Figure 4.6: MAP estimate of ln(k) and porosity for CG without preconditioning.

4.2 Generating Conditional Realizations of Heterogeneous Reservoir

We consider three synthetic examples to investigate the implementation details of the conjugate gradient algorithm and to compare its performance with that of Gauss-Newton method. In each example, we use the two optimization algorithms to compute realizations of permeability and porosity fields conditioned to gas-well pressure data at the wells. In the first example, we consider fully-penetrating wells and in the second example, we consider a restricted-entry well. The third example deals with a five well system.

4.2.1 3D Case with Fully Penetrating Wells

This example pertains to a square reservoir with 20 grid blocks in the x and y directions and 4 grid blocks in the z direction. Each z block is referred to as individual layer while discussing the results. The areal grid is 100 ft by 100 ft and all grid blocks in the z direction have thickness (height) equal to 10 ft. The reservoir thickness is uniform and equal to 40 ft. The reservoir contains two wells. Well 1 is located at areal gridblock $(5, 5)$ and well 2 is located at areal gridblock $(15, 15)$. We consider data obtained from a four day test, during which, each well is either produced at a constant rate or shut-in as shown in Table 4.3. Note that well 1 is shut-in for the first two days and then produced at a rate of 35 MMscf/D. Well 2 is produced at of 40 MMscf/D and then shut-in for a two day buildup test.

The reservoir is areally isotropic i.e., $k_x = k_y = k$. The vertical permeability field k_z is assumed to be uncorrelated with the horizontal permeability field and we generate a realization of all gridblock k_z's. The true distributions of porosity and log permeability fields from which the synthetic production data are generated, represent unconditional realizations of correlated Gaussian random fields with anisotropic spherical variograms. The range of the variogram for $\ln(k)$ is 600 ft in the x-direction and 400 ft in the y-direction. The variance of $\ln(k)$ (sill of the variogram) is specified

Well No.	Location	Flow rate(MM scf/D)	Duration(days)
1	5,5	0	2
1	5,5	35	2
2	15,15	40	2
2	15,15	0	2

Table 4.3: Well flow rates

as $\sigma_k^2 = 0.50$. The anisotropic variogram for porosity is identical to the one for $\ln(k)$ except the variance for porosity is specified as $\sigma_\phi^2 = 0.002$. The variance for $\ln(k_z)$ is assumed to be $\sigma_z^2 = 0.50$. The correlation coefficient between log-permeability and porosity is specified as 0.70. The true $\ln(k)$ and porosity distributions were obtained by unconditional simulation using a prior mean of 4.0 for $\ln(k)$ and a prior mean of 0.25 for porosity. (The value $\ln(k) = 4$ corresponds to $k = 55$ md). The true $\ln(k_z)$ was obtained by a similar procedure except that we specify the prior mean for $\ln(k_z)$ as -2.9. (The value $\ln(k_z) = -2.9$ corresponds to $k_z = 0.055$ md). The prior mean for all well skin factors is set equal to 4.0, which is the true value of skin factor used when generating synthetic production data. We specify the variance for well skin factors as 0.1. At this point in time, our investigation is not oriented towards the resolution of skin factors. He et al. [12] has given a detailed discussion on the simultaneous estimation of skin factors and rock property field from well-test pressure data. The relevant reservoir and fluid propertiesy are summarized in Table 4.4.

As mentioned previously, "true" synthetic well-test pressure data were generated using the true rock property fields and well skin factors as simulator input data and then producing according to the rate sequence shown in Table 4.3. The observed well-test pressure data were then obtained by adding noise to these true values. In generating these observed data, we assumed that pressure measurement errors were independent identically distributed Gaussian random variables with mean zero and

Mean porosity	0.25
Mean permeability, ln(k)	4.0
Mean skin factor	4.0
Porosity variance	0.002
Permeability variance	0.5
Skin variance	0.1
Correlation coefficient between ln(k) and ϕ	0.70
Variogram range in x-direction (ft)	600
Variogram range in y-direction (ft)	400
Initial pressure (psi)	3230
Gas specific gravity	0.75
Reservoir temperature (^0R)	700
Wellbore radius (ft)	0.3

Table 4.4: Well/reservoir data.

variance equal to 1 psi^2. For observed data, we used 22 pressure data at each well.

Starting from unconditional realizations, we generated conditional realizations using the randomized maximum likelihood method. Each conditional realizations requires that we minimize the objective function of Eq. 2.10. For each (m_{uc}, d_{uc}) pair, we minimized the objective function both by the conjugate gradient method and the Gauss-Newton method with restricted step. Fig. 4.7 shows the typical behavior observed for this particular problem. For the case shown in Fig. 4.7, the Gauss-Newton method required 9 iterations to obtain convergence whereas, the conjugate gradient method required 12 iterations. In generating 20 conditional realizations, the number of iterations required to obtain convergence in the conjugate gradient method varied from 12 to 20.

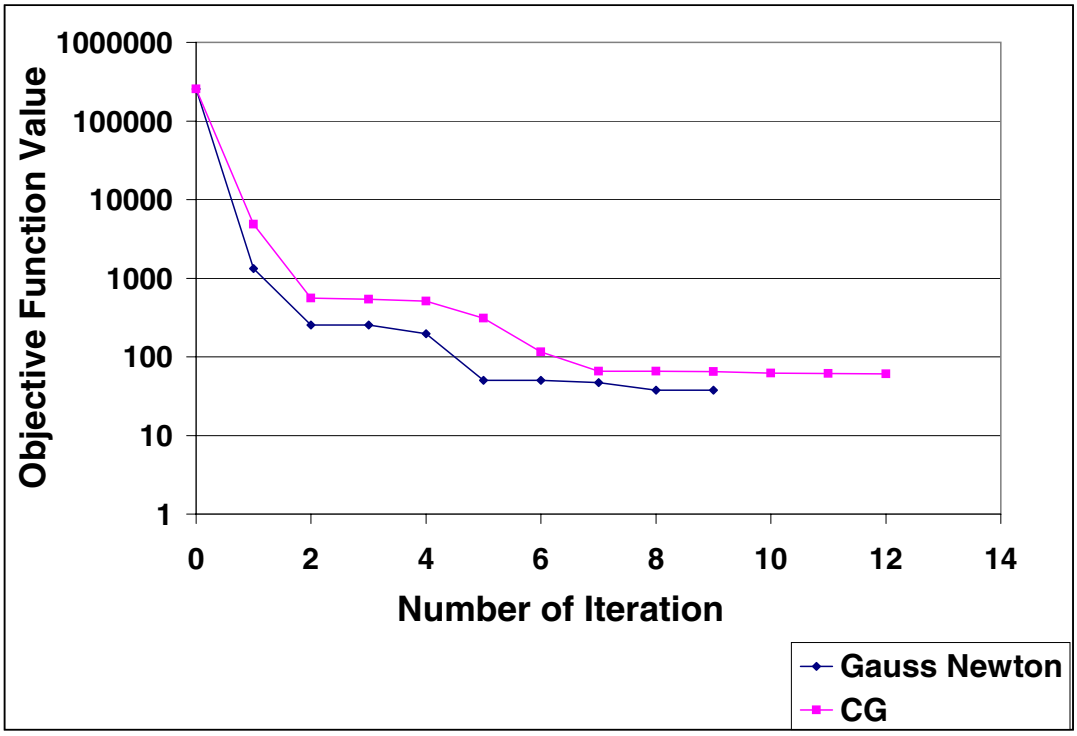


Figure 4.7: Objective function minimization by conjugate gradient and Gauss-Newton method.

As discussed previously, the conjugate gradient method avoids the need to compute individual sensitivity coefficients, and thus, requires significantly less computational work per iteration unless the number of conditioning data is relatively small. Note however, the conjugate gradient method converges to a higher value of the objective function than does the Gauss-Newton method. Although this is troubling, both methods give reasonable matches of the pressure data. The data mismatch, defined by

$$e_d = \Big(\frac{1}{N_d}\sum_{i=1}^{N_d}(d_i - d_{uc,i})\Big)^{1/2}, \tag{4.5}$$

is 0.952 psi based on the Gauss-Newton results and 1.65 psi based on the CG results. The data mismatch, defined by

$$e_d = \Big(\frac{1}{N_d}\sum_{i=1}^{N_d}(d_i - d_{obs,i})\Big)^{1/2}, \tag{4.6}$$

is 0.77 psi based on the Gauss-Newton results and 1.31 psi based on the CG results. In the preceding equations, N_d is the total number of conditioning data, d_i is the ith component of the pressure data calculated from conditional realization, and $d_{uc,i}$ denotes the ith component of the unconditional realization of the data.

Fig. 4.8 compares the pressure data calculated from the conditional realization generated using the CG method for optimization compared with the unconditional realization of the observed data. Note that the agreement is good.

Fig. 4.9(a) shows the true horizontal log-permeability field for layer 1, Fig. 4.9(b) shows the realization obtained from the Gauss-Newton method and Fig. 4.9(c) shows the realization obtained from the conjugate gradient method. At least on this scale, conditional realizations obtained from the two optimization procedures look similar and reflect the same heterogeneity pattern displayed by the true log-permeability field. (Results for porosity and for the other three layers are similar.) The convergence results displayed in Fig. 4.7, however, as well as a more careful examination of results indicates that the two optimization procedures are not converging to exactly

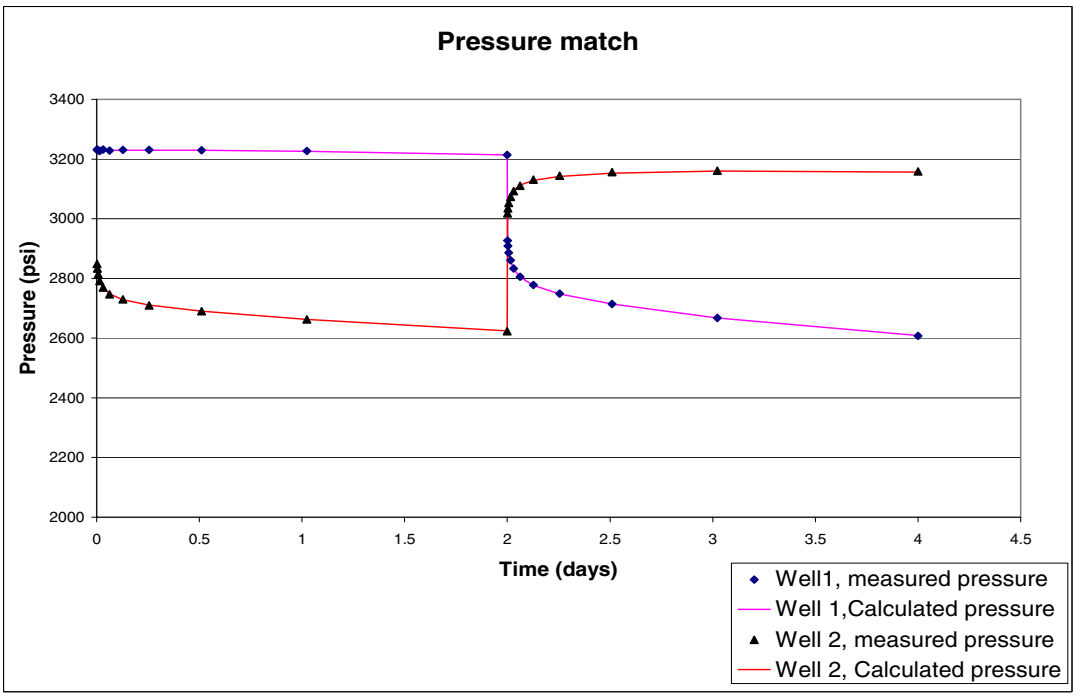
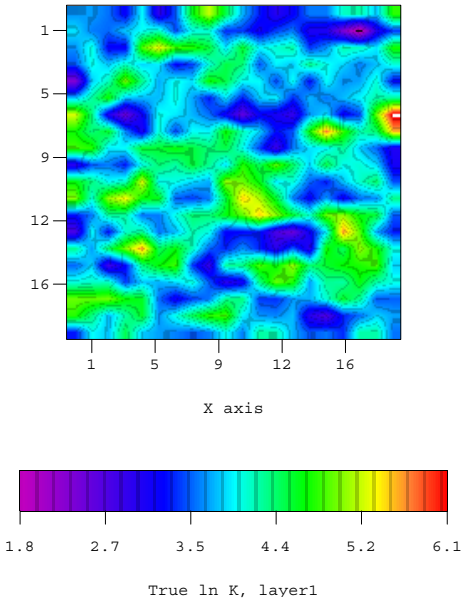


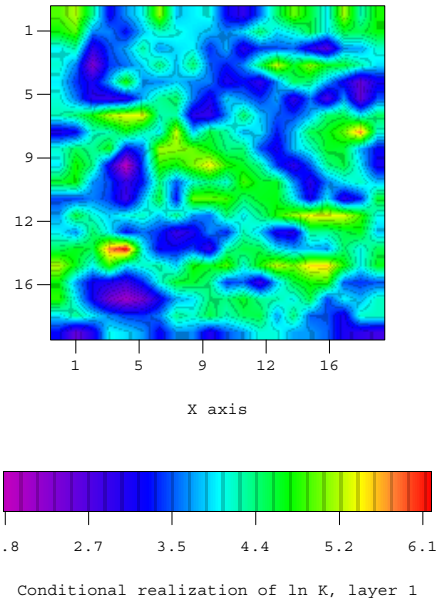
Figure 4.8: Pressure match obtained by conjugate gradient method.

the same minimum. For example, Fig. 4.10(a) shows the layer 3 correction to the unconditional permeability field that was made during the history matching process using the Gauss-Newton method. The largest permeability modification (about 92 md) was made at the gridblock that contains well 2 and changes in permeability tend to decay as the distance from the well increases and changes more than 4 gridblocks away from the well tend to be small. (There are exceptions to the later statement; see gridblock $(10, 16)$ where the change in permeability is about 30 md). Fig. 4.10(b) shows the corresponding changes made to layer 3 gridblock permeabilities when using the conjugate gradient method for optimization. Here the modification to the well 2 gridblock permeability is only about 28.4 md and the change at gridblock $(13, 11)$ is actually slightly larger, about 29.0 md.

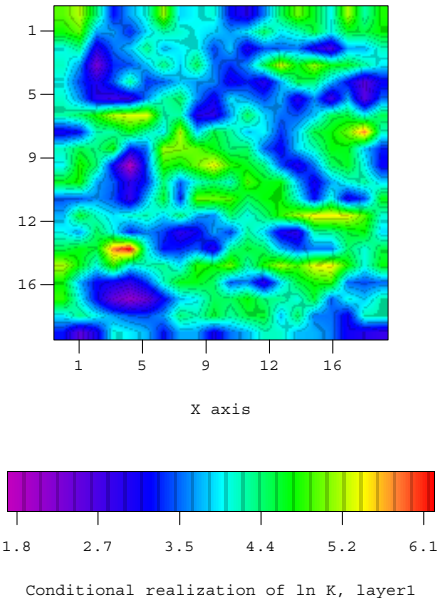
In the example under consideration, all the layers are perforated and vertical flow is negligible. Thus, pressure data is insensitive to vertical permeability and no



(a) True horizontal permeability (ln k).

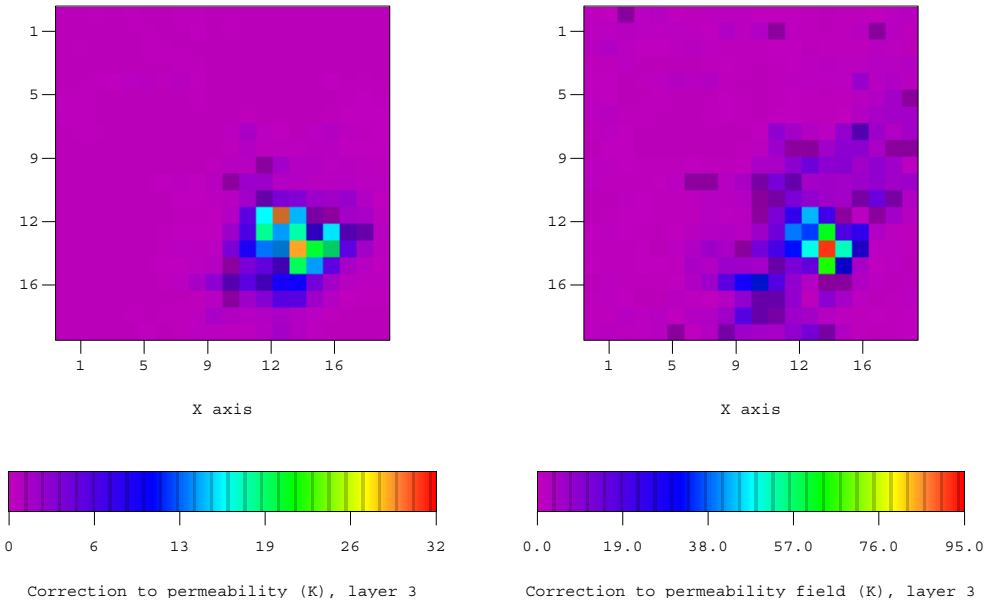


(b) Realization obtained by Gauss-Newton method.



(c) Realization obtained by conjugate gradient method.

Figure 4.9: True permeability field and comparison of conditional realization obtained by Gauss-Newton and conjugate gradient method (for layer 1).



(a) Correction by Gauss-Newton.

(b) Correction by conjugate gradient.

Figure 4.10: Correction to unconditional permeability field by the two methods (layer 3).

correction to the vertical permeability field was made during the optimization process.

4.2.2 Computational Comparison Between Gauss-Newton and CG

In this example the Gauss-Newton took 9 iterations and 3 step reductions in the final iteration. So as discussed in Section 4.3.1, the equivalent number of simulation run is $(1 + 44 + 1) \times 9 + 3 = 417$. The conjugate gradient took 12 iterations and 4 step reductions in the final iteration. So the number of equivalent simulation run is $(1 + 1 + 1 + 1) \times 12 + 4 = 52$. This shows the far less computational expense required in the conjugate-gradient method.

4.2.3 Restricted-Entry Case:

In this case, only the top layer is open to flow. The production profiles for the two wells are as shown in Table 4.5.

As in the previous example, 22 synthetic pressure data were generated at each well to represent the set of observed data. Except for the change in the specified flow rates and pressure data, all other data and prior information is identical to the case considered in previous example.

Fig. 4.11 shows the behavior of both the Gauss-Newton method and the CG algorithm. The results refer to matching the same unconditional realization of the pressure data (d_{uc}) using the same unconditional realization (m_{uc}) of the model as the initial guess when minimizing the objective function of Eq. 2.10. In this case, the Gauss-Newton method converges in 6 iterations, whereas, the CG method requires 97 iterations and again converges to a higher value of the objective function. We implemented CG with a different number of maximum iterations before restart and in this case, we tried 50 and 100 iterations. However in both the cases the results were practically the same.

Although the results of Fig. 4.11 indicate that the CG method converges to

Well No.	Location	Flow rate(MM scf/D)	Duration(days)
1	5,5	0	2
1	5,5	10	2
2	15,15	15	2
2	15,15	0	2

Table 4.5: Specified production rates, restricted-entry example.

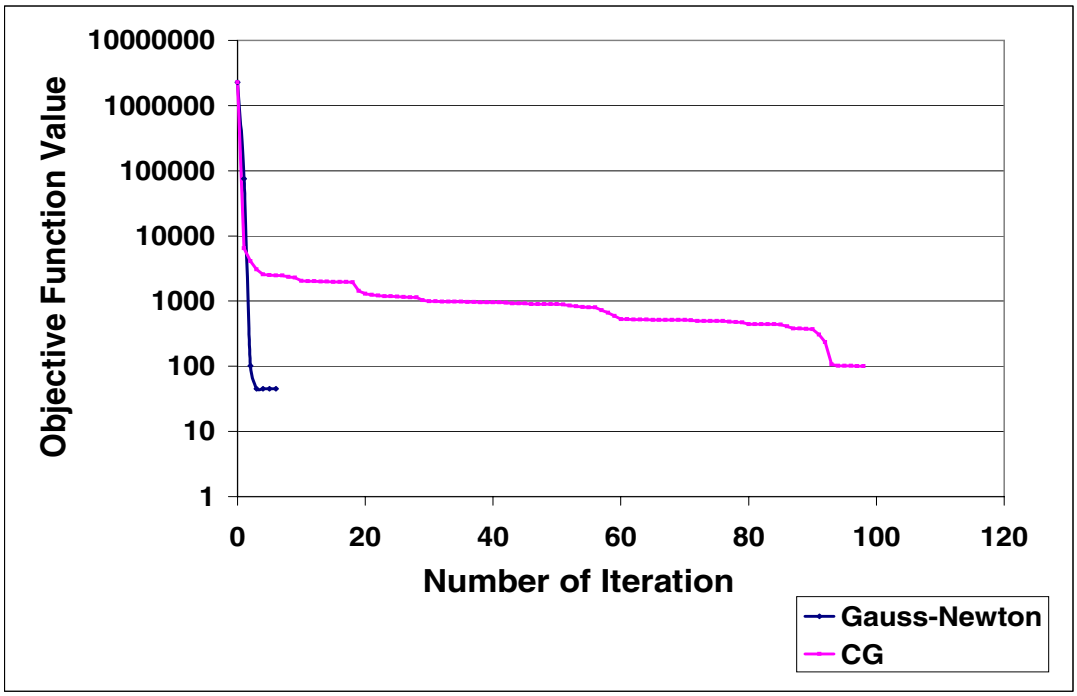


Figure 4.11: Objective function minimization by conjugate gradient and Gauss-Newton method.

a higher value of the objective function than does the Gauss-Newton algorithm, as shown in Fig. 4.12, the pressure calculated from the realization generated with the CG algorithm is in good agreement with the unconditional realization of the pressure data, d_{uc}. The data mismatch, given by Eq. 4.5, is 1.15 psi based on the Gauss-Newton results and is 2.20 psi based on the CG results. Based on Eq. 4.6, the data mismatch is 0.812 psi for the Gauss-Newton results and is 1.90 psi for the CG results.

For this example, vertical flow does occur, so the pressure data is sensitive to vertical permeability. We compare the correction made to the unconditional model during the optimization process because the final conditional realizations obtained by both the methods are similar. Figs. 4.13(a) and 4.13(b), show the correction to the vertical permeability of layer 1 made during the history matching process. Fig. 4.13(a) pertains to results from the Gauss-Newton procedure and Fig. 4.13(b) pertains to the conjugate gradient method. Note that the correction is localized to a relatively small region around the two well blocks which is where we expect vertical flow to be more significant. With the Gauss-Newton method, the maximum correction to k_z is only about 0.06 md but this represents a large percentage change in the k_z value of 0.055 md which corresponds to the prior mean for $\ln(k_z)$. Note the corrections made to the k_z field in the CG case is more localized. The two methods give similar change to vertical permeability in the region around well 1, but Gauss-Newton also gives a significant correction to the region around well 2, whereas, for the conjugate gradient results, the perturbations in vertical permeability in the region around well 2 is small. This suggests that the CG method converged to a different local minimum and explains the higher root mean squared pressure mismatch obtained from the CG results. Fig. 4.14 shows the results for layer 3 and layer 4 of (k_z) field. Note that as compared to layer 1 and 2, the corrections are less in the other two layers. In the plots corrections seem to be zero because of the scale, though numerically they are not. But the magnitude is very small for both the optimization methods.

Fig. 4.15 shows the corrections made to the horizontal k field by the two

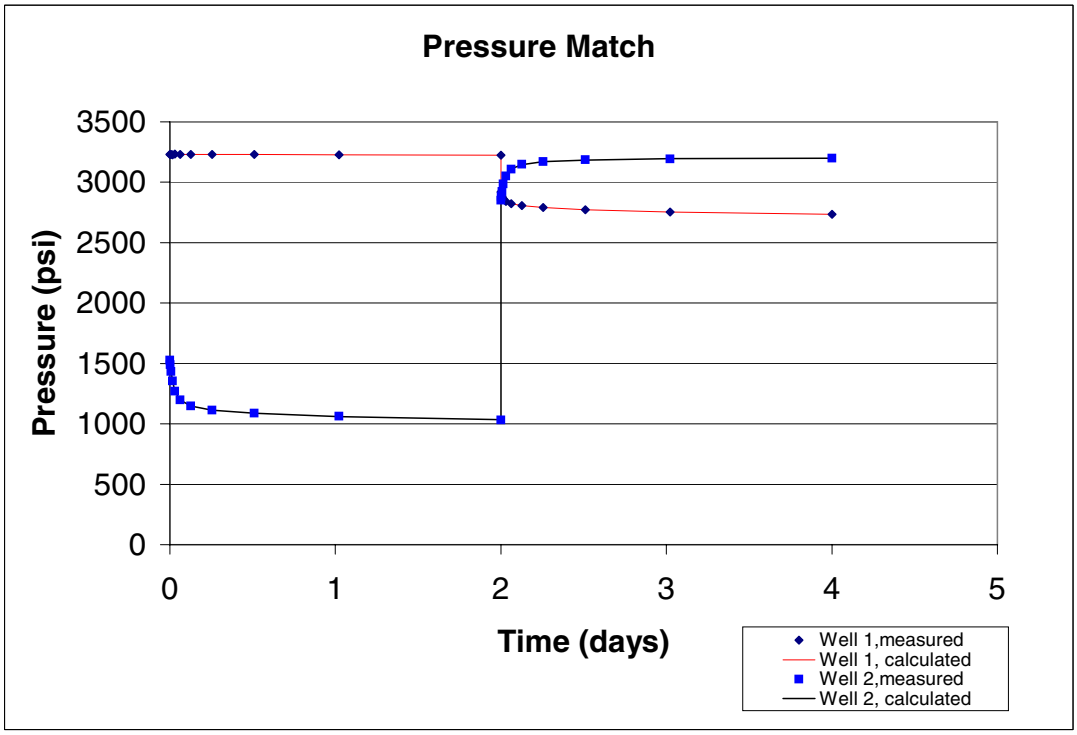
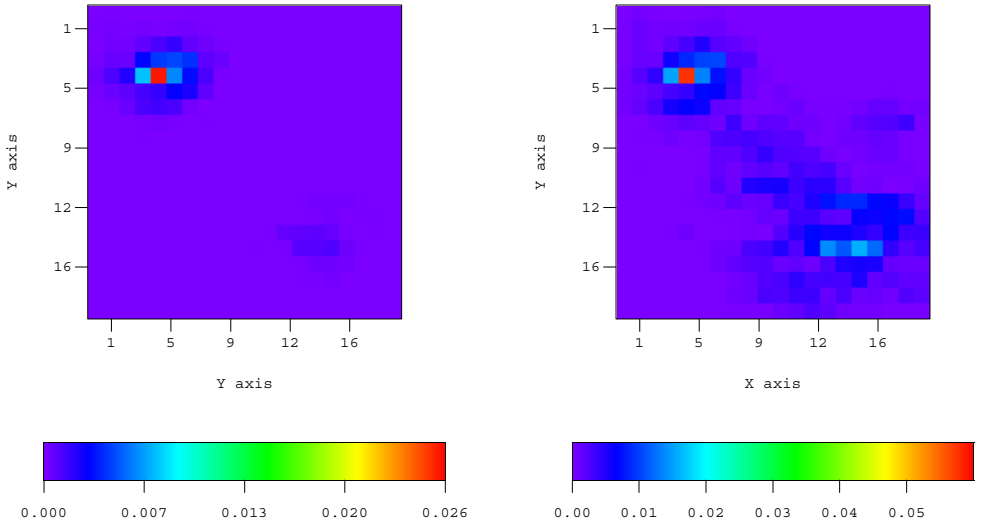
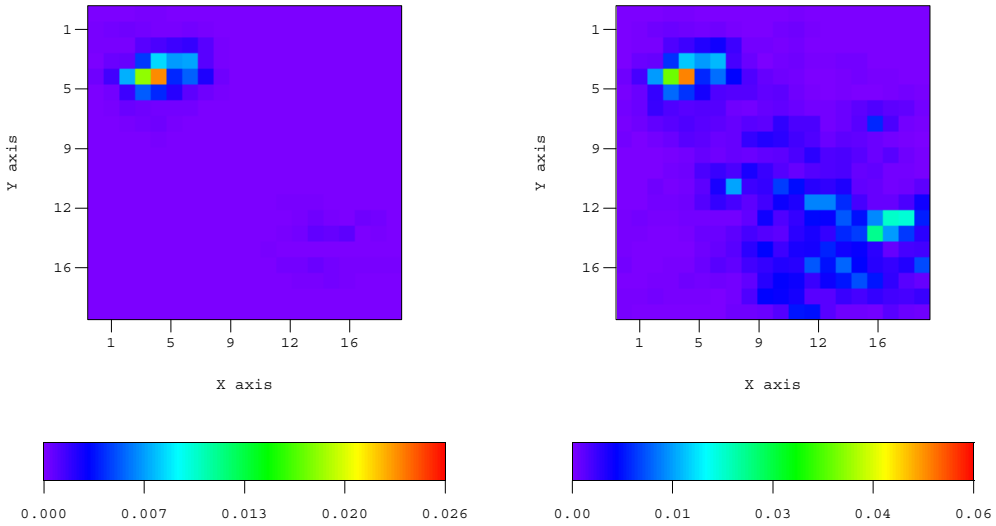


Figure 4.12: Pressure match obtained by conjugate gradient method.



(a) Correction by CG (layer 1) .

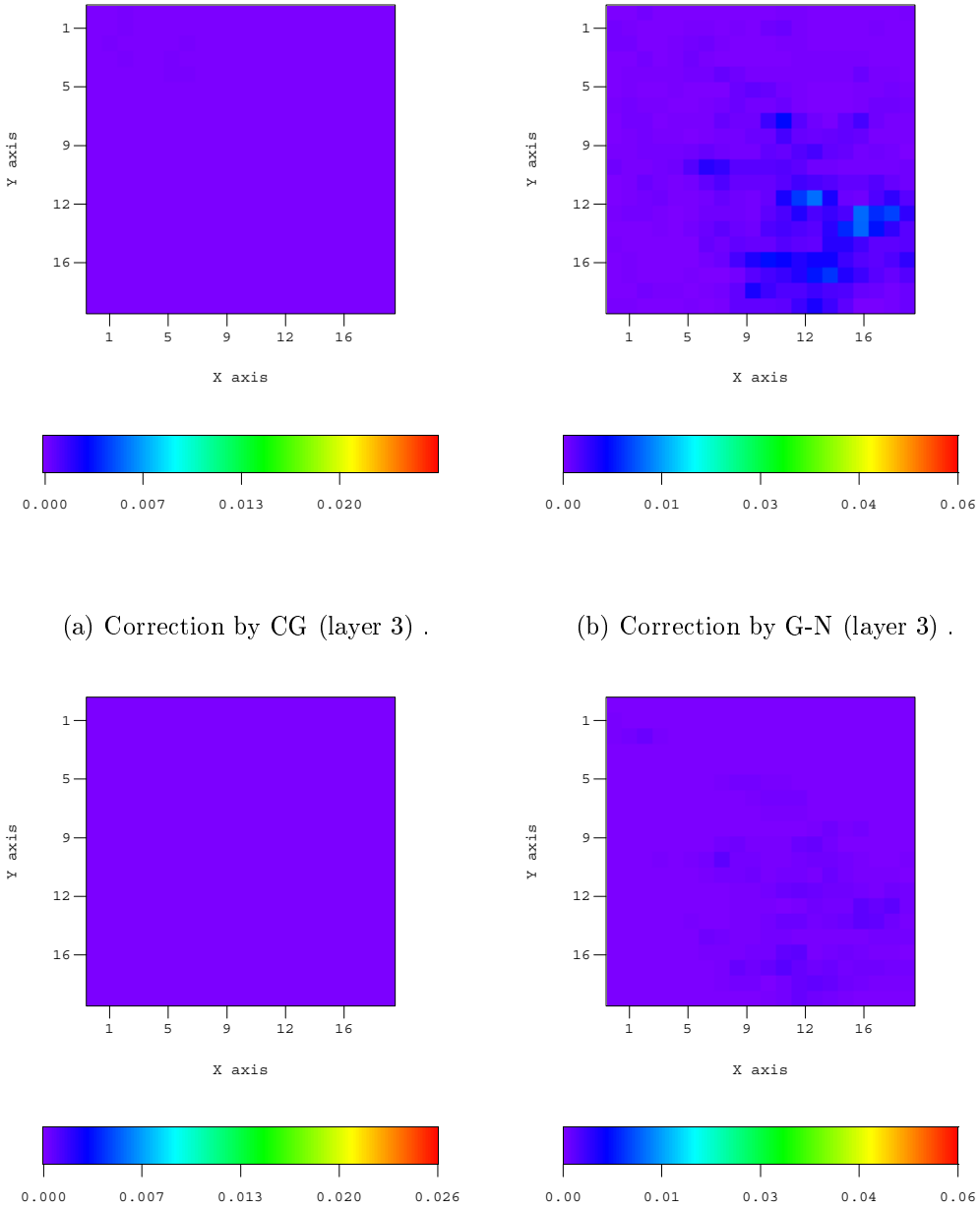
(b) Correction by G-N (layer 1) .



(c) Correction by CG (layer 2) .

(d) Correction by G-N (layer 2) .

Figure 4.13: Correction to vertical permeability field by the two methods (layer 1 and 2).



(a) Correction by CG (layer 3) .

(b) Correction by G-N (layer 3) .

(c) Correction by CG (layer 4) .

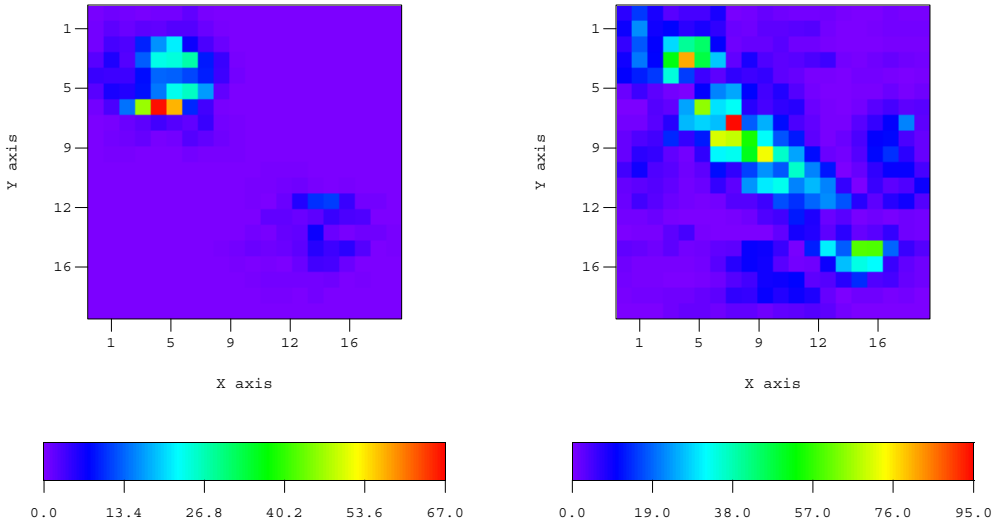
(d) Correction by G-N (layer 4) .

Figure 4.14: Correction to vertical permeability field by the two methods (layer 3 and 4).

algorithms. The Gauss-Newton shows higher correction in all the layers. The scales are different in the plots just to show the corrections more distinctly. Also corrections in layer 3 and 4 are almost negligible as compared to layer 1 and 2 and hence are not shown here. The corresponding figures for porosity fields are shown in Fig. 4.16.

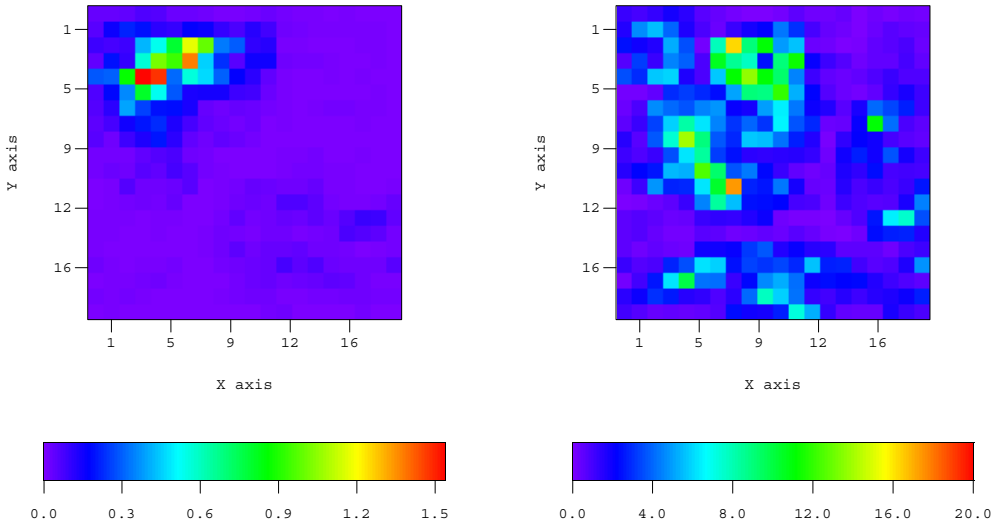
4.2.4 Computational Comparison Between Gauss-Newton and CG

In this example the gauss-Newton took 6 iterations with 10 step cut in the final iteration. So as discussed in Section 4.1.3, the equivalent number of simulation run is $(1 + 44 + 1) \times 6 + 10 = 286$. The conjugate gradient method took 97 iterations and 9 step cut. So the equivalent number of simulation run is $(1 + 1 + 1 + 1) \times 97 + 10 = 398$. This shows higher computational expense involved with the conjugate gradient method.



(a) Correction by CG (layer 1) .

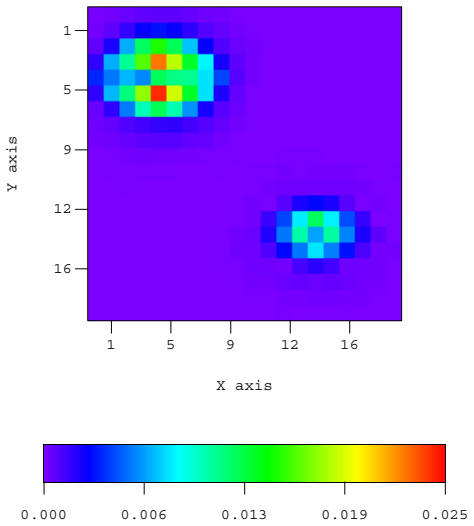
(b) Correction by G-N (layer 1) .



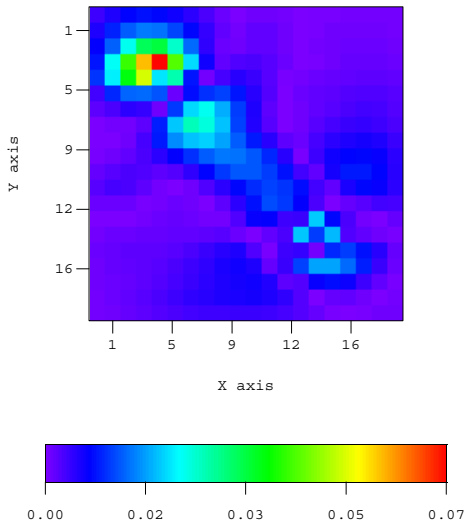
(c) Correction by CG (layer 2) .

(d) Correction by G-N (layer 2) .

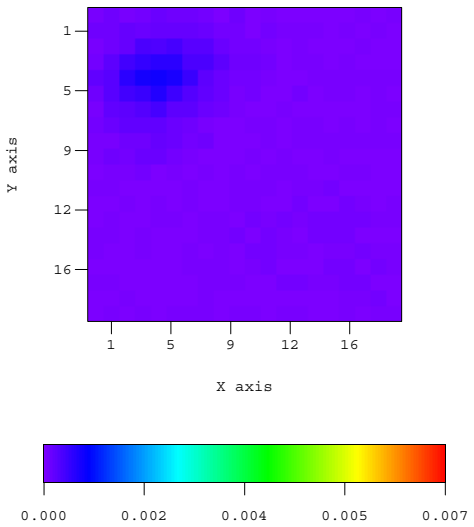
Figure 4.15: Correction to horizontal permeability field by the two methods (layer 1 and 2).



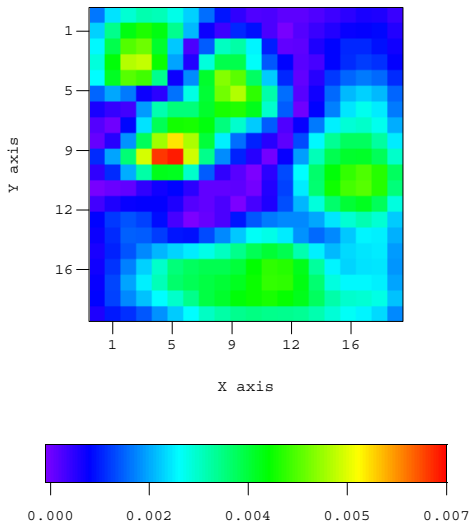
(a) Correction by CG (layer 1) .



(b) Correction by G-N (layer 1) .



(c) Correction by CG (layer 2) .



(d) Correction by G-N (layer 2) .

Figure 4.16: Correction to porosity field by the two methods (layer 1 and 2).

4.2.5 MAP Estimate for the Restricted-Entry Case:

Here we show the MAP estimate for the restricted entry case considered in the previous section. Recall the prior mean for $\ln(k)$ is 4.0, the prior mean for $\ln(k_z)$ is -2.9 and the prior mean for porosity is 0.25. Fig. 4.17 shows the MAP estimate of the k_z field obtained by the conjugate gradient method. As expected, the MAP estimate is a smooth one with modification to the prior model around the wells. Also the correction to the prior model is significant in the first and second layer as compared to the other two layers. The fourth layer shows the minimum correction and is almost negligible. Fig. 4.18 and Fig. 4.19 show the MAP estimate of the horizontal permeability (k) and porosity field respectively obtained by conjugate gradient method. The first layer results are plotted in a different scale to highlight the changes made to the prior model. Both the results show significant modification from the prior model in the first layer only. This is because the producing well is perforated only in the first layer and so the horizontal movement of fluid will be prominent only in the first layer. This makes the pressure data more sensitive to the top layer k and porosity fields.

Fig. 4.20, Fig. 4.21 and Fig. 4.22 show the results obtained by the Gauss-Newton method for the same problem. The results are very similar as compared to the conjugate gradient method.

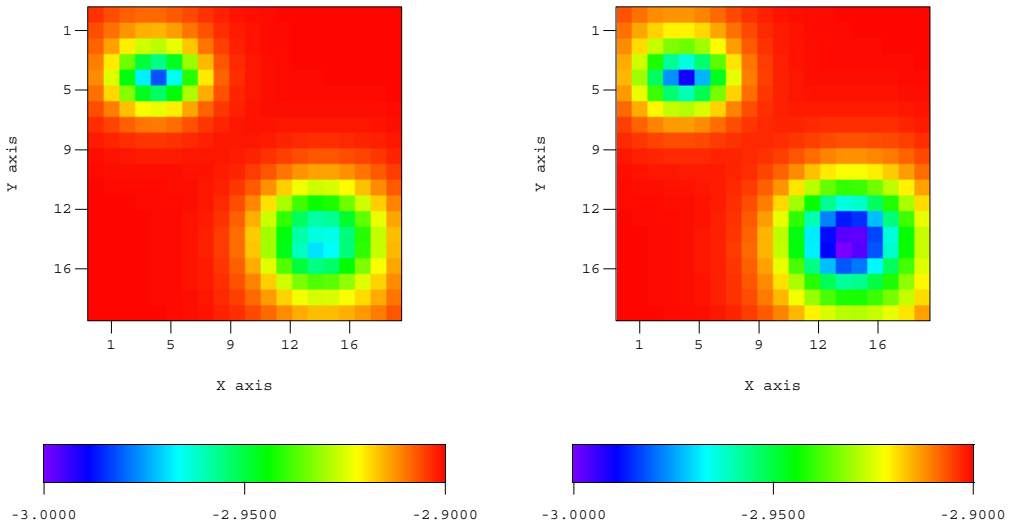
Fig. 4.23 shows the convergence rate for Gauss-Newton and conjugate gradient method. Gauss-Newton works remarkably well in this problem and converges in only seven iterations, whereas conjugate gradient takes 220 iterations to converge. The maximum number of iterations allowed before restart was set to 50 and the CG got restarted by the criteria ($\beta \leq 0$) before it ever reached 50 iterations. In total CG got restarted 20 times by this criteria.

Fig. 4.24 shows the pressure matched obtained for the MAP estimate. The root mean squared data mismatch for the actual observed data is 1.66 by the conjugate

gradient and 0.83 by the Gauss-Newton method.

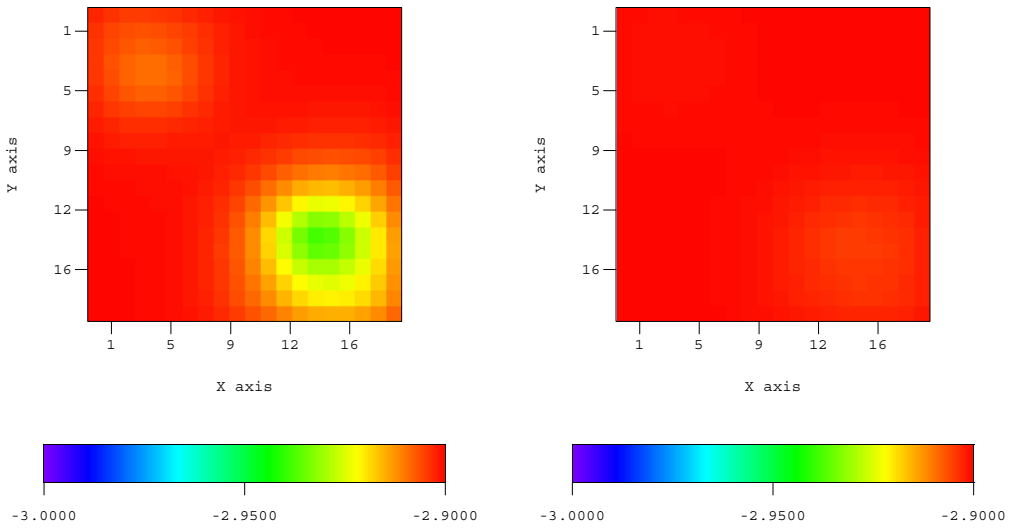
4.2.6 Computational Comparison Between Gauss-Newton and CG

In this example, the Gauss-Newton took 7 iterations without any step reductions. So the equivalent number of simulation run is $(1 + 44 + 1) \times 7 + 0 = 322$. The conjugate gradient method took 220 iterations without any step reductions. This gives equivalent number of simulation run as $(1 + 1 + 1 + 1) \times 220 + 0 = 880$. (See Section 4.1.3 for details of this calculation). This shows much higher computational effort involved in case of the conjugate gradient method.



(a) $\ln(k_z)$ field (layer 1) .

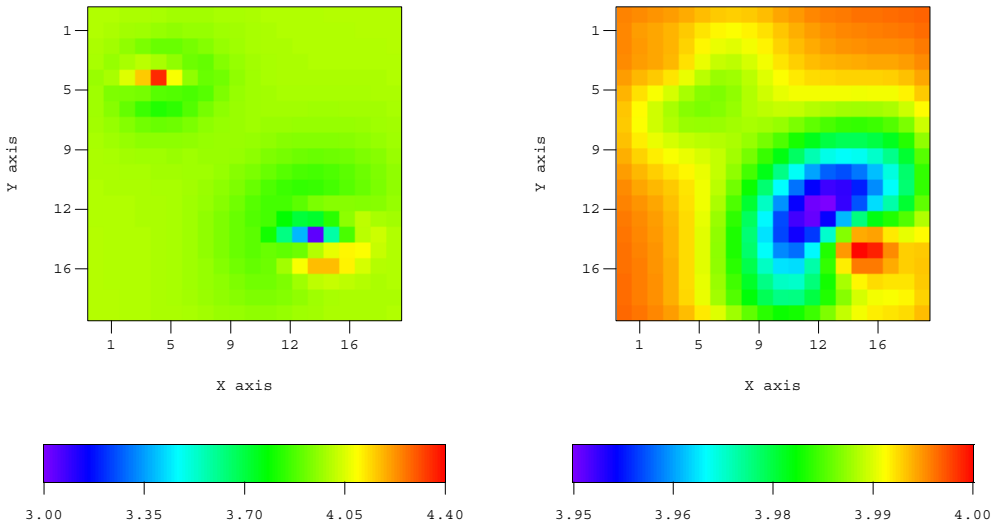
(b) $\ln(k_z)$ field (layer 2) .



(c) $\ln(k_z)$ field (layer 3) .

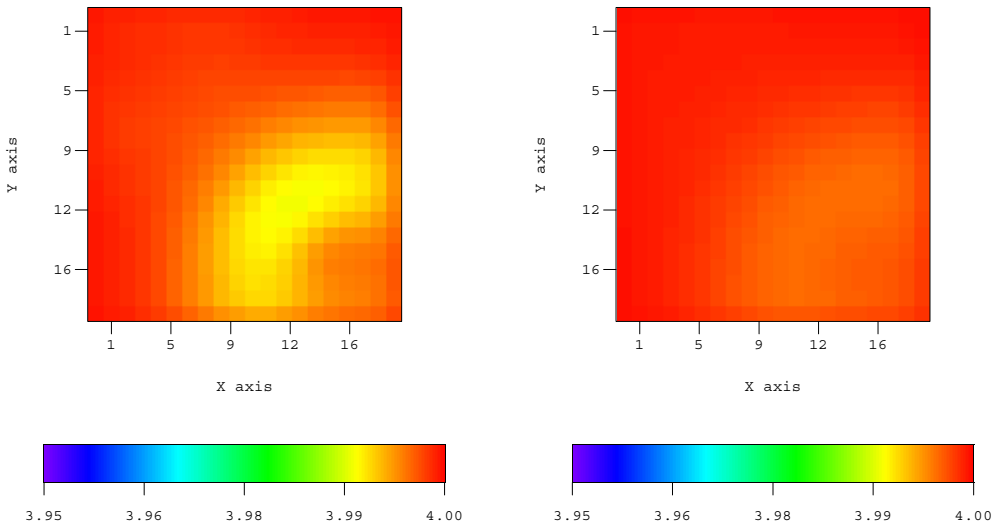
(d) $\ln(k_z)$ field (layer 4) .

Figure 4.17: MAP estimate of $\ln(k_z)$ field by conjugate gradient method.



(a) $\ln(k)$ field (layer 1) .

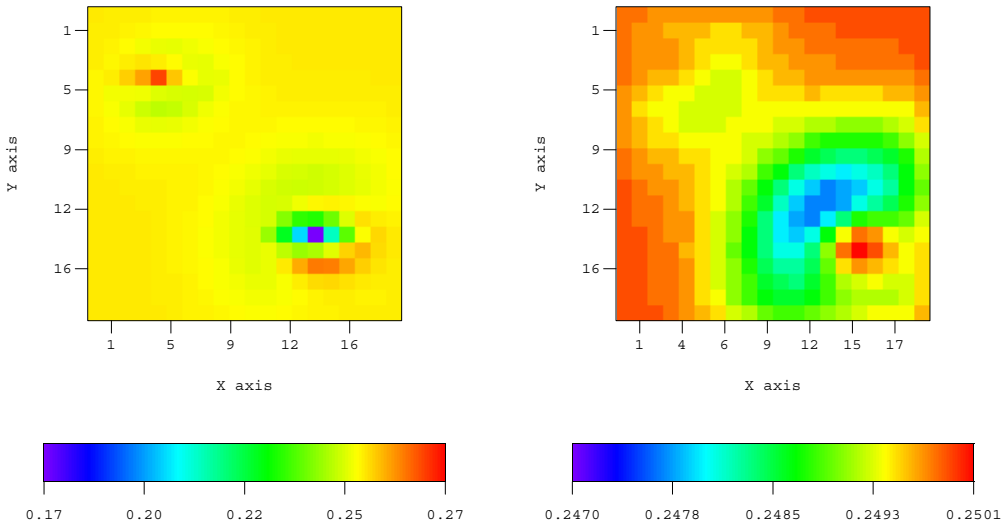
(b) $\ln(k)$ field (layer 2) .



(c) $\ln(k)$ field (layer 3) .

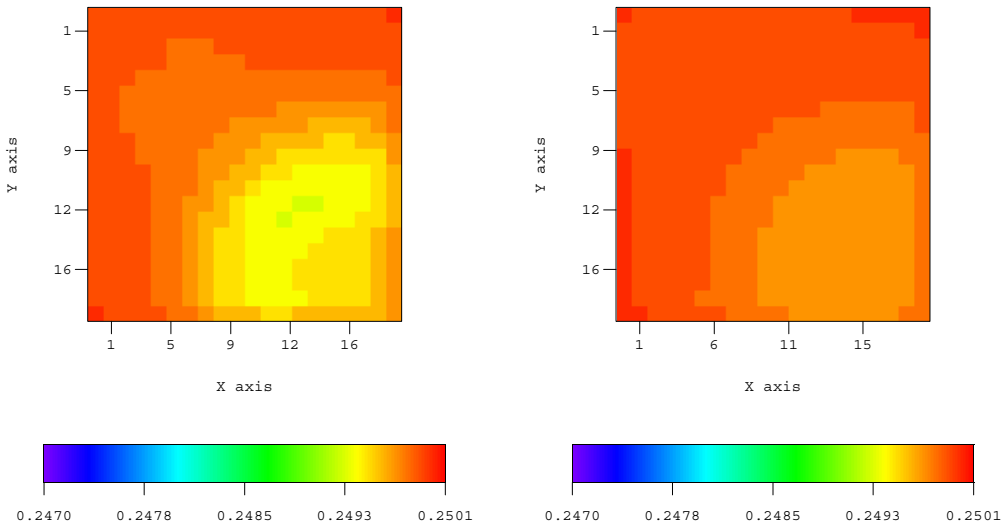
(d) $\ln(k)$ field (layer 4) .

Figure 4.18: MAP estimate of horizontal $\ln(k)$ field by conjugate gradient method.



(a) Porosity field (layer 1) .

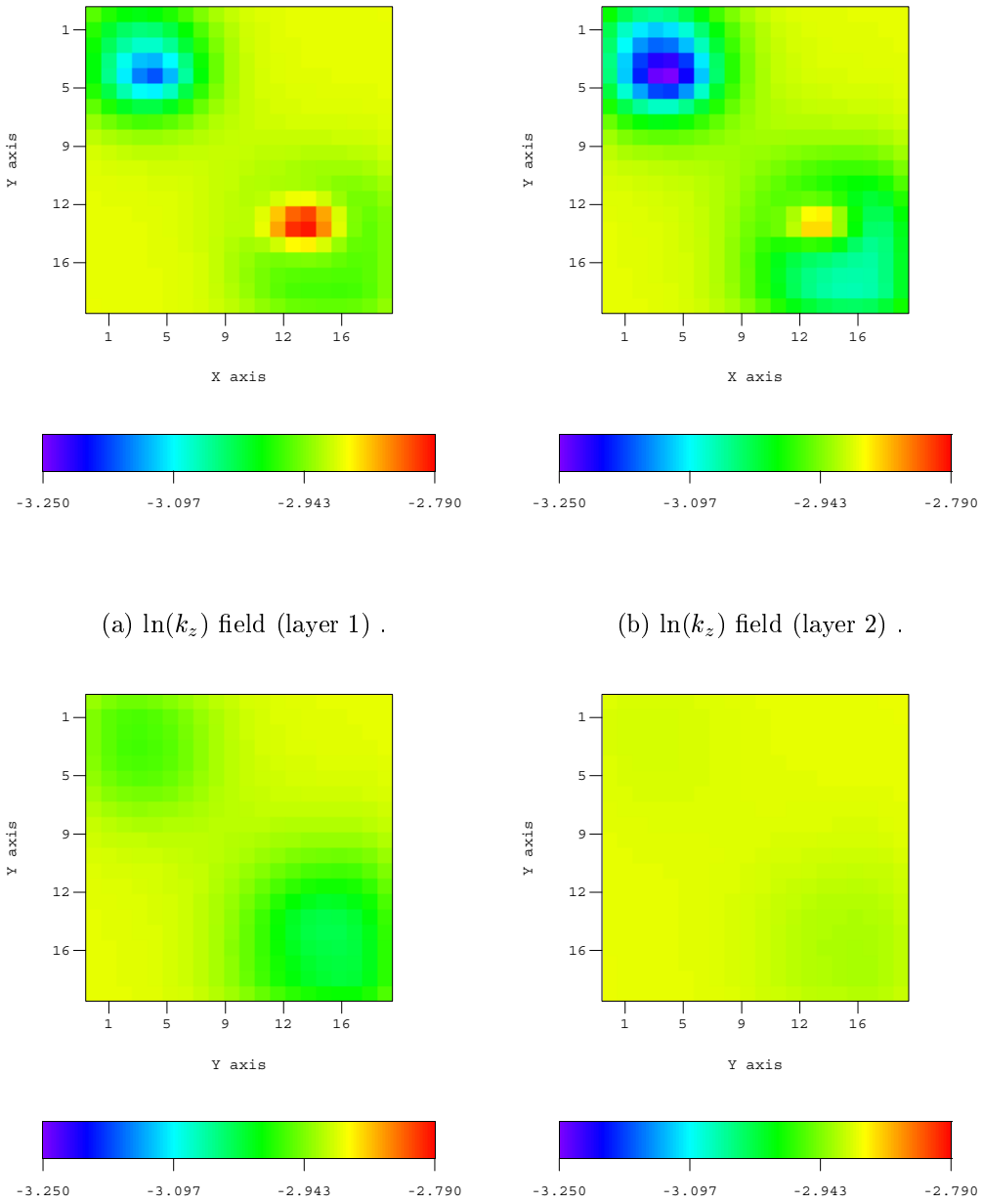
(b) Porosity field (layer 2) .



(c) Porosity field (layer 3) .

(d) Porosity field (layer 4) .

Figure 4.19: MAP estimate of porosity field by conjugate gradient method.



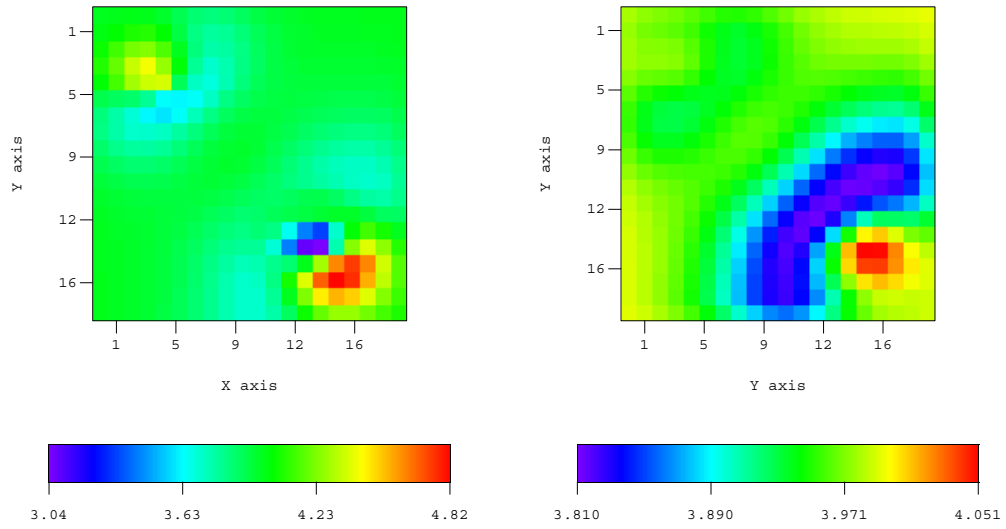
(a) $\ln(k_z)$ field (layer 1) .

(b) $\ln(k_z)$ field (layer 2) .

(c) $\ln(k_z)$ field (layer 3) .

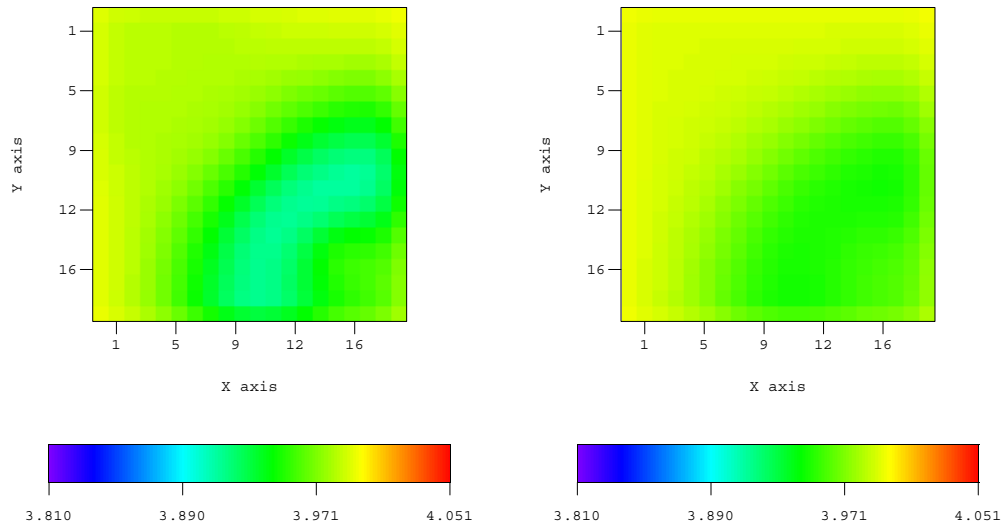
(d) $\ln(k_z)$ field (layer 4) .

Figure 4.20: MAP estimate of $\ln(k_z)$ field by Gauss-Newton method.



(a) $\ln(k)$ field (layer 1) .

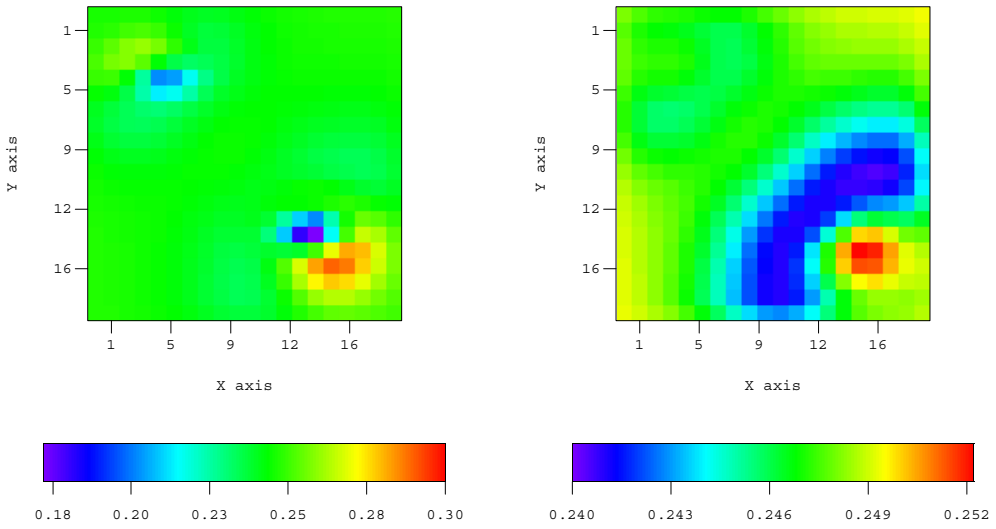
(b) $\ln(k)$ field (layer 2) .



(c) $\ln(k)$ field (layer 3) .

(d) $\ln(k)$ field (layer 4) .

Figure 4.21: MAP estimate of horizontal $\ln(k)$ field by Gauss-Newton method.



(a) Porosity field (layer 1) .

(b) Porosity field (layer 2) .

(c) Porosity field (layer 3) .

(d) Porosity field (layer 4) .

Figure 4.22: MAP estimate of porosity field by Gauss-Newton method.

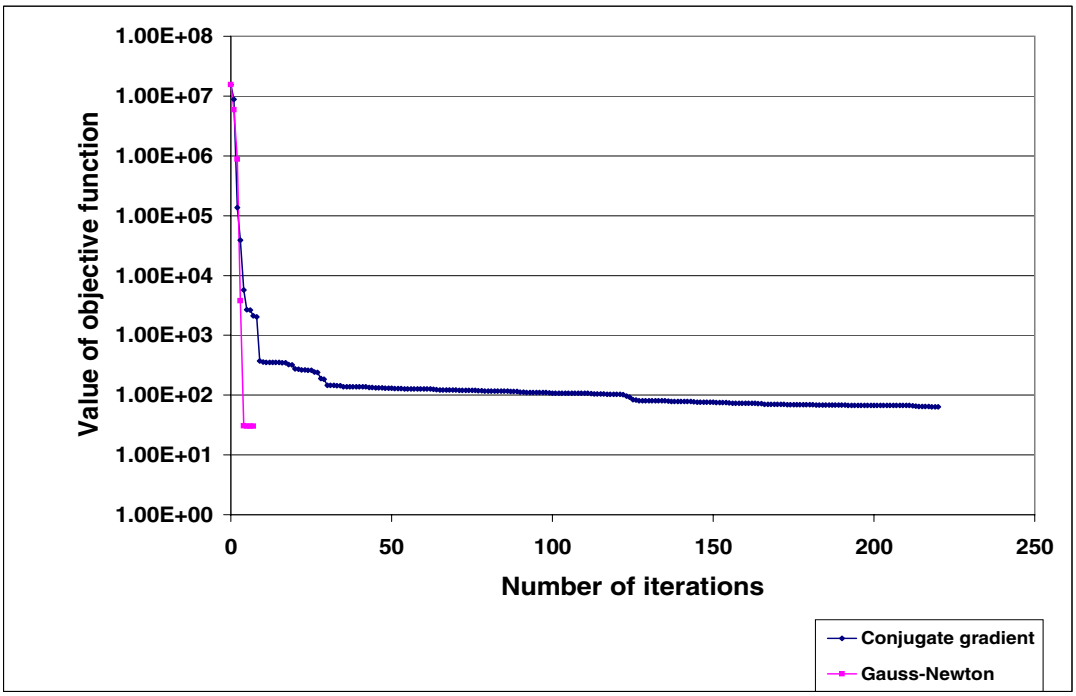


Figure 4.23: Objective function minimization by conjugate gradient and Gauss-Newton method for the MAP estimate.

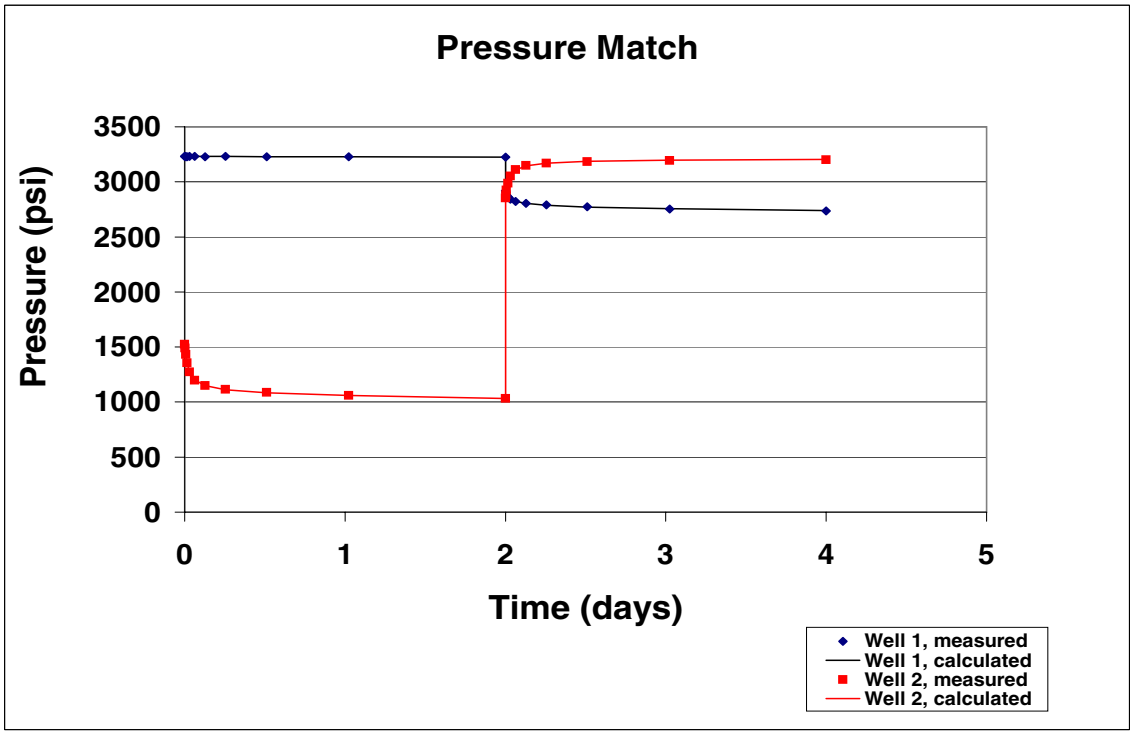


Figure 4.24: Pressure match obtained by conjugate gradient method for the MAP estimate.

Well	Location	Flowrate (MMscf/day)	Duration (days)
1	5,5	30.0	2.00
2	15,5	0.00	2.00
3	10,10	25.0	1.00
3	10,10	0.00	1.00
4	5,15	0.00	2.00
5	15,15	35.0	2.00

Table 4.6: Specified production rates, five well case.

4.2.7 3D Case with Five Fully Penetrating Wells:

In this section, we consider a more complicated production profile with more wells and different flow schemes. The reservoir data and the model parameters details remain the same as in the previous cases shown in Table 4.3. We consider five wells, three of them producing and two observation wells. Two of the wells are put on constant rate production, whereas the third one is produced first at constant production rate and then shut in for a build-up test. The production scheme is given in Table 4.6. We generate 20 data points for each well and so the total number of conditioning data used is 100. The true pressure data is perturbed with measurement errors with variance 1psi^2. The results obtained are similar as to the previous cases. Fig 4.25 shows the behavior of the Gauss-Newton and CG in minimizing the objective function. As compared to the Gauss-Newton method, the CG takes far more iterations to achieve convergence. Gauss Newton converges to a value of 106 in 15 iterations, whereas CG took 95 iterations to reach the minimum value of 155. Thus, as in the previous cases, CG converges to a somewhat higher value than Gauss-Newton. From experimentation of the convergence performance of CG, we found that CG is restarted within the optimization procedure 35 times by the criteria number two as

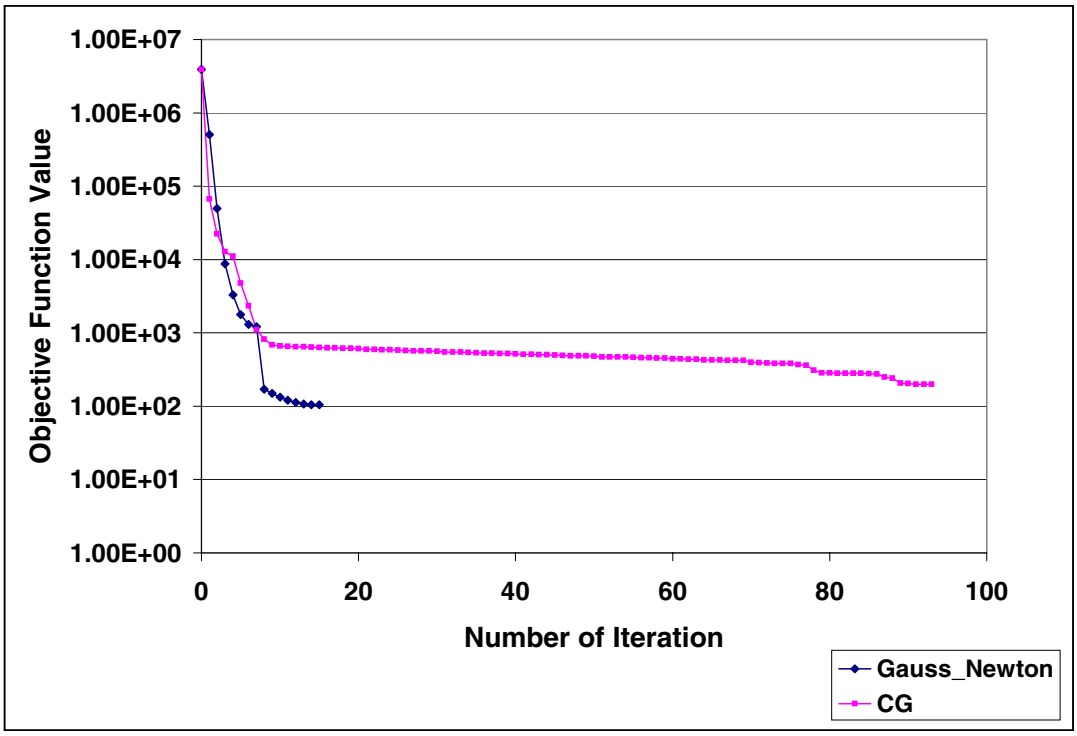


Figure 4.25: Objective function minimization by conjugate gradient and Gauss-Newton method.

discussed in important remarks in chapter I (i.e. restart CG whenever $r_k^T d_k \leq 0$). Also most of these restarting takes place within the "flat portion" of the convergence curve. This means that our inexact line search method is not accurate enough to give a descent direction in the next iteration and CG gets restarted to solve this problem. This frequent restarting reduces the convergence rate of CG considerably. Fig 4.26 shows the pressure match obtained by the conjugate gradient method. Based on Eq. 4.5, the root mean squared pressure mismatch upon convergence was 1.90 by CG and 1.10 by Gauss-Newton. The corresponding data mismatch based on Eq. 4.6 is 1.61 for CG and 0.64 for Gauss-Newton respectively.

Fig 4.27 shows the true porosity field used to generate the true pressure data and the conditional realization obtained by CG and Gauss-Newton method for

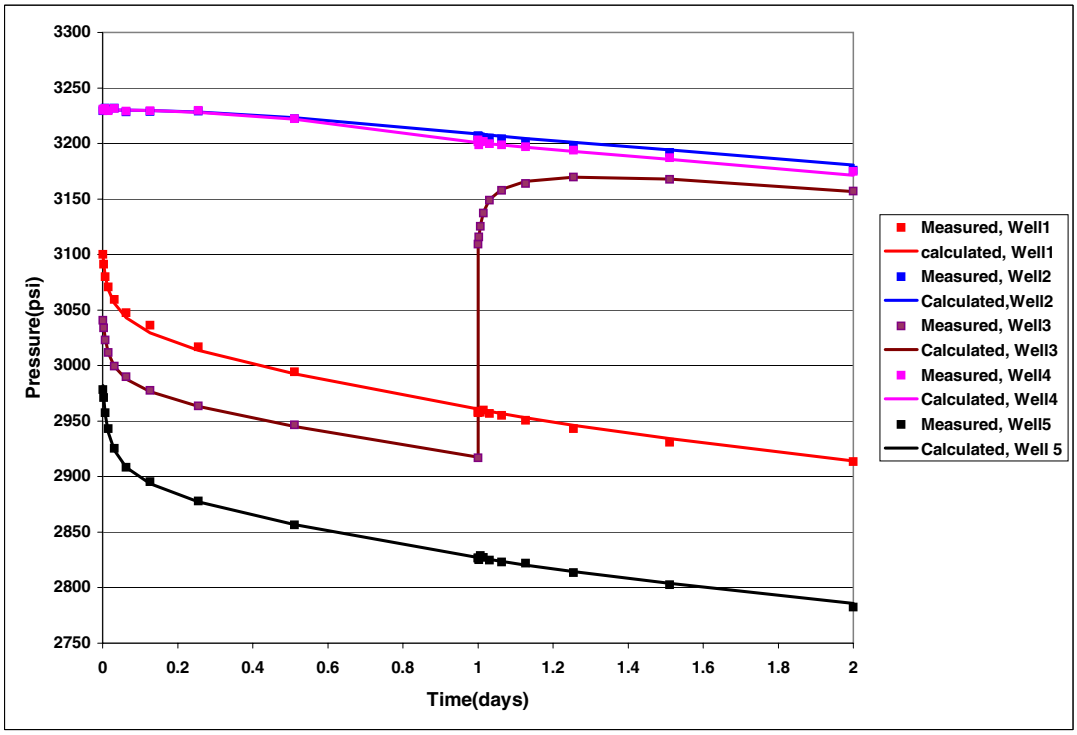
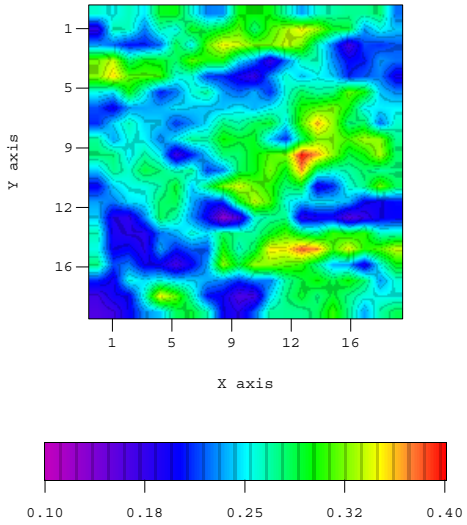
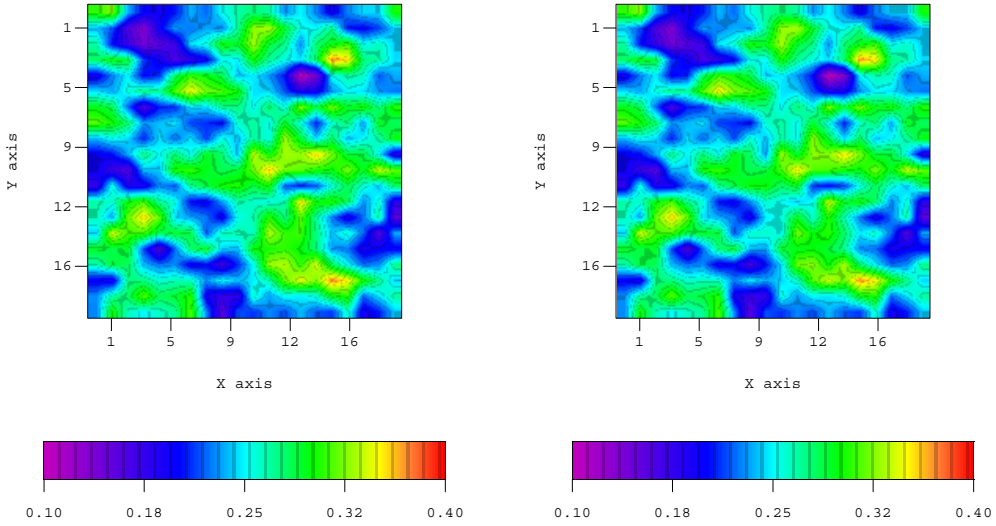


Figure 4.26: Pressure match obtained by conjugate gradient method.



(a) True porosity field (layer 1).



(b) Realization obtained by Gauss Newton method.

(c) Realization obtained by conjugate gradient method.

Figure 4.27: True porosity field and comparison of conditional realization obtained by Gauss-Newton and conjugate gradient method (for layer 1) .

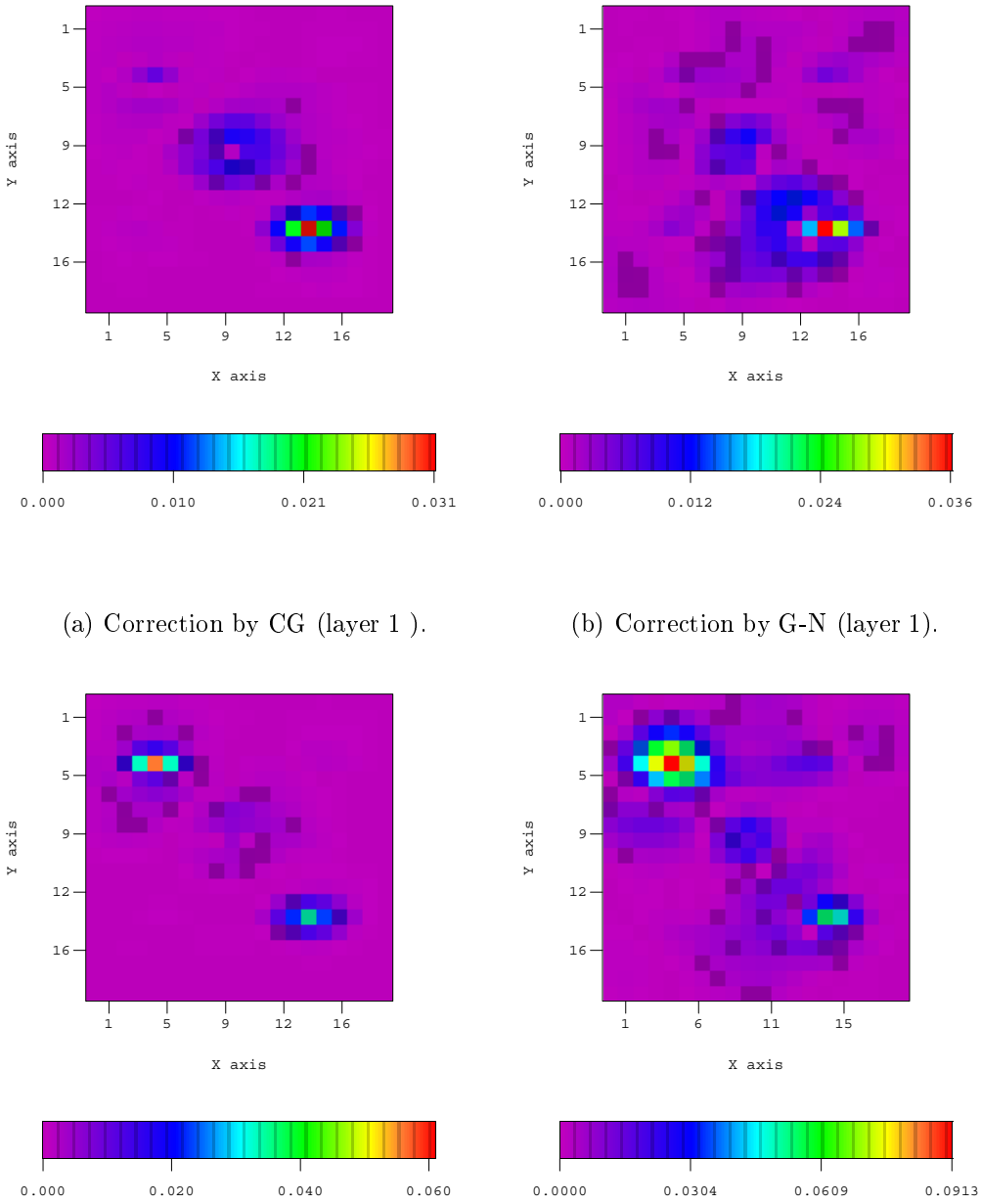
the layer 1. It can be seen that the conditional realizations are similar by both the methods. However, when we look at the correction made to the porosity fields during the conditioning process, we observe that the corrections obtained from the two processes are not the same, though they result in similar realizations. Fig 4.28 shows the correction done to layer 1 and layer 4 porosity by the two methods during the conditioning process. In the layer 1, maximum correction has been made at the location of well 5 by both the methods. CG makes a correction of about 0.03 and Gauss-Newton makes a correction of about 0.04. Also in both the pictures there is a region of significant porosity change around the location of well number 3. However Gauss-Newton seems to make porosity changes on a wider area than the CG (see the regions away from the well). Similarly for layer 4, in both the models, major correction has been made at the location of well number 1. The correction by CG is about 0.065 and by Gauss-Newton is 0.09. Also note that both methods result in changes around well number 5 and well number 3. Thus even though the magnitude of the changes differ from method to method, qualitatively the two methods yield similar corrections. The results for the other layers are not shown but yield results consistent with the preceding statement.

Similarly, Fig 4.29 shows the modification to the individual layer permeability fields from the two optimization algorithms. We compare results for the layer 1 and layer 2 where the changes are the largest. For layer 1, both the methods make the highest correction at location of well number 1. The correction is about 53.9 in CG and 66.0 in Gauss-Newton. Also note the regions of correction around the wells number 3 and 5. But in the Gauss-Newton results the region between well number 3 and well number 5 shows a more correction as compared to CG. Comparing the results for layer 2, we also observe similar results. Both the methods make highest correction at the location of well number 1. The change is 171.71 in CG and 102.33 in Gauss-Newton. We can also see correction around well number 3 and well number 5 in both the methods. Again the corrections in CG looks more localized as compared

to Gauss-Newton.

4.2.8 Computational Comparison Between Gauss-Newton and CG

In this example, the gauss-Newton took 15 iterations with 16 step reductions spread over all the iterations. This gives the roughly equivalent number of simulation run as $(1 + 100 + 1) \times 15 + 16 = 1546$. On the other hand, the conjugate gradient took 95 iterations with 31 step reductions spread over the iterations. This gives the number of simulation runs as $(1 + 1 + 1 + 1) \times 95 + 31 = 419$. (See Section 4.1.3 for details of these calculations).



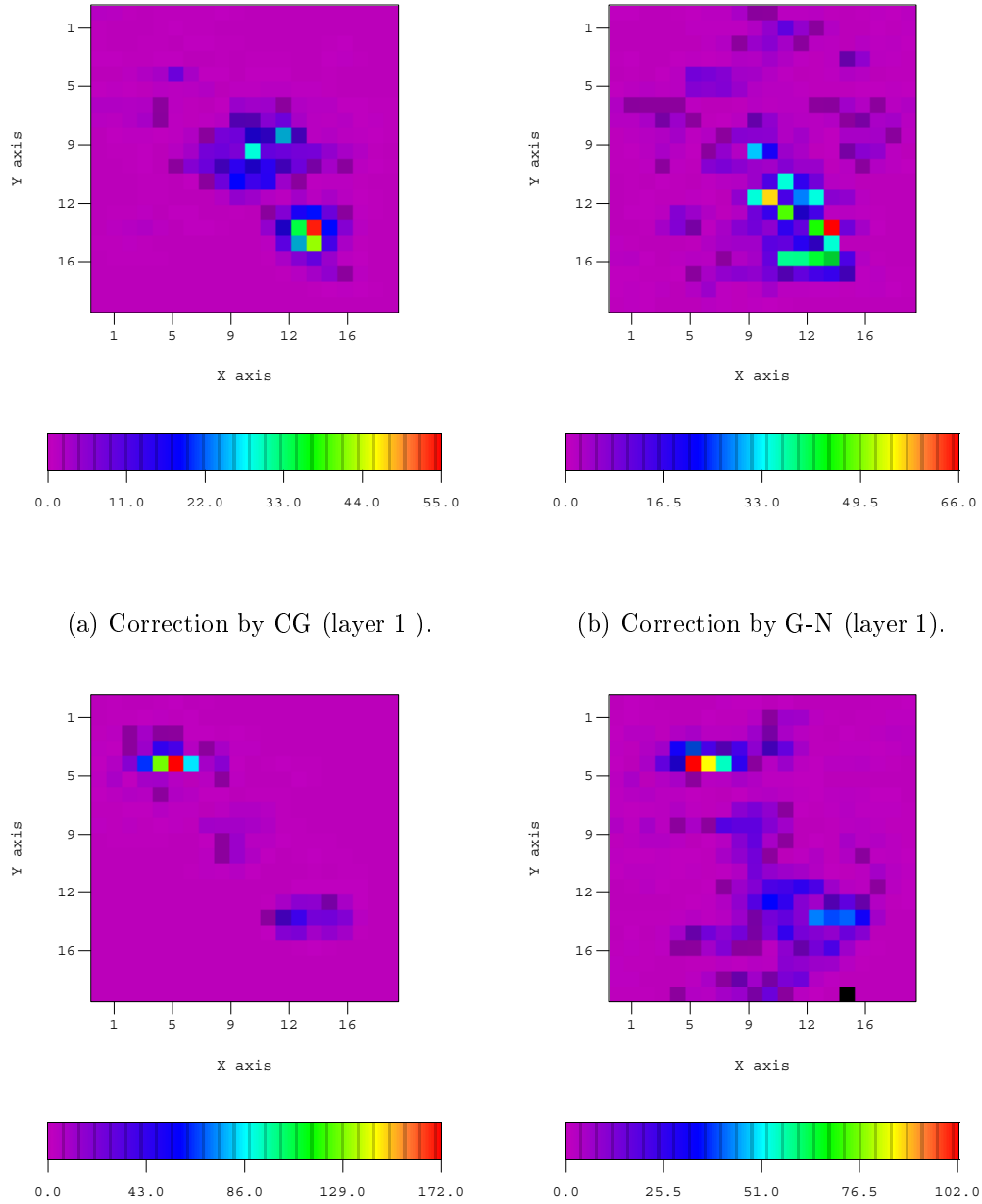
(a) Correction by CG (layer 1).

(b) Correction by G-N (layer 1).

(c) Correction by CG (layer 4).

(d) Correction by G-N (layer 4).

Figure 4.28: Comparison of correction to the porosity field by CG and Gauss-Newton (layer 1 and layer 4).



(a) Correction by CG (layer 1).

(b) Correction by G-N (layer 1).

(c) Correction by CG (layer 2).

(d) Correction by G-N (layer 2).

Figure 4.29: Comparison of correction to the permeability field (k) by CG and Gauss-Newton (layer1 and layer 2).

CHAPTER V

CONCLUSIONS

In this work, we have considered the problem of generating the maximum a posteriori estimate and realizations of log-permeability and porosity fields conditioned to pressure data for single-phase flow of gas for both two-dimensional and three-dimensional cases. This exercise also focuses on the issue of using conjugate gradient method as the optimization algorithm as an alternative to Gauss-Newton method with restricted step. We must calculate the sensitivity of each pressure data (in case of the Gauss-Newton) or gradient of the objective function (in case of the conjugate gradient) with respect to the model parameters while running the optimization algorithm and we have presented an efficient algorithm to calculate these sensitivities. The Gauss-Newton algorithm involves solving an adjoint system of equations N_d times (N_d is the number of conditioning data being used) to generate the whole sensitivity matrix, while the conjugate gradient requires solving the adjoint system of equation only once to calculate the gradient of the objective function. The sensitivity results obtained by the adjoint method have been checked with the finite-difference (direct) method for a variety of cases and the agreement between the two sets of results is excellent. Also note that in the examples we are conditioning the reservoir model only to pressure data, but the methodology to condition to rate data has been implemented.

The performance of Gauss-Newton and conjugate gradient algorithm has been compared and analyzed for a number of cases. The two methods result in reservoir models that satisfy the pressure data very well. On a larger scale, the final models obtained by the two methods are similar, but the magnitude of corrections incorpo-

rated in the final model on a gridblock to gridblock basis is not exactly same. Also the correction caused by Gauss-Newton is more spread out; whereas the correction by conjugate gradient is more localized around the wells. In most of the cases considered here, the conjugate gradient converges to a somewhat higher value of the objective function corresponding to some local minimum; whereas, Gauss-Newton converges to a lower value. This results in somewhat different models by the two methods. However the pressure match obtained by both the methods is quite satisfactory.

In most cases considered, the Gauss-Newton method required far fewer iterations than the conjugate gradient algorithm to obtain convergence. Although one iteration in Gauss-Newton is much more time consuming than one iteration in conjugate gradient, the high number of iterations required in conjugate gradient pose a threat to its implementation in the current form for a large scale problem. Nevertheless, we have found some interesting facts about the conjugate gradient itself while analyzing its convergence rate. The conjugate gradient method without preconditioning takes far more iterations to converge compared to the one with preconditioning. Also the results obtained without preconditioning is very localized around the wells. We are using the inverse of the prior covariance matrix as the preconditioning matrix and since, this is a poor approximation to the actual Hessian, we are of the belief that a preconditioner approximating the Hessian closely will improve the convergence rate of the conjugate gradient algorithm. How to generate such an approximation in a computationally efficient way is unclear at this point. We have also addressed the issue whether to use an exact line search method to calculate the step size in the conjugate gradient method. We found that during the initial stages when the gradient of the objective function is high, inexact line search can achieve a fast convergence rate. But when the gradient of the objective function is small, which suggests a flat region in the objective function, inexact line search may result in restarting of the conjugate gradient algorithm. But since each iteration within the inner Newton-Raphson method is almost computationally equivalent to one full iteration of the conjugate

gradient (with a single step in Newton-Raphson iteration), exact line search does not offer any significant benefits. Hence, we believe it is preferable to do just one Newton iteration for step size calculation.

Finally, we have applied the computational technique to several synthetic cases implementing both the Gauss-Newton and the conjugate gradient algorithm. The results give estimates of log-permeability (both horizontal and vertical) and porosity fields which honor the pressure data. The process can be applied to generate reservoir models following the principle of automatic history matching.

Bibliography

[1] Lee A.L., Gonzalez M.H., and Eakin B.E. The viscosity of natural gases. *Journal of Petroleum Technology*, 237, 1966.

[2] Zhuoxin Bi, Dean S. Oliver, and Albert C. Reynolds. Conditioning 3d stochastic channels to pressure data (SPE-56682). In *1999 SPE Annual Technical Conference and Exhibition*, 1999.

[3] R. D. Carter, L. F. Kemp, and A. C. Pierce. Discussion of comparison of sensitivity coefficient calculation methods in automatic history matching. *Soc. Petrol. Eng. J.*, pages 205–208, April 1982.

[4] 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.

[5] 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.

[6] W. H. Chen, G. R. Gavalas, John H. Seinfeld, and Mel L. Wasserman. A new algorithm for automatic history matching. *Soc. Petrol. Eng. J.*, pages 593–608, December 1974.

[7] L. Chu and A. C. Reynolds. A general efficient method for generating sensitivity coefficients. In *Well Testing, Reservoir Characterization and Reservoir Simu-*

lation, Petroleum Reservoir Exploitation Projects, pages 100–133. University of Tulsa, 1995.

[8] L. Chu, A. C. Reynolds, and D. S. Oliver. Computation of sensitivity coefficients for conditioning the permeability field to well-test pressure data. *In Situ*, 19(2):179–223, 1995.

[9] Roger Fletcher. *Practical Methods of Optimization.* John Wiley & Sons, New York, 1987.

[10] 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.

[11] N. He, A. C. Reynolds, and D. S. Oliver. Three-dimensional reservoir description from multiwell pressure data and prior information. *Soc. Pet. Eng. J.*, pages 312–327, Sept. 1997.

[12] Nanqun He. *Three-Dimensional Reservoir Description by Inverse Problem Theory Using Well-Test Pressure Data and Geostatistical Data.* Ph.D. thesis, University of Tulsa, Tulsa, Oklahoma, 1997.

[13] Nanqun He, Albert C. Reynolds, and Dean S. Oliver. Three-dimensional reservoir description from multiwell pressure data and prior information (SPE-36509). In *1996 SPE Annual Technical Conference and Exhibition*, 1996.

[14] J. E. Killough, Y. Sharma, A. Dupuy, and R. Bissell. 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.

[15] Peter K. Kitanidis. Quasi-linear geostatistical theory for inversing. *Water Resour. Res.*, 31(10):2411–2419, 1995.

[16] 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.

[17] He Nanqun. Three dimensional reservoir description by inverse theory using well-test pressure data and geostatistical data. *Phd. Thesis*, 1997.

[18] Dean S. Oliver. On conditional simulation to inaccurate data. *Mathematical Geology*, 28(6):811–817, 1996.

[19] Dean S. Oliver, Nanqun He, and Albert C. Reynolds. Conditioning permeability fields to pressure data. In *European Conference for the Mathematics of Oil Recovery*, pages 1–11, September 1996.

[20] 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.

[21] Albert C. Reynolds, Nanqun He, and Dean S. Oliver. Reducing uncertainty in geostatistical description with well testing pressure data. In *Fourth International Reservoir Characterization Technical Conference*, 2–4 March 1997.

[22] J. R. Schewchuk. An introduction to the conjugate gradient method without agonizing pain. Technical report, Lecture Notes, Carnegie Mellon University, 1994.

[23] Albert Tarantola. *Inverse Problem Theory: Methods for Data Fitting and Model Parameter Estimation.* Elsevier, Amsterdam, The Netherlands, 1987.

[24] Z. Wu, A. C. Reynolds, and D. S. Oliver. Conditioning geostatistical models to two-phase production data, (SPE-49003). In *1998 SPE Annual Technical Conference and Exhibition*, 1998.

[25] W. Xu, T. T. Tran, R. M. Srivastava, and A. G. Journel. Integrating seismic data in reservoir modeling: the collocated cokriging approach, (SPE-24742). In *1992 SPE Annual Technical Conference and Exhibition*, 1992.

APPENDIX A

CALCULATION OF GAS PROPERTY

A.1 Calculation of Gas Viscosity (μ_g)

Viscosity of gas is calculated based on the Lee et al. [1] correlation. This equation can be presented as,

$$\mu_g = 10^{-4} K \exp\left[X\left(\frac{\rho_g}{62.4}\right)^Y\right], \tag{A.1}$$

where,

$$K = \frac{(9.4 + 0.02M_g)T^{1.5}}{(209 + 19M_g + T)}, \tag{A.2}$$

and

$$X = 3.5 + (986/T) + 0.01M_g, \tag{A.3}$$

and

$$Y = 2.4 - 0.2X, \tag{A.4}$$

The term ρ_g is the gas density at reservoir pressure and temperature in unit of $\mathrm{lbm/ft^3}$; T is the reservoir temperature in ^0R; and M_g is the apparent molecular weight of the gas.

A.2 Gas Density Calculation (ρ_g)

Gas density can be calculated from

$$\rho_g = 2.7\gamma_g \frac{p}{zT}, \tag{A.5}$$

where ρ_g is the gas density in lbm/ft^3; γ_g is the gas specific gravity; p is pressure of gas in psia; T is absolute temperature of gas in ^0R; and z is the real-gas deviation factor.

A.3 Calculation of Gas Deviation Factor (z)

Calculate the pseudo-critical pressure (p_{pc}) in psia and temperature (T_{pc}) in ^0R using the following formula.

$$p_{pc} = 756.8 - 131\gamma_g - 3.6\gamma_g^2, \tag{A.6}$$

and

$$T_{pc} = 169.2 + 349.5\gamma_g - 74\gamma_g^2. \tag{A.7}$$

Calculate reduced pressure (p_{pr}) and temperature (T_{pr}) as

$$p_{pr} = \frac{p}{p_{pc}} \text{ and } T_{pr} = \frac{T}{T_{pc}} \tag{A.8}$$

Finally the z-factor is obtained from

$$z = \frac{0.06125 p_{pr} t \exp\left(-1.2(1-t)^2\right)}{y} \tag{A.9}$$

where $t = 1/T_{pr}$ and y is the real solution of the equation

$$\begin{aligned} F(y) \quad = \quad & 0 = -0.06125 p_{pr} t \exp\left(-1.2(1-t)^2\right) + \\ & \frac{y + y^2 + y^3 - y^4}{(1-y)^3} - (14.76 - 9.76t^2 + 4.58t^3)y^2 \\ & + (90.7t - 242.2t^2 + 42.2t^3)y^{(2.18+2.82t)}. \end{aligned} \tag{A.10}$$

The above equation is solved by Newton-Raphson method.

A.4 Calculation of Gas Formation Volume Factor (B_g)

B_g can be calculated as

$$B_g = 0.00504\frac{zT}{p}, \tag{A.11}$$

where B_g is in bbl/scf; p is in psia; T is in ${}^0R$ and z is the real gas deviation factor.