E48 OPTIMIZATION ALGORITHMS FOR AUTOMATIC HISTORY MATCHING OF PRODUCTION DATA

FENGJUN ZHANG and ALBERT C. REYNOLD
Department of Petroleum Engineering, University of Tulsa, 600 South College Avenue, Tulsa, O

Abstract

Within the framework of Bayesian statistics, realizations or estimates of rock property fields can be generated b automatic history matching of production data using a prior model to provide regularization. In this contex automatic history matching requires the minimization of an objective function which includes both model an data mismatch terms. For large scale problems, the computational efficiency and robustness of the optimizatio algorithms used for minimization are of paramount importance. From a comparison of algorithms for a variety c history matching problems, a scaled limited memory Broyden-Fletcher-Goldfarb-Shanno algorithm was identifie as the most promising for large scale optimization problems.

Introduction

We consider the application of automatic history matching to estimate or simulate reservoir model variables tha honor production data and are consistent with a prior geostatistical model. Reservoir variables may represen gridblock permeabilities and porosities, local or zonal transmissibility or pore volume multipliers, or geometri variables that describe the shape, size and location of objects. The Bayesian formulation of the problems considere here is identical to the one considered by Li et al. (2001), except that work considered only the Gauss-Newton (GN method and a modified Levenberg-Marquardt (MLM) algorithm for minimization. However, unless the number c model parameters or the number of data is relatively small, implementation of these methods in a way that require calculation of the sensitivity of each predicted data to each model parameter is impractical. Here, we compare variety of gradient-based algorithms on the basis of computational efficiency and memory requirements. Our goa is to identify optimization algorithms that can be applied for automatic history matching when the number c conditioning production data ranges from a few hundred to several thousand and the number of reservoir variable ranges from several hundred to tens of thousands. Although reparameterization methods, such as zonation, gra zones, pilot points, spectral decomposition, and subspace methods, are sometimes used to reduce the number c sensitivities calculated and the number of variables estimated directly, no model reparameterization is applied fo the problems considered in this paper.

Quasi-Newton (variable metric) methods, which are based on generating an approximation to the inverse of th Hessian matrix, require only the gradient of the objective function and thus avoid the computation of individua sensitivity coefficients needed to directly form the Hessian matrix. Here, only the Broyden-Flecher-Goldfarb-Shann (BFGS) quasi-Newton method is considered since it has proved to be more robust in practice than other algorithms see Kolda et al. (1998). It is well known that scaling can improve the convergence attributes of quasi-Newton (QN methods, and numerous suggestions have been made for calculating scaling factors; see, in particular, Oren an Luenberger (1974) and Shanno and Phua (1978). Here, we identify scaling procedures that have worked well for th limited number of history matching problems that we have tried. For large scale optimization problems, memor requirements may be reduced by replacing the BFGS algorithm with the limited memory BFGS (LBFGS) algorithn in the form developed and implemented by Nocedal (1980).

Another minimization algorithm which uses only the gradient of the objective function is the nonlinear conjugat gradient algorithm. Here, we use the Polak-Ribière form of the nonlinear conjugate gradient algorithm; see Fletche (1987). Efficiency of the conjugate gradient (CG) method depends primarily on the preconditioner used. Intuitively one expects that BFGS will converge faster than CG unless one can find preconditioners for CG that contain simila curvature information to that which is contained in the inverse Hessian approximation used in BFGS. Here, w suggest two preconditioners. Although both often improve the performance of the the CG method, the resultin algorithms are still significantly less robust than appropriately scaled BFGS and LBFGS algorithms.

Quasi-Newton methods have been employed previously in automatic history matching-problems by Yang an Watson (1988), Masumoto (2000) and Savioli and Grattoni (1992). While these studies generally found that self-scaling BFGS method is more robust and computationally efficient than CG, steepest descent, and standar unscaled BFGS algorithms, the number of reservoir variables estimated was less than 25 in all examples considered

Makhlouf et al. (1993) applied the conjugate gradient method to estimate 450 gridblock permeabilities by history matching production data obtained under multiphase flow conditions. In one example, 110 iterations were required to obtain convergence. In the second example considered, 222 iterations were required to obtain convergence. However, the authors apparently did not use preconditioning.

Deschamps et al. (1998) presented an interesting comparison of the relative efficiency of several optimization methods for automatic history matching of production data. They suggest that the most efficient optimization method will be a hybrid scheme and specifically advocate schemes that combine a Gauss-Newton method with another procedure. They reject the Levenberg-Marquardt scheme as relying too heavily on the steepest descent method and do not present any comparisons based on this method. On the other hand, some results (see, for example, Li et al. (2001)) indicate that a modified Levenberg-Marquardt (MLM) algorithm can be superior to the Gauss-Newton method when initial data mismatches are very large. The Levenberg-Marquardt algorithm used by Li et al. (2001), however, is a nonstandard one and is based on a regularized objective function.

Deschamps et al. also compared the hybrid methods with a pure quasi-Newton method. For the two history matching problems they considered, the quasi-Newton method required on the order of three times as many "equivalent simulation runs" to obtain convergence as most of the other methods. For the first example presented, a synthetic case, the quasi-Newton method converged to a much higher value of the objective function than all the other methods. However, it is not clear which quasi-Newton method they used, how they initialized the inverse Hessian approximation, or whether they used scaling. The scaled quasi-Newton methods considered in this paper perform much better than would be suggested by the results of Deschamps et al. (1998).

The History Matching Problem

Throughout, m denotes the vector of reservoir variables; m_{prior} denotes the vector of prior means; d_{obs} denotes the set of conditioning production data; $d_p(m)$ denotes the corresponding predicted data for a given model m; C_M denotes the prior covariance matrix; C_D denotes the covariance matrix for data measurement errors; N_m denotes the number of reservoir variables and N_d denotes the number of conditioning production data. The prior probability density function (pdf) for m is assumed to be multivariate Gaussian so by Bayes theorem, the posterior pdf for m conditional to the data is given by

$$f(m|d_{\text{obs}}) = a \exp\{-O(m)\}, \tag{1}$$

where a is the normalizing constant and

$$O(m) = \frac{1}{2}(m - m_{\text{prior}})^T C_M^{-1}(m - m_{\text{prior}}) + \frac{1}{2}(d_p(m) - d_{\text{obs}})^T C_D^{-1}(d_p(m) - d_{\text{obs}}). \tag{2}$$

Minimization of $O(m)$ yields the maximum a posteriori (MAP) estimate. A conditional realization of m can be generated by the randomized maximum likelihood method, (see, e.g., Oliver et al. (1996)). In this method, a realization is generated by minimizing

$$O_r(m) = \frac{1}{2}(m - m_{uc})^T C_M^{-1}(m - m_{uc}) + \frac{1}{2}(d_p(m) - d_{uc})^T C_D^{-1}(d_p(m) - d_{uc}), \tag{3}$$

where m_{uc} is an unconditional realization generated from the prior pdf, and d_{uc} is obtained by adding noise to d_{obs}. Throughout, we simply let $\hat{O}$ denote the objective function that we wish to minimize.

Optimization Algorithms

Gauss-Newton and Levenberg-Marquardt

In generating the MAP estimate, the search direction in the modified Levenberg-Marquardt (MLM) algorithm can be calculated from either of the following formulas:

$$\delta m_{k+1} = -\left[(1+\lambda_k)C_M^{-1} + G_k^T C_D^{-1} G_k\right]^{-1}\left[C_M^{-1}(m_k - m_{\text{prior}}) + G_k^T C_D^{-1}(d_p(m_k) - d_{\text{obs}})\right]; \tag{4}$$

$$\delta m_{k+1} = \frac{m_k - m_{\text{prior}}}{1+\lambda_k} + C_M G_k^T\left[(1+\lambda_k)C_D + G_k C_M G_k^T\right]^{-1}\left[\frac{G_k(m_k - m_{\text{prior}})}{1+\lambda_k} - (d_p(m_k) - d_{\text{obs}})\right]. \tag{5}$$

Choosing $\lambda_k = 0$ in Eqs. 4 and 5 gives the two corresponding formulas for the Gauss-Newton (GN) method. If the GN or MLM algorithm is applied to construct a realization by minimizing O_r given in Eq. 3, then d_{obs} and m_{prior}, should be replaced by d_{uc} and m_{uc}, respectively, in Eqs. 4 and 5. Once δm_{k+1}, has been computed, we can compute $m_{k+1} = m_k + \mu_k \delta m_{k+1}$ where μ_k is the step size, which can be calculated by the restricted step method;

see Fletcher (1987). Although a restricted step procedure can also be applied when MLM is used, we use a simpl procedure where we decrease λ_k by a factor of 10 if $\hat{O}(m_{k+1}) < \hat{O}(m_k)$, and if not, we increase λ_k by a factor c 10 and redo the step. An initial value of $\lambda_0 = 1000$ works satisfactorily for most problems.

Applying Eq. 5 requires solving an $N_d \times N_d$ matrix problem. If this matrix problem is solved iteratively b the conjugate gradient method, then one does not need to explicitly compute G; one only needs to be able t calculate Gu and $G^T v$ for vectors u and v at each iteration of the conjugate gradient method. Mackie and Madde (1993) presented an implementation of this procedure in the geophysics literature; Chu et al. (2000) introduced similar procedure into the petroleum engineering literature for single-phase flow problems. A somewhat differen and clearer presentation of how one can compute Gu and $G^T v$ is given in Abacioglu (2001). Computation of G requires a forward run of the simulation. Computation of $G^T v$ requires one solution of the adjoint system. As w do not believe this method is viable when the number of data is large, we have not considered it here.

BFGS and LBFGS

The search direction in the Gauss-Newton method is obtained by solving

$$H_k \delta m_{k+1} = -g_k, \tag{6}$$

where k is the iteration index, g_k denotes the gradient of the objective function evaluated at m_k, and H_k denote the Hessian matrix given by

$$H_k = C_M^{-1} + G_k^T C_D^{-1} G_k. \tag{7}$$

The $N_d \times N_m$ matrix G_k represents the coefficient matrix evaluated at m_k. If N_d and N_m are both large, th evaluation of all entries of G_k by either the adjoint or gradient simulator method is not feasible.

In quasi-Newton methods, H_k^{-1} is approximated by a symmetric positive definite matrix $\tilde{H}_k^{-1}$, which is correcte or updated from iteration to iteration. All updating formulas involve the difference in gradients, $y_k = g_{k+1} - g_k$ and the difference in iterates, $s_k = m_{k+1} - m_k$. The BFGS updating formula is given by

$$\tilde{H}_{k+1}^{-1} = \gamma_k \Big(\tilde{H}_k^{-1} - \frac{\tilde{H}_k^{-1} y_k y_k^T \tilde{H}_k^{-1}}{y_k^T \tilde{H}_k^{-1} y_k} + v_k v_k^T \Big) + \frac{s_k s_k^T}{s_k^T y_k}, \tag{8}$$

where γ_k is the scaling factor and

$$v_k = (y_k^T \tilde{H}_k^{-1} y_k)^{1/2} \Big(\frac{s_k}{s_k^T y_k} - \frac{\tilde{H}_k^{-1} y_k}{y_k^T \tilde{H}_k^{-1} y_k} \Big). \tag{9}$$

If no scaling is done, $\gamma_k = 1$ for all k. If scaling is done only at the first update, then $\gamma_k = 1$ for $k > 0$. In ou applications, $\tilde{H}_{k+1}^{-1}$ is an $N_m \times N_m$ matrix. Thus, application of Eq. 8 requires the storage and multiplication o $N_m \times N_m$ matrices. If the number of reservoir variables is very large, the form of the LBFGS, (limited memory BFGS) algorithm implemented by Nocedal (1980) becomes an extremely attractive alternative.

It is well known that the BFGS update formula of 8 can be rewritten as

$$\tilde{H}_{k+1}^{-1} = \gamma_k V_k^T \tilde{H}_k^{-1} V_k + \rho_k s_k s_k^T, \tag{10}$$

where $\rho_k = 1/y_k^T s_k$, and $V_k = I - \rho_k y_k s_k^T$. To motivate LBFGS, Nocedal rewrote Eq. 10 as

$$\begin{aligned} \tilde{H}_{k+1}^{-1} =& V_k^T V_{k-1}^T \cdots V_{k-p+1}^T (\gamma_k \tilde{H}_0^{-1}) V_{k-p+1} \cdots V_{k-1} V_k \\ &+ V_k^T \cdots V_{k-p+2}^T \rho_{k-p+1} s_{k-p+1} s_{k-p+1}^T V_1 \cdots V_k \\ &\vdots \\ &+ V_k^T \rho_{k-1} s_{k-1} s_{k-1}^T V_k \\ &+ \rho_k s_k s_k^T, \end{aligned} \tag{11}$$

where $p = \min\{L, k+1\}$ and the parameter L is an integer chosen by the user. Although we have used the same notation for the scaling parameter in Eqs. 11 and 10, the two equations are identical if and only if $\gamma_k = 1$ for al $k > 0$. Nocedal (1980) implemented Eq. 11 in a way that avoids storing any matrices except $\tilde{H}_0^{-1}$ and he suggests using a diagonal matrix for $\tilde{H}_0^{-1}$, so the storage required is minimal. His algorithm has the following advantages: (i) for $k > 0$, $\tilde{H}_k^{-1}$ is not computed explicitly or stored; instead, the vector $\tilde{H}_k^{-1} g_k$ is formed directly and the search direction δm_{k+1} is set equal to the negative of this vector; (ii) $\tilde{H}_k^{-1} g_k$ is calculated using only dot products of the vectors $\tilde{H}_0^{-1} g_0$, s_l and y_l, for $l = k-1, k-2, ..., \max\{0, k-L\}$. If L is greater than the total number o

iterations allowed, then LBFGS becomes equivalent to BFGS if $\gamma_k = 1$ for $k > 0$. Otherwise, LBFGS requires less memory and less computational time per iteration than BFGS even if Nocedal's implementation is used for the BFGS algorithm. If L is too small, however, the number of iterations required for convergence is increased. For the history-matching problems we have considered to date, $L = 30$ has proved to be a good choice. For the history-matching problems considered in this paper, larger values of L resulted in a negligible improvement in convergence properties, but using values of L much smaller than 20 resulted in a noticeable degradation in the rate of convergence and resulted in higher values of the objective function at convergence.

Nocedal suggested choosing $\tilde{H}_0^{-1}$ as a diagonal matrix. From Eq. 7, the obvious choice would then be the diagonal of C_M, but with this choice the smoothing effects of multiplication by C_M are lost and the rock property fields are much rougher than would be expected based on the prior model for reservoir variables. A better choice is $\tilde{H}_0^{-1} = C_M$, which then requires storage of C_M and multiplication of vectors by C_M using sparse matrix techniques.

Line Search

Each iteration of the algorithm requires an approximate line search to determine the step size in the search direction. The choice of the line search can affect both the convergence properties and the computational efficiency. An exact line search would determine the step size α_k so that $\alpha = \alpha_k$ minimizes $\hat{O}(m_k + \alpha\delta m_{k+1})$ over α. Each new approximation for α_k requires at least one re-evaluation of the objective function which requires one forward simulation run. As is typical in practice, we do not do an exact line search; instead, we terminate the line search when the Wolfe conditions are satisfied; see Fletcher (1987). In our approximate line search, we (a) do one iteration of the Newton-Raphson algorithm starting with an initial guess equal to zero to obtain the approximation α_k^1; (b) determine a quadratic $q(\alpha)$, such that $q(0) = \hat{O}(m_k)$, $q'(0) = \nabla\hat{O}(m_k)$ and $q(\alpha_k^1) = \hat{O}(m_k + \alpha_k^1 d_k)$ and set the new approximation of α_k equal to the minimum of $q(\alpha)$; (c) successively cut the step size by a factor of 10. We stop the line search whenever the Wolfe conditons are satisfied. In the most cases, this occurs after step (a) and the necessity to activate step (c) is extremely rare.

Computational Requirements

Here we give a rough assessment of the computational efficiency of GN (Gauss-Newton), MLM (Modified Levenberg-Marquardt), PCG (preconditioned conjugate gradient), BFGS and LBFGS. Our assessment of the computational efficiency and memory requirements of the BFGS algorithm, and our implementation of the BFGS algorithm, are based on the update formula given in Eq. 10. In the evaluation of computational efficiency, we count only the number of adjoint solutions and the number of reservoir simulation runs required by each method. Moreover, we count one adjoint solution over the total time interval of a simulation run as one equivalent simulation run, even though in our implementation, an adjoint solution typically takes less than one half the time of a simulation run. Moreover, we do not account for computational savings that may be obtained by solving the adjoint system with multiple right-hand sides in cases where several sensitivity coefficients are calculated; see Wu et al. (1999). We also assume that only one iteration of the approximate line search algorithm is done. We provide only a summary of the results without details.

In GN and MLM, if the data are evenly distributed in the time domain, $N_d/2 + 1$ simulation runs are required at each GN or MLM iteration. In BFGS, LBFGS and PCG, the total computational cost of implementing one iteration is equivalent to 3 simulation runs. Thus, BFGS, LBFGS and PCG are $(N_d/2 + 1)/3$ times faster than GN and MLM for each iteration. For example, if we have 1000 data, one iteration of GN or MLM will require 167 times as much time as one iteration of BFGS, LBFGS and PCG.

Table 1 gives a rough estimate of the number of double precision real numbers used by each algorithm when applied to minimize the objective function of Eq. 2 or Eq. 3. (Recall that N_d is the number of production data, N_m is the number of model parameters, and L is the number of previous vectors used in the LBFGS algorithm.) Only the memory used by the algorithm itself is counted, e.g., the memory required to run the reservoir simulator is not included. For convenience, we use one memory unit to stand for the memory occupied by one double precision real number. From the results of Table 1, we see that the full-memory version of BFGS uses the most memory which is on the order of N_m^2; conjugate gradient uses the least memory which is on the order of N_m, and Gauss-Newton or Levenberg-Marquardt and limited memory BFGS have intermediate memory requirements. The memory used by limited memory BFGS depends on the number of previous vectors (denoted by L in Table 1) used to construct the update of the approximate inverse Hessian.

Scaling

For BFGS and LBFGS, scaling can have a significant effect on the rate of convergence. The self-scaling variable metric (SSVM) method developed by Oren and Luenberger (1974) and Oren (1974) is motivated by the desire

Table 1: Memory used by each algorithm

	No. of DP real numbers
GN/LM	$(2 + 2 \times N_d) \times N_m$
CG	$10 \times N_m$
PCG	$10 \times N_m$ + memory for preconditioner
BFGS	$(12 + N_m) \times N_m$
LBFGS	$(9 + 2L) \times N_m$

to choose a γ_{k-1} so that the condition number of $R_k = H_k^{1/2} \tilde{H}_k^{-1} H_k^{1/2}$ is as close to one as possible. If $\tilde{H}_k^{-1}$ is identical to the inverse of the true Hessian, H_k, then this condition number is equal to one. For a quadratic objective function, these authors provide theoretical conditions and a method for computing γ_k that insure that (i) $\lambda_{\min} \leq 1 \leq \lambda_{\max}$ where $\lambda_{\min}$ and $\lambda_{\max}$, respectively, denote the minimum and maximum eigenvalues of R_k; and (ii) the condition number of R_{k+1} is less than or equal to the condition number of R_k. A quasi-Newton method which satisfies these two conditions is referred to as a self-scaling variable metric method. (Throughout, a quadratic objective function refers to a quadratic which has a Hessian matrix that is real symmetric positive definite.) For stability considerations, it is also desirable that the condition number of $\tilde{H}_k^{-1}$ not be too large, see Oren and Spedicato (1976). In particular, if $\tilde{H}_k^{-1}$ is a singular matrix, then for $l > k$, all m^l will be restricted to a subspace of N_m-dimensional Euclidean space; see Murray (1972). If the model which minimizes the objective function is not in this subspace, the algorithm can not converge to this model. Oren and Spedicato (1976) proposed an "optimal" conditioning of variable metric methods; specifically, they provide a procedure for calculating γ_k so that an upper bound for the condition numbers of $\tilde{H}_{k+1}^{-1}$ and $\tilde{H}_k \tilde{H}_{k+1}^{-1}$ is minimized. According to their results, one should actually consider switching between different update formulas from the general Broyden family from iteration to iteration. As for the results of Oren and Luenberger, these results assume that the quasi-Newton method is applied to a quadratic objective function and that an exact line search is performed at each iteration.

From computational experiments, we have found that the general switching rule proposed by Oren and Spedicato exhibits poorer convergence properties than are obtained by applying the BFGS update at every iteration and then computing the optimal γ_k from the formula they provide. In a sense, however, our preferred method for computing γ_k for LBFGS discussed later represents a modification of the switching rule proposed by Oren and Spedicato.

The principal objection to SSVM algorithms that has been raised is that the sequence $\tilde{H}_{k+1}$ does not converge to the true inverse Hessian in n iterations in the case where the objective function is an n-dimensional quadratic, whereas variable metric methods from the Broyden family satisfy this quadratic termination property if $\gamma_k = 1$ for all $k > 0$. Motivated by this reasoning, Shanno and Phua (1978) suggested that one should only scale at the first iteration and then do no further scaling.

In our work, we have found the optimal condition of Oren and Spedicato (1976) is robust and efficient if the BFGS updating formula, Eq. 8, is applied at every iteration. For the BFGS method, their procedure for calculating γ_k reduces to

$$\gamma_k = \frac{s_k^T y_k}{y_k^T \tilde{H}_k^{-1} y_k}; \tag{12}$$

we use this equation to compute the scaling factor for BFGS. Shanno and Phua (1978) suggested scaling only $\tilde{H}_0^{-1}$ using γ_0 computed from Eq. 12 with $k = 0$ and then setting $\gamma_k = 1$ for all $k > 0$. This scheme is the one referred to as the self-scaling variable metric method in Yang and Watson (1988).

For the LBFGS algorithm, we have found a variant of the optimal switching rule given by Oren and Spedicato (1976) which works well. Specifically, we compute

$$\tilde{\tau}_k = \frac{s_k^T \tilde{H}_0 s_k}{s_k^T y_k}, \tag{13}$$

$$\tilde{\sigma}_k = \frac{s_k^T y_k}{y_k^T \tilde{H}_0^{-1} y_k}, \tag{14}$$

and then determine the scaling factor γ_k by the following rule:

$$\gamma_k = \begin{cases} \tilde{\tau}_k & \text{if } \tilde{\tau}_k < 1.0 \\ \tilde{\sigma}_k. & \text{otherwise.} \end{cases} \tag{15}$$

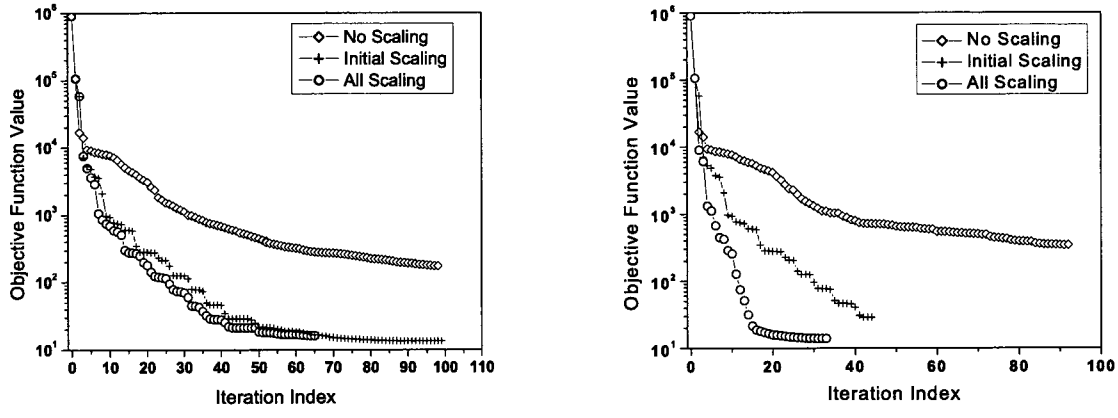
Again, we can scale only at the first iteration or scale at all iterations.

In our applications, the prior covariance matrix is used as the initial approximation to the inverse Hessian, i.e., $\tilde{H}_0^{-1} = C_M$. A somewhat simpler procedure, which yields a less robust algorithm, is to set $\tilde{H}_0^{-1}$ equal to the diagonal matrix $\tilde{D}$ obtained from C_M by setting all off diagonal elements equal to zero. Even when $\tilde{H}_0^{-1} = C_M$, we have found that it is satisfactory to compute $\tilde{\tau}_k$ by

$$\tilde{\tau}_k = \frac{s_k^T \tilde{D}^{-1} s_k}{s_k^T y_k}, \tag{16}$$

instead of using C_M^{-1} in place of $\tilde{D}^{-1}$ in Eq. 16. This method avoids solving the matrix problem $C_M x_k = s_k$ for x_k.

Figs. 1 (a) and (b) show the behavior of the objective function obtained by using BFGS and LBFGS, respectively, with different scaling schemes. The results pertain to history matching WOR, GOR and pressure data for the 2D three-phase flow example considered later in this paper. In both figures, the diamonds represent the case where BFGS and LBFGS were applied without scaling. The plus signs represent the case where BFGS and LBFGS were scaled only at the first iteration. The circles represent the case where BFGS and LBFGS were scaled at all iterations. The results clearly demonstrate the advantages of scaling; i.e., convergence is obtained in significantly fewer iterations, and at convergence, a much lower value of the objective function is obtained. Also note that scaling at all iterations is superior to scaling only at the first iteration, especially for the LBFGS algorithm. The scaling factor of Eq. 12 can not be applied with Nocedal's limited memory algorithm because $\tilde{H}_k^{-1}$, for $k > 0$, is never explicitly computed or stored. When Eq. 12, with $\tilde{H}_k^{-1}$ replaced by $\tilde{H}_0^{-1}$ was applied to compute the scaling factor for LBFGS, 74 iterations was required to obtain convergence and the value of the objective function at convergence was equal to 18. In the examples presented below, scaling was done at all iterations when using the BFGS and LBFGS algorithms.



(a) BFGS

(b) LBFGS

Fig. 1: Comparison of scaling procedures.

Results

3D single gas phase example

This example pertains to flow in a single-phase gas reservoir. Reservoir simulation was done on a $20 \times 20 \times 4$ grid. Two wells were completed in this reservoir. Well 1 was shut in for two days and then was produced at the rate of 4×10^4 Mscf/day for two days. Well 2 produced at the rate of 3.5×10^4 Mscf/day for the first two days and was then shut in for the following two days. We used 22 measured data from each well as conditioning data. Thus, a total of 44 data were history matched. The reservoir variables are the gridblock porosities, horizontal and vertical permeabilities and the well skin factors. The total number of reservoir variables is 4808. Stochastic simulation was done using the randomized maximum likelihood method. Ten realizations were generated using five different optimization algorithms: (i) MLM, (ii) preconditioned conjugate gradient (PCG) with C_M as the preconditioner (CM-PCG), (iii) PCG with the approximate inverse Hessian generated from the LBFGS equation used as the preconditioner (LBFGS-PCG), (iv) BFGS and (v) LBFGS. Table 2 gives the number of iterations required to obtain convergence and the value of the objective function at convergence for each minimization algorithm and for each realization. Results obtained by averaging the results for each set of ten conditional realizations are given

in the last column of the table. From the results, we see that LBFGS and LBFGS-PCG behave similarly. T convergence properties of both of these algorithms are superior to CM-PCG and BFGS; i.e., LBFGS and LBFG PCG require fewer iterations to obtain convergence and yield a lower value of the objective function at convergen Based on results presented in the computational requirements section, on average, LBFGS and LBFGS-PCG a about 7.7 times faster than MLM per iteration. As shown in Table 2, however, MLM requires about 2.3 times few iterations to converge. Therefore, on average, LBFGS and LBFGS-PCG are about 3.3 times faster than MLM f this particular example.

Table 2: Comparison between algorithms for a 3D single gas example.

		R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	Average
MLM	Obj.	38	30	21	33	43	28	27	38	40	34	33.2
	No. Iter.	8	14	12	8	13	13	21	8	14	10	12
CM-PCG	Obj.	153	146	70	94	347	42	184	213	230	45	152
	No. Iter.	12	20	11	23	19	64	5	17	19	23	21.3
BFGS	Obj.	59	52	35	41	70	41	39	53	54	87	53
	No. Iter.	33	17	18	19	33	38	18	42	35	52	30.5
LBFGS-PCG	Obj.	44	51	29	38	57	33	36	47	54	36	42.5
	No. Iter.	23	17	23	31	23	38	21	35	20	30	26.1
LBFGS	Obj.	43	41	31	38	55	33	35	54	54	36	42
	No. Iter.	34	22	18	38	21	35	26	24	21	34	27.3

2D three-phase example

This synthetic example pertains to a two-dimensional, three-phase flow problem simulated on 15 × 15 × 1 gri The gridblock porosities are fixed. The truth case, from which synthetic production data were generated, is show in Fig. 3 (a). Note that there are three distinct zones with log-permeability uniform in each zone. This examp has the advantage that the problem is small, so all methods require only modest computer resources. Moreove because only three log-permeability values are involved, it is easy to visualize the quality of the MAP estimate log-permeability. The prior covariance matrix is generated from an isotropic spherical variogram with range equ to 6 gridblocks. Four producers are located near the four corners of the reservoir and one injector is located at th center. GOR, WOR and flowing bottomhole pressure (p_{wf}) data from the four producers and p_{wf} data from th injector are used as observed conditioning data. The total number of data is 364.

MLM, CM-PCG, LBFGS-PCG, BFGS, LBFGS with the diagonal of C_M as the initial Hessian inverse ap proximation (LBFGS-DCM) and LBFGS with the full matrix of C_M as the initial Hessian inverse approximatio (LBFGS-FCM) were tested for this problem. Fig. 2 shows the behavior of the objective function. As shown, MLI converged in 9 iterations to an estimate m such that $O(m) = 13.3$. CM-PCG and LBFGS-PCG, respectivel reduced the objective function from 906,670 to 28.9 and 35.2 in 100 iterations; at the hundredth iteration, CM PCG and LBFGS-PCG had not converged based on the stopping criteria. The BFGS algorithm converged in 6 iterations to an estimate m such that $O(m) = 16.2$; LBFGS-DCM converged in 40 iterations to an estimate m such that $O(m) = 13.7$ and LBFGS-FCM converged in 33 iterations to an estimate m such that $O(m) = 14.$ Based on the discussion in the computational requirements section, for this example, LBFGS-FCM is expecte to be roughly 61 times faster than MLM per iteration. However, MLM required about 4 times fewer iteration to converge. Therefore, LBFGS-FCM was expected to be about 16 times faster than MLM overall based on th approximate results of the computational requirements section. Table 3 shows the real CPU time used by th different algorithms. In terms of the real CPU time, optimization with the LBFGS-FCM algorithm was about 1 times faster than optimization with MLM. Fig. 3 (b) shows the log-permeability field obtained by LBFGS-FCM We can see this model is very close to the log-permeability field obtained by the MLM method and is similar t the true model. The log-permeability field obtained with the LBFGS-DCM is not shown here but is much roughe than those shown because the smoothing effect of multiplication by C_M is lost when we use only the diagonal C_M as the initial approximate inverse Hessian.

Also note that in this case, both preconditioned nonlinear conjugate gradient methods perform relatively poorly and unlike in the gas reservoir example, generating an approximate inverse Hessian using LBFGS formulas gives worse preconditioning matrix than simply using C_M as the preconditioner. We should note, that when we appl the LBFGS formula in conjunction with the preconditioned CG algorithm, the y_k's and g_k's used in the formul are computed from the CG equations and are hence different than those that would be obtained with the actua LBFGS method.

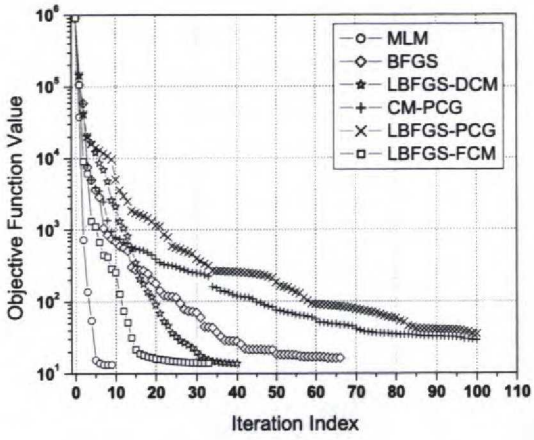
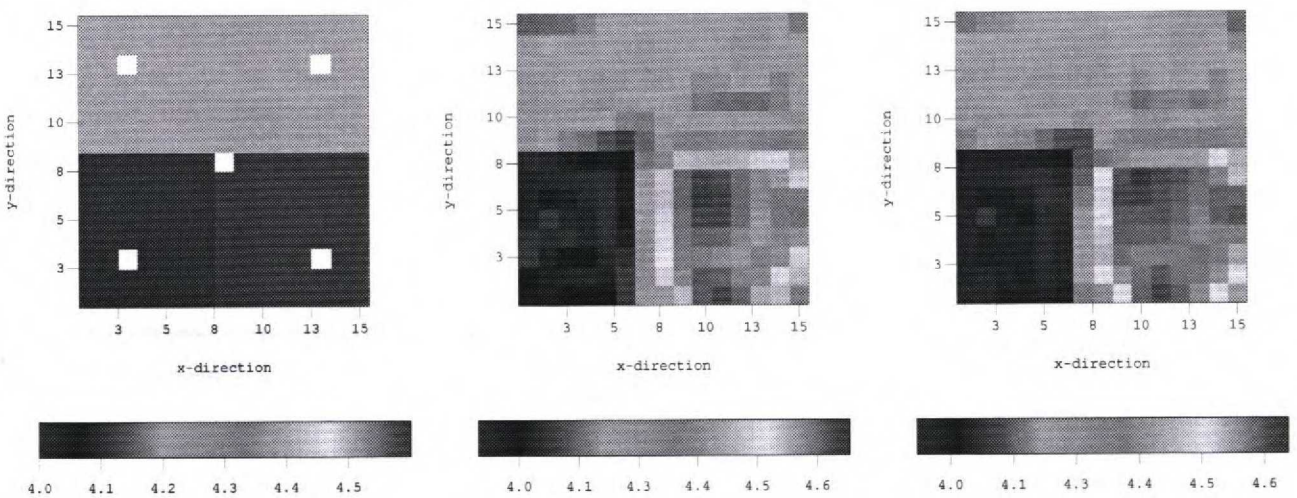


Fig. 2: Behavior of the objective function.

Table 3: Comparison of the CPU time used by different minimization algorithms

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



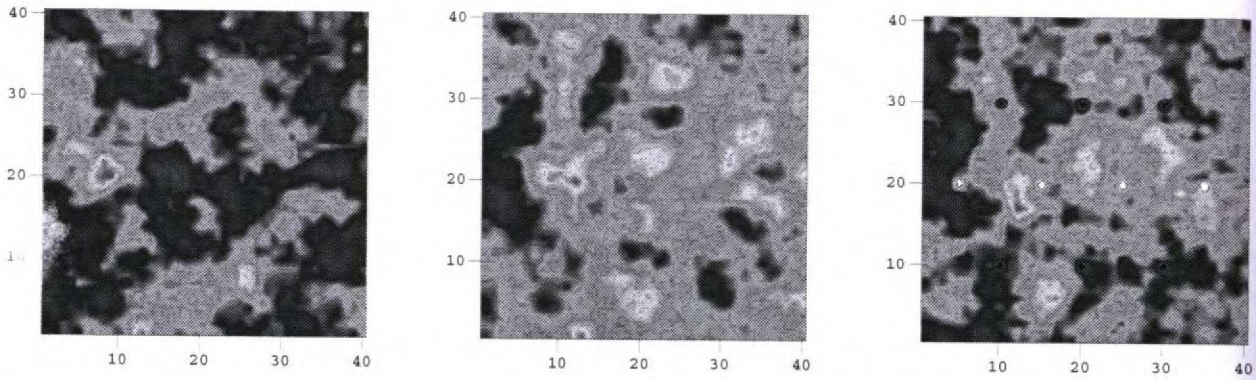
(a) True (b) LBFGS-FCM (c) MLM

Fig. 3: The log-permeability field.

3D three-phase example

In this example, a 3D three-phase flow problem on a $40 \times 40 \times 6$ grid is considered. The true log-permeability field is an unconditional realization generated by Gaussian co-simulation. The top layer of the true log-permeability field is shown in Fig. 4 (c). The porosity field is fixed. Six producers and four water injection wells are completed in the reservoir. Wellbore pressure (p_{wf}), GOR and WOR data from the producers and p_{wf} data from the injectors are used as conditioning data; the total number of data is 880. Fig. 4 (a) shows the top layer of the unconditional realization of the model which was used as the initial guess in the history-matching process. Fig. 4 (b) shows

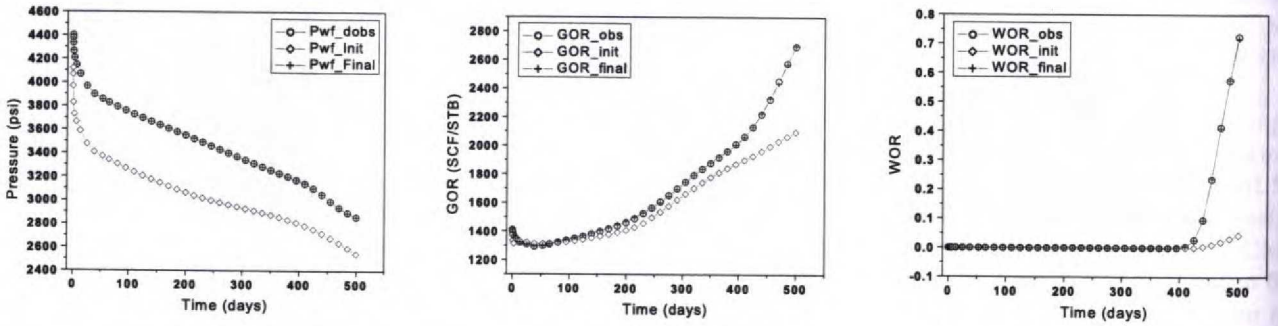
the corresponding layer of the model obtained by history matching the production data. Optimization was do with the scaled LBFGS algorithm. The objective function was reduced from a initial value of 3×10^8 to 675 in iterations. Note that the conditional realization obtained by history matching captures the main features of t true model.



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

Fig. 4: Unconditional, conditional and true realizations of the log-permeability field in the top layer.

Figs. 5 (a) through (c) show the data match for the pressure, GOR and WOR from one producer. In all the figures, diamonds represent the data obtained from the unconditional realization, i.e., the initial model, circl represent the observed data and the plus signs represent the data obtained from the model which was obtain by history matching all observed data. Note that good matches were obtained. Matches of similar quality we obtained at all wells. It is important to note that the size of the problem precluded the application of MLM a the standard BFGS algorithm with the personal computer used for the study.



(a) Pressure at producer (b) GOR (c) WOR

Fig. 5: Data matches.

Conclusions

We compared the convergence performance of a set of gradient-based nonlinear optimization algorithms includi modified Levenberg-Marquardt, preconditioned conjugate gradient, BFGS and limited memory BFGS on a set history matching problems. The implementation of BFGS used was based on explicitly computing and storing t approximate inverse Hessian at each iteration; although computationally inefficient, this implementation allows o to apply all scaling procedures that have been suggested in the literature. Our results indicate that for large sca history matching problems, the limited memory BFGS algorithm requires significantly less time and less memo than the modified Levenberg-Marquardt and BFGS algorithms, but yields results of comparable quality based

the value of the objective function obtained at convergence, and the model obtained at convergence. Scaling has a significant effect on the performance of the LBFGS and BFGS algorithms. The scaling factors used here result in significant improvement in the convergence properties of the algorithm as compared to the no scaling case. For the examples considered here, our implementations of preconditioned conjugate gradient algorithms were less robust than the scaled BFGS and LBFGS algorithms.

Acknowledgements

This work was supported by the member companies of TUPREP and the U.S. Department of Energy under Award No. DE-FC26-00BC15309. However, any opinions, findings, conclusions or recommendations herein are those of the authors and do not necessarily reflect the views of the DOE.

References

Abacioglu, Y., *The Use of Subspace Methods for Efficient Conditioning of Reservoir Models to Production Data*, Ph.D. thesis, University of Tulsa, Tulsa, Oklahoma, 2001.

Chu, L., M. Komara, and R. A. Schatzinger, An efficient technique for inversion of reservoir properties using iteration method, *SPE Journal*, **5**(1), 71–81, 2000.

Deschamps, T., T. Grussaute, D. Mayers, and R. Bissel, The results of tesing six different gradient optimisers on two history matching problems, in *European Conference on the Mathematics of Oil Recovery, VI*, pp. B–24, 1998.

Fletcher, R., *Practical Methods of Optimization*, second edn., John Wiley & Sons, New York, 1987.

Kolda, T. K., D. P. O'leary, and L. Nazareth, BFGS with update skipping and varying memory, *SIAM J. Optim*, **8**(4), 1060–1083, 1998.

Li, R., A. C. Reynolds, and D. S. Oliver, History matching of three-phase flow production data, SPE 66351, in *Proceedings of the 2001 SPE Reservoir Simulation Symposium*, 2001.

Mackie, R. L. and T. R. Madden, Three-dimensional magnetotelluric inversion using conjugate gradients, *Geophys. J. Int.*, **115**, 215–229, 1993.

Makhlouf, E. M., W. H. Chen, M. L. Wasserman, and J. H. Seinfeld, A general history matching algorithm for three-phase, three-dimensional petroleum reservoirs, *SPE Advanced Technology Series*, **1**(2), 83–91, 1993.

Masumoto, K., Pressure derivative matching method for two phase fluid flow in heterogeneous reservoir, *SPE-59462*, 2000.

Murray, W., *Numerical Methods for Unconstrained Optimization*, Academic Press, New York-London, 1972.

Nocedal, J., Updating quasi-Newton matrices with limited storage, *Math. Comp.*, **35**, 773–782, 1980.

Oliver, D. S., N. He, and A. C. Reynolds, Conditioning permeability fields to pressure data, in *European Conference for the Mathematics of Oil Recovery, V*, pp. 1–11, 1996.

Oren, S. S., Self-scaling variable metric (SSVM) algorithms II: Implementation and experiments, *Management Science*, **20**, 863–874, 1974.

Oren, S. S. and D. Luenberger, Self-scaling variable metric (SSVM) algorithms I: Criteria and sufficient conditions for scaling a class of algorithms, *Management Science*, **20**, 845–862, 1974.

Oren, S. S. and E. Spedicato, Optimal conditioning of self-scaling variable metric algorithms, *Mathematical Programming*, **10**, 70–90, 1976.

Savioli, G. B. and C. A. Grattoni, On the inverse problem application to reservoir characterization, *SPE-025522*, 1992.

Shanno, D. F. and K.-H. Phua, Matrix conditioning and nonlinear optimization, *Mathematical Programming*, **14**, 149–160, 1978.

Wu, Z., A. C. Reynolds, and D. S. Oliver, Conditioning geostatistical models to two-phase production data, *Soc. Petrol. Eng. J.*, **3**(2), 142–155, 1999.

Yang, P.-H. and A. T. Watson, Automatic history matching with variable-metric methods, *SPE Reservoir Engineering*, **3**(3), 995–1001, 1988.