# An Improved Implementation of the LBFGS Algorithm for Automatic History Matching

**Guohua Gao,** SPE, Chevron Corp.; **Albert C. Reynolds,** SPE, U. of Tulsa

**Summary**

For large scale history matching problems, where it is not feasible to compute individual sensitivity coefficients, the limited memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS) is an efficient optimization algorithm, (Zhang and Reynolds, 2002; Zhang, 2002). However, computational experiments reveal that application of the original implementation of LBFGS may encounter the following problems: (i) converge to a model which gives an unacceptable match of production data; (ii) generate a bad search direction that either leads to false convergence or a restart with the steepest descent direction which radically reduces the convergence rate; (iii) exhibit overshooting and undershooting, i.e., converge to a vector of model parameters which contains some abnormally high or low values of model parameters which are physically unreasonable. Overshooting and undershooting can occur even though all history matching problems are formulated in a Bayesian framework with a prior model providing regularization.

We show that the rate of convergence and the robustness of the algorithm can be significantly improved by: (1) a more robust line search algorithm motivated by the theoretical result that the Wolfe conditions should be satisfied; (2) an application of a data damping procedure at early iterations or (3) enforcing constraints on the model parameters. Computational experiments also indicate that (4) a simple rescaling of model parameters prior to application of the optimization algorithm can improve the convergence properties of the algorithm although the scaling procedure used can not be theoretically validated.

## Introduction

Minimization of a smooth objective function is customarily done using a gradient based optimization algorithm such as the Gauss-Newton (GN) method or Levenberg-Marquardt (LM) algorithm. The standard implementations of these algorithms (Tan and Kalogerakis, 1991; Wu et al., 1999; Li et al., 2003), however, require the computation of all sensitivity coefficients in order to formulate the Hessian matrix. We are interested in history matching problems where the number of data to be matched ranges from a few hundred to several thousand and the number of reservoir variables or model parameters to be estimated or simulated ranges from a few hundred to a hundred thousand or more. For the larger problems in this range, the computer resources required to compute all sensitivity coefficients would prohibit the use of the standard Gauss-Newton and Levenberg-Marquardt algorithms. Even for the smallest problems in this range, computation of all sensitivity coefficients may not be feasible as the resulting GN and LM algorithms may require the equivalent of several hundred simulation runs. The relative computational efficiency of GN, LM, nonlinear conjugate gradient and quasi-Newton methods have been discussed in some detail by Zhang and Reynolds (2002) and Zhang (2002).

For optimization problems which are so large that it is not feasible to compute all sensitivity coefficients by either the adjoint method (Chen et al., 1974; Chavent et al., 1975; Li et al., 2003) or the so-called gradient simulator method (Anterion et al., 1989), it is reasonable to employ methods which require only the direct computation of the gradient of the objective function to be minimized. Such methods include truncated-Newton, (nonlinear) conjugate gradient and quasi-Newton methods (Nocedal and Wright, 1999). The discussion and computational results of Zhang and Reynolds (2002) suggest that the limited memory Broyden-Fletcher-Goldfarb-Shanno algorithm is more robust and computationally efficient than truncated-Newton or nonlinear conjugate gradient for large scale history matching problems.

Our main objective here is to modify the LBFGS algorithm presented by Zhang and Reynolds (2002) to further improve its robustness and computational efficiency for large scale history matching problems formulated within a Bayesian framework. To achieve this objective, we introduce a different line search strategy, rescale model parameters, and apply damping factors to the data or apply constraints to model parameters to control overshooting and undershooting. We show that with these modifications, the LBFGS algorithm can reduce the objective function to be minimized to a value consistent with theoretical expectations. Unlike the implementation applied by Zhang and Reynolds (2002), the line search strategy used here requires that the strong Wolfe conditions are satisfied at every iteration. Theoretical convergence results on quasi-Newton methods assume that the Wolfe or strong Wolfe conditions are satisfied at every iteration. Although experience with several history matching problems indicates the strong Wolfe conditions are satisfied at most iterations with the line search strategy used in the LBFGS implementation of Zhang and Reynolds (2002), results presented here indicate that failure to satisfy the Wolfe conditions at even a couple of iterations can significantly decrease the rate of convergence.

Even with the improved line search method implemented here, undershooting and overshooting can occur, particularly in cases where the initial model gives predicted production data that differ significantly from the observed production data we wish to history match. In applying the Gauss-Newton method to history matching problems (Wu et al., 1999), used an artificially high variance of production data measurement errors at early iterations to damp the changes in model parameters and thus avoid undershooting and overshooting; for the same purpose, Li et al. (2003) applied a modified Levenberg-Marquardt algorithm. Here, we employ a damping procedure to control undershooting/overshooting problems. We also introduce a constrained version of the LBFGS algorithm to control undershooting/overshooting. The algorithm introduces a natural rescaling of model parameters so that a prior rescaling is unnecessary.

We illustrate the significantly improved convergence properties of the new implementation of LBFGS by history matching multiphase flow production data for example problems including the well known PUNQS3 example (Floris et al., 2001).

## Determination of Search Direction with the LBFGS Algorithm

Throughout, $m$ denotes the $N_m$-dimensional column vector of model parameters we wish to estimate. (In our history matching examples, entries of $m$ will correspond to gridblock porosities and horizontal and vertical log-permeabilities.) Here, we let $f(m)$ denote a generic objective function that we wish to minimize and $m_k$ denotes the estimate of $m$ obtained at the $k$th iteration of the optimization algorithm with $m_0$ denoting the initial guess.

In gradient based methods which incorporate a line search, a



This paper (SPE 90058) was first presented at the 2004 SPE Annual Technical Conference and Exhibition, Houston, 26–29 September. Original manuscript received for review 18 January 2005. Revised manuscript received 17 August 2005. Manuscript peer approved 18 August 2005.

downhill search direction, $p_k$ at iteration $k+1$ is determined as

$$p_k = -B_k \nabla f_k, \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (1)$$

where $B_k$ is an $N_m \times N_m$ positive definite matrix and $\nabla f_k$ denotes the gradient of $f$ with respect to $m$ evaluated at $m_k$.

If the Hessian matrix $H_k$ evaluated at $m_k$ is positive definite, it is appropriate to replace $B_k$ by the inverse of $H_k$ which represents Newton's method. However, calculation of the Hessian requires computation of the sensitivity coefficient matrix, which is not computationally feasible if both the number of observed data and the number of model parameters are large.

The basic idea of a quasi-Newton method is to replace $H_k^{-1}$ by some approximation which can be generated with much less computational work than would be required to calculate all the entries of $H_k$ and then solve $H_k p_k = -\nabla f_k$ to obtain the search direction $p_k$. Moreover, the approximation to the inverse Hessian should be positive definite to ensure that we always obtain a downhill search direction. We denote the approximate Hessian inverse by $\tilde{H}_k^{-1}$. The objective function in some neighborhood of $m_k$ is approximated by a quadratic model given by

$$q_k(p) = f_k + [\nabla f_k]^T p + \frac{1}{2} p^T \tilde{H}_k p, \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (2)$$

where $p$ is an $N_m$-dimensional column vector. Note if $\tilde{H}_k$ is the true Hessian, then $q_k(p)$ is simply a second order Taylor series approximation for the objective function expanded about $m_k$. If $\tilde{H}_k$ is positive definite, then $q_k$ has a unique minimum which is taken on when $p = p_k$, where

$$p_k = -\tilde{H}_k^{-1} \nabla f_k. \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (3)$$

The vector $p_k$ represents the search direction. Letting $\alpha_k > 0$ denote the stepsize determined by a line search procedure, the new estimate of the model that minimizes the objective function $f(m)$ is given by

$$m_{k+1} = m_k + \alpha_k p_k. \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (4)$$

Similar to Eq. 2, we can approximate the objective function by the quadratic

$$q_{k+1}(p) = f_{k+1} + [\nabla f_{k+1}]^T p + \frac{1}{2} p^T \tilde{H}_{k+1} p, \quad \ldots\ldots\ldots \quad (5)$$

where the subscript $k+1$ denotes evaluation at $m_{k+1}$. It is reasonable to require that $\tilde{H}_{k+1}$, the approximate Hessian at the next iteration, be determined so that the gradient of $q_{k+1}(p)$ evaluated at $p = -\alpha_k p_k$ is equal to the gradient of $q_k(p)$ evaluated at $p = 0$, i.e., to require that both $q_k(p)$ and $q_{k+1}(p)$ have the same gradient at $m_k$. Note this requirement would be satisfied if the true objective function were a quadratic in a neighborhood that included both $m_{k+1}$ and $m_k$. Based on this requirement and Eqs. 2 and 5, it is easy to show (Nocedal and Wright, 1999), that the updated Hessian $\tilde{H}_{k+1}$ should satisfy the following condition:

$$\tilde{H}_{k+1}^{-1} y_k = s_k, \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (6)$$

where

$$s_k = \alpha_k p_k = m_{k+1} - m_k, \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (7)$$

and

$$y_k = \nabla f_{k+1} - \nabla f_k. \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (8)$$

Eq. 6 is referred to as the quasi-Newton condition or the secant equation.

If $\tilde{H}_{k+1}^{-1}$ is positive definite, we must have $y_k^T \tilde{H}_{k+1}^{-1} y_k > 0$ for any nonzero $y_k$. It follows that the quasi-Newton condition of Eq. 6 can hold only if

$$y_k^T s_k = s_k^T y_k > 0. \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (9)$$

Thus, it is critical that the line search generate a step size $\alpha_k$ such that $s_k = \alpha_k p_k$ and $y_k$ satisfies Eq. 9.

There are infinite number of positive symmetric matrices that satisfy the secant equation (Eq. 6). Among them, the one that is closest to the current approximate inverse Hessian $\tilde{H}_k^{-1}$ is chosen as the updated inverse Hessian for the next iteration. As shown in Nocedal and Wright (1999), this choice corresponds to the Broyden-Fletcher-Goldfard-Shanno updating formula, which can be written as

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

where $\rho_k = 1/y_k^T s_k$, $V_k = I - \rho_k y_k s_k^T$, $\gamma_k$ is the scaling factor for the $k$th iteration and $I$ is the $N_m \times N_m$ identity matrix. If no scaling is done, $\gamma_k = 1$ for all $k$. If scaling is done only at the first iteration ($k = 0$), then $\gamma_k = 1$ for $k > 0$. Detailed theoretical and practical information on the choice of the scaling factors can be found in Zhang and Reynolds (2002) and Zhang (2002) and the many references provided therein. In our applications, we use scaling at each iteration with the scaling factor computed in the same way as in Zhang and Reynolds (2002).

The LBFGS formula for calculating the search direction $p_{k+1} = -\tilde{H}_{k+1}^{-1} \nabla f_{k+1}$ is derived by substituting Eq. 10 into Eq. 3 with $k$ replaced by $k+1$. Following Nocedal (1980) and Nocedal and Wright (1999), the resulting equation can be written as

$$\begin{aligned} p_{k+1} = & -\big(V_k^T V_{k-1}^T \cdots V_{k-p+1}^T (\gamma_k \tilde{H}_0^{-1}) V_{k-p+1} \cdots V_{k-1} V_k \\ & + V_k^T \cdots V_{k-p+2}^T \rho_{k-p+1} s_{k-p+1} s_{k-p+1}^T V_{k-p+2} \cdots V_k \\ & \vdots \\ & + V_k^T \rho_{k-1} s_{k-1} s_{k-1}^T V_k + \rho_k s_k s_k^T \big) \nabla f_{k+1}. \quad \ldots\ldots \quad (11) \end{aligned}$$

In the preceding equation, $p = \min\{L, k+1\}$ where the integer $L$ specifies the number of previous $s_k$ and $y_k$ vectors used to update the approximation to the inverse Hessian. With this approach the only inverse Hessian that needs to be calculated or stored is $\tilde{H}_0^{-1}$, the initial approximation to the inverse Hessian. As in Zhang and Reynolds (2002), we use Nocedal's implementation which applies Eq. 11 in a way that requires only the computation of dot products involving the vectors $\tilde{H}_0^{-1} \nabla f_0$, $s_l$ and $y_l$, for $l = k-1, k-2, \ldots, \max\{0, k-L\}$. If $k+1 < L$, then $p = k+1$, and Eq. 11 reduces to standard BFGS algorithm if $\gamma_k = 1$ for $k > 0$. When $k+1 > L$, the LBFGS algorithm only uses the gradient information from the latest $L$ iterations, while the BFGS algorithm uses the gradient information of all $k$ iterations. In this case, LBFGS requires less memory and less computational time per iteration than BFGS even if Nocedal's implementation is used for the BFGS algorithm. For the examples they considered, Zhang and Reynolds (2002) found that $L = 30$ was sufficient and using larger values of $L$ did not significantly improve the rate of convergence. For the examples presented in this paper, we used $L = 100$, but did not do computational experiments to try to find the minimum satisfactory value of $L$. Our choice of $L$ was based on two factors: (i) the results of Zhang and Reynolds (2002) suggest we choose $L \geq 30$; (ii) it is known that if $L$ is too small, the number of iterations required for convergence is greater and the algorithm tends to be less robust (Nocedal and Wright, 1999). Moreover, as the optimal choice of $L$ is problem dependent, we tried to ensure that we did not use a value of $L$ so small that it would degrade the performance of the algorithm.

#### Determination of Step Size

After the search direction $p_k$ has been computed from Eq. 11, the step size is computed by a line search procedure. An exact line search would determine the step size $\alpha_k$ as the value of $\alpha$ that minimizes $\phi(\alpha) = f(m_k + \alpha p_k)$, i.e.,

$$\alpha_k = \operatorname{argmin} f(m_k + \alpha p_k). \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (12)$$

For a nonlinear problem, a line search requires an iterative procedure and each iteration requires at least one evaluation of the ob-

jective function $f(m)$. Thus, in practice, an exact line search is normally replaced by an approximate line search. For the history matching problems considered here, evaluation of $f(m)$ requires one run of the reservoir simulation so it is important to implement a reliable line search procedure which requires a very small number of iteration to obtain an approximation of $\alpha_k$ which gives an adequate decrease in the objective function.

**The Strong Wolfe Conditions.** In order to prove that the LBFGS algorithm converges, it is necessary to impose some conditions on the step size $\alpha_k$ to guarantee that the line search is done accurately enough to provide a sufficient decrease in the objective function at each iteration (Nocedal and Wright, 1999). To prove that the LBFGS method converges, the imposed conditions must also ensure that the updated approximation to the inverse Hessian is positive definite. One standard way to ensure a positive definite approximate inverse Hessian and guarantee convergence is to require that the line search satisfy the strong Wolfe conditions (Dennis and Schnabel, 1996; Nocedal and Wright, 1999). The strong Wolfe conditions require the step size $\alpha_k$ to satisfy

$$f(m_k+\alpha_k p_k) \le f(m_k)+c_1\alpha_k[\nabla f(m_k)]^T p_k, \quad \ldots\ldots\ldots \quad (13)$$

$$|[\nabla f(m_k+\alpha_k p_k)]^T p_k| \le c_2|[\nabla f(m_k)]^T p_k|, \quad \ldots\ldots\ldots \quad (14)$$

where $0 < c_1 < 0.5$ and $c_1 < c_2 < 1$. With $\phi(\alpha)$ defined by

$$\phi(\alpha) = f(m_k+\alpha p_k), \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (15)$$

Eqs. 13 and 14 can be rewritten as

$$\phi(\alpha_k) \le \phi(0)+c_1\alpha_k\phi'(0), \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (16)$$

$$|\phi'(\alpha_k)| \le c_2|\phi'(0)|. \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (17)$$

The first strong Wolfe condition (Eq. 13 or 16) essentially stipulates that the step size $\alpha_k$ should give a sufficient decrease in the objective function but actually only guarantees this if the step size is bounded away from zero. In **Fig. 1,** the solid curve represents $\phi(\alpha)$; the dotted line connecting the solid squares represents the line $\phi_1(\alpha) = \phi(0) + c_1\phi'(0)\alpha$. In Eqs. 13 and 16, any very small value of $c_1$ will do; we use $c_1 = 10^{-4}$, the value recommended by Nocedal and Wright (1999). The first strong Wolfe condition is not enough by itself to ensure that the objective function decreases sufficiently, because, as shown in Fig. 1, it is satisfied for all sufficiently small step sizes. An unacceptably small step size is ruled out by applying the second strong Wolfe condition or curvature condition (Eq. 14, or 17). The two dashed lines connecting open circles in Fig. 1 illustrate the second strong Wolfe condition. These two lines are the tangential lines to the curve defined by $\phi(\alpha)$ at $\alpha_1$ and $\alpha_2$ with slopes given by $\phi'(\alpha_1) = c_2\phi'(0)$ and $\phi'(\alpha_2) = -c_2\phi'(0)$. The second strong Wolfe condition states that step size $\alpha$ is acceptable only if $|\phi'(\alpha)| \le c_2|\phi'(0)|$, i.e., $\alpha$ is in the interval $[\alpha_1, \alpha_2]$. The smaller the value of $c_2$, the smaller the distance between $\alpha_1$ and $\alpha_2$. Choosing a smaller value for $c_2$ makes the acceptable step size closer to the minimizer, however, it requires more iterations of the line search algorithm to obtain an $\alpha$ that satisfies the second strong Wolfe condition. The choice of $c_2$ is ad hoc. Nocedal and Wright (1999) recommended $c_2 = 0.9$ for quasi-Newton algorithms, and we use this value in the example presented later.

As stated previously, it is critical that the line search generate a step size $\alpha_k$ such that $s_k = \alpha_k p_k$ satisfies Eq. 9. We now show this is true provided the second strong Wolfe condition holds. From the second strong Wolfe condition, we know $[\nabla f_{k+1}]^T p_k \ge c_2[\nabla f_k]^T p_k$ since $[\nabla f_k]^T p_k < 0$ when $p_k$ is a downhill direction. It follows that

$$\begin{aligned} y_k^T s_k &= [\nabla f_{k+1} - \nabla f_k]^T \alpha_k p_k \\ &= \alpha_k[\nabla f_{k+1}]^T p_k - \alpha_k[\nabla f_k]^T p_k \\ &\ge (c_2-1)\alpha_k[\nabla f_k]^T p_k. \quad \ldots\ldots\ldots\ldots\ldots \quad (18) \end{aligned}$$

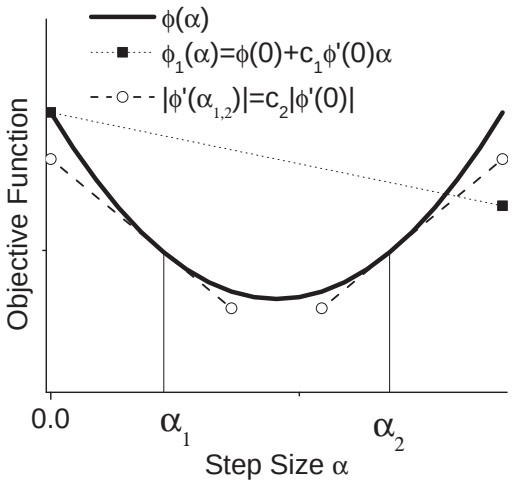



**Fig. 1—Illustration of Wolfe conditions for line search.**

Thus $y_k^T s_k > 0$ is satisfied if we choose $c_2 < 1$ and the second strong Wolfe condition holds. The second strong Wolfe condition not only guarantees a sufficient decrease of the objective function, but also guarantees that the updated Hessian inverse $\tilde{H}_{k+1}^{-1}$ is positive definite. If the second strong Wolfe condition fails at one iteration, the updated Hessian inverse, $\tilde{H}_{k+1}^{-1}$, may be not positive definite and may result in an uphill search direction at the next iteration.

**Previous Line Search Algorithms.** A popular line search algorithm was developed by Nocedal and Wright (1999). It is a two stage algorithm. The first stage is a trial stage to obtain a step size that brackets an interval within which the Wolfe conditions are satisfied. The second stage is a zoom stage to find a step size that satisfies the Wolfe conditions by applying a quadratic fit, a cubic fit or a bisection algorithm. Although robust, bisection is generally not an efficient method for a one-dimensional line search. A quadratic fit may generate negative or even infinite step size and thus fail to guarantee the strong Wolfe conditions. (The line search algorithm implemented by Zhang and Reynolds (2002) utilizes a quadratic fit at the second iteration of the line search algorithm and may also encounter this same difficulty.)

Fitting a quadratic through $\phi(0)$, $\phi'(0)$ and $\phi(\alpha_k)$ and finding the value at which this quadratic is a minimum gives

$$\alpha_k^{new} = -\frac{\phi'(0)\alpha_k^2}{2[\phi(\alpha_k)-\phi(0)-\phi'(0)\alpha_k]}, \quad \ldots\ldots\ldots\ldots \quad (19)$$

where $\phi(\alpha_k) = f(m_k+\alpha_k p_k)$, and $\phi'(0) = [\nabla f(m_k)]^T p_k$. If $\phi(\alpha)$ is strongly convex, then $\phi(\alpha_k)-\phi(0)-\phi'(0)\alpha_k > 0$ and Eq. 19 yields a positive step size. However, if the objective function is not convex, then a negative or even an infinite step size may be obtained from this equation.

For the history matching problems of interest to us, we have encountered two common situations where the objective function along a line in the direction of the search direction may be non-convex. The first case pertains to the situation where the current model, $m_k$, is close to the "critical" point at which gas or water breaks through in one or more of the production wells. In this case, if a step size moves the model across the critical point, then the objective function may decrease more quickly as the step size increases because the GOR or WOR mismatch term may decrease dramatically. A non-convex objective function may also occur when the well production condition changes from constant bottom hole pressure to a constant production rate. In this case, the pressure mismatch part of the objective function will decrease more rapidly as the step size increases.

**Improved Line Search Algorithm.** To avoid the problems discussed above and guarantee that we obtain a step size that satisfies the Wolfe conditions, we introduce an improved line search procedure. The procedure is based on using a normalized search direction and a normalized step size, which are defined respectively by

$$p_{N,k} = \frac{p_k}{\|p_k\|_\infty}, \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (20)$$

and

$$\beta_k = \alpha_k \|p_k\|_\infty. \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (21)$$

Here $\|.\|_\infty$ denotes the infinity norm. Note Eq. 4 may be rewritten as

$$m_{k+1} - m_k = \alpha_k p_k = \beta_k p_{N,k}. \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (22)$$

The motivation for using the normalized step size is that although the value of $\alpha_k$ may change significantly from iteration to iteration **( Fig. 2),** the value of the normalized step size $\beta_k$ does not vary as much from iteration to iteration **(Fig. 3)**. Thus, the value of the normalized step size at iteration $k-1$ provides an initial guess for the normalized step size at iteration $k$. Note that in Figs. 2 and 3, we have also shown results where we used the $\ell_2$ norm in place of the infinity norm. The results suggests that the choice of the norm does not impact the results.

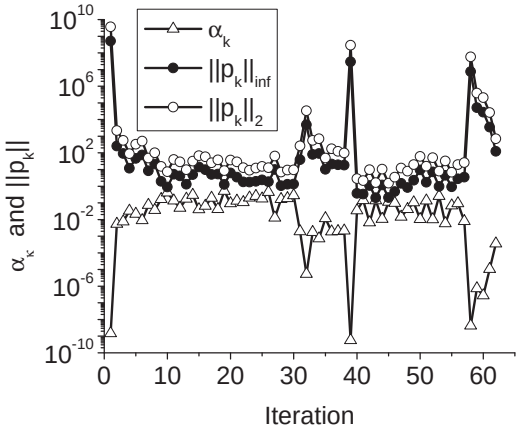



**Fig. 2—Step size and the norm of the search direction vector.**

In the line search algorithm, $\beta_{k,1}$ and $\beta_{k,2}$ will denote the two latest approximations to the normalized stepsize, $\beta_k$. At the first iteration of the line search, we set $\beta_{k,1} = 0$ and generate $\beta_{k,2}$ from $\beta_{k-1}$, the final stepsize used in the previous LBFGS iteration to compute $m_k = m_{k-1} + \beta_{k-1} p_{N,k-1}$. For now, we assume the initial value of $\beta_{k,2}$ is equal to $\beta_{k-1}$; an alternate procedure for generating the initial value of $\beta_{k,2}$ is presented later. With these choices and $\phi(\beta)$ defined by

$$\phi(\beta) = f(m_k + \beta p_{N,k}), \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (23)$$

it follows that

$$\begin{aligned} &\beta_{k,2} > \beta_{k,1} \geq 0, \\ &\phi(\beta_{k,1}) \leq \phi(0), \\ &\phi'(\beta_{k,1}) < 0, \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \end{aligned} \quad (24)$$

The line search algorithm we use guarantees that Eq. 24 holds for each pair of $\beta_{k,1}$ and $\beta_{k,2}$ generated during the line search iterative process. At each step of the iterative process, we check to see if the newest positive value of $\beta_k$ satisfies the strong Wolfe condition. At the first iteration of the line search, this means that if setting $\beta_k = \beta_{k,2}$ and computing $m_{k+1}$ from Eq. 22 gives a result such

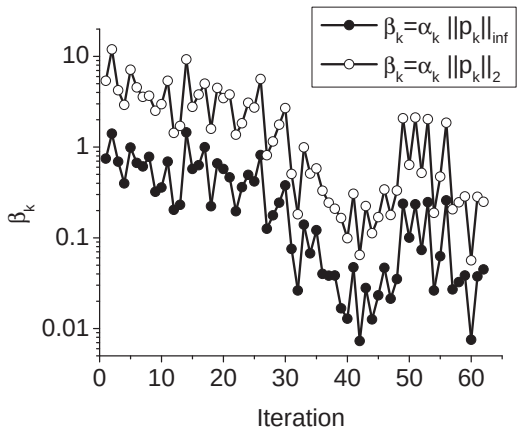



**Fig. 3—Normalized step size plot in two and infinity norms.**

that the strong Wolfe conditions are satisfied, then the line search is complete, $m_{k+1}$ is accepted as the updated model and we proceed to the next iteration of the LBFGS algorithm. For the examples we have done with the initial value of $\beta_{k,2}$ generated by the procedure given later, this situation occurs at about two-thirds of the iterations and there is no need to use a line search iteration procedure.

When the initial value of $\beta_{k,2}$ does not satisfy the strong Wolfe conditions, a line search iteration procedure is used. The algorithm is based on finding an interval $[\beta_{k,1}, \beta_{k,2}]$ which contains a subinterval on which the strong Wolfe conditions are satisfied. Then we can reduce the size of this interval until we find a value of $\beta_k$ which satisfies the strong Wolfe conditions. To update the interval, we first suppose that the derivative of $\phi(\beta)$ can be approximated by a line through the two points $(\beta_{k,1}, \phi'(\beta_{k,1}))$ and $(\beta_{k,2}, \phi'(\beta_{k,2}))$. The equation of this line is given by

$$\ell(\beta) = \phi'(\beta_{k,2}) + \Big(\frac{\phi'(\beta_{k,1}) - \phi'(\beta_{k,2})}{\beta_{k,1} - \beta_{k,2}}\Big)(\beta - \beta_{k,2}). \quad \ldots \quad (25)$$

This line crosses the $\beta$ axis at $\beta = \beta_{k,\text{new}}$ where

$$\begin{aligned} \beta_{k,\text{new}} &= \beta_{k,1} + \frac{1}{1-\rho}(\beta_{k,2} - \beta_{k,1}) \\ &= \beta_{k,2} + \frac{\rho}{1-\rho}(\beta_{k,2} - \beta_{k,1}), \quad \ldots\ldots\ldots\ldots\ldots \end{aligned} \quad (26)$$

with $\rho$ given by

$$\rho = \frac{\phi'(\beta_{k,2})}{\phi'(\beta_{k,1})}. \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (27)$$

Note that if $\phi(\beta)$ is a convex quadratic function, then $\beta_k^{new}$ gives the unique minimum of $\phi(\beta)$, i.e., corresponds to an exact line search. In general, however, this is not the case. Nevertheless, we show that we can use $\beta_{k,\text{new}}$ to update our approximation to $\beta_k$ or to the interval which contains a subinterval on which the strong Wolfe conditions hold.

Eq. 26 may generate negative ($\rho > 1$) or infinite ($\rho = 1$) values of $\beta_{k,\text{new}}$, or the new step size generated may be too close to $\beta_{k,2}$ ($\rho \to 0$) or $\beta_{k,1}$ ($\rho \to \infty$). To avoid these undesirable features, we set minimum and maximum allowable values of $\rho$. These values depend on whether the value of $\rho$ computed from Eq. 27 is negative or not and are specified by

$$\rho_{\min} = \begin{cases} -4, & \text{if } \rho < 0 \\ 0.25, & \text{if } \rho \geq 0 \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (28)$$

and

$$\rho_{\max} = \begin{cases} -0.25, & \text{if } \rho < 0 \\ 0.75. & \text{if } \rho \geq 0 \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (29)$$

If the value of $\rho$ computed from Eq. 27 is smaller than $\rho_{min}$, we set $\rho = \rho_{min}$ and recompute $\beta_{k,new}$ from Eq. 26. If $\rho$ is greater than $\rho_{max}$, then we set $\rho = \rho_{max}$ and recompute $\beta_{k,new}$. If the value of $\rho$ computed from Eq. 27 satisfies $\rho > 0$ and $\phi(\beta_{k,2}) > \phi(\beta_{k,1})$, then we simply set $\rho = -1$ which gives a modified value of $\beta_{k,new}$ which is equal to the average of $\beta_{k,1}$ and $\beta_{k,2}$. Because the algorithm always ensures that $\phi'(\beta_{k,1}) < 0$, the condition $\rho > 0$ implies that $\phi'(\beta_{k,2}) < 0$. If the derivative is negative at both values of $\beta$ and $\phi(\beta_{k,2}) > \phi(\beta_{k,1})$, then $\phi(\beta)$ must have both a local maximum and a local minimum on the interval $[\beta_{k,1}, \beta_{k,2}]$. We are interested in finding the subinterval on which the minimum occurs and setting $\rho = -1$ corresponds to using bisection to locate this subinterval. Note that the procedures for modifying $\rho$ ensure that the final value of $\rho$ is not equal to zero. Since we always ensure that $\beta_{k,1} < \beta_{k,2}$ (see Eq. 24), any value of $\beta_{k,new}$ computed directly from Eq. 26 satisfies $\beta_{k,new} > \beta_{k,1}$ because Eq. 29 guarantees $\rho < 1$.

As soon as $\beta_{k,new}$ is computed we check to see if the strong Wolfe conditions are satisfied with $\beta_k = \beta_{k,new}$ and $m_{k+1}$ computed from Eq. 22. If so, the search is terminated and $m_{k+1}$ gives the updated model at iteration $k+1$ of the LBFGS algorithm. If the strong Wolfe conditions are not satisfied, then we update $\beta_{k,1}$ and $\beta_{k,2}$ according to the detailed procedure given below. There are two cases to consider.

- Case 1: This case applies if $\rho < 0$ or $\phi(\beta_{k,2}) \geq \phi(\beta_{k,1})$. If $\phi'(\beta_{k,new}) < 0$ and $\phi(\beta_{k,new}) \leq \phi(\beta_{k,1})$, then $\beta_{k,2}$ remains unchanged and we set $\beta_{k,1} = \beta_{k,new}$; otherwise, $\beta_{k,1}$ remains unchanged and we set $\beta_{k,2} = \beta_{k,new}$.
- Case 2: This is the case where $\rho > 0$ and $\phi(\beta_{k,2}) < \phi(\beta_{k,1}) \leq \phi(0)$. When $\rho > 0$, Eqs. 24 and 27 ensure that $\phi'(\beta_{k,1})$ and $\phi'(\beta_{k,2})$ are both negative. Since our procedures for selecting $\rho$ also guarantee that $\rho \leq .75$ and we always have $\beta_{k,1} < \beta_{k,2}$ (see Eq. 24), it follows from Eq. 26 that $\beta_{k,new} > \beta_{k,2} > \beta_{k,1}$. Since we also have $\phi(\beta_{k,2}) < \phi(\beta_{k,1})$, one can find a value of $\beta_k$ greater than $\beta_{k,2}$ which gives a value of $\phi$ smaller than $\phi(\beta_{k,2})$ so we should move the "search interval" to the right. Thus, when Case 2 applies, we update the $\beta_k$ interval by first setting $\beta_{k,1} = \beta_{k,2}$ and then setting $\beta_{k,2} = \beta_{k,new}$.

The motivation for Case 1 is most easily understood by considering two separate subcases. (i) This pertains to the situation where the original $\rho$ calculated from Eq. 27 is positive and $\phi(\beta_{k,2}) \geq \phi(\beta_{k,1})$, so we set $\rho = -1$ and $\beta_{k,new} = 0.5(\beta_{k,1} + \beta_{k,2})$. Then, as noted previously, the interval $[\beta_{k,1}, \beta_{k,2}]$ must contain a point which corresponds to a minimum of $\phi(\beta)$. Since we always have $\phi'(\beta_{k,1}) < 0$, if $\phi'(\beta_{k,new}) < 0$ and $\phi(\beta_{k,new}) \leq \phi(\beta_{k,1})$, we can find a value of $\beta_k > \beta_{k,new}$ such that $\phi(\beta_k) < \phi(\beta_{k,new}) \leq \phi(\beta_{k,1})$ so we should update the $\beta_k$ search interval by setting $\beta_{k,1} = \beta_{k,new}$ and leaving $\beta_{k,2}$ unchanged. (ii) This case pertains to the case where the original $\rho$ calculated from Eq. 27 is negative, so even if this value is modified by applying Eq. 28 or Eq. 29, $\rho$ remains negative. Thus Eq. 27 implies $\phi'(\beta_{k,2}) > 0$ and Eqs. 26 and 24 imply that $\beta_{k,new} < \beta_{k,2}$. (iia) If $\phi'(\beta_{k,new}) < 0$ and $\phi(\beta_{k,new}) \leq \phi(\beta_{k,1})$, then there must exist a value of $\beta_k > \beta_{k,new}$ which gives a value of $\phi$ which is smaller than $\phi(\beta_{k,1})$. Thus we should set $\beta_{k,1} = \beta_{k,new}$ and keep $\beta_{k,2}$ unchanged. Otherwise, we must have $\phi'(\beta_{k,new}) > 0$ or $\phi(\beta_{k,new}) > \phi(\beta_{k,1})$. Since $\phi'(\beta_{k,1}) < 0$ and $\phi'(\beta_{k,2}) > 0$, if $\phi'(\beta_{k,new}) > 0$, then $\phi$ takes on a minimum at some value of $\beta_k$ between $\beta_{k,1}$ and $\beta_{k,new}$ and we should set $\beta_{k,2} = \beta_{k,new}$ and keep $\beta_{k,1}$ unchanged regardless of whether $\phi(\beta_{k,new})$ is larger or smaller that $\phi(\beta_{k,1})$. (iib) If $\phi'(\beta_{k,new}) < 0$, but $\phi(\beta_{k,new}) > \phi(\beta_{k,1})$, then $\phi$ must have both a minimum and a maximum between $\beta_{k,1}$ and $\beta_{k,new}$ and, again, we should set $\beta_{k,2} = \beta_{k,new}$ and keep $\beta_{k,1}$ unchanged.

Assuming that $\phi(\beta) = f(m_k + \beta p_{N,k})$ has a minimum, this process will eventually find an interval $(\beta_{k,1}, \beta_{k,2})$ which contains a minimum and from this point on, the length of this interval will be decreased until we finally obtain a step size $\beta_{k,new}$ that satisfies the strong Wolfe conditions. (At a minimum, the strong Wolfe conditions will hold provided $c_1$ is sufficiently small.)

If $f$ is an arbitrary objection function, there is no guarantee that a minimum exists along a downhill search direction as $\phi(\beta)$ could be a monotonically decreasing function of $\beta$. However, with the Bayesian formulation used here, the objective function that is minimized is bounded below by zero and it is easy to show that, under reasonable conditions, we must be able to find a value of $\beta$ at which the strong Wolfe conditions are satisfied. In the case where the objective function $f$ is non-negative, it is easy to show that there exists a $\beta$ which satisfies the strong Wolfe conditions if the following limit exists:

$$\lim_{\beta \to \infty} \phi'(\beta) = \lim_{\beta \to \infty} [\nabla f(m_k + \beta p)]^T p. \quad \ldots\ldots\ldots\ldots\ldots \quad (30)$$

Note that if $f(m)$ is non-negative for all $m$, $\phi(\beta) = f(m_k + \beta p)$ is also non-negative. It follows that the limit in Eq. 30 is non-negative, for if this limit were negative, $\phi(\beta)$ would have to be negative for sufficiently large values of $\beta$. If the limit is positive, then there exists a $\beta$ with $\phi'(\beta) > 0$ so $\phi$ has a minimum on $[\beta_{k,1}, \beta]$. If the limit in Eq. 30 is zero, then there exists a $\beta_n > 0$ such that for all $\beta > \beta_n$, $\phi'(\beta)$ is sufficiently small so that the second strong Wolfe condition is satisfied. The first strong Wolfe condition will also be satisfied if $c_1$ is sufficiently small.

**Initial Guess of Step Size for Each Iteration.** Although the final value of $\beta_{k-1}$ used to compute $m_k = m_{k-1} + \beta_{k-1} p_{N,k-1}$ can be used as the initial approximation for $\beta_k$, we found that a better initial approximation ($\beta_{k,2}$) for $\beta_k$ can be obtained. The procedure assumes that $\phi_{k-1}(\beta)$ defined by $\phi_{k-1}(\beta) = f(m_{k-1} + \beta p_{N,k-1})$ can be approximated by a convex quadratic function in a neighborhood that includes $\beta_{k-1}$, $\beta_{k-1,1}$ and $\beta_{k-1,2}$ so that $\phi'_{k-1}(\beta)$ can be approximated by a linear function of $\beta$. More specifically, we approximate $\phi'_{k-1}(\beta)$ by a line through the two points $(\beta_{k-1,0}, \phi'_{k-1}(\beta_{k-1,0}))$ and $(\beta_{k-1}, \phi'_{k-1}(\beta_{k-1}))$. Here, we choose $\beta_{k-1,0}$ based on the following procedure: (1) set $\beta_{k-1,0} = 0$ if the step size fit procedure discussed above is not invoked, i.e., if the Wolfe conditions were satisfied with the initial guess $\beta_{k-1,2}$ for $\beta_{k-1}$ so that no iterations of our line search procedure were done; (2) otherwise, set $\beta_{k-1,0} = \beta_{k-1,1}$ if $\phi'_{k-1}(\beta_{k-1}) \geq 0$; or set $\beta_{k-1,0} = \beta_{k-1,2}$ if $\phi'_{k-1}(\beta_{k-1}) < 0$. The motivation for (2) can easily be explained. If $\phi'_{k-1}(\beta_{k-1}) > 0$, then $\phi'_{k-1}(\beta)$ must be zero at some point between $\beta_{k-1,1}$ and $\beta_{k-1}$. On the other hand, if $\phi'_{k-1}(\beta_{k-1}) < 0$, then a minimum of $\phi'_{k-1}(\beta)$ is located beyond $\beta_{k-1} \geq \beta_{k-1,1}$ so it seems preferable to approximate $\phi'_{k-1}(\beta)$ by a line on the interval $[\beta_{k-1}, \beta_{k-1,2}]$ and use the value of $\beta$ where this line crosses the $\beta$ axis as the approximation of the zero of $\phi'_{k-1}(\beta)$.

Regardless of how $\beta_{k-1,0}$ is chosen, the function that represents the line through $(\beta_{k-1,0}, \phi'_{k-1}(\beta_{k-1,0}))$ and $(\beta_{k-1}, \phi'_{k-1}(\beta_{k-1}))$ is zero at the value of $\beta^*_{k-1}$ given by

$$\beta^*_{k-1} = \frac{\beta_{k-1}\phi'(\beta_{k-1,0}) - \beta_{k-1,0}\phi'(\beta_{k-1})}{\phi'(\beta_{k-1,0}) - \phi'(\beta_{k-1})}. \quad \ldots\ldots\ldots \quad (31)$$

As the procedure only gives an approximation of the zero of $\phi'_{k-1}(\beta)$, we do not allow the resulting initial approximation ($\beta_{k,2}$) for $\beta_k$ to be too different from $\beta_{k-1}$. To ensure this is the case, we define the step size modification factor by

$$\eta_{k-1} = \frac{\beta^*_{k-1}}{\beta_{k-1}}; \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (32)$$

if the calculated value of $\eta_{k-1}$ is greater than 2, we set $\eta_{k-1} = 2$ and if the calculated value is less than $1/2$, we set $\eta_{k-1} = 1/2$. Finally, we define our initial guess for $\beta_k$ by

$$\beta_{k,2} = \beta_{k-1}\eta_{k-1}. \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (33)$$

### Bayesian Formulation

We use a Bayesian framework (Tarantola, 1987; Li et al., 2003; Zhang and Reynolds, 2002; Zhang et al., 2003) for our history matching problem. In this context, $d_{obs}$ is an $N_d$-dimensional column vector that contains all observed production data and $C_D$ denotes

the covariance matrix for measurement errors. In the example application presented here, measurement errors are modelled as independent random variables with zero means and prescribed variances so $C_D$ is a diagonal matrix. We let $d = g(m)$ represent the predicted production data for a given model $m$. In history matching, $g(m)$ represents running the simulator with $m$ as input to obtain data $d = g(m)$ corresponding to the observed production data, $d_{\text{obs}}$.

The $N_m$-dimensional column vector $m$ denotes the model or vector of model parameters to be estimated or stochastically simulated. For the example application considered in this work, $m$ consists of gridblock log-permeabilities (horizontal and vertical) and porosities. The vector $m_{\text{prior}}$ denotes the vector of prior means. Assuming the prior probability density function (pdf) for $m$ is multivariate Gaussian with mean $m_{\text{prior}}$ and covariance matrix $C_M$, the a posteriori pdf for $m$ conditional to the observed production data, $d_{\text{obs}}$, is given by

$$f(m|d_{\text{obs}}) = c\exp[-O(m)], \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (34)$$

where $c$ is the normalizing constant, and $O(m)$ is given by

$$O(m) = \frac{1}{2}(g(m)-d_{\text{obs}})^T C_D^{-1}(g(m)-d_{\text{obs}}) + \frac{1}{2}(m-m_{\text{prior}})^T C_M^{-1}(m-m_{\text{prior}}) \quad \ldots\ldots\ldots \quad (35)$$

The maximum a posteriori (MAP) estimate of $m$ is the model that minimizes $O(m)$. As our objective here is to illustrate the effectiveness of the modified implementation of the LBFGS algorithm, we consider only the generation of the MAP estimate. However, for evaluation of the uncertainty in performance predictions, it would be preferable to generate predictions with multiple realizations (samples) of the a posteriori pdf. The implementation of the LBFGS algorithm given here is equally effective for generating multiple realizations (Gao and Reynolds, 2004).

Based on results presented in Tarantola (1987), the minimum of $2O(m)$ follows a chi-squared distribution with expectation $N_d$ and variance $2N_d$. Assuming that the MAP estimate should always give a value of the objective function within five standard deviations of the expected value, the MAP estimated (denoted by $m_\infty$) generated is acceptable only if the objective function evaluated at $m_\infty$ satisfies

$$N_d - 5\sqrt{2N_d} \le 2O(m_\infty) \le N_d + 5\sqrt{2N_d}. \quad \ldots\ldots\ldots \quad (36)$$

If $O(m_\infty)$ does not satisfy this result, then one should expect that something is wrong, e.g., LBFGS has converged to a local minimum which gives an unacceptable match of data, the prior model is inappropriate or the covariance matrix $C_D$, which must account for both modelling and measurement errors, underestimates these errors. Strictly speaking, Eq. 36 is based on the assumption that the predicted data is linearly related to the model (Tarantola, 1987). However, computational results suggest that the condition specified by Eq. 36 is still valid even when the relation between data and the model is nonlinear.

#### Rescaling of Model Parameters

For the LBFGS algorithm, computational experiments indicate that the convergence properties are often improved by a simple rescaling of the model parameters. Here, the original model parameters are shifted by their prior means $m_{\text{prior},i}$ and rescaled with their standard deviations, the $\sigma_{m,i}$'s. Both $m_{\text{prior},i}$ and $\sigma_{m,i}$ are determined from the specified prior pdf of model parameters. In fact, $\sigma_{m,i}$ is the square root of the $i$th diagonal element of matrix $C_M$. We define the new rescaled variables, $x_i$, $i = 1,2,...,N_m$, by

$$x_i = \frac{m_i - m_{\text{prior},i}}{\sigma_{m,i}}, \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (37)$$

The covariance matrix for the random vector $x$ is identical to the correlation matrix for $m$. It is possible to generate examples where the condition number of the correlation matrix for $m$ is greater than the condition number of $C_M$. Thus, unless $C_M$ is diagonal, one can not even guarantee that our simple rescaling procedure improves the scaling of the part of the objective function corresponding to the prior model. For the model mismatch term arising from the prior model, the optimal rescaling would be obtained by replacing $m$ by

$$x = C_M^{-1/2}(m - m_{\text{prior}}), \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (38)$$

but for large scale problems it is not feasible to compute the square root of $C_M^{-1}$; thus, we have opted to use the simple rescaling of Eq. 37. For the examples we have considered, this simple rescaling often improves the rate of convergence of the LBFGS algorithm.

#### Controlling Overshooting and Undershooting

In some history matching examples, one finds that an optimization algorithm exhibits undershooting or overshooting, i.e. converges to a model with unreasonably small or large values of some of the model parameters, e.g., gridblock permeabilities and/or porosities (Wu et al., 1999; Li et al., 2003). This problem often occurs when the initial guess for the model gives predicted data far from the observed data, i.e., results in a large data mismatch. The problem is most severe when the initial data mismatches are unbalanced, i.e., at one or two wells, the initial production data mismatches are much larger than those in the other wells. In this case, the large data mismatches control the adjustments to the model during the early iterations of the optimization algorithm and often result in overshooting or undershooting problems in the first few iterations. In this situation, damping the production data with large initial mismatches is an effective method to control the problem of overshooting and undershooting. Applying a constrained optimization algorithm provides an alternate method to keep model parameters within upper and lower bounds, but to fully control undershooting and overshooting with constrained optimization may require starting with very tight bounds on the model parameters at early iterations.

**Damping the Data Mismatch Term.** The data mismatch part of the objective function can be reweighted using a damping matrix $\Lambda$. $\Lambda$ is a diagonal matrix with its diagonal element given by $\lambda_i > 1$. The objective function with a damped data mismatch term is given by

$$O_{\text{damp}}(m) = \frac{1}{2}(g(m)-d_{\text{obs}})^T \tilde{C}_D^{-1}(g(m)-d_{\text{obs}}) + \frac{1}{2}(m-m_{\text{prior}})^T C_M^{-1}(m-m_{\text{prior}}). \quad \ldots\ldots \quad (39)$$

In the preceding equation, the matrix $\tilde{C}_D$ defined by

$$\tilde{C}_D = \Lambda C_D \Lambda, \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (40)$$

represents the modified data covariance matrix. Eq. 39 can be obtained from Eq. 35 by replacing the original data covariance matrix by $\tilde{C}_D$. With $\sigma_{d,i}^2$ denoting the variance of the $i$th measurement error, i.e., the $i$th entry of $C_D$, the $i$th entry of $\tilde{C}_D$ is $\sigma_{d,i}^2\lambda_i^2$. Thus if $\lambda_i > 1$, the standard deviation of the $i$th measurement error has been increased by the factor $\lambda_i$, and consequently, the relative weighting of the data mismatch term has been decreased in the objective function of Eq. 39 and the relative weighting of the model mismatch term has been increased. The effect is that, in the minimization process, the changes in the model will be smaller and the tendency to obtain abnormally high or low values of model parameters will be decreased.

We can specify different damping factors for different types of data. We can also specify a different damping factor for each datum. One way to specify the damping factor for $i$th datum, $\lambda_i$, is based on the difference between the observed value, $d_{\text{obs},i}$, and the corresponding value of the datum predicted with the initial model. If the difference is larger than $\kappa$ standard deviations of the measurement error for this datum, we applying damping. The damping factors are calculated by

$$\lambda_i = \max\left[1, \left|\frac{g_i(m_0) - d_{obs,i}}{\kappa\sigma_i}\right|\right], \quad \ldots\ldots\ldots\ldots\ldots \quad (41)$$

where $g_i(m_0)$ is the $i$th predicted datum generated using the initial model $m_0$, and $\kappa > 1$. This is the procedure we use.

As we wish to use the correct $C_D$ in constructing the final MAP estimate, damping is applied until the original normalized objective function $O_N(m) = 2O(m)/N_d$ decreases to a value less than $\kappa$. This is the procedure we use in the history matching example presented later, but a multi-step procedure can also be used. In a multi-step procedure, one can adjust the value of $\lambda_i$ every 5 to 10 iterations where at each readjustment, one applies Eq. 41 with $g_i(m_0)$ replaced by $g_i(m_k)$ where $m_k$ is the latest iterate. In this process, all damping is removed once we obtain a normalized objective function smaller than $\kappa$.

Definition of $\kappa$ is somewhat ad hoc. In the example presented here $\kappa = 3$ as this value tends to work well for synthetic history matching examples. However, if $\kappa = 3$ results in apparent undershooting/overshooting at the end of the first iteration, $\kappa$ would be decreased. We have never encountered this situation. If $\kappa = 3$ results in a negligible change in the objective function, we have over damped and $\kappa$ should be increased.

**Applying Constraints.** If the overshooting/undershooting is caused by large initial data mismatch, applying damping can effectively control it. However, the overshooting/undershooting can occasionally occur because of the inherent ill-conditioning of the inverse problem even when the initial data mismatch is not very large. For such cases, a constrained optimization algorithm can be used to control undershooting and overshooting. For cases where undershooting and overshooting is due to a large initial data mismatch, the constrained optimization algorithm used here provides an alternative to the damping procedure and yields comparable results.

Let $m_l$ and $m_u$, respectively, denote vectors with $j$th entry given by $m_{l,j}$ and $m_{u,j}$ which respectively represent the lower and upper bounds for the $j$th model parameter. One method for transforming the constrained optimization problem into an unconstrained optimization problem is to find a suitable transformation that can map the upper bound to $\infty$ and the lower bound to $-\infty$. Through this transformation, the boundaries are removed. In general, the transformation needs to be invertible (one-to-one) which is normally achieved by using a monotonic transformation. One choice for such a transformation is given by the logarithmic transformation discussed below.

We first apply a linear transformation to rescale the model parameters by their upper and lower bounds. Defining

$$m_{m,j} = \frac{1}{2}(m_{u,j} + m_{l,j}), \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (42)$$

and

$$m_{r,j} = \frac{1}{2}(m_{u,j} - m_{l,j}), \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (43)$$

the rescaled model parameter are defined by

$$x_j = \frac{m_j - m_{m,j}}{m_{r,j}}, \text{ for } j = 1,2,\ldots N_m. \quad \ldots\ldots\ldots\ldots\ldots \quad (44)$$

When $m_j \to m_{u,j}$, $x_j \to 1$, and when $m_j \to m_{l,j}$, $x_j \to -1$. The final transformed parameters are defined by the following log-transformation

$$s_j = \ln\left(\frac{x_j + 1}{1 - x_j}\right), \text{ for } j = 1,2,\ldots N_m. \quad \ldots\ldots\ldots\ldots\ldots \quad (45)$$

The log-transformation maps $x_j = 1$ to $s_j = \infty$, and $x_j = -1$ to $s_j = -\infty$. From Eq. 45, we can show that the relationship between the original variable $m_j$ and the transformed variable $s_j$ is given by

$$m_j = m_{m,j} + m_{r,j}\left(\frac{\exp(s_j) - 1}{\exp(s_j) + 1}\right), \quad \ldots\ldots\ldots\ldots\ldots \quad (46)$$

and

$$s_j = \ln\left(\frac{m_j - m_{l,j}}{m_{u,j} - m_j}\right). \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (47)$$

If $m_j \to m_{u,j}$, then $s_j \to \infty$ and if $m_j \to m_{l,j}$, then $s_j \to -\infty$. Thus, the boundaries due to the constraints are removed. The relationship between the derivatives of any function $f(m)$ with respect to $m_j$ and its derivative with respect to $s_j$ can be obtained from the chain rule. From Eq. 46, it follows that

$$\frac{dm_j}{ds_j} = \frac{(m_{u,j} - m_j)(m_j - m_{l,j})}{m_{u,j} - m_{l,j}}, \quad \ldots\ldots\ldots\ldots\ldots \quad (48)$$

and then the chain rule gives

$$\frac{\partial f}{\partial s_j} = \frac{\partial f}{\partial m_j}\frac{dm_j}{ds_j} = \frac{(m_{u,j} - m_j)(m_j - m_{l,j})}{m_{u,j} - m_{l,j}}\frac{\partial f}{\partial m_j}. \quad (49)$$

The second derivative of $f(m)$ with respect to $s_i$ and $s_j$ for $i \neq j$ is given by

$$\frac{\partial^2 f}{\partial s_j \partial s_i} = \frac{dm_i}{ds_i}\frac{\partial^2 f}{\partial m_j \partial m_i}\frac{dm_j}{ds_j}. \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (50)$$

The second derivative of $f(m)$ with respect to $s_j$ is given by

$$\frac{\partial^2 f}{\partial s_j^2} = \frac{\partial^2 f}{\partial m_j^2}\left[\frac{dm_j}{ds_j}\right]^2 + \frac{\partial f}{\partial m_j}\frac{dm_j}{ds_j}\frac{d}{dm_j}\left[\frac{dm_j}{ds_j}\right], \quad \ldots\ldots\ldots \quad (51)$$

where

$$\frac{d}{dm_j}\left[\frac{dm_j}{ds_j}\right] = \frac{m_{u,j} + m_{l.j} - 2m_j}{m_{u,j} - m_{l,j}}. \quad \ldots\ldots\ldots\ldots\ldots \quad (52)$$

The model mismatch part of the objective function is given by

$$O_m(m) = \frac{1}{2}[m - m_{\text{prior}}]^T C_M^{-1}[m - m_{\text{prior}}] = O_s(s), \quad \ldots \quad (53)$$

where the explicit equation for $O_s$ as a function of $s$ can be obtained by replacing each component of $m$ by the right side of Eq. 46. The gradient of the model mismatch part of the objective function $O_m(m)$ with respect to the original model vector $m$ is given by

$$\nabla_m O_m = C_M^{-1}[m - m_{\text{prior}}]. \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (54)$$

The gradient of the model mismatch part of the objective function $O_s(s)$ with respect to $s$ can be obtained from the chain rule and is given by

$$\nabla_s O_s = \Lambda_1 \nabla_m O_m(m) = \Lambda_1 C_M^{-1}[m - m_{\text{prior}}], \quad \ldots\ldots\ldots \quad (55)$$

where $\Lambda_1$ is a diagonal matrix with $j$th diagonal entry given by

$$\lambda_{1,j} = \frac{dm_j}{ds_j}. \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (56)$$

The Hessian matrix of $O_m(m)$ with respect to $m$ is given by

$$H_m = \nabla_m[\nabla_m O_m(m)]^T = C_M^{-1}. \quad \ldots\ldots\ldots\ldots\ldots \quad (57)$$

Because of this equation, we use $C_M$ as the initial approximation of the inverse Hessian if optimization is done in terms of the original model parameters. When the log-transformation is applied, however, we do optimization in terms of the transformed variables. Thus, we need to find the Hessian based on the model mismatch part in terms of $s$ evaluated at the initial guess. The Hessian matrix of $O_s(s)$ with respect to $s$ is given by

$$H_s = \nabla_s[\nabla_s O_s(s)]^T, \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (58)$$

and also can be obtained by the chain rule using Eqs. 50, 51 and 57. It follows that

$$H_s = \Lambda_1 C_M^{-1}\Lambda_1 + \Lambda_2, \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (59)$$

where $\Lambda_2$ is a diagonal matrix with $j$th diagonal entry given by

$$\lambda_{2,j} = \frac{dm_j}{ds_j}\frac{d}{dm_j}\left[\frac{dm_j}{ds_j}\right]\frac{\partial O_m}{\partial m_j}. \quad \ldots\ldots\ldots\ldots\ldots \quad (60)$$

As optimization is done in terms of $s$, we wish to use $[H_s]^{-1}$ evaluated at the initial guess for $m$ as the initial guess of the approximate Hessian inverse in the LBFGS algorithm. However, in general, $[H_s]^{-1}$ is dense and cannot be easily computed for large scale problems. Note, however, that the diagonal matrix $\Lambda_2$ is a null matrix if $m = m_{\text{prior}}$ or if $m$ equals the average of its lower and upper bounds. Then the initial guess for the Hessian inverse for the LBFGS algorithm can be chosen as

$$H_0^{-1} = \Lambda_{1,\text{mprior}}^{-1} C_M \Lambda_{1,\text{mprior}}^{-1}, \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (61)$$

or

$$H_0^{-1} = \Lambda_{1,\text{ave}}^{-1} C_M \Lambda_{1,\text{ave}}^{-1}. \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (62)$$

The subscript mprior denotes evaluation at $m = m_{\text{prior}}$ and the subscript ave denotes evaluation at the average of the upper and lower bounds.

### History Matching of Full Scale PUNQS3

**Reservoir Model Description.** The PUNQS3 reservoir example is constructed based on a real field (Floris et al., 2001). The model consists of five layers. The simulation model contains $19 \times 28 \times 5$ grid blocks, of which 1761 blocks are active. The field is bounded to the east and south by a fault, and links to the north and west to a fairly strong aquifer. A small gas cap is located in the center of the dome shaped structure. The field initially contains 6 production wells located around the gas oil contact. We introduce 66 water injection wells around the WOC line to simulate the aquifer. The grid block size in $x$- direction and $y$- direction is uniform, $\triangle x = \triangle y = 590.55$ (ft).

There are six producers, PROD1(10,22,4/5), PROD4(9,17,4/5), PROD5(17,11,3/4), PROD11(11,24,3/4), PROD12(15,12,4/5) and PROD15(17,22,4). Here, the numbers in parentheses refer to the gridblock indices. For example, PROD1(10,22,4/5) means that production well one is located in the gridblock centered at $(x_{10}, y_{22})$ (tenth gridblock in the $x$- direction, twenty second gridblock in the $y$- direction), and perforated in the fourth and fifth gridblock in the $z$- direction.

The porosities in each layer are normally distributed. The horizontal and vertical permeabilities are log-normally distributed. The porosity, $\ln(k)$ and $\ln(k_z)$ within each layer are correlated to each other, but there is no correlation between properties in different layers. The correlation coefficients between porosity, $\ln(k)$ and $\ln(k_z)$ in each layer are all specified as 0.8.

The observed data are obtained by adding noise (measurement error) to the "true" production data generated by running the reservoir simulator with the true model. In generating noise, the standard deviations of measurement error for bottom hole pressure, GOR and WOR, respectively, were specified as $\sigma_p = 43.509$ (psi), $\sigma_{GOR,i} = 0.1 GOR_{\text{true},i}$ (scf/STB), and $\sigma_{WOR} = 0.01$ (STB/STB). Measurement errors were assumed to be independent Gaussian random variables with mean zero. By adding noise to the true data, we obtain the observed or noisy production data which we history match. In all cases, we match production data obtained during the first 2920 days (8 years) of production. After history matching these production data, we generate predictions for another 8.5 years of production based on the estimated model obtained by history matching. In this example, there are a total of 2604 observed data in $d_{\text{obs}}$, 900 bottom hole pressure data, 852 GOR data and 852 WOR data. The correct geostatistical model (variogram) for each layer is used to define the prior covariance matrix.

**Methods Considered.** We will discuss four different implementations of our history matching procedure. With all implementations, optimization is done with the LBFGS algorithm and the improved line search procedure developed in this paper is applied. Rescaling refers to changing variables according to Eq. 37 prior to applying the LBFGS algorithm. The four procedures we consider are (implementation 1) rescaling without applying damping or constraints; (implementation 2) rescaling with damping but no constraints; (implementation 3) damping without applying constraints or rescaling, (implementation 4) no damping with constraints. Because the constrained algorithm is based on the log-transformation which automatically does a rescaling of variables, we do not apply Eq. 37 if the constrained optimization procedure is used.

When damping is applied, we first damp the production data using the damping factors determined by Eq. 41 with $\kappa = 3$. This optimization step is terminated when the normalized objective function calculated without damping becomes smaller than 3 which takes no more than five to ten iterations. Then we use the model obtained with damped production data as the initial model, and apply the LBFGS algorithm with no further damping.

For the PUNQ example, the standard deviations of porosity, $\ln(k)$ and $\ln(k_z)$ are quite large and thus do not prohibit extremely large changes in model parameters. To avoid extremely large or extremely small values of model parameters, at early iterations of the constrained optimization algorithm, we require the updated value of each component of $m$ to be within 1.5 standard deviations of its prior mean, i.e., the upper and lower bounds for the $i$th component of $m$ are defined by

$$m_{u,i} = m_{\text{prior},i} + 1.5\sigma_{m,i}, \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (63)$$

$$m_{l,i} = m_{\text{prior},i} - 1.5\sigma_{m,i}, \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (64)$$

where $m_{\text{prior},i}$ is the prior mean of $m_i$, and $\sigma_{m,i}$ is the standard deviation of $m_i$. If the lower bound of porosity given by Eq. 64 is smaller than 0.005, it is set to 0.005. When the normalized objective function is smaller than 3, we restart the history matching program with the unconstrained optimization algorithm. The model parameters are automatically rescaled with the log-transformation. When we restart with an unconstrained optimization algorithm, we rescale the variables according to Eq. 37 prior to applying the LBFGS algorithm.

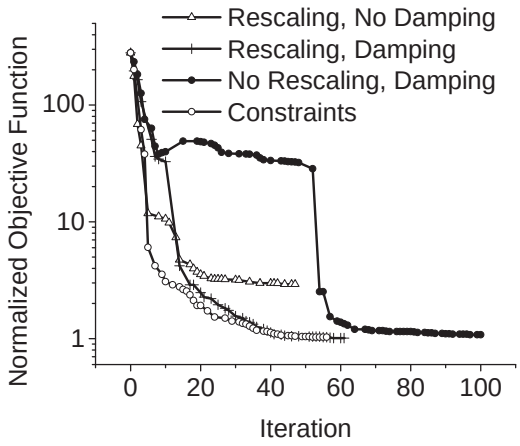



**Fig. 4—Normalized objective function for unconstrained optimization.**

**Convergence Behavior. Fig. 4** illustrates the behavior of the normalized objective function ( $2O(m)/N_d$) for the four different implementations of the algorithm. In this figure, the curve through open triangles represents the normalized objective function for implementation 1 (rescaling, no damping, no constraints); the curve through pluses represents the behavior of the objective function for implementation 2 (rescaling with damping but no constraints); the curve through solid circles represents implementation 3 (no rescaling with damping but no constraints); and the curve with open circles is for implementation 4 (the constrained algorithm with no damping).

For implementation 1, the normalized objective function converged to a value of 3, which is larger than the expected value of 1

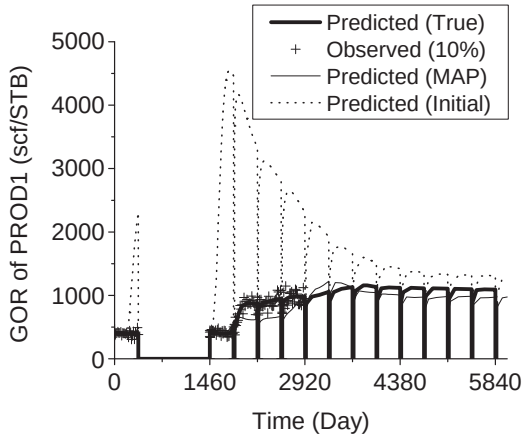



**Fig. 5—GOR data match, PROD1, rescaling with no damping and no constraints.**

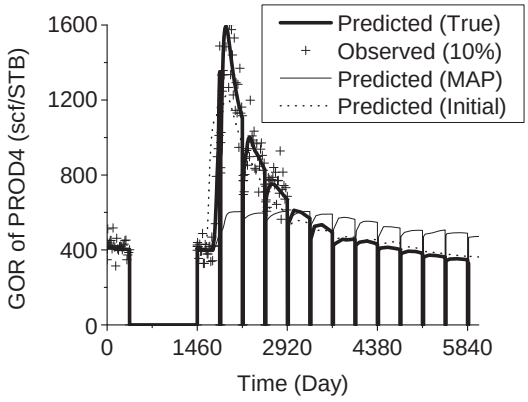



**Fig. 6—GOR data match, PROD4, rescaling with no damping and no constraints.**

and does not satisfy the criterion of Eq. 36, i.e., the resulting model is not acceptable as it does not give a sufficiently good match of production data. Although results not shown indicate that the convergence properties of the LBFGS algorithm are improved by rescaling, rescaling by itself does not result in optimum convergence performance. By comparing implementations 2 and 3, however, we see that rescaling is of value. When damping without rescaling was applied, we reached the maximum allowable value of 100 iterations without satisfying the convergence criteria, whereas, with rescaling and damping, we lowered the objective function to a slightly lower value and LBFGS converged in 60 iterations. For this example, implementations 2, 3 and 4 all resulted in an objective function close to 1 at termination of the algorithm, i.e., the criterion of Eq. 36 is satisfied. In terms of the number of iterations, the constrained implementation and LBFGS with rescaling and damping performed similarly in terms of the rate of convergence. With each of these methods, convergence is obtained in about 60 iterations where the convergence criteria is based on requiring that the relative change in the objective function be less than $10^{-4}$ and the relative change in the model (based on the infinity norm) over an iteration be less than $10^{-2}$. As suggested by the results of Fig. 4, if $10^{-4}$ were increased to $10^{-3}$, convergence would be obtained in 50 or fewer iterations and the models obtained at convergence would not be noticeably different.

As a good initial guess of the step size is obtained by applying the procedure introduced earlier in this paper, for about two thirds of the LBFGS iterations, the initial step size for the line search algorithm satisfies the Wolfe conditions and is accepted. In such cases, only one simulation run and one adjoint solution are required to complete the LBFGS iteration. For most of the other LBFGS iterations, only one additional iteration of the line search algorithm is used and in such cases, the LBFGS iteration requires two forward simulation runs and two adjoint solutions. For the example considered here, on average, an LBFGS iteration requires fewer than 1.5 simulation runs and 1.5 associated adjoint solutions. Based on this result, if 60 LBFGS iterations are done, fewer than 90 forward simulation runs and 90 adjoint solutions are required. The time required for an adjoint solution is roughly equal to the $1/N_n$ times the time required for the forward simulation run where the $N_n$ is the average number of Newton-Raphson iterations per time step of the forward simulation run. For the example considered here, the computational effort of the simulation and adjoint runs used to obtain convergence with LBFGS is roughly equivalent to 120 reservoir simulation runs.

**Data Match. Fig. 5** shows the GOR data match for Well PROD1 obtained with implementation 1 (rescaling, but no damping or constraints) of the LBFGS algorithm. In this figure, the thick solid curve is the true data generated with the true model; the plus marks are the observed data; the thin solid curve represents the predicted data generated with the MAP estimate; and the dotted curve represents the GOR predicted with the initial model where all model parameters were set equal to their prior means. The same basic notation is also used in **Figs. 6 through 10.** Well PROD1 is located at (10,22,4/5). The initial GOR data mismatch in well PROD1 is much larger than the GOR mismatch at the other wells. The maximum value of the initial GOR data mismatch for PROD1 reaches 4000 (Scf/STB), which is 100 times larger than the standard deviation (about 40 Scf/STB) of the GOR measurement error. The GOR data mismatch term in this well dominates the other production data in the early iterations of the LBFGS algorithm and leads to a poor GOR data match at other wells. As shown in Fig. 6, the GOR data mismatch in well PROD4 after history matching is worse than the initial GOR mismatch. The large and unbalanced initial GOR data mismatch also results in changes in model parameters aimed primarily at reducing the GOR mismatch in PROD1. Specifically to decrease the GOR in well PROD1, the horizontal and vertical permeabilities around this well in the second layer are greatly decreased to form a barrier to flow from the gas cap, as illustrated by the results of Fig. 11(b) and Fig. 12(b). Because porosity is strongly correlated with the horizontal and vertical log-permeabilities, the porosity in the second layer is also decreased significantly; results for the estimated porosity fields are not shown here. It is important to note that Implementation 1, not only led to abnormally large decreases in properties in Layer 2, but also results in overshooting; as shown in Figs. 15(b) and 16(b), implementation 1 results in $\ln(k)$ values in excess of 12 and $\ln(k_z)$ in excess of 10 in Layer 4. The damping procedure introduced earlier can be used to eliminate the undershooting/overshooting.

Figs. 7 and 8 illustrate GOR data matches in well PROD1 and PROD4 obtained when rescaling and damping (implementation 2) are used in the LBFGS optimization procedure. We see that GOR data in both wells (PROD1 and PROD4) are well matched with the observed data and are actually quite close to the true data. We predicted the pressure and GOR performance for another 8.5 years of production. Figs. 7 and 8 show that the predicted GOR data for another 8.5 years of production are also close to the data predicted with the true model. Fig. 9 shows the match of bottom hole pressure data for well PROD1 based on implementation 2 of the LBFGS algorithm. Similar to the results shown in Fig. 9, the initial pressure mismatches in all six production wells are very large, over 1000 psi for much of the time period, but after history matching the production data, the pressure data misfit is greatly decreased. Fig. 10 shows the match of WOR data for well PROD11 based on implementation 2 of the LBFGS algorithm. A good match of WOR data from the first 8 years of production and a reasonable prediction of WOR performance for another 8.5 years of production were obtained with the MAP estimate obtained by applying the damping

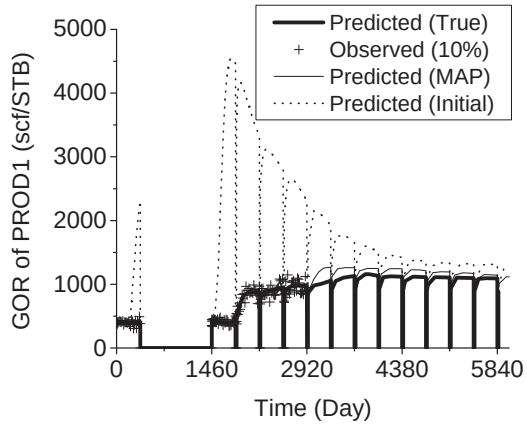



**Fig. 7—GOR data match, PROD1, damping, no constraints.**

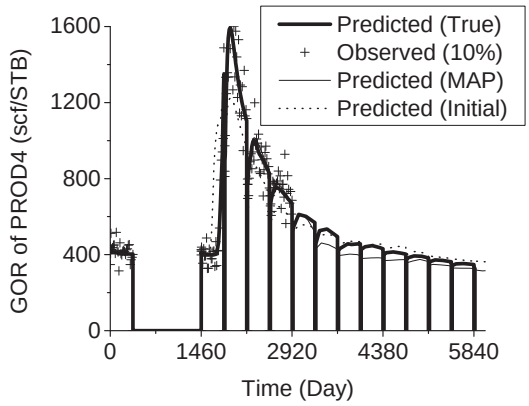



**Fig. 8—GOR data match, PROD4, damping, no constraints.**

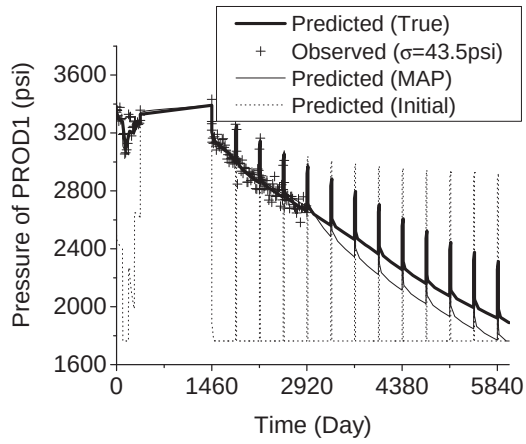



**Fig. 9—Pressure data match, PROD1, damping, no constraints.**

procedure. Although not shown here, GOR, WOR and pressure matches of similar quality were obtained using constrained optimization based on the log transformation (Implementation 4).

**Truth and MAP Estimates.** Figs. 11(a), 13(a) and 15(a), respectively, show the true horizontal log-permeability field in layers 2, 3 and 4. Figs. 12(a), 14(a) and 16(a) show the true $\ln(k_z)$ field in layers 2, 3 and 4. Figs. 11 through 16 also show the MAP estimates of $\ln(k)$ and $\ln(k_z)$ obtained with implementations 1, 2 and 4 of the LBFGS algorithm. Although not shown here, the MAP estimate obtained with implementation 3 (damping but no rescaling) was similar to the one obtained with implementation 2 (damping and rescaling). As noted previously, the estimated model obtained by history matching without damping (implementation 1) exhibits overshooting and undershooting.

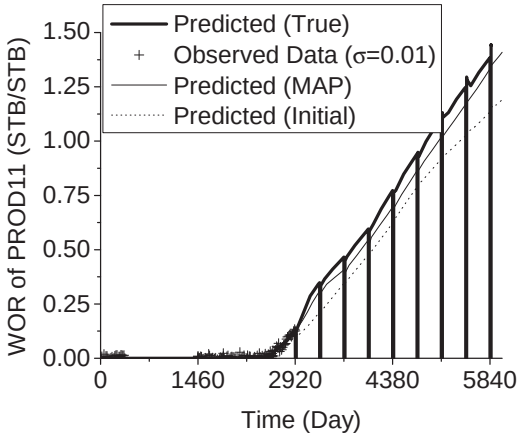



**Fig. 10—WOR data match, PROD11, damping, no constraints.**

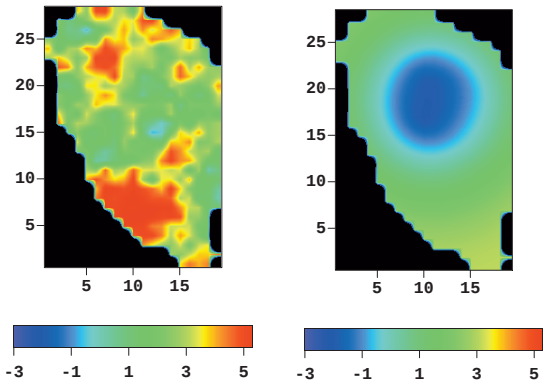
From the results of **Figs. 11–16,** we see that the undershooting and overshooting problems are eliminated by applying the damping procedure with rescaling (implementation 2) or the constrained optimization algorithm (implementation 4), i.e., we obtain reasonable permeability fields which show no evidence of undershooting or overshooting. Although not shown, a reasonable estimate of the porosity field was also obtained with implementations 2 and 4.

The MAP estimate is generated by minimizing the objective function given by Eq. 35, which is composed of two terms, a data mismatch term and a model mismatch term. Minimization of the objective function ideally yields a model that is "as close as possible" to its prior mean while honoring observed data. Because of this the the MAP estimate is expected to be much smoother than an realization generated from either the prior or posterior pdf, and hence smoother than the true model which in our example represents an unconditional realization from the prior model. In fact, in the case where the data is linearly related to the model, the MAP estimate is equal to the mean of the posterior pdf, $f(m|d_{\text{obs}})$, and could be approximated accurately by taking the average of a large number of samples (realizations) from $f(m|d_{\text{obs}})$ (Tarantola, 1987). Thus, the MAP is expected to much smoother than a typical realization and hence much smoother than the truth which was generated as an unconditional realization from the prior model. Note that this is the case when optimization is done using damping plus rescaling or constraints are applied; see Figs. 11 through 16.

## Conclusions

The LBFGS algorithm is a feasible and reliable optimization method for large scale history matching problems provided care is taken to control undershooting/overshooting. Overshooting and undershooting can be controlled by applying either a damping procedure which uses an artificially high variance for data measurement errors at early iterations or a constrained optimization algorithm which limits the changes in model parameters at early iterations. With these procedures, reasonable MAP estimates are obtained and an acceptable match of production data is obtained based on the criterion of Eq. 36.

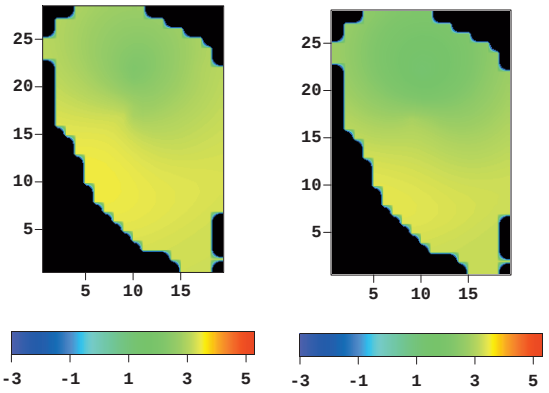
It is important that the line search algorithm is sufficiently accurate so that the strong Wolfe conditions are satisfied at each iteration




(a) True




(b) Implementation 1

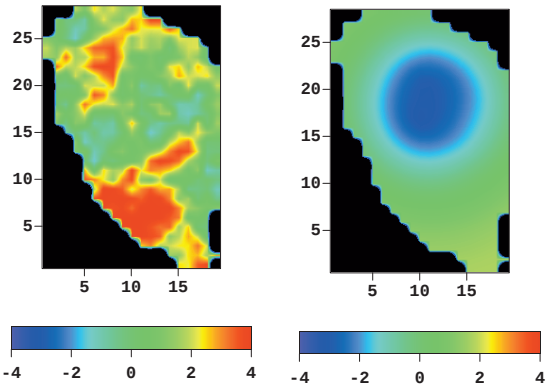



(c) Implementation 2




(d) Implementation 4

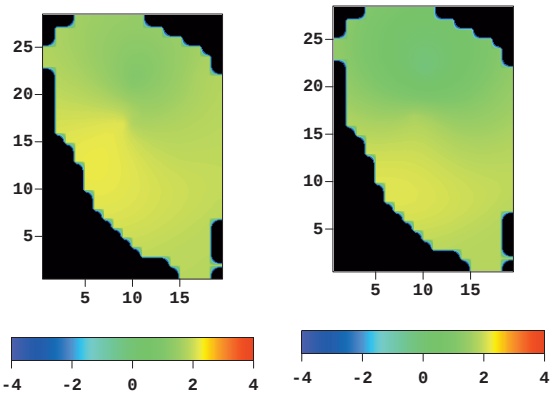
**Fig. 11—True and MAP estimates of $\ln(k)$ in layer 2, implementations 1, 2 and 4.**




(a) True




(b) Implementation 1




(c) Implementation 2




(d) Implementation 4

**Fig. 12—True and MAP estimates of $\ln(k_z)$ in layer 2, implementations 1, 2 and 4.**

of the LBFGS algorithm. Failure to do so can cause the approximate inverse Hessian matrix to fail to be positive definite and lead to a restart of the LBFGS algorithm.

The new line search procedure developed here is reliable for guaranteeing that the strong Wolfe conditions are satisfied. With the method proposed for generating the initial guess for an appropriate step size, the initial step size satisfies the strong Wolfe conditions at the majority of the LBFGS iterations.

## Nomenclature

$C$ = covariance matrix
$d_{\text{obs}}$ = vector of observed data
$f$ = general objective function
$GOR$ = gas oil ratio, scf/STB
$H$ = Hessian matrix
$k$ = permeability, md
$m$ = vector of model parameters
$O$ = objective function
$p$ = search direction vector, or pressure, psi
$WOR$ = water oil ratio, STB/STB
$\alpha$ = step size
$\beta$ = normalized step size
$\gamma$ = scaling factor
$\lambda$ = damping factor
$\sigma$ = standard deviation of a random variable
$\phi$ = porosity or line search objective function

## Subscripts

$k$ = iteration index
$l$ = lower bound
obs = observed
prior = = prior mean
true = true value
$u$ = upper bound


## Acknowledgments

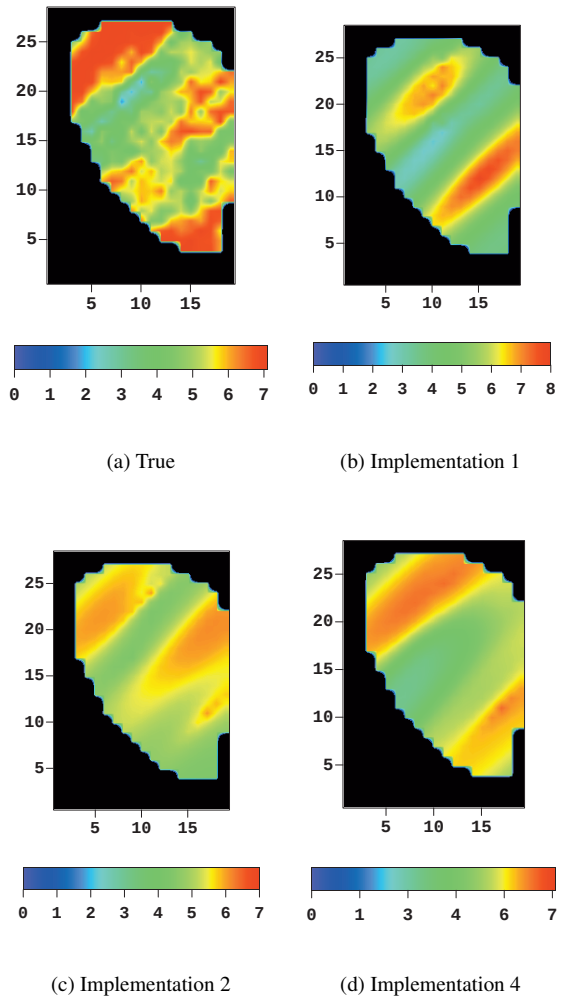
This work was supported by the member companies of TUPREP and the U.S. Department of Energy under Award No. DE-FC26-00BC15309. However, any opinions, findings, conclusions or recommendations herein are those of the authors and do not necessarily reflect the views of the DOE.


## References


Anterion, F., Eymard, R., and Karcher, B.:"Use of Parameter Gradients for Reservoir History Matching," paper SPE 18433 presented at the 1989 SPE Symposium on Reservoir Simulation, Houston, 6-8 February.

Chavent, G., Dupuy, M., and Lemonnier, P.: "History Matching by Use of Optimal Theory," *SPEJ* (1975) **15**, No. 1, 74; *Trans.*, AIME, **259**.

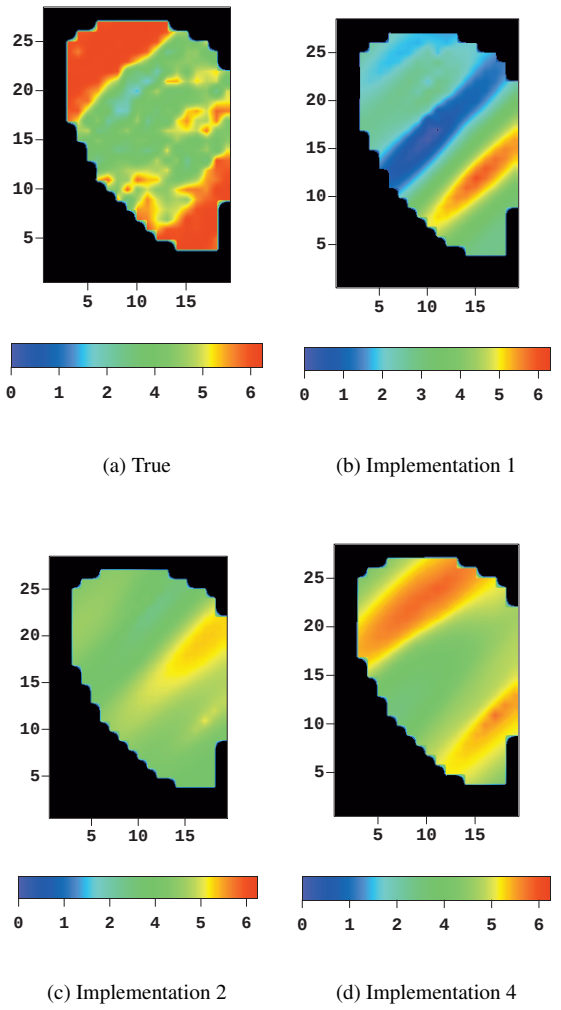
Chen, W.H., Gavalas, G.R., Seinfeld, J.H., and Wasserman, M.L.: "A New Algorithm for Automatic History Matching," *SPEJ* (1974) **14**, No. 6, 593; *Trans.*, AIME, **257**.



(a) True (b) Implementation 1

(c) Implementation 2 (d) Implementation 4

**Fig. 13—True and MAP estimates of $\ln(k)$ in layer 3, implementations 1, 2 and 4.**



(a) True (b) Implementation 1

(c) Implementation 2 (d) Implementation 4

**Fig. 14—True and MAP estimates of $\ln(k_z)$ in layer 3, implementations 1, 2 and 4.**


Dennis, J.E. and Schnabel, R.B.: *Numerical Methods for Unconstrained Optimization and Nonlinear Equations,*" 1996 SIAM, Philadelphia.

Floris, F.J.T., Bush, M.D., Cuypers, M., Roggero, F., and Syversveen, A.-R.: "Methods for quantifying the uncertainty of production forecasts: A comparative study," *Petroleum Geoscience* (2001) 7 (Supp.), 87.

Gao, G. and Reynolds, A.C.: "An improved implementation of the LBFGS algorithm for automatic history matching with application to the PUNQ data set," TUPREP Research Report 21 (2004) 14.

Li, R., Reynolds, A.C., and Oliver, D.S.: "History Matching of Three-Phase Flow Production Data," *SPEJ* (2003) **8**, No. 4.

Nocedal, J.: "Updating quasi-Newton matrices with limited storage," *Math. Comp.* (1980) **35**, No. 151, 773.

Nocedal, J. and Wright, S.J.: *Numerical Optimization*, Springer, New York City (1999).

Tan, T.B. and Kalogerakis, N.: "A Fully Implicit, Three-Dimensional, Three-Phase Simulator With Automatic History-Matching Capability," paper SPE 21205 presented at the 1991 SPE Symposium on Reservoir Simulation, Anaheim, California, 17-20 February

Tarantola, A.: *Inverse Problem Theory: Methods for Data Fitting and Model Parameter Estimation*, Elsevier, Amsterdam, The Netherlands (1987).

Wu, Z., Reynolds, A.C., and Oliver, D.S.: "Conditioning Geostatistical Models to Two-Phase Production Data," *SPEJ* (1999) **4**, No. 2, 142.

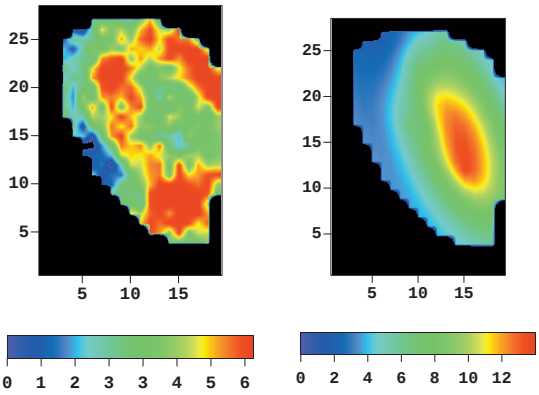
Zhang, F.: "Automatic History Matching of Production Data for Large Scale Problems," PhD dissertation, U. of Tulsa, Tulsa, Oklahoma (2002).

Zhang, F. and Reynolds, A.C.: "Optimization algorithms for automatic history matching of production data," *Proc.*, 2002 European Conference on the Mathematics of Oil Recovery, Frieberg, Germany, 3-6 September.

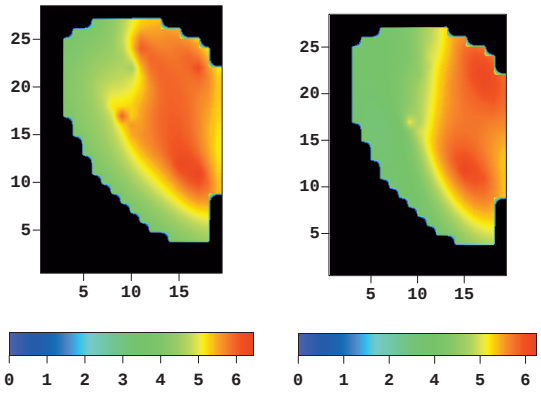
Zhang, F., Skjervheim, J.A., Reynolds, A.C., and Oliver, D.S.: "Automatic History Matching in a Bayesian Framework, Example Applications," *SPEREE* (2005) **8**, No. 3, 214.



**Guohua Gao** is a reservoir engineer at Chevron Corp. in Houston, TX. e-mail: GUOH@chevron.com. His current research interests include reservoir characterization, automatic history matching, and optimization. He holds a BS degree in mechanical engineering from Xian Jiaotong U., a MS degree in mechanical engineering from the U. of Petroleum, China, and a PhD degree in petroleum engineering from the U. of Tulsa. He worked 15 years as a professor at Xian Petroleum Inst. **Albert C. Reynolds** holds the McMan Chair in Petroleum Engineering at the U. of Tulsa, where he has been a faculty member since 1970. He has authored or co-authored over 100 technical papers and one book. His research interests include optimization, assessment of uncertainty, history matching and well testing. He holds a B.A. degree from the U. of New Hampshire, a MS

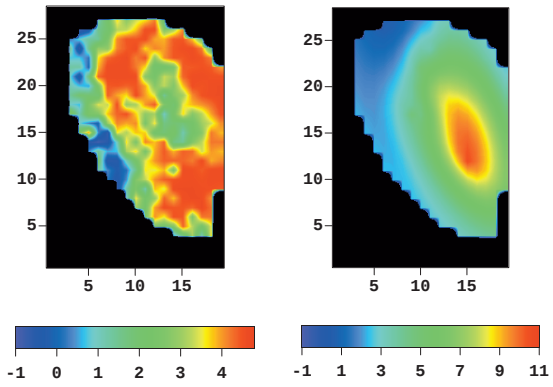



(a) True (b) Implementation 1

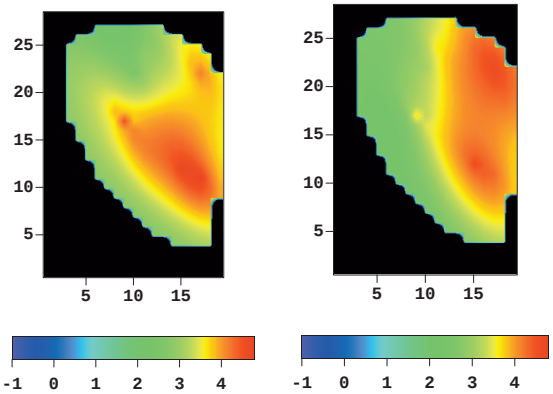



(c) Implementation 2 (d) Implementation 4

**Fig. 15—True and MAP estimates of $\ln(k)$ in layer 4, implementations 1, 2 and 4.**




(a) True (b) Implementation 1




(c) Implementation 2 (d) Implementation 4

**Fig. 16—True and MAP estimates of $\ln(k_z)$ in layer 4, implementations 1, 2 and 4.**

degree from Case Inst. of Tech., and a PhD degree from Case Western Reserve University, all in mathematics.