AGU PUBLICATIONS

Water Resources Research

RESEARCH ARTICLE
10.1002/2016WR019028

A computationally efficient parallel Levenberg-Marquardt algorithm for highly parameterized inverse model analyses

Youzuo Lin[1], Daniel O'Malley[2], and Velimir V. Vesselinov[2]

[1]Geophysics Group (EES-17), Earth and Environment Science Division, Los Alamos National Laboratory, Los Alamos, New Mexico, USA, [2]Computational Earth Sciences Group (EES-16), Earth and Environment Science Division, Los Alamos National Laboratory, Los Alamos, New Mexico, USA

Key Points:
- We generate a Krylov subspace and obtain a much smaller approximated problem by projecting the original problem down to the subspace
- We employ both coarse and fine-grained parallelism to our LM method to parallelize the implementation of the LM algorithm in two levels
- We employ a subspace recycling technique to take the advantage of the solution space previously generated

Correspondence to:
Y. Lin,
ylin@lanl.gov

Citation:
Lin, Y., D. O'Malley, and V. V. Vesselinov (2016), A computationally efficient parallel Levenberg-Marquardt algorithm for highly parameterized inverse model analyses, *Water Resour. Res.*, *52*, 6948–6977, doi:10.1002/2016WR019028.

Received 5 APR 2016
Accepted 17 AUG 2016
Accepted article online 19 AUG 2016
Published online 15 SEP 2016

Abstract Inverse modeling seeks model parameters given a set of observations. However, for practical problems because the number of measurements is often large and the model parameters are also numerous, conventional methods for inverse modeling can be computationally expensive. We have developed a new, computationally efficient parallel Levenberg-Marquardt method for solving inverse modeling problems with a highly parameterized model space. Levenberg-Marquardt methods require the solution of a linear system of equations which can be prohibitively expensive to compute for moderate to large-scale problems. Our novel method projects the original linear problem down to a Krylov subspace such that the dimensionality of the problem can be significantly reduced. Furthermore, we store the Krylov subspace computed when using the first damping parameter and recycle the subspace for the subsequent damping parameters. The efficiency of our new inverse modeling algorithm is significantly improved using these computational techniques. We apply this new inverse modeling method to invert for random transmissivity fields in 2-D and a random hydraulic conductivity field in 3-D. Our algorithm is fast enough to solve for the distributed model parameters (transmissivity) in the model domain. The algorithm is coded in Julia and implemented in the MADS computational framework (http://mads.lanl.gov). By comparing with Levenberg-Marquardt methods using standard linear inversion techniques such as QR or SVD methods, our Levenberg-Marquardt method yields a speed-up ratio on the order of $\sim 10^1$ to $\sim 10^2$ in a multicore computational environment. Therefore, our new inverse modeling method is a powerful tool for characterizing subsurface heterogeneity for moderate to large-scale problems.

1. Introduction

Inverse problems in groundwater modeling are usually under-determined and ill posed, because of the limited data coverage [*Sun*, 1994; *Carrera and Neuman*, 1986]. In recent years, with the help of regularization techniques [*Tarantola*, 2005; *Engl et al*., 1996], there is a trend to increase the number of model parameters [*Hunt et al*., 2007]. It has been discussed in many references that these highly parameterized models have great potential for characterizing subsurface heterogeneity [*Hunt et al*., 2007; *Tonkin and Doherty*, 2005].

As both the data measurements and model parameters increase in number, the size of the inversion problem becomes larger. Even though regularized inverse modeling methods yield hydrogeologically reasonable results [*Barajas-Solano et al*., 2014; *van den Doel and Ascher*, 2006; *Liu and Ball*, 1999], the major issue hindering the highly parameterized large inverse problems is the computational burden [*Doherty and Hunt*, 2010]. Specifically, the computational costs mostly come from the calculation of the Jacobian matrix and from solving the linear systems for the search direction within the Levenberg-Marquardt algorithm. Singular value decomposition (SVD) based linear solver is frequently used in solving inverse modeling; however, according to *Doherty and Hunt* [2010], employing the traditional Arnoldi SVD to solve a highly parameterized problem with a number of the model parameters greater than 2500 will require inordinate computational time.

Different methods have been proposed and developed to alleviate the problem of expensive computational costs. One of the directions is based on the subspace approximation. Several types of subspaces have been utilized including principle components subspace [*Kitanidis and Lee*, 2014; *Lee and Kitanidis*, 2014; *Tonkin and Doherty*, 2005], Krylov subspace [*Liu et al*., 2014; *Saibaba and Kitanidis*, 2012], subspace spanned by reduced-order model [*Liu et al*., 2014], and active subspace [*Constantine et al*., 2014]. Specifically, in the

work of *Tonkin and Doherty* [2005], a hybrid regularized inversion method is developed. A subspace is first generated using the principle orthogonal directions obtained from SVD. A set of super parameters are obtained in the subspace and Tikhonov regularization is then imposed on the super parameters. Because the dimension of the subspace is much smaller than the original parameter space, the computational cost in solving the model parameters is shown to be less expensive than that of solving the parameters in the original model space. In the work of *Kitanidis and Lee* [2014], the low-rank approximation technique is employed to reduce both the computation and memory requirements. The authors have further applied the same technique to hydraulic tomography in *Lee and Kitanidis* [2014] and yield very high accuracy with reduced computational cost. *Liu et al*. [2013] solve hydraulic tomography as a geostatistical inverse problem, which is similar to *Lee and Kitanidis* [2014]. Instead of using randomization to obtain a low-rank matrix, they employ reduced-order models (ROMs) to approximate the high-dimensional system with a low-dimensional problem to reduce the computational costs. In the follow-up work of *Liu et al*. [2014], they solve the linear system for search direction using the minimum residual method (MINRES) [*Golub and Van Loan*, 1996] to take the advantage of the symmetric positive definite (SPD) system matrix [*Paige and Saunders*, 1975]. *Saibaba and Kitanidis* [2012] incorporates the hierarchical matrices technique with a matrix-free Krylov subspace approach to improve the computational efficiency. In the work of *Constantine et al*. [2014], a subspace is constructed by exploring the directions of the strongest variability based on functional gradient. The author and his collaborators have further applied a similar idea to analyzing the sensitivity of a hydrologic model [*Jefferson et al*., 2015].

Another possibility for increasing computational performance is to employ parallelism in the inversion algorithms. Two different types of parallelism techniques can be categorized with respect to the level of the parallel operations. In the work of *Coleman and Plassmann* [1992], a parallelized row-oriented Householder QR method is utilized to solve for the search directions. Similarly, *Zhu and Zhang* [2013] also explore the parallelism of the QR factorization in a GPU environment. The commonality of the aforementioned methods is that they both employ the fine-grained parallelism, i.e., the parallelism is only applied to subprocedures. Hence, it can only be helpful to improve the computational efficiency locally. In order to further improve the overall computational efficiency, a coarse-grained parallelism is preferable. An example of the coarse-grained parallelism is the latest version of the software package, PEST++ Version 3 [*Welter et al*., 2015]. In this version, a parallel run manager, YAMR (Yet Another run ManageR), has been integrated. Another example of the coarse-grained parallelism is the "Parallel PEST" option used in the software packages of PEST [*Doherty and Hunt*, 2010].

The Levenberg-Marquardt (LM) algorithm has been used extensively in solving nonlinear inverse problems in groundwater modeling because of its robustness [*Finsterle and Kowalsky*, 2011; *Tonkin and Doherty*, 2005; *Nowak and Cirpka*, 2004; *Cooley*, 1985]. In this work, we also use it to solve our minimization problem. There are two portions of the LM algorithm that comprise most of the computational costs: the calculation of the Jacobian matrix and the computation of the linear system for search direction. To calculate the Jacobian matrix efficiently, we employ the adjoint-state method [*Custodio et al*., 2012; *Strang*, 2007] to obtain the Jacobian matrix instead of the finite difference approach. Therefore, our focus to improve computational efficiency is on the calculation for search direction.

We develop a computationally efficient Levenberg-Marquardt (LM) algorithm incorporating both a subspace approximation and parallelism to solve the highly parameterized inverse modeling problems. The novelties and major features of our work are the following. We employ a Krylov subspace recycling technique to the linear solver in the Levenberg-Marquardt algorithm. We first employ the Golub-Kahan-Lanczos (GKL) bidiagonalization technique to generate the Krylov subspace and show that the damping parameter is independent of the generated Krylov subspace. Our developed approach takes full advantage of this separability of the damping parameter from the solution space, and therefore can benefit more from the recycling technique compared to other existing work using an idea similar to recycling [*Doherty*, 2015; *Tonkin and Doherty*, 2005]. We also employ both coarse and fine-grained parallelism to our LM method such that the implementation of our LM algorithm can be parallelized in both global and local-level computational procedures to improve the overall efficiency.

Our subspace approximation method belongs to the Krylov subspace category. In particular, we first reformulate the search direction associated linear system as an equivalent least squares problem and then employ the GKL-bidiagonalization-based least squares with QR factorization (LSQR) method [*Paige and Saunders*, 1982a, 1982b] to solve this least squares problem for the search direction. LSQR is a Krylov-

subspace-based iterative linear solver. It projects the original problem down to a subspace and solves the projected problem, instead of obtaining the solution in the original parameter space. Usually, the projected problem can be a very good approximation to the original problem, but has much smaller dimensionality. Therefore, solving the projected problem can significantly reduce the computational costs.

The Levenberg-Marquardt algorithm linearly combines the search directions from steepest descent method and Newton-type methods. Correspondingly, the weight of the contribution from the steepest descent method is defined as a damping parameter. The damping parameter plays an important role in ensuring that the Jacobian matrix is positive definite and the search direction in the parameter space yields an optimal balance between first-order and second-order optimization steps and avoids the local minima. Therefore, selecting the damping parameter is a critical issue with the LM algorithm. This is where we employ our coarse-grained parallelism. There have been a number of research efforts on estimating an appropriate value of the damping parameter based on the previous and the present values of the objective function or Jacobian [*Naveen et al*., 2010; *Lampton*, 2009; *Araneda*, 2004]. *Naveen et al*. [2010] use the relative changes in the objective function to adjust the value of the damping parameter. The common point among all these methods is that the optimal damping parameter needs to be selected in a sophisticated sequential strategy. We, on the other hand, solve several linear systems for multiple damping parameters in each LM iteration simultaneously, and use the one that yields the best objective function value as the search direction. However, distributing the linear systems among different CPU cores can increase the communication overhead. Utilizing the fact that the subspace basis generated and spanned by the lower-dimension space is independent of the damping parameter, we recycle the Krylov subspace basis for all the new damping parameters. Therefore, the search directions can be all obtained efficiently and simultaneously.

Through our numerical cost analysis, we show that using our techniques, our new LM method improves the computational efficiency significantly. The computational complexity can be reduced by an order of the model dimension after the first damping parameter in solving the linear system at each LM iteration.

To evaluate the performance of our algorithm, we test our new Levenberg-Marquardt algorithm to solve for random transmissivities (in 2-D) and random conductivities (in 3-D) from steady state observations of hydraulic head. The hydraulic heads were "observed" from the solution of the steady state groundwater equation using a reference transmissivity/conductivity field at a number of observation points (monitoring wells). We implement our algorithm in Julia [*Bezanson et al*., 2014] as part of the MADS computational framework [*Vesselinov*, 2012]. By comparing with a Levenberg-Marquardt method using standard linear inversion techniques such as QR and SVD methods, our Levenberg-Marquardt method yields speed-up ratio in the order of $\sim 10^1$ to $\sim 10^2$ in a multicore computational environment.

In the following sections, we first briefly describe the fundamentals of inverse modeling and nonlinear optimization techniques including the Levenberg-Marquardt method (section 2). We develop a Krylov-subspace approximated Levenberg-Marquardt method (section 3), then we further incorporate parallelism and the subspace recycling technique to our method (section 4). We then apply our method to test problems and discuss the results (section 5). Finally, concluding remarks are presented in section 6.

2. Theory

In this section, we will review the fundamentals of hydrogeologic inverse modeling, and the classical Levenberg-Marquardt method.

2.1. Inverse Modeling

We consider a two-dimensional steady state groundwater flow equation on a square domain $[a, b]\times[c, d]$

$$\begin{aligned} \nabla \cdot (T\nabla h) &= g, \\ g(x, y) &= 0, \\ \left.\frac{\partial h}{\partial x}\right|_{a,y} &= \left.\frac{\partial h}{\partial x}\right|_{b,y} = 0, \\ h(x, c) = 0, \;\; h(x, d) &= 1, \end{aligned} \tag{1}$$

where h is the hydraulic head, T is the transmissivity, and g is a source/sink (here, set to zero).

The forward modeling problem in equation (1) can be written as

$$h = f(T), \tag{2}$$

where $f(\cdot)$ is the forward operator mapping from the model parameter space to the measurement space. Specifically, the operator $f(\cdot)$ maps the model parameter T onto the state variable h according to equation (1). The operator $f(\cdot)$ is nonlinear in that the map from the model parameters T to the state variable h is not a linear map.

It may be worthwhile to mention that the computational technique developed in this paper is not limited to the specific forward-modeling operator defined in equation (1). It can be applied broadly to forward-modeling operators with various physical meanings.

Correspondingly, the problem of hydrogeologic inverse modeling is to estimate the transmissivity provided with available measurements. Usually, such a problem is posed as a minimization problem

$$\mathbf{m} = \arg\min_{\mathbf{m}} \left\{ ||\mathbf{d} - f(\mathbf{m})||_2^2 \right\}, \tag{3}$$

where $\mathbf{d}$ represents a recorded hydraulic head data set and $\mathbf{m}$ is the inverted model parameter, $\|\mathbf{d} - f(\mathbf{m})\|_2^2$ measures the data misfit, and $||\cdot||_2$ stands for the L_2 norm. Solving equation (3) yields a model $\mathbf{m}$ that minimizes the mean-squared difference between observed and synthetic data. However, because of the limited data coverage, solving the inverse problem based on equation (3) is ill posed. Moreover, because of the nonlinearity of the forward modeling operator f, the solution of the inverse problem may become trapped in a local minimum of the misfit function. Regularization techniques can be used to address the nonuniqueness of the solution and reduce the ill-posedness of the inverse problem.

The most commonly used regularization technique is the Tikhonov regularization [*Vogel*, 2002; *Hansen*, 1998]. Incorporated with equation (3), we have inverse modeling with Tikhonov regularization

$$\mathbf{m} = \arg\min_{\mathbf{m}} \{ l(\mathbf{m}) \}, \tag{4}$$

$$= \arg\min_{\mathbf{m}} \{ ||\mathbf{d} - f(\mathbf{m})||_2^2 + \lambda ||\mathbf{m}||_2^2 \}, \tag{5}$$

where the parameter λ is the Tikhonov regularization parameter, which controls the amount of regularization in the inversion.

To further account for the errors in the data measurement and the model, we follow the work in *Kitanidis and Lee* [2014] and *Lee and Kitanidis* [2014], and employ the generalized least squares that weights the data misfit and regularization terms in equation (5) using covariance matrices

$$\begin{aligned} \mathbf{m} &= \arg\min_{\mathbf{m}} \{ g(\mathbf{m}) \} \\ &= \arg\min_{\mathbf{m}} \{ ||\mathbf{d} - f(\mathbf{m})||_R^2 + \lambda ||\mathbf{m} - \mathbf{m_0}||_Q^2 \}, \end{aligned} \tag{6}$$

where $\mathbf{m_0}$ is the prior model parameters. The weighted data misfit and regularization terms are defined as

$$||\mathbf{d} - f(\mathbf{m})||_R^2 = (\mathbf{d} - f(\mathbf{m}))' R^{-1} (\mathbf{d} - f(\mathbf{m})), \tag{7}$$

$$||\mathbf{m} - \mathbf{m_0}||_Q^2 = (\mathbf{m} - \mathbf{m_0})' Q^{-1} (\mathbf{m} - \mathbf{m_0}), \tag{8}$$

where R is the covariance matrix of the data error and Q is the covariance matrix of the model parameters.

In Appendix A, we provide the residuals and gradients of equations (5) and (6) and show that their gradients share the same form

$$\text{Grad} = -J^T r(\mathbf{m}), \tag{9}$$

where the Jacobian matrix J and residual vector $r(\mathbf{m})$ are further defined according to the cases of either the ordinary least squares problem as in equation (5) or the generalized least squares problem as in equation (6).

In order to numerically solve the minimization problem in equation (6), we employ the Levenberg-Marquardt method because of its robustness [*Nocedal and Wright*, 2000; *Bertsekas*, 1999].

2.2. Classical Levenberg-Marquardt Method

The Levenberg-Marquardt (LM) method is a trust-region optimization method, which is usually posed as [*Nocedal and Wright*, 2000; *Bertsekas*, 1999]

$$\mathbf{m}^{(k+1)} = \mathbf{m}^{(k)} + \mathbf{p}^{(k)}, \tag{10}$$

where k is the iteration index and $\mathbf{p}^{(k)}$ is the search direction. The LM method combines the methods of steepest descent and Gauss-Newton. It can be seen as a damped Gauss-Newton method. The LM method comes after the work of *Levenberg* [1944] and *Marquardt* [1963]. In the Levenberg's original version of the method, the search direction is defined as

$$\mathbf{p}^{(k)} = -\left[(J^{(k)})' J^{(k)} + \mu I\right]^{-1} \text{Grad}. \tag{11}$$

Marquardt [1963] modified Levenberg's approach by using a different damping term

$$\mathbf{p}^{(k)} = -\left[(J^{(k)})' J^{(k)} + \mu \,\text{diag}\left((J^{(k)})' J^{(k)}\right)\right]^{-1} \text{Grad}. \tag{12}$$

The damping parameter, μ, is defined to be $\mu > 0$, which ensures the search direction of $\mathbf{p}^{(k)}$ is a descent direction. When the values of μ are large, the damping terms in equations (11) and (12) dominate the search direction, hence the search direction approximates that of the steepest descent. On the other hand, when the values of μ are small, the first terms in equations (11) and (12) become more significant than the damping term, hence, the search direction approximates a Gauss-Newton search direction. Therefore, by changing the value of the damping parameter μ, the LM method is capable of behaving like steepest descent method or Gauss-Newton method.

The classical Levenberg-Marquardt algorithm has been widely used in inverse modeling, because it is more robust than the Gauss-Newton method and yields a faster rate of convergence than the steepest descent method [*Finsterle and Kowalsky*, 2011; *Fienen et al*., 2009; *Nowak and Cirpka*, 2004; *Simunek et al*., 1998]. However, the efficiency and robustness of Levenberg-Marquardt algorithm is dependent on the correct selection of the damping parameter [*Nielsen*, 1999]. Improper selection of the damping parameter may lead to poor convergence or even divergence [*Naveen et al*., 2010; *Lampton*, 2009; *Araneda*, 2004]. Furthermore, at every LM iteration, trials of different damping parameter values lead to additional solutions of a linear system for search a direction, which increase the computational cost significantly. Therefore, it is very desirable to develop a Levenberg-Marquardt algorithm which not only yields accurate and robust results but reduces the cost of these additional solutions of a linear systems for a search direction.

3. Krylov-Subspace Approximated Levenberg-Marquardt Algorithm for Inverse Modeling

Krylov subspace methods are one of the most important tools for solving large-scale sparse problems [*Saad*, 2003; *Golub and Van Loan*, 1996]. For most hydraulic inversion problems, the data coverage is limited. In this section, we will first explore the matrix structure of the LM method and then discuss how to employ the Krylov subspace approximation technique to improve the efficiency of classical LM method.

3.1. Linear System Reformulation and Matrix Structure

According to the derivation in Appendix A, we can rewrite the gradient of the objective function as

$$\text{Grad} = -(J^{(k)})' r^{(k)}. \tag{13}$$

Therefore, solving for $\mathbf{p}^{(k)}$ in equations (11) and (12) can be posed equivalently as searching for the minima of the following two least squares problems, respectively,

$$\mathbf{p}^{(k)} = \arg\min_{\mathbf{p}^k} \left\{ \left\| \begin{bmatrix} J^{(k)} \\ \sqrt{\mu} I \end{bmatrix} \mathbf{p}^{(k)} - \begin{bmatrix} r^{(k)} \\ 0 \end{bmatrix} \right\|_2 \right\}, \tag{14}$$

$$\mathbf{p}^{(k)} = \arg\min_{\mathbf{p}^k} \left\{ \left|\left| \begin{bmatrix} J^{(k)} \\ \sqrt{\mu}\ \mathrm{diag}((J^{(k)})'J^{(k)}) \end{bmatrix} \mathbf{p}^{(k)} - \begin{bmatrix} r^{(k)} \\ 0 \end{bmatrix} \right|\right|_2 \right\}, \tag{15}$$

where equation (14) is the Levenberg version of the LM method and equation (15) is the Marquardt version of the LM method.

For most existing LM methods, QR decomposition or singular value decomposition (SVD) based direct solvers are used to solve equations (14) and (15) for the search direction, $\mathbf{p}^{(k)}$, because of their stability [*Madsen et al*., 2004; *Nielsen*, 1999]. However, the computational costs associated with those two methods are high for problems with a large number of parameters.

We observe that in both equations (14) and (15), the system matrices consist of two parts: the Jacobian matrix $J^{(k)}$ and a diagonal matrix, the identity matrix I in equation (14) or the diagonal matrix, $\mathrm{diag}((J^{(k)})'J^{(k)})$ in equation (15). At a given iteration k, the Jacobian matrix remains the same while the damping parameter μ varies. A significant amount of computational cost can be wasted without considering this special matrix structure. Second, for many inverse modeling problems, the model and data space are usually very large, leading to large Jacobian matrices. Hence, direct linear solvers based on QR or SVD methods might not be as efficient as iterative solvers [*Saad*, 2003]. Therefore, by taking these two points into consideration, we develop a computational method that can take advantage of the matrix structure of the problem and avoid the direct solution of the linear systems.

3.2. Krylov-Subspace Approximated Levenberg-Marquardt Method

We provide the details regarding the generation of the Krylov subspace utilizing Golub-Kahan-Lanczos (GKL) technique in Appendix B.

A straight-forward application of the GKL technique to the classical LM method is to let the system matrix J in equation (B1) equal the left-hand side of equations (14) and (15)

$$J = \begin{bmatrix} J^{(k)} \\ \sqrt{\mu} I \end{bmatrix} \tag{16}$$

or

$$J = \begin{bmatrix} J^{(k)} \\ \sqrt{\mu}\ \mathrm{diag}((J^{(k)})'J^{(k)}) \end{bmatrix}. \tag{17}$$

However, such a direct application of the GKL technique will be rather computationally expensive. We notice that the formulations of our problem in equations (14) and (15) are different from the one in equation (B1), in which there is no damping term. Below we show that the bidiagonalization technique described in Appendix B can still be applied to the damped least squares problems in equations (14) and (15) without adding extra computational costs.

3.2.1. Levenberg's Algorithm: The Case of Equation (14)

First we consider the case of equation (14). Employing a similar bidiagonalization procedure as in equations (B3) and (B4), we will have

$$\min_{\mathbf{p}^{(k)}} \left\{ \left|\left| \begin{bmatrix} J^{(k)} \\ \sqrt{\mu} I \end{bmatrix} \mathbf{p}^{(k)} - \begin{bmatrix} r^{(k)} \\ 0 \end{bmatrix} \right|\right|_2 \right\} \rightarrow \min_{\mathbf{y}^{(k)}} \left\{ \left|\left| \begin{bmatrix} B^{(k)} \\ \sqrt{\mu} I \end{bmatrix} \mathbf{y}^{(k)} - \beta^{(1)} \mathbf{e}^{(1)} \right|\right|_2 \right\}. \tag{18}$$

The subspace generated in the above equation (18) is

$$\begin{aligned} \mathcal{K}_k &= \mathrm{span}\left\{ (J^{(k)})'J^{(k)} + \mu I, (J^{(k)})'r^{(k)} \right\}, \\ &= \mathrm{span}\left\{ (J^{(k)})'J^{(k)}, (J^{(k)})'r^{(k)} \right\}. \end{aligned} \tag{19}$$

We observe that in equation (18), the bidiagonalization procedure is independent of the damping parameter μ. Furthermore, comparing the Krylov subspace generated in equation (19) for the damped least squares problem of equation (14) to the one in equation (B2), we observe that the Krylov subspace generated for the damped least squares problem is also independent of the damping parameter μ. These two facts can

be utilized to reduce the computational cost of solving for the search direction and therefore improve the efficiency of the algorithm.

The right-hand side of equation (18) is an augmented least squares problem with a system-matrix consisting of a lower-bidiagonal matrix $B^{(k)}$ and a diagonal matrix $\sqrt{\mu}I$. We employ Givens rotations to eliminate the diagonal matrix $\sqrt{\mu}I$ and meanwhile transform the lower-bidiagonal matrix $B^{(k)}$ into a upper bidiagonal matrix. All the details are provided in Appendix C.

Many efficient methods can be used to solve the resulting least squares problem in equation (C4) such as Gaussian elimination and back-substitution [*Golub and Van Loan*, 1996]. We employ a three-term recursion to obtain the solution, which is similar to those in equations (B13–B15),

$$\mathbf{m}^{(k)}=\mathbf{m}^{(k-1)}+(\phi^{(k)})'(\mathbf{z}^{(k)})', \qquad (\mathbf{z}^{(k)})'=\frac{1}{(\rho^{(k)})'}(\mathbf{v}^{(\mathbf{k})}-(\theta^{(k-1)})'(\mathbf{z}^{(k-1)})'). \tag{20}$$

3.2.2. Marquardt's Algorithm: The Case of Equation (15)

The major difference between Levenberg's formulation in equation (14) and Marquardt's formulation in equation (15) is that there is a nonidentity squared diagonal matrix as the lower part of the system matrix contained in Marquardt's algorithm.

Another equivalent form of equation (15) is

$$\mathbf{p}^{(k)}=\arg\min_{\mathbf{p}^{(k)}}\left\{\|J^{(k)}\mathbf{p}^{(k)}-r^{(k)}\|_2^2+\mu\|D\mathbf{p}^{(k)}\|_2^2\right\}, \tag{21}$$

where $D=\mathrm{diag}\left((J^{(k)})'J^{(k)}\right)$.

By using a variable substitution

$$\overline{\mathbf{p}}^{(k)}=D\mathbf{p}^{(k)}, \tag{22}$$

we can transform equation (21) into

$$\overline{\mathbf{p}}^{(k)}=\arg\min_{\overline{\mathbf{p}}^{k}}\left\{\|\bar{J}^{(k)}\overline{\mathbf{p}}^{(k)}-r^{(k)}\|_2^2+\mu\|\overline{\mathbf{p}}^{(k)}\|_2^2\right\}, \tag{23}$$

where $\bar{J}^{(k)}=J^{(k)}D^{-1}$.

The transformed equation (23) now shares the same form as that in equation (14), therefore all the aforementioned techniques can be applied to the Marquardt's algorithm in equation (23) and solve for new variable $\overline{\mathbf{p}}^{(k)}$. Once the minimization problem in equation (23) is solved, the original variable $\mathbf{p}^{(k)}$ can be obtained by

$$\mathbf{p}^{(k)}=D^{-1}\overline{\mathbf{p}}^{(k)}. \tag{24}$$

Because the matrix D is a diagonal matrix, solving the above equation is trivial.

4. Coarse-Grained Parallelized Levenberg-Marquardt Algorithm for Inverse Modeling

Parallelism has been used to increase the computational efficiency of the Levenberg-Marquardt method. Both coarse and fine-grained parallelism have been employed to LM methods in literature. As for a coarse-grained parallelism strategy, a parallel run manager, YAMR (Yet Another run ManageR), has been integrated with the latest version of the software package, PEST++ Version 3 [*Welter et al.*, 2015]. Another example of a coarse-grained parallelism strategy is the "Parallel PEST" option used in the software package of PEST [*Doherty and Hunt*, 2010]. A more common utilization of parallelism in the LM method is fine-grained parallelism, where parallelism is employed to manipulate a specific numerical operation. For example, in both the work of *Coleman and Plassmann* [1992] and *Zhu and Zhang* [2013], parallelism is applied to the QR factorization. Fine-grained parallelism can help improve the computational efficiency locally. However, because of the sequential selection scheme of the damping parameter, the efficiency of the locally parallelized LM methods will be bounded by sequential characteristics in selecting the optimal damping parameter. We develop an LM algorithm with a combination of both fine and coarse-grained parallelism. Specifically, in the

level of coarse-grained parallelism we solve for multiple search directions simultaneously instead of relying on a sequential damping parameter search. In the level of fine-grained parallelism, we utilize multicore versions of all the Basic-Linear-Algebra-Subprograms (BLAS). To further increase the computational efficiency, we employ the subspace recycling technique to exploit the separability of damping parameters from the solution subspace.

4.1. Parallel Selection of the Damping Parameter

Without loss of generality, we use Marquardt's algorithm in equation (15) to discuss our strategy to select the damping parameter μ.

In the sequential selection scheme of the damping parameter, the gain factor in equation (F1) is calculated to justify the validity of the damping parameter. Even though the linear solver procedure can be implemented in parallel, the overall computation can stagnate in selecting the appropriate damping parameter. We, on the other hand, generate multiple damping parameters at each iteration

$$\mu=\mu_0\times 10^{y}, \tag{25}$$

where $y=-n/2,-n/2+1,\ldots,n/2-1,n/2$ and n is the number of the damping parameters which are being used. Out of the n potential search directions, we select the one which yields the smallest function objective value. The objective function values can also be computed in parallel. Once the current iteration is completed, we update μ_0 according to equation (F2). The major benefit of parallel selection of the damping parameter is the avoidance of the sequential selection of the damping parameter at every LM iteration.

4.2. Recycled Krylov Subspace for Multiple Damping Parameters

Through the discussion of the Krylov subspace technique in section 3, we notice that provided with an initial damping parameter of μ_0 we will obtain not only the search direction of $\mathbf{p}^{(k)}$, but also the Krylov subspace generated during the GKL bidiagonalization procedure. As discussed previously, the bidiagonalization procedure in equation (18) is independent of the damping parameter μ and the Krylov subspace generated in equation (19) is also independent of the selection of the damping parameter. Hence, at every LM iteration step we can further save the computational cost at every CPU core by recycling the subspace generated from solving the linear system using the initial damping parameter μ_0.

Specifically, with n damping parameters generated according to equation (25), our LM method consists of two operations at each iteration: "linear solver" and "recycling." The operation of "linear solver" is to solve for the search direction using the first damping parameter μ_0, and the "recycling" is to obtain all the search directions by recycling the subspace for the rest of the damping parameters. The only procedures that we need to recompute for the "recycling" are the Givens rotation and the update of the solution as in equations (C3) and (20). Relative to the bidiagonalization step, the computational costs of Givens rotation and the solution update are trivial. Hence, the "recycling" is computationally negligible compared to the "linear solver" operation.

Therefore, our strategy to select the optimal damping parameter is to solve for the search direction in equation (15) using an initial damping parameter value μ_0, then simultaneously solve for multiple search directions using different values of the damping parameters according to equations (C3), (20), (24), and (25). Theoretically, both the steps of "linear solver" and "recycling" can be employed on master and worker nodes. However, considering the size of the target problems that we are interested in mostly ranging from moderate to large scale, we therefore prefer to employ both "linear solver" and "recycling" steps on the master node before transferring the upgrade vectors to the worker nodes. Such an implementation can be especially beneficial when there are limited resources such as computing nodes or storage devices, and a large number of damping parameters need to be evaluated. It might therefore be beneficial in preventing our applications from being memory bound. We will demonstrate in the next section that the total computational cost of multiple damping parameters is nearly the same as the cost with one damping parameter.

In Appendix D, we provide a detailed descriptions as Algorithm 1 to summarize our new Levenberg-Marquardt method based on Krylov subspace approximation and parallel selection of the damping parameters. We name our new LM method as "LM-RLSQR" because of the "R"ecycled-Krylov-subspace. We also employ traditional "LSQR" techniques (without recycling) for comparison.

5. Numerical Results

In this section, we provide numerical examples to demonstrate the efficiency of our new Levenberg-Marquardt algorithm. The reference problem is the steady state groundwater equation on the square domain, $[0, 1]\times[0, 1]$, with fixed hydraulic head at $y = 0$ and $y = 1$, zero flux boundaries at $x = 0$ and $x = 1$.

The groundwater equation was solved using the finite difference method on a uniform grid. The parameter grids are composed of x transmissivity nodes and y transmissivity nodes. The adjoint-state method [*Strang*, 2007] was used to compute the Jacobian matrix. For the purposes of calibration, the transmissivity and hydraulic head were "observed" from a solution of the steady state groundwater equation using a reference transmissivity field.

To have a comprehensive comparison, we provide five sets of tests. In section 5.1, we compare the performance of our recycled-LSQR algorithm to other commonly used linear solvers. In section 5.2, we report the performance of our method in different levels of the model heterogeneity. In section 5.3, we provide a comparison of our parallel Levenberg-Marquardt algorithm using recycled-LSQR solver to the parallel Levenberg-Marquardt algorithms using other linear solvers in calibrating 2-D models with different numbers of parameters. In section 5.4, we employ our method to calibrate a more complex 3-D model and report its corresponding results. In section 5.5, we compare our parallel Levenberg-Marquardt algorithm to the sequential Levenberg-Marquardt algorithm. We denote our method as "LM-RLSQR." We further denote two other reference LM methods, Levenberg-Marquardt method using the QR factorization as "LM-QR" and Levenberg-Marquardt method using the SVD decomposition as "LM-SVD."

We select Julia as our programing tool because of its efficiency and simplicity. Julia is a high-level programming language designed for scientific computing [*Bezanson et al*., 2014]. The Julia code for our LM algorithm is available as a part of the upcoming open-source release of Julia version of MADS (Model Analysis and Decision Support) at "http://mads.lanl.gov/" [*Vesselinov et al*., 2015]. For the methods of LM-QR and LM-SVD, the QR factorization and SVD decomposition are both implemented using the system routines provided in the Julia packages. Julia also contains both sequential and parallel BLAS operations, which provide us the fine-grained parallelism for the LM algorithms.

As for the computing environment, we run the tests on a Linux desktop with 40 cores of 2.3 GHz Intel Xeon E5-2650 CPUs, and 64.0 GB memory. To maximize the usage of all the CPU cores for our parallel algorithm, we specify 10 workers in the coarse-grained parallelism level and four threads for each of the workers in the fine-grained parallelism.

The stopping criterion is an important issue for any iterative method including the Levenberg-Marquardt algorithm. In our work, we employ two stopping criterion shown below to justify the convergence of the iteration,

$$(J^{(k)})' r^{(k)} \leq \mathrm{TOL}_1, \tag{26}$$

$$||\mathbf{p}^{(k)}||_2 \leq \mathrm{TOL}_2\,(\mathrm{TOL}_2 + ||\mathbf{m}^{(k)}||_2), \tag{27}$$

where $\mathrm{TOL}_1 = 1.0\times10^{-6}$ and $\mathrm{TOL}_2 = 1.0\times10^{-3}$. If either equation (26) or equation (27) are satisfied, the iteration procedure will stop.

5.1. Test on the Projection LSQR Algorithm

In our first numerical example, we first test the accuracy of our recycled LSQR method ("LM-RLSQR") using a reference synthetic log transmissivity field in Figure 2a. The reference method is the parallel LM method with QR factorization ("LM-QR"). The reference model is solved on a grid containing 50 × 50, pressure nodes and a total of 5100 model parameters (50 × 51 log-transmissivities along x axis, 51 × 50 log-transmissivities along y axis). We generate a ground truth, which is shown in Figure 2a. We utilize the variance ($\sigma^2_{\mathbf{m}}$) and an exponent ($\beta_{\mathbf{m}}$—related to the fractal dimension of the field and the power law of the field's spectrum) to characterize the heterogeneity of the considered fields [*Peitgen and Saupe*, 1988]. In this example, we set the variance $\sigma^2_{\mathbf{m}} = 0.25$ and power $\beta_{\mathbf{m}} = -3.5$. The white circles in the figure are the locations for the hydraulic head observations.

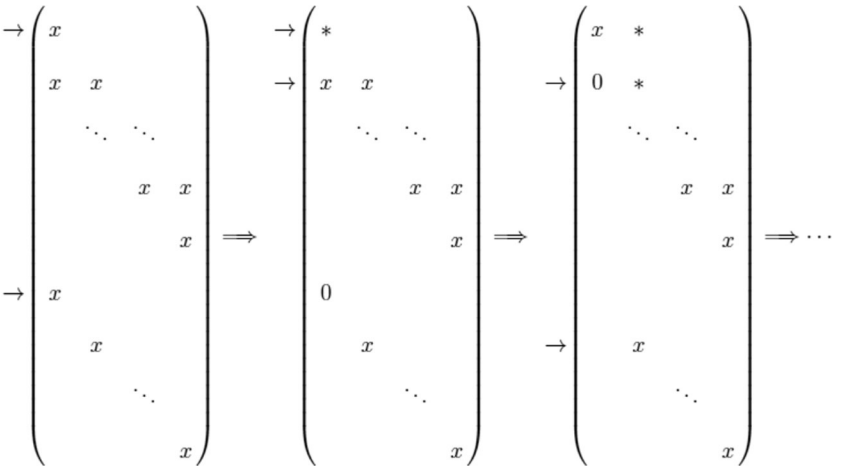

Figure 1. Two Givens rotations are employed to eliminate both the lower diagonal elements in B_k and the diagonal elements in the identity matrix, I. The arrow "$\rightarrow$" points to the current operating row. The symbol "x" means the element of the matrix, and the symbol "$*$"means the element being modified.

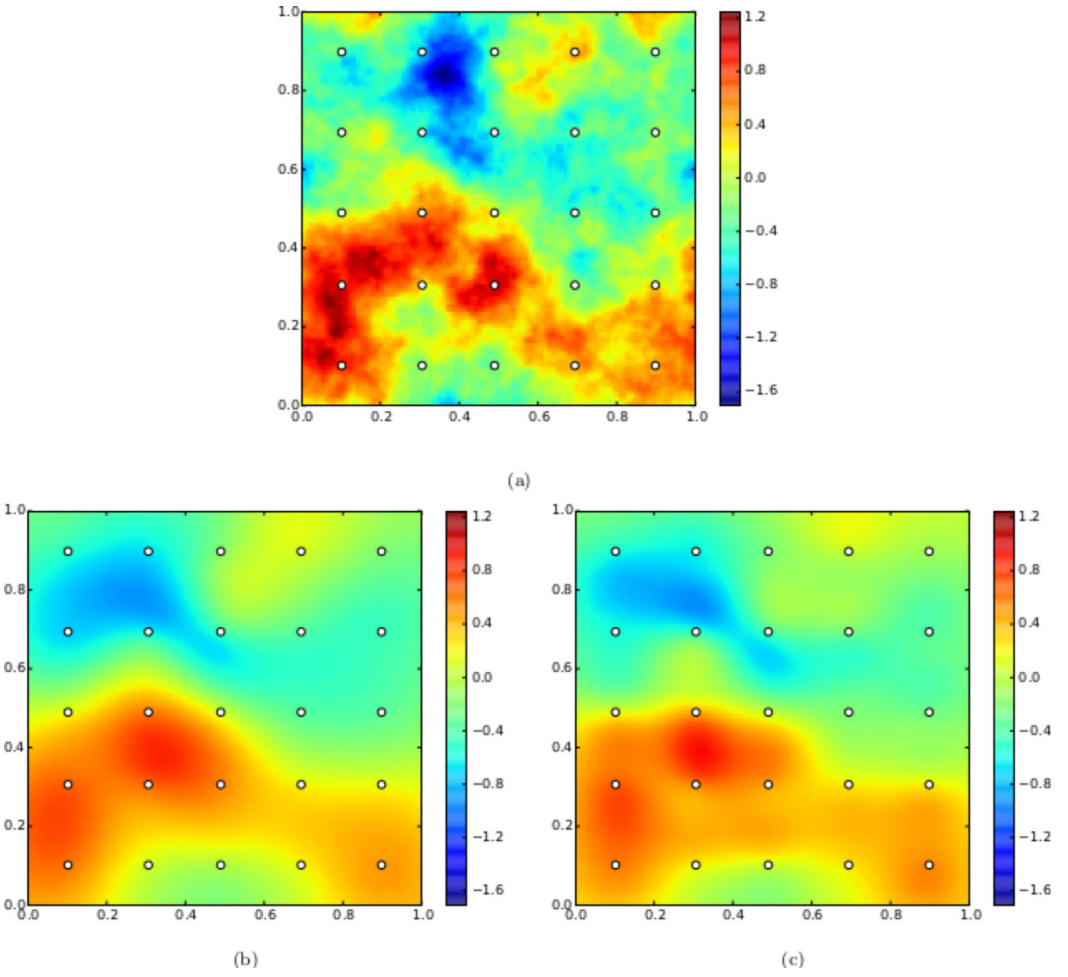

Figure 2. Synthetic log transmissivity field (a) with variance $\sigma^2_{\mathbf{m}}$=0.25 and power $\beta_{\mathbf{m}}$=−3.5. Hydraulic conductivity and hydraulic head observation locations are indicated with circles. The results of (b) the inverse modeling solved by LM-QR and our (c) new Levenberg-Marquardt algorithm are shown. They are visually identical to each other. The RME values of the results in Figures 2b and 2c are 49.0 and 51.0%, respectively. Hence, our method yields a comparable result to that obtained using the LM-QR method.

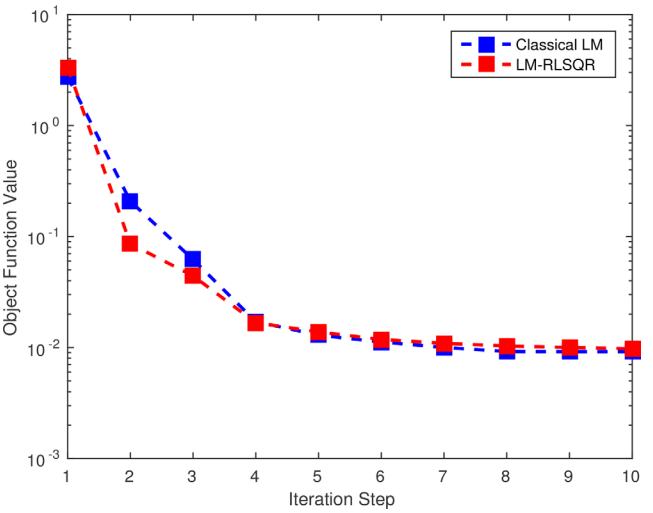

Figure 3. The convergence of the classical Levenberg-Marquardt algorithm using QR factorization in blue and our efficient Levenberg-Marquardt algorithm in red (LM-RLSQR) are provided. The rates of convergence using these two methods are very close to each other.

We implement the LM-RLSQR method using 10 damping parameters simultaneously and provide the inversion result using LM-QR and LM-RLSQR in Figures 2b and 2c, respectively. Comparing to the true model in Figure 2a, our method obtains a good result, representing both the high and low log-permeability regions. Visually, our method yields a comparable result to the one obtained using LM-QR in Figure 2b. To further quantify the inversion error of different inverse modeling methods, we calculate the relative-model-error (RME) of the inversion

$$\mathrm{RME}(\mathbf{m}) = \frac{||\mathbf{m}-\mathbf{m}_{\mathrm{ref}}||_2}{||\mathbf{m}_{\mathrm{ref}}||_2}, \qquad (28)$$

where $\mathbf{m}$ is the inversion and $\mathbf{m}_{\mathrm{ref}}$ is the ground truth. According to equation (28), the RME value of inversion result in Figure 2b using LM-QR is 49.0%. By contrast the RME value of our result in Figure 2c is 51.0% with about 2.0% difference, which quantitatively verifies that our method yields comparable results to the LM-QR method.

We provide the plot of the rate of convergence of our method in Figure 3 for the different methods. For comparison, we also provide the result using the classical sequential LM method using the QR-factorization-based linear solver. We observe that both our LM-RLSQR and the classical LM method yield a very similar rate of the convergence. At each iteration, these two methods yield almost the same objective function values. Therefore, together with the inversion result in Figure 2, we demonstrate that LM-RLSQR yields a comparable accuracy to the classical Levenberg-Marquardt method.

To compare the computational efficiency, we measure the computational time used in solving the linear systems in equation (21) for three methods in Figure 4: the parallel LM method with QR factorization ("LM-QR"), the parallel LM method with SVD decomposition ("LM-SVD"), and our method ("LM-RLSQR"). The color-bar stands for the computational time. The time cost using LM-RLSQR is in Figure 4a, and the costs for LM-QR and LM-SVD are provided in Figures 4b and 4c, respectively. As for LM-RLSQR, the first row in Figure 4a is lighter than the remaining rows, indicating that most of the computational time is spent on the first damping parameter, and then immediately reduced to almost zero for the remaining damping parameters. As for the LM-QR method, the time costs are much more expensive than those of the LM-RLSQR method. The costs in first row in Figure 4b are more expensive than the remaining rows because it involves the full calculation of the normal equation. As for the LM-SVD method, the time costs are the most expensive out of all three methods and are evenly distributed over all damping parameters.

We further provide the average and total time cost in Figure 5 based on Figure 4 to obtain a quantitative comparison. Specifically, Figure 5a is the average time cost in every damping parameter for three methods

$$\mathrm{AverageTime}[\mathrm{i}] = \frac{\sum_{\mathrm{j=Iteration}} \mathrm{Time}[\mathrm{i},\mathrm{j}]}{\mathrm{Number\ of\ Iteration}}, \qquad (29)$$

where i is the index for the damping parameter, j is the index for the iteration, and the variable of $\mathrm{Time}[\mathrm{i},\mathrm{j}]$ is the time cost of solving the linear system using the ith damping parameter at the jth iteration.

Figure 5b is the total time costs at each iteration step for all three methods

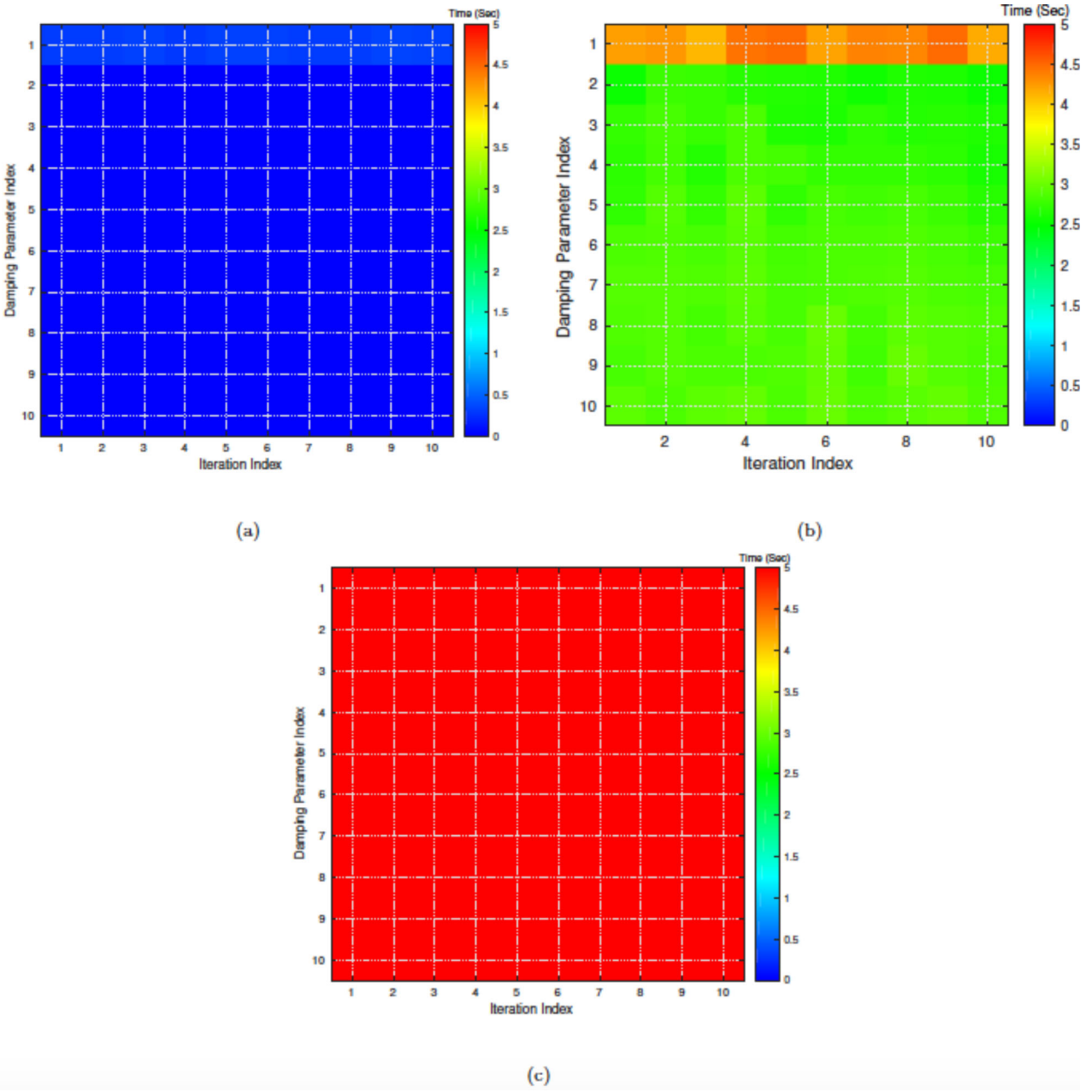

Figure 4. The computational time costs in solving the linear systems using different damping parameters at all iteration steps for (a) our new LM method, (b) LM-QR method, and (c) LM-SVD method. The color-bar stands for the computational time. The *x* axis represents the LM iteration step. The *y* axis represents index of the damping parameter. Our new method is the most efficient in solving the linear system out of three methods.

$$\text{Total Time}[j] = \sum_{i=\text{dampingparameter}} \text{Time}[i,j], \tag{30}$$

where *i*, *j*, and Time[i, j] has the same meaning as the one in equation (29).

In Figure 5a, our method (in red) spends about 0.3 s on average for the first damping parameter, then its time costs significantly reduce to about 10^{-4} s for the rest of the damping parameters. The LM-QR method (in blue) spends about 4.3 s for the first damping parameter and 2.8 s for the rest of the damping parameters. The LM-SVD method (in green) is much more time-consuming than both LM-QR and LM-RLSQR methods. It consistently spends over 100 s for each of the damping parameters.

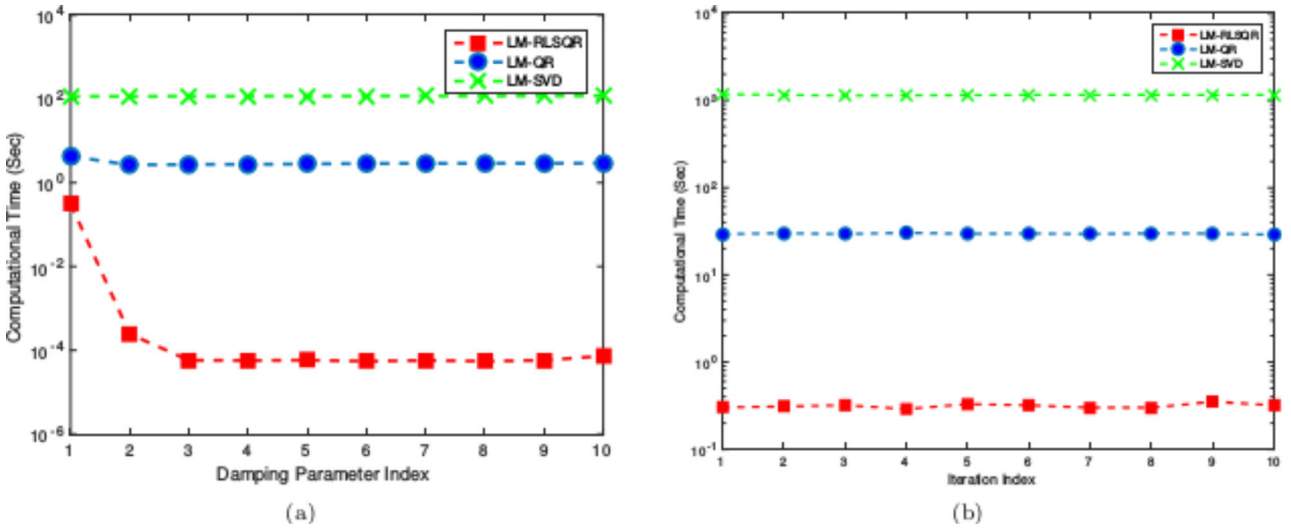

Figure 5. (a) The average time cost in solving the linear systems of every damping parameter for three methods and (b) the total computational time in solving the linear systems of all three methods at each iteration step. Our method (in red) yields the least computational time in solving the linear systems comparing to the LM-QR method (in blue) and the LM-SVD method (in green).

In Figure 5b, the overall computational time at each iteration step of LM-RLSQR (in red) is the lowest one among the three methods. For each iteration, LM-RLSQR only costs about 0.3 s opposed to 29.6 and 1152.5 s for the LM-QR and LM-SVD methods, respectively.

Therefore, LM-RLSQR significantly reduced the computational costs in solving the linear systems comparing to both LM-QR and LM-SVD methods. LM-RLSQR's advantages come in two parts. One contribution comes from the recycling of the Krylov subspace, which reduces the computational complexity by an order of the model dimension for all but the first damping parameter. This can be observed from Figures 4a and 5a, where there is little computational time spent in the rest of the damping parameters. The other contribution comes from employment of the subspace approximation of the original problem as we discussed in section 3. Instead of solving the problem in the original space, LM-RLSQR solves the approximated problem in a low-dimensional subspace spanned by the basis derived in equation (19), thereby enabling it to be very efficient. To conclude for this test, our LM-RLSQR method can be much more efficient than both the LM-QR and LM-SVD in solving the linear systems using the same set of damping parameters.

5.2. Test on the Levels of Heterogeneity

We use the variance ($\sigma^2_{\mathbf{m}}$) and exponent ($\beta_{\mathbf{m}}$) from the power law spectrum to characterize the heterogeneity of the considered fields [*Peitgen and Saupe*, 1988]. According to *Nowak and Cirpka* [2004], the performance of the LM algorithm can be impacted with respect to the levels of the heterogeneity. In this section, we test our method on different levels of the heterogeneity and compare the results to the LM-QR method. The model consists a total of 5100 model parameters (50 $\times$ 51 log-transmissivities along x axis, 51 $\times$ 50 log-transmissivities along y axis).

Similar to the tests in *Nowak and Cirpka* [2004], we first vary model heterogeneity by changing the value of the variance ($\sigma^2_{\mathbf{m}}$). The comparison results are reported in Table 1. Five different levels of variances are tested, $\sigma^2_{\mathbf{m}}$=0.1, 0.4, 1.6, 3.2, and 6.4. The efficiency of our method and the reference method are compared and reported based on the "time cost on linear solver" and "overall time cost" (Columns 2 and 4). With different levels of variance, our method is much more efficient than the LM-QR method. The average speed-up ratio is about 20 times for the linear solver and 5 times for the overall cost. The accuracy of results are reported in Columns 3 and 5 using the RME value defined in equation (28). Comparing Column 3 to Column 5, both methods yield results with comparable accuracy.

Another variable specifying the heterogeneity of a model is the exponent, β. While these fields are not defined in terms of a correlation length, the parameter β plays a role similar to the correlation length. As β decreases, the fields become smoother (analogous to increasing the correlation length), and as β increases, the fields become rougher (analogous to decreasing the correlation length) [*Peitgen and Saupe*,

Table 1. Performance Comparison on the Levels of the Heterogeneity (Change of Variance, $\sigma^2_{\mathbf{m}}$) Using the LM-QR Method (Columns 2 and 3) and Our Method of LM-RLSQR (Columns 4 and 5)[a]

	LM-QR		LM-RLSQR	
Variance ($\sigma^2_{\mathbf{m}}$)	Time (s)	RME (Equation (28))	Time (s)	RME (Equation (28))
0.1	58.3/66.9	0.50	3.3/14.4	0.50
0.4	57.6/66.1	0.49	3.1/13.3	0.52
1.6	58.2/66.8	0.63	3.2/14.1	0.63
3.2	58.9/67.5	0.65	3.1/12.9	0.67
6.4	61.7/70.7	0.72	3.1/13.4	0.75

[a]A total of 5100 model parameters (50 × 51 log-transmissivities along *x* axis, 51 × 50 log-transmissivities along *y* axis) are created. The time profile in Columns 2 and 4 are reported in the format of "time cost on linear solver" and "overall time cost." The relative model error (RME) reported in Columns 3 and 5 are reported using equation (28). Our method is much more efficient than the LM-QR method with comparable accuracy with respect to various levels of heterogeneity.

1988]. We vary the value of the fractal field, $\beta = -2.0, -2.3, -2.6, -2.9$, and -3.2 and report the corresponding results of time costs and RME in Table 2. Again, our method yields similar accuracy to those obtained using LM-QR method. Specifically, observed from Table 2 as the model becomes more heterogeneous (larger β cases).

To summarize, through the results reported in Tables 1 and 2, the performance of our method of LM-RLSQR in different levels of the heterogeneity is comparable to that of the LM-QR method.

5.3. Test on the Scalability of LM-RLSQR Algorithm

To better understand the performance of our method under a variety of circumstances, we test our method of LM-RLSQR using different sizes of the problem and varying levels of heterogeneity. The number of hydraulic head nodes in the problem are 364, 1300, 2964, 5100, 9384, 10,512, and 12,324. The largest three among these involve 9384, 10,512, and 12,324 parameters, and for these we solve inverse problem using three different sets of parameters for the random field $(\sigma^2_{\mathbf{m}}, \beta_{\mathbf{m}}) = (0.25, -3.5)$, $(0.4, -3.2)$, and $(1.6, -2.9)$, respectively. We provide the true and inverted transmissivity fields for the three largest models with 9384, 10,512, and 12,324 parameters in Figure 6.

In this example, we test our inversion algorithm using 10 damping parameters. The inversion results for the three largest problems are provided in Figures 6b, 6d, and 6f. We first compare the accuracy of the inversion results using our method to those obtained using the LM-QR method and report in Figure 7 the RME value defined in equation (28). The RME values of the inversion results using our method (in red) and those obtained using the LM-QR method (in blue) are comparable to each other in Figure 7. Hence, our inversion method yields good accuracy with respect to varying sizes of the models, consistently. Also, we notice that with the same number of observations, the quality of the inversion result does not improve when the number of model parameters is increased. This is because of the ill-posedness of the problem. The limited data coverage becomes the limiting factor in the resulting inversion accuracy.

In Figure 8, we provide the computational time for all three methods and the speed-up ratio of our method versus the LM-QR and LM-SVD methods. Specifically, both the overall computational time in solving the linear systems and the time for the inversion are provided.

Table 2. Performance Comparison on the Levels of the Heterogeneity (Change of Exponent, β) Using the LM-QR Method (Columns 2 and 3) and Our Method of LM-RLSQR (Columns 4 and 5)[a]

	LM-QR		LM-RLSQR	
Exponent (β)	Time (s)	RME (Equation (28))	Time (s)	RME (Equation (28))
−2.0	58.5/66.0	0.88	3.0/11.7	0.86
−2.3	57.7/66.2	0.80	3.0/12.1	0.79
−2.6	58.3/66.4	0.71	3.0/12.7	0.71
−2.9	58.3/65.9	0.62	3.1/11.9	0.63
−3.2	57.1/65.8	0.55	3.1/12.4	0.57

[a]A total of 5100 model parameters (50 × 51 log-transmissivities along *x* axis, 51 × 50 log-transmissivities along *y* axis) are created. The time profile in Columns 2 and 4 are reported in the format of "time cost on linear solver" and "overall time cost." The relative model error (RME) reported in Columns 3 and 5 are reported using equation (28). Our method is much more efficient than the LM-QR method with comparable accuracy with respect to various levels of heterogeneity.

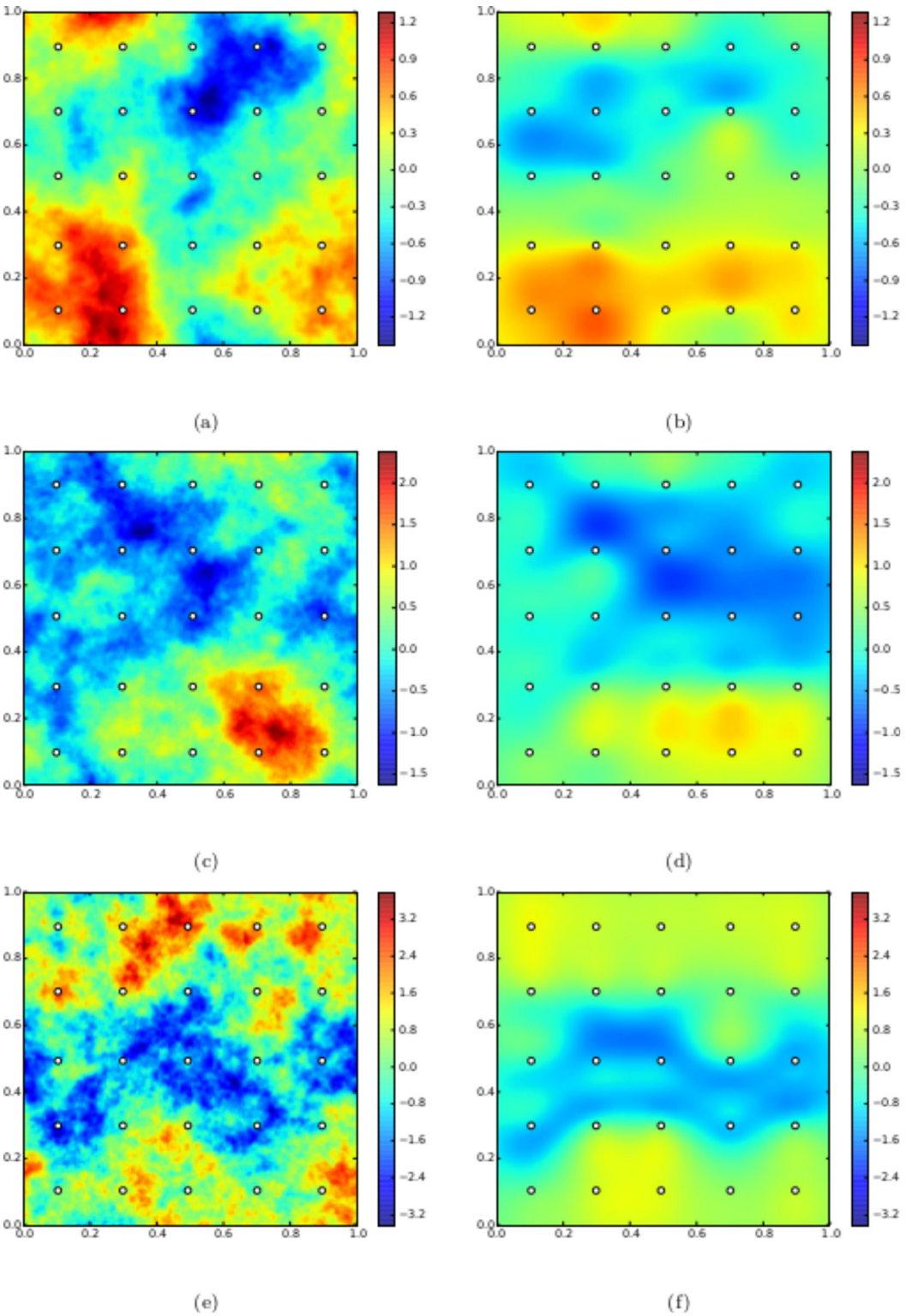

Figure 6. The results using our method for different sizes of the problem: (a, c, e) the true and (b, d, f) inverted log transmissivity fields using 9384, 10,512, and 12,324 model parameters. Three different types of heterogeneity are employed to create the true models, $(\sigma^2_{\mathbf{m}}, \beta_{\mathbf{m}}) = (0.25, -3.5)$, $(0.4, -3.2)$, and $(1.6, -2.9)$, respectively. Our method yields consistently good accuracy in all cases comparing to LM-QR method.

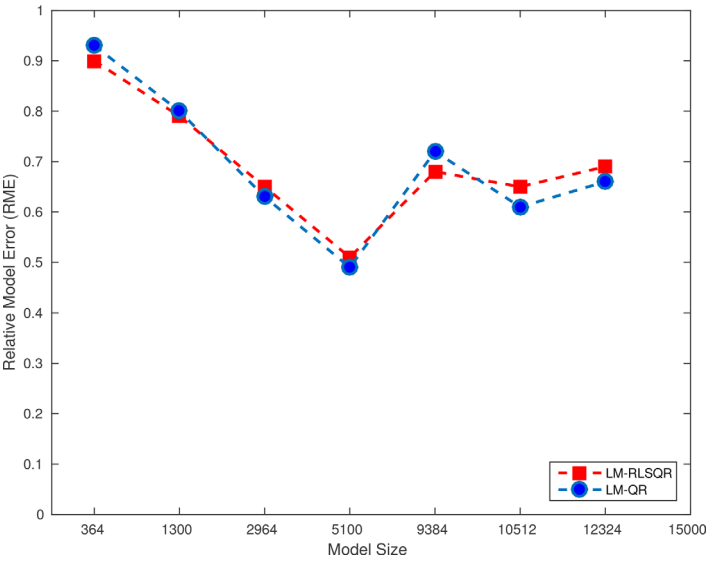

Figure 7. The RME values provided in equation (28) of the inversion results using our method (in red) and those obtained using the LM-QR method (in blue). Both methods yield comparable RME values with respect to different sizes of the models.

Figures 8a and 8b show the results for the overall computational time on the linear solver. In Figure 8a, all three methods start with comparable time cost when the problem sizes are small. However, as the problem size increases, the LM-SVD method (in green) becomes much more expensive than the other two methods. Because of limited computational resources, we only include results up to the number of the model parameters around 9384 for LM-SVD method. The LM-QR method (in blue) is a more efficient algorithm compared to the LM-SVD method. For the problem with 12,324 model parameters, LM-QR method requires about 927.7 s. For our method (in red), the computational cost is even more efficient than that of the LM-QR method. For the problem with 12,324 model parameters, our method only spends around 15.7 s.

Another way to visualize the computational efficiency is to compute the relative speed-up ratio between our method and the LM-QR and LM-SVD methods. The speed-up ratio, r, is calculated by

$$r = \frac{\text{Time}_1}{\text{Time}_2}, \tag{31}$$

where Time_1 corresponds to the computational time of the reference method and Time_2 corresponds to the computational time of our method. We provide the speed-up ratios of our method over the LM-QR and LM-SVD methods in solving the linear systems shown in Figure 8b. As the problem size increases, the speed-up ratio also increases. The largest speed-up ratio is about 59 for LM-QR method and 881 for LM-SVD method. Therefore, our method yields a significant speed-up ratio over both the LM-QR and LM-SVD methods in solving the linear systems.

In addition to the computational time solving the linear systems, a portion of the computational time is used to calculate the Jacobian matrix, the forward model, and communicate among the computational nodes. In Figures 8c and 8d, we provide the overall computational time in the inversion for all three methods when solving different sizes of the problems. The computational time costs are provided in Figure 8c and the corresponding speed-up ratio is in Figure 8d. The discrepancies among all three methods in the time cost become smaller when counting the overall computation time. Our method still yields the most efficient results compared to LM-QR and LM-SVD methods. Visualized in Figure 8d, our method yields about 16 times speed-up ratio over the LM-QR method, and 216 speed-up ratio over the LM-SVD method.

5.4. Test on a Three-Dimensional Model

The 3-D model solves the steady state groundwater equation on a domain that is 500 [m]×200 [m]×10 [m] with extraction of 10 gallons per minute (6.31×10^{-4} m^3/s) of groundwater from a well near the middle of the domain (see the depression in head in Figure 10). The model uses fixed head boundary conditions on the east and west boundaries and zero flux boundaries on the north, south, top, and bottom boundaries. The inverse analysis estimates 11,052 parameters, 10,926 of which are unknown hydraulic conductivities and 126 of which correspond to the unknown fixed head boundary conditions on the east and west boundaries. A synthetically generated reference log conductivity field was used to obtain the "observations." The reference field was a sample Gaussian random field with an anisotropic exponential covariance model

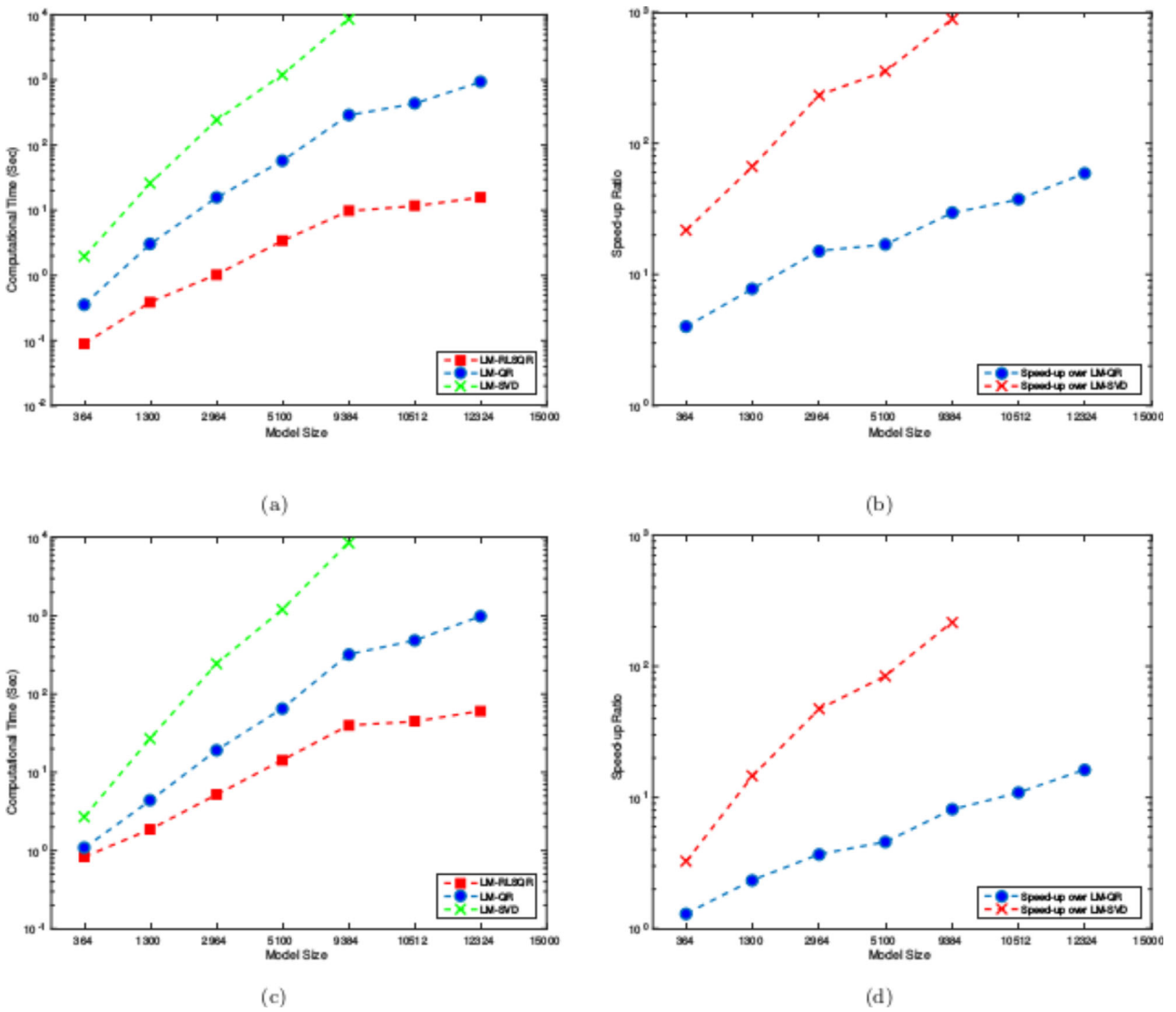

Figure 8. The computational time (a) for different model sizes in solving the linear systems and (b) the corresponding speed-up ratio. The total computational time (c) for inversion with different model sizes (including solving the linear system, forward modeling, and data communication, etc.), and (d) the corresponding speed-up ratio. As the model size increases, our method yields much smaller computational time costs than those of the LM-QR or LM-SVD methods. The largest speed-up ratio in solving the linear system is about 59 times compared to LM-QR method and 881 times compared to LM-SVD method; while the largest speed-up ratio in inversion is about 16 times opposed to LM-QR method and 216 times opposed to the LM-SVD method.

$$\mathrm{Cov}(\ln[k(x_1,y_1,z_1)],\ln[k(x_2,y_2,z_2)])=\sigma^2\exp\left(-\frac{\sqrt{(x_1-x_2)^2+(y_1-y_2)^2+4(z_1-z_2)^2}}{s}\right), \tag{32}$$

where $\sigma=2$, $s=50$ [m], and the mean of the field was $-4\ln(10)$. The hydraulic conductivity was then taken to be $e^{\ln k}$ [m/s]. We show the three layers of the 3-D model from the top to the bottom in Figures 9a–9c, respectively.

The hydraulic-head observations were taken from each of two screens in 17 observation wells distributed throughout the domain (see Figure 10). The two screens at each well are located in the top and bottom layers. The objective function to be minimized consists of three terms: (1) a term describing the data misfit, (2) a term regularizing the log-conductivities, and (3) a term describing the misfit from the prior information about the east and west boundary conditions. The true boundary conditions are 1 [m] on the west boundary and 0 [m] on the east boundary. Prior knowledge of the boundary conditions at the east and west

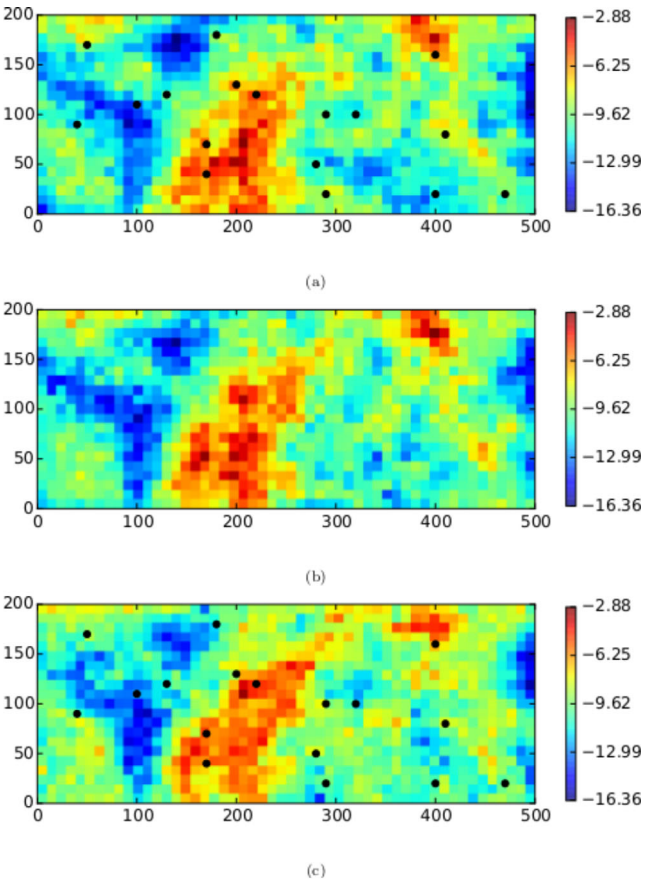

Figure 9. The (a) top, (b) middle, and (c) bottom layers of the 3-D model. The "true" log conductivity is obtained from a sample Gaussian random field with an anisotropic exponential covariance model defined in equation (32) with $\sigma = 2$, s=50 [m], and the mean of the log conductivity field was $-4\ln(10)$.

boundaries was assumed to be available, but different from the true boundary conditions. The prior "expected" boundary condition on the west boundary was 0.99 [m] and -0.01 [m] on the east boundary.

We employ both "LM-QR" (the reference method) and our method, "LM-RLSQR" to the observations in Figure 10. For this test, we use five damping parameters at each LM iteration, i.e., $n = 5$ in equation (25) and produce the inversion results in Figure 11. Specifically, the inversion results of the log-hydraulic conductivity along the top, middle, and the bottom layers of the 3-D model using the reference "LM-QR" method are shown in Figures 11a, 11c, and 11e, respectively. The results of the three layers using our "LM-RLSQR" method are shown in Figures 11b, 11d, and 11f. Visually, both methods yield results that are similar to each other. Quantitatively, the RME value of the inversion to the ground truth using "LM-QR" method is 18.0%, while the RME value of the inversion using "LM-RLSQR" is 18.5%. This RME only includes errors in the log conductivity. The inversion results of the boundary conditions using both methods are provided in Figure 12, where both the reference method and our method produce very similar results. However, our method is much more efficient than "LM-QR" method. The computational time on the linear solver using "LM-RLSQR" is 17.6 second, opposed to 203.4 s for the "LM-QR" method. The overall computational times using "LM-RLSQR" and the "LM-QR" methods are 44.3 and 220.6 s, respectively. The speed-up ratio of our method over the "LM-QR" method is about 11 times for linear solver, and 5 times for the overall.

5.5. Test With Sequential Methods

Classical Levenberg-Marquardt algorithms are usually implemented sequentially. Its efficiency and performance can be significantly impacted by the number of trials needed to find optimal values of the damping parameter. In order to obtain the optimal damping parameter at each iteration, *Nielsen* [1999] suggested a set of parameters: ρ_1=0.25, ρ_2=0.75, $\beta = 2$, and $\gamma = 3$. The parameters ρ_1, ρ_2, β, and γ are defined in Appendix F. However, this selection of parameters may not always be ideal and the ideal choice is dependent upon the characteristics of the inverse problem.

In this section, we provide a comparison of our method to two implementations of the classical sequential LM algorithms: one with a set of parameters that always produces a good damping parameter on the first try, and another other one where several damping parameter trials are sometimes required before a good damping parameter is obtained. The linear solver for the sequential LM algorithm is based on QR factorization, because of its superior performance to the SVD-based linear solver. We set up the model dimension to be 35 $\times$ 35 and report the results in Figure 13.

Figure 13a are the convergence plots of our parallel LM method and the classical sequential LM method. Both our method and the reference converge in 10 steps. In Figure 13b, we provide the number of trials for

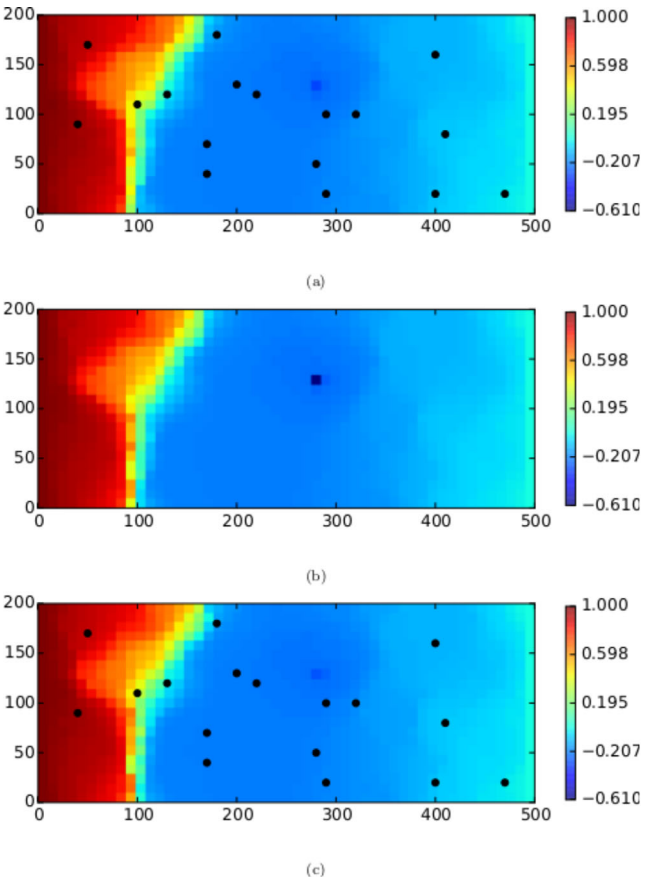

Figure 10. The hydraulic heads obtained using the "true" heterogeneity field along the (a) top, (b) middle, and (c) bottom layers of the 3-D model domain. The black dots show the locations of the observation wells.

finding the optimal damping parameter at each iteration for the two classical sequential LM algorithms. We note that if the parameters are not very well tuned (blue line), more searching trials will be required. The search for the optimal damping parameter means that extra linear systems must be solved, thereby increasing the computational costs. We plot the time profiles for all three cases in Figure 13c: our parallel LM method, and the classical LM methods using the optimal and nonoptimal parameter sets as shown in Figure 13b. Both the computational time profiles on linear solver (blue) and inversion (yellow) are provided. The extra computational time costs can be observed by comparing the two implementations of the classical LM methods. Our parallel LM method has the least computational time costs for both the linear solver and the inversion.

6. Conclusions

We have developed a computationally efficient Levenberg-Marquardt algorithm for highly parameterized inverse modeling that is well-suited to a parallel computational environment. Our method involves both coarse and fine-grained parallelism. At the level of coarse-grained parallelism, we develop a parallel search scheme, which handles multiple damping parameter at the same time. For each damping parameter, we employ a Krylov-subspace-recycling linear solver to solve the linear system in the Levenberg-Marquardt algorithm. Specifically, we first build a subspace with the Krylov basis obtained from solving the linear system using the first damping parameter, and then we further project the linear systems using the rest of the damping parameters down to the generated subspace.

Through our computational cost analysis, we show that the efficiency of the Levenberg-Marquardt algorithm can be significantly improved by these computational techniques. The actual computational complexity can be reduced by an order of the model dimension for all but the first damping parameter. We then applied our method to solve a steady state groundwater equation and compared the performance to two other widely used methods (LM-QR and LM-SVD) in both parallel and sequential computing environments. Our numerical examples demonstrate that our new method yields accurate results, which are comparable to those obtained from LM-QR and LM-SVD methods. Most importantly, our method is much more efficient than LM-QR and LM-SVD methods.

To summarize, with the significant improvement of the computational power in the past decade, the linear algebra solver is often the bottleneck for the successful utilization of Levenberg-Marquardt algorithm in hydraulic inverse modeling. The contribution of our work is to separate the damping parameter from the solution space and employ a Krylov-subspace recycling technique to reduce the computational costs. Our method can be mostly effective and efficient for inverse-model applications when a large number of model parameters need to be calibrated and the Jacobian matrix (representing derivatives of model observations and regularization terms with respect to model parameters) can be computed relatively efficiently (for

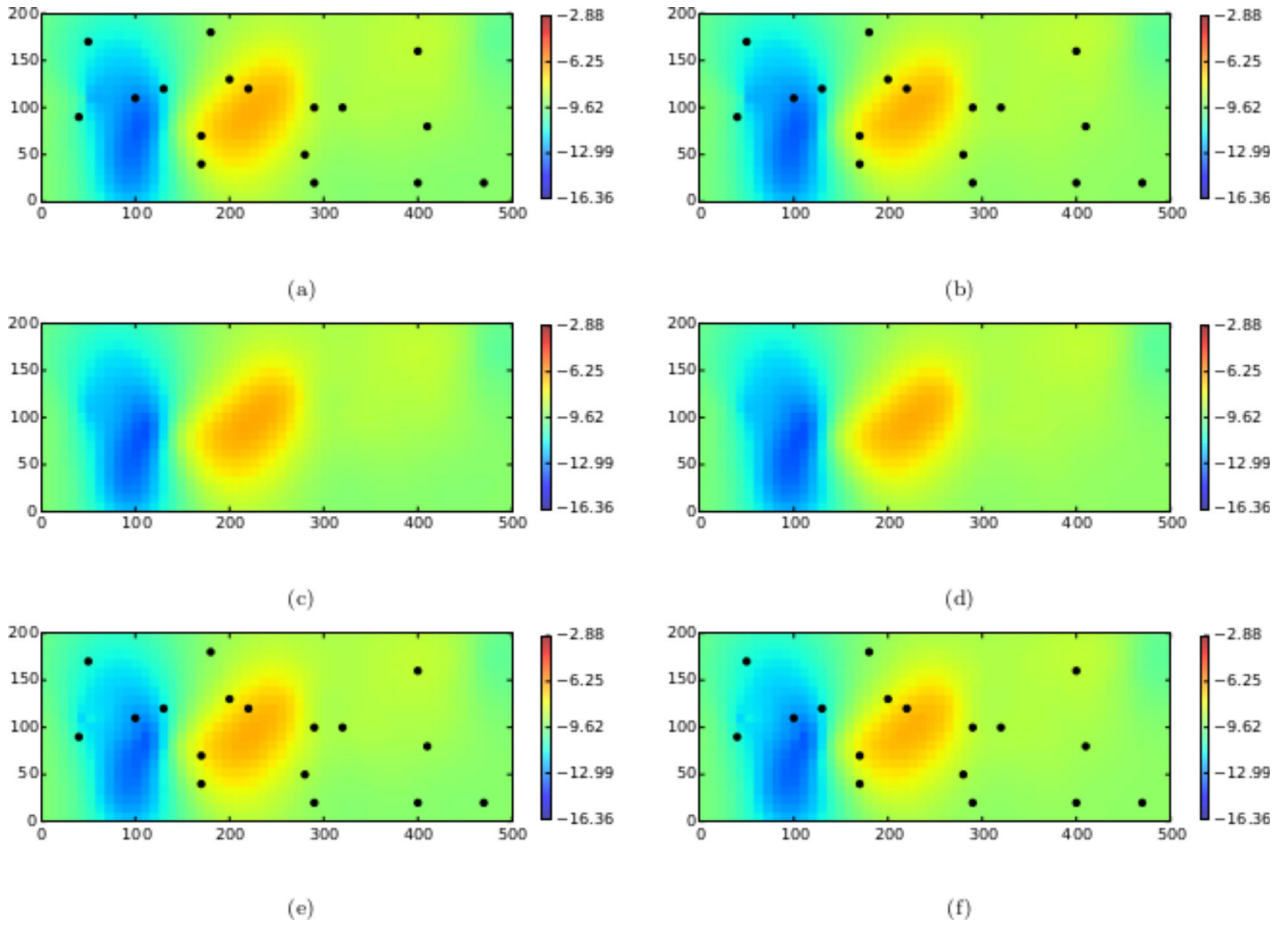

Figure 11. The inversion results of the (a) top, (c) middle, and (e) bottom layers of the 3-D model using the reference method of "LM-QR," and those of the (b) top, (d) middle, and (f) bottom layers obtained using our "LM-RLSQR" method. Both methods yield comparable results. However, "LM-RLSQR" is much more computationally efficient than "LM-QR."

example, using adjoint methods or analytical solutions). On the other hand, as for applications where the calculation of Jacobian matrix is the dominant computational cost, other computational techniques such as model reduction or Jacobian-free methods may be more effective. In general, our method has great

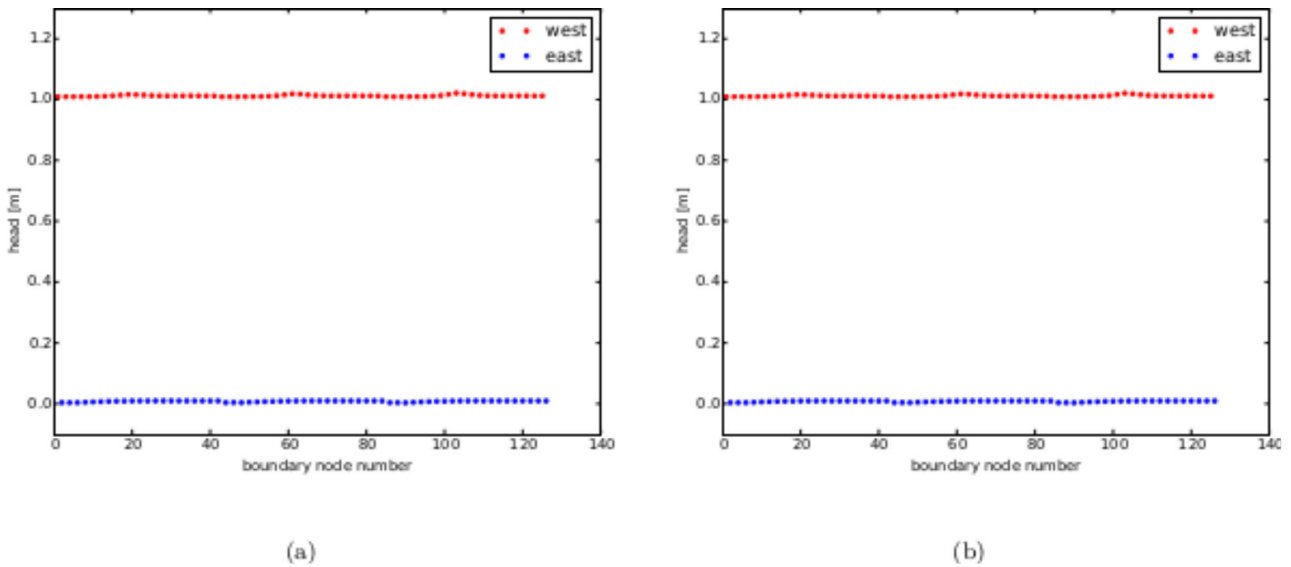

Figure 12. The inversion results of the boundary conditions using (a) "LM-QR" method and our (b) "LM-RLSQR" method. Both the methods yield similar results.

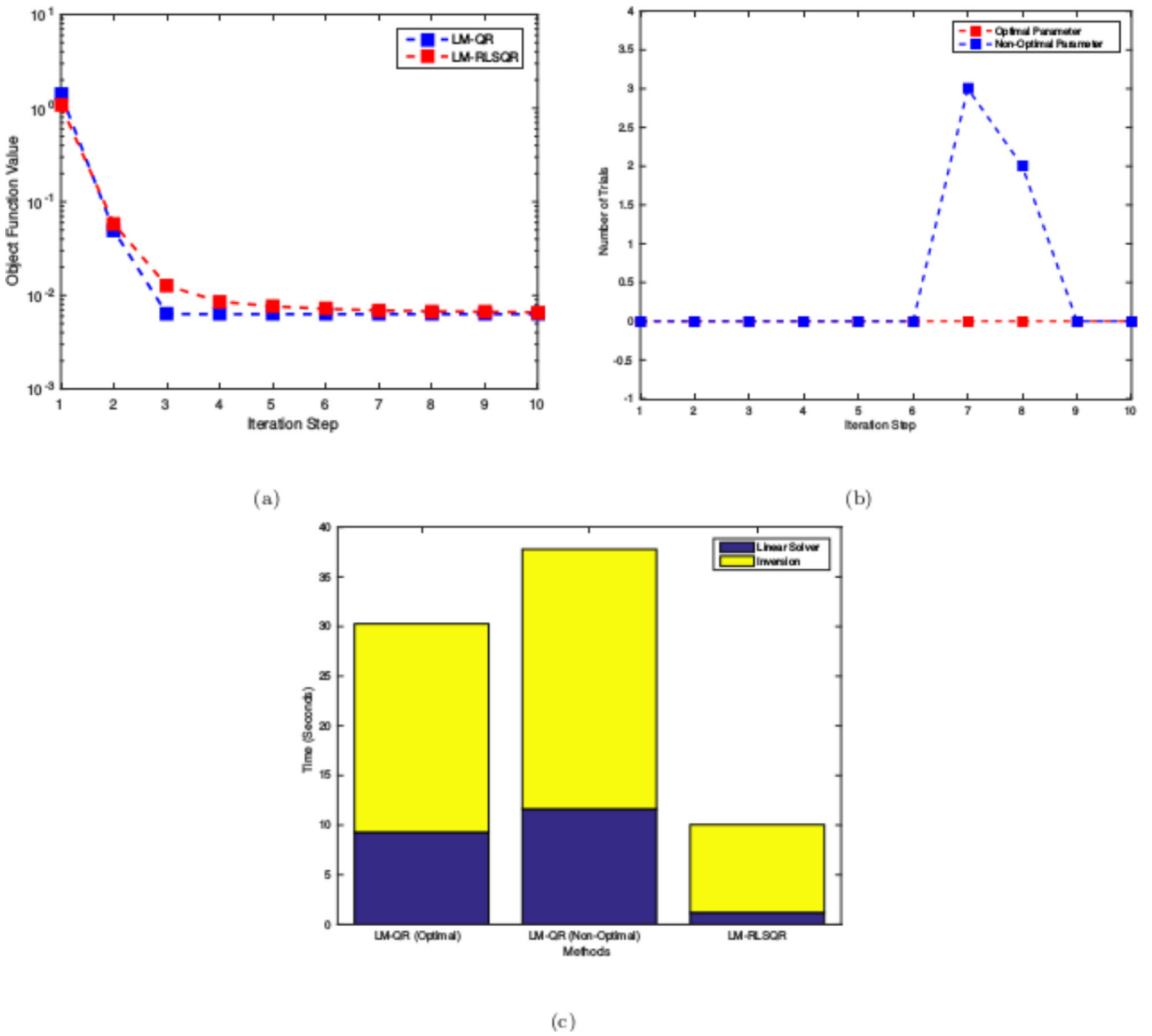

Figure 13. (a) A comparison of the convergence of the classical sequential LM algorithm versus our "LM-RLSQR" method. (b) A comparison of the number of the trials for finding the damping values using optimal versus nonoptimal parameter sets. (c) The time profiles for both the linear solver and the inversion using classical sequential LM method with and without optimal parameter sets as well as our parallel LM method. Our method has the least computational time costs for both the linear solver and the inversion compared to the classical sequential LM method with and without optimal parameter sets.

potential to characterize subsurface heterogeneity for moderate or even large-scale problems with a large number of model parameters where the Jacobian can be computed efficiently.

Appendix A: Residual and Gradient of Generalized Least Squares Problems

We rewrite equation (5) as

$$\mathbf{m} = \underset{\mathbf{m}}{\operatorname{argmin}} \{l(\mathbf{m})\} = \underset{\mathbf{m}}{\operatorname{argmin}} \{||r_l(\mathbf{m})||_2^2\} = \left|\left|\begin{bmatrix}\mathbf{d}-f(\mathbf{m})\\ \sqrt{\lambda}\,\mathbf{m}\end{bmatrix}\right|\right|_2^2, \tag{A1}$$

and the residual vector $r_l(\mathbf{m})$ is defined as

$$r_l(\mathbf{m}) = \begin{bmatrix} \mathbf{d} - f(\mathbf{m}) \\ \sqrt{\lambda}\,\mathbf{m} \end{bmatrix}. \tag{A2}$$

Based on the residual vector in equation (A2), we derive the Jacobian matrix

$$J_l = \left[\frac{\partial (r_l)_i}{\partial m_j}\right]_{i=1\ldots\tilde{n}, j=1\ldots\tilde{m}} = \begin{bmatrix} \nabla (r_l)_1^T \\ \nabla (r_l)_2^T \\ \vdots \\ \nabla (r_l)_{\tilde{n}}^T \end{bmatrix}, \tag{A3}$$

where $\tilde{m}$ is the number of model parameter and $\tilde{n}$ is the number of observations. The gradient of equation (A1) can be written as

$$\mathrm{Grad_l} = -J_l^T\, r_l(\mathbf{m}). \tag{A4}$$

Similarly, we rewrite the generalized least squares form in equation (6)

$$\mathbf{m} = \arg\min_{\mathbf{m}} \{g(\mathbf{m})\} = \arg\min_{\mathbf{m}} \{||r_g(\mathbf{m})||_2^2\} = \left|\left| \begin{bmatrix} R^{-\frac{1}{2}} (\mathbf{d} - f(\mathbf{m})) \\ \sqrt{\lambda}\, Q^{-\frac{1}{2}} (\mathbf{m} - \mathbf{m_0}) \end{bmatrix} \right|\right|_2^2, \tag{A5}$$

where the matrices R and Q are defined according to equations (7) and (8). The residual vector of the generalized least squares can be defined as

$$r_g(\mathbf{m}) = \begin{bmatrix} R^{-\frac{1}{2}} (\mathbf{d} - f(\mathbf{m})) \\ \sqrt{\lambda}\, Q^{-\frac{1}{2}} (\mathbf{m} - \mathbf{m_0}) \end{bmatrix}, \tag{A6}$$

and the Jacobian matrix J_g of equation (A5) is

$$J_g = \left[\frac{\partial (r_g)_i}{\partial m_j}\right]_{i=1\ldots\tilde{n}, j=1\ldots\tilde{m}} = \begin{bmatrix} \nabla (r_g)_1^T \\ \nabla (r_g)_2^T \\ \vdots \\ \nabla (r_g)_{\tilde{n}}^T \end{bmatrix}. \tag{A7}$$

Hence, the gradient of the generalized least squares problem in equation (A5) has the same form as that of the ordinary least squares problem

$$\mathrm{Grad_g} = -J_g^T\, r_g(\mathbf{m}), \tag{A8}$$

where the Jacobian matrix J_g and the residual vector r_g are given in equations (A6) and (A7), respectively.

Without loss of generality, we pose the gradient of the inverse modeling as a general form of

$$\mathrm{Grad} = -J^T\, r(\mathbf{m}), \tag{A9}$$

where the Jacobian matrix J and residual vector $r(\mathbf{m})$ are further defined according to the specific least squares formulation utilized.

Appendix B: Krylov Subspace Approximation Using Golub-Kahan-Lanczos Bidiagonalization

Solving for the search direction of $\mathbf{p}^{(k)}$ in equation (14) or (15) is a typical linear system solver. To describe the technique of Krylov subspace without loss of generality, in this appendix we provide with a general linear least squares problem

$$\min_{\mathbf{m}} ||J\,\mathbf{m} - \mathbf{r}||_2^2. \tag{B1}$$

The Krylov subspace is usually defined as

$$\mathcal{K}_k = \mathcal{K}_k(J'J, J'\mathbf{r}), \tag{B2}$$

where k is the dimension of the subspace. For most of the applications, $k \ll \text{rank}(J)$.

The Golub-Kahan-Lanczos (GKL) bidiagonalization technique provides a method to obtain the basis to span the Krylov subspace in equation (B2). The GKL bidiagonalization is an iterative procedure. At every iteration, the current most significant basis will be obtained. The iteration stops when the subspace reaches a good approximation to the original space. Here we provide the major procedures of the GKL bidiagonalization.

We start the GKL procedure with the right-hand side vector $\mathbf{r}$

$$\beta^{(1)}\mathbf{u}^{(1)} = \mathbf{r}, \quad \alpha^{(1)}\mathbf{v}^{(1)} = J'\mathbf{u}^{(1)}, \tag{B3}$$

where $||\mathbf{u}^{(1)}||_2 = ||\mathbf{v}^{(1)}||_2 = 1$, and for $j = 1, 2, \ldots$, we take

$$\begin{cases} \beta^{(j+1)}\mathbf{u}^{(j+1)} = J\mathbf{v}^{(j)} - \alpha^{(j)}\mathbf{u}^{(j)}, \\ \alpha^{(j+1)}\mathbf{v}^{(j+1)} = J'\mathbf{u}^{(j+1)} - \beta^{(j+1)}\mathbf{v}^{(j)}, \end{cases} \tag{B4}$$

where $\alpha^{(j+1)} \geq 0$ and $\beta^{(j+1)} \geq 0$.

After k steps of the recursion in equations (B3) and (B4), we can decompose the matrix J into three matrices: $U^{(k+1)}$, $B^{(k)}$, and $V^{(k)}$

$$V^{(k)} = \left[\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \ldots, \mathbf{v}^{(k)}\right], \quad U^{(k+1)} = \left[\mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \ldots, \mathbf{u}^{(k+1)}\right], \tag{B5}$$

$$B^{(k)} = \begin{bmatrix} \alpha^{(1)} & & & \\ \beta^{(2)} & \alpha^{(2)} & & \\ & \beta^{(3)} & \ddots & \\ & & \ddots & \alpha^{(k)} \\ & & & \beta^{(k+1)} \end{bmatrix}. \tag{B6}$$

These three matrices satisfy

$$\beta^{(1)}U^{(k+1)}\mathbf{e}^{(1)} = \mathbf{r}, \tag{B7}$$

$$JV^{(k)} = U^{(k+1)}B^{(k)}, \tag{B8}$$

$$J'U^{(k+1)} = V^{(k)}(B')^{(k)} + \alpha^{(k+1)}V^{(k+1)}\mathbf{e}'(k+1), \tag{B9}$$

where the unit vector $\mathbf{e}^{(i)}$ has value 1 at the ith location and zeros elsewhere, i.e., $\mathbf{e}^{(i)} = [0, \ldots, 1, \ldots 0]$.

The basis of the Krylov subspace in equation (B2) is now given by the column vectors in V_k [*Paige and Saunders*, 1982a,1982b]

$$\mathcal{K}_k = \mathcal{K}_k(J'J, J'\mathbf{r}) = \text{span}(V^{(k)}). \tag{B10}$$

Once the Krylov subspace is generated, we can project the original problem in equation (B1) down to the subspace generated by $\text{span}(V^{(k)})$ and yield an approximated linear least squares problem

$$\min_{\mathbf{y}^{(k)}} ||B^{(k)}\mathbf{y}^{(k)} - \beta^{(1)}\mathbf{e}^{(1)}||_2^2, \tag{B11}$$

where the approximated solution $\mathbf{y}^{(k)}$ satisfying

$$\mathbf{m}^{(k)} = V^{(k)}\mathbf{y}^{(k)}. \tag{B12}$$

The projected problem in equation (B11) usually yields a good approximation to the original one in equation (B1) with much lower dimension. Hence, solving equation (B11) for the approximate solution $\mathbf{y}_k$ is computationally much more efficient than solving the original problem in equation (B1). The matrix of B_k is $(k+1)\times k$, which is much smaller than the original problem size. Instead of solving the projected least squares problem in equation (B11) using direct methods, *Paige and Saunders* [1982a,1982b] developed a

three-term-recurrences formulation to obtain the solution of the original problem in equation (B1) directly. Specifically, applying the QR decomposition of $B^{(k)}$ we have

$$(Q^{(k)})'B^{(k)} = \begin{bmatrix} R^{(k)} \\ 0 \end{bmatrix}, \quad (Q^{(k)})'(\beta^{(1)}\mathbf{e}^{(1)}) = \begin{bmatrix} \mathbf{f}^{(k)} \\ \bar{\phi}^{(k+1)} \end{bmatrix}, \tag{B13}$$

where $R^{(k)}$ and $\mathbf{f}^{(k)}$ are

$$R^{(k)} = \begin{bmatrix} \rho^{(1)} & \theta^{(1)} & & & \\ & \rho^{(2)} & \theta^{(2)} & & \\ & & \ddots & \ddots & \\ & & & \rho^{(k-1)} & \theta^{(k-1)} \\ & & & & \rho^{(k)} \end{bmatrix}, \quad \mathbf{f}^{(k)} = \begin{bmatrix} \phi^{(1)} \\ \phi^{(2)} \\ \vdots \\ \phi^{(k-1)} \\ \phi^{(k)} \end{bmatrix}. \tag{B14}$$

Therefore, according to *Paige and Saunders* [1982a,1982b], the three-term recursion to update the solution $\mathbf{m}^{(k)}$ at each iteration step can be obtained

$$\mathbf{m}^{(k)} = \mathbf{m}^{(k-1)} + \phi^{(k)}\mathbf{z}^{(k)}, \quad \mathbf{z}^{(k)} = \frac{1}{\rho^{(k)}}(\mathbf{v}^{(k)} - \theta^{(k-1)}\mathbf{z}^{(k-1)}). \tag{B15}$$

The major computational cost of generating the Krylov subspace using the GKL bidiagonalization technique to solve the linear system in equation (B1) is the recursion procedure in equations (B3) and (B4). The three-term-recursion procedure to update the solution in equation (B13) to equation (B15) is comparatively computationally cheap.

Appendix C: Givens Rotations for Augmented Least Squares Problems

To solve the projected problem in equation (18), we can employ two Givens rotations at each iteration to eliminate two elements: one is the lower off-diagonal element in $B^{(k)}$ and the other one is the diagonal element in the identity matrix I. A schematic illustration of this procedure is shown in Figure 1. Specifically, provided with a Givens matrix $G_{i,j}$ for the ith row and jth row, at each iteration, we will have

$$G_{1,2}\, G_{1,k+2} \begin{bmatrix} B^{(k)} \\ \sqrt{\mu} I \end{bmatrix}. \tag{C1}$$

Because of the orthogonality of the Givens matrix, we can transform the least squares problem in equation (18) into an equivalent problem

$$\min_{\mathbf{y}^{(k)}} \left\{ \left\| G_{1,2}\, G_{1,k+2} \left(\begin{bmatrix} B^{(k)} \\ \sqrt{\mu} I \end{bmatrix} \mathbf{y}^{(k)} - \beta^{(1)}\mathbf{e}^{(1)} \right) \right\|_2 \right\}. \tag{C2}$$

We perform k Givens' rotation steps

$$\min_{\mathbf{y}^{(k)}} \left\{ \left\| G_{k,k+1}\, G_{k,2k+1} \cdots G_{1,2}\, G_{1,k+2} \left(\begin{bmatrix} B^{(k)} \\ \sqrt{\mu} I \end{bmatrix} \mathbf{y}^{(k)} - \beta^{(1)}\mathbf{e}^{(1)} \right) \right\|_2 \right\}, \tag{C3}$$

which yields an equivalent least squares problem

$$\min_{\mathbf{y}^{(k)}} \left\{ \left\| \left(\begin{bmatrix} (R^{(k)})' \\ 0 \end{bmatrix} \mathbf{y}^{(k)} - \begin{bmatrix} (\mathbf{f}^{(k)})' \\ (\phi^{(k)})' \end{bmatrix} \right) \right\|_2 \right\}. \tag{C4}$$

It is worth mentioning that the upper bidiagonal matrix $(R^{(k)})'$ in equation (C4) now contains the damping parameter of μ.

As an example, we provide all the details of the matrix operations based on a simple 3×2 matrix $B^{(k)}$, 2×2 identity matrix I, and a scalar $\mu = 1$. Let

$$\tilde{B} = \begin{bmatrix} B^{(k)} \\ \sqrt{\mu} I \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 2 & 3 \\ 0 & 4 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{C5}$$

We employ a rotation matrix to $\tilde{B}$

$$G_{1,4} = \begin{bmatrix} 0.71 & 0 & 0 & 0.71 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ -0.71 & 0 & 0 & 0.71 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{C6}$$

such that the element $\tilde{B}(4,1)$ can be eliminated

$$G_{1,4} \begin{bmatrix} B^{(k)} \\ \sqrt{\mu} I \end{bmatrix} = \begin{bmatrix} \mathbf{1.4} & 0 \\ 2 & 3 \\ 0 & 4 \\ \mathbf{0} & 0 \\ 0 & 1 \end{bmatrix}, \tag{C7}$$

where the modified elements are in bold.

In order to transform the resulting matrix into an upper bidiagonal form, we need to eliminate the element of $\tilde{B}(2,1)$ by another Givens rotation

$$G_{1,2} = \begin{bmatrix} 0.58 & 0.82 & 0 & 0 & 0 \\ -0.82 & 0.58 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{C8}$$

$$G_{1,2}\ G_{1,4} \begin{bmatrix} B^{(k)} \\ \sqrt{\mu} I \end{bmatrix} = \begin{bmatrix} \mathbf{2.4} & \mathbf{2.4} \\ \mathbf{0} & \mathbf{1.7} \\ 0 & 4 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}. \tag{C9}$$

Observed from equation (C9), the first column of the matrix $\tilde{B}$ has been rotated into an upper bidiagonal matrix with the lower identity matrix element eliminated. Based on a similar idea, we can further rotate the second column of $\tilde{B}$. As the operations proceed, we will transform the matrix $\tilde{B}$ into an upper bidiagonal matrix

$$\tilde{B} \rightarrow \begin{bmatrix} 2.4 & 2.4 \\ 0 & 4.5 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}. \tag{C10}$$

Appendix D: Levenberg-Marquardt With Paralleled Recycled LSQR Algorithm

In this appendix, we provide a detailed description to summarize our new Levenberg-Marquardt algorithm based on Krylov subspace approximation and parallel selection of the damping parameters.

Algorithm 1 Levenberg-Marquardt with Paralleled Recycled LSQR Algorithm (LM-RLSQR)

Require: $\mathbf{m}^{(0)}$, $k_{\max}$;

Ensure: $\mathbf{m}^{(k)}$

1: Initialize *found*=false, *Jacobian*=true;

2: Initialize the damping parameter, μ, according to equation (25);

3: Transform the problem according to equation (22), if desired;

4: **while** {(**not** *found*) **and** $(k < k_{\max})$} **do**

5: **for** i=0 TO n-1 **do** % This is a parallel-for loop in a multi-core computing environment

6: Generate the*i*th damping parameter,$\mu_i^{(k)}$;

7: if {i = 0} then

8: Calculate the new Jacobian matrix according to the adjoint-state method in *Strang* [2007];

9: Generate the Krylov subspace according to equations (B3) and (B4), and then project the problem down to the Krylov subspace using equation (18);

10: Solve for $\overline{\mathbf{p}}^{(k)}$ using the first damping parameter of $\mu_0^{(k)}$ according to equation (20);

11: **else**

12: Solve for $\overline{\mathbf{p}}^{(k)}$ in the generated subspace using $\mu_i^{(k)}$ according to equation (20);

13: **end if**

14: **end for**

15: **if** {Stopping criterion are satisfied} **then**

16: *found*=true;

17: Return with solution$\mathbf{m}^{(k)}$;

18: **else**

19: Transform the solution using equation (24);

20: Evaluate the objective function values for each search direction, $\mathbf{p_i}^{(k)}$ (this is implemented in parallel);

21: Use the best $\mathbf{p_j}^{(k)}$ to update $\mathbf{m}^{(k+1)}=\mathbf{m}^{(k)}+\mathbf{p_j}^{(k)}$;

22: Update the seed damping parameter $\mu_j^{(k)}$ according to equation (F2);

23: $k \leftarrow k+1$;

24: **end if**

25: **end while**

Appendix E: Computational Cost Analysis

To better understand the cost of our new Levenberg-Marquardt algorithm, we provide a computational cost analysis of our method. Considering most of the numerical operations in Algorithm 1 involve only matrix and vector operations, we use the floating point operations per second (FLOPS) and the big-$\mathcal{O}$ notation to quantify the computational cost [*Golub and Van Loan*, 1996]. In numerical linear algebra, BLAS operations are categorized into three levels. Level-1 operations involve an amount of data and arithmetic that is

linear in the dimension of the operation. Those operations involving a quadratic amount of data and a quadratic amount of work are Level-2 operations [*Golub and Van Loan*, 1996]. Following this notation, vector dot-product, addition and subtraction are examples of BLAS Level-1 operations (BLAS 1). Matrix-vector multiplication is a BLAS Level-2 operation (BLAS 2).

To set up the problem, we assume that the number of model parameter is $\tilde{m}$, the number of observations is $\tilde{n}$, hence the size of the Jacobian matrix $\tilde{n}\times\tilde{m}$. As we mentioned earlier, we employ the adjoint-state method [*Custodio et al*., 2012; *Strang*, 2007] to calculate the Jacobian matrix and only focus on the calculation of the search direction.

E1. Computational Costs of Algorithm 1

Provided with a Jacobian matrix, the computational costs of our method at each iteration step come from two parts: the cost of solving the linear system with the first damping parameter and the cost of solving the linear system with the rest of the damping parameters. The most expensive part in solving for the search direction with the first damping parameter is equations (B3) and (B4), where three matrix-vector multiplications are involved. Assume the dimension of the Krylov subspace is k_1, the costs associated with equations (B3), (B4), and (20) are

$$\mathrm{COST}_{LM-RLSQR-1} \approx k_1 \cdot \mathcal{O}(\tilde{m}\times\tilde{n}). \tag{E1}$$

The computational cost of solving for the search directions of the rest of the damping parameters only come from equation (20), which is

$$\mathrm{COST}_{LM-RLSQR-Rest} \approx k_1(n-1)\cdot\mathcal{O}(\tilde{m}), \tag{E2}$$

where n is the number of μ values that are being used.Therefore, the total computational costs of Algorithm 1 at every iteration are

$$\begin{aligned}\mathrm{COST}_{LM-RLSQR} &= \mathrm{COST}_{LM-RLSQR-1}+\mathrm{COST}_{LM-RLSQR-Rest},\\ &\approx k_1(\mathcal{O}(\tilde{m}\times\tilde{n})+(n-1)\mathcal{O}(\tilde{m})).\end{aligned} \tag{E3}$$

Observing the total cost in equation (E3), the costs associated with the rest of the damping parameters are much smaller in comparison to the costs with the initial damping parameter, since typically $n \ll \tilde{n}$. In practice, we often use $n = 10$, whereas $\tilde{n}$ is often hundreds, thousands, or even hundreds of thousands.

E2. Computational Costs of Classical Levenberg-Marquardt Method

To have a comprehensive comparison, we also provide the computational costs of the sequential Levenberg-Marquardt method based on the QR factorization as the linear solver (LM-QR).

Again, given an initial guess of the damping parameter, the computational costs associated is

$$\mathrm{COST}_{LM-QR-1} \approx \mathcal{O}(\tilde{n}\times\tilde{m}^2)+\mathcal{O}(\tilde{m}^3)+\mathcal{O}(\tilde{n}\times\tilde{m})+\mathcal{O}(\tilde{m}^2), \tag{E4}$$

where the first term is associated with forming of the normal equation, the second term is associated with the QR factorization, the third term is associated with forming the right-hand side, and the fourth term is associated with the back-substitution for the solution.

Once the damping parameter is updated, some parts of the normal equation in equation (11) or equation (12) can be saved and reused. However, the expensive QR factorization and the back-substitution cannot be avoided, therefore the costs for the updated damping parameter will be

$$\mathrm{COST}_{LM-QR-Rest} \approx \mathcal{O}(\tilde{m}^3)+\mathcal{O}(\tilde{n}\times\tilde{m})+\mathcal{O}(\tilde{m}^2). \tag{E5}$$

Hence, the total costs of sequential LM-QR at every iteration are

$$\begin{aligned}\mathrm{COST} &= \mathrm{COST}_{LM-QR-1}+\mathrm{COST}_{LM-QR-Rest},\\ &\approx \mathcal{O}(\tilde{n}\times\tilde{m}^2)+k_2(\mathcal{O}(\tilde{m}^3)+\mathcal{O}(\tilde{n}\times\tilde{m})+\mathcal{O}(\tilde{m}^2)),\end{aligned} \tag{E6}$$

where k_2 is the number of damping parameters that are being used.

By comparing the computational costs of our method in equation (E3) to the costs of the classical LM method in equation (E6), one sees that our method significantly reduces the computational costs in the "linear solver" operation.

E3. Computational Costs Comparison to Other Krylov Subspace Solver

In this section, we compare the computational costs of employing our method to that of other commonly used Krylov subspace solvers in solving the Marquardt's formulation in equation (12). We select the Bi-Conjugate Gradient method (BiCG) method [*Saad*, 2003] as the reference method. The BiCG iterative method is designed for nonsymmetric systems. If the system becomes symmetric, the BiCG method will collapse to classical Conjugate Gradient (CG) method [*Watkins*, 2010].

The classical Marquardt's formulation in equation (12) can be written equivalently in the normal equation

$$\min_{\mathbf{p}^{(k)}} \{||J^{(k)}\mathbf{p}^{(k)}-(-r^{(k)})||_2^2+\mu||D\mathbf{p}^{(k)}||_2^2\} \quad \rightarrow \left((J^{(k)})^T J^{(k)}+\mu D^T D\right)\mathbf{p}^{(k)}=-(J^{(k)})^T r^{(k)}, \quad \rightarrow \tilde{A}^{(k)}\,\mathbf{p}^{(k)}=\tilde{\mathbf{b}}^{(k)}, \tag{E7}$$

where the system matrix $\tilde{A}^{(k)}=\left((J^{(k)})^T J^{(k)}+\mu D^T D\right)$ and the right-hand side vector $\tilde{\mathbf{b}}^{(k)}=-(J^{(k)})^T r^{(k)}$.

The computational cost of employing BiCG iterative method to solve the linear system in equation (E7) is dominated by the matrix-vector multiplication with system $\tilde{A}^{(k)}$. However, rather than explicitly constructing the system $\tilde{A}^{(k)}$, we decompose the matrix-vector multiplication with system $\tilde{A}^{(k)}$ into nested matrix-vector multiplications, i.e., one with $J^{(k)}$ and the other with $(J^{(k)})^T$. Following this, the computational complexity can be reduced from $\mathcal{O}(\tilde{n}^3)$ to $\mathcal{O}(\tilde{n}^2)$. Therefore, the overall computational cost of employing BiCG iterative solver to the classical Marquardt's formulation is

$$\mathrm{COST}_{BiCG} \approx n \cdot k_2 \cdot \mathcal{O}(\tilde{m}\times\tilde{n}), \tag{E8}$$

where n is the number of damping parameters and k_2 is the dimension of the Krylov subspace.

On the other hand, if we transform the classical Marquardt's formulation to the standard form shown in equation (23), the GKL bidiagonalization can be employed to the transformed Jacobian matrix $\bar{J}^{(k)}$ instead of to the augmented system matrix. The resulting computational cost has been provided in equation (E3).

Comparing equation (E3) to equation (E8), the cost in equation (E3) is much smaller. The cost has been reduced by an order of magnitude for the remaining trials of the damping parameter because of the subspace recycling.

E4. Computational Cost Comparison to "SVD-Assist" Functionality in PEST

Another type of subspace-recycling technique named "SVD-Assist" is developed in *Tonkin and Doherty* [2005] and incorporated in the software package PEST [*Doherty*, 2015]. In this section, we provide some comparisons of our method to the "SVD-Assist" in PEST.

The major commonality of "SVD-Assist" and our method is that both methods are based on subspace approximation and recycling. Some substantial differences between these two techniques exist both conceptually and computationally. The idea of the "SVD-Assist" is to construct a few "super" parameters as basis vectors to form a subspace such that the model in the original space can be projected down to the new subspace. The approach to obtain the "super" parameters is based on the SVD decomposition of the weighted Jacobian matrix.

Conceptually, "SVD-Assist" is based on a linearity assumption that apparently will be violated for the nonlinear models. Hydrogeological models typically fall in the nonlinear category. To avoid this limitation, "SVD-Assist" methodology employs periodic recalculations of the super parameters during the optimization process [*Doherty and Hunt*, 2010, p. 6]. However, in "SVD-Assist" there is no theoretical guidance to define when the recalculations of the "super" parameters is needed. By contrast, our method does not impose any linearity assumption. Furthermore, our method realizes the separation of the damping parameters from the solution subspace, while "SVD-Assist" does not.

Computationally, the costs of constructing the subspace using "SVD-Assist" and our method can be also different. "SVD-Assist" can apply two schemes to obtain the subspace basis, the singular vectors, according to the PEST User Manual [*Doherty*, 2015]. One option is based on the direct SVD decomposition, and the other

option is to employ LSQR, i.e., to reformulate the SVD as a two-step decomposition: a bidiagonalization on the Jacobian matrix followed with a SVD decomposition on the bidiagonal matrix [*Golub and Kahan*, 1965]. The computational costs associated with these two schemes of direct SVD and LSQR based are $\mathcal{O}(\tilde{m}^3)$ and $\mathcal{O}(\tilde{m}^2)$, respectively. The SVD decomposition is the default and generally more frequently used option. On the other hand, the cost of our method is $\mathcal{O}(\tilde{m}^2)$, consistently. Hence, the computational cost of our method is either better or comparable to "SVD-Assist."

One of the benefits of "SVD-Assist" is that a full Jacobian matrix is only needed when the super parameters are recalculated. Our method relies on the use of fast methods for computing the Jacobian matrix. "SVD-Assist" provides an advantage when these methods are not available, because it does not require them. When the computation of the Jacobian is the performance bottleneck (e.g., if finite difference methods are being used to compute the Jacobian for an expensive model), our method will provide little benefit while "SVD-Assist" may be helpful.

Appendix F: Selection of the Damping Parameter in Levenberg-Marquardt Algorithm

Marquardt [1963] developed a popular updating scheme. Assuming an approximation $L(\mathbf{h})$ to $r(\mathbf{m}+\mathbf{h})$ with a small step of $\mathbf{h}$, a quantity called the gain factor can be defined as

$$\rho=\frac{r(\mathbf{m})-r(\mathbf{m}+\mathbf{h})}{L(\mathbf{0})-L(\mathbf{h})}. \tag{F1}$$

Based on the value of ρ, we update the damping parameter via

$$u(x) = \begin{cases} \mu=\beta\cdot\mu & \text{if } \rho<\rho_1 \\ \mu=\dfrac{\mu}{\gamma} & \text{if } \rho>\rho_2 \\ \mu=\mu & \text{otherwise,} \end{cases} \tag{F2}$$

where $0<\rho_1<\rho_2<1$ and $\beta,\gamma>1$, specifically ρ_1=0.25, ρ_2=0.75, $\beta=2$, and $\gamma=3$ are good values in many cases [*Nielsen*, 1999].

Acknowledgments
Youzuo Lin and Velimir V. Vesselinov were support by Los Alamos National Laboratory Environmental Programs Projects. Daniel O'Malley was supported by a Los Alamos National Laboratory (LANL) Director's Postdoctoral Fellowship. In addition, Velimir V. Vesselinov was supported by the DiaMonD project (An Integrated Multifaceted Approach to Mathematics at the Interfaces of Data, Models, and Decisions, U.S. Department of Energy Office of Science, grant 11145687). We also thank Jeremy T. White, two anonymous reviewers, and the Associate Editor for their valuable comments that help improve our paper. All the data are available from the authors upon request (ylin@lanl.gov).

References

Araneda, E. (2004), A variation of the Levenberg-Marquardt method: An attempt to improve efficiency, MS thesis, Mass. Inst. of Technol., Cambridge.
Barajas-Solano, D. A., B. E. Wohlberg, V. V. Vesselinov, and D. M. Tartakovsky (2014), Linear functional minimization for inverse modeling, *Water Resour. Res.*, *51*, 4516–4531, doi:10.1002/2014WR016179.
Bertsekas, D. P. (1999), *Nonlinear Programming*, Athena Scientific, Belmont, Mass.
Bezanson, J., A. Edelman, S. Karpinski, and V. B. Shah (2014), Julia: A fresh approach to numerical computing, Julia Computing Inc, Allston, Mass. [Available at http://http://julialang.org/.]
Carrera, J., and S. P. Neuman (1986), Estimation of aquifer parameters under transient and steady state conditions: 1. Maximum likelihood method incorporating prior information, *Water Resour. Res.*, *22*, 199–210.
Coleman, T. F., and P. E. Plassmann (1992), A parallel nonlinear least-squares solver: Theoretical analysis and numerical results, *SIAM J. Sci. Stat. Comput.*, *13*(3), 771–793.
Constantine, P. G., E. Dow, and Q. Wang (2014), Active subspace methods in theory and practice: Applications to kriging surfaces, *SIAM J. Sci. Comput.*, *36*(4), A1500–A1524.
Cooley, R. L. (1985), A comparison of several methods of solving nonlinear regression groundwater flow problems, *Water Resour. Res.*, *21*(10), 1525–1538.
Custodio, E., A. Gurgui, and J. P. Lobo Ferreira (2012), *Groundwater Flow and Quality Modelling*, Springer, N. Y.
Doherty, J. (2015), *Calibration and Uncertainty Analysis for Complex Environmental Models*, Watermark Numer. Comput., Brisbane, Queensland, Australia. [Available at http://www.pesthomepage.org.]
Doherty, J., and R. J. Hunt (2010), Approaches to highly parameterized inversion: A guide to using PEST for groundwater-model calibration, *U.S. Geol. Surv. Sci. Invest. Rep.*, *2010-5169*, 70 pp.
Engl, H. W., M. Hanke, and A. Neubauer (1996), *Regularization of Inverse Problems*, Kluwer Acad, Dordrecht, Netherlands.
Fienen, M., R. Hunt, D. Krabbenhoft, and T. Clemo (2009), Obtaining parsimonious hydraulic conductivity fields using head and transport observations: A Bayesian geostatistical parameter estimation approach, *Water Resour. Res.*, *45*, W08405, doi:10.1029/2008WR007431.
Finsterle, S., and M. B. Kowalsky (2011), A truncated Levenberg-Marquardt algorithm for the calibration of highly parameterized nonlinear models, *Comput. Geosci.*, *37*, 731–738.
Golub, G., and W. Kahan (1965), Calculating the singular values and pseudo-inverse of a matrix, *J. Soc. Ind. Appl. Math.*, *2*(2), 205–224.
Golub, G. H., and C. F. Van Loan (1996), *Matrix Computations*, 3rd ed., Johns Hopkins Univ. Press, Baltimore, Md.

Hansen, P. C. (1998), *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion*, Soc. for Ind. and Appl. Math, Philadelphia, Pa.
Hunt, R. J., J. Doherty, and M. J. Tonkin (2007), Are models too simple? Arguments for increased parameterization, *Groundwater*, *45*, 254–262.
Jefferson, J. L., J. M. Gilbert, P. G. Constantine, and R. M. Maxwell (2015), Active subspaces for sensitivity analysis and dimension reduction of an integrated hydrologic model, *Comput. Geosci.*, *83*, 127–138.
Kitanidis, P. K., and J. Lee (2014), Principal component geostatistical approach for large-dimensional inverse problem, *Water Resour. Res.*, *50*, 5428–5443.
Lampton, M. (2009), Damping-undamping strategies for the Levenberg Marquardt nonlinear least squares method, *Comput. Phys.*, *11*, 110–115.
Lee, J., and P. K. Kitanidis (2014), Large-scale hydraulic tomography and joint inversion of head and tracer data using the principal component geostatistical approach (PCGA), *Water Resour. Res.*, *50*, 5410–5427.
Levenberg, K. (1944), A method for the solution of certain non-linear problems in least squares, *Q. Appl. Math.*, *2*, 164–168.
Liu, C., and P. Ball (1999), Application of inverse methods to contaminant source identification from aquitard diffusion profiles at Dover AFB, *Delaware, Water Resour. Res.*, *35*, 1975–1985.
Liu, X., Q. Zhou, J. T. Birkholzer, and W. A. Illman (2013), Geostatistical reduced-order models in underdetermined inverse problems, *Water Resour. Res.*, *59*, 6587–6600, doi:10.1002/wrcr.20489.
Liu, X., Q. Zhou, P. K. Kitanidis, and J. T. Birkholzer (2014), Fast iterative implementation of large-scale nonlinear geostatistical inverse modeling, *Water Resour. Res.*, *50*, 198–207, doi:10.1002/2012WR013241.
Madsen, K., H. B. Nielsen, and O. Tingleff (2004), Methods for non-linear least squares problems, 2nd ed., technical report, Tech. Univ. of Denmark, Kongens Lyngby, Denmark.
Marquardt, D. (1963), An algorithm for least-squares estimation of nonlinear parameters, *SIAM J. Appl. Math.*, *11*, 431–441.
Naveen, M., S. Jayaraman, V. Ramanath, and S. Chaudhuri (2010), Modified Levenberg Marquardt algorithm for inverse problems, *Simulated Evol. Learning*, *6457*, 623–632.
Nielsen, H. B. (1999), Damping parameter in Marquardt's method, technical report, Tech. Univ. of Denmark, Kongens Lyngby, Denmark.
Nocedal, J., and S. Wright (2000), *Numerical Optimization*, Springer, N. Y.
Nowak, W., and O. A. Cirpka (2004), A modified Levenberg-Marquardt algorithm for quasi-linear geostatistical inversing, *Adv. Water Resour.*, *27*, 737–750.
Paige, C. C., and M. A. Saunders (1975), Solution of sparse indefinite systems of linear equations, *SIAM J. Numer. Anal.*, *12*, 617–629.
Paige, C. C., and M. A. Saunders (1982a), LSQR: An algorithm for sparse linear equations and sparse least squares, *ACM Trans. Math. Software*, *8*, 43–71.
Paige, C. C., and M. A. Saunders (1982b), LSQR: Sparse linear equations and least squares problems, *ACM Trans. Math. Software*, *8*, 195–224.
Peitgen, H. O., and D. Saupe (1988), *The Science of Fractal Images*, Springer, N. Y.
Saad, Y. (2003), *Iterative Methods for Sparse Linear Systems*, 2nd ed., Soc. for Ind. and Appl. Math, Philadelphia, Pa.
Saibaba, A. K., and P. K. Kitanidis (2012), Efficient methods for large-scale linear inversion using a geostatistical approach, *Water Resour. Res.*, *48*, W05522, doi:10.1029/2011WR011778.
Simunek, J., R. A. Jaramillo, M. G. Schaap, J. P. Vandervaere, and M. T. Genuchten (1998), Using an inverse method to estimate the hydraulic properties of crusted soils from tension-disc infiltrometer data, *Geoderma*, *86*, 61–81.
Strang, G. (2007), *Computational Science and Engineering*, vol.1, Wellesley Cambridge Press, Wellesley, Mass.
Sun, N. (1994), *Inverse Problems in Groundwater Modeling*, Kluwer Acad, Dordrecht, Netherlands.
Tarantola, A. (2005), *Inverse Problem Theory*, Soc. for Ind. and Appl. Math, Philadelphia, Pa.
Tonkin, M. J., and J. Doherty (2005), A hybrid regularized inversion methodology for highly parameterized environmental models, *Water Resour. Res.*, *41*, W10412, doi:10.1029/2005WR003995.
van den Doel, K., and U. M. Ascher (2006), On level set regularization for highly ill-posed distributed parameter estimation problems, *J. Comput. Phys.*, *216*(2), 707–723.
Vesselinov, V. V. (2012), MADS: Model analysis and decision support in C, Los Alamos National Laboratory, Los Alamos, N. M. [Available at http://mads.lanl.gov/.]
Vesselinov, V. V., et al. (2015), MADS.jl: (Model Analyses and Decision Support) in Julia, Los Alamos National Laboratory, Los Alamos, N. M. [Available at http://mads.lanl.gov/.]
Vogel, C. (2002), *Computational Methods for Inverse Problems*, Soc. for Ind. and Appl. Math, Philadelphia, Pa.
Watkins, D. S. (2010), *Fundamentals of Matrix Computations*, John Wiley, N. Y.
Welter, D. E., J. T. White, R. J. Hunt, and J. E. Doherty (2015), Approaches in highly parameterized inversion: PEST++ version 3, a parameter estimation and uncertainty analysis software suite optimized for large environmental models, *U.S. Geol. Surv. Tech. Methods*, *Book 7, Chap. C12*, 54 pp.
Zhu, X., and D. Zhang (2013), Efficient parallel Levenberg-Marquardt model fitting towards real-time automated parametric imaging microscopy, *PLOS ONE*, *8*, 0076665.