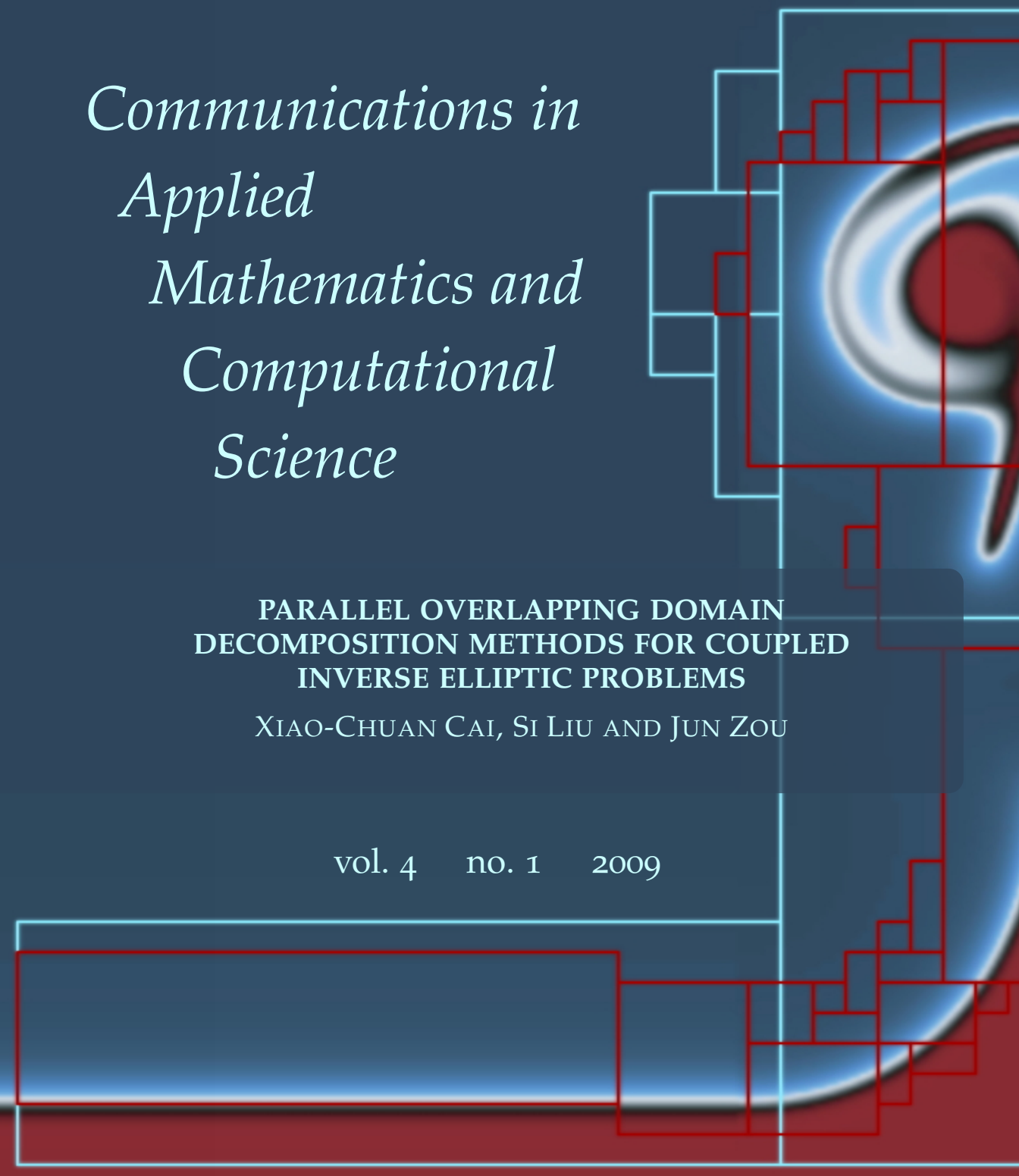
Communications in Applied Mathematics and Computational Science



PARALLEL OVERLAPPING DOMAIN DECOMPOSITION METHODS FOR COUPLED INVERSE ELLIPTIC PROBLEMS

Xiao-Chuan Cai, Si Liu and Jun Zou

vol. 4 no. 1 2009

mathematical sciences publishers

COMM. APP. MATH. AND COMP. SCI.
Vol. 4, No. 1, 2009

PARALLEL OVERLAPPING DOMAIN DECOMPOSITION METHODS FOR COUPLED INVERSE ELLIPTIC PROBLEMS

XIAO-CHUAN CAI, SI LIU AND JUN ZOU

We study an overlapping domain decomposition method for solving the coupled nonlinear system of equations arising from the discretization of inverse elliptic problems. Most algorithms for solving inverse problems take advantage of the fact that the optimality system has a natural splitting into three components: the state equation for the constraints, the adjoint equation for the Lagrange multipliers, and the equation for the parameter to be identified. Such algorithms often involve interiterations between the three separate solvers, and the intercomponent iteration is sequential. Several fully coupled or so-called one-shot approaches exist, and the main challenges in these approaches are that the system has stronger nonlinearity, and the corresponding Jacobian system is more ill-conditioned, in addition to being three times larger. Here we investigate a class of overlapping Newton–Krylov–Schwarz algorithms for solving such coupled systems, obtained with a pointwise ordering of the variables, and show numerically that, with a reasonably large overlap, the algorithm is capable of finding the solution even with noise and discontinuous coefficients. More importantly, we show that this approach is fully parallel and scalable with respect to the size of the problems.

1. Introduction

As parallel computers become more powerful, researchers are paying more attention to inverse problems which are more difficult and expensive to solve than forward problems [1; 11; 15; 24]. A key step in designing a high performance parallel algorithm is to formulate the problem with as few sequential calculations as possible. Here we study a parallel domain decomposition method for solving the system of nonlinear equations arising from the fully coupled finite difference discretization of some inverse elliptic problems in two-dimensional space.

MSC2000: 65N21, 65N55, 65Y05.

Keywords: inverse problems, domain decomposition, parallel computing, partial differential equations constrained optimiztion, inexact Newton.

The work of XCC and SL was partially supported by DOE FC02-04ER25595, NSF CNS 0722023 and CCF-0634894; the work of JZ was partially supported by the Hong Kong RGC grants (Projects 404105 and 404407).

Traditionally these problems are solved by using Uzawa-type algorithms which split the system into two or three subsystems solved individually. Subiterations are required between the subsystems. The subsystems are easier to solve than the global coupled system, but the iterations between subsystems are sequential in nature. There are several fully coupled approaches in which all variables are solved at the same time. They are often referred to as the one-shot method or the all-at-once method; see for example [3; 12; 16; 21]. In these approaches, the resulting linear and nonlinear systems of equations are three times larger, have stronger nonlinearity and are more ill-conditioned. Solving these fully coupled systems is a major challenge for any iterative methods.

The focus of this paper is to investigate a parallel domain decomposition preconditioning technique for the coupled systems. We show that with the powerful domain decomposition based preconditioner the convergence of the iterative methods can be obtained even for some difficult cases when the solution is discontinuous and when the observation data has high level of noise.

We consider an inverse elliptic problem [14]: Find the coefficient function $\rho(x)$ in the system

$$\begin{cases} -\nabla \cdot (\rho \nabla u) = f, & x \in \Omega, \\ u(x) = 0, & x \in \partial\Omega. \end{cases} \tag{1}$$

A widely used approach for solving the inverse problem is the output least-squares Tikhonov regularization method, which formulates the ill-posed inverse problem into different stabilized optimization problems, depending on the type of data available [6; 8; 13; 14]. For example, when the measurement of $u(x)$ is given, denoted as $z(x)$, the inverse problem can be transformed into the minimization problem:

$$\text{minimize } J(\rho, u) = \frac{1}{2}\int_\Omega (u-z)^2\,dx + \frac{\beta}{2}\int_\Omega |\nabla\rho|^2\,dx, \tag{2}$$

which is often referred to as the L^2 *least-squares problem*.

When the measurement of $\nabla u(x)$ is given, denoted as $\nabla z(x)$, the inverse problem can be transformed into the minimization problem:

$$\text{minimize } J(\rho, u) = \frac{1}{2}\int_\Omega \rho\,|\nabla u - \nabla z|^2\,dx + \frac{\beta}{2}\int_\Omega |\nabla\rho|^2\,dx, \tag{3}$$

which is often referred to as the H^1 *least-squares problem*. Both minimization problems (2) and (3) are subject to the constraint (1) satisfied by the pair (ρ, u), and the β-term is called the regularization term, and the constant β is the regularization parameter.

Instead of solving the constraint optimization problems (2) and (3), we turn to solving the saddle-point problems associated with the Lagrangian functional $\mathcal{L}$:

$$\mathcal{L}(\rho, u, \lambda) = \frac{1}{2}\int_\Omega (u-z)^2\,dx - \int_\Omega (\nabla\cdot\rho\nabla u + f)\lambda\,dx + \frac{\beta}{2}\int_\Omega |\nabla\rho|^2\,dx \quad (4)$$

for the L^2 case, or

$$\mathcal{L}(\rho, u, \lambda) = \frac{1}{2}\int_\Omega \rho|\nabla u - \nabla z|^2\,dx - \int_\Omega (\nabla\cdot\rho\nabla u + f)\lambda\,dx + \frac{\beta}{2}\int_\Omega |\nabla\rho|^2\,dx \quad (5)$$

for the H^1 case [8]. Hence the solutions to both minimization problems can be obtained by solving the corresponding optimality systems: Find (ρ, u, λ) such that

$$\begin{cases} (\nabla_\rho \mathcal{L})\,p = 0, \\ (\nabla_u \mathcal{L})\,w = 0, \\ (\nabla_\lambda \mathcal{L})\,\mu = 0, \end{cases} \quad (6)$$

for any (p, w, μ). More explicitly, we can reduce (6) to

$$\begin{cases} F^{(\rho)} \equiv -\beta\Delta\rho + \nabla u\cdot\nabla\lambda = 0, \\ F^{(u)} \equiv -\nabla\cdot(\rho\nabla\lambda) + (u-z) = 0, \\ F^{(\lambda)} \equiv -\nabla\cdot(\rho\nabla u) - f = 0, \end{cases} \quad (7)$$

in the L^2 case. Similarly, in the H^1 case, we have

$$\begin{cases} F^{(\rho)} \equiv -\beta\Delta\rho + \nabla u\cdot\nabla\lambda + 1/2|\nabla u - \nabla z|^2 = 0, \\ F^{(u)} \equiv -\nabla\cdot(\rho\nabla\lambda) + \nabla\cdot(\rho\nabla z) + f = 0, \\ F^{(\lambda)} \equiv -\nabla\cdot(\rho\nabla u) - f = 0. \end{cases} \quad (8)$$

Both systems (7) and (8) use the same boundary conditions

$$\begin{cases} (\partial\rho/\partial n) = 0, \\ u = 0, \\ \lambda = 0, \end{cases} \quad (9)$$

on $\partial\Omega$. The Dirichlet boundary conditions for u and λ are natural. The homogeneous Neumann boundary condition on ρ is the side effect of the H^1 regularization in (2) and (3). A derivation of the boundary condition $\partial\rho/\partial n = 0$ is given in the Appendix.

For solving the coupled systems, several techniques are available. For example, in [6; 13; 14], an augmented Lagrangian method was used and the solution was obtained by Uzawa-type algorithms, which decouples the problems into subproblems associated with ρ, u and λ separately, and as a result, only smaller problems need to be solved. The global convergence of these approaches was established in [7; 13]. In [2], a fully coupled discretization was used for some source term inverse

problems, and the coupled system was first reduced by block eliminations and then solved by a conjugate gradient (CG) method. Tremendous progress has been made in using domain decomposition methods for optimization problems with partial differential equations constraints [1; 2; 17; 18; 19]. The Newton–Krylov–Schwarz methodology [18] studied here is a general purpose approach that is not attached to any specific formulation of the inverse problem. We require a globally convergent inexact Newton's method with line search for the nonlinear systems; a Krylov subspace method for the Jacobian systems; and, most importantly, an additive Schwarz preconditioner, which works well with a scheme we use for the orderings of the unknowns and the functions. Although we are still unable to show theoretically that the linear and nonlinear iterative solvers are convergent, our parallel numerical experiments have shown clearly that this approach works well for some rather difficult test cases with jump coefficients and a high level of noise.

Section 2 addresses some basic properties of the linear and nonlinear systems and a reordering scheme to avoid zero pivoting. We also describe a Newton–Krylov–Schwarz algorithm for solving the fully coupled systems. We give numerical experiments in Section 3, followed by concluding remarks in Section 4.

2. Scalable solvers

2.1. ***A reordered fully coupled system.*** We use the so-called fully coupled ordering, by which we mean that all three variables defined at the same mesh point are always together throughout the calculations. The unknowns are ordered mesh point by mesh point, in contrast to physical variable by physical variable as required by most existing sequential quadratic programming (SQP) methods. At each mesh point, x_{ij}, the unknowns are ordered in the order of $\rho_{ij}, u_{ij}, \lambda_{ij}$, that is,

$$U = (\rho_{11}, u_{11}, \lambda_{11}, \rho_{21}, u_{21}, \lambda_{21}, \ldots, \rho_{nx\,ny}, u_{nx\,ny}, \lambda_{nx\,ny})^T. \tag{10}$$

If we order the functions in exactly the same order,

$$F = (F_{11}^{(\rho)}, F_{11}^{(u)}, F_{11}^{(\lambda)}, F_{21}^{(\rho)}, F_{21}^{(u)}, F_{21}^{(\lambda)}, \ldots, F_{nx\,ny}^{(\rho)}, F_{nx\,ny}^{(u)}, F_{nx\,ny}^{(\lambda)})^T = 0, \tag{11}$$

then the Jacobian in the L^2 case is a symmetric matrix of the form (12) with dense 3×3 blocks of the form (13)

$$J = \left(\begin{array}{ccc|ccc|ccc} \times & \times & & \times & & & & & \\ \times & \times & \times & & \times & & & & \\ & \times & \times & & & \times & & & \\ \hline \times & & & \times & \times & & \times & & \\ & \times & & \times & \times & \times & & \times & \\ & & \times & & \times & \times & & & \times \\ \hline & & & \times & & & \times & \times & \\ & & & & \times & & \times & \times & \times \\ & & & & & \times & & \times & \times \end{array}\right)_{n \times n(block)}, \tag{12}$$

where

$$\times = \begin{pmatrix} * & * & * \\ * & * & * \\ * & * & 0 \end{pmatrix}_{3\times 3}. \tag{13}$$

However, the zero value on the diagonal of (13) causes a pivoting problem in our LU factorization based solvers. To avoid the zero pivot situation, we reorder the unknowns (switching the u variable with the λ variable):

$$U = (\rho_{11}, \lambda_{11}, u_{11}, \rho_{21}, \lambda_{21}, u_{21}, \ldots, \rho_{nx\,ny}, \lambda_{nx\,ny}, u_{nx\,ny})^T = 0, \tag{14}$$

but keep the ordering of the functions unchanged as in (11). This reordering of function values does not change the block structure of the matrix, but the 3×3 block now takes the form:

$$\times = \begin{pmatrix} * & * & * \\ * & * & * \\ * & 0 & * \end{pmatrix}_{3\times 3}, \tag{15}$$

which is no longer symmetric. In this paper, our algorithms are not based on the structure of (13), but on the structure of (15), which is based on the ordering scheme (11) + (14).

For the purpose of parallel processing, the mesh points are ordered subdomain by subdomain. The ordering of the subdomains is not important since we use an additive method whose performance has nothing to do with the subdomain ordering.

2.2. ***Newton–Krylov method.*** The Newton–Krylov–Schwarz (NKS) methods [4] are a family of general-purpose parallel algorithms for solving systems of nonlinear algebraic equations. NKS has three main components: (i) an inexact Newton method for the nonlinear system; (ii) a Krylov subspace linear solver for the Jacobian systems (restarted GMRES [20]); and (iii) a Schwarz type preconditioner [22; 23]. We only study the regular additive Schwarz preconditioner in this paper, even though in some cases, the restricted version of the additive Schwarz method [5] maybe better.

We carry out the Newton iterations:

$$U_{k+1} = U_k - \lambda_k J(U_k)^{-1} F(U_k), \quad k = 0, 1, \ldots, \tag{16}$$

where U_0 is an initial approximation to the solution, $J(U_k) = F'(U_k)$ is the Jacobian at U_k, and λ_k is the steplength determined by a linesearch procedure [9; 10]. The inexactness of Newton's method is reflected in the fact that we do not solve the Jacobian systems exactly. The accuracy of the Jacobian solver is determined by some $\eta_k \in [0, 1)$ and the condition

$$\|F(U_k) + J(U_k)s_k\| \leq \eta_k \|F(U_k)\|. \tag{17}$$

The overall algorithm can be described as follows:

(1) Inexactly solve the linear system $J(U_k)s_k = -F(U_k)$ for s_k using a preconditioned GMRES(30).

(2) Perform a full Newton step with $\lambda_0 = 1$ in the direction s_k.

(3) If the full Newton step is unacceptable, we backtrack using the cubic backtracking procedure until a new λ is obtained that makes $U_+ = U_k + \lambda_k s_k$ an acceptable step.

(4) Set $U_{k+1} = U_+$ and return to step 1 unless a stopping condition has been met.

In step 1 above we use a right-preconditioned restarted GMRES to solve the linear system; that is, the vector s_k is obtained by approximately solving the right preconditioned Jacobian system

$$J(U_k)M_k^{-1}s_k' = -F(U_k),$$

where M_k^{-1} is a one-level additive Schwarz preconditioner and $s_k = M_k^{-1}s_k'$.

2.3. ***One-level additive Schwarz preconditioning.*** To formally define M_k^{-1}, we need to introduce a partition of Ω. We first partition the domain into nonoverlapping subdomains Ω_l, $l = 1, \ldots, N$, where N is the same as the number of processors (np). In order to obtain an overlapping decomposition of the domain, we extend each subdomain Ω_l to a larger region Ω_l', that is, $\Omega_l \subset \Omega_l'$. Only simple box decomposition is considered in this paper — all the subdomains Ω_l and Ω_l' are rectangular and made up of integral numbers of fine mesh cells. The size of Ω_l is $H_x \times H_y$ and the size of Ω_l' is $H_x' \times H_y'$, where the H's are chosen so that the overlap (ovlp) is uniform in the number of fine grid cells all around the perimeter, that is,

$$\text{ovlp} = (H_x' - H_x)/2 = (H_y' - H_y)/2$$

for interior subdomains. For boundary subdomains, we simply cut off the part that is outside Ω.

On each extended subdomain Ω_l', we construct a subdomain preconditioner B_l which is the discretization of the Frechet derivative taken at the current iteration,

$$J = \begin{pmatrix} \dfrac{\partial F^{(\rho)}}{\partial \rho} & \dfrac{\partial F^{(\rho)}}{\partial \lambda} & \dfrac{\partial F^{(\rho)}}{\partial u} \\ \dfrac{\partial F^{(u)}}{\partial \rho} & \dfrac{\partial F^{(u)}}{\partial \lambda} & \dfrac{\partial F^{(u)}}{\partial u} \\ \dfrac{\partial F^{(\lambda)}}{\partial \rho} & \dfrac{\partial F^{(\lambda)}}{\partial \lambda} & \dfrac{\partial F^{(\lambda)}}{\partial u} \end{pmatrix}. \tag{18}$$

In the L^2 case, the Frechet derivative at the point (ρ, λ, u) takes the form

$$F'_{L^2} = \begin{pmatrix} -\beta\Delta & \nabla u \cdot \nabla & \nabla\lambda \cdot \nabla \\ -(\Delta\lambda + \nabla\lambda \cdot \nabla) & -\nabla \cdot (\rho\nabla) & I \\ -(\Delta u + \nabla u \cdot \nabla) & 0 & -\nabla \cdot (\rho\nabla) \end{pmatrix}. \tag{19}$$

Similarly, in the H^1 case, we have

$$F'_{H^1} = \begin{pmatrix} -\beta\Delta & \nabla u \cdot \nabla & \nabla\lambda \cdot \nabla + (\nabla u - \nabla z) \cdot \nabla \\ -(\Delta\lambda + \nabla\lambda \cdot \nabla) + (\Delta z + \nabla z \cdot \nabla) & -\nabla \cdot (\rho\nabla) & 0 \\ -(\Delta u + \nabla u \cdot \nabla) & 0 & -\nabla \cdot (\rho\nabla) \end{pmatrix}. \tag{20}$$

Using the derivative and some boundary conditions, we can define the subdomain problems. For example, in the L^2 case, we have

$$\begin{cases} -\beta\Delta p + \nabla\lambda \cdot \nabla w + \nabla u \cdot \nabla\mu = g_1 & \text{in } \Omega'_l, \\ -\nabla \cdot (p\nabla\lambda) + w - \nabla \cdot (\rho\nabla\mu) = g_2 & \text{in } \Omega'_l, \\ -\nabla \cdot (p\nabla u) - \nabla \cdot (\rho\nabla w) = g_3 & \text{in } \Omega'_l, \\ p = w = \mu = 0 & \text{on } \partial\Omega'_l \cap \Omega, \\ (\partial p/\partial n) = w = \mu = 0 & \text{on } \partial\Omega'_l \cap \partial\Omega. \end{cases} \tag{21}$$

The solution (p, w, μ) and the right side (g_1, g_2, g_3) of the subdomain problem are not important at all. We only need the operator form (21) to construct a local solver B_l defined on the subdomain $\partial\Omega'_l$. Note that homogeneous Dirichlet boundary conditions are used on the internal subdomain boundary, and the original boundary conditions are used on the physical boundary, if present. A similar system is used for the H^1 least-squares problem.

Alternatively we can obtain B_l by extracting its elements from the global Jacobian matrix; that is, $B_l^{i,j} = \{J_{ij}\}$, where the node indexed by (i, j) belongs to the interior of Ω'_l. The entry J_{ij} is calculated with finite differences $J_{ij} = (F_i(U_j + \epsilon) - F_i(U_j))/\epsilon$, where $0 < \epsilon \ll 1$ is a constant. The additive Schwarz preconditioner can be written as

$$M_k^{-1} = I_1 B_1^{-1} (I_1)^T + \cdots + I_N B_N^{-1} (I_N)^T. \tag{22}$$

Let n be the total number of mesh points, and n'_l the total number of mesh points in Ω'_l, then I_l is an $3n \times 3n'_l$ extension matrix that extends each vector defined on Ω'_l to a vector defined on the entire fine mesh by padding an $3n'_l \times 3n'_l$ identity matrix with zero rows.

3. Numerical experiments

In this paper, we assume the problem is defined on $\Omega = (0, l_x) \times (0, l_y)$, which is covered by a uniform mesh of size h. To discretize the equations we use the usual

5-point central finite difference method for all variables. For the L^2 formulation (7), we define

$$F_{ij}^{(\rho)} = \beta \frac{4\rho_{ij} - \rho_{i-1\,j} - \rho_{i+1\,j} - \rho_{i\,j-1} - \rho_{i\,j+1}}{h^2} + \frac{u_{i+1\,j} - u_{i-1\,j}}{2h}\frac{\lambda_{i+1\,j} - \lambda_{i-1\,j}}{2h} + \frac{u_{i\,j+1} - u_{i\,j-1}}{2h}\frac{\lambda_{i\,j+1} - \lambda_{i\,j-1}}{2h},$$

$$F_{ij}^{(u)} = -\frac{\rho_{i+1/2\,j}(\lambda_{i+1\,j} - \lambda_{i\,j}) - \rho_{i-1/2\,j}(\lambda_{i\,j} - \lambda_{i-1\,j})}{h^2} - \frac{\rho_{i\,j+1/2}(\lambda_{i\,j+1} - \lambda_{i\,j}) - \rho_{i\,j-1/2}(\lambda_{i\,j} - \lambda_{i\,j-1})}{h^2} + (u_{ij} - z_{ij}),$$

$$F_{ij}^{(\lambda)} = -\frac{\rho_{i+1/2\,j}(u_{i+1\,j} - u_{i\,j}) - \rho_{i-1/2\,j}(u_{i\,j} - u_{i-1\,j})}{h^2} - \frac{\rho_{i\,j+1/2}(u_{i\,j+1} - u_{i\,j}) - \rho_{i\,j-1/2}(u_{i\,j} - u_{i\,j-1})}{h^2} - f_{ij},$$

where the half-point values of ρ are calculated using the average of the two neighboring values. Similarly, we obtain the discretization of the H^1 formulation (8):

$$F_{ij}^{(\rho)} = \beta \frac{4\rho_{ij} - \rho_{i-1\,j} - \rho_{i+1\,j} - \rho_{i\,j-1} - \rho_{i\,j+1}}{h^2} + \frac{u_{i+1\,j} - u_{i-1\,j}}{2h}\frac{\lambda_{i+1\,j} - \lambda_{i-1\,j}}{2h} + \frac{u_{i\,j+1} - u_{i\,j-1}}{2h}\frac{\lambda_{i\,j+1} - \lambda_{i\,j-1}}{2h} + \frac{1}{2}\left(\left(\frac{u_{i+1\,j} - u_{i-1\,j}}{2h} - \nabla_x z|_{ij}\right)^2 + \left(\frac{u_{i\,j+1} - u_{i\,j-1}}{2h} - \nabla_y z|_{ij}\right)^2\right),$$

$$F_{ij}^{(u)} = -\frac{\rho_{i+1/2\,j}(\lambda_{i+1\,j} - \lambda_{i\,j}) - \rho_{i-1/2\,j}(\lambda_{i\,j} - \lambda_{i-1\,j})}{h^2} - \frac{\rho_{i\,j+1/2}(\lambda_{i\,j+1} - \lambda_{i\,j}) - \rho_{i\,j-1/2}(\lambda_{i\,j} - \lambda_{i\,j-1})}{h^2} + \frac{\rho_{i+1\,j}\nabla_x z|_{i+1\,j} - \rho_{i-1\,j}\nabla_x z|_{i-1\,j}}{2h} + \frac{\rho_{i\,j+1}\nabla_y z|_{i\,j+1} - \rho_{i\,j-1}\nabla_y z|_{i\,j-1}}{2h} + f_{ij},$$

$$F_{ij}^{(\lambda)} = -\frac{\rho_{i+1/2\,j}(u_{i+1\,j} - u_{i\,j}) - \rho_{i-1/2\,j}(u_{i\,j} - u_{i-1\,j})}{h^2} - \frac{\rho_{i\,j+1/2}(u_{i\,j+1} - u_{i\,j}) - \rho_{i\,j-1/2}(u_{i\,j} - u_{i\,j-1})}{h^2} - f_{ij}.$$

To form an algebraic system of nonlinear equations from the finite difference equations, we need to order the unknowns and the corresponding functions. The ordering of the unknowns and the equations is not a big deal at all for accuracy

concerns, but is critically important for the linear Jacobian solver and for the preconditioning techniques. For example, if we order the unknowns variable by variable — that is, first ρ values for all mesh points, and second u values for all mesh points, and last λ values for all mesh points, then the Jacobian matrix takes the following block form:

$$J = \begin{pmatrix} \frac{\partial F^{(\rho)}}{\partial \rho} & \frac{\partial F^{(\rho)}}{\partial u} & \frac{\partial F^{(\rho)}}{\partial \lambda} \\ \frac{\partial F^{(u)}}{\partial \rho} & \frac{\partial F^{(u)}}{\partial u} & \frac{\partial F^{(u)}}{\partial \lambda} \\ \frac{\partial F^{(\lambda)}}{\partial \rho} & \frac{\partial F^{(\lambda)}}{\partial u} & \frac{\partial F^{(\lambda)}}{\partial \lambda} \end{pmatrix} = \begin{pmatrix} \frac{\partial F^{(\rho)}}{\partial \rho} & \frac{\partial F^{(\rho)}}{\partial u} & \frac{\partial F^{(\rho)}}{\partial \lambda} \\ \frac{\partial F^{(u)}}{\partial \rho} & I & \frac{\partial F^{(u)}}{\partial u} \\ \frac{\partial F^{(\lambda)}}{\partial \rho} & \frac{\partial F^{(\lambda)}}{\partial u} & 0 \end{pmatrix}_{3n \times 3n} . \tag{23}$$

Many interesting algorithms are designed based on the particular block structure of (23), and many algorithms fail to work also because of the structure of (23).

We study the performance of the proposed algorithm using four test cases. The first test case is from [14], and the purpose is to verify the accuracy of the algorithm. To understand the scalability of the algorithm, we introduce three more test problems.

To test the robustness of the algorithms, we add some noise to the observation data as

$$z^{\delta} = z + \delta \ \text{rand}\ (x, y) \tag{24}$$

or

$$\nabla z^{\delta} = \nabla z + \delta\, (\text{rand}\ (x, y), \text{rand}\ (x, y))^{T}, \tag{25}$$

depending on whether the formulation is L^2 or H^1. Here rand (x, y) is a random scalar function available in the C library, and δ is responsible for the magnitude of the noise. Some results with different levels of noise ($\delta = 0\%$, 1% and 10%) will be presented. Since u needs to satisfy the elliptic equation, we assume that u has some continuity and differentiability, as does ∇u. Therefore, we smooth u in L^2 formulation or ∇u in H^1 formulation before we start the Newton iteration. This is necessary especially when the noise level is high. In particular, when the noise level is 10%, we replace the value of u or ∇u by the weighted average value around it. And the weight function is defined as

$$\begin{array}{ccccc} \frac{1}{16} & & \frac{1}{8} & & \frac{1}{16} \\ & \searrow & \downarrow & \swarrow & \\ \frac{1}{8} & \rightarrow & \frac{1}{4} & \leftarrow & \frac{1}{8} \\ & \nearrow & \uparrow & \nwarrow & \\ \frac{1}{16} & & \frac{1}{8} & & \frac{1}{16} \end{array} .$$

We repeat this operation 3 times in all of our tests when $\delta = 10\%$. No smoothing is applied when δ is smaller than 10%.

To measure the accuracy of the numerical solution, we assume that the exact solution of the test cases are known, and error_u and error_ρ are the normalized discrete L^2 norms of the errors defined by

$$\text{error}_u = \sqrt{\sum (u_{ij} - u_{ij}^{\text{exact}})^2 \frac{h_x h_y}{l_x l_y}} \qquad \text{and} \qquad \text{error}_\rho = \sqrt{\sum (\rho_{ij} - \rho_{ij}^{\text{exact}})^2 \frac{h_x h_y}{l_x l_y}},$$

where h_x and h_y are mesh sizes along x and y directions, and l_x and l_y are sizes of the computational domain along the x and y directions, respectively.

In Newton's method, we use the initial guess

$$U_0 = (\rho^{(0)}, u^{(0)}, \lambda^{(0)})^T = (1, z, 0)^T.$$

For the L^2 formulation, z is the observation value. For the H^1 formulation, z is not directly available, but is obtained as a line integral of $\nabla_x z$ or $\nabla_y z$ along the x or y direction from one of the boundary points. In our test, at the point (x_i, y_j),

$$z(x_i, y_j) = z(x_0, y_j) + \sum_{l=1}^{i} (\nabla_x z)|_{x_l} h_x$$

if we take the integral along the x direction, or

$$z(x_i, y_j) = z(x_i, y_0) + \sum_{l=1}^{j} (\nabla_y z)|_{y_l} h_y$$

if we take the integral along the y direction.

In the test runs, we stop the Newton iteration if the following condition is satisfied

$$\|F(U_k)\| \le \max\left\{10^{-6}\|F(U_0)\|, 10^{-10}\right\}. \tag{26}$$

For the Jacobian solver, the GMRES iteration is stopped if

$$\|F(U_k) + J(U_k)s_k\| \le \max\left\{10^{-6}\|F(U_k)\|, 10^{-10}\right\}. \tag{27}$$

All the subdomain problems are solved with LU factorization. We implement the proposed algorithms using the Portable Extensible Toolkit for Scientific Computation (PETSc), developed at Argonne National Laboratory. Note that the timing results are obtained on a cluster of Linux PCs. The timings are just for references and should not be taken too seriously since the network of the PC cluster is slow and is also shared by many users.

3.1. ***Test cases.*** We next describe four test cases with the observation function

$$z(x, y) = \sin(\pi x)\sin(\pi y),$$

and several different ρ functions and on several different computational domains.

Test 1. In the first test we take $\Omega = (0, 1) \times (0, 1)$, and the right side f is constructed such that

$$\rho = 1 + 6x^2 y(1 - y)$$

is the elliptic coefficient to be identified. Note that this function does not satisfy $\partial\rho/\partial n = 0$ on the north boundary ($y = 1$), the south boundary ($y = 0$) and the east boundary ($x = 1$) of the domain.

Test 2. In the second test we take $\Omega = (0, 1) \times (0, 1)$ and the right side f is chosen so that the elliptic coefficient to be identified is

$$\rho = 1 + 100(xy(1 - x)(1 - y))^2.$$

Note that this function satisfies $\partial\rho/\partial n = 0$ on the entire boundary of the domain.

Test 3. In the third test we take a domain $\Omega = (0, l_x) \times (0, l_y)$ and the right side f is chosen so that the elliptic coefficient to be identified is

$$\rho = 1 + (-1)^{l_i + l_j} 6\big[(x - l_i)(y - l_j)(1 - (x - l_i))(1 - (y - l_j))\big]^2$$

when $(x, y) \in [l_i, l_i + 1) \times [l_j, l_j + 1)$, l_i and l_j are integers less than l_x and l_y, respectively.

We mention that ρ is a smooth function. Several different values of l_x and l_y are tested and the details are given where the test results are shown.

Test 4. In the fourth test we take a domain $\Omega = (0, l_x) \times (0, l_y)$ and the right side f is chosen so that the elliptic coefficient to be identified is

$$\rho = \begin{cases} 1 & \text{for } x - l_i \le 1/2, \\ 2 & \text{for } x - l_i > 1/2, \end{cases} \tag{28}$$

when l_i is even, $x \in [l_i, l_i + 1)$ and

$$\rho = \begin{cases} 2 & \text{for } x - l_i \le 1/2, \\ 1 & \text{for } x - l_i > 1/2, \end{cases} \tag{29}$$

when l_i is odd, $x \in [l_i, l_i + 1)$, and l_i is an integer less than l_x. This is a piecewise constant function defined on the computational domain and this function has several jumps along the x-direction.

Our discretization scheme and the solution algorithms do not require any a priori knowledge of the locations of the jumps. We mention that there are several techniques that are designed specifically for problems with discontinuous coefficients [6; 7; 14].

3.2. *Results and discussions of numerical experiments.*

3.2.1. *Test 1.* In this test, we solve the Jacobian systems using a global Gaussian elimination, therefore the domain decomposition preconditioned iterative solver introduced in the previous section plays no role at all. As mentioned earlier, this equation does not satisfy the boundary condition

$$\frac{\partial \rho}{\partial n} = 0.$$

We see from Figure 1 that the numerical solution has some visible difference from the exact solution. To satisfy the above boundary condition, the numerical solution has some distortion on the north, south and east boundaries. This is more obvious near the two corners. Nevertheless, the results match that of [14]. When the noise level is high, the effect of $\partial\rho/\partial n = 0$ is larger near the corners for the L^2 formulation of the inverse problem. The H^1 formulation is less sensitive to this boundary condition.

From Table 1, we see that our method converges well. It takes about 3–5 Newton iterations to converge. For a given level of noise, the results are more accurate when we choose a finer mesh. When the noise level is high, we need larger β values for the Newton to converge. Comparing the solution plots in Figure 1, we see that the H^1 formulation generally gives us better solutions than the L^2 formulation, but somehow it takes slightly more Newton iterations to converge than the L^2 case. This is also true for our other test problems.

3.2.2. *Test 2.* The exact and numerical solutions are shown in Figure 2. It can be seen that the numerical solutions are quite accurate (Table 2) in the whole domain since the equation satisfies the Neumann boundary condition $\partial\rho/\partial n = 0$ on all its boundary. This is different from Test 1 where the accuracy is low near the corners. We tested three meshes 41×41, 81×81, and 161×161. When the Jacobian systems are solved exactly with a global Gaussian elimination, the total number of Newton iterations ranges from 3–6, and the iteration numbers are not sensitive to the level of noise.

We next look into the performance of the Newton–Krylov–Schwarz algorithm, in particular, we would like to know how the convergence depends on the mesh size, the number of subdomains, and the overlapping size. First, we study the processor-scalability. We solve the problem on a 321×321 mesh using different numbers of processors. We show the results, in terms of iteration numbers and the computing time totals, in Table 3. The number of Newton iterations does not change when we change the number of processors or the overlapping size. If we fix the number of subdomains or the number of processors, as we increase the overlapping size, the number of GMRES iterations decreases. The computing time decreases to a point and then begins to increase. This suggests that an optimal overlapping

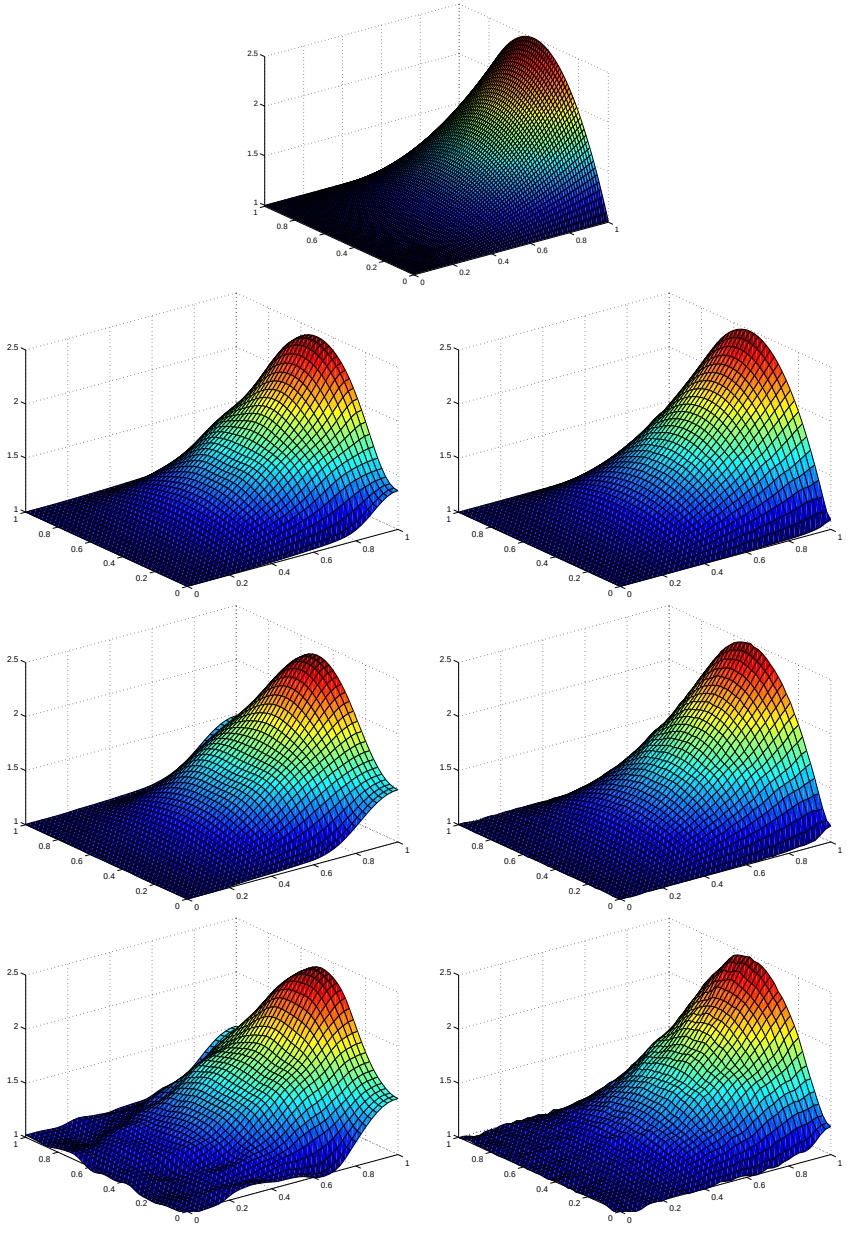


Figure 1. Test 1. Top: surface plot of the exact solution ρ. Bottom six pictures are numerical solutions with $\delta = 0\%$, $\delta = 1\%$ and $\delta = 10\%$. Left: L^2 formulation; right: H^1 formulation.

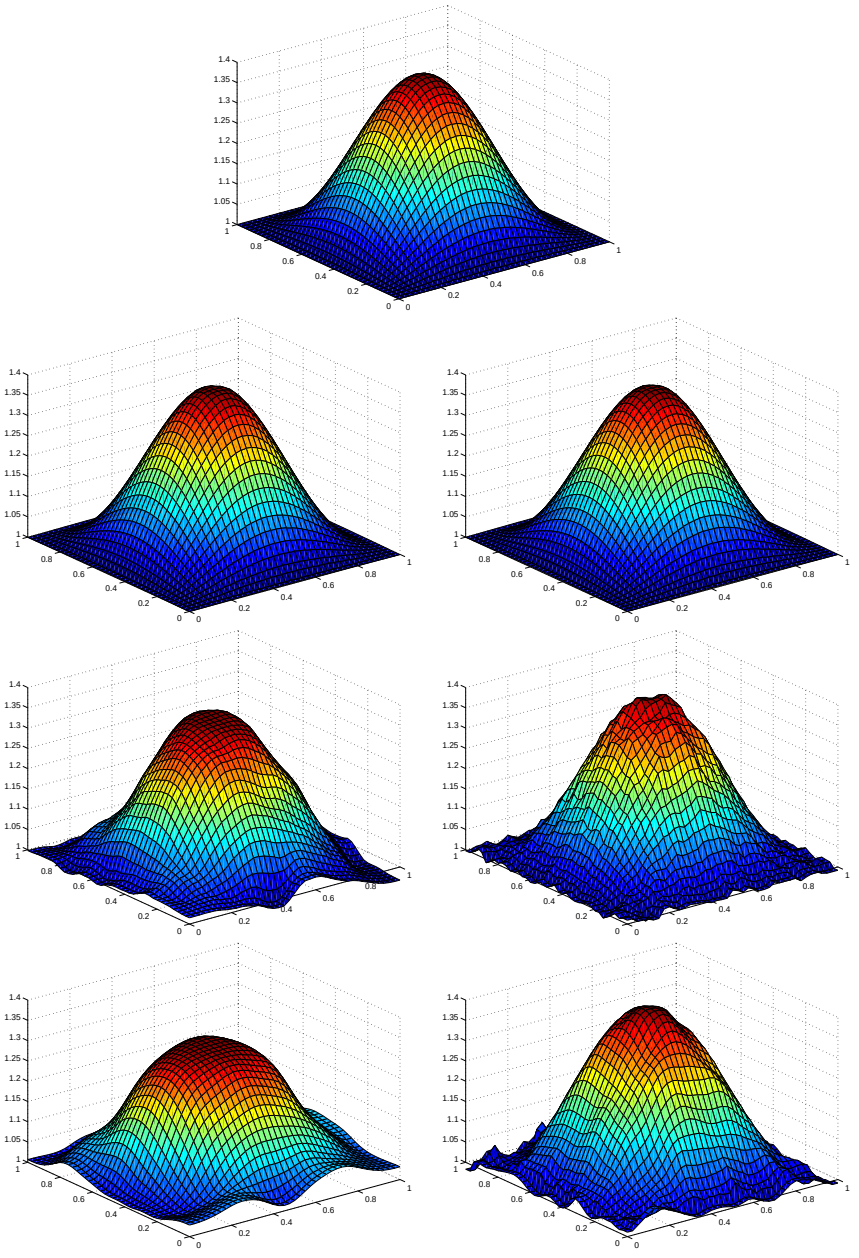


Figure 2. Test 2. Top: surface plot of exact solution ρ. Bottom six pictures are numerical solutions with $\delta = 0\%$, $\delta = 1\%$ and $\delta = 10\%$. Left: L^2 formulation; right: H^1 formulation.

	error_u	error_ρ	Newton
L^2 formulation, 41×41 mesh			
$\beta = 10^{-6}, \delta = 0$	0.000535	0.042644	3
$\beta = 10^{-5}, \delta = 1\%$	0.002032	0.073478	4
$\beta = 10^{-4}, \delta = 10\%$	0.009951	0.143455	4
L^2 formulation, 81×81 mesh			
$\beta = 10^{-6}, \delta = 0$	0.000455	0.034766	3
$\beta = 10^{-5}, \delta = 1\%$	0.001759	0.062192	4
$\beta = 10^{-4}, \delta = 10\%$	0.007615	0.119419	4
L^2 formulation, 161×161 mesh			
$\beta = 10^{-6}, \delta = 0$	0.000424	0.031326	3
$\beta = 10^{-5}, \delta = 1\%$	0.001683	0.058537	4
$\beta = 10^{-4}, \delta = 10\%$	0.006975	0.113078	4
H^1 formulation, 41×41 mesh			
$\beta = 10^{-5}, \delta = 0$	0.000277	0.020434	5
$\beta = 10^{-5}, \delta = 1\%$	0.000302	0.020677	5
$\beta = 10^{-4}, \delta = 10\%$	0.006932	0.036644	5
H^1 formulation, 81×81 mesh			
$\beta = 10^{-5}, \delta = 0$	0.000083	0.010343	4
$\beta = 10^{-5}, \delta = 1\%$	0.000103	0.010697	4
$\beta = 10^{-4}, \delta = 10\%$	0.001959	0.021829	4
H^1 formulation, 161×161 mesh			
$\beta = 10^{-6}, \delta = 0$	0.000018	0.003760	5
$\beta = 10^{-5}, \delta = 1\%$	0.000039	0.007377	4
$\beta = 10^{-4}, \delta = 10\%$	0.000496	0.017599	5

Table 1. Test 1. Errors and number of Newton iterations for three meshes with three levels of noise in L^2 and H^1 formulations.

size exists if the goal is to minimize the total computing time when the number of processors is not changed. On the fixed mesh, and with a fixed overlapping size, the number of GMRES iterations increases as we use more processors. This is expected since this is a single-level algorithm. Second, we check the h-scalability of our algorithm. Here, we increase the mesh size for the test problem and the number of processors at the same ratio in order for each processor to have a fixed number of mesh points. Table 4 shows the results with different overlapping sizes for np$=4$,

	error_u	error_ρ	Newton
L^2 formulation, 41×41 mesh			
$\beta = 10^{-6}, \delta = 0$	0.000078	0.003163	3
$\beta = 10^{-5}, \delta = 1\%$	0.000765	0.010723	3
$\beta = 10^{-4}, \delta = 10\%$	0.008222	0.038667	3
L^2 formulation, 81×81 mesh			
$\beta = 10^{-6}, \delta = 0$	0.000073	0.003177	3
$\beta = 10^{-5}, \delta = 1\%$	0.000532	0.010070	3
$\beta = 10^{-4}, \delta = 10\%$	0.003849	0.029056	3
L^2 formulation, 161×161 mesh			
$\beta = 10^{-6}, \delta = 0$	0.000072	0.003203	3
$\beta = 10^{-5}, \delta = 1\%$	0.000504	0.009908	3
$\beta = 10^{-5}, \delta = 10\%$	0.002064	0.026190	4
H^1 formulation, 41×41 mesh			
$\beta = 10^{-5}, \delta = 0$	0.000385	0.001559	5
$\beta = 10^{-5}, \delta = 1\%$	0.000377	0.005097	6
$\beta = 10^{-4}, \delta = 10\%$	0.006927	0.020951	4
H^1 formulation, 81×81 mesh			
$\beta = 10^{-5}, \delta = 0$	0.000089	0.000386	4
$\beta = 10^{-5}, \delta = 1\%$	0.000108	0.003493	4
$\beta = 10^{-4}, \delta = 10\%$	0.001907	0.009897	4
H^1 formulation, 161×161 mesh			
$\beta = 10^{-6}, \delta = 0$	0.000022	0.000098	4
$\beta = 10^{-5}, \delta = 1\%$	0.000029	0.002355	4
$\beta = 10^{-4}, \delta = 10\%$	0.000460	0.006295	4

Table 2. Test 2. Errors and number of Newton iterations for three meshes with three levels of noise in L^2 and H^1 formulations.

16 and 64. Both the number of Newton iterations and the number of GMRES iterations are almost constants. The computing time is close to be quadrupled when the size of the problem is quadrupled with np fixed.

3.2.3. *Test 3.* For forward elliptic problems, one can always obtain a large test problem (with a large number of degree of freedoms) by refining the mesh, but for inverse elliptic problems, sometimes it does not make sense to use a very fine mesh

np	Newton	ovlp= 1	ovlp= 2	ovlp= 4	ovlp= 8	ovlp= 16
L^2 formulation, $\beta = 10^{-6}$, $\delta = 0\%$						
4	3	45(134.68)	33(124.00)	19(111.96)	13(111.61)	8(118.08)
16	3	66(28.17)	46(24.35)	34(23.18)	22(24.73)	14(32.06)
64	3	128(8.73)	92(7.36)	63(6.64)	42(7.20)	25(9.17)
L^2 formulation, $\beta = 10^{-5}$, $\delta = 1\%$						
4	3	43(131.74)	26(115.07)	19(111.84)	14(111.91)	9(118.45)
16	3	61(26.86)	45(23.89)	31(22.37)	23(24.55)	15(32.46)
64	3	134(8.99)	94(7.44)	62(6.59)	45(7.41)	25(9.74)
L^2 formulation, $\beta = 10^{-5}$, $\delta = 10\%$						
4	4	49(184.48)	36(168.30)	23(154.74)	16(152.48)	10(159.11)
16	4	72(39.41)	54(34.92)	40(32.98)	25(34.12)	19(44.94)
64	4	179(15.21)	118(11.82)	79(10.30)	54(10.99)	35(14.75)
H^1 formulation, $\beta = 10^{-5}$, $\delta = 0\%$						
4	5	52(233.36)	34(202.76)	21(184.87)	14(182.29)	12(194.77)
16	4	96(47.03)	63(37.67)	41(32.62)	26(33.54)	17(43.48)
64	4	171(14.44)	110(11.07)	71(9.50)	46(9.81)	25(12.77)
H^1 formulation, $\beta = 10^{-5}$, $\delta = 1\%$						
4	5	48(227.45)	33(200.85)	20(184.23)	14(182.46)	10(194.16)
16	4	75(39.69)	55(34.75)	30(28.65)	22(31.67)	15(42.30)
64	4	142(11.60)	89(9.45)	53(7.79)	40(9.14)	23(12.28)
H^1 formulation, $\beta = 10^{-4}$, $\delta = 10\%$						
4	4	61(199.28)	43(176.87)	26(156.70)	18(151.73)	12(159.66)
16	4	89(44.54)	60(36.26)	45(34.41)	26(33.77)	18(43.76)
64	4	141(12.23)	104(10.58)	66(9.01)	46(9.82)	26(12.94)

Table 3. Test 2. Processor scalability in L^2 and H^1 formulations on a 321×321 mesh, with total number of Newton iterations, average number of GMRES iterations per Newton, and total computing time in seconds shown in ( ), with different overlapping sizes.

since the accuracy is determined by the mesh size and the level of noise. When the level of noise is fixed, one may not always obtain a better solution even if a finer mesh is used. To test the scalability of the proposed algorithm and software, we construct larger test problems by increasing the computational domain.

In this test case the solution has multiple peaks, and the exact and numerical solutions are shown in Figure 3. We observe that the errors are kept at the same level when we change the size of the computational domain for different numbers of processors. The H^1 results are a little better than the L^2 results. According to the results shown in Table 5 we see that the number of Newton iterations is almost a constant for different numbers of processors, but the number of GMRES iterations slightly increases, which is expected for a single-level method. In some cases when the computational domain is very large and the noise level is high, Newton fails to converge unless we use a larger β.

3.2.4. *Test 4.* This problem is also defined on a large domain as in Test 3. The difference is that ρ is a discontinuous function with several jumps in the x-direction as shown in Figure 4. The surface plots of the computational results of ρ in L^2 and H^1 formulation are shown in Figure 4. The results obtained with the L^2 formulation are continuous and quite smooth even if the actual solutions should have jumps. The H^1 formulation leads to piecewise continuous solutions and keeps the discontinuity much better than the L^2 formulation. As for the scalability of the algorithm, Table 6 shows that the performance is very similar to that of Test 3.

4. Concluding remarks

A parallel one-level Newton–Krylov–Schwarz method was investigated for solving the nonlinear system of algebraic equations arising from the fully coupled finite difference discretization of inverse elliptic problems in both the L^2 and H^1 formulations. We tested the algorithms for problems with smooth solutions and for problems with discontinuous solutions. Acceptable solutions were obtained even when the level of noise is quite large. The mesh and processor scalabilities of the algorithm were studied for meshes up to 641×641 in two dimensions and with up to 64 processors. The iterative method was optimally scalable with respect to the mesh size. The number of iterations increases as the number of processors increases, which was expected for the one-level method. The algorithmic and software framework is applicable to a wide range of inverse problems, and our future research includes the extension of the algorithms to three dimensions, the study of other regularization techniques and their impact on the linear and nonlinear solvers, and the development of multilevel versions of the algorithm which will be needed for parallel computers with a large number of processors.

Appendix

In this section, we prove that if we choose the regularization term

$$N(\rho) = \frac{1}{2} \int_\Omega |\nabla \rho|^2 \, dx,$$

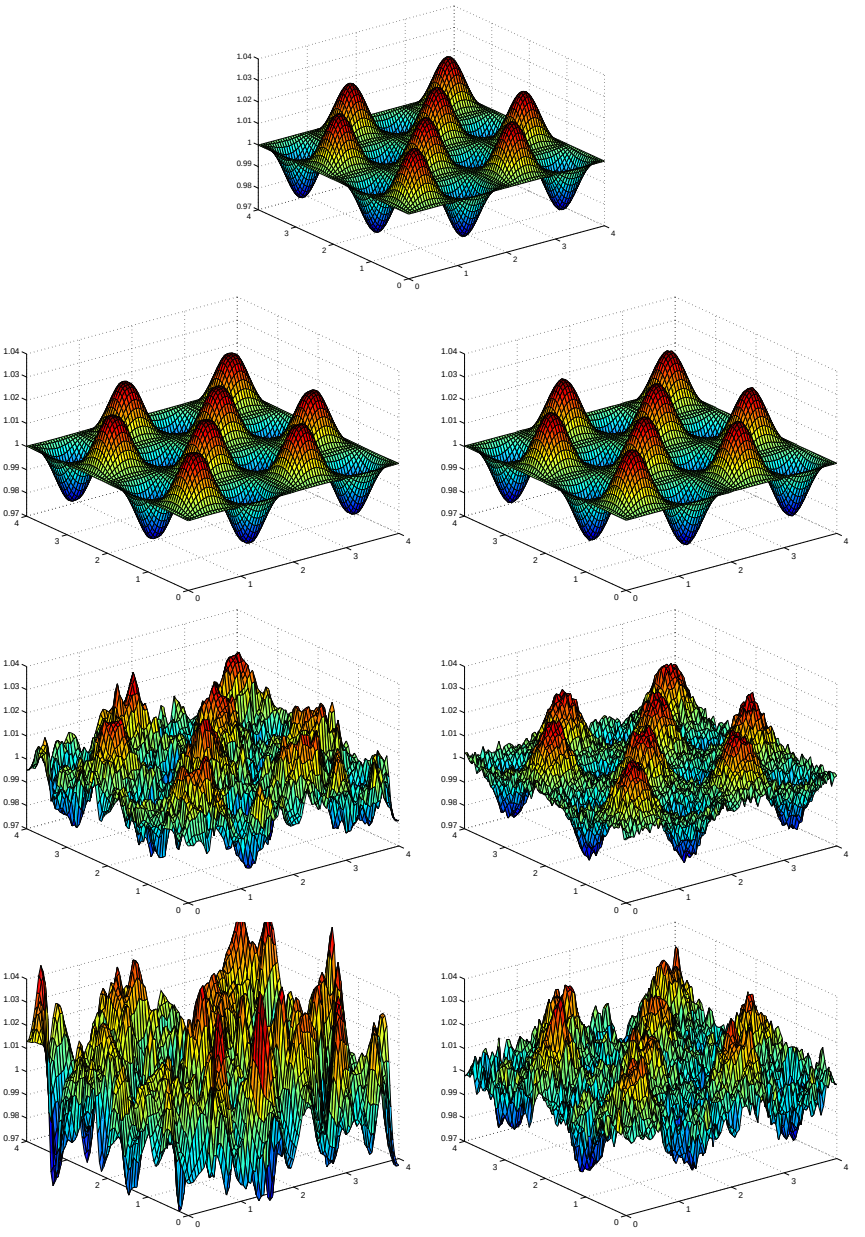


Figure 3. Test 3 on computational domain $(0,4)\times(0,4)$. Top: surface plot of the exact solution ρ. Bottom six pictures are numerical solutions with $\delta = 0\%$, $\delta = 1\%$ and $\delta = 10\%$. Left: L^2 formulation; right: H^1 formulation.

np	Newton	GMRES	Newton	GMRES	Newton	GMRES
	81×81 mesh		161×161 mesh		321×321 mesh	
L^2 formulation, $\beta = 10^{-6}$, $\delta = 0\%$						
4	3	7(2.84)	3	6(18.97)	3	6(142.96)
16	3	14(0.69)	3	14(4.51)	3	14(32.06)
64	3	38(0.32)	3	40(1.16)	3	42(7.21)
L^2 formulation, $\beta = 10^{-5}$, $\delta = 1\%$						
4	3	7(2.85)	3	7(19.19)	3	6(143.01)
16	3	18(0.75)	3	17(4.73)	3	15(32.48)
64	3	47(0.38)	3	45(1.24)	3	45(7.41)
L^2 formulation, $\beta = 10^{-4}$, $\delta = 10\%$						
4	3	9(2.99)	3	8(19.72)	3	8(146.54)
16	3	24(0.86)	3	23(5.34)	3	22(35.48)
64	3	75(0.54)	3	72(1.69)	3	66(9.38)
H^1 formulation, $\beta = 10^{-5}$, $\delta = 0\%$						
4	4	7(3.61)	4	7(24.77)	4	7(234.55)
16	4	17(0.94)	4	19(4.80)	4	17(43.50)
64	4	44(0.47)	4	47(1.23)	4	46(9.79)
H^1 formulation, $\beta = 10^{-5}$, $\delta = 1\%$						
4	4	8(3.66)	4	7(24.55)	5	7(234.02)
16	4	16(0.93)	4	15(5.89)	4	15(42.30)
64	4	44(0.47)	4	41(1.49)	4	40(9.15)
H^1 formulation, $\beta = 10^{-4}$, $\delta = 10\%$						
4	4	8(3.70)	4	8(25.23)	4	8(190.87)
16	4	17(0.95)	4	17(6.16)	4	18(43.77)
64	4	41(0.45)	4	48(1.66)	4	46(9.82)

Table 4. Test 2. Mesh size scalability in L^2 and H^1 formulations, with total number of Newton iterations, average number of GMRES iterations per Newton, and total computing time in seconds, shown in ( ), for different meshes and numbers of processors; ovlp $= 1/5$ diameter of the subdomain.

np	domain	mesh	error_u	error_ρ	Newton	GMRES
		L^2 formulation, $\beta = 10^{-6}$, $\delta = 0\%$				
1	$(0,1)\times(0,1)$	81×81	0.000004	0.000177	2	1
4	$(0,2)\times(0,2)$	161×161	0.000004	0.000190	2	15
16	$(0,4)\times(0,4)$	321×321	0.000004	0.000195	2	31
64	$(0,8)\times(0,8)$	641×641	0.000005	0.000220	2	130
		L^2 formulation, $\beta = 10^{-5}$, $\delta = 1\%$				
1	$(0,1)\times(0,1)$	81×81	0.000359	0.004668	3	1
4	$(0,2)\times(0,2)$	161×161	0.000383	0.004655	3	14
16	$(0,4)\times(0,4)$	321×321	0.000381	0.004637	3	30
64	$(0,8)\times(0,8)$	641×641	0.000378	0.004637	3	61
		L^2 formulation, $\beta = 10^{-4}$, $\delta = 10\%$				
1	$(0,1)\times(0,1)$	81×81	0.003209	0.015846	3	1
4	$(0,2)\times(0,2)$	161×161	0.002886	0.017710	3	12
16	$(0,4)\times(0,4)$	321×321	0.002876	0.014799	3	24
64	$(0,8)\times(0,8)$	641×641	0.002844	0.014890	3	43
		H^1 formulation, $\beta = 10^{-5}$, $\delta = 0\%$				
1	$(0,1)\times(0,1)$	81×81	0.000066	0.000263	4	1
4	$(0,2)\times(0,2)$	161×161	0.000064	0.000260	5	30
16	$(0,4)\times(0,4)$	321×321	0.000065	0.000263	5	58
64	$(0,8)\times(0,8)$	641×641	0.000071	0.000278	5	123
		H^1 formulation, $\beta = 10^{-4}$, $\delta = 1\%$				
1	$(0,1)\times(0,1)$	81×81	0.000085	0.001706	4	1
4	$(0,2)\times(0,2)$	161×161	0.000067	0.001624	4	18
16	$(0,4)\times(0,4)$	321×321	0.000078	0.001569	4	42
64	$(0,8)\times(0,8)$	641×641	0.000091	0.001550	4	76
		H^1 formulation, $\beta = 10^{-3}$, $\delta = 10\%$				
1	$(0,1)\times(0,1)$	81×81	0.001894	0.006133	3	1
4	$(0,2)\times(0,2)$	161×161	0.001752	0.005660	4	14
16	$(0,4)\times(0,4)$	321×321	0.001907	0.005773	4	26
64	$(0,8)\times(0,8)$	641×641	0.001933	0.004450	4	41*

Table 5. Test 3 with ovlp = 16. *Bottom line: $\beta = 10^{-2}$.

np	domain	mesh	error_u	error_ρ	Newton	GMRES
L^2 formulation, $\beta = 10^{-6}$, $\delta = 0\%$						
1	$(0,1)\times(0,1)$	81×81	0.000879	0.142454	3	1
4	$(0,2)\times(0,2)$	161×161	0.000879	0.142054	3	7
16	$(0,4)\times(0,4)$	321×321	0.000863	0.142271	3	21
64	$(0,8)\times(0,8)$	641×641	0.000861	0.142213	3	64
L^2 formulation, $\beta = 10^{-5}$, $\delta = 1\%$						
1	$(0,1)\times(0,1)$	81×81	0.002342	0.175353	4	1
4	$(0,2)\times(0,2)$	161×161	0.002313	0.174540	4	7
16	$(0,4)\times(0,4)$	321×321	0.002280	0.174360	4	17
64	$(0,8)\times(0,8)$	641×641	0.002265	0.174064	4	50
L^2 formulation, $\beta = 10^{-4}$, $\delta = 10\%$						
1	$(0,1)\times(0,1)$	81×81	0.006600	0.220451	4	1
4	$(0,2)\times(0,2)$	161×161	0.006522	0.219846	4	8
16	$(0,4)\times(0,4)$	321×321	0.006148	0.217108	4	19
64	$(0,8)\times(0,8)$	641×641	0.005972	0.215940	4	42
H^1 formulation, $\beta = 10^{-5}$, $\delta = 0\%$						
1	$(0,1)\times(0,1)$	81×81	0.000093	0.078365	4	1
4	$(0,2)\times(0,2)$	161×161	0.000093	0.078148	6	29
16	$(0,4)\times(0,4)$	321×321	0.000093	0.078040	6	56
64	$(0,8)\times(0,8)$	641×641	0.000093	0.077985	6	95
H^1 formulation, $\beta = 10^{-4}$, $\delta = 1\%$						
1	$(0,1)\times(0,1)$	81×81	0.000301	0.106186	4	1
4	$(0,2)\times(0,2)$	161×161	0.000298	0.105925	4	29
16	$(0,4)\times(0,4)$	321×321	0.000299	0.105777	4	38
64	$(0,8)\times(0,8)$	641×641	0.000301	0.105717	4	64
H^1 formulation, $\beta = 10^{-3}$, $\delta = 10\%$						
1	$(0,1)\times(0,1)$	81×81	0.002268	0.143252	4	1
4	$(0,2)\times(0,2)$	161×161	0.002167	0.142860	4	15
16	$(0,4)\times(0,4)$	321×321	0.002293	0.142457	4	28
64	$(0,8)\times(0,8)$	641×641	0.002334	0.142601	5	55

Table 6. Test 4 with ovlp = 16.

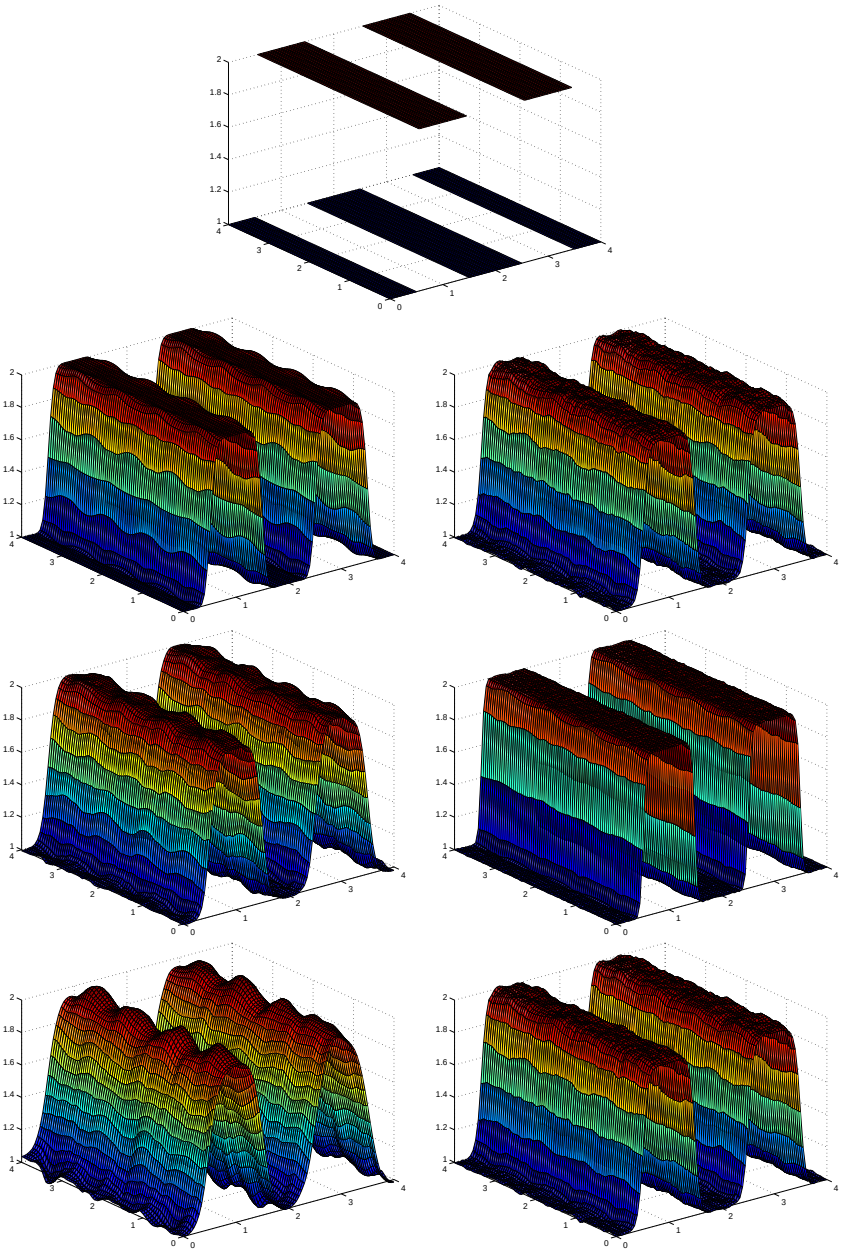


Figure 4. Test 4 on computational domain $(0,4) \times (0,4)$. Top: surface plot of the exact solution ρ. Bottom six pictures are numerical solutions with $\delta = 0\%$, $\delta = 1\%$ and $\delta = 10\%$. Left: L^2 formulation; right: H^1 formulation.

as in (4) and (5), then ρ automatically satisfies the zero Neumann boundary condition

$$\frac{\partial \rho}{\partial n} = 0.$$

To see this, we take the derivative of $\mathcal{L}$ in (4) with respect to ρ at any direction $p \in H^1(\Omega)$ to obtain

$$(\nabla_\rho \mathcal{L}) p = \beta \int_\Omega \nabla \rho \cdot \nabla p \, dx + \int_\Omega p \nabla u \cdot \nabla \lambda \, dx = 0 \,. \tag{A.1}$$

Applying the integration by parts to the first term in (A.1),

$$\int_\Omega \nabla \rho \cdot \nabla p \, dx = - \int_\Omega \Delta \rho \; p \, dx + \int_{\partial\Omega} \frac{\partial \rho}{\partial n} p \, ds \,,$$

we obtain

$$\int_\Omega (-\beta \Delta \rho + \nabla u \cdot \nabla \lambda) \; p \, dx + \beta \int_{\partial\Omega} \frac{\partial \rho}{\partial n} \; p \, ds = 0$$

for any p. This implies that, if $\beta \neq 0$,

$$\frac{\partial \rho}{\partial n} = 0$$

on $\partial\Omega$. The same result can be obtained for the H^1 formulation (4).

References

[1] V. Akcelik, G. Biros, A. Draganescu, O. Ghattas, J. Hilland, and B. van Bloeman Waanders, *Dynamic data-driven inversion for terascale simulations: real-time identification of airborne contaminants*, Proceedings of Supercomputing, 2005.

[2] V. Akçelik, G. Biros, O. Ghattas, K. Long, and B. van B. Waanders, *A variational finite element method for source inversion for convective-diffusive transport*, Finite Elem. Anal. Des. **39** (2003), no. 8, 683–705, 14th Robert J. Melosh Competition (Durhan, NC, 2002). MR 1985391

[3] U. M. Ascher and E. Haber, *A multigrid method for distributed parameter estimation problems*, Electron. Trans. Numer. Anal. **15** (2003), 1–17, Tenth Copper Mountain Conference on Multigrid Methods (Copper Mountain, CO, 2001). MR 2005a:65148 Zbl 1031.65108

[4] X.-C. Cai, W. D. Gropp, D. E. Keyes, R. G. Melvin, and D. P. Young, *Parallel Newton–Krylov–Schwarz algorithms for the transonic full potential equation*, SIAM J. Sci. Comput. **19** (1998), no. 1, 246–265, Special issue on iterative methods (Copper Mountain, CO, 1996). MR 99b:65138 Zbl 0917.76035

[5] X.-C. Cai and M. Sarkis, *A restricted additive Schwarz preconditioner for general sparse linear systems*, SIAM J. Sci. Comput. **21** (1999), no. 2, 792–797. MR 2000f:65133 Zbl 0944.65031

[6] T. F. Chan and X.-C. Tai, *Identification of discontinuous coefficients in elliptic problems using total variation regularization*, SIAM J. Sci. Comput. **25** (2003), no. 3, 881–904. MR2005c:65091 Zbl 1046.65090

[7] Z. Chen and J. Zou, *Finite element methods and their convergence for elliptic and parabolic interface problems*, Numer. Math. **79** (1998), no. 2, 175–202. MR 99d:65313 Zbl 0909.65085

[8] ———, *An augmented Lagrangian method for identifying discontinuous parameters in elliptic systems*, SIAM J. Control Optim. **37** (1999), no. 3, 892–910. MR 2000d:65203 Zbl 0940.65117

[9] J. E. Dennis, Jr. and R. B. Schnabel, *Numerical methods for unconstrained optimization and nonlinear equations*, Classics in Applied Mathematics, no. 16, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1996, Corrected reprint of the 1983 original. MR 96i:90002 Zbl 0847.65038

[10] S. C. Eisenstat and H. F. Walker, *Globally convergent inexact Newton methods*, SIAM J. Optim. **4** (1994), no. 2, 393–422. MR 95c:65078 Zbl 0814.65049

[11] H. W. Engl, M. Hanke, and A. Neubauer, *Regularization of inverse problems*, Mathematics and its Applications, no. 375, Kluwer Academic Publishers Group, Dordrecht, 1996. MR97k:65145 Zbl 0859.65054

[12] E. Haber and U. M. Ascher, *Preconditioned all-at-once methods for large, sparse parameter estimation problems*, Inverse Problems **17** (2001), no. 6, 1847–1864. MR 2002j:35308 Zbl 0995.65110

[13] K. Ito and K. Kunisch, *The augmented Lagrangian method for parameter estimation in elliptic systems*, SIAM J. Control Optim. **28** (1990), no. 1, 113–136. MR 91d:35228 Zbl 0709.93021

[14] Y. L. Keung and J. Zou, *An efficient linear solver for nonlinear parameter identification problems*, SIAM J. Sci. Comput. **22** (2000), no. 5, 1511–1526. MR 2001m:65156 Zbl 0978.35096

[15] A. Kirsch, *An introduction to the mathematical theory of inverse problems*, Applied Mathematical Sciences, no. 120, Springer, New York, 1996. MR 99c:34023 Zbl 0865.35004

[16] G. Kuruvila, S. Ta'asan, and M. D. Salas, *Airfoil optimization by the one-shot method*, AGARD Report 803, 1994.

[17] T. P. Mathew, M. Sarkis, and C. E. Schaerer, *Analysis of block matrix preconditioners for elliptic optimal control problems*, Numer. Linear Algebra Appl. **14** (2007), no. 4, 257–279. MR 2008d:65038

[18] E. E. Prudencio, R. Byrd, and X.-C. Cai, *Parallel full space SQP Lagrange–Newton–Krylov–Schwarz algorithms for PDE-constrained optimization problems*, SIAM J. Sci. Comput. **27** (2006), no. 4, 1305–1328. MR 2006k:49088 Zbl 1092.49024

[19] E. E. Prudencio and X.-C. Cai, *Parallel multilevel restricted Schwarz preconditioners with pollution removing for PDE-constrained optimization*, SIAM J. Sci. Comput. **29** (2007), no. 3, 964–985. MR 2008d:76033

[20] Y. Saad, *Iterative methods for sparse linear systems*, second ed., Society for Industrial and Applied Mathematics, Philadelphia, PA, 2003. MR 2004h:65002 Zbl 1031.65046

[21] A. Shenoy, M. Heinkenschloss, and E. M. Cliff, *Airfoil design by an all-at-once method*, Int. J. Comput. Fluid Dyn. **11** (1998), no. 1-2, 3–25, Flow control and optimization. MR 1682727 Zbl 0940.76084

[22] B. F. Smith, P. E. Bjørstad, and W. D. Gropp, *Domain decomposition: parallel multilevel methods for elliptic partial differential equations*, Cambridge University Press, Cambridge, 1996. MR 98g:65003 Zbl 0857.65126

[23] A. Toselli and O. Widlund, *Domain decomposition methods—algorithms and theory*, Springer Series in Computational Mathematics, no. 34, Springer, Berlin, 2005. MR 2005g:65006 Zbl 1069.65138

[24] C. R. Vogel, *Computational methods for inverse problems*, Frontiers in Applied Mathematics, no. 23, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2002, With a foreword by H. T. Banks. MR 2003i:65004 Zbl 1008.65103

Received March 24, 2008.

XIAO-CHUAN CAI: cai@cs.colorado.edu
Department of Computer Science, University of Colorado at Boulder, Boulder, CO 80309, United States

SI LIU: si.liu@colorado.edu
Department of Applied Mathematics, University of Colorado at Boulder, Boulder, CO 80309, United States

JUN ZOU: zou@math.cuhk.edu.hk
Department of Mathematics, The Chinese University of Hong Kong, Shatin, N. T., Hong Kong, China