

Parameter Estimation with the Augmented Lagrangian Method for a Parabolic Equation¹

T. K. NILSSEN² AND X. C. TAI³

Communicated by P. Tseng

Abstract. In this paper, we investigate the numerical identification of the diffusion parameters in a linear parabolic problem. The identification is formulated as a constrained minimization problem. By using the augmented Lagrangian method, the inverse problem is reduced to a coupled nonlinear algebraic system, which can be solved efficiently with the preconditioned conjugate gradient method. Finally, we present some numerical experiments to show the efficiency of the proposed methods, even for identifying highly discontinuous parameters.

Key Words. Parameter estimation, inverse problems, parabolic equations, augmented Lagrangian methods, conjugate gradient methods.

1. Introduction

The purpose of this paper is to investigate some numerical methods for identifying efficiently the unknown spatial varying coefficient $q(x)$ from the following linear parabolic problem:

$$(u_t, v) + (q(x)\nabla u, \nabla v) = (f, v), \quad \forall v \in H_0^1(\Omega), t \in (0, T), \quad (1a)$$

$$u(x, 0) = u_0(x), \quad \text{in } \Omega. \quad (1b)$$

Here, Ω is a bounded domain in \mathbb{R}^d , $d \geq 1$, with piecewise smooth boundary $\partial\Omega$, (\cdot, \cdot) denotes the inner product of $L^2(\Omega)$, and $f(\cdot, t) \in H^{-1}(\Omega)$, for $t \in (0, T)$, $T < +\infty$, is a given source term.

In this work, we assume that we have sparse point observation of u , i.e. measurements of u at some single points $u_{\text{obs}}(x^i, t)$, $i = 1, \dots, N$, for

¹This work was partially supported by the Research Council of Norway, Grant NFR-128224/431.

²Postdoctoral Fellow, Scientific Computing, Simula Research Laboratory, Oslo, Norway.

³Professor, Department of Mathematics, University of Bergen, Bergen, Norway.

$t \in (0, T)$. These measurements may contain noise; u and q are obtained from solving a regularized output-least-squares problem.

Earlier, Keung and Zou (Ref. 1) have considered parameter estimation in (1), using an Armijo algorithm. Other studies of parameter estimation in parabolic systems can be found in Refs. 1–4. Here, the hybrid method of Refs. 5–9 is used, i.e. both the state variable u and the coefficient q are regarded as unknown variables and the equation is considered as a constraint. The augmented Lagrangian method is used to solve the constrained minimization problem. To find a saddle point for the augmented Lagrangian functional, we have to solve a coupled nonlinear system. In Chan and Tai (Ref. 5), the nonlinear systems were solved by the sequential linearization approach of Ref. 9. To solve the linearized equations, it is expensive to assemble the system matrices. In this work, we use also the linearization approach of Ref. 9, but the costs of assembling the matrices in the time dependent systems are even more expensive. Thus, we propose an approach that avoids the assembling of matrices. Another benefit of the new approach is that the meshes that we use for the state variable u and the coefficient q can be independent of each other. This makes the algorithm flexible w.r.t. varying the dimension of the space where q is approximated, which depends typically on the information available from measurements.

The paper is organized as follows. Section 2 presents how the numerical parameter estimation problem is solved with the augmented Lagrangian method. In section 3, we show how the CG method can be used to execute the steps in the augmented Lagrangian method without assembling matrices. Finally, in Section 4, we give some numerical experiments to show the efficiency of the method.

2. Augmented Lagrangian Method

First, we present a discretization for the forward problem and then specify the augmented Lagrangian approach in a discrete setting. The forward problem can be discretized in different ways. The augmented Lagrangian functional will vary for different approximations. In this paper, we use a finite-element method for spatial approximation and an implicit Euler scheme for the time variable. In real industrial applications, finite-difference or finite-volume methods may be used for the spatial variables and explicit schemes could be used for the time integration. Then, the augmented Lagrangian functional has to be modified correspondingly.

2.1. Approximations for the Forward Problem. For simplicity, assume that $\Omega \subset \mathbb{R}^d$ is a polyhedral domain and that \mathcal{T}^h is a regular triangulation

of Ω with simplicial elements [cf. Ciarlet (Ref. 10)]. The superscript h denotes the diameter of the largest simplex of the triangulation. Let V_h be the standard piecewise linear finite-element space over this triangulation, with functions vanishing on the boundary $\partial\Omega$. This is the finite-element space where u is defined.

To define the space for q , we let \mathcal{T}^H be a similar triangulation of Ω as the one above with either simplicial or rectangular elements. Let W_H^c denote the piecewise constant finite-element space over this triangulation and let

$$W_H = \{q \in W_H^c \mid 0 < q_{\min} \leq q \leq q_{\max} < \infty\},$$

where q_{\min} and q_{\max} are given. Note that the triangulations for V_h and W_H might differ. In practical applications, the dimension for V_h is normally required to be much higher than the dimension of W_H . In case that \mathcal{T}^h is a refined mesh of \mathcal{T}^H , the implementation is simpler.

For the temporal discretization, we divide the time interval $(0, T)$ into M equal subintervals by using $t^n = n\tau, n = 0, \dots, M$, with $\tau = T/M$. We initialize by setting

$$u_h^0 = I^h(u_0(x)) \in V_h,$$

where I^h is the linear interpolation operator into V_h using the nodal point values. The discretized solution u_h^n is then defined recursively by solving

$$\left((u_h^n - u_h^{n-1}) / \tau, v \right) + (q \nabla u_h^n, \nabla v) = (f^n, v), \quad \forall v \in V_h, \tag{2}$$

where $f^n = f(x, t^n)$. Equation (2) and the initial condition define

$$u_h = \left(u_h^0, \dots, u_h^M \right)^T \in (V_h)^{M+1}.$$

In the rest of the paper, we drop the subscript h .

In the augmented Lagrangian method, we consider the equation as a constraint. Below, we introduce the equation error [see (4)]. To minimize the equation error, we need a proper norm to measure it. We use the following two inner products to produce different measures for the equation error:

$$(u, v)_V = (u, v) + \tau(\nabla u, \nabla v), \tag{3a}$$

$$(u, v)_V = (u, v). \tag{3b}$$

The corresponding norm is $\|\cdot\|_V^2 = (\cdot, \cdot)_V$ respectively. We assume that the finite element mesh used for V_h is shape regular and quasiuniform so that

the inverse inequality for the finite-element functions is valid; see p. 124 and Eq. (3.2.28) of Ref. 10. Using the inverse inequality, in case that $\tau = O(h^2)$, it can be proved that the two norms produced by the two inner products are equivalent with an equivalence constant independent of h and τ for functions from V_h . In such cases, we will use the inner product (3b). When τ is large, then we need to use the inner product (3a). In order to evaluate the corresponding norm, we have to solve a large linear system. This can be avoided by using equivalent norms produced by multigrid or domain decomposition methods as in Ref. 11.

For $q \in W_H$ and $u \in (V_h)^{M+1}$, the equation error $e \in (V_h)^M$ is defined so that

$$(e^n, v)_V = (u^n - u^{n-1}, v) + \tau(q \nabla u^n, \nabla v) - \tau(f^n, v), \quad \forall v \in V_h. \tag{4}$$

We see that e^n depends on q, u^n, u^{n-1} . For any given $q \in W_H$ and $u \in (V_h)^{M+1}$, we say that (q, u) satisfies Equation (2) if $e^n = 0, \forall n$.

Let the operators $\Delta_h, \nabla_h,$ and ∇_h be the discretized counterparts for the Laplacian $\Delta,$ the divergence operator, and gradient operator ∇ respectively over the finite-element space V_h . If we use the inner product (3a), then the equation for e^n can be written in the following explicit form:

$$e^n(q, u) = (I - \tau \Delta_h)^{-1} (u^n - u^{n-1} - \tau \nabla_h \cdot (q \nabla_h u^n) - \tau f^n).$$

The operator $(I - \tau \Delta_h)^{-1}$ can be approximated by corresponding operators produced by domain decomposition or multigrid methods, analogous to the ones in Ref. 11. If we use the inner product (3b), the equation error is

$$e^n(q, u) = u^n - u^{n-1} - \tau \nabla_h \cdot (q \nabla_h u^n) - \tau f^n.$$

2.2. Discretized Minimization. The following minimization problem is solved to find the diffusion coefficient q from point observations of the state variable u :

$$\min \sum_n \tau E(u^n) + \beta R(q), \quad \text{over } (q, u) \in W_H \times (V_h)^{M+1}, \tag{5a}$$

$$\text{s.t. } e(q, u) = 0 \quad \text{and} \quad u^0 = I^h(u_0(x)), \tag{5b}$$

where

$$E(u^n) = \sum_i (1/2) \left(u^n(x^i) - u_{\text{obs}}^n(x^i) \right)^2 \tag{5c}$$

denotes the tracking part of the objective function, where

$$u_{\text{obs}}^n(x^i) = u_{\text{obs}}(x^i, t^n).$$

$R(q)$ is a regularization functional and β is a small positive parameter that can be determined by the mesh sizes and the noise level. In this paper, we use

$$R(q) = \|q\|_{L^2}^2 \quad \text{or} \quad R(q) = \|\nabla q\|_{(L^2)^2}^2.$$

Note that these regularizations are both quadratic in q . The term $\sum \tau E(u^n)$ should be understood as a discrete-time integration of $E(u)$, i.e. $\int_0^T E(u) dt$.

2.3. Augmented Lagrangian Method. We use an augmented Lagrangian method to solve the constrained minimization problem (5). The discretized augmented Lagrangian functional $L^c : W_H \times (V_h)^{M+1} \times (V_h)^M \rightarrow \mathbb{R}$ is

$$L^c(q, u, \lambda) = \sum_{n=1}^M \tau E(u^n) + \beta R(q) + \sum_{n=1}^M (\lambda^n, e^n)_V + \sum_{n=1}^M (c/2) \|e^n\|_V^2. \quad (6)$$

λ denotes the Lagrange multiplier associated with the constraint $e(q, u) = 0$ and $c > 0$ is a penalization constant. In the discrete setting, L^c has a saddle point and the saddle point is a minimizer for (5); see Refs. 6,9,12. Note that the sums in (6) start from $n = 1$. Thus, the initial condition comes into L^c only through e^1 .

We will use the modified Uzawa algorithm below to find the saddle points for L^c . The linear convergence for this algorithm was proved in Refs. 9,12.

Algorithm 2.1. Global Minimization Algorithm.

- Step 1. Choose initial values for $\lambda_0 \in (V_h)^M, u_0 \in (V_h)^{M+1}$ and set $k = 1$.
- Step 2. Find q_k from

$$L^c(q_k, u_{k-1}, \lambda_{k-1}) = \min_{q \in W_H} L^c(q, u_{k-1}, \lambda_{k-1}). \quad (7)$$

- Step 3. Set $u_k^0 = u^0$ and find $\{u_k^n\}_{n=1}^M$ from

$$L^c(q_k, u_k, \lambda_{k-1}) = \min_{u \in (V_h)^{M+1}} L^c(q_k, u, \lambda_{k-1}). \quad (8)$$

Step 4. Update the Lagrange multiplier by

$$\lambda_k = \lambda_{k-1} + ce(q_k, u_k).$$

If not converged, set $k = k + 1$ and go to Step 2.

The objective functionals of (8) and (7) are quadratics. Hence, the computation of the solutions amounts to solving linear (algebraic) systems of equations, respectively. Due to the tremendous size of the matrices, we can only assume to have matrix-times-vector products available. Consequently, we apply the CG method, which is well-known to operate efficiently with only matrix-times-vector products. The corresponding algorithm is called the global minimization algorithm, since it searches for the global minimum w.r.t. the time levels. In Section 3, we will introduce two algorithms which compute only local minima w.r.t. the time levels. We point out that the above notion of global minimization does not refer to global convergence, which is convergence of the algorithm for arbitrary initial guesses.

3. Implementation Issues with the CG Method

In this section, we study an efficient technique to solve the subproblems (7) and (8). We use the notations

$$L_q^c \cdot p = [\partial L^c(q, u, \lambda) / \partial q] \cdot p, \quad L_u^c \cdot w = [\partial L^c(q, u, \lambda) / \partial u] \cdot w$$

to denote the Gateaux derivatives of the functional $L^c(q, u, \lambda)$. The notations

$$\begin{aligned} L_{qq}^c \cdot (p, p) &= [\partial^2 L^c(q, u, \lambda) / \partial q^2] \cdot (p, p), \\ L_{uu}^c \cdot (w, w) &= [\partial^2 L^c(q, u, \lambda) / \partial u^2] \cdot (w, w) \end{aligned}$$

are used for the second-order derivatives.

The augmented Lagrangian functional $L^c(q, u, \lambda)$ is linear w.r.t. λ . For fixed (u, λ) , the functional $L^c(q, u, \lambda)$ is quadratic w.r.t. q ; for fixed (q, λ) , the functional $L^c(q, u, \lambda)$ is quadratic w.r.t. u . Thus, there must exist linear operators

$$\mathcal{A}(u): W_H \mapsto W_H, \quad \mathcal{B}(q): (V_h)^M \mapsto (V_h)^M$$

and functions

$$\gamma_1(u, \lambda) \in W_H, \quad \gamma_2(q, \lambda) \in (V_h)^M$$

such that

$$\partial L^c / \partial q = \mathcal{A}(u)q - \gamma_1(u, \lambda), \quad \partial L^c / \partial u = \mathcal{B}(q)u - \gamma_2(q, \lambda).$$

Due to the quadratic nature of the augmented Lagrangian functional, it is true that

$$(\mathcal{A}(u)p, p) = L_{qq}^c \cdot (p, p), \quad (\mathcal{B}(q)w, w) = L_{uu}^c \cdot (w, w),$$

for $p \in W_H$ and $w \in (V_h)^M$. In the implementations, $\mathcal{A}(u)$ and $\mathcal{B}(q)$ are matrices depending on u and q respectively; γ_1 and γ_2 are vectors depending on (u, λ) and (q, λ) respectively. Thus, the subproblems (7) and (8) are equivalent to solving equations of the following form:

$$\partial L^c / \partial q = \mathcal{A}(u)q - \gamma_1(u, \lambda) = 0, \quad \text{for (7),} \tag{9}$$

$$\partial L^c / \partial u = \mathcal{B}(q)u - \gamma_2(q, \lambda) = 0, \quad \text{for (8).} \tag{10}$$

In Chan and Tai (Ref. 5), the matrices and vectors corresponding to $\mathcal{A}, \mathcal{B}, \gamma_1, \gamma_2$ are assembled at each iteration and the linear systems are solved exactly. The cost of the assembling for time-dependent problems is larger. Therefore, we look for ways to solve the system without the assembling of matrices and vectors.

To solve the linear systems (9) and (10), we use the preconditioned conjugate gradient (PCG) method; see e.g. Hackbusch (Ref. 13). The systems is on the form $Ax = b$, where A is a symmetric positive-definite matrix, x is the unknown vector, and b is a right-hand-side vector. The preconditioned system can be written as

$$BAx = Bb,$$

where B is a symmetric positive-definite preconditioner. Note that we do not have to form the matrix A to implement the CG method. We need only an algorithm to calculate the residual vector $r = b - Ax$ and the scalar $X^T Ax$ for a given vector x . Similarly for B , we need only an algorithm to calculate Bx for given x .

In this work, domain decomposition methods are used as preconditioners. Then, only small subproblems defined on subdomains have to be solved.

3.1. Minimization with the CG Method. For the application of the CG method for solving Equation (9), we have to compute the vector $\mathcal{A}q - \gamma_1$ and the scalar $p^T \mathcal{A}p$ for given vectors p and q . Similarly, we have to

calculate the vector $\mathcal{B}u - \gamma_2$ and the scalar $w^T \mathcal{B}w$ for given vectors u and w to solve Equation (10).

From the definition of L^c , we get that

$$L_q^c \cdot p = \beta R_q(q) \cdot p + \sum_n (\partial e^n / \partial q) \cdot p, \lambda^n)_V + c \sum_n (\partial e^n / \partial q) \cdot p, e^n)_V.$$

We define

$$d_1^n = (\partial e^n / \partial q) \cdot p$$

and use the definition of e^n to see that

$$(d_1^n, v)_V = \tau(p \nabla u^n, \nabla v), \quad \forall v \in V_h;$$

thus,

$$L_q^c \cdot p = \beta R_q(q) \cdot p + \sum_n \tau(p \nabla u^n, \nabla(\lambda^n + c e^n)).$$

Let $\{\phi_j\}$ be the basis for W_H^c . Then, the residual is

$$r_1(x) = \sum_j r_1^j \phi_j(x),$$

where

$$r_1^j = L_q^c \cdot \phi_j.$$

Later, we let $\rho_1 = [r_1^j]$ denote the residual vector for Equation (9). When solving (9), u^n and λ^n are known and we need only to compute e^n for each n to get ρ_1

We use the quadratic property of L^c to see that

$$(\mathcal{A}p, p) = \beta R_{qq}(q) \cdot (p, p) + c \sum_n \|d_1^n\|_V^2.$$

Thus, we have to calculate d_1^n for all n to get $(\mathcal{A}p, p)$.

In order to use the CG method to solve (10), we define

$$d_2^n = (\partial e^n / \partial u) \cdot w.$$

We have

$$(d_2^n, v)_V = (w^n - w^{n-1}, v) + \tau(q \nabla w^n, \nabla v), \quad \forall v \in V_h.$$

The residual for Equation (10) is

$$\begin{aligned} L_u^c \cdot w &= \tau \sum_{n,i} (u^n(x^i) - u_{\text{obs}}^n(x^i))w^n(x^i) + \sum_n (d_2^n, \lambda^n + ce^n)_V \\ &= \tau \sum_{n,i} (u^n(x^i) - u_{\text{obs}}^n(x^i))w^n(x^i) \\ &\quad + \sum_n (w^n, (\lambda^n + ce^n) - (\lambda^{n+1} + ce^{n+1})) \\ &\quad + \sum_n \tau (q \nabla w^n, \nabla(\lambda^n + ce^n)). \end{aligned}$$

Here, we have defined

$$\lambda^{M+1} = ce^{M+1} = w^0 = 0,$$

which gives us the property

$$\sum_{n=1}^M (w^n - w^{n-1}, \lambda^n + ce^n) = \sum_{n=1}^M (w^n, (\lambda^n + ce^n) - (\lambda^{n+1} + ce^{n+1})).$$

Let $\{\psi_j\}$ be a basis for V_h . Let $\rho_2 = [r_2^{j,n}]$ be the residual vector for Equation (10), which contains residuals on all the time levels for all the nodal basis functions; i.e.,

$$\begin{aligned} r_2^{j,n} &= \tau \sum_i (u^n(x^i) - u_{\text{obs}}^n(x^i))\psi_j(x^i) + (\psi_j, (\lambda^n + ce^n) - (\lambda^{n+1} + ce^{n+1})) \\ &\quad + \tau (q \nabla \psi_j, \nabla(\lambda^n + ce^n)). \end{aligned}$$

The residual function for the time level n can be written as

$$r_2^n(x) = \sum_j r_2^{j,n} \psi_j(x).$$

Note that, when computing the directional derivatives d_1 and d_2 , we have to evaluate a stiffness matrix and eventually a domain decomposition version of $(I - \tau \Delta_h)^{-1}$, for each time level. This is roughly the same work as solving a parabolic equation.

For the second-order derivative, we have that

$$(\mathcal{B}w, w) = \tau \sum_{n,i} (w^n(x^i))^2 + c \sum_n \|d_2^n\|_V^2.$$

3.2. Efficient Minimization Algorithms. The Most time-consuming part of the minimization algorithm is the solution of (10). This is because the dimension of V_h is larger than the dimension of W_H and because u is a function of both space and time. In this subsection, we suggest two alternative minimization algorithms to Algorithm 2.1.

3.2.1. Marching Minimization Algorithm. For the forward problem (2), the solution for the time levels are computed in sequence, i.e. u^{n-1} is computed before u^n . However, in (10) all time levels are coupled together. Thus, we must solve a large system. In the following, we present an algorithm which computes the solution sequentially for each time level. The result is not the exact solution of (10), but it turns out to be a good approximation.

To be more precise, let us define

$$F(u^n, u^{n-1}) = \tau E(u^n) + (\lambda^n, e^n)_V + (c/2) \|e^n\|_V^2,$$

which means that

$$L^c(q, u, \lambda) = \sum_n F(u^n, u^{n-1}) + \beta R(q).$$

F does depend also on λ and q , but since we only use this notation for solving (10), we omit q and λ in $F(u^n, u^{n-1})$ for notational simplicity. The algorithm reads as follows.

Algorithm 3.1. Marching Minimization Algorithm.

Step 1. Choose initial values for $\lambda_0 \in (V_h)^M, u_0 \in (V_h)^{M+1}$ and set $k = 1$.

Step 2. Find q_k from

$$q_k = \arg \min_{q \in W_H} L^c(q, u_{k-1}, \lambda_{k-1}).$$

Step 3. Set $u_k^0 = u^0$ and find $\{u_k^n\}_{n=1}^M$ sequentially for $n = 1, 2, \dots, M$ such that

$$u_k^n = \arg \min_{v \in V_h} F(v, u_k^{n-1}). \tag{11}$$

Step 4. Update the Lagrange multiplier by

$$\lambda_k = \lambda_{k-1} + ce(q_k, u_k).$$

If not converged, set $k = k + 1$ and go to Step 2.

When solving (11), the newest values for q and λ are used.

Similar as above, we define

$$d_3^n = (\partial e^n / \partial u^n) \cdot w^n,$$

which satisfies

$$(d_3^n, v)_V = (w^n, v) + \tau(q \nabla w^n, \nabla v), \quad \forall v \in V_h.$$

The Gateaux derivative of $F(u^n, u^{n-1})$ w.r.t. u^n in the direction w^n is

$$\begin{aligned} F_{u^n} \cdot w^n &= \tau E_{u^n}(u^n) \cdot w^n + (d_3^n, \lambda^n)_V + c(d_3^n, e^n)_V \\ &= \tau \sum_i (u^n(x^i) - u_{\text{obs}}^n(x^i)) w^n(x^i) + (w^n, \lambda^n + ce^n) \\ &\quad + \tau(q \nabla w^n, \nabla(\lambda^n + ce^n)) \end{aligned}$$

and the second-order derivative is

$$F_{u^n u^n} \cdot (w^n, w^n) = \tau \sum_i (w^n(x^i))^2 + c \|d_3^n\|_V^2.$$

We can solve now (11) for each time level with the CG method.

3.2.2. Gauss-Seidel Algorithm. Here, we present a block Gauss-Seidel algorithm to approximate the solution of (10).

Algorithm 3.2. Block Gauss-Seidel Minimization Algorithm.

Step 1. Choose initial values for $\lambda_0 \in (V_h)^M, u_0 \in (V_h)^{M+1}$ and set $k = 1$.

Step 2. Find q_k from

$$q_k = \arg \min_{q \in W_H} L^C(q, u_{k-1}, \lambda_{k-1}).$$

Step 3. Set $\tilde{u}_0 = u_{k-1}$ and $m = 1$. While $\|\rho_2\| \geq \epsilon$, set $\tilde{u}_m^0 = u^0$ and find $\{\tilde{u}_m^n\}_{n=1}^M$ sequentially for $n = 1, 2, \dots, M$ such that

$$\tilde{u}_m^n = \arg \min_{v \in V_h} (F(v, \tilde{u}_m^{n-1}) + F(\tilde{u}_m^{n+1}, v)). \tag{12}$$

Set $m = m + 1$. When $\|\rho_2\| < \epsilon$, set $u_k = \tilde{u}_{m+1}$.

Step 4. Update the Lagrange multiplier by

$$\lambda_k = \lambda_{k-1} + ce(q_k, u_k).$$

If not converged, set $k = k + 1$ and go to Step 2.

Here, we have defined

$$\begin{aligned} \tilde{u}_m^{M+1} &= 0, \quad \forall m, \\ F(\tilde{u}^{M+1}, v) &= 0, \quad \forall v, \end{aligned}$$

for notational simplicity. $\|\rho_2\|$ is the L^2 -norm of the residual from (10) in the previous subsection and ϵ is a small number indicating the stopping tolerance. When solving (12), then q_k and λ_{k-1} are used. In the experiments, we set the maximum iteration number to be 20 in the while loop.

We use the same notations as in the previous subsection. The Gateaux derivative of $F(u^n, u^{n-1}) + F(u^{n+1}, u^n)$ w.r.t. u^n in the direction w^n is

$$\begin{aligned} &F_{u^n}(u^n, u^{n-1}) \cdot w^n + F_{u^n}(u^{n+1}, u^n) \cdot w^n \\ &= \tau \sum_i (u^n(x^i) - u_{\text{obs}}^n(x^i)) w^n(x^i) \\ &\quad + (w^n, ce^n + \lambda^n) + \tau(q \nabla w^n, \nabla(\lambda^n + ce^n)) - (w^n, ce^{n+1} + \lambda^{n+1}), \end{aligned}$$

and the second-order derivative is

$$\begin{aligned} &F_{u^n u^n}(u^n, u^{n-1}) \cdot (w^n, w^n) + F_{u^n u^n}(u^{n+1}, u^n) \cdot (w^n, w^n) \\ &= \tau \sum_i (w^n(x^i))^2 + \|d_3^n\|_V^2 + c \|w^n\|_V^2. \end{aligned}$$

4. Numerical Experiments

In this section, we present some 2D experiments for the proposed algorithms. We set

$$\Omega = (0, 1) \times (0, 1), \quad T = 0.01, \quad u_0(x) = \sin(\pi x_1) \sin(\pi x_2),$$

and q piecewise constant,

$$q(x) = i, \quad x \in \Omega_i, \quad \text{for } i = 1, \dots, 4, \tag{13}$$

where

$$\begin{aligned} \Omega_1 &= (0, 1/2] \times (0, 1/2], & \Omega_2 &= (1/2, 1) \times (0, 1/2], \\ \Omega_3 &= (0, 1/2] \times (1/2, 1), & \Omega_4 &= (1/2, 1) \times (1/2, 1). \end{aligned}$$

The source function is

$$f(x, t) = \sum_{i=1}^4 \delta(x - y_i) - 4\delta(x - y_5),$$

where $\{y_i\}_{i=1}^4$ are four fixed points near the corners, y_5 is the center of Ω , and δ is the Dirac delta function.

Ω is triangulated by first dividing it into squares of size $h \times h$. Then by the diagonal with positive slope, each square is divided into two triangles to get T^h . The number of time steps is $M = T/\Delta t$ and T^H is built up of $H \times H$ squares.

As a preconditioner for the PCG method, we have used domain decomposition to approximate the operator $(I - \tau \Delta)^{-1}$. The domain decomposition method is of the multiplicative Schwarz type with overlapping grid. This is tested with and without coarse grid. The coarse grid problem is solved exactly.

In the examples, we have plotted $\|q_k - q\|_{L^2}$ with increasing k value, where q is the true parameter. We run the Uzawa iterations to a desired accuracy or iteration number and stopped it by inspection of these plots.

The initial values of q and u in the CG method are the solution from the previous iteration of the Uzawa algorithm. In the first iteration of the Uzawa algorithm, we use the spatial linear interpolation of $u_{\text{obs}}^n(x^i)$ as an initial value of u . As an initial value for q in the first step of the Uzawa algorithm, we use q_0 equal to a constant. The constant that is used is the average of the exact parameter, i.e.,

$$q_0 = \int_{\Omega} q \, dx / |\Omega|.$$

In practical problems, sometimes one will have some estimate of the average. Our experiments have shown us that the algorithm is not sensitive to the initial value in q . The Lagrange multiplier is initially $\lambda_0 = 0$. The stopping criterion for the CG method is that the relative L^2 -norm of the residual is $\leq 10^{-6}$. The c -value is determined experimentally.

In the following examples, we have observed convergence when either the inner products (3a) or (3b) are used. We have seen that the CG method converges in less iterations when (3a) is used, but since (3b) is cheaper, this has been preferred in our examples. The constraints on W_H , $q_{\min} \leq q \leq q_{\max}$, were never active.

Example 4.1. The three suggested algorithms are compared for

$$H = 1/4, \quad h = 1/8, \quad M = 20, \quad T = 0.01, \quad \beta = 0 \text{ (no noise)}.$$

In the center of each square element of \mathcal{T}^H , there is one observation point, i.e. 16 observation points for u (or $N=16$) and 16 parameters representing q . The c -values were determined experimentally to $8 \cdot 10^{-5}$ in Algorithm 2.1, $2 \cdot 10^{-5}$ in Algorithm 3.1, and $8 \cdot 10^{-5}$ in Algorithm 3.2.

The convergence rates of q are shown in Figure 1. Algorithm 2.1 shows the best result, while Algorithm 3.1 seems to converge slightly slower. However, each iteration in Algorithm 3.1 is about five times faster than the iteration in Algorithm 2.1. Algorithm 3.2 converges slowest, is most time consuming for each iteration, and therefore is not to prefer.

From these experiments, it seems that Algorithm 3.1 is to prefer. Each iteration is very fast and its convergence rate is almost as good as in Algorithm 2.1. If we work on problems with more time steps, Algorithm 3.1 will be even faster compared to Algorithm 2.1.

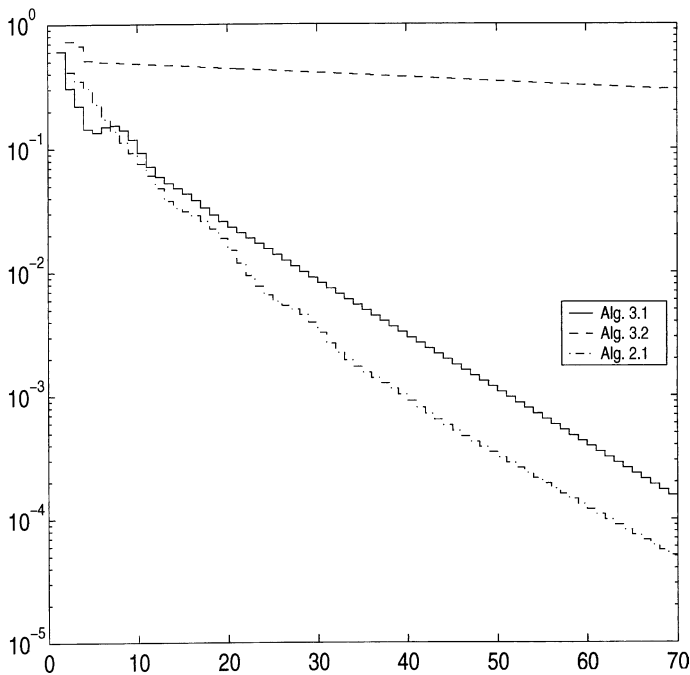


Fig. 1. $\|q_k - q\|_{L^2}$ versus k . Logarithmic scale on the vertical axis. The dashed curve is the result from Algorithm 3.2, the solid curve is the result from Algorithm 3.1, the dash-dotted curve comes from Algorithm 2.1.

Example 4.2. In this example, we test the algorithms when there is a multiplicative noise added to the observations; i.e.,

$$u_{\text{obs}}^n(x^i) = u^n(x^i) + \sigma u^n(x^i) \text{ rand}(i, n), \tag{14}$$

where $\text{rand}(i, n)$ is a vector of normally distributed numbers with expectation 0 and standard deviation 1; $\sigma = 10^{-3}$ is referred to as the noise level. The example is specified by

$$H = 1/12, \quad h = 1/24, \quad M = 10, \quad T = 0.01.$$

We have tested L^2 -norm and H^1 -seminorm regularizations, i.e.,

$$R(q) = \int_{\Omega} q^2 dx, \quad R(q) = \int_{\Omega} \nabla_q \cdot \nabla q dx,$$

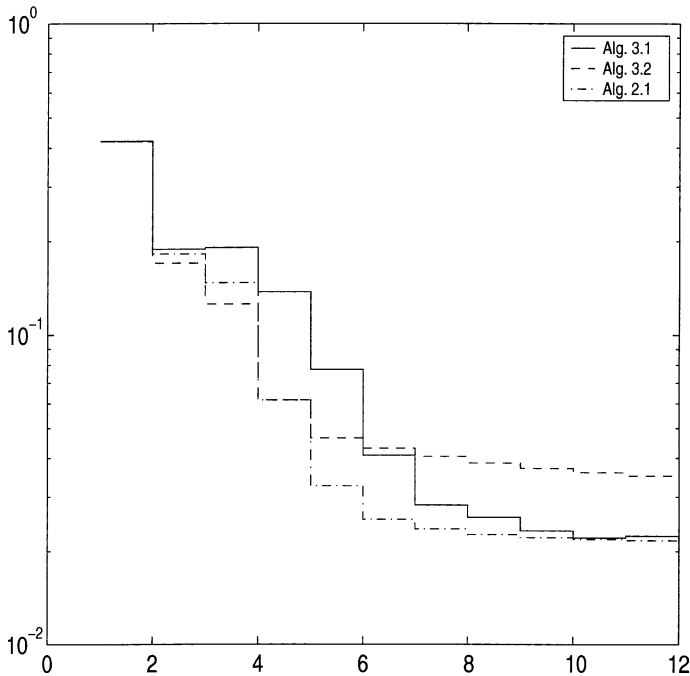


Fig. 2. $\|q_k - q\|_{L^2}$ versus k . Logarithmic scale on the vertical axis. The noise level is $\sigma = 10^{-3}$. The dashed curve is the result from Algorithm 3.2, the solid curve is the result from Algorithm 3.1, the dash-dotted curve comes from Algorithm 2.1.

respectively. For the L^2 -norm regularization, we get the best result when the regularization parameter is zero. But for the H^1 -seminorm regularization, we get an improvement when $\beta \neq 0$. The β -values are determined experimentally to $\beta = 1.6 \cdot 10^{-7}$ in Algorithm 2.1, $\beta = 3.5 \cdot 10^{-7}$ in Algorithm 3.1, and $\beta = 5 \cdot 10^{-10}$ in Algorithm 3.2.

The convergence rate of q is shown in Figure 2. The algorithms are stopped after 12 iterations. In Algorithm 2.1, the penalization parameter is $c = 1.6 \cdot 10^{-5}$ and it performs best; Algorithm 3.1 with $c = 10^{-5}$ is almost as good, but faster; Algorithm 3.2 with $c = 2 \cdot 10^{-7}$ seems to converge slowest.

Figure 3 shows the result after 12 iterations, when Algorithm 3.1 is used (cf. the solid curve in Figure 2). The relative L^2 error in q is

$$\|q_{12} - q\|/\|q\| \approx 0.0084.$$

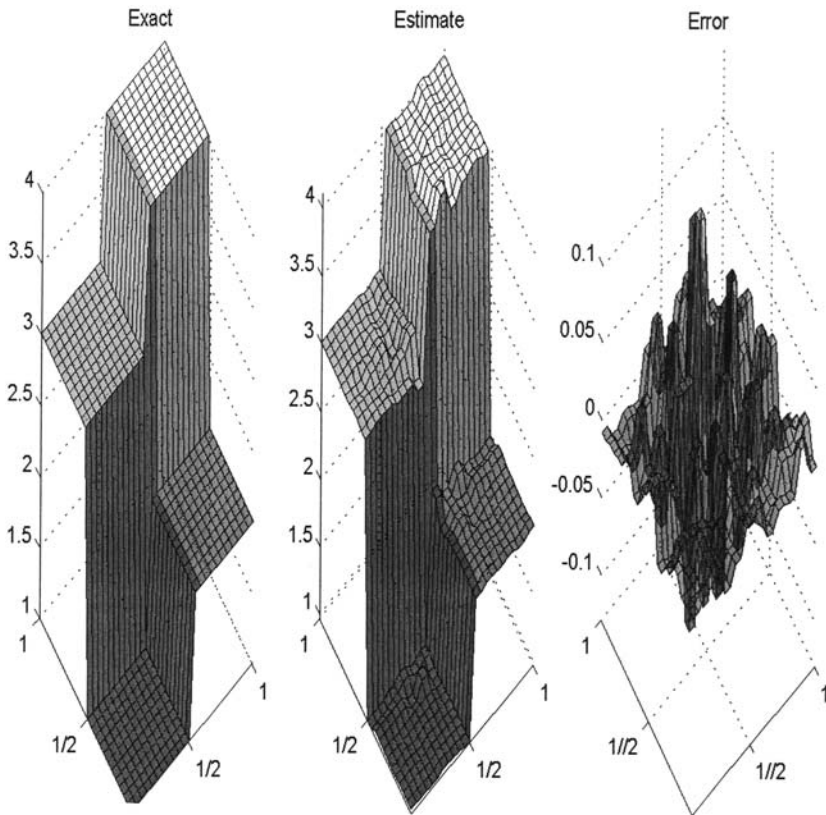


Fig. 3. Exact and estimated $q(x)$ and error after 12 iterations for noise level $\sigma = 10^{-3}$.

Example 4.3. In the above examples, the interfaces of the discontinuities for q matched the finite-element grid for q . Here, we show an example where that is not the case. The example is specified by

$$q(x) = i, \quad x \in \Omega_i, \quad i = 1, 2, 3,$$

where

$$\Omega_1 = (0, 1/3] \times (0, 1), \quad \Omega_2 = (1/3, 2/3] \times (0, 1), \quad \Omega_3 = (2/3, 1) \times (0, 1)$$

(see Figure 4) and

$$H = 1/16, \quad h = 1/48, \quad M = 10, \quad T = 0.01, \quad \beta = 0, \quad c = 1.6 \cdot 10^{-5}.$$

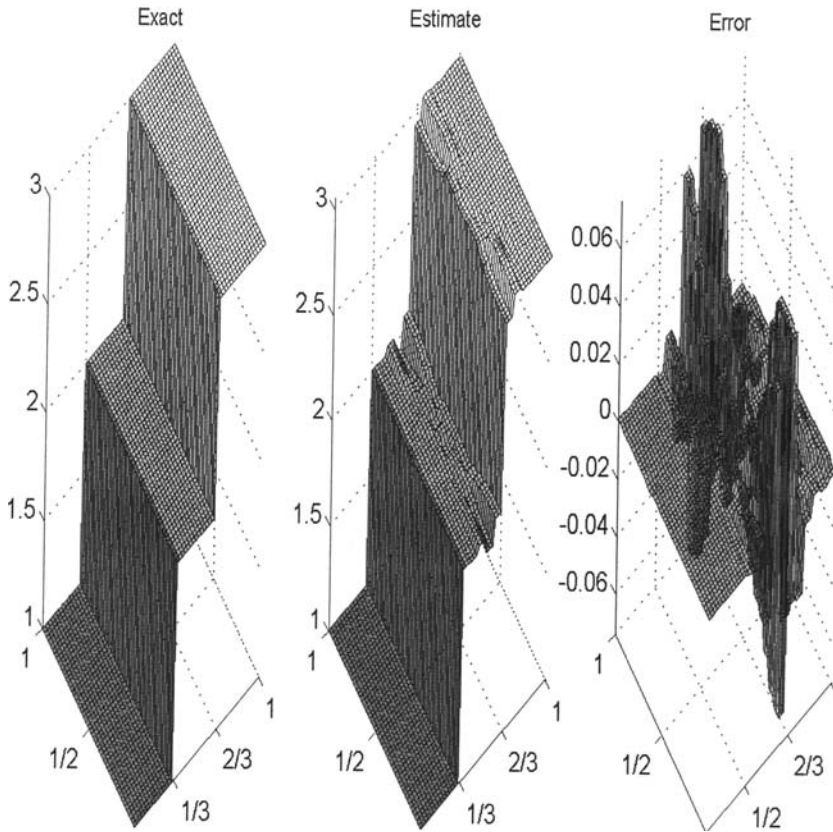


Fig. 4. Exact and estimated $q(x)$ and error after 97 iterations.

In this way, the $H \times H$ grid does not match the discontinuities of $q(x)$. The observation points are located in the upper right corner of each square of the $H \times H$ grid or, more specifically,

$$x^{i+(j-1)16} = ((i-1/3)H, (j-1/3)H), \quad i, j = 1, \dots, 16.$$

In Figure 4, we observe that the error is relatively small compared to $q(x)$.

5. Conclusions

In this paper, we have studied the augmented Lagrangian method for identifying the diffusion parameters q from sparse point observation of u in Equation (1). When using the Uzawa algorithm, finding u is the most time consuming part. We have suggested three different algorithms for this and Algorithm 3.1 seems to perform best.

References

1. KEUNG, Y. L., and ZOU, J., *Numerical Identifications of Parameters in Parabolic Systems*, Inverse Problems, Vol. 14, pp. 83–100, 1998.
2. ENGL, H. W., and ZOU, J., *A New Approach to Convergence Rate Analysis of Tikhonov Regularization for Parameter Identification in Heat Conduction*, Inverse Problems, Vol. 16, pp. 1907–1923, 2000.
3. GOU, B., and ZOU, J., *An Augmented Lagrangian Method for Parameter Identification in Parabolic Systems*, Journal of Mathematical Analysis and Applications, Vol. 263, pp. 49–68, 2001.
4. KÄRKKÄINEN, T., *Error Estimates for Distributed Parameter Identification in Parabolic Problems with Output Least Squares and Crank-Nicolson Method*, Applications of Mathematics, Vol. 42, pp. 259–277, 1997.
5. CHAN, T. F., and TAI, X. C., *Identification of Discontinuous Coefficients from Elliptic Problems Using Total Variation Regularization*, SIAM Journal on Scientific Computing, Vol. 25, pp. 881–904, 2003.
6. ITO, K., and KUNISCH, K., *The Augmented Lagrangian Method for Parameter Estimation in Elliptic Systems*, SIAM Journal on Control and Optimization, Vol. 28, pp. 113–136, 1990.
7. KEUNG, Y. L., and ZOU, J., *An Efficient Linear Solver for Nonlinear Parameter Identification Problems*, SIAM Journal on Scientific Computing, Vol. 22, pp. 1511–1526, 2000.
8. KUNISCH, K., and PEICHL, G., *Estimation of a Temporally and Spatially Varying Diffusion Coefficient in a Parabolic System by an Augmented Lagrangian Technique*, Numerische Mathematik, Vol. 59, pp. 473–509, 1991.
9. KUNISCH, K., and TAI, X. C., *Sequential and Parallel Splitting Methods for Bilinear Control Problems in Hilbert Spaces*, SIAM Journal on Numerical Analysis, Vol. 34, pp. 91–118, 1997.

10. CIARLET, P. G., *The Finite-Element Method for Elliptic Problems*, North-Holland Publishing Company, Amsterdam, Holland, 1978.
11. TAI, X.-C., FRØYEN, J., ESPEDAL, M. S., and CHAN, T. F., *Overlapping Domain Decomposition and Multigrid Methods for Inverse Problems*, Domain Decomposition Methods, 10th International Conference on Domain Decomposition Methods, Contemporary Mathematics, Vol. 218, pp. 523–529, 1998; see also URL: <http://www.mi.uib.no/~tai/>.
12. CHEN, Z., and ZOU, J., *An Augmented Lagrangian Methods for Identifying Discontinuous Parameters in Elliptic Systems*, SIAM Journal on Control and Optimization, Vol. 37, pp. 892–910, 1999.
13. HACKBUSCH, W., *Iterative Solution of Large Sparse Systems of Equations*, Applied Mathematical Sciences, Springer Verlag, New York, NY, Vol. 95, 1994.