Department
of
APPLIED MATHEMATICS

Parameter estimation with the augmented Lagrangian method for a parabolic equation.

by

Trygve K. Nilssen and Xue-Cheng Tai

Report no. 153 January 2001

UNIVERSITY OF BERGEN
Bergen, Norway

Department of Mathematics
University of Bergen
5008 Bergen
Norway

ISSN 0084-778x

Parameter estimation with the augmented Lagrangian method for a parabolic equation.

by

Trygve K. Nilssen and Xue-Cheng Tai

Report no. 153

January 2001

PARAMETER ESTIMATION WITH THE AUGMENTED LAGRANGIAN METHOD FOR A PARABOLIC EQUATION

TRYGVE K. NILSSEN AND XUE-CHENG TAI

ABSTRACT. In this paper, we investigate the numerical identification of the diffusion parameters in parabolic problems. 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 method, even for identifying highly discontinuous parameters.

1. INTRODUCTION

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

$$u_t - \nabla \cdot (q(x)\nabla u) = f(x,t) \quad \text{in } \Omega \times (0,T), \tag{1.1}$$

with the initial-boundary condition

$$u(x,0) = u_0(x) \ \text{ in } \Omega \quad \text{and} \quad u(x,t) = g(x,t) \ \text{ on } \partial\Omega \times (0,T). \tag{1.2}$$

Here Ω can be any bounded domain in R^d, $d \geq 1$, with piecewise smooth boundary $\partial\Omega$, and $f(\cdot,t) \in H^{-1}(\Omega)$ $t \in (0,T)$, is a given source term.

The identification process is carried out in a way that the solution u matches its observation data u_d optimally. In many practical applications, it is easier to measure the solution u at various points in the medium than to measure the parameter $q(x)$ itself. In this work we assume that we have available measurements of u at some single points for all time $u_d(x_i,t)$, $\quad i = 1,\ldots,n; \quad t \in (0,T)$. The measurements may contain noise.

For the identification problem, the hybrid method of [4, 5, 6, 1] will be used, i.e. both the state variable u and the coefficient q will be regarded as unknown variables and the equation is considered as a constraint. The augmented Lagrangian method will be used to solve the constrained minimization problem. To find a saddle point for the

This work was partially supported by the Research Council of Norway (NFR), under grant 128224/431.

Lagrangian functional, we need to solve some coupled nonlinear systems. In Chan and Tai [1], the nonlinear systems were solved by the sequential linearization approach of [6]. To solve the linearized equations, it is rather expensive to assemble the system matrices. In this work, we also use the linearization approach of [6] for time dependent problems with sparse point observations. The costs of assembling the matrices in the linearized systems are even more expensive. Thus we have proposed an approach that avoids assembling matrices. Another good point about the approach is that the meshes we use for the state variable u and the coefficient q can be independent of each other. This makes the algorithm very flexible with respect to varying the dimension of the space where q is approximated, which is dependent of the information available from the measurements.

The rest of this paper is organized as follows: The next section presents the numerical scheme that is used to solve the forward problem, and it shows how the numerical parameter estimation problem is formulated and solved with the augmented Lagrangian method. Thereafter we show how the conjugate gradient method can be used to execute the steps in the augmented Lagrangian method in a way which avoids assembling matrices. Finally we give some numerical experiments to show the efficiency of the method.

2. The augmented Lagrangian method

We shall first present an approximation 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 Lagrangian functional will be different with different approximations. In this paper, we are going to use a finite element method for spacial 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 spacial variables and explicit schemes could be used for the time integration. The Lagrangian functional then need to be modified correspondingly.

2.1. Approximations for the forward problem.

For simplicity, assume that $\Omega \subset R^d$ is a polyhedral domain and $\mathcal{T}^h$ is a regular triangulation of Ω with simplicial elements (cf. Ciarlet [3]). 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. This is the finite element space where u and f are 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 denotes the piecewise constant finite element space over this triangulation. 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 much simpler.

To fully discretize the parabolic system, we also need a time discretization. To get that, we divide the time interval $(0,T)$ into M equal subintervals by using $t^n = n\tau$, $n = 0,\ldots,M$ 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:

$$\left(\frac{u_h^n - u_h^{n-1}}{\tau}, v\right) + (q\nabla u_h^n, \nabla v) = (f^n, v), \quad \forall v \in V_h. \tag{2.1}$$

In the above and also later, $(\cdot,\cdot)$ is used as the inner product of $L^2(\Omega)$ and $f^n = f(x,t_n)$. The equation (2.1) defines

$$u_h = \begin{pmatrix} u_h^0 \\ \vdots \\ u_h^M \end{pmatrix} \in (V_h)^{M+1}.$$

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

In the augmented Lagrangian method, we need to regard the equation as a constraint. To minimize the equation error, we need to use a proper norm to measure the equation error. In our simulations, we have used the following two inner products to produce different measures for the equation error:

$$a).\quad (u,v)_V = (u,v) + \tau(\nabla u, \nabla v), \qquad b).\quad (u,v)_V = (u,v). \tag{2.2}$$

The corresponding norm is $\|\cdot\|_V^2 = (\cdot,\cdot)_V$. When $\tau = O(h^2)$, 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 (2.2.b). When τ is big, then we need to use the inner product (2.2.a). In order to evaluate this norm, we need to solve a large linear system. This can be avoided by using equivalent norms produced by multigrid or domain decomposition methods as in [7].

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

$$\begin{aligned}(e^n, v)_V = &\left(u^n - u^{n-1}, v\right) \\ &+\tau(q\nabla u^n, \nabla v) - \tau(f^n, v), \quad \forall v \in V_h, \quad \forall n > 0.\end{aligned} \tag{2.3}$$

We see that e^n depends on q, u^n and u^{n-1}. For any given $q \in W_H$ and $u \in (V_h)^{M+1}$, we say that (q,u) satisfies the equation (2.1) if $e^n = 0, \forall n$.

In an explicit form, the equation for e^n in fact can be written in the following form if we use the inner product (2.2.a):

$$e^n(q,u) = (I - \tau\Delta_h)^{-1}(u^n - u^{n-1} - \tau\nabla_h \cdot (q\nabla_h u^n) - \tau f^n), \tag{2.4}$$

where the subscript h denotes that we use a discretized version of the operator. The operator $(I - \tau\Delta_h)^{-1}$ can be replaced by some corresponding operators produced by domain decomposition or multigrid methods, see [7]. If we use the inner product (2.2.b), the equation error is:

$$e^n(q,u) = u^n - u^{n-1} - \tau\nabla_h \cdot (q\nabla_h u^n) - \tau f^n. \tag{2.5}$$

2.2. **Discretized minimization.** In case that only point observations are available for the state variable u, we shall solve the following minimization problem to find the corresponding coefficient q:

$$\min_{e(q,u)=0} \tau \sum_n E(u^n) + \beta R(q) \tag{2.6}$$

subject to $q \in W_H$ and $u \in (V_h)^{M+1}$ satisfying $u^0 = I^h(u_0(x))$. Here

$$E(u^n) = \sum_i \frac{1}{2}|u^n(x_i) - u_d^n(x_i)|^2 \tag{2.7}$$

and $R(q)$ is a regularization functional which will be specified later, and β is a small positive parameter that will be determined by the mesh sizes and the noise level.

2.3. **The augmented Lagrangian method.** We shall use an augmented Lagrangian method to solve the constrained minimization problem (2.6). The discretized augmented Lagrange functional $L_c : W_H \times (V_h)^{M+1} \times (V_h)^M \to R$ is:

$$L_c(q,u,\lambda) = \tau \sum_n E(u^n) + \beta R(q) + \sum_n (\lambda^n, e^n)_V + \sum_n \frac{c}{2}\|e^n\|_V^2 .$$

In the above, $c > 0$ is a penalization constant which does not need to be very big. In the discrete setting, it is known that L_c has a saddle point and the saddle point is a minimizer for (2.6), see [4, 6, 2].

We will use the following modified Uzawa algorithm to find saddle-points for this functional. A linear convergence for this algorithm has been proved in [6, 2].

Algorithm 2.1. *(The global minimization algorithm)*

(1) *Choose initial values for* $\lambda_0, u_0 \in V_h$ *and set k=1.*

(2) *Find* q_k *from*

$$L_c(q_k, u_{k-1}, \lambda_{k-1}) = \min_{q \in W_H} L_c(q, u_{k-1}, \lambda_{k-1}). \tag{2.8}$$

(3) *Set* $u_k^0 = u^0$ *and find* $u_k = \{u_k^n\}_{n=1}^M$ *from*

$$L_c(q_k, u_k, \lambda_{k-1}) = \min_{u \in V_h} L_c(q_k, u, \lambda_{k-1}). \tag{2.9}$$

(4) *Update the Lagrange-multiplier as*

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

If not converged: Set k=k+1 and GOTO (2).

3. Implementation issues with the conjugate gradient method

In this section, we study an efficient method to solve the two subminimization problems in the modified Uzawa algorithm. We will use the notations $L_c' \cdot q = L_c'(q,u,\lambda)\cdot p = \frac{\partial L_c(q,u,\lambda)}{\partial q}\cdot p$ and $L_c' \cdot w = L_c'(q,u,\lambda)\cdot w = \frac{\partial L_c(q,u,\lambda)}{\partial u}\cdot w$ to denote the Gateaux derivatives of the functional $L_c(q,u,\lambda)$. Note that when writing $L_c' \cdot p$, the p indicates that we take the derivative with respect to q in the direction p. Similary the w in $L_c' \cdot w$ indicates that we take the derivative with respect to u in the direction w. The notations $L_c''(q,u,\lambda)\cdot(p,p) = \frac{\partial^2 L_c(q,u,\lambda)}{\partial q^2}\cdot(p,p)$ and $L_c''(q,u,\lambda)\cdot(w,w) = \frac{\partial^2 L_c(q,u,\lambda)}{\partial u^2}\cdot(w,w)$ are used for the second order derivatives.

The augmented Lagrangian functional $L_c(q,u,\lambda)$ is linear with respect to λ. For fixed (u,λ), the functional $L_c(q,u,\lambda)$ is quadratic with respect to q, and for fixed (q,λ), the functional $L_c(q,u,\lambda)$ is quadratic with respect to u. Thus there must exist linear operators $\mathcal{A}(u) : W_H \mapsto W_H$ and $\mathcal{B}(q) : V_h \mapsto V_h$ and functions $\alpha_1(u,\lambda) \in W_H$ and $\alpha_2(q,\lambda) \in V_h$ such that

$$\frac{\partial L_c}{\partial q} = \mathcal{A}(u)q - \alpha_1(u,\lambda), \tag{3.1}$$

$$\frac{\partial L_c}{\partial u} = \mathcal{B}(q)u - \alpha_2(q,\lambda). \tag{3.2}$$

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

$$(\mathcal{A}(u)p,p) = L_c''(q,u,\lambda)\cdot(p,p) \qquad \forall p \in V_h, \tag{3.3}$$

$$(\mathcal{B}(q)w,w) = L_c''(q,u,\lambda)\cdot(w,w) \qquad \forall w \in W_H. \tag{3.4}$$

In the implementations, $\mathcal{A}(u)$ and $\mathcal{B}(q)$ are matrices depending on u and q respectively, and α_1 and α_2 are vectors depending on (u,λ) and (q,λ) respectively. Thus the subproblems (2.8) and (2.9) are equivalent to solving equations of the following form:

$$\frac{\partial L_c}{\partial q} = \mathcal{A}(u)q - \alpha_1(u,\lambda) = 0, \tag{3.5}$$

$$\frac{\partial L_c}{\partial u} = \mathcal{B}(q)u - \alpha_2(q,\lambda) = 0. \tag{3.6}$$

In Chan and Tai [1], the matrices $\mathcal{A}, \mathcal{B}$ and the vectors α_1, α_2 are assembled at each iteration and the linear systems are solved exactly. The cost of the assembling for time dependent problems is getting

too high. Therefore we look for ways to solve the system without assembling the matrices and the vectors. In the next subsection we will present the preconditioned conjugate gradient method, and then show that this method can be used to solve the equations $\mathcal{A}q = \alpha_1$ and $\mathcal{B}u = \alpha_2$.

3.1. **The preconditioned conjugate gradient method.** The preconditioned conjugate gradient method solves the equation $Ax = b$ with a symmetric positive definite preconditioner B. The algorithm is written as:

$$
\begin{aligned}
&k = 0,\ x_0 = 0,\ r_0 = b,\ z_0 = B^{-1}r_0,\ p_1 = z_0,\\
&\textbf{while}\quad r_k \neq 0,\\
&\qquad \alpha_k = r_{k-1}^T z_{k-1}/p_k^T A p_k,\\
&\qquad x_k = x_{k-1} + \alpha_k p_k,\\
&\qquad r_k = r_{k-1} - \alpha_k A p_k = b - A x_k,\\
&\qquad Solve \quad B z_k = r_k,\\
&\qquad \beta_{k+1} = r_k^T z_k / r_{k-1}^T z_{k-1},\\
&\qquad p_{k+1} = z_k + \beta_{k+1} p_k,\\
&\textbf{end}\\
&x = x_k.
\end{aligned}
$$

In the algorithm, we do not need to form the matrix A. For simulations, we just need subroutines to calculate $b - Ax$ and $p^T Ap$ for given vectors x and p.

Domain decomposition and multigrid methods shall be used for the preconditioner B. When using domain decomposition methods, only very small subproblems defined on the subdomains need to be solved. If multigrid method is used, we do not need to solve any systems of linear algebraic equations. We just need to update the residuals of some equations over all the nodal points from the different levels.

In order to use conjugate gradient methods to solve equation (2.8) and (2.9), it is enough to design some subroutines to calculate $(\mathcal{A}p, p)$, $(\mathcal{B}w, w)$ and the corresponding residuals for the two equations with given p and w.

3.2. **Minimization with the conjugate gradient method.** All we need to solve equation (3.5), is to calculate $\mathcal{A}q - \alpha_1$ and $p^T \mathcal{A} p$ for given vectors q and p. Similarly, we need to calculate $\mathcal{B}u - \alpha_2$ and $w^T \mathcal{B} w$ for given vectors u and w to solve equation (3.6).

¿From the definition of L_c, we get that

$$
L_c' \cdot p = \beta R'(q) \cdot p + \sum_n \left(\frac{\partial e^n}{\partial q} \cdot p, \lambda^n\right)_V + c \sum_n \left(\frac{\partial e^n}{\partial q} \cdot p, e^n\right)_V. \tag{3.7}
$$

Thus, we need to calculate d_1^n for all n to get ().

In order to use conjugate gradient method to solve (3.6), we d Define $d_1^n = \frac{\partial e^n}{\partial q} \cdot p$. From the definition of e^n, we see that d_1^n satisfies

$$(d_1^n, v)_V = \tau(p\nabla u^n, \nabla v) \qquad \forall v \in V_h, \tag{3.8}$$

and thus

$$\begin{aligned} L_c' \cdot p &= \beta R'(q) \cdot p + \sum_n (d_1^n, \lambda^n + ce^n)_V \\ &= \beta R'(q) \cdot p + \sum_n \tau(p\nabla u^n, \nabla(\lambda^n + ce^n)). \end{aligned} \tag{3.9}$$

Assume that $\{\phi_j\}$ are the basis functions for W_H and $\mathbf{r}_1 = [r_1(j)]$ is the residual vector for equation (3.5), i.e. $r_1(j) = L_c' \cdot \phi_j$. From the calculations above, we see that $r_1(j)$ can be calculated by

$$r_1(j) = \beta R'(q) \cdot \phi_j + \sum_n \tau(\phi_j \nabla u^n, \nabla(\lambda^n + ce^n)).$$

When solving (3.5), u^n and λ^n are known and we only need to compute e^n for each n to get the residual $\mathbf{r}_1$ from the above formula.

To calculate $(\mathcal{A}p, p)$ for a given $p \in W_H$, we use the quadratic property (3.3) to get that

$$(\mathcal{A}p, p) = L_c''(q, u, \lambda) \cdot (p, p) = \beta R''(q) \cdot (p, p) + c \sum_n (d_1^n, d_1^n)_V \tag{3.10}$$

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

In order to use conjugate gradient method to solve (3.6), we define $d_2^n = \frac{\partial e^n}{\partial u} \cdot w$. We see that d_2^n satisfies

$$(d_2^n, v)_V = (w^n - w^{n-1}, v) + \tau(q\nabla w^n, \nabla v) \qquad \forall v \in V_h. \tag{3.11}$$

Similarly, we have the following formulations for calculating the residual for equation (3.6):

$$
\begin{aligned}
L_c' \cdot w &= \sum_n \tau E'(u^n) \cdot w^n + \sum_n (\frac{\partial e^n}{\partial u} \cdot w, \lambda^n)_V + c \sum_n (\frac{\partial e^n}{\partial u} \cdot w, e^n)_V \\
&= \tau \sum_{n,i} (u^n(x_i) - u_d^n(x_i)) w^n(x_i) + \sum_n (d_2^n, \lambda^n + ce^n)_V \\
&= \tau \sum_{n,i} (u^n(x_i) - u_d^n(x_i)) w^n(x_i) + \\
&\quad \sum_n (w^n - w^{n-1}, \lambda^n + ce^n) + \sum_n \tau (q \nabla w^n, \nabla(\lambda^n + ce^n)) \\
&= \tau \sum_{n,i} (u^n(x_i) - u_d^n(x_i)) w^n(x_i) \\
&\quad + \sum_n (w^n, (\lambda^n + ce^n) - (\lambda^{n+1} + ce^{n+1})) \\
&\quad + \sum_n \tau (q \nabla w^n, \nabla(\lambda^n + ce^n)),
\end{aligned} \tag{3.12}
$$

where we have defined $\lambda^{M+1} = ce^{M+1} = w^0 = 0$ to simplify the notation. For the second order derivative it is true that:

$$
\begin{aligned}
(\mathcal{B}w, w) = L_c'' \cdot (w, w) &= \tau \sum_n E''(u^n) \cdot (w, w) + c \sum_n (d_2^n, d_2^n)_V \\
&= \tau \sum_{n,i} (w^n(x_i))^2 + c \sum_n (d_2^n, d_2^n)_V.
\end{aligned} \tag{3.13}
$$

Assume that $\{\psi_j\}$ are the basis functions for V_h and $\mathbf{r}_2 = [r_2(n,j)]$ is the residual vector for equation (3.6) which contains residuals on all the time levels for all the nodal basis functions. Then $r_2(n,j)$ can be calculated by

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

These calculations can be used in the conjugate gradient method to solve the equations $\mathcal{A}u = \alpha_1$ and $\mathcal{B}q = \alpha_2$ to execute (3.5) and (3.6) in the modified Uzawa algorithm.

3.3. **Efficient minimization algorithms.** The most time consuming part of the minimization algorithm is the solving of (3.6), i.e. the minimization of u. This is because $u(x,t)$ is a function of both space and time, and therefore have most degrees of freedom. Usually the dimension of the space V_h is bigger than the dimension of W_H. In this section we suggest two alternative minimization algorithms to the one

presented in the section above. The new minimization algorithms will not minimize (2.9) exactly as in Algorithm 2.1. Instead, we are trying to use some cheaper and approximate solver for the sub-minimization problem (3.6).

3.3.1. *Matching minimization algorithm.* For the forward problem (2.1), it is known that u^{n-1} must be computed before we can compute u^n. This is not correct for the sub-minimization problem (3.6). For (3.6), all the u^n are coupled to each other. If we first compute u^1 and then take the computed u^1 to compute u^2 as described in the following and continue, the obtained solution $u = \{u^n\}_{n=0}^M$ is not a minimizer for (3.6), but is a rather good approximation for the minimizer of (3.6).

To be more precise, let us define

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

It is clear that $F(u^n, u^{n-1})$ also depends on λ and q. Since we only use this notation for the solving of (3.6), we will omit q and λ in $F(u^n, u^{n-1})$ for notational simplicity. It is easy to see that

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

The following algorithm will be used as a replacement for Algorithm 2.1:

Algorithm 3.1. *(The matching minimization algorithm)*

(1) *Choose initial values for* $\lambda_0, u_0 \in V_h$ *and set k=1.*

(2) *Find* q_k *from*

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

(3) *Set* $u_k^0 = u^0$ *and find* $u_k = \{u_k^n\}_{n=1}^M$ *sequentially for* $n = 1, 2, \cdots M$ *such that*

$$u_k^n = \arg\min_{v \in V_h} F(v, u_k^{n-1}). \tag{3.14}$$

(4) *Update the Lagrange-multiplier as*

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

If not converged: Set k=k+1 and GOTO (2).

When solving (3.14), the newest values for q and λ are used. Step (3) in Algorithm 3.1 defines $u_k = \{u_k^n\}_{n=0}^M$ sequentially for the time steps.

To use the conjugate gradient method to solve this new minimization problem we should do some calculations similar to those in the previous section. The difference is that we now take the Gateaux derivative in the direction of one time level w^n instead of in all time levels.

We define $d_3^n = (e^n)' \cdot w^n = \frac{\partial e^n}{\partial u^n} \cdot w^n$. Then d_3^n satisfies

$$(d_3^n, v)_V = (w^n, v) + \tau(q\nabla w^n, \nabla v) \qquad \forall v \in V_h. \tag{3.15}$$

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

$$\begin{aligned} F' \cdot w^n &= \tau E'(u^n) \cdot w^n + (\frac{\partial e^n}{\partial u^n} \cdot w^n, \lambda^n)_V + c(\frac{\partial e^n}{\partial u^n} \cdot w^n, e^n)_V \\ &= \tau \sum_i (u^n(x_i) - u_d^n(x_i)) w^n(x_i) + (d_3^n, \lambda^n + ce^n)_V \\ &= \tau \sum_i (u^n(x_i) - u_d^n(x_i)) w^n(x_i) \\ &\quad + (w^n, \lambda^n + ce^n) + \tau(q\nabla w^n, \nabla(\lambda^n + ce^n)), \end{aligned} \tag{3.16}$$

and the second order derivative is

$$\begin{aligned} F'' \cdot (w^n, w^n) &= \tau E''(u^n) \cdot (w^n, w^n) + c(d_3^n, d_3^n)_V \\ &= \tau \sum_i (w^n(x_i))^2 + c(d_3^n, d_3^n)_V. \end{aligned} \tag{3.17}$$

3.3.2. *A Gauss-Seidel algorithm.* Another alternative to find an approximate solution for (3.6) is to use the following block Gauss-Seidel algorithm to compute u_k^n:

Algorithm 3.2. *(The block Gauss-Seidel minimization algorithm)*

(1) *Choose initial values for $\lambda_0, u_0 \in V_h$ and set k=1.*

(2) *Find q_k from*

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

(3) *Set $\tilde{u}_0 = u_{k-1}$ and $m = 1$.*
While $\|\mathbf{r}_2\| \geq \epsilon$ do:
Set $\tilde{u}_m^0 = u^0$ and find $\tilde{u}_m = \{\tilde{u}_m^n\}_{n=1}^M$ sequentially for $n = 1, 2, \cdots M$ such that

$$\tilde{u}_m^n = \arg\min_{v \in V_h} (F(v, \tilde{u}_m^{n-1}) + F(\tilde{u}_{m-1}^{n+1}, v)). \tag{3.18}$$

$m = m + 1$.
End while.
Set $u_k = \tilde{u}_m$.

(4) *Update the Lagrange-multiplier as*

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

If not converged: Set k=k+1 and GOTO (2).

Here $\|\mathbf{r}_2\|$ is the L^2-norm of the residual from the previous section (3.6), and ϵ is the stopping criteria. When solving (3.18), the newest values for q and λ are used.

We use the same notations as in the previous subsection. The Gateux derivative of $F(u^n, u^{n-1}) + F(u^{n+1}, u^n)$ with respect to u^n in the direction w^n is calculated:

$$\begin{aligned}
&F'(u^n, u^{n-1}) \cdot w^n + F'(u^{n+1}, u^n) \cdot w^n \\
&= \tau \sum_i (u^n(x_i) - u_d^n(x_i)) w^n(x_i) \\
&\quad + (w^n, \lambda^n + ce^n) + \tau(q\nabla w^n, \nabla(\lambda^n + ce^n)) \\
&\quad + ((e^{n+1})' \cdot w^n, ce^{n+1} + \lambda^{n+1})_V \\
&= \tau \sum_i (u^n(x_i) - u_d^n(x_i)) w^n(x_i) \qquad (3.19) \\
&\quad + (w^n, ce^n + \lambda^n) + \tau(q\nabla w^n, \nabla(\lambda^n + ce^n)) \\
&\quad - (w^n, ce^{n+1} + \lambda^{n+1}),
\end{aligned}$$

and the second order derivative is calculated:

$$\begin{aligned}
&F''(u^n, u^{n-1}) \cdot (w^n, w^n) + F''(u^{n+1}, u^n) \cdot (w^n, w^n) \\
&= \tau \sum_i (w^n(x_i))^2 + c(d_2^n, d_2^n)_V + c(w^n, w^n)_V. \qquad (3.20)
\end{aligned}$$

4. Numerical experiments

We now show some numerical experiments on the proposed method for parameter identification. For the tests, we have taken $\Omega = [0,1] \times [0,1]$, $T = 0.01$, $u_0(x) = \sin(\pi x)\cos(\pi y)$, $g(x) = 0$ and $q(x)$ is piecewise constant:

$$q(x) = \begin{cases} q_1, & x \in [0, 0.5] \times [0, 0.5] \\ q_2, & x \in [0, 0.5] \times [0.5, 1] \\ q_3, & x \in [0.5, 1] \times [0, 0.5] \\ q_4, & x \in [0.5, 1] \times [0.5, 1]. \end{cases} \qquad (4.1)$$

In the examples $q_i = i$, $i = 1, \ldots, 4$ unless otherwise defined. The source function is

$$f(x) = \sum_{i=1}^{4} \delta(x - x_i) - 4\delta(x - x_5), \qquad (4.2)$$

where x_i for $i = 1, \ldots, 4$ are the corners and x_5 is the center of Ω and δ is Diracs delta function.

The domain is triangulated by first dividing it into squares of size $h \times h$. Then each square is divided into two triangles by the diagonal with positive slope to get $\mathcal{T}^h$. The element functions $u(x,t)$ and $f(x,t)$ are defined over this triangulation with linear finite element functions. The number of time steps is $M = \frac{T}{\Delta t}$. Square meshes $\mathcal{T}^H$ are used for approximating q, and H is used to denote the mesh size. Moreover, the finite element functions for q are piecewise constants over each square.

In the implementations, the mesh $\mathcal{T}^h$ that we use for approximating u, is always a refinement of the mesh $\mathcal{T}^H$ we use for q.

With the numerical method described in Section 2, the forward problem can be solved. The solution from this, u, will then be used as a source for the observations that our algorithms will use to recover the permeability, q.

In Example 4.5 we will add normally distributed noise to the observations in a multiplicative way, i.e.:

$$u_d(x_i,t) = u(x_i,t) + \sigma\, u(x_i,t)\, \mathrm{rand}(x_i,t). \tag{4.3}$$

Here $\mathrm{rand}(x_i,t)$ is a vector of normally distributed numbers with expectation 0 and standard deviation 1. We refer to σ as the noise level.

In the figures in the following examples we illustrate the convergence rates of the Uzawa algorithms. In all examples we plot $\|q_k - q\|_{L^2}$ with increasing k-value, where q is the true permeability. In Example 4.1 we also illustrate the convergence rates of u in L^2-norm, i.e. we plot $\int_0^T \|u_k - u\|_{L^2} dt$. In the examples the Uzawa algorithm has been stopped by inspection of these plots.

As an initial value for q, we use q_0 equal to a constant. The constant that is used is the average of the exact permeability, i.e: $q_0 = \frac{1}{|\Omega|}\int_\Omega q\, dx$. The initial value for u is the spatial linear interpolation of $u_d(x_i,t)$. The c-value is found by trial and error.

In the following examples we have observed convergence when either the inner products (2.2.a) or (2.2.b) are used. We have seen that the conjugate gradient method converges in fewer iterations when (2.2.a) is used, but since (2.2.b) is cheaper this has been preferable in our examples.

For simplicity, we have taken $R(q) = \int_\Omega q^2 dx$. For examples without noise we can set the regularization parameter to $\beta = 0$.

As a preconditioner B for the conjugate gradient method we have used domain decomposition to approximate the operator $(I - \tau\Delta)^{-1}$. This is tested with and without coarse grid.

The stopping criteria for the conjugate gradient method is that the relative L^2-norm of the residual is $\leq \epsilon$. In the following examples we have chosen $\epsilon = 10^{-6}$

In all examples except Example 4.4 we have used $H = \frac{1}{4}$, $h = \frac{1}{8}$, $M = 20$ and $T = 0.01$. In the center of each square element of $\mathcal{T}^H$, there is one observation point. That means we have 16 observation points for u_d and 16 parameters representing q.

Example 4.1. In the first example we use the global minimization Algorithm 2.1. In this example the c-value was set to $8 \cdot 10^{-5}$. The convergence rate of q is shown in Figure 1, and the convergence rate of u is shown in Figure 2.

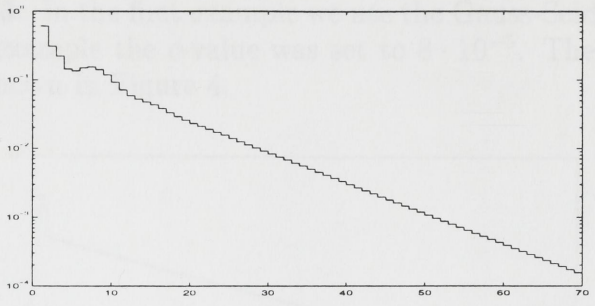

FIGURE 1. $\|q_k - q\|_{L^2}$ versus k. Logarithmic scale on the vertical axis.

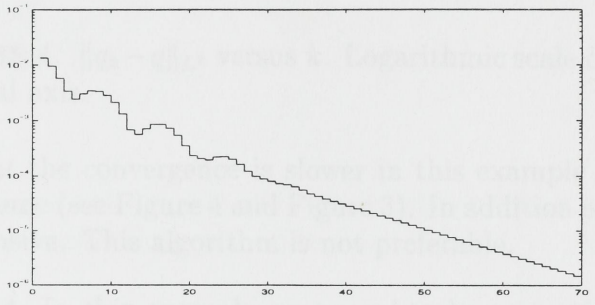

FIGURE 2. $\int_0^T \|u_k - u\|_{L^2} dt$ versus k. Logarithmic scale on the vertical axis.

Example 4.2. In this example we use the matching minimization Algorithm 3.1. The c-value was set to $8 \cdot 10^{-5}$. The convergence rate of q is shown in Figure 3.

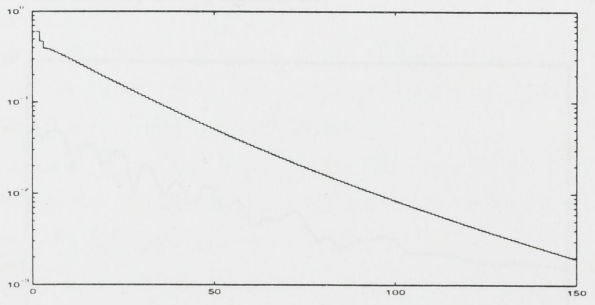

FIGURE 3. $\|q_k - q\|_{L^2}$ versus k. Logarithmic scale on the vertical axis.

We see that this algorithm takes more iterations to converge (see Figure 3). Since each iteration is much cheaper than in the previous example, this algorithm is much quicker.

Example 4.3. In the first example we use the Gauss-Seidel Algorithm 3.2. In this example the c-value was set to $8 \cdot 10^{-5}$. The convergence rate of q is shown in Figure 4.

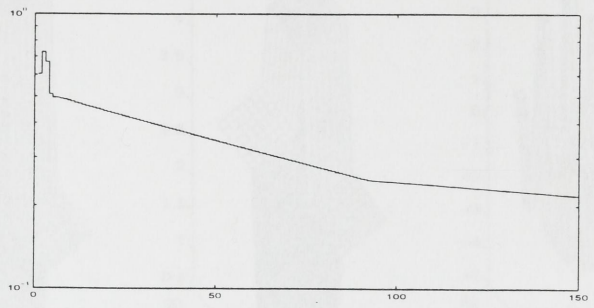

FIGURE 4. $\|q_k - q\|_{L^2}$ versus k. Logarithmic scale on the vertical axis.

We see that the convergence is slower in this example than for the matching scheme (see Figure 4 and Figure 3). In addition each iteration is more expensive. This algorithm is not preferable.

Example 4.4. In this example we see what happens if the number of parameters representing q is bigger than the number of observation points for u. The example is specified with $H = \frac{1}{8}$, $h = \frac{1}{16}$ $M = 10$, $T = 0.01$ and $c = 2 \cdot 10^{-6}$. $H = \frac{1}{8}$ gives 64 parameters representing q. The observation points u_d is taken to be the corners of $\mathcal{T}^H$ in the interior of Ω, i.e. we have 49 uniformly distributed observation points for u. The global minimization Algorithm 2.1 is used to identify $q(x)$. We see that it takes more iterations to converge and that the convergence is unstable, see Figure 5.

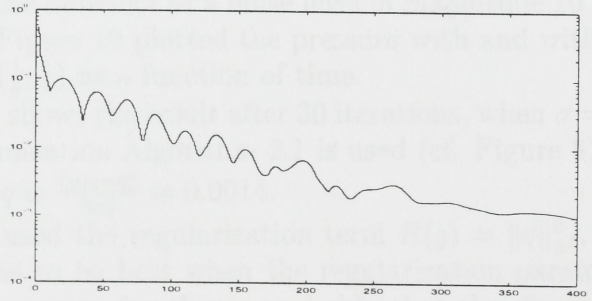

FIGURE 5. $\|q_k - q\|_{L^2}$ versus k. Logarithmic scale on the vertical axis. We have used 64 parameters representing q and 49 observation points for u.

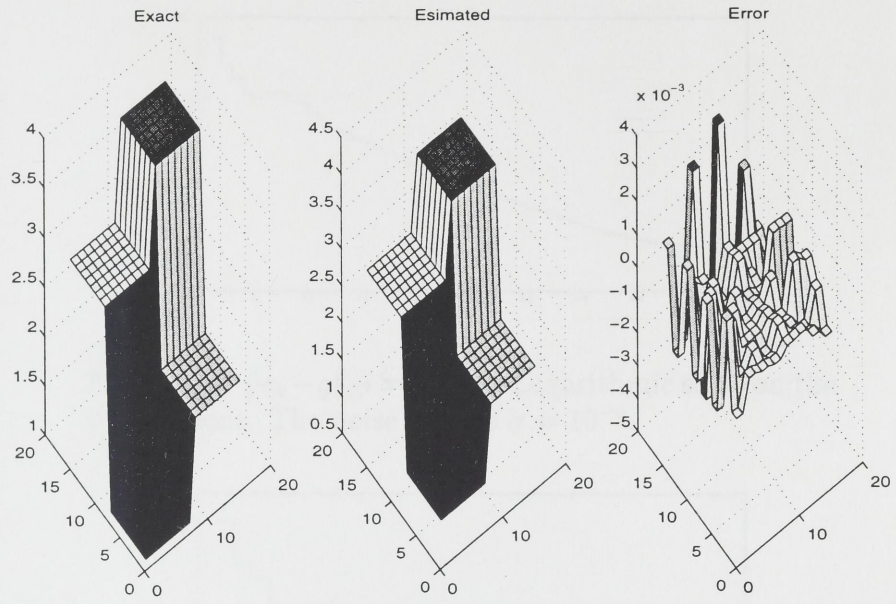

FIGURE 6. The exact- and approximate solution and error in permeability $q(x)$ after 400 iterations.

Example 4.5. We repeat the test in Example 4.1 and Example 4.2, but add noise to the observations (see Equation (4.3)). The convergence rate of q when the noise level is $\sigma = 10^{-3}$ is shown in Figure 7, and the same convergence rate when the noise level is $\sigma = 10^{-2}$ is shown in Figure 8. Here the global minimization Algorithm 2.1 is used to identify $q(x)$.

In Figure 9 we show the convergence rate when the matching minimization Algorithm 3.1 is used, and the noise level is $\sigma = 10^{-2}$. The c-value was here set to $2 \cdot 10^{-5}$.

To show the influence of a noise level of magnitude 10^{-2} on the data, we have in Figure 10 plotted the pressure with and without noise in a point $x_1 = (\frac{1}{8}, \frac{1}{8})$ as a function of time.

Figure 11 shows the result after 30 iterations, when $\sigma = 10^{-2}$ and the global minimization Algorithm 2.1 is used (cf. Figure 8). The relative L^2 error in q is $\frac{\|q_{30}-q\|}{\|q\|} \approx 0.0014$.

We have used the regularization term $R(q) = \|q\|^2_{L^2}$. However, the results seems to be best when the regularization parameter is chosen $\beta = 0$. The reason for this is probably that the dimension on $\mathcal{T}^H$ is relatively small in our examples.

Example 4.6. In oil reservoirs the permeability does often have very large jumps. In this example we try Algorithm 2.1 with permeability as described in (4.1) with $q_i = 10^{i-3}$, $i = 1, \ldots, 4$.

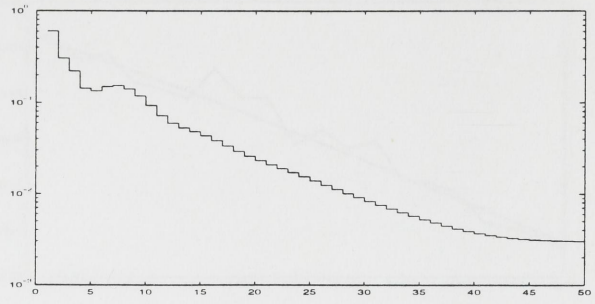

FIGURE 7. $\|q_k - q\|_{L^2}$ versus k. Logarithmic scale on the vertical axis. The noise level is $\sigma = 10^{-3}$.

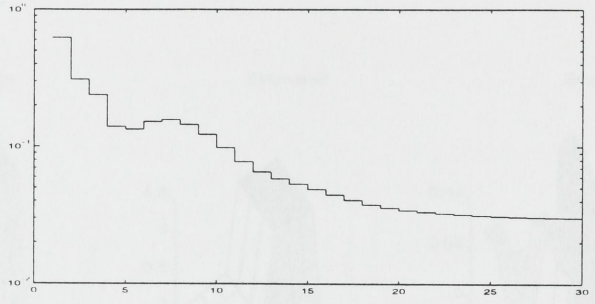

FIGURE 8. $\|q_k - q\|_{L^2}$ versus k. Logarithmic scale on the vertical axis. The noise level is $\sigma = 10^{-2}$.

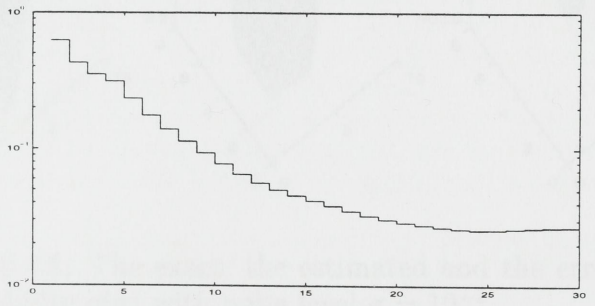

FIGURE 9. $\|q_k - q\|_{L^2}$ versus k. Logarithmic scale on the vertical axis. The noise level is $\sigma = 10^{-2}$ and the matching minimization Algorithm 3.1 is used.

In this example the c-value was set to $2.6 \cdot 10^{-5}$. The convergence rate of q is shown in Figure 12. We see that the convergence is a little bit slower than in the previous examples. In addition every iteration is about twice as costly as in Example 4.1, because the conjugate gradient method converges slower.

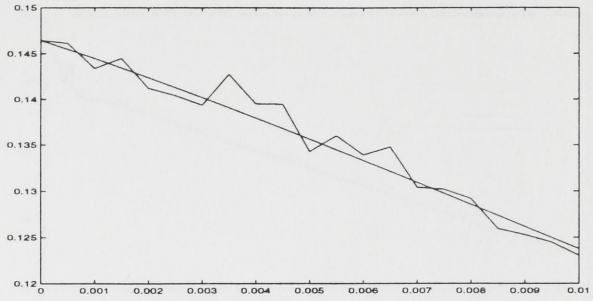

FIGURE 10. The pressure with and without noise in position $x_1 = (\frac{1}{8}, \frac{1}{8})$ i.e. $u(x_1, t)$ and $u_d(x_1, t)$, $t \in (0, T)$. The noise level is $\sigma = 10^{-2}$.

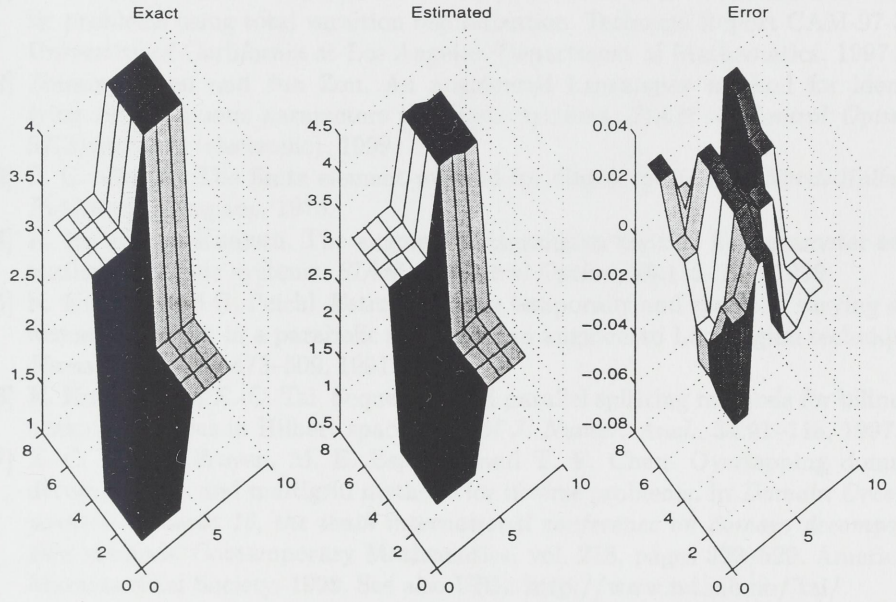

FIGURE 11. The exact, the estimated and the error in permeability $q(x)$ with noise level $\sigma = 10^{-2}$.

5. CONCLUSION

We have seen how the augmented Lagrangian method can be used to solve parameter estimation problems in parabolic PDEs. When using the Uzawa algorithm, the minimization for the pressure is the most time consuming part. In this paper we have suggested three different algorithms for this minimization. The global minimization algorithm performs well in all examples, but the matching scheme is much quicker.

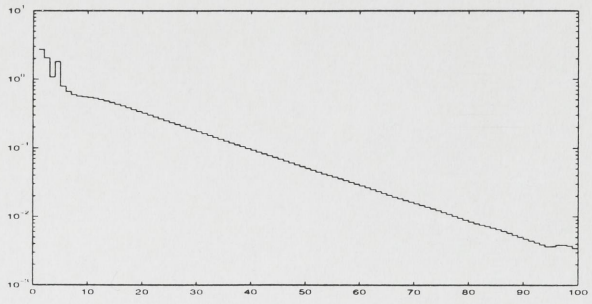

FIGURE 12. $\|q_k - q\|_{L^2}$ versus k. Logarithmic scale on the vertical axis.

REFERENCES

[1] T. F. Chan and X.-C. Tai. Identification of discontinuous coefficient from elliptic problems using total varaition regularization. Technical Report CAM-97-35, University of Carlifornia at Los Angeles, Department of Mathematics, 1997.

[2] Zhiming Chen and Jun Zou. An augmented Lagrangian method for identifying discontinuous parameters in elliptic systems. *SIAM J. Control Optim.*, 37(3):892–910 (electronic), 1999.

[3] P. G. Ciarlet. The finite element method for elliptic problems. *North-Holland Publishing Company*, 1978.

[4] K. Ito and K. Kunisch. The augmented lagrangian method for parameter estimation in elliptic systems. *SIAM J. Control Optim.*, 28:113–136, 1990.

[5] K. Kunisch and G. Peichl. Estimation of a temporally and spatially varying diffusion coefficient in a parabolic system by an augmented Lagrangian technique. *Numer. Math*, 59:473–509, 1991.

[6] K. Kunisch and X.-C. Tai. Sequential and parallel splitting methods for bilinear control problems in Hilbert spaces. *SIAM J. Numer. Anal.*, 34:91–118, 1997.

[7] X.-C. Tai, J. Froyen, M. E. Espedal, and T. F. Chan. Overlapping domain decomposition and multigrid methods for inverse problems. In *Domain Decomposition methods 10, the tenth international conference on domain decomposition methods*, Contemporary Mathematics, vol. 218, pages 523–529. American Mathematical Society, 1998. See also URL: http://www.mi.uib.no/~tai/.

DEPARTMENT OF MATHEMATICS, UNIVERSITY OF BERGEN, JOHANNES BRUNSGT. 12, 5007 BERGEN, NORWAY

E-mail address: TrygveKastberg.Nilssen@mi.uib.no

DEPARTMENT OF MATHEMATICS, UNIVERSITY OF BERGEN, JOHANNES BRUNSGT. 12, 5007 BERGEN, NORWAY

E-mail address: Xue-Cheng.Tai@mi.uib.no