# Parameter Identification by Regularization for Surface Representation via the Moving Grid Approach

STEFAN KINDERMANN*AND ANDREAS NEUBAUER*

Institut für Industriemathematik, Johannes-Kepler-Universität,
A-4040 Linz, Austria

**Abstract.** We consider the identification of a diffusion parameter in a second order elliptic equation in two dimensions by interior measurements. The diffusion parameter is assumed to have discontinuities. For its reconstruction we propose regularization algorithms with an adaptive grid. The grid is adapted according to a measure of the smoothness of the regularized solution. For the numerical computation we compare several iterative methods such as the minimal error method, the steepest descent method and an inexact iteratively regularized Gauss-Newton method. The computations show that these algorithms can effectively identify the discontinuites.

## 1. Introduction

Our interest lies in the identification of a possibly discontinuous diffusion coefficient $\tilde{\gamma}$ defined on a domain $\Omega \subset \mathbb{R}^2$ in the equation

$$-\mathrm{div}(\tilde{\gamma}\nabla u) = f \quad u_{|\partial\Omega} = 0 \tag{1.1}$$

where $f \in L^2(\Omega)$ is given, and $u(x), x \in \Omega$ is measured.

To ensure ellipticity, we assume that positive constants $\gamma_1, \gamma_2$ exist such that $\gamma_1 \leq \tilde{\gamma}(x) \leq \gamma_2$ almost everywhere. We want to find that parts of $\tilde{\gamma}$ which differ from a constant background diffusivity which we assume to be 1 without loss of generality. Hence, our unknown is $\gamma$ with $\tilde{\gamma} = 1 + \gamma$. In the sequel, we denote the nonlinear operator that maps $\gamma = \tilde{\gamma} - 1$ to the solution of (1.1) by $F$. The problem we are faced with is an ill-posed equation

$$F(\gamma) = u, \tag{1.2}$$

where $u$ are the given data.

In many cases a standard regularization using Hilbert space norms (see, e.g., [4]) implemented for example by Tikhonov regularization does not give satisfactory results, since the discontinuites are either smeared out, if regularization is too strong or oscillations occur when the regularization norm is too weak.

In particular for discontinuous solutions the use of the BV-seminorm has been quite effective for this kind of problems (cf., e.g., [3, 10]).

*Supported by the Austrian Science Foundation Funds under grand SFB F013/F1317

However the BV-functional suffers from the drawback that it is not differentiable, so usually a differentiable approximation to this functional is used. Moreover, BV-regularization shows the so-called staircase effect [15], i.e., regularized solutions have the tendency to become piecewise constant.

Our motivation to use an adaptive grid for discontinuous solutions comes from the work on regularization by curve and surface representations (cf. [7, 8, 9]). Here the discontinuous functions are regarded as curves or surfaces. Regularization is applied to their parameterizations. In the discretized case this has the effect that the unknown $\gamma$ is always defined on a grid, which changes with the iteration and is adapted to the regularized solution.

In [5, 6] it was shown that in one and two dimensions $H^1$-functions and a suitably adaptive grid may be used to approximate any BV-function in the sense of weak convergence.

The methods in [7, 8, 9, 13, 14] used a grid which is defined via optimization problems. Unfortunately, in higher dimensions, these optimization functionals are rather flat leading to slow convergence. The resulting grid has the property that the mesh size is small whereever the solution exhibits discontinuities, as expected. So instead of computing the grid via optimization it seems more efficient to adapt the meshsize directly to the smoothness of $\gamma$. This adaption can be implemented efficiently by the deformation method.

We show that with this moving grid method we can use standard algorithms from regularization theory in Hilbert spaces and still obtain good results for discontinuous solutions with a small extra amount of recalculating the grid in each step.

## 2. Moving grid method

### 2.1. General algorithm

For a numerical approach to problem (1.2), we have to set up a discrete approximation to $\gamma$ by, for instance, finite element functions defined on a suitable grid.

In particular, if $\gamma$ has discontinuities, and if the grid is kept fixed a reasonable approximation of $\gamma$ requires a small meshsize leading to a large number of unknowns. On the other hand, an adaptive grid allows a better resolution of discontinuites with a moderate number of variables. Thus, it seems advantageous to combine regularization with an adaptive grid.

Moreover, by our method we may use a discretization of $\gamma$ that is smoother than $\gamma$ itself, for example continuous ansatz functions even for discontinuous $\gamma$. Since the grid will be adaptive to the solution, the gridsize will be small whereever $\gamma$ has jumps, and this compensates the approximation error at the nonsmooth parts.

In our examples we work with weakly differentiable ansatz functions. The grid will be made finer, wherever some measure of smoothness indicates a large gradient of $\gamma$ which is regarded as possible lack of differentiablility.

This idea of a moving grid has been used for numerical computations in partial differential equations, see, e.g., [16]. For instance, in the framework of hyperbolic equations the development of shock waves and the corresponding lack of smoothness of the solutions may not be handled efficiently by a fixed grid. In the context of ill-posed problems, an adaptive grid approach has been successfully applied in [11] to linear integral equations.

We restrict ourselves to a moving grid, i.e., the adaptive grid is a transformation of a fixed, uniform one. Therefore, we need a transformation function $\phi$ which is one-to-one and onto on $\Omega$

$$\phi : \quad \Omega \to \Omega. \tag{2.1}$$

A sufficient condition for a $C^1$-function $\phi$ to fulfill these conditions is that

$$\phi(\partial\Omega) = \partial\Omega \quad \text{and} \quad \det D\phi > 0 \quad \text{in } \Omega \tag{2.2}$$

If $c$ is defined on $\Omega$ on a fixed uniform grid, then we can find an approximation to $\gamma$ on an adaptive grid by $c(\phi^{-1})$, where $\phi$ is an appropriate transformation function (again defined on a uniform grid).

We briefly describe the main ideas of our algorithm: in each step we find an approximation $c_n(\phi_n^{-1})$ of the solution, which is obtained by applying regularization to the equation

$$F(c(\phi_n^{-1})) = u \tag{2.3}$$

with fixed $\phi_n$. In the next step we compute an error estimator, which measures the smoothness of $c_n(\phi_n^{-1})$. Then we calculate a new transformation function $\phi_{n+1}$ depending on this error estimation.

Thus, the general steps of the algorithm look as follows:

1. Start with a uniform grid and the identity as transformation function $\phi_0(\xi) = \xi \in \Omega \subset \mathbb{R}^2$, $n = 0$.

2. Compute $c_n$ by regularization of the equation

$$F(c(\phi_n^{-1})) = u,$$

set $\gamma_n := c_n(\phi_n^{-1})$.

3. If a stopping criteria is satisfied set $\gamma = \gamma_n$, otherwise

4. update the transformation function

$$\phi_{n+1} = T(\phi_n, c_n), \tag{2.4}$$

where $T$ is the method of choice to define the moving grid; go to step 2

A detailed description of steps 2 and 4 is given below.

### 2.2. Regularization method

Note that in step 2 we apply regularization only to the function $c$ and not to $c(\phi_n^{-1})$. This motivation comes from the previous mentioned idea of regularization for surface representations, where it has been observed that $c(\xi) = \gamma(\phi(\xi))$ can be chosen in $H^1$ even for functions $\gamma$ being merely in BV.

Using the estimates in [9], it can be shown that $F$ is a continuous and Fréchet-differentiable operator from $H^1$ to $L^2$. Hence, we may use, e.g., Tikhonov regularization with the $H^1$-(semi-)norm as regularization term. In this case $c_n$ is computed by the minimization problem

$$c_n = \operatorname{argmin}_{c \in H^1(\Omega)} J(c, \phi_n) \tag{2.5}$$
$$J(c, \phi_n) := \|F(c(\phi_n^{-1})) - u_\delta\|^2 + \alpha \|\nabla c\|^2_{L^2(\Omega)}. \tag{2.6}$$

Here $u_\delta$ denotes the measured data, possibly contaminated with noise, where $\delta$ is the noise level, i.e.

$$\|u - u_\delta\|_{L^2} \leq \delta.$$

With fixed $\phi_n$, (2.5) is a convergent regularization method for the considered problem (cf. [4]).

If $\phi_n$ satisfies (2.2), then the minimization problem (2.5) can also be written as

$$c_n(\phi_n^{-1}) = \operatorname{argmin}_{w \in H^1} \|F(w) - u\|^2 + \alpha \|([D\phi_n]^{-T} \nabla w)(\det D\phi_n^{-1})^{\frac{1}{2}}\|^2_{L^2(\Omega)} \tag{2.7}$$

with $w = c(\phi_n^{-1})$. Here and in the following $D\phi$ denotes the Jacobian matrix of $\phi$.

(2.7) indicates that each regularization step can be seen as usual Tikhonov regularization with a weighted norm that is adapted to the grid.

Note that in (2.7) $\gamma_n = c_n(\phi_n^{-1})$ is always in $H^1$. This does not contradict our aim to approximate discontinuous coefficients $\gamma \notin H^1$, since the algorithm does not yield a uniform bound of $\|\gamma_n\|_{H^1}$ for all $n$. In fact, only $\|([D\phi_n]^{-T} \nabla \gamma_n)(\det D\phi_n^{-1})^{\frac{1}{2}}\|_{L^2(\Omega)}$ will be bounded.

Although our algorithm uses $\phi_n$ where $\det D\phi_n(\xi) > 0$ always holds in $\Omega$, after some iterations regions will occur, where $\det D\phi_n$ is numerically close to 0. These will be the parts of $\gamma_n$ corresponding to the discontinuites of $\gamma$.

If we allowed generalized diffeomorphisms with $\det D\phi_n = 0$ on some part of $\Omega_0 \subset \Omega$ (i.e., the grid were degenerate in this case), then $\gamma_n$ could have discontinuites that approximate the discontinuites of the exact unknown $\gamma$. This idea has been exploited in [7, 8, 9] in one and two dimensions.

Of course, we are not restricted to Tikhonov regularization, any other convergent regularization method will be appropriate, in fact, for the numerical realization we prefer iterative regularization methods. We implemented three of them: The minimal error method, the steepest descent method and an inexact iteratively regularized Gauss-Newton method.

To describe the ideas we introduce some notation. In step 2 we have to solve (2.3), where $\phi_n$ is kept fixed and $c$ is the unknown. We use $F_{\phi_n}(c) := F(c(\phi_n^{-1}))$, and $F'_{\phi_n}(c)$ for the Fréchet derivative with respect to $c$, and $F'_{\phi_n}(c)^*$ for its adjoint (in the space $H^1_0$).

For exact data the iterate $c_n$ in step 2 is computed by

$$c_n = \lim_{k \to \infty} c_{n,k}\,,$$

where $c_{n,k}$ is obtained out of one of the following iteration methods:

The minimal error and the steepest descent method use the iteration

$$c_{n,k+1} = c_{n,k} + \alpha_k s_k \qquad s_k = -F'_{\phi_n}(c_{n,k})^*(F'_{\phi_n}(c_{n,k}) - u_\delta) \tag{2.8}$$

with

$$\alpha_k = \begin{cases} \dfrac{\|F_{\phi_n}(c_{n,k}) - u_\delta\|^2_{L^2}}{\|s_k\|^2_{H^1_0}}, & \text{for the minimal error method}\,, \\ \dfrac{\|s_k\|^2_{H^1_0}}{\|F'_{\phi_n}(c_{n,k})s_k\|^2_{L^2}}, & \text{for the steepest descent method}\,. \end{cases} \tag{2.9}$$

Both iteration methods start with an appropriate initial value $c_{n,0}$, which we set $c_{n,0} = \gamma_{n-1}(\phi_n)$, $c_{0,0} = 0$. These iterations can be seen as Landweber Iteration with an iteration dependent steplength $\alpha_k$. Convergence and convergence rates for these two methods have been investigated in [12]. If the data are exact, i.e. $\delta = 0$, then $c_{n,k}$ converges under suitable conditions on $F_{\phi_n}$ to a minimal norm solution of (2.3). In the noisy case the iteration is stopped according to Morozov's discrepancy principle, i.e., at the first iterate $c_{n,k}$ satisfying

$$\|F_{\phi_n}(c_{n,k}) - u_\delta\|_{L^2} \leq \tau\delta$$

with a suitable parameter $\tau > 1$.

The third iteration method we use is a variant of the iteratively regularized Gauss-Newton algorithm. For the exact algorithm a new update for $c_{n,k+1}$ is defined by the equation

$$\begin{aligned}\left(F'_{\phi_n}(c_{n,k})^* F'_{\phi_n}(c_{n,k}) + \alpha_k I\right)(c_{n,k+1} - c_{n,k}) = \\ -F'_{\phi_n}(c_{n,k})^*(F_{\phi_n}(c_{n,k}) - u_\delta) + \alpha_k c_{n,k}\,.\end{aligned} \tag{2.10}$$

The sequence $\alpha_k$ plays the role of a regularization parameter and can be chosen as geometrically decaying sequence: $\alpha_k = \alpha_0 r^k$ with $0 < r < 1$. Again Morozov's discrepancy principle is used to stop the iteration in the presence of data noise. For the iteratively regularized Gauss-Newton iteration convergence and convergence rates have been proven in [1].

In our computations we prefer an inexact Gauss-Newton method: instead of solving system (2.10) exactly, we use a Conjugate Gradient method to approximate the exact solution. That means that $c_{n,k+1} = c_{n,k} + \nu_l$, where $\nu_l$ denotes the $l^{th}$ step of a Conjugate Gradient method applied to the equation

$$\left(F'_{\phi_n}(c_{n,k})^* F'_{\phi_n}(c_{n,k}) + \alpha_k I\right)\nu = -F'_{\phi_n}(c_{n,k})^*(F_{\phi_n}(c_{n,k}) - u_\delta) + \alpha_k c_{n,k}. \tag{2.11}$$

In the CG method the operator on the left-hand side has to be applied to functions $\nu$. This means that we only have to calculate directional derivatives and hence it is not necessary in the discretized version to compute and store the matrix corresponding to the operator $F'_{\phi_n}(c_{n,k})^* F'_{\phi_n}(c_{n,k})$.

Since for the evaluation of $F, F'h, F'^*z$ we always have to solve a PDE, we intend to use a limited number of CG-iterations to save computation time. Thus we propose to keep the number of CG-iterations fixed ($l = 10$ suffices). Alternatively, we stop the iteration if equation (2.11) is solved with a precision to 10% of the initial error. These stopping criteria keep the computational effort moderate, however, since the equation is not solved exactly the question of convergence of the algorithm arises. Practically, the method shows convergence, and we expect that it could also be verified theoretically from the following reasons: note that the first step in a CG-iteration for (2.11) is identical to a steepest descent step for the Tikhonov functional. On the other hand, an exact solution (i.e., $\lim_{l\to\infty} \nu_l$) of (2.11) is identical to one Gauss-Newton step for this functional. Both iterations – steepest descent and iteratively regularized Gauss-Newton – yield convergence under reasonable conditions, and our inexact iteration is somewhere in between. Thus, we expect convergence also for the iteration with a fixed number of CG-steps $l$.

## 2.3. Deformation method

We now turn to step 4 in our algorithm above: the transformation $T$ is defined via the deformation method. It provides direct control over the cell size. We briefly describe the main ideas following [16]:

Given a positive monitoring function $m(\zeta,t)>0$ depending on the space variable $\zeta$ and time $t$, we want to construct a deformation function $\phi(\xi,t)$ such that

$$\begin{aligned} \det D\phi(\xi,t) &= m(\phi(\xi,t),t)\,, \quad \xi\in\Omega, t>0\,, \\ \phi(\xi,0) &= \phi_{\text{init}}(\xi) \qquad\quad \xi\in\Omega \end{aligned} \tag{2.12}$$

In numerical computations for PDEs $m$ usually describes the smoothness of the solutions. If $m$ is small – indicating lack of smoothness – then the volume of a grid element will be small too. A necessary solvability condition for $m$ is the normalization property:

$$\int_\Omega \left(\frac{1}{m(\zeta,t)}-1\right)d\zeta = 0 \tag{2.13}$$

Although (2.12) is a highly nonlinear PDE there is an elegant algorithm to solve it (cf. [2]).

First, we define a velocity field $v(\zeta,t)$ by

$$\begin{aligned} \operatorname{div} v(\zeta,t) &= -\frac{\partial}{\partial t}\frac{1}{m(\zeta,t)} \qquad \zeta\in\Omega, t\geq 0 \\ \langle\, v(\zeta,t), n(\zeta,t)\,\rangle &= 0 \quad \zeta\in\partial\Omega, t>0, \end{aligned}$$

here $n(\zeta,t)$ denotes the unit outward normal to $\partial\Omega$. $v$ may be calculated by the gradient $v=\nabla w$ of the solution of the Neumann problem (for fixed $t\geq 0$)

$$\begin{aligned} \Delta w(\zeta,t) &= -\frac{\partial}{\partial t}\frac{1}{m(\zeta,t)}\,, \quad \zeta\in\Omega\,, \\ \frac{\partial}{\partial n}w &= 0\,, \qquad\qquad\quad \text{on } \partial\Omega\,. \end{aligned} \tag{2.14}$$

Note that the solvability condition of the Neumann problem is satisfied by the normalization property (2.13). A function $\phi(\xi,t)$ satisfying (2.12) is obtained as solution of the system of ordinary differential equations (for fixed $\xi\in\Omega$)

$$\begin{aligned} \frac{d}{dt}\phi(\xi,t) &= v(\phi(\xi,t),t)m(\phi(\xi,t),t)\,, \quad t>0\,, \\ \phi(\xi,0) &= \phi_{\text{init}}(\xi)\,. \end{aligned} \tag{2.15}$$

In our case, $t$ plays the role of a homotopy parameter connecting the initial grid at $t=0$ with the final grid satisfying (2.12).

In each iteration in step 4 of our algorithm we compute the deformation function $\phi_{n+1}$ by

$$\det D\phi_{n+1}(\xi) = m_n(\phi_{n+1}(\xi)) \tag{2.16}$$

In our numerical realization we choose

$$m_n(\zeta) := \frac{C_n}{(1+\beta|\nabla\gamma_n(\zeta)|^2)^{\frac{1}{2}}}, \quad \gamma_n(\zeta) = c_n(\phi_n^{-1}(\zeta)), \tag{2.17}$$

$\beta > 0$ being a fixed parameter, and $C_n$ such that the normalization property (2.13) holds.

Since we are using the monitoring function (2.17) with this would require to invert $\phi_n$. However, in [11] a variant of the above algorithm is described to circumvent this inversion. The idea is to include the transformation function $\phi_n$ from the previous step by defining

$$\phi_{n+1}(\xi) := \phi_n(\sigma(\xi, 1))$$

with an unknown function $\sigma(\xi, t), t \in [0, 1]$, such that (2.16) holds:

$$\det D\phi_{n+1}(\xi) = \det D\phi_n(\sigma(\xi, 1)) \det D\sigma(\xi, 1) = m_n(\phi_n(\sigma(\xi, 1))).$$

This yields an equation for $\sigma$:

$$\det D\sigma(\xi, 1) = \tilde{m}_n(\sigma(\xi, 1)) \tag{2.18}$$

with

$$\tilde{m}_n(\xi) = \frac{m_n(\phi_n(\xi))}{\det D\phi_n(\xi)}.$$

If we denote components of the deformation function in the $n$-th step by $a, b$, i.e., $\phi_n(\xi) = (a(\xi), b(\xi))$ and $\gamma_n(\zeta) = c(\phi_n^{-1}(\zeta))$, the chain rule yields ($\xi = (\xi_1, \xi_2)$)

$$\begin{aligned} \tilde{m}_n(\xi) &= \frac{C_n}{\det D(a(\xi), b(\xi))(1 + \beta|\nabla\gamma_n(\phi_n(\xi))|^2)^{\frac{1}{2}})} \\ &= \frac{C_n}{(a_{\xi_1}b_{\xi_2} - a_{\xi_2}b_{\xi_1})^2 + \beta((b_{\xi_2}c_{\xi_1} - b_{\xi_1}c_{\xi_2})^2 + (a_{\xi_1}c_{\xi_2} - a_{\xi_2}c_{\xi_1})^2))^{\frac{1}{2}}}. \end{aligned}$$

We start with $\sigma(\xi, 0) = \mathrm{id}(\xi) = \xi$ and use the parameter $t \in [0, 1]$ to connect $\sigma(\xi, 0)$ with $\sigma(\xi, 1)$. The function $\sigma(\xi, t)$ is chosen to solve

$$\det \sigma(\xi, t) = \frac{1}{(1 - t) + t\frac{1}{\tilde{m}(\sigma(\xi, t))}}, \quad t \in [0, 1]\,. \tag{2.19}$$

This equation has the form (2.12) and can be solved as above ((2.14), (2.15)). Note that by the choice how the right-hand side in (2.19) depends on $t$, $w(x, t)$ in (2.14) will not depend on $t$, and has to be solved only once in step 4.

## 3. Numerical realization

### 3.1. Approximation of the direct problem

For the numerical computations we restrict ourselves to the unit square in $\mathbb{R}^2$, $\Omega = [0, 1]^2$. Our algorithm requires to solve equation (1.1) on $\Omega$ with $\tilde{\gamma} = 1 + \gamma$. In the weak formulation we have to find $u \in H_0^1(\Omega)$ such that

$$\langle\, (1 + \gamma)\nabla u, \nabla\psi \,\rangle = \langle\, f, \psi \,\rangle \qquad \forall\psi \in H_0^1(\Omega)\,, \tag{3.1}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product on $L^2(\Omega)$. Since $\gamma = c(\phi^{-1})$ and since we want to avoid inverting $\phi$, we transform the differential equation. In fact, if $u = F_\phi(c)$ solves (3.1), then $\tilde{u} := u \circ \phi$ solves

$$\langle (1+c) M \nabla \tilde{u}, \nabla \psi \rangle = \langle (f \circ \phi) \det D\phi, \psi \rangle \qquad \forall \psi \in H_0^1(\Omega)\,, \tag{3.2}$$

with the matrix-valued function

$$M(\xi) = \det D\phi(\xi) [D\phi(\xi)]^{-1} [D\phi(\xi)]^{-T}\,.$$

For a numerical computation we use Courant (i.e. piecewise linear and continuous) elements for $c$ and for the transformation function $\phi$ on the same uniform grid. The nodal points of the grid are given by

$$\xi_{i,j} = (\tfrac{i}{N}, \tfrac{j}{N})\,, \quad i,j = 0, \ldots, N\,. \tag{3.3}$$

This yields $N^2$ small squares, which are again subdivided into two triangles by the diagonal running from the lower left to the upper right corner. The $2N^2$ triangle elements are thus the triangles with corners

$$\text{and} \qquad \xi_{i,j}, \xi_{i,j+1}, \xi_{i+1,j+1}\,, \qquad i,j = 0, \ldots, N-1\,.$$

As usual the linear finite element basis functions $\psi_{i,j}(\xi)$ are first order polynomials on each triangle and satisfy $\psi_{i,j}(\xi_{k,l}) = \delta_{i,k}\delta_{j,l}$. $\psi_{i,j}$ is supported on the at most 6 triangles which have $\xi_{i,j}$ as corner.

Given $c, \phi$, then a numerical approximation $\tilde{u}_N$ of $\tilde{u}$ solving (3.2) is given as solution of

$$\begin{aligned} \langle (1+c) M \nabla \tilde{u}_N, \nabla \psi_{i,j} \rangle &= \langle (f \circ \phi) \det D\phi, \psi_{i,j} \rangle \quad \forall i,j = 1, \ldots, N-1\,, \\ \tilde{u}_N &= \sum_{i,j=1}^{N-1} \tilde{u}_{i,j} \psi_{i,j}\,. \end{aligned} \tag{3.4}$$

This system is equivalent to a numerical approximation of the original weak form (3.1) on a transformed grid, i.e., the operator $F_\phi(c)$ is approximated by

$$F_\phi^N(c) := u_N = \sum_{i,j=1}^{N-1} \tilde{u}_{i,j} (\psi_{i,j} \circ \phi^{-1})\,. \tag{3.5}$$

Since we only control the cell size of the transformed grid and not the shape, the discretized equation (3.4) can be badly conditioned if the shape of the triangles on the transformed grid become nearly degenerate. This can be avoided by subdividing these degenerate triangles appropriately.

By our choice of $\phi$ and $c$, the Jacobian matrix $D\phi$ and $\nabla \psi_{i,j}$ are constant on each triangle, and $c$ is a first order polynomial on each element, thus the integrals on the left-hand side of (3.4) can be evaluated exactly. For the right-hand side we calculate the values $f \circ \phi$ at the nodal points and then interpolate this function linearly on each triangle to evaluate the integral. By this procedure we obtain a $(N-1)^2 \times (N-1)^2$ linear system

$$A(c,\phi) \vec{u} = \vec{f} \tag{3.6}$$

where $\vec{u}$ is a vector with entries $\tilde{u}_{i,j}$, $i,j = 1, \ldots, N-1$, $A(c,\phi)$ is a sparse, band limited matrix with 6 nonzero off-diagonals, and $\vec{f}$ is a load vector with entries $\langle (f \circ \phi) \det D\phi, \psi_{i,j} \rangle$, $i,j = 1, \ldots, N-1$. Equation (3.6) is solved using MATLAB's PCG-method.

### 3.2. Realization of the regularization method

For the computation of the regularized solutions (cf. (2.8), (2.9), (2.11)) we have to replace the operator $F_{\phi_n}(c_{n,k})$ by $F_{\phi_n}^N(c_{n,k})$ (see (3.5)).

The effort for computing the derivative and its adjoint into one direction is of the same order as one evaluation of $F_{\phi_n}^N(c_{n,k})$. In fact, for a calculation of $u_N = F_{\phi_n}^N(c_{n,k})$ we have to solve the equation (see (3.5), (3.6))

$$A(c_{n,k}, \phi_n)\vec{u} = \vec{f}$$

once, and for the derivative $w_N := (F_{\phi_n}^N)'(c_{n,k})h = \sum_{i,j=1}^{N-1} \tilde{w}_{i,j}(\psi_{i,j} \circ \phi^{-1})$ we additionally have to solve

$$A(c_{n,k}, \phi_n)\vec{w} = -A(h-1, \phi_n)\vec{u}\,.$$

Note that we are only interested in elements $h$ in the discretized space

$$X_N := \{\sum_{i,j=1}^{N-1} h_{i,j}\psi_{i,j} \,:\, h_{i,j} \in \mathbb{R}\} \subset H_0^1\,.$$

The adjoint of the operator $(F_{\phi_n}^N)'(c_{n,k}) : X_N \to L^2$ applied to some element $z = \sum_{i,j=1}^{N-1} z_{i,j}(\psi_{i,j} \circ \phi^{-1})$ is given by the solution of

$$\langle\, \nabla((F_{\phi_n}^N)'(c_{n,k})^* z), \nabla\psi_{i,j} \,\rangle = -\langle\, \nabla\tilde{u}^T M \nabla\tilde{\eta}, \psi_{i,j} \,\rangle\,, \qquad \forall i,j = 1, \ldots, N-1\,,$$

in $X_N$, where $\eta = \sum_{i,j=1}^{N-1} \eta_{i,j}\psi_{i,j}$ solves

$$A(c_{n,k}, \phi)\vec{\eta} = G\vec{z}$$

with the Gramian matrix

$$G_{(i,j);(l,k)} = \langle\, \psi_{i,j} \det D\phi, \psi_{l,k} \,\rangle\,. \tag{3.7}$$

Note that $c_{n,k}$ has to satisfy $\gamma_1 \le 1 + c_{n,k} \le \gamma_2$. Therefore, we project $c_{n,k}$ onto this convex set, in each iteration step if necessary.

In our regularization methods we have two iteration loops: the index $n$ corresponds to the update of the grid and the iteration indexed by $k$ corresponds to the regularization step. An obvious improvement of the algorithm can be expected by including the information of the regularization iteration into the regridding step and combining these two iterations into one:

So instead of finishing the iteration with respect to $k$ until the stopping rule is satisfied, we perform at most $k_0$ steps, where $k_0$ is a small fixed number. This means that even if the stopping rule is not yet satisfied after $k_0$ steps, we perform a regridding step and set $c_{n+1,0} = c_{n,k_0}$. The numerical results indicate that this mixed iteration converges.

#### 3.2.1. Realization of the deformation method

For the computation of the grid update $\phi_{n+1}(\xi) = \phi_n(\sigma(\xi, 1))$ we have to solve a Neumann problem for the Poisson equation (2.14) once and the system of ordinary differential equation (2.15). To do this we first of all compute the monitoring function $\tilde{m}_n$, which is piecewise constant on each triangle element. Note that the Poisson equation is defined on a uniform grid, it is again solved by MATLAB's PCG-algorithm. $\tilde{m}_n$ is scaled by taking $C_n$ as $\int_\Omega \tilde{m}_n^{-1}(\xi)d\xi$ such that the solvability condition (2.13) holds. This integral can be evaluated exactly, because $\tilde{m}_n$ is piecewise constant.

The system of ordinary differential equations is solved by the classical fourth-order Runge-Kutta method. Since the right-hand side of (2.15) is piecwise constant on the triangles we use bilinear interpolation to obtain a continuous function.

Note that $\phi_n$ has to map the boundary of $\Omega$ onto itself. This is achieved by keeping $\phi_n$ fixed at the corner points of the unit square. Moreover, the first component of $\phi$ is not changed at the lines $y = 0$ and $y = 1$, and vice versa the second component at $x = 0$ and $x = 1$.

Since, due to discretization, it may happen that the function $\sigma$ has a negative determinant, we also include a smoothing step then by setting the values of $\sigma$ at the corner points $\xi_{i,j}$ to the mean value of neighboring nodal points. However, it turned out in our computations that such a smoothing step is rarely necessary as long as $\beta$ is not too large. For large values of $\beta$ a smaller stepsize in the Runge-Kutta method was sufficient to guarantee the positivity of the determinant.

If the grid is updated, i.e., a new transformation function $\phi_{n+1}$ is computed we also have to recalculate several vectors and matrices depending on the grid. In fact, we have to compute the Gramian matrix (3.7), and the load vector $\vec{f}$ (3.6). The stiffness matrix $A(c_{n,k}, \phi_n)$ has to be updated in every step, too.

## 4. Numerical results

For the numerical experiments we used $f(x, y) = \sin(2\pi x)\cos(\pi y)$ as right-hand side in (1.1).

The following examples were considered:

**Example 4.1.** Circle: $\gamma = 1 + 2\chi_{B_{0.55,0.45}(0.3)}$

**Example 4.2.** Ramp: $\gamma(x, y) = 1 + 2\frac{x-0.25}{0.35}\chi_{\{(x,y)|0.25\leq x\leq 0.6, 0.2\leq y\leq 0.8\}}$

**Example 4.3.** Moon: $\gamma = 1 + 2(\chi_{B_{0.55,0.5}(0.3)}(1 - \chi_{B_{0.4,0.5}(0.25)}))$

**Example 4.4.** Circle and rectangle:
$\gamma = 1 + 2(\chi_{B_{0.35,0.65}(0.15)} + \chi_{\{(x,y)|0.1\leq x-y\leq 0.6, 0.7\leq x+y\leq 1.1\}}$

($B_{x_0,y_0}(r)$ denotes the circle with midpoint at $(x_0, y_0)$ and radius $r$).

For all examples the data points were first computed using a fine uniform grid with $N = 120$ and then contaminated by random noise. This grid is much finer than the one used to calculate the regularized solutions (usually $N = 40$). By this we avoid so-called inverse crimes, namely to use the same setup for the calculation of the simulated data and the regularization itself.

Since all matrices in our above algorithm are sparse we need a storage effort of order $\mathcal{O}(N^2)$. This shows the advantage of using a CG-method for the iteratively regularized Gauss-Newton iteration, since the full matrix in (2.11) has $\mathcal{O}(N^4)$ entries.

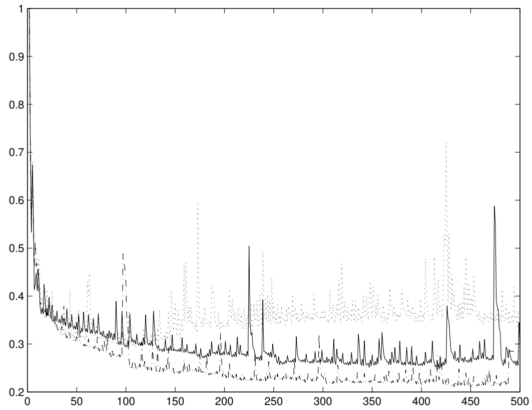


Figure 4.1: Error reduction vs. iteration: minimal error method

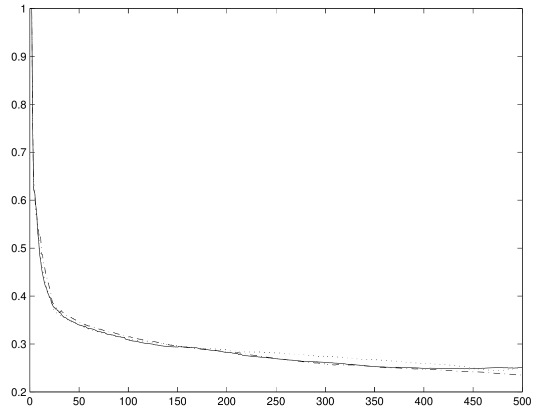


Figure 4.2: error reduction vs. iteration: steepest descent method

Viewing computational time, the most expensive step is to solve equation (3.6). Basically, the PCG-algorithm needs $\mathrm{cond}^{\frac{1}{2}} \log \epsilon^{-1}$ iterations to obtain an error reduction of $\epsilon$. Here cond is the condition number of the preconditionend stiffnessmatrix $A(c, \phi)$. Each iteration needs a complexity of $\mathcal{O}(N^2)$ flops. Note that it is not necessary to solve the equation to a precision which is below the data noise.

We emphasize that the regridding step is not very expensive at all. In fact it requires to solve one PDE and the Runge-Kutta step. After all, its complexity is about of the same order as one function evaluation of $F^N_{\phi_n}(c_{n,k})$.

We first report about the differences of the iteration methods we used. Figures 4.1, 4.2 show the relative $L^2$-error $\frac{\|\gamma_{n,k}-\gamma^\dagger\|}{\|\gamma_0-\gamma^\dagger\|}$ versus the iteration number for the minimal error method and the steepest descent method for Example 4.1 with discretization $N = 40$ and unperturbed data. Here every step of the form (2.8) is counted as one iteration. Hence if the grid is updated after $k_0$ steps of (2.8) the total iteration number is $n_{tot} = (n-1)k_0 + k$ in the previous notation.

We chose different intervalls for the grid update step. The dotted line indicates a grid update in every second step, i.e., $c_{n+1,0} = c_{n,2}$. The full lines correspond to a grid update in every 5-th step and the dashed lines to one in every 10-th step.

The two iteration methods show a different behavior: the error reduction for the minimal error method is not as smooth as for the steepest descent method. Moreover, the former exhibits a stronger dependence on the choice of the grid update intervalls $k_0$, whereas the latter is quite insensitive to it. In fact, for the minimal error method we found the best results for $k_0 = 10$. However, the minimal error method performs better with respect to the required CPU-time. For an error reduction of 70% the steepest descent method needed more than three times the CPU-time for the minimal error method.

The inexact iteratively regularized Gauss Newton iteration yields the best results. It is quite insensitive to the choice of $k_0$ as the steepest descent method and performs better with respect to the CPU-time than the minimal error method.

Figures 4.3 - 4.6 show the results for our examples with exact and noisy data (5% noise). We used $N = 40$, $\beta = 10$ and the inexact iteratively regularized Gauss-Newton method (2.11) with $\alpha_{k+1} = 0.9\alpha_k$. A grid update was done in every second step (i.e., $k_0 = 2$). For exact data the iteration was stopped at $\alpha_k = 10^{-10}$ and for noisy data we used the discrepancy principle as stopping rule.

The results show that we can identify the location of discontinuities quite well. Obviously, for noisy data the resolution is not so sharp than for exact ones. Note that even for exact data we have noise due to discretization. The second example exhibits that our algorithm does not suffer from the staircasing effect of several BV-regularizations. We observed that it is difficult to identify $\gamma$ in regions where the gradient of $u$ vanishes or is small. This effect can be expected, since $\gamma$ is not identifable at points where $\nabla u = 0$.

Moreover, for exact data, the results for $\gamma$ show a rather low dependence on the choice of the parameter $\beta$ and the number of regridding steps. There seems to be a broad region of values of $\beta$ yielding similar results. For noisy data we obtained slightly better results with respect to the resolution of discontinuites with larger values of $\beta$ and more regridding steps.

The choice of the method for solving the ordinary differential equation (2.15) is quite important. We additionally tried to use an explicit Euler method instead of the Runge-Kutta with unsatisfactory results. Using a method with lower order often yields negative determinants and requires smoothing steps of the grid, which makes the resolution of discontinuities not so sharp.

Finally, we want to mention that our regularization algorithm is less dependent on the choice of the size of the initial grid, since we are using a variable grid that is adapted to the solution. A comparison of our results with numerical computations where the gridsize was chosen $N = 100$ instead of $N = 40$ showed practically no improvement for Examples 4.1 – 4.3. It is obvious that the possibility to choose a coarser grid saves a lot of computation time. In Example 4.4 two disjoint regions of discontinuity have to be identified. For a good resolution more grid lines are needed for the gap between the two regions. Hence, the result was slightly better for the finer grid.

# References


[1] B. Blaschke, A. Neubauer, and O. Scherzer, *On convergence rates for the iteratively regularized Gauss-Newton method*, IMA Journal of Numer. Anal. 17 (1997), 421–436.

[2] P. Bochev, G. Liao, and G. dela Pena, *Analysis and computation for adaptive moving grids by deformation*, Numer. Methods of PDEs 12 (1996), 489–506.

[3] Z. Chen and J. Zou, *An augmented Lagrangian method for identifying discontinous parameters in elliptic systems*, SIAM J. Control. Optim 37 (1999), 892–910.

[4] H. W. Engl, M. Hanke, and A. Neubauer, *Regularization of Inverse Problems*, Kluwer, Dordrecht, 1996.

[5] S. Kindermann, *Regularization of Ill-Posed Problems with Discontinuous Solutions by Curve and Surface Representations*, PhD thesis, University of Linz, October 2001.

[6] S. Kindermann and A. Neubauer, *Each $\mathcal{BV}$-function is representable by an $\mathcal{H}^1$-curve*, Technical Report 1/1999, Industrial Mathematics Institute, University of Linz, 1999, submitted.

[7] ——, *Identification of discontinuous parameters by regularization for curve representations*, Inverse Problems 15 (1999), 1559–1572.

[8] ——, *Estimation of discontinuous parameters of elliptic partial differential equations by regularization for surface representations*, Inverse Problems 17 (2001), 789–803.

[9] ——, *Regularization for surface representations of discontinuous solutions of linear ill-posed problems*, Numer. Funct. Anal. Optim. 22 (2001), 79–105.

[10] R. Luce and S. Perez, *Parameter identification for an elliptic partial differential equation with distributed noisy data*, Inverse Problems 15 (1999), 291–307.

[11] A. Neubauer, *Estimation of discontinuous solutions of ill-posed problems by regularization for surface representations: numerical realization via moving grids*, SFB-Report 01-28, University of Linz, 2001, submitted.

[12] A. Neubauer and O. Scherzer, *A convergent rate result for a steepest descent method and a minimal error method for the solution of nonlinear ill-posed problems*, ZAA 14 (1995), 369–377.

[13] ——, *Reconstruction of discontinuous solutions from blurred data*, in: R. L. Barbour, M. J. Carvlin, and M. A. Fiddy, eds., Computational, Experimental, and Numerical Methods for Solving Ill-Posed Inverse Imaging Problems: Medical and Nonmedical Applications, Vol. 3171 of Proceedings of SPIE, SPIE, Washington, 1997, 34–41.

[14] ——, *Regularization for curve representations: uniform convergence for discontinuous solutions of ill-posed problems*, SIAM J. Appl. Math. 58 (1998), 1891–1900.

[15] W. Ring, *Structural properties of solutions of total variation regularization problems*, M2AN, Math. Model. Numer. Anal. 34 (2000), 799–810.

[16] B. Semper and G. Liao, *A moving grid finite element method using grid deformation*, Numer. Methods of PDEs 11 (1995), 603–615.

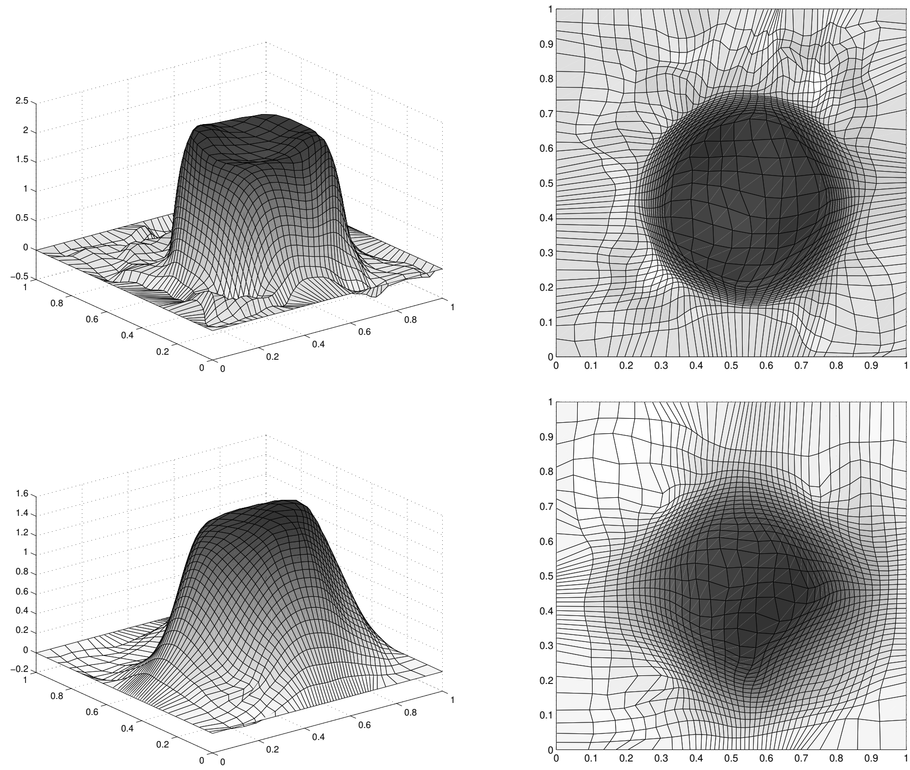


Figure 4.3: Reconstruction for Example 4.1 for exact data and below for 5% noise

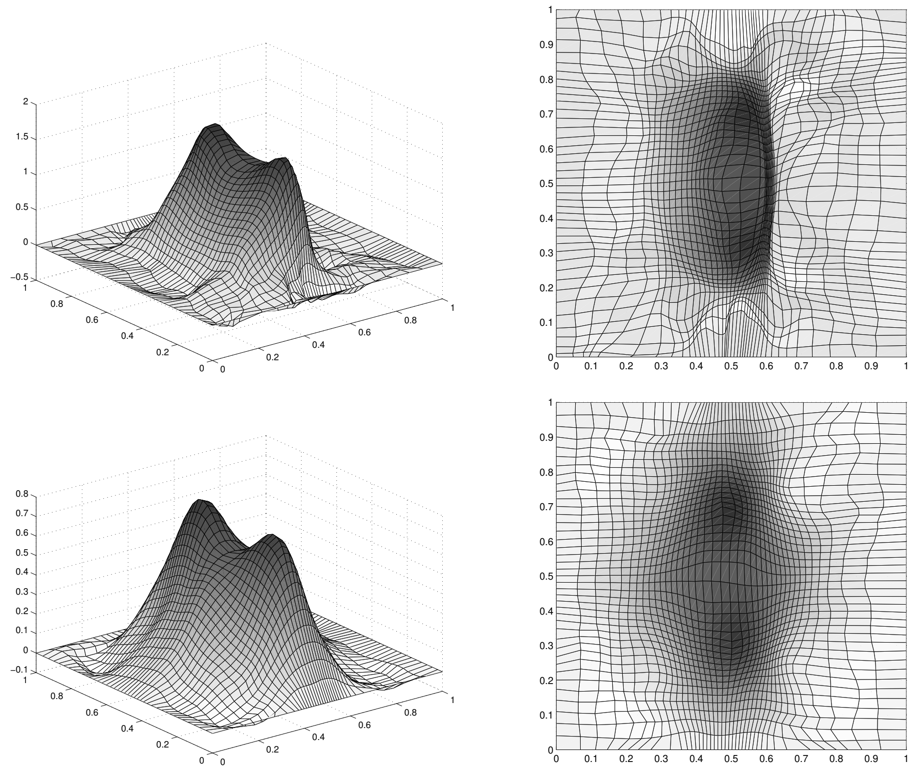


Figure 4.4: Reconstruction for Example 4.2 for exact data and below for 5% noise

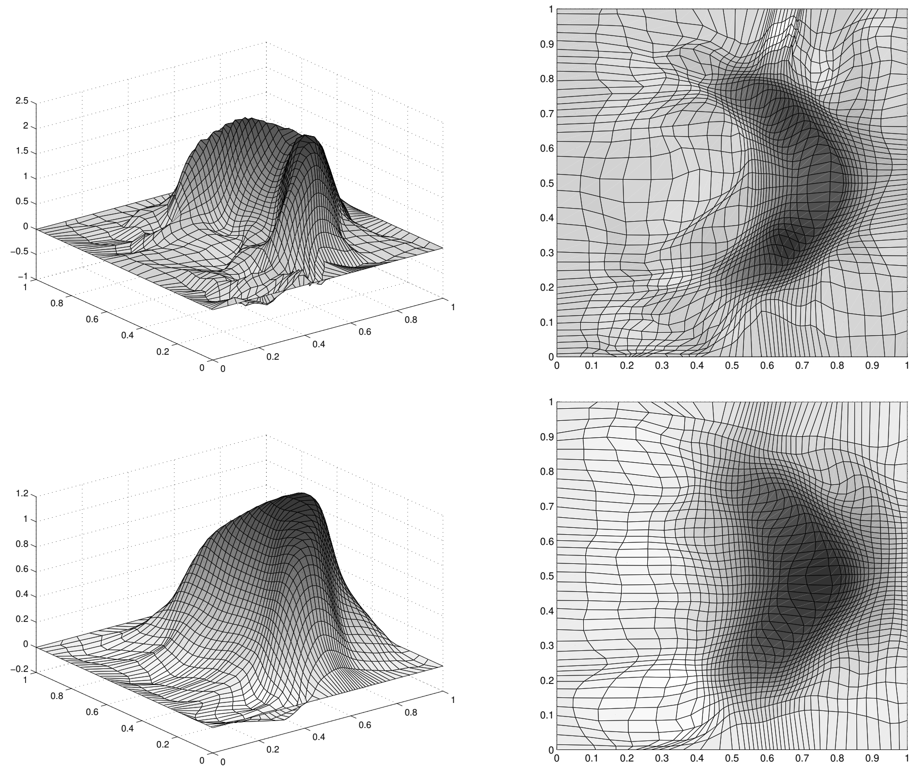

Figure 4.5: Reconstruction for Example 4.3 for exact data and below for 5% noise

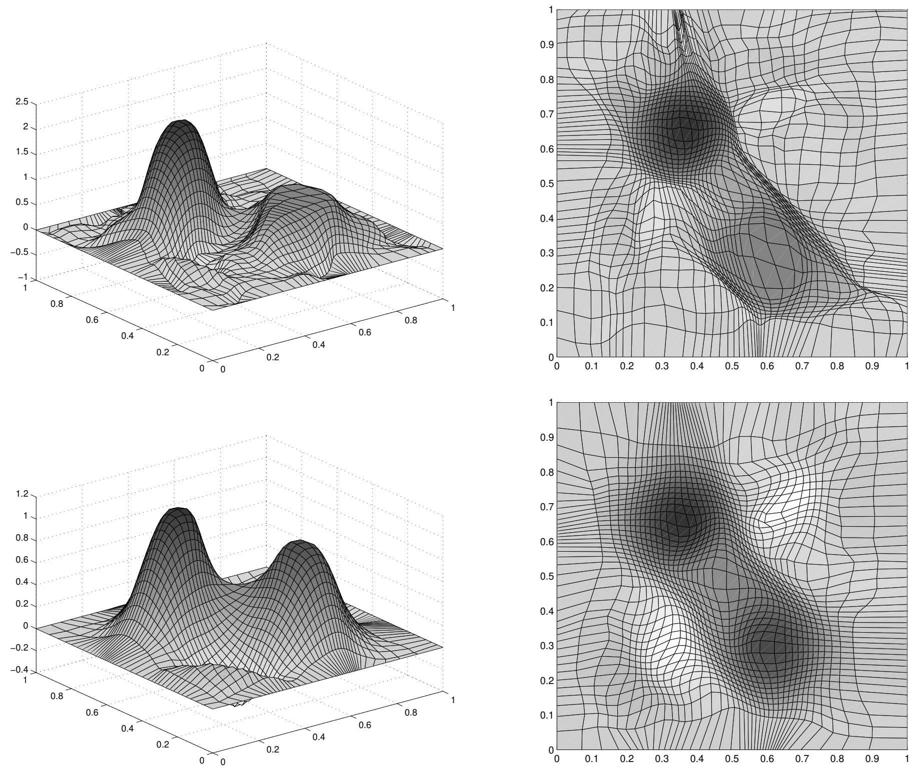

Figure 4.6: Reconstruction for Example 4.4 for exact data and below for 5% noise