Estimation of Discontinuous Solutions of Ill-Posed Problems by Regularization for Surface Representations: Numerical Realization via Moving Grids*

ANDREAS NEUBAUER

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

Abstract. In this paper we discuss the numerical realization of a new regularization method, *regularization for surface representations*, which is well-suited for ill-posed problems with discontinuous solutions: this realization is essentially based on *moving grids*. After describing the method we present several numerical examples showing that this combination with moving grids is a powerful tool to identify discontinuities in two-dimensional problems.

Key Words: Nonlinear Tikhonov regularization, ill-posed problems, discontinuous solutions, regularization for surface representations, moving grids, deformation method.

AMS Subject Classifications: 65J15, 65J20, 65R30

1. Introduction

In this paper we study the estimation of discontinuous solutions of linear or nonlinear ill-posed problems

$$F(f) = g \tag{1.1}$$

from noisy measurements g^δ of g satisfying $\|g^\delta - g\| \le \delta$, where

$$F : \mathcal{D}(F)(\subset X) \to Y$$

and X and Y are Hilbert spaces.

Tikhonov regularization is well known to stabilize ill-posed problems (cf., e.g., [3]). In this method an exact solution of (1.1) is approximated by a minimizer f_α^δ of the functional

$$\|F(f) - g^\delta\|^2 + \alpha p(f - f^*)\,, \tag{1.2}$$

where f^* is an initial guess of the exact solution and $p(\cdot)$ is a properly chosen penalty term. Usually one uses the penalty term

$$p(f - f^*) = \|f - f^*\|^2\,. \tag{1.3}$$

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

However, this is not appropriate for ill-posed problems with discontinuous solutions, since it has a smoothing effect on the regularized solutions.

Using the bounded variation norm in (1.2) as penalty term has turned out to be an effective regularization method (cf., e.g., [1, 2, 10]). A major drawback of this approach, however, is that this norm is not differentiable.

In [9], a new approach for regularizing problems with discontinous solutions was introduced, *regularization for curve representations*. The essence of this method is to replace a discontinuous function by its continuous graph. This allows a combination with the usual Tikhonov regularization in Hilbert spaces ((1.2), (1.3)) and, therefore, all the results about convergence from the general theory on nonlinear Tikhonov regularization are applicable. The method was successfully applied to one-dimensional parameter estimation problems in [5].

In [7], the method was generalized to two-dimensional problems, *regularization for surface representations*, and applied to the linear problem of deblurring images. The idea of this method is as follows:

Let f be a C^1-function, then its graph $G_f := \{(x, y, f(x, y)) \in \mathbb{R}^3\}$ defines a surface in $\mathbb{R}^3$. Let $(a(u, v), b(u, v), c(u, v))$ be an equivalent parameterization of G_f, then f can be recovered by $f(x, y) = c((a, b)^{-1}(x, y))$. It was shown in [7] that discontinuous functions f that are in a certain subset of the functions of bounded variation may be parameterized by smooth parameterizations. To guarantee that the parameterized surface may be interpreted as the graph of a function, we restricted ourselves to parameterizations

$$a(u, v) = a(u) \qquad\qquad b(u, v) = b(v) \tag{1.4}$$

satisfying $(a, b, c) \in \mathcal{D}$ with

$$\begin{aligned} \mathcal{D} &:= \{(a, b, c) \in X \,:\, a(1) = 1 = b(1),\ \dot{a} \geq 0 \text{ a.e.},\ \dot{b} \geq 0 \text{ a.e.}\}\,, \\ X &:= \{(a, b, c) \in H^1[0, 1] \times H^1[0, 1] \times H_0^1(\Omega) \,:\, a(0) = 0 = b(0)\}\,. \end{aligned} \tag{1.5}$$

This method, which was succesfully applied to two-dimensional parameter estimation problems in [6], is still quite restrictive: due to the special choice of surface parameterizations, discontinuities in solutions $f = c(a^{-1}(x), b^{-1}(y))$ are only allowed to occur on lines parallel either to the x- or the y-axis.

The general case of parameterizations was treated from a theoretical point of view in the PhD-Thesis of Stefan Kindermann [4]. He showed that a general parameterization, $f((a(u, v), b(u, v)) = c(u, v)$, exists for all functions of bounded variation with compact support. He also gave conditions on F that guarantee that the Tikhonov regularized solutions converge to the exact one.

Let us consider the two-dimensional linear integral equation

$$g(s, t) = Ff(s, t) := \int_\Omega k(s, t, x, y) f(x, y)\, d(x, y)\,, \quad (s, t) \in \Omega\,, \tag{1.6}$$

where $\Omega = [0, 1]^2 \subset \mathbb{R}^2$ is the unit square and $k \in L^2(\Omega^2)$. Then $F : L^2(\Omega) \to L^2(\Omega)$ is compact. The reformulation of this equation in terms of (a, b, c) yields

$$g(s, t) = G(a, b, c)(s, t) := \int_\Omega k(s, t, a(u, v), b(u, v)) c(u, v) |J(a, b)(u, v)|\, d(u, v) \tag{1.7}$$

with

$$J(a, b)(u, v) = a_u(u, v) b_v(u, v) - a_v(u, v) b_u(u, v)\,.$$

This is now a nonlinear integral equation with respect to a and b. It follows from [4] that G is at least well defined for those $a, b, c \in H^1(\Omega)$ that are parametrizations of functions of bounded variation with compact support.

This problem is stabilized via nonlinear Tikhonov regularization, where we can use the standard penalty term, i.e., we are looking for minimizers $(a_\alpha^\delta, b_\alpha^\delta, c_\alpha^\delta)$ of the functional

$$\|G(a,b,c) - g^\delta\|_{L^2}^2 + \alpha(\|\nabla a\|_{L^2}^2 + \|\nabla b\|_{L^2}^2 + \|\nabla c\|_{L^2}^2)\,. \tag{1.8}$$

For $(a, b) : \Omega \to \Omega$ to be admissible in the sense that the parameterized surface may be interpreted as the graph of a function, it is necessary that (a, b) is one to one and onto. Besides some boundary conditions this means that $J(a, b)$ is not allowed to change sign (except for a set of measure zero). For the special parameterization (1.4) this was guaranteed through the conditions in (1.5). These conditions were also easy to check in the numerical realization. If, in the general case, we used bilinear finite element functions for the approximation of a, b, and c, (1.8) would turn into a nonlinear minimization problem with respect to the coefficients of a, b, and c governed by nonlinear constraints in each node of the finite element grid of Ω. This is much too involved.

Therefore, we will present a numerical realization based on moving grids which is much faster and yields excellent results.

The method of moving grids is described in the next section. In Section 3 we combine it with regularization for surface representations. Finally, numerical results are presented in Section 4.

2. Moving grids: The deformation method

Moving grids have been developed for the numerical solution of time dependent partial differential equations. A drawback of using fixed grids occurs when the solution of the PDE exhibits large variations due to, for example, shock waves or moving fronts. Due to its static feature the fixed grid is unable to efficiently and accurately resolve such variations. One can improve this by using adaptive grids. The idea is to generate the grids such that nodes will be concentrated in regions where the solution changes rapidly and fewer grid points are used in regions where negligible changes occur.

There are essentially two strategies for grid adaption: *local refinement* and *moving grids*. In local refinement nodes are inserted where and when they are needed. This method is flexible and easy to conform the boundary. However, the solver and data structure have to be modified after insertion or deletion of nodes. For moving grids the total number of nodes and the connectivity between them is fixed. The nodes are redistributed where and when they are needed. There are different methods to move the nodes around. We only describe the so called *traditional deformation method*, which is based on a result from differential geometry. It provides direct control over the cell size of the adaptive grid and the node velocities are directly determined. It turned out that it is also efficient for the numerical calculation of discontinuous solutions of ill-posed problems in combination with regularization for surface representations. The description follows the lines in [8].

Suppose that u satisfies

$$u_t = L(u)\,, \tag{2.1}$$

where L is a differential operator defined on a physical domain $\Omega \subset \mathbb{R}^2$. The idea is now to construct a transformation $\phi : \tilde{\Omega} \times [0, T] \to \Omega$ which moves a fixed number of grid points on Ω to adapt to the numerical solution. Of course ϕ must be one to one and onto. It is well known that if the Jacobian determinant of a transformation ϕ is positive in $\tilde{\Omega}$, then ϕ is one to one in all of $\tilde{\Omega}$. This ensures that the grid will not fold onto itself. Therefore, the deformation method constructs ϕ such that $\det \nabla\phi = m(\phi, t)$, where m is a positive monitor function. This assures precise control over the cell size relative to the fixed initial grid.

Suppose that the solution of (2.1) has been computed at time step $t = t_{k-1}$ and a preliminary computation has been done at time level $t = t_k$. Assume that we have some positive error estimator $e(\eta, t)$ at the time steps t_{k-1} and t_k. Let us define the monitor function

$$m(\eta, t) = \frac{\gamma(t)}{e(\eta, t)} ,$$

where $\gamma(t)$ is a positive scaling parameter so that

$$\int_\Omega \frac{1}{m(\eta, t_k)} d\eta = |\tilde{\Omega}| \tag{2.2}$$

holds. Note that m is small in regions where the error is large and large in regions where the error is small. We then seek a transformation $\phi : \tilde{\Omega} \times [t_{k-1}, t_k] \to \Omega$ such that

$$\begin{aligned} \det \nabla\phi(\xi, t) &= m(\phi(\xi, t), t) , \quad \xi \in \tilde{\Omega} , \ t_{k-1} \le t \le t_k , \\ \phi(\xi, t_{k-1}) &= \phi_{k-1}(\xi) , \qquad \xi \in \tilde{\Omega} , \end{aligned} \tag{2.3}$$

where ξ is a grid node of an initial grid and $\phi_{k-1}(\xi)$ represents the coordinates of the node at $t = t_{k-1}$. In addition we specify that $\phi(\xi) \in \partial\Omega$ for all $\xi \in \partial\tilde{\Omega}$. Note that (2.2) is necessary for (2.3) to be true and that (2.3) ensures that the size of the transformed cells will be proportional to m, i.e., the grid will be appropriately condensed in regions of high errors and strechted in regions of small errors.

The solution of (2.3) can be done efficiently via the following two steps: the first step is to find a vector field $\rho(\eta, t)$ satisfying

$$\begin{aligned} \operatorname{div} \rho(\eta, t) &= -\frac{\partial}{\partial t}\left(\frac{1}{m(\eta, t)}\right) , \quad \eta \in \Omega , \ t_{k-1} \le t \le t_k , \\ \langle \rho(\eta, t), n(\eta) \rangle &= 0 , \qquad\qquad\qquad \eta \in \partial\Omega , \ t_{k-1} \le t \le t_k , \end{aligned}$$

where n denotes the outward normal to $\partial\Omega$. The vector field ρ can be calculated by solving for w in the scalar Poisson equation (for fixed t)

$$\Delta w(\eta, t) = -\frac{\partial}{\partial t}\left(\frac{1}{m(\eta, t)}\right) , \ \eta \in \Omega , \qquad \frac{\partial w}{\partial n} = 0 , \ \eta \in \partial\Omega , \tag{2.4}$$

and then setting $\rho = \nabla w$. Note that the constraint

$$\int_\Omega \frac{\partial}{\partial t}\left(\frac{1}{m(\eta, t)}\right) d\eta = 0$$

for the Neumann boundary condition is satisfied due to (2.2).

The second step is to solve for $\phi(\xi,t)$ from the ODE system (for fixed $\xi \in \tilde{\Omega}$)

$$\begin{aligned} \frac{d}{dt}\phi(\xi,t) &= m(\phi(\xi,t),t)\nabla w(\phi(\xi,t),t)\,, \quad t_{k-1} \le t \le t_k\,, \\ \phi(\xi,t_{k-1}) &= \phi_{k-1}(\xi)\,. \end{aligned} \tag{2.5}$$

It was shown in [11] that this two step procedure actually yields the solution of (2.3).

3. Numerical realization

In this section we want to show how a combination of regularization for surface representations with moving grids for adapting the finite element grid yields a fast and effective method to identify discontinuities in two-dimensional problems.

From our numerical calculations for the special parameterization (1.4) in [7, 6] we saw that, as expected, the grid is much finer where the solution is steep. The idea is now to replace the minimization problem (1.8) by a sequence of problems where the minimization is done only with respect to c and afterwards (a,b) is adjusted via moving grids according to an appropriate monitor function.

Obviously, in our case the time t in the moving grid method only plays an artificial role: it is somehow a homotopy parameter connecting the initial grid at stage $t=0$ with the final one at e.g. $t=1$. Let us assume that (a_k,b_k,c_k) denotes the parameterization at iteration step k. Then a proper choice for the monitor function is such that

$$\begin{aligned} m(\phi_k(\xi),0) &= \det \nabla_\xi \phi_k(\xi)\,, & \xi \in \tilde{\Omega}\,, \\ m(\eta,1) &= \frac{1}{\gamma(1+\beta|\nabla_\eta f_k(\eta)|^2)^{\frac{1}{2}}}\,, & \eta \in \Omega\,, \end{aligned} \tag{3.1}$$

where $f_k(\eta) = c_k(\phi_k^{-1}(\eta))$, $\phi_k(\xi) = (a_k(\xi), b_k(\xi))$, $\gamma > 0$ is a constant to satisfy (2.2), and $\beta > 0$ is a constant for deciding how much the steepness should influence the cell size. In the following we assume that $\Omega = \tilde{\Omega} = [0,1]^2$.

Since we want to avoid the calculation of the inverse of ϕ_k we propose the following way of calculating ϕ_{k+1}:

$$\phi_{k+1}(\xi) := \phi(\xi,1) \qquad \text{and} \qquad \phi(\xi,t) := \phi_k(d(\xi,t)) \tag{3.2}$$

According to (2.3) we obtain

$$\begin{aligned} \det \nabla\phi(\xi,t) &= \det \nabla\phi_k(d(\xi,t)) \det \nabla d(\xi,t) \\ &= m(\phi(\xi,t),t) \;=\; m(\phi_k(d(\xi,t)),t)\,, \end{aligned}$$

and hence $d : \tilde{\Omega} \times [0,1] \to \tilde{\Omega}$ satisfies

$$\begin{aligned} \det \nabla d(\xi,t) &= \tilde{m}(\xi,t)\,, \quad \xi \in \tilde{\Omega}\,,\ 0 \le t \le 1\,, \\ d(\xi,0) &= \xi\,, \qquad\quad \xi \in \tilde{\Omega}\,, \end{aligned} \tag{3.3}$$

where

$$\tilde{m}(\xi,t) := \frac{m(\phi_k(d(\xi,t)),t)}{\det \nabla\phi_k(d(\xi,t))}\,.$$

Note that (3.3) looks like (2.3) with ϕ and m replaced by d and $\tilde{m}$, respectively. As monitor function $\tilde{m}$, taking into account (3.1), we have chosen

$$\tilde{m}(\xi,t) := \frac{1}{(1-t) + t\gamma \det \nabla_\xi \phi_k(\xi)(1+\beta|\nabla_\eta f_k(\phi_k(\xi))|^2)^{\frac{1}{2}}} .$$

The advantage of this choice will be seen later. Together with the chain rule we obtain that

$$\begin{aligned} \tilde{m}(\xi,t) &= \frac{1}{(1-t)+t\gamma D(a_k,b_k,c_k,\beta)^{\frac{1}{2}}(\xi)} \\ D(a,b,c,\beta) &:= (a_u b_v - a_v b_u)^2 + \beta((b_v c_u - b_u c_v)^2 + (a_u c_v - a_v c_u)^2) \end{aligned} \tag{3.4}$$

Since we have to satisfy (2.2), γ in (3.4) is chosen such that

$$1 = \gamma \int_\Omega D(a_k,b_k,c_k,\beta)^{\frac{1}{2}}(\xi)\, d\xi .$$

For our numerical calculations we approximate (a_k, b_k, c_k) by bilinear finite element functions on a uniform grid, i.e.,

$$a_k = \sum_{i,j=0}^n a_{ij}^k \varphi_{ij} , \quad b_k = \sum_{i,j=0}^n b_{ij}^k \varphi_{ij} , \quad c_k = \sum_{i,j=0}^n c_{ij}^k \varphi_{ij} , \tag{3.5}$$

where φ_{ij} are the bilinear finite element functions satisfying $\varphi_{ij}(\frac{\tilde{i}}{n},\frac{\tilde{j}}{n}) = \delta_{i\tilde{i}}\delta_{j\tilde{j}}$. Besides the restriction that (a_k, b_k) is one to one we need that (a_k, b_k) maps the boundary of $\Omega = [0,1]^2$ onto itself. This is achieved if the nodes of the grid, (a_{ij}^k, b_{ij}^k) satisfy

$$a_{0j} = 0 = b_{i0} , \qquad a_{nj} = 1 = b_{in} , \qquad 0 \le i,j \le n . \tag{3.6}$$

We obtain an update for (a_k, b_k) via (3.2), (3.3). We know from the last section that the solution of (3.3) can be computed by a two-step procedure:
According to (2.4), as first step we have to solve the scalar Poisson equation

$$\Delta w(\xi) = 1 - \gamma D(a_k,b_k,c_k,\beta)^{\frac{1}{2}}(\xi) , \; \xi \in \tilde{\Omega} , \qquad \frac{\partial w}{\partial n} = 0 , \; \eta \in \partial\tilde{\Omega} . \tag{3.7}$$

Note that due to the special choice of the monitor function in (3.4), the function w does not depend on t and, hence, has to be calculated only once for each iteration step. We do this by looking for the weak solution of this equation using bilinear finite element functions on the same uniform grid as chosen in (3.5), i.e.

$$w = \sum_{i,j=0}^n w_{ij}\varphi_{ij} . \tag{3.8}$$

Note that then the Neuman boundary condition is trivially satisfied if

$$w_{i0} = w_{i1} , \; w_{in-1} = w_{in} , \quad w_{0j} = w_{1j} , \; w_{n-1j} = w_{nj} , \qquad 0 \le i,j \le n .$$

We obtain a linear system of equations where the $(n-1)^2 \times (n-1)^2$ matrix is sparse. It has a blocktridiagonal structure and each block is again tridiagonal. For the calculation of the right hand side of this system we approximate $D(a_k,b_k,c_k,\beta)^{\frac{1}{2}}(\xi)$ via bilinear spline interpolation. Note that due to our choice of (a_k, b_k, c_k) in (3.5) the function $D(a_k, b_k, c_k, \beta)^{\frac{1}{2}}(\xi)$ is only piecewise continuous, where discontinuities occur on the grid lines $(\frac{i}{n}, v)$ and $(u, \frac{j}{n})$. Using bilinear spline interpolation we approximate it by a continuous function. As coefficients of the splines we take the average of all possible values of $D(a_k, b_k, c_k, \beta)^{\frac{1}{2}}(\xi)$ (up to four) at the nodes $\xi = (\frac{i}{n}, \frac{j}{n})$. The solution of the corresponding system is calculated using an SSOR method.

The next step is now to solve the ODE system (2.5) for fixed node $\xi = (\frac{i}{n}, \frac{j}{n})$, i.e.

$$\begin{aligned} \frac{d}{dt} d(\xi, t) &= \tilde{m}(d(\xi, t), t) \nabla w(d(\xi, t)) \, , \quad 0 \le t \le 1 \, , \\ d(\xi, 0) &= \xi \, , \end{aligned} \tag{3.9}$$

with $\tilde{m}$ as in (3.4). This system is solved with the classical fourth order Runge Kutta method. Since ∇w for w as in (3.8) is not continuous, we calculate ∇w approximately using central difference quotients, where we take care that nodes at the boundary do not leave the boundary (see (3.6)).

Due to (3.2), we obtain the new grid points via

$$(a_{ij}^{k+1}, b_{ij}^{k+1}) = \phi_k(d((\tfrac{i}{n}, \tfrac{j}{n}), 1)) \, . \tag{3.10}$$

Although from the theory of the last section a folding of the grid is not possible, it can happen that due to numerical approximations for very small cells $\det \nabla \phi_{k+1}$ becomes negative. If this is the case we smooth the nodes by the following simple averaging procedure

$$a_{ij}^{k+1} := \tfrac{1}{9} \sum_{\mu,\nu=-1}^{1} a_{i+\mu\, j+\nu}^{k+1} \, , \qquad b_{ij}^{k+1} := \tfrac{1}{9} \sum_{\mu,\nu=-1}^{1} b_{i+\mu\, j+\nu}^{k+1} \, . \tag{3.11}$$

The formula above is used for interior nodes. For boundary nodes the formula is adjusted in accordance with (3.6).

Summarizing we end up in the following algorithm as realization of regularization for surface representations using moving grids:

Algorithm 3.1.

Step 0: As initial grid choose $a_{ij}^0 := \frac{i}{n}$, $b_{ij}^0 := \frac{j}{n}$, i.e. a uniform grid, and set $k = 0$.

Step 1: Minimize the functional

$$\|G(a_k, b_k, c) - g^\delta\|_{L^2}^2 + \alpha \|\nabla c\|_{L^2}^2 \, , \qquad \alpha > 0 \, ,$$

with respect to c (with c as in (3.5)). The solution is denoted by c_k.

Step 2: Update (a_k, b_k) using moving grids, i.e.:
(a) Solve (3.7) with an SSOR method as described above.
(b) Solve (3.9) with the classical fourth order Runge Kutta method.
(c) Compute the coordinates of the new grid via (3.10).
(d) Adjust the nodes according to (3.11) if necessary.

Step 3: If the change in (a_k, b_k) was large enough, goto Step 1, otherwise Stop.
The final iterate is denoted by $(a_\alpha^\delta, b_\alpha^\delta, c_\alpha^\delta)$.

Note that in this algorithm $\det \nabla \phi$ is always positive. It is not possible as for the approach with the special parameterization (1.4) that $\det \nabla \phi$ vanishes (see (1.5)). However, the cell size can become arbitrarily small and hence the regularized solutions $f_\alpha^\delta = c_\alpha^\delta((a_\alpha^\delta, b_\alpha^\delta)^{-1})$ can be arbitrarily steep.

4. Numerical results

In the last section we described the numerical realization of regularization for surface representations using moving grids for adapting the finite element grid. We will now present several examples where we applied Algorithm 3.1 to two-dimensional linear integral equations (1.6). The operator G of the transformed equation, given in (1.7), is linear with respect to c. Therefore, the minimization problem in Step 1 of Algorithm 3.1 can be computed explicitley by solving the linear system

$$(A + \alpha B)c = r\,, \tag{4.1}$$

where

$$\begin{aligned} A &:= [\langle T_k\varphi_{ij}, T_k\varphi_{\tilde{i}\tilde{j}} \rangle]\,, \\ B &:= [\langle \nabla\varphi_{ij}, \nabla\varphi_{\tilde{i}\tilde{j}} \rangle]\,, \\ r &:= [\langle T_k\varphi_{ij}, g^\delta \rangle]\,, \\ T_k &:= G(a_k, b_k, \cdot)\,. \end{aligned} \tag{4.2}$$

The solution c is the coefficient vector used in (3.5) to express c_k.

We have considered deblurring problems where the kernel of the integral equation (1.6) is given by

$$k(s,t,x,y) = \tilde{k}(|(s,t)-(x,y)|)\,, \qquad \tilde{k}(r) = \begin{cases} (1-\frac{1}{\rho}r^2)^4 & \text{if } r^2 \le \rho\,, \\ 0 & \text{else}\,. \end{cases} \tag{4.3}$$

For ρ we have chosen the values $\frac{1}{100}$ and $\frac{1}{10}$. The bilinear finite element functions in (3.5) were chosen with $n = 20$. The regularization parameter in Step 1 of Algorithm 3.1 was chosen to be $\alpha = 10^{-9}$.

The most time consuming step in each iteration is to build the full (441×441)-matrix A (more than 95% of the CPU time). A speed-up of the program could, therefore, be achieved via parallelization. Nevertheless, this algorithm was already 20 times faster than the one we used to calculate the regularized solutions in [7] with the special parameterization (1.4).

Example 4.1. As kernel of the integral equation we used (4.3) with $\rho = \frac{1}{100}$. The exact solution is given by

$$f(x,y) := \chi_{B_{0.2}(0.6,0.5)}\,.$$

For this solution we were able to compute the right hand side g exactly. Figure 4.1 shows the regularized solution calculated with Algorithm 3.1: the left picture shows the initial grid, the picture in the middle the grid after 10 iterations, and the right picture shows the graph of the 10th iterate. As we can see the location of the discontinuities is identified quite well after only 10 iterations.

In Figures 4.2 we show that the method is robust with respect to data noise. We added random noise to the exact data such that δ was equal to 5% of the norm of the exact data.

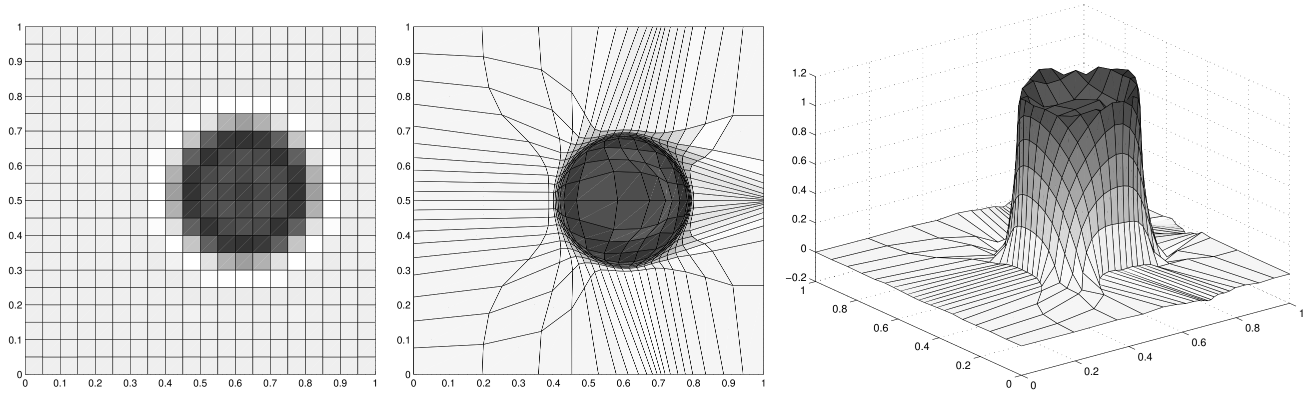

Figure 4.1: Initial grid, grid and graph of 10th iterate for Example 4.1

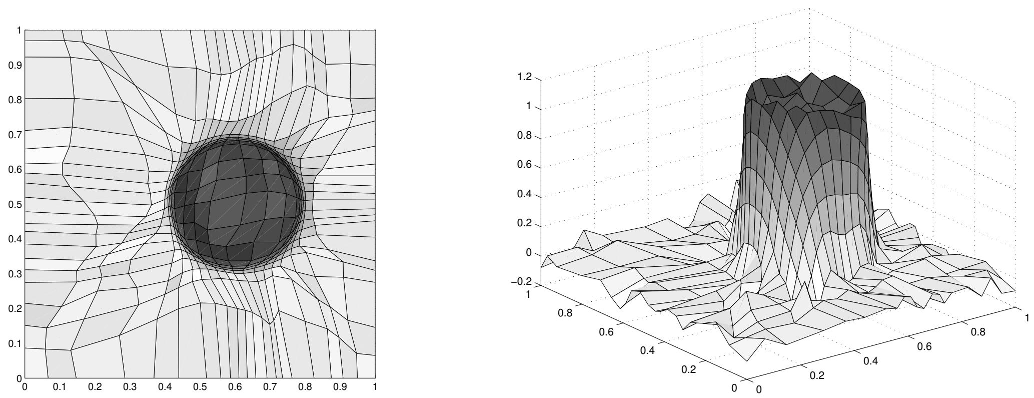

Figure 4.2: Grid and graph of 10th iterate for perturbed data in Example 4.1

Example 4.2. In this example we treat Example 2 from [7]; $\rho = \frac{1}{10}$ in (4.3). The exact solution is given by

$$f(x,y) := 10\chi_{B_{0.2}(0.5,0.3)} \,.$$

Also for this solution we were able to compute the right hand side g exactly. Figure 4.3 shows the comparison of regularization for surface regularization with the special parameterization (1.4) and BV-regularization. Since the exact solution has its discontinuities along a circle, we have not been able to identify the location of the discontinuities in [7]. In Figure 4.4 we show the result obtained with the new approach in Algorithm 3.1. One can see quite well where the discontinuities are. The result is not as good as in Example 4.1. The reason for this is that the radius of the point spread function (the kernel of the integral equation) is three times larger. A better result could be possibly obtained by taking a finer quadrature rule for the calculation of the integrals occuring in the elements of the matrix A (see (4.2)). However, this would also increase the CPU time. A comparison of Figures 4.3 and 4.4 shows that Algorithm 3.1 yields better results than BV-regularization.

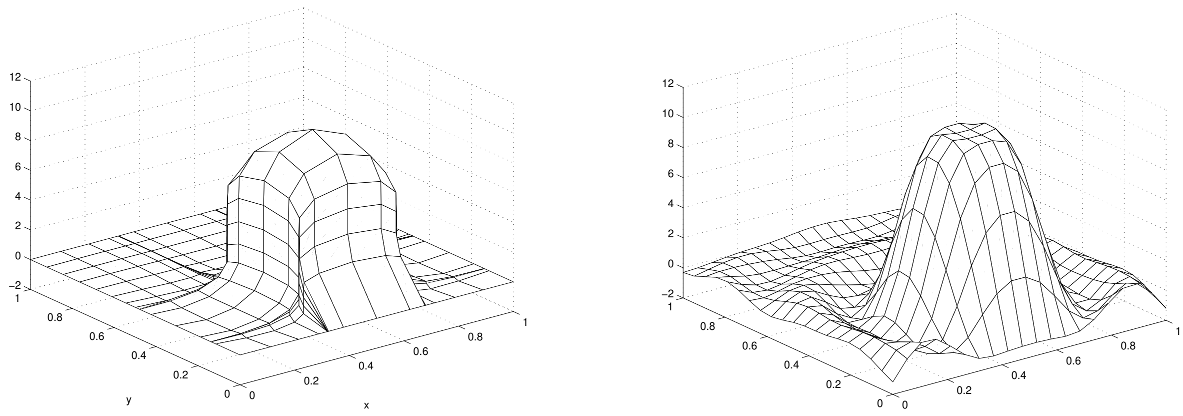

Figure 4.3: Surface- and BV-Regularization from [7] for Example 4.2

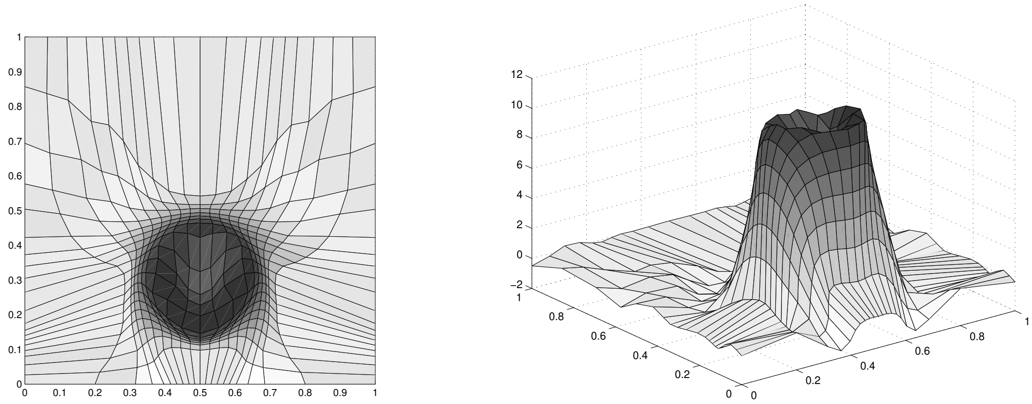

Figure 4.4: Grid and graph of 10th iterate for perturbed data in Example 4.2

Example 4.3. In this example we treat Example 3 from [7]; $\rho = \frac{1}{10}$ in (4.3). The exact solution is given by

$$f(x, y) := 8\chi_{B_{0.1}(0.2,0.5)} + 10\chi_{B_{0.15}(0.8,0.3)} + 5\chi_{B_{0.2}(0.5,0.7)}\,.$$

As for Example 4.2 we show the comparison of regularization for surface regularization with the special parameterization (1.4) and BV-regularization as well as the result obtained with the Algorithm 3.1 in Figures 4.5 and 4.6, respectively. Similar arguments hold as in Example 4.2. In Figure 4.6 black circles indicate where the real locations of the discontinuities are. One can see that $n = 20$ is too small to be able to identify the locations accurately enough. An increase of n in (3.5), however, would increase the needed storage and the CPU-time by a quite a bit. Note that the storage requirement and the number of operations for computing A grows with $O(n^4)$, solving the linear system in (4.1) even grows with $O(n^6)$.

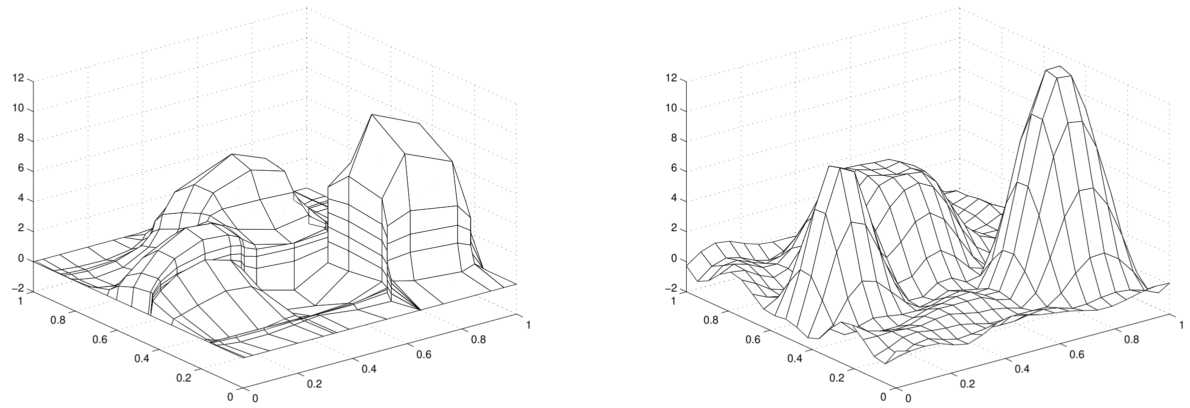

Figure 4.5: Surface- and BV-Regularization from [7] for Example 4.3

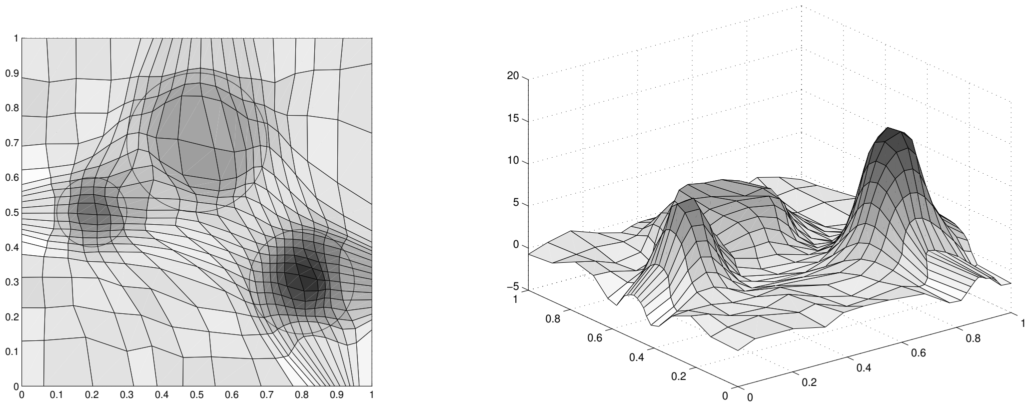

Figure 4.6: Grid and graph of 10th iterate for perturbed data in Example 4.3

Example 4.4. In this example the kernel of the integral equation was chosen as in Example 4.1, i.e., we used (4.3) with $\rho = \frac{1}{100}$. The exact solution is given by

$$f(x,y) := 8\chi_{A_1} + 10\chi_{A_2} + 5\chi_{A_3}\,,$$

where A_1, A_2, and A_3 are areas bounded by straight lines as follows:

$$\begin{aligned} A_1 &:= \{(x,y)\in\mathbb{R}^2\,:\,70x-30y+9\geq 0\,, 2x+10y-9\leq 0\,, 90x-70y+7\leq 0\,,\\ &\qquad 4x+10y-7\geq 0\}\,,\\ A_2 &:= \{(x,y)\in\mathbb{R}^2\,:\,3x+10y-3\geq 0\,, 8x-10y+2\geq 0\,, 10x+4y-5\leq 0\}\,,\\ A_3 &:= \{(x,y)\in\mathbb{R}^2\,:\,7x+10y-10\leq 0\,, 10x+3y-10\leq 0\,, 5x+10y-6\geq 0\,,\\ &\qquad 10x-5y-4\geq 0\}\,. \end{aligned}$$

The right hand side g was approximated by bilinear splines on a uniform grid with nodes $(\frac{i}{100}, \frac{j}{100})$, $0 \leq i, j \leq 100$. The node values were calculated by using a very fine quadrature rule in (1.6), namely a 4-point Gauss quadrature rule on subintervalls of length $\frac{1}{100}$ which is five times finer than the one used in the identification algorithm.

Figure 4.7 shows the regularized solution calculated with Algorithm 3.1: the left picture shows the initial grid, the picture in the middle the grid after 10 iterations, and the right picture shows the graph of the 10th iterate. Black lines in the picture in the

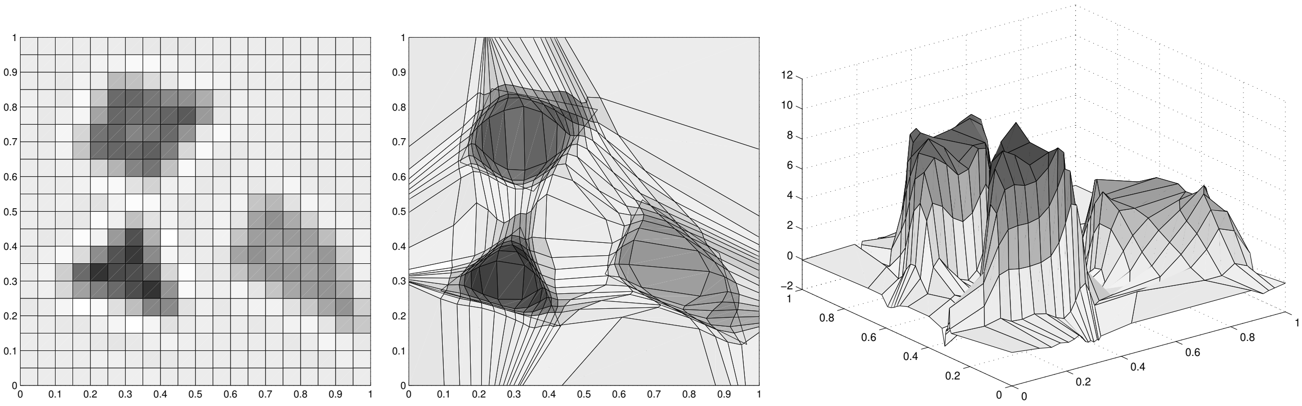

Figure 4.7: Initial grid, grid and graph of 10th iterate for Example 4.4

middle indicate the locations of the discontinuities. As one can see, these locations are identified quite well except for the sharp corners. Better results could be obtained by increasing n (compare Example 4.3).

We think that all the examples above show quite well that regularization for surface representations in combination with moving grids is a good alternative to bounded variation regularization. Since the location of the discontinuities is found rather fast one could possibly get a faster method by the following hybrid approach: first identify the location of the discontinuities using Algorithm 3.1 and then use a standard regularization method to identify the function values having the knowledge where the discontinuities are.

These ideas and the application of this new approach to parameter estimation problems as well as theoretical aspects of convergence of these finite-dimensional regularized solutions will be future work.

References

[1] R. Acar and C. Vogel, *Analysis of bounded variation penalty methods for ill-posed problems*, Inverse Problems 10 (1994), 1217–1229.

[2] G. Chavent and K. Kunisch, *Regularization of linear least squares problems by total bounded variation*, ESAIM: Control, Optimisation and Calculus of Variations 2 (1997), 359–376.

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

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

[5] S. Kindermann and A. Neubauer, *Identification of discontinuous parameters by regularization for curve representations*, Inverse Problems 15 (1999), 1559–1572.

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

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

[8] G. Liao, F. Liu, G. dela Pena, D. Peng, and S. Osher, *Level set based deformation methods for adaptive grids*, J. Comp. Phys. 159 (2000), 103–122.

[9] A. Neubauer and O. Scherzer, *Regularization for curve representations: uniform convergence for discontinuous solutions of ill-posed problems*, SIAM J. Appl. Math. 58 (1998), 1891–1900.

[10] L. I. Rudin, S. Osher, and E. Fatemi, *Nonlinear total variation based noise removal algorithms*, Phys. D 60 (1992), 259–268.

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