Vietnam Journal of Mathematics manuscript No.
(will be inserted by the editor)

Faster gradient descent and the efficient recovery of images

Hui Huang · Uri Ascher

November 2013

Abstract Much recent attention has been devoted to gradient descent algorithms where the steepest descent step size is replaced by a similar one from a previous iteration or gets updated only once every second step, thus forming a *faster gradient descent method*. For unconstrained convex quadratic optimization these methods can converge much faster than steepest descent. But the context of interest here is application to certain ill-posed inverse problems, where the steepest descent method is known to have a smoothing, regularizing effect, and where a strict optimization solution is not necessary.

Specifically, in this paper we examine the effect of replacing steepest descent by a faster gradient descent algorithm in the practical context of image deblurring and denoising tasks. We also propose several highly efficient schemes for carrying out these tasks independently of the step size selection, as well as a scheme for the case where both blur and significant noise are present.

In the above context there are situations where many steepest descent steps are required, thus building slowness into the solution procedure. Our general conclusion regarding gradient descent methods is that in such cases the faster gradient descent methods offer substantial advantages. In other situations where no such slowness buildup arises the steepest descent method can still be very effective.

Keywords Artificial time · Gradient descent · Image processing · Denoising · Deblurring · Lagged steepest descent · Regularization

1 Introduction

The tasks of deblurring and denoising are fundamental in image restoration and have received a lot of attention in recent years; see, e.g., [33,10,27,13,23] and

Hui Huang
Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, China
E-mail: hui.huang@siat.ac.cn

Uri Ascher
Department of Computer Science, University of British Columbia, Vancouver, Canada
E-mail: ascher@cs.ubc.ca

references therein. These are inverse problems, and they can each be formulated as the recovery of a 2D surface model m from observed (or given) 2D data b based on the equation

$$b = F(m) + \epsilon. \tag{1}$$

Here $F(m)$ is the predicted data, which is a linear function of the sought model m, and ϵ is additive noise. Both m and b are defined on a rectangular pixel grid, and we assume without loss of generality that this grid is square, discretizing the unit square Ω with n^2 square cells of length $h = 1/n$ each. Thus, $m = \{m_{i,j}\}_{i,j=0}^{n}$.

In the sequel it is useful to consider m in two other forms. In the first of these, m is reshaped into a vector in $\mathbb{R}^N$, $N = (n+1)^2$. Doing the same to b and F, we can write the latter as a matrix-vector multiplication, given by

$$F(m) = Jm, \tag{2}$$

where the *sensitivity* matrix $J = \frac{\partial F}{\partial m}$ is constant. Note that N can easily exceed $1,000,000$ in applications; however, there are fast matrix-vector multiplication algorithms available to calculate F for both deblurring [33] and (trivially) denoising.

The second form is really an extension of m into a piecewise smooth function $m(x, y)$ defined on Ω. This allows us to talk about integral and differential terms in m, as we proceed to do below, with the understanding that these are to be discretized on the same grid to attain their true meaning.

Both inverse problems are ill-posed, the deblurring one being so even without the presence of noise [10]. Some regularization is therefore required. Tikhonov-type regularization is a classical way to handle this, leading to the optimization problem

$$\min_{m} \; T(m;\beta) \equiv \frac{1}{2}\|Jm - b\|^2 + \beta R(m), \tag{3}$$

where $R(m)$ is the regularization operator and $\beta > 0$ is the regularization parameter [16,32]. It is important to realize that the determination of β is part of the regularization process. The least-squares norm $\|\cdot\|$ is used in the data-fitting term for simplicity, and it corresponds to the assumption that the noise is Gaussian. The necessary condition for optimality in (3) yields the algebraic system

$$G(m) \equiv \nabla_m T(m;\beta) \equiv J^T(Jm - b) + \beta R_m = 0, \tag{4}$$

where R_m is the gradient of R.

The choice of $R(m)$ should incorporate *a priori* information, such as piecewise smoothness of the model to be recovered, which is the vehicle for noise removal. Consider the one-parameter family of Huber switching functions, whereby R is defined as the same-grid discretization of

$$R(m) = \int_{\Omega} \rho(|\boldsymbol{\nabla} m|), \tag{5a}$$

$$\rho(\sigma) = \rho(\sigma;\gamma) = \begin{cases} \sigma, & |\sigma| \geq \gamma \\ \sigma^2/(2\gamma) + \gamma/2, & |\sigma| < \gamma \end{cases}. \tag{5b}$$

The corresponding necessary conditions (4) are the same-grid discretization of an elliptic PDE with the leading differential term given by

$$\begin{aligned} R_m(m) &= L(m)\cdot m, \\ L(m) &= -\nabla\cdot\left(\frac{1}{|\boldsymbol{\nabla} m|_\gamma}\boldsymbol{\nabla}\right), \quad |\boldsymbol{\nabla} m|_\gamma = \max\{\gamma, |\boldsymbol{\nabla} m|\}. \end{aligned} \tag{6}$$

Thus, for γ large enough so that always $\max\{\gamma, |\boldsymbol{\nabla} m|\} = \gamma$ the objective function is a convex quadratic and (4) is a linear symmetric positive definite system. However, this choice smears out image edges. At the other extreme, $\gamma = 0$ yields the total variation (TV) regularization [11,30,33,17]. But this requires modification when m is flat to avoid blowup in L. For most examples reported here we employ the adaptive choice

$$\gamma = \frac{h}{|\Omega|}\int_\Omega |\boldsymbol{\nabla} m|, \tag{7}$$

proposed in [3], which basically sets a resolution-dependent switch, modifying TV.

For the approximate solution of the optimization problem (3), consider combining two iterative techniques. The first iteration applies for the case where γ is small enough so that R_m is nonlinear in m, as is the case when using (7). In this case a fixed point iteration called lagged diffusivity (or IRLS) is employed [33], where m^0 is an initial guess and m^{k+1} is defined as the solution of the linear problem

$$J^T(Jm - b) + \beta L(m^k)m = 0, \tag{8}$$

for $k = 0, 1, 2, \ldots$. See [3,9] for a proof of global convergence of this iteration. In practice a rapid convergence rate is observed at first, typically slowing down only in a regime where the iteration process would be cut off anyway.

Next, for the solution of (3) or the potentially large linear system (8) consider the iterative method of gradient descent given by

$$m^{k+1} = m^k - \tau_k\left(J^T(Jm^k - b) + \beta L(m^k)m^k\right), \quad k = 0, 1, 2, \ldots. \tag{9}$$

Such a method was advocated for the denoising problem already in [30,28], and it corresponds to a forward Euler discretization of the embedding of the elliptic PDE in a parabolic PDE with artificial time t, written as

$$\frac{\partial m}{\partial t} = -\left(J^T(Jm - b) + \beta R_m\right), \quad t \geq 0. \tag{10}$$

See also [20].

The step size τ_k in (9) had traditionally been determined for the quadratic case by exact line search, yielding the steepest descent (SD) method. But this generally results in slow convergence, as slow as when using the best uniform step size, unless the condition number of $J^TJ + \beta L$ is small [1,24,2]. The convergence rate then is far slower than that of the method of conjugate gradients (CG). In recent years much attention has been paid to gradient descent methods where the steepest descent step size value from the previous rather than the current iteration is used [5], or where it gets updated only once every second iteration [29, 19]. These step selection strategies yield in practice a much faster convergence for the gradient descent method applied to convex quadratic optimization, although they are still slower than CG and their theoretical properties are both poorer and more mysterious [2]. Let us refer to these variants as *faster gradient descent* methods. These methods automatically combine an occasionally large step that may severely violate the forward Euler absolute stability restriction but yields rapid convergence with small steps that restore stability and smoothness. The resulting dynamical system is chaotic [14].

Faster gradient descent methods have seen practical use in the optimization context of quadratic objective functions subject to box constraints [12], especially when applied to compressed sensing and other image processing problems [18,6]. For unconstrained optimization they are generally majorized by CG, and the same holds true for their preconditioned versions. Here, however, the situation is more special. For one thing, the PDE (10) has in the case of denoising, where J is the identity, the interpretation of modeling the physical process of anisotropic diffusion (isotropic for sufficiently large γ). This has a smoothing effect that is desirable for the denoising problem. CG is also a smoother [21], but it does not have the same physical interpretation. Moreover, CG is more susceptible to perturbation caused by the lagged diffusivity or IRLS method, where the quadratic problem solved varies slightly from one iteration to the next (see also [14]). Related and important practically, the iteration process (9) need not be applied all the way to strict convergence, because a desirable regularization effect is obtained already after relatively few iterations. To understand this, recall that the parameter β in (3) is still to be determined. Its value affects the Tikhonov filter function and relates to the amount of noise present [33]. But it is well known that an alternative to the Tikhonov filter function is the exponential filter function—see, e.g., [33,8]—and the latter is approximately obtained by what corresponds to integrating (10) up to only a *finite time*; see [4,20] and references therein. In fact, upon integration to finite time one may optionally set $\beta = 0$, replacing the Tikhonov regularization altogether. The question now is, in the current problem setting, do faster gradient descent methods perform better than steepest descent, taking larger steps and yet maintaining the desired regularization effect at the same time? More specifically, does the more rapid convergence provided by the large steps and the regularization (i.e., piecewise smoothing) effect provided by the small steps automatically combine to form a method that achieves the desired effect in much fewer steps?

In [2] we started to answer this question for the denoising problem. Here we continue and extend that line of investigation. In Section 2 we complete the results presented in [2] and propose a new, efficient, explicit-implicit denoising scheme with an edge sharpening option.

In Section 3, the main section of this article, we apply the methods described above to the much harder deblurring problem, and also propose a new method for the case where there are both blur and significant noise present.

The two step size selections that are being compared for the deblurring problem can be written as

$$\text{SD:} \qquad \tau_k = \frac{(G(m^k))^T G(m^k)}{(G(m^k))^T (J^T J + \beta L(m^k)) G(m^k)}, \tag{11a}$$

$$\text{LSD:} \qquad \tau_k = \frac{(G(m^{k-1}))^T G(m^{k-1})}{(G(m^{k-1}))^T (J^T J + \beta L(m^{k-1})) G(m^{k-1})}. \tag{11b}$$

These are the steepest descent (SD) and lagged steepest descent (LSD) formulas for the case of large γ, where the matrix $-L$ is just the discretized Laplacian subject to natural boundary conditions, and they play a similar role for the nonlinear case in the context of lagged diffusivity.

All presented numerical examples were run on an Intel Pentium 4 CPU 3.2 GHz machine with 512MB RAM, and CPU times are reported in seconds. Conclusions are offered in Section 4.

2 Denoising

For the denoising problem we have $F(m) = m$, so the sensitivity matrix J is the identity, and it is trivially sparse and well-conditioned. In [2] we have considered the well-known gradient descent algorithm

$$m^0 = b, \tag{12a}$$

$$m^{k+1} = m^k - \tau_k R_m(m^k), \quad k = 0, 1, 2, \ldots, \tag{12b}$$

obtained as a special case of (9) upon rescaling the artificial time by β and then letting $\beta \to \infty$. This gets rid of the annoying need to determine β. The influence of the data is only through the initial conditions, i.e., it is a pure diffusion simulation. Correspondingly, in the step size definitions (11), G is replaced by R and $J^T J + \beta L$ is replaced by L.

The results in [2] clearly indicate not only the superior performance of the parameter selection (7) over using large γ but also the improved speed of convergence using LSD over SD, occasionally by a significant factor. For instance, cleaning the `Cameraman` image used below in Fig. 3, which is 256×256 and corrupted by 20% Gaussian white noise, agreeable results are obtained after 382 iterations using SD as compared to only 117 iterations using LSD. These numbers arise upon using the same relative error norm, defined by

$$e^k = \|m^{k+1} - m^k\| / \|m^{k+1}\|, \tag{13}$$

to stop the iteration in both cases. Unless otherwise noted we have used a sufficiently strict tolerance to ensure that the resulting images are indistinguishable. Similar results are obtained for other test images.

Still, these methods can often be further improved. With LSD, occasionally the larger step sizes could produce a slightly rougher image than desired (compare Figs. 4(f) and 4(e) in [2]), whereas SD occasionally simply takes too long. We may therefore wish to switch to solving the Tikhonov equations (4), which here read

$$m - b + \beta R_m = 0. \tag{14}$$

However, this brings back the question of effectively selecting the parameter β.

Fortunately, there is a fast way to determine β, at least for the case where the noise is Gaussian. Using (5) with (7), the reconstructed model becomes much closer to the true image than to the noisy one already after a few LSD or even SD iterations, see [2]. The computable misfit, defined by

$$\eta_k = \|m^k - b\| / n, \tag{15}$$

therefore provides a very good, cheaply obtained approximation for the noise level. Thus, denoting the roughly denoised image by $\bar{m}$, we have $\int_\Omega |\bar{m} - b|^2 \approx \eta^2$. According to the discrepancy principle, we wish our intended reconstruction m to maintain this constraint invariant in "time", that is

$$\frac{d}{dt}\int_\Omega |m(t) - b|^2 = 0,$$

which by (10) yields

$$0 = \int_\Omega (m-b)\frac{\partial m}{\partial t} = -\left(\int_\Omega |m-b|^2 + \beta\int_\Omega (m-b)R_m\right).$$

This therefore leads us to determine β as

$$\beta = -\frac{\int_\Omega |\bar{m} - b|^2}{\int_\Omega (\bar{m} - b)R_m(\bar{m})}. \tag{16}$$

See also [27, Section 11.2].

Now that we have β (and $\bar{m}$ for an initial guess) we solve the equations (14) with (5) and (7), which corresponds to an implicit artificial-time integration step, using a combination of lagged diffusivity (IRLS) and CG with a multigrid preconditioner; see [3] for details.

In Figs. 1(d) and 1(e), we compare the result of this hybrid explicit-implicit scheme with that of the pure explicit scheme (12). The pre-denoised image after 17 SD steps, shown in Fig. 1(c), acts as a warm start and produces a good regularization upon setting $\beta = 0.083$ by (16) for the following implicit process. Then, after only 3 IRLS iterations, the denoised image in Fig. 1(e) looks already quite comparable with—even slightly smoother than—the one in Fig. 1(d), which is continually denoised using SD for a stricter relative error tolerance. The processing time of the hybrid scheme is only a small fraction of that required by the explicit scheme, even with the faster LSD step size. Figs. 4(e) in [2] and 3(d) in the present article tell us essentially the same story for a different example.

Sharpening the reconstructed image

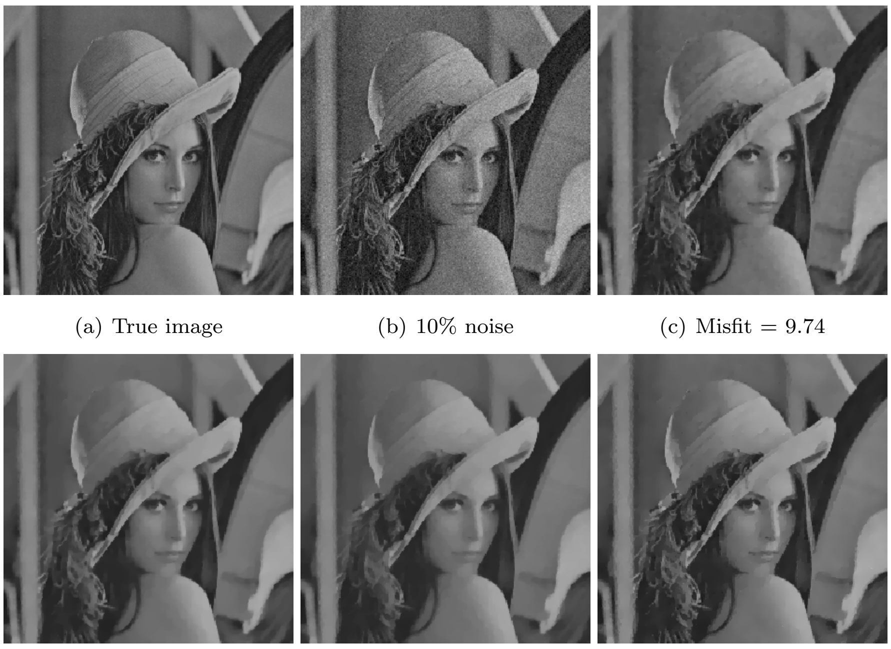
Besides employing the implicit method to improve the quality of pre-denoised images, we may wish to sharpen the reconstructed image in a way that TV cannot provide [31]. One possibility is to sharpen them by making $R(m)$ in (12) gradually more and more non-convex [28,7,31,15] and so reducing penalty on large jumps. This can be done by replacing the Huber switching function depicted in Fig. 2(a) with the Tukey function depicted in Fig. 2(b).

The Tukey function, scaled similarly to the Huber function, is defined by

$$\rho(\sigma) = \begin{cases} \frac{1}{3} & |\sigma| \geq \hat{\gamma}, \\ (\frac{\sigma^2}{\hat{\gamma}^2} - \frac{\sigma^4}{\hat{\gamma}^4} + \frac{\sigma^6}{3\hat{\gamma}^6}) & |\sigma| < \hat{\gamma}. \end{cases} \tag{17}$$

Since

$$\rho'(\sigma) = \phi(\sigma) = \begin{cases} 0 & |\sigma| \geq \hat{\gamma}, \\ \frac{2\sigma}{\hat{\gamma}^2}(1 - (\frac{\sigma}{\hat{\gamma}})^2)^2 & |\sigma| < \hat{\gamma}, \end{cases}$$



(a) True image (b) 10% noise (c) Misfit = 9.74

(d) Misfit = 11.17 (e) Misfit = 11.93 (f) Misfit = 10.39

Fig. 1 Comparing different denoising schemes on the 256 × 256 `Lena` image: (a) true image; (b) image corrupted by 10% Guassian white noise serves as data; (c) image denoised using SD step sizes, stopping when e^k is below 10^{-4}, 17 iters; (d) image denoised using SD step sizes, stopping when e^k is below 10^{-5}, 309 iters, 51.9 sec; (e) image denoised by 3 IRLS iterations with $\beta = 0.083$ after 17 SD steps, the total CPU time = 2.7 + 5.2 = 7.9 sec; (f) image sharpened using 10 SD steps with Tukey function after 17 pre-denoising SD steps with Huber function, the total CPU time = 2.7 + 2.2 = 4.9 sec.

the resulting edge-stopping function is

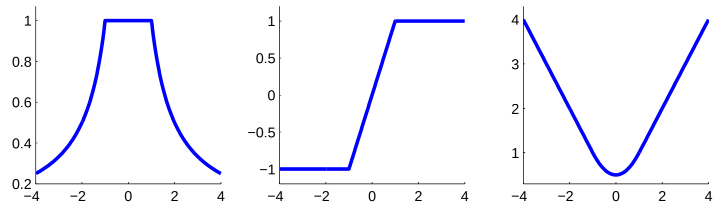
$$g(\sigma) = \frac{\phi(\sigma)}{\sigma} = \begin{cases} 0 & |\sigma| \geq \hat{\gamma}, \\ \frac{2}{\hat{\gamma}^2}(1 - (\frac{\sigma}{\hat{\gamma}})^2)^2 & |\sigma| < \hat{\gamma}. \end{cases}$$

To ensure that the Tukey and Huber functions start rejecting outliers at the same value we set

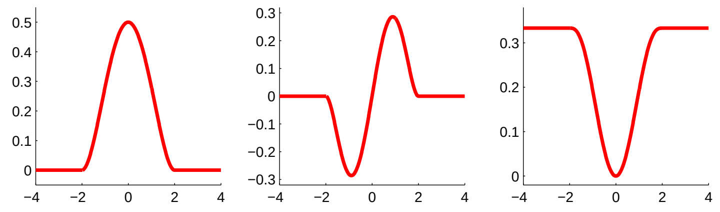
$$\hat{\gamma} = \sqrt{5}\gamma,$$

where γ for the Huber function is still defined by (7).

Thus, the influence of the Tukey function decreases all the way to zero. Comparing the two functions $\phi(\sigma) = \sigma g(\sigma)$ in the horizontal center of Fig. 2, the Huber function gives all outliers a constant weight of one whereas the Tukey function gives zero weight to outliers whose magnitude is above a certain value. From such shapes of ϕ we can correctly predict that smoothing with Tukey produces sharper boundaries than smoothing with Huber. We can also see how the choice of edge-stopping function acts to hold off excessive smoothing: given a piecewise constant image where all discontinuities are above a certain threshold, Tukey will leave the image unchanged whereas Huber will not. Results in Figs. 1(f) and 3(e)

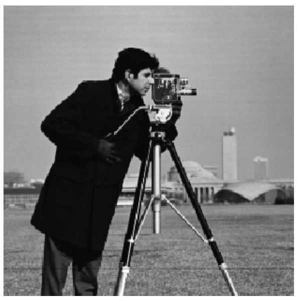


(a) Huber switching function

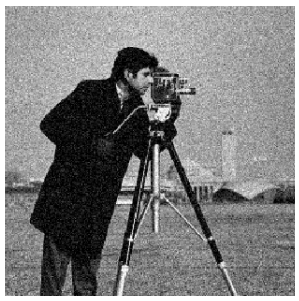


(b) Tukey function

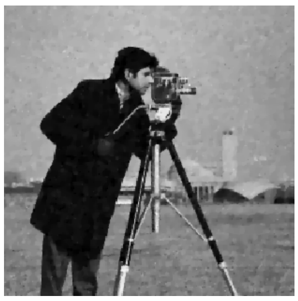
Fig. 2 Comparing the Huber switching function with the Tukey function: graphs of the edge-stopping function $g(\sigma)$ are in the left column; graphs of $\phi(\sigma) = \sigma g(\sigma)$ are in the middle column; graphs of $\rho(\sigma) = \int \sigma g(\sigma) d\sigma$ are in the right column.



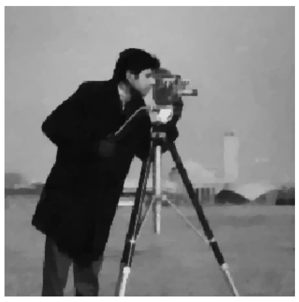
(a) True image



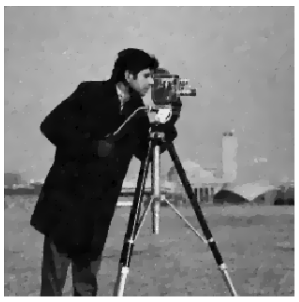
(b) 20% noise



(c) Misfit = 18.93



(d) Misfit = 22.67



(e) Misfit = 19.66

Fig. 3 Smoothing and sharpening the 256×256 `Cameraman` image: (a) true image; (b) image corrupted by 20% Guassian white noise serves as data; (c) pre-denoised using 21 SD steps for a rougher relative error tolerance of 10^{-4}; (d) then denoised by 3 IRLS iterations with $\beta = 0.15$ determined by (16), the total CPU time = 3.4 + 5.2 = 8.6 sec; (e) alternatively, sharpened using 10 SD steps with the Tukey function starting from the result of (c), the total CPU time = 3.4 + 2.2 = 5.6 sec.

confirm our predictions. After a quick pre-denoising with Huber, only 10 SD steps with Tukey result in obviously sharper discontinuities, i.e., image edges, than those without switching the regularization operator; see Figs. 1 and 3.

It is important to note that a rough pre-denoising is *necessary* here. It helps us avoid strengthening undesirable effects of heavy noise by using the Tukey function.

From an experimental point of view, we recommend to apply the explicit-implicit LSD scheme if a smooth image is desired; otherwise, employ the explicit Tukey regularization, starting from the result of the rough explicit Huber regularization, to get a sharper version.

3 Deblurring

Here we consider the deblurring problem discussed in [33,10,22]. The blurring of an image can be caused by many factors: (i) movement during the image capture process, by the camera or, when long exposure times are used, by the subject; (ii) out-of-focus optics, use of a wide-angle lens, atmospheric turbulence, or a short exposure time, which reduces the number of photons captured; (iii) scattered light distortion in confocal microscopy. Mathematically, in most cases blurring can be linearly modeled to be *shift-invariant* with a point spread function (PSF), denoted by $f(x,y)$. Further, it is well known in signal processing and systems theory [25,26] that a shift-invariant linear operator must be in the form of convolution, written as

$$F(m) = f(x,y) * m(x,y) = \int_{\mathbb{R}^2} f(x-x', y-y') m(x', y') dx' dy'. \tag{18}$$

So, an observed blurred image b is related to the ideal sharp image $m(x,y)$ by

$$b = f(x,y) * m(x,y) + \epsilon,$$

where $*$ denotes convolution product and the point spread function $f(x,y)$ may vary in space. Thus, the matrix-vector multiplication Jm represents a discrete model of the distortion operator (18) convolved by the PSF.

In the spatial domain, the PSF describes the degree to which an optical system blurs or spreads a point of light. The PSF is the inverse Fourier transform of the optical transfer function (OTF). In the frequency domain, the OTF describes the response of a linear, position-invariant system to an impulse. The distortion is created by convolving the PSF with the original true image, see [33,10]. Note that distortion caused by a PSF is just one type of data degradation, and the clear image m_{true} generally does not exist in reality. This image represents the result of perfect image acquisition conditions. Nonetheless, in our numerical experiments we have a "ground truth" model m_{true} which is used to synthesize data and judge the quality of reconstructions.

In the implementation we apply the same discretization as in Chapter 5 of [33], and then the convolution (18) is discretized into a matrix-vector multiplication, yielding a problem of the form (1), (2), where J is an $N \times N$ symmetric, doubly block Toeplitz matrix. Such a blurring matrix J can be constructed by a Kronecker product. However, J is now a full, large matrix, and avoiding its explicit construction and storage is therefore desirable. Using a gradient descent algorithm we actually only need two matrix-vector products to form $J^T(Jm)$, and there is no reason to construct or store the matrix J itself. Moreover, it is possible to use a 2D fast Fourier transform (FFT) algorithm to reduce the computational cost of the relevant matrix-vector multiplication from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \log N)$. Specifically, after discretizing the integral operator (18) in the form of convolution, we have a fully discrete model

$$b_{i,j} = \sum_{p=0}^{n} \sum_{q=0}^{n} f_{i-p,j-q} m_{p,q} + \epsilon_{i,j}, \quad 0 \le i,\ j \le n,$$

where $\epsilon_{i,j}$ denotes random noise at the grid location (i,j). In general, the discrete PSF $\{f_{i,j}\}_{i,j=0}^{n}$ is 2D-periodic, defined as

$$\begin{aligned} f_{i,j} &= f(i',j), \quad \text{whenever} \quad i = i' \ \mathtt{mod}\ n+1, \\ f_{i,j} &= f(i,j'), \quad \text{whenever} \quad j = j' \ \mathtt{mod}\ n+1. \end{aligned}$$

This suggests we only need consider $f \in \mathbb{R}^{(n+1)\times(n+1)}$, because by periodic extension we can easily get $(n+1, n+1)$-periodic arrays f^{ext}, for which

$$f^{ext}_{i,j} = f_{i,j}, \quad \text{whenever} \quad 0 \le i,\ j \le n.$$

Then the FFT algorithm yields

$$f^{ext} * m = (n+1)\, \mathcal{F}^{-1}\{\mathcal{F}(f) \,.*\, \mathcal{F}(m)\},$$

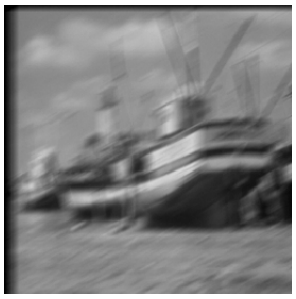
where $\mathcal{F}$ is the discrete Fourier transform and $.*$ denotes component-wise multiplication.

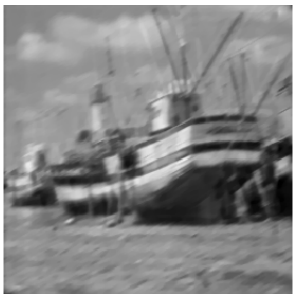
Thus, we consider the gradient descent algorithm (9), starting from the data $m^0 = b$, and compare the step size choices (11a) vs. (11b). As before, strictly speaking, these would be steepest descent and lagged steepest descent only if L defined in (6) were constant, i.e., using least-squares regularization. But we proceed to freeze L for this purpose anyway, which amounts to a lagged diffusivity approach.

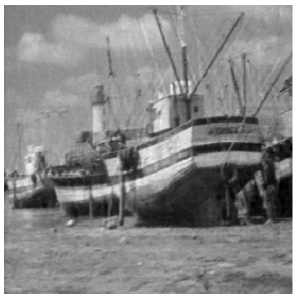
One difference from the denoising problem is that here we do not have an easy tool for determining the regularization parameter β, and it is determined experimentally instead. However, for the problems discussed below this turns out not to be a daunting task.

To illustrate and analyze our deblurring algorithm, we generate some degraded data at first. We use the MATLAB function FSPECIAL to create a variety of correlation kernels, i.e., PSFs, and then deliberately blur clear images by convolving them with these different PSFs. The function FSPECIAL(TYPE,PARAMETERS) accepts a filter type plus additional modifying parameters particular to the type of filter chosen. Thus, FSPECIAL('MOTION',LEN,THETA) returns a filter to approximate, once convolved with an image, the linear motion of a camera by LEN pixels with an angle of THETA degrees in a counterclockwise direction, which therefore becomes a vector for horizontal and vertical motions (see Fig. 4); FSPECIAL('LOG',HSIZE,SIGMA) returns a rotationally symmetric Laplacian of Gaussian filter of size HSIZE with standard deviation SIGMA (see Fig. 6); FSPECIAL('DISK',RADIUS) returns a circular mean filter within the square matrix of side 2 RADIUS+1 (see Fig. 7); FSPECIAL('UNSHARP',ALPHA) returns a 3×3 unsharp contrast enhancement filter, which

(a) True image


(b) MOTION blur


(c) $\beta = 10^{-3}$, 23 LSD iters


(d) $\beta = 10^{-4}$, 33 LSD iters

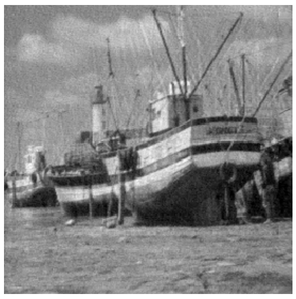
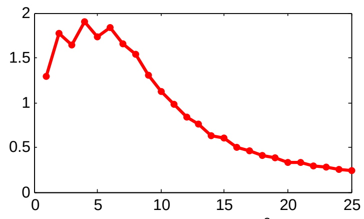
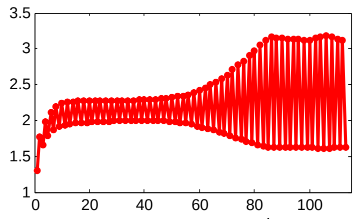
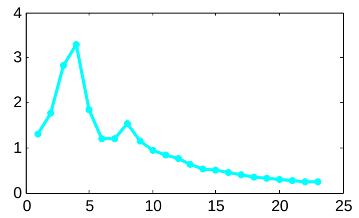

(e) $\beta = 10^{-5}$, 38 LSD iters

Fig. 4 TYPE = 'MOTION', LEN = 15, THETA = 30, $\eta = 1$.


(a) SD − τ $\beta = 10^{-3}$


(b) SD − τ $\beta = 10^{-4}$


(c) LSD − τ $\beta = 10^{-3}$

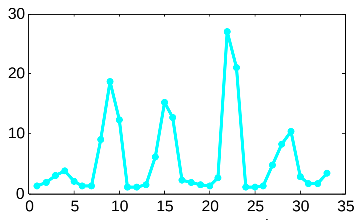
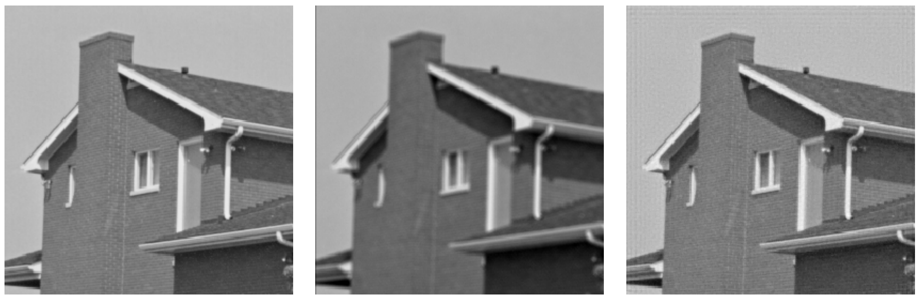
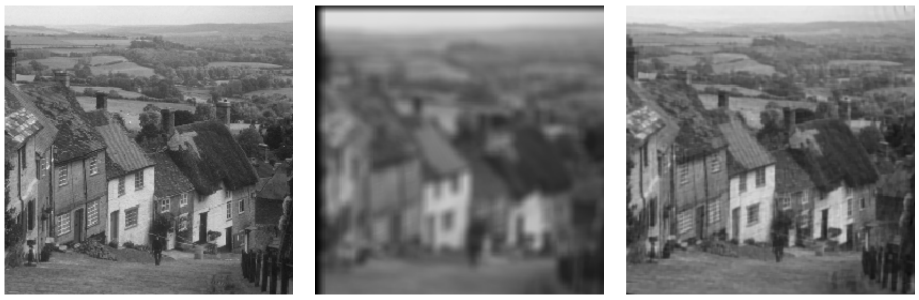

(d) LSD − τ $\beta = 10^{-4}$

Fig. 5 SD step size (11a) vs. LSD step size (11b) for the deblurring example in Fig. 4 under the relative error tolerance of 10^{-4}: (a) and (c) compare SD with LSD for $\beta = 10^{-3}$, 25 SD iters vs. 23 LSD iters; (b) and (d) compare them for $\beta = 10^{-4}$, 113 SD iters vs. 33 LSD iters. Note the horizontal scale (number of iterations) difference between (b) and (d).



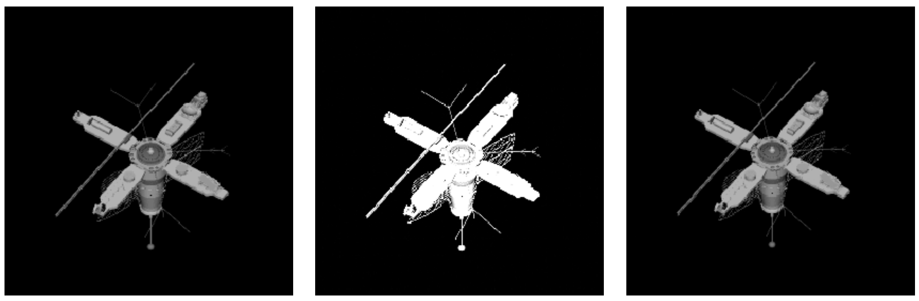
(a) True image (b) LOG blur (c) 114 SD or 31 LSD iters

Fig. 6 TYPE = 'LOG', HSIZE = [512, 512], SIGMA = 0.5, $\eta = 1$, $\beta = 10^{-4}$.



(a) True image (b) DISK blur (c) 99 SD or 26 LSD iters

Fig. 7 TYPE = 'DISK', RADIUS = 5, $\eta = 1$, $\beta = 10^{-4}$.



(a) True image (b) UNSHARP blur (c) 17 SD or 15 LSD iters

Fig. 8 TYPE = 'UNSHARP', ALPHA = 0.2, $\eta = 1$, $\beta = 10^{-4}$.

enhances edges and other high frequency components by subtracting a smoothed unsharp version of an image from the original image, and the shape of which is controlled by the parameter ALPHA (see Fig. 8); FSPECIAL('GAUSSIAN',HSIZE,SIGMA) returns a rotationally symmetric Gaussian low-pass filter of size HSIZE with standard deviation SIGMA (see Fig. 9); FSPECIAL('LAPLACIAN',ALPHA) returns a 3×3 filter approximating the shape of the two-dimensional Laplacian operator and the parameter ALPHA controls the shape of the Laplacian (see Fig. 10).

All six images presented and used in this section are 256×256. For the first four experiments, we only add a small amount of random noise into blurred images, say 1% ($\eta = 1$), and stop deblurring when the relative error norm (13) is below 10^{-4}. For a given PSF, the only parameter required is the regularization parameter β. From Fig. 4 we can clearly see that the smaller β is, the sharper the restored solution is, including both image and noise. So the `Boat` image reconstructed with $\beta = 10^{-3}$ in Fig. 4(c) still looks blurry and that cannot be improved by running more iterations. The `Boat` image deblurred with $\beta = 10^{-5}$ in Fig. 4(e) becomes much clearer; however, unfortunately, such a small β also brings the undesirable effect of noise amplification. The setting $\beta = 10^{-4}$ seems to generate the best approximation of the original scene, and so it does in the following three deblurring experiments.

As in the case of denoising, the LSD step size selection (11b) usually yields faster convergence than SD. When β is small, e.g., 10^{-4} or less, the step size (11a) is very close to the strict steepest descent selection obtained for a constant L, and so the famous two-periodic cycle of [1] (see also [2]) appears in the step sequence, resulting in a rather slow convergence; see Fig. 5(b). The lagged step size (11b) breaks this cycling pattern (see Fig. 5(d)), providing a much faster convergence for the same error tolerance. Moreover, with the same parameter β, the reconstructed images using both SD and LSD are quite comparable, and it is difficult to tell any differences between them by the naked eye. For $\beta = 10^{-5}$, 335 SD steps are required to reach a result comparable to Fig. 4(e) and the corresponding CPU time is 441.6 sec. Using LSD we only need 48.7 sec, reflecting the fact that the CPU time is roughly proportional to the number of steps required.

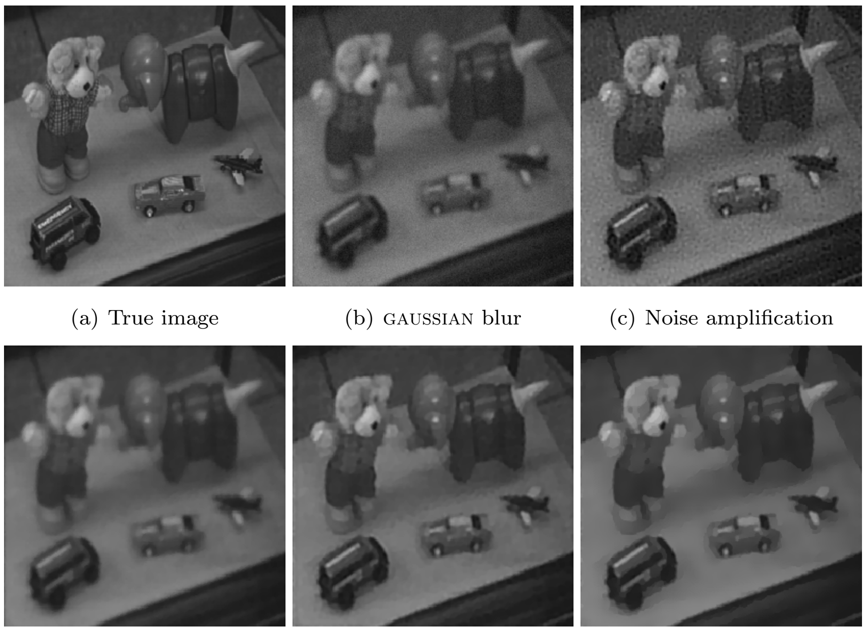
Figs. 6 – 8 reinforce our previous observations that the gradient descent deblurring algorithm with LSD step selection (11b) works very well, and that the improvement of LSD over SD is even more significant here than in the case of denoising. This is especially pronounced when a small value of β must be chosen and when accuracy considerations require more than 20 or so steepest descent steps.

In [2] we have discussed other faster gradient descent methods. The half-lagged steepest descent (HLSD) method [29, 19] was generally found there to consistently be at par with LSD, while other variants performed somewhat worse. In the present article we have applied HLSD in place of LSD for the above four examples, where the step size selection counts most. With HLSD the formula (11a) is applied only at each even-numbered step and then the same step size gets reused in the following odd-numbered one. The results were found to be again comparable to those using LSD, both in terms of efficiency and in terms of quality.

We have also experimented with a version of CG where we locally pretend, as in (11), that the minimization problem is quadratic. However, the CG method is well-known to be more sensitive than gradient descent to violations of its premises, and its iteration counts when applied to each of the examples in Figs. 4 – 7 were consistently about 20% higher than the better of LSD and HLSD.

Deblurring noisier images

The operation of deblurring essentially sharpens the image, whereas denoising essentially smooths it. Thus, trouble awaits any algorithm when both significant blur and significant noise are present in the given data set.



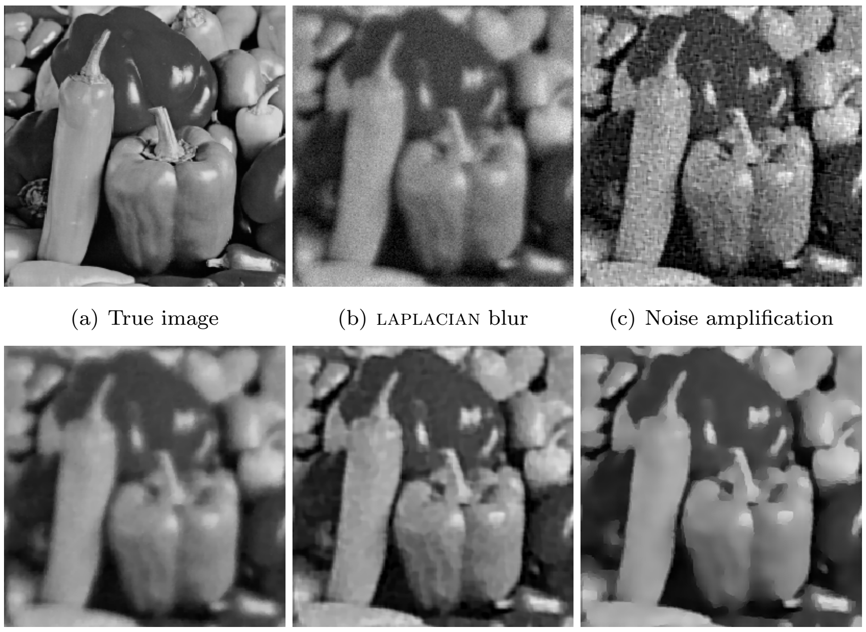
(a) True image (b) GAUSSIAN blur (c) Noise amplification

(d) Pre-denoised (e) Deblurred (f) Sharpened

Fig. 9 TYPE = 'GAUSSIAN', HSIZE = [512, 512], SIGMA = 1.5: (b) data b corresponding to $\eta = 5$; (c) directly deblurring using $\beta = 5 \times 10^{-4}$ fails; (d) pre-denoising: for the relative error tolerance of 10^{-4}, 9 LSD steps of (12) are required; (e) 22 steps of (9) with $\beta = 5 \times 10^{-4}$ and LSD (11b) are required for the next deblurring; (f) 10 sharpening steps with the Tukey function (17) follow. The total CPU time = 16.1 sec.

In our present setting, if we add more noise to the blurred images when synthesizing the data, say $\eta \geq 5$, then directly running the deblurring gradient descent algorithm as above may fail due to the more severe effect of noise amplification; see, e.g., Figs. 9(c) and 10(c). After a few iterations, the restored image can have a speckled appearance, especially for a smooth object observed at low signal-to-noise ratios. These speckles do not represent any real texture in the image, but are artifacts of fitting the noise in the given image too closely. Noise amplification can be reduced by increasing the value of β, but this may result in a still blurry image, as in Fig. 4(c). Since the effects of noise and blur are opposite, a quick and to some extent effective remedy is *splitting*, described next.

At first, we only employ denoising up to a coarser tolerance, e.g., 10^{-4}. This can be carried out in just a few steps of (12) with either the SD or LSD step size selection. Starting with the lightly denoised image, as in Figs. 9(d) and 10(d), we next apply the deblurring algorithm (9) with the LSD step sizes to correct PSF distortion. Since now the noise level becomes higher, we slightly increase the value of the regularization parameter and apply $\beta = 5 \times 10^{-4}$ for the `Toy` example and $\beta = 10^{-3}$ for the `Pepper` example. Observe that, even though some speckles still appear on the image cartoon components, the results presented in Figs. 9(e) and 10(e) are much more acceptable than those in Figs. 9(c) and 10(c), which were deblurred by the same number of iterations without pre-denoising. Finally,



(a) True image (b) LAPLACIAN blur (c) Noise amplification

(d) Pre-denoised (e) Deblurred (f) Sharpened

Fig. 10 TYPE = 'LAPLACIAN', ALPHA = 0.2: (b) data b corresponding to $\eta = 10$; (c) directly deblurring using $\beta = 10^{-3}$ fails; (d) pre-denoising: for the relative error tolerance of 10^{-4}, 10 LSD steps of (12) are required; (e) 25 steps of (9) with $\beta = 10^{-3}$ and LSD (11b) are required for the next deblurring; (f) 10 sharpening steps with the Tukey function (17) follow. The total CPU time = 14.5 sec.

we can use the Tukey regularization in (12) to further improve the reconstruction, carefully yet rapidly removing unsuitable speckles and enhancing the contrast, i.e., *sharpening*. The results are demonstrated in Figs. 9(f) and 10(f). The total CPU time, given in the captions of Figs. 9 and 10, clearly shows the efficiency of this hybrid deblurring-denoising scheme.

4 Conclusions

In this paper we have examined the effect of replacing steepest descent (SD) by a faster gradient descent algorithm, specifically, lagged steepest descent (LSD), in the practical context of image deblurring and denoising tasks. We have also proposed several highly efficient schemes for carrying out these tasks, independently of the step size selection.

Our general conclusion is that in situations where many (say, over 20) steepest descent steps are required, thus building slowness into the solution procedure, the faster gradient descent method offers substantial advantages.

Specifically, four scenarios have been considered. The first is a straightforward denoising process using anisotropic diffusion [2]. Here the LSD step selection offers an efficiency improvement by a factor of roughly 3.

In contrast, the second denoising scenario does not allow slowness buildup by SD because after a quick rough denoising we switch to an implicit method, with a good estimate for β at hand, or to a sharpening phase using (17). The resulting method is new and effective, although not because of a dose of LSD.

Switching to the more interesting and challenging deblurring problem, the third scenario envisions the presence of little additional noise, so we directly employ the gradient descent method (9) with J as described in Section 3, R given by (5) and (7), and $\beta = 10^{-4}$. This allows for slowness buildup when using SD step sizes, and the faster LSD variant then excels, becoming up to 10 times more efficient. The HLSD variant is overall as effective as LSD.

Finally, in the presence of significant noise effective deblurring becomes a harder task. We propose a splitting approach whereby we switch between the previously developed denoising and deblurring algorithms. This again creates a situation where LSD does not contribute much improvement over SD. The splitting approach has been demonstrated to be relatively effective, although none of the reconstructions in Fig. 10, for instance, is amazingly good. The problem itself can become very hard to solve satisfactorily, unless some rather specific knowledge about the noise is available and can be used to carefully remove it before deblurring begins.

A future problem to be considered concerns the case where the PSF causing blurring is not known.

References

1. Akaike, H.: On a successive transformation of probability distribution and its application to the analysis of the optimum gradient method. Ann. Inst. Stat. Math. Tokyo **11**, 1–16 (1959)
2. Ascher, U., Doel, K.v.d., Huang, H., Svaiter, B.: Gradient descent and fast artificial time integration. M2AN **43**, 689–708 (2009)
3. Ascher, U., Haber, E., Huang, H.: On effective methods for implicit piecewise smooth surface recovery. SIAM J. Sci. Comput. **28**, 339–358 (2006)
4. Ascher, U., Huang, H., Doel, K.v.d.: Artificial time integration. BIT **47**, 3–25 (2007)
5. Barzilai, J., Borwein, J.: Two point step size gradient methods. IMA J. Num. Anal. **8**, 141–148 (1988)
6. Berg, E.v.d., Friedlander, M.: Probing the Pareto frontier for basis pursuit solutions. SIAM J. Scient. Comput. **31**, 890–912 (2008)
7. Black, M.J., Sapiro, G., Marimont, D.H., Heeger, D.: Robust anisotropic diffusion. IEEE trans. image processing **7(3)**, 421–432 (1998)
8. Calvetti, D., Reichel, L.: Lanczos-based exponential filtering for discrete ill-posed problems. Numer. Algorithms **29**, 45–65 (2002)
9. Chan, T., Mulet, P.: On the convergence of the lagged diffusivity fixed point method in total variation image restoration. SIAM J. Numer. Anal. **36**, 354–367 (1999)
10. Chan, T., Shen, J.: Image Processing and Analysis: Variational, PDE, Wavelet and Stochastic Methods. SIAM (2005)
11. Claerbout, J., Muir, F.: Robust modeling with erratic data. Geophysics **38**, 826–844 (1973)
12. Dai, Y., Fletcher, R.: Projected Barzilai-Borwein methods for large-scale box-constrained quadratic programming. Numerische. Math. **100**, 21–47 (2005)
13. Daubechies, I., de Friese, M., de Mol, C.: An iterative thresholding algorithm for linear inverse problems with sparsity constraints. Comm. Pure Appl. Math **57**, 1413–1457 (2004)
14. Doel, K.v.d., Ascher, U.: The chaotic nature of faster gradient descent methods. J. Scient. Comput. **DOI10.1007/s10915-011-9521-3** (2011)
15. Durand, F., Dorsey, J.: Fast bilateral filtering for the display of high-dynamic-range images. ACM Trans. Graphics (SIGGRAPH) **21(3)**, 257–266 (2002)
16. Engl, H.W., Hanke, M., Neubauer, A.: Regularization of Inverse Problems. Kluwer (1996)

17. Farquharson, C., Oldenburg, D.: Non-linear inversion using general measures of data misfit and model structure. Geophysics J. **134**, 213–227 (1998)
18. Figueiredo, M., Nowak, R., Wright, S.: Gradient projection for sparse reconstruction: application to compressed sensing and other inverse problems. IEEE J. Special Topics on Signal Processing **1**, 586–598 (2007)
19. Friedlander, A., Martinez, J., Molina, B., Raydan, M.: Gradient method with retard and generalizations. SIAM J. Num. Anal. **36**, 275–289 (1999)
20. Gilyazov, S., Gol'dman, N.: Regularization of Ill-Posed Problems by Iterative Methods. Kluwer (2000)
21. Hansen, P.C.: Rank Deficient and Ill-Posed Problems. SIAM, Philadelphia (1998)
22. Hardy, J.W.: Adaptive Optics for Astronomical Telescopes. Oxford University Press, NY (1998)
23. Mallat, S.: A Wavelet Tour of Signal Processing: the Sparse Way. Academic Press (2009). 3rd Ed.
24. Nocedal, J., Sartenar, A., Zhu, C.: On the behavior of the gradient norm in the steepest descent method. Comput. Optim. Appl. **22**, 5–35 (2002)
25. Oppenheim, A.V., Schafer, R.W.: Discrete-Time Signal Precessing. Prentice-Hall, Engewood Cliffs, NJ (1989)
26. Oppenheim, A.V., Willsky, A.S.: Signals and Systems. Prentice-Hall, Engewood Cliffs, NJ (1996)
27. Osher, S., Fedkiw, R.: Level Set Methods and Dynamic Implicit Surfaces. Springer (2003)
28. Perona, P., Malik, J.: Scale-space and edge detection using anisotropic diffusion. IEEE Transactions on Pattern Analysis and Machine Intelligence **12**(7), 629–639 (1990)
29. Raydan, M., Svaiter, B.: Relaxed steepest descent and Cauchy-Barzilai-Borwein method. Comput. Optim. Appl. **21**, 155–167 (2002)
30. Rudin, L., Osher, S., Fatemi, E.: Nonlinear total variation based noise removal algorithms. Physica D **60**, 259–268 (1992)
31. Sapiro, G.: Geometric Partial Differential Equations and Image Analysis. Cambridge (2001)
32. Tikhonov, A.N., Arsenin, V.Y.: Methods for Solving Ill-posed Problems. John Wiley and Sons, Inc. (1977)
33. Vogel, C.: Computational methods for inverse problem. SIAM, Philadelphia (2002)