Algorithms and software for total variation image reconstruction via first-order methods

Dahl, Joachim; Hansen, Per Christian; Jensen, Søren Holdt; Jensen, Tobias Lindstrøm

Published in:
Numerical Algorithms

DOI (link to publication from Publisher):
10.1007/s11075-009-9310-3

Publication date:
2010

Document Version
Publisher's PDF, also known as Version of record

Numer Algor (2010) 53:67–92
DOI 10.1007/s11075-009-9310-3

ORIGINAL PAPER

Algorithms and software for total variation image reconstruction via first-order methods

Joachim Dahl · Per Christian Hansen · Søren Holdt Jensen · Tobias Lindstrøm Jensen

Received: 14 October 2008 / Accepted: 19 June 2009 /
Published online: 8 July 2009

Abstract This paper describes new algorithms and related software for total variation (TV) image reconstruction, more specifically: denoising, inpainting, and deblurring. The algorithms are based on one of Nesterov's first-order methods, tailored to the image processing applications in such a way that, except for the mandatory regularization parameter, the user needs not specify any parameters in the algorithms. The software is written in C with interface to Matlab (version 7.5 or later), and we demonstrate its performance and use with examples.

This work is part of the project CSI: Computational Science in Imaging, supported by grant no. 274-07-0065 from the Danish Research Council for Technology and Production Sciences. J. Dahl's work was carried out at Aalborg University.

J. Dahl
AnyBody Technology A/S, Niels Jernes Vej 10,
9220 Aalborg Ø, Denmark
e-mail: dahl.joachim@gmail.com

P. C. Hansen (✉)
Department of Informatics and Mathematical Modelling,
Technical University of Denmark, Building 321,
2800 Lyngby, Denmark
e-mail: pch@imm.dtu.dk

S. H. Jensen · T. L. Jensen
Department of Electronic Systems, Aalborg University,
Niels Jernesvej 12, 9220 Aalborg Ø, Denmark

S. H. Jensen
e-mail: shj@es.aau.dk

T. L. Jensen
e-mail: tlj@es.aau.dk

Keywords Total variation · Denoising · Inpainting · Deblurring · First-order methods · Matlab

Mathematics Subject Classifications (2000) 65K19 · 65R32

1 Introduction

Image reconstruction techniques have become important tools in computer vision systems and many other applications that require sharp images obtained from noisy and otherwise corrupted ones. At the same time the total variation (TV) formulation has proven to provide a good mathematical basis for several basic operations in image reconstruction [5], such as *denoising*, *inpainting*, and *deblurring*. The time is ripe to provide robust and easy-to-use public-domain software for these operations, and this paper describes such algorithms along with related Matlab and C software. To our knowledge, this is the first public-domain software that includes all three TV image reconstruction problems. The software is available from http://www.netlib.org/numeralgo in the file na28, the Matlab files have been tested on Matlab versions 7.5–7.8, and they require version 7.5 or later.

We note that some Matlab codes are already available in the public domain, see the overview in Table 1. In Section 7 we compare the performance of our algorithms with those in Table 1; such a comparison is not straightforward as these codes solve slightly different problems and do not use comparable stopping criteria. Our comparisons show that our algorithms indeed scale well for large-scale problems compared to the existing methods.

The optimization problems underlying the TV formulation of image restoration cannot easily be solved using standard optimization packages due to the large dimensions of the image problems and the non-smoothness of the objective function. Many customized algorithms have been suggested in the literature, such as subgradient methods [1, 7], dual formulations [4, 24], primal-dual methods [6, 16, 21], graph optimization [9], second-order cone programming [13], etc. However, the implementation of all these methods for large-scale problem is not straightforward.

Our algorithms are based on recently published first-order methods developed by Nesterov [17–20], but tailored specifically to the problems in image restoration that we consider. The new first-order methods have $O(1/\epsilon)$ complexity, where ϵ is the accuracy of the solution. These methods show promising potential in large-scale optimization but have, so far, been used only scarcely for image processing algorithms—except for very recent work in [2] and [22].

Compared to [22], we provide practical complexity bounds and stopping criteria, we included inpainting into Nesterov's framework, and we use rank reduction to improve the speed and numerical stability of the deblurring algorithm. Our approach allows us to choose all necessary parameters in the algorithms in a suitable fashion, such that only the regularization parameter

Table 1 Freely available Matlab codes for TV reconstruction

Code:	`tvdenoise`—denoising.
Author:	Pascal Getreuer, Dept. of Mathematics, UCLA, Los Angeles.
Comments:	Chambolle's algorithm [4] (dual formulation), stopping criterion, very fast, also treats color images.
Availability:	Matlab Central File Exchange: www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=16236
Code:	`perform_tv_denoising`—denoising.
Author:	Gabriel Peyré, CNRS, CEREMADE, Université Paris Dauphine.
Comments:	Chambolle's algorithm [4] (dual formulation), no stopping criterion, fast.
Availability:	Toolox—A Toolbox for General Purpose Image Processing: www.ceremade.dauphine.fr/~peyre/matlab/image/content.html
Code:	`TVGP`—denoising.
Authors:	M. Zhu, Dept. of Mathematics, UCLA, Los Angeles. S. Wright, Dept. of Computer Sciences, Univ. of Wisconsin, Madison. T. F. Chan, Dept. of Mathematics, UCLA, Los Angeles.
Comments:	Gradient projection algorithm for the dual formulation, software, stopping criterion, very fast. Described in [24].
Availability:	TV-Regularized Image Denoising Software: www.cs.wisc.edu/~swright/TVdenoising
Code:	`SplitBregmanROF`—denoising.
Authors:	Tom Goldstein and Stanley Osher, Dept. of Mathematics, UCLA, Los Angeles.
Comments:	Bregman iterations, C++ code with Matlab mex interface, stopping criterion, very fast. Described in [14].
Availability:	Split Bregman Denoising: www.math.ucla.edu/~tagoldst/code.html
Code:	`tv_dode_2D`—inpainting.
Author:	Carola-Bibiane Schönlieb, Centre for Mathematical Sciences, Cambridge University, UK.
Comments:	Script with built-in stopping criterion, no interface, slow. Described in [11].
Availability:	Domain Decomposition for Total Variation Minimization: homepage.univie.ac.at/carola.schoenlieb/webpage_tvdode/tv_dode_numerics.htm
Code:	`Fixed_pt` and `Primal_dual`—deblurring.
Author:	Curtis R. Vogel, Dept. of Mathematical Sciences, Montana State University, Bozeman.
Comments:	Scripts with no stopping criteria or interface. Described in [21].
Availability:	Codes for the book *Computational Methods for Inverse Problems:* www.math.montana.edu/~vogel/Book/Codes/Ch8/2d
Code:	`FTVdG`—deblurring.
Authors:	Junfeng Yang, Nanjing University, China. Yin Zhang, Wotao Yin, and Yilun Wang, Dept. of Computational and Applied Mathematics, Rice University, Houston.
Comments:	Script with stopping criteria, fast, treats color images. Described in [23].
Availability:	FTVd: A Fast Algorithm for Total Variation based Deconvolution. www.caam.rice.edu/~optimization/L1/ftvd/v3.0/

must be specified by the user. More experienced users can set additional parameters if needed. Our algorithms and implementations are robust, user friendly, and suited for large problems.

Our paper starts with a brief summary of the notation in Section 2. We then present our three methods for TV-based denoising, inpainting, and deblurring in Sections 3–5; the presentation follows that of Nesterov, but with a simplified notation tailored to our image processing applications. Next, in Section 6 we illustrate the use of our methods and software with three examples, and in Section 7 we demonstrate the performance and the computational complexity of our methods. Brief manual pages for the Matlab functions are given in Appendix A.

2 Notation

In this package we consider $m \times n$ grayscale images, represented by the image arrays B (the noisy/corrupted image) and X (the reconstructed image). For our mathematical formulation it is convenient to represent the images by the two vectors x and b of length mn, given by

$$x = \mathrm{vec}(X), \qquad b = \mathrm{vec}(B),$$

where "vec" denotes column-wise stacking.

Associated with each pixel X_{ij} is a 2×1 gradient vector, and we approximate this gradient via finite differences. To set the notation, we first define two $m \times n$ arrays X'_{c} and X'_{r} with the finite-difference approximations to the partial derivatives in the directions of the columns and rows:

$$X'_{\mathrm{c}} = D_m\, X, \qquad X'_{\mathrm{r}} = X\, D_n^T,$$

where the two matrices D_m and D_n hold the discrete approximations to the derivative operators, including the chosen boundary conditions. Then we write the *gradient approximation* for pixel ij as the 2×1 vector

$$D_{(ij)}\, x = \begin{pmatrix} (X'_{\mathrm{c}})_{ij} \\ (X'_{\mathrm{r}})_{ij} \end{pmatrix} \in \mathbb{R}^{2\times 1}, \tag{1}$$

where the notation (ij) for the subscript denotes that we operate on the pixel with index ij, and $D_{(ij)}$ is a matrix of dimensions $2 \times mn$. For one-sided difference approximations at the "inner pixels", we have

$$D_{(ij)} = \left[e_{i+1+(j-1)m} - e_{i+(j-1)m}\ ,\ e_{i+jm} - e_{i+(j-1)m} \right]^T,$$

in which e_k denotes the kth canonical unit vector of length mn. We also define the matrix D (of dimensions $2mn \times mn$) obtained by stacking all the $D_{(ij)}$ matrices:

$$D = \begin{pmatrix} D_{(11)} \\ \vdots \\ D_{(mn)} \end{pmatrix}. \tag{2}$$

In Appendix B we show that the 2-norm of this matrix satisfies $\|D\|_2^2 \leq 8$. The approximation to the gradient norm satisfies $\|D_{(ij)}\, x\|_2^2 = (X'_{\mathrm{c}})_{ij}^2 + (X'_{\mathrm{r}})_{ij}^2$.

We also need to introduce the vector $u \in \mathbb{R}^{2mn}$ of *dual variables*, and similar to before we use the notation $u_{(ij)}$ for the 2-element sub-vector of u that conforms with (2) and corresponds to pixel ij.

The total variation (TV) of a function $f(s,t)$ in a domain Ω is defined as the 1-norm of the gradient magnitude, i.e., $\int_\Omega \|\nabla f\|_2 \, ds\, dt$ in which $\|\nabla f\|_2^2 = (\partial f/\partial s)^2 + (\partial f/\partial t)^2$. For our discrete problem, we define the analogous *discrete TV function* associated with the image X as

$$\mathcal{T}(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} \|D_{(ij)}\, x\|_2, \tag{3}$$

i.e., the sum of all the 2-norms of the gradient approximations.

In our algorithms we need to extract elements of a vector $x \in \mathbb{R}^N$ specified by an index-set $\mathcal{I} = \{i_1,\ i_2,\ \ldots,\ i_{|\mathcal{I}|}\}$ with indices i_k between 1 and N. Here, $|\mathcal{I}|$ denotes the number of elements in $\mathcal{I}$. If all the elements in $\mathcal{I}$ are distinct (i.e., $i_k \neq i_l$ when $k \neq l$), then the complementary set is $\mathcal{I}_c := \{1, \ldots, N\} \setminus \mathcal{I} = \{j_1,\ j_2,\ \ldots,\ j_{N-|\mathcal{I}|}\}$ again with indices j_k between 1 and N.

3 Denoising

Given a noisy image $B = X^{\text{exact}} + \text{noise}$, the discrete TV denoising problem amounts to minimizing $\mathcal{T}(x)$ subject to a constraint on the difference between the reconstruction x and the data b. This ensures that the reconstructed image is closely related to the noisy image, but "smoother" as measured by the TV function (3). The discrete TV denoising problem can thus be formulated as

$$\begin{array}{l} \text{minimize } \sum_{i=1}^{m} \sum_{j=1}^{n} \|D_{(ij)}\, x\|_2 \\ \text{subject to } \|x - b\|_2 \leq \delta, \end{array} \tag{4}$$

which is a *second-order cone programming problem* (SOCP) [3]. The dual problem is also a SOCP, given by

$$\begin{array}{l} \text{maximize } -\delta\, \|D^T u\|_2 + b^T D^T u \\ \text{subject to } \|u_{(ij)}\|_2 \leq 1, \quad i = 1, \ldots, m,\ \ j = 1, \ldots, n, \end{array} \tag{5}$$

where $u \in \mathbb{R}^{2mn}$ is the dual variable. The two problems have the same optimal value, because Slater's constraint qualification is satisfied, cf. [3]. The SOCP in (4) can, in principle, be solved using standard interior-point algorithms, but the large dimensions typically render such an approach intractable.

3.1 The first-order method

Instead of using interior point algorithms, we adapt a first-order algorithm developed by Nesterov [17, 18] (similar to the approaches in [2] and [22]). Nesterov's algorithm is an efficient scheme for minimization of saddle point problems over bounded convex sets. The basic idea of this algorithm is to make a *smooth* $O(\epsilon)$-approximation with Lipschitz continuous derivatives to the non-differentiable TV function, and then subsequently minimize this approximation using an optimal first-order method for minimization of convex functions with Lipschitz continuous derivatives.

To adapt the TV denoising problem to Nesterov's method, we follow [3, §5.4] and rewrite (4) as a saddle point problem of the form

$$\min_{x\in Q_\mathrm{p}} \max_{u\in Q_\mathrm{d}} u^T D\, x,$$

where we have defined the primal and dual feasible sets

$$\begin{aligned} Q_\mathrm{p} &= \{x \mid \|x - b\|_2 \le \delta\}, \\ Q_\mathrm{d} &= \left\{u \mid \|u_{(ij)}\|_2 \le 1,\ i = 1,\dots,m,\ \ j = 1,\dots,n\right\}. \end{aligned}$$

To each set Q_p and Q_d we associate a so-called *prox-function*, which we choose as, respectively,

$$f_\mathrm{p}(x) = \tfrac{1}{2}\|x - b\|_2^2 \qquad \text{and} \qquad f_\mathrm{d}(u) = \tfrac{1}{2}\|u\|_2^2.$$

These functions are bounded above as

$$\Delta_\mathrm{p} = \max_{x\in Q_\mathrm{p}} f_\mathrm{p}(x) = \tfrac{1}{2}\delta^2 \qquad \text{and} \qquad \Delta_\mathrm{d} = \max_{u\in Q_\mathrm{d}} f_\mathrm{d}(u) = \tfrac{1}{2}mn.$$

As a smooth approximation for $\mathcal{T}(x)$ we then use an additive modification of $\mathcal{T}(x)$ with the prox-function associated with Q_d:

$$\mathcal{T}_\mu(x) = \max_{u\in Q_\mathrm{d}} \left\{u^T D\, x - \mu\ f_\mathrm{d}(u)\right\}. \tag{6}$$

The approximation $\mathcal{T}_\mu(x)$ then bounds $\mathcal{T}(x)$ as $\mathcal{T}_\mu(x) \le \mathcal{T}(x) \le \mathcal{T}_\mu(x) + \mu\Delta_\mathrm{d}$, meaning that if we set $\mu = \epsilon/(2\Delta_\mathrm{d}) = \epsilon/(mn)$ then we have an $(\epsilon/2)$-approximation of $\mathcal{T}(x)$. Furthermore, following [18], it can be shown that $\mathcal{T}_\mu(x)$ has Lipschitz continuous derivatives with constant

$$\mathcal{L}_\mu = \mu^{-1}\|D\|_2^2 \le 8/\mu,$$

and its gradient is given by

$$\nabla\mathcal{T}_\mu(x) = D^T u,$$

where u is the solution to (6) for a given x.

Nesterov's optimal first-order method for minimizing the convex function $\mathcal{T}_\mu(x)$ with Lipschitz continuous derivatives is listed in Fig. 1. We terminate the algorithm when the duality gap satisfies

$$\sum_{i=1}^{m}\sum_{j=1}^{n} \|D_{(ij)}\, x\|_2 + \delta \|D^T u\|_2 - u^T D\, b \;< \epsilon.$$

When the iterations are stopped by this criterion, leading to the solution x^ϵ, then we are ensured that the found solution is close to the exact solution x^* in the sense that $\mathcal{T}(x^\epsilon) - \mathcal{T}(x^*) < \epsilon$. We remark that with our formulation of the problem it is difficult to relate the parameter ϵ to the error $\|x^\epsilon - x^*\|_2$ a priori (while this is possible in the dual formulation in [24] where the primal variable is a function of the dual variable).

By specifying the threshold ϵ for the duality gap, we can determine the parameter $\mu = \epsilon/(mn)$ used in the TV denoising algorithm to evaluate $\mathcal{T}_\mu(x)$ (6). Nesterov showed in [18] that at most

$$\mathcal{N} = \frac{4\|D\|_2}{\epsilon}\sqrt{\Delta_{\mathrm{p}}\,\Delta_{\mathrm{d}}} \tag{7}$$

iterations are required to reach an ϵ-optimal solution. For the discrete TV denoising algorithm we obtain the bound

$$\mathcal{N}_{\mathrm{denoise}} = \frac{2\|D\|_2}{\epsilon}\delta\sqrt{mn} \le \frac{4\sqrt{2mn}}{\epsilon}\,\delta. \tag{8}$$

We return to the choice of ϵ in Section 7.

3.2 Efficient implementation

The key to an efficient implementation of our algorithm is to evaluate $g^{[k]}$ in step 1) and solve the two subproblems 2) and 3) efficiently. This is ensured by our choice of prox-functions f_{p} and f_{d}. By a simple change of variables it

Fig. 1 Nesterov's first-order method for discrete TV denoising. We stop the iterations when the duality gap is less than ϵ

Given data b and a tolerance ϵ.

Set $x^{[0]} = b$ (a feasible starting point), $\mu = \frac{\epsilon}{2\Delta_{\mathrm{d}}}$, and $\mathcal{L}_\mu = \frac{\|D\|_2^2}{\mu}$.

For $k = 0, 1, 2, \ldots$

1) Evaluate $g^{[k]} = \nabla\mathcal{T}_\mu(x^{[k]})$.

2) Find $y^{[k]} = \arg\min_{x\in Q_{\mathrm{p}}}\left\{(x - x^{[k]})^T g^{[k]} + \frac{1}{2}\mathcal{L}_\mu\|x - x^{[k]}\|_2^2\right\}$.

3) Find $z^{[k]} = \arg\min_{x\in Q_{\mathrm{p}}}\left\{\mathcal{L}_\mu\, f_{\mathrm{p}}(x) + \sum_{i=0}^{k}\frac{i+1}{2}(x - x^{[i]})^T g^{[k]}\right\}$.

4) Update $x^{[k+1]} = \frac{2}{k+3}\, z^{[k]} + \frac{k+1}{k+3}\, y^{[k]}$.

turns out that all three quantities can be written as the solution to a simple quadratically constrained problem of the form

$$\begin{aligned} &\text{minimize } \tfrac{1}{2}\theta^T\theta - \theta^T c \\ &\text{subject to } \|\theta\|_2 \le \eta, \end{aligned}$$

whose solution is simply given by $\theta = c \,/\, \max\{1, \|c\|_2/\eta\}$. In step 1) we must evaluate $g^{[k]} = \nabla \mathcal{T}_\mu(x^{[k]})$ and it is easy to show that the gradient is given by $\nabla \mathcal{T}_\mu(x^{[k]}) = D^T u^{[k]}$, where $u^{[k]}$ is given by

$$u^{[k]} = \arg\max_{u \in Q_{\mathrm{d}}} u^T D\, x^{[k]} - \frac{\mu}{2}\|u\|_2^2.$$

The mn sub-vectors $u_{(ij)}^{[k]}$ of $u^{[k]}$ are thus given by

$$u_{(ij)}^{[k]} = D_{(ij)} x^{[k]} \,/\, \max\left\{\mu, \|D_{(ij)} x^{[k]}\|_2\right\}.$$

In step 2) it follows from a simple variable transformation that

$$y^{[k]} = \left(\mathcal{L}_\mu\left(x^{[k]} - b\right) - g^{[k]}\right) \,/\, \max\left\{\mathcal{L}_\mu,\ \left\|\mathcal{L}_\mu\left(x^{[k]} - b\right) - g^{[k]}\right\|_2 / \delta\right\} + b,$$

and in step 3) we similarly obtain

$$z^{[k]} = -w^{[k]} / \max\left\{\mathcal{L}_\mu,\ \left\|w^{[k]}\right\|_2 / \delta\right\} + b,$$

where we have introduced $w^{[k]} = \sum_{i=0}^{k} \frac{1}{2}(i+1)\, g^{[i]}$.

The computations in each of the steps 1) to 4) are done efficiently in $O(mn)$ operations. If needed, the algorithm is also very easily to parallelize; the subproblem 1) can be divided in several separate problems, and steps 2) and 3) can be executed in parallel. The memory requirements are also very modest, requiring only memory for storing the five mn-vectors $g^{[k]}$, $w^{[k]}$, $x^{[k]}$, $y^{[k]}$, $z^{[k]}$, plus a temporary mn-vector—which is equivalent to the storage for 6 images in total. By exploiting the structure of D, it is not necessary to store the vector $u^{[k]}$ but only $u_{(ij)}^{[k]}$.

4 Inpainting

In this section we extend the total-variation denoising algorithm to include *inpainting*, i.e., the process of filling in missing or damaged parts of a (possibly noisy) image, cf. [5]. The basic idea is still to compute a reconstruction that is "smooth" in the TV sense, and identical to the data in all the non-corrupted pixels (or close to these data if they are noisy).

Specifically, let $\mathcal{I}$ be the index set for x corresponding to the corrupted pixels in X. The complementary index set $\mathcal{I}_c$ is the set of non-corrupted pixels. The basic TV inpainting problem can then be formulated as

$$\begin{aligned} &\text{minimize } \textstyle\sum_{i=1}^{m}\sum_{j=1}^{n} \|D_{(ij)}\, x\|_2 \\ &\text{subject to } \|\,(x-b)_{\mathcal{I}_c}\|_2 \le \delta, \end{aligned}$$

with the dual problem

$$\begin{aligned} &\text{maximize } -\delta\|(D^T u)_{\mathcal{I}_c}\|_2 + b_{\mathcal{I}_c}^T (D^T u)_{\mathcal{I}_c} \\ &\text{subject to } \|u_{(ij)}\|_2 \le 1, \quad i = 1,\dots,m,\ j = 1,\dots,n \\ &\qquad\qquad (D^T u)_{\mathcal{I}} = 0. \end{aligned}$$

In this primal-dual formulation, the dual feasible set is not simple because of the equality constraint $(D^T u)_{\mathcal{I}} = 0$ and hence the subproblem in step 1) of Fig. 1 will be complicated. Instead we bound the primal feasible set by adding an artificial norm-constraint on the pixels in the inpainting region, leading to the revised formulation

$$\begin{aligned} &\text{minimize } \textstyle\sum_{i=1}^m \sum_{j=1}^n \|D_{(ij)}\, x\|_2 \\ &\text{subject to } \|(x-b)_{\mathcal{I}_c}\|_2 \le \delta \\ &\qquad\qquad \|(x-d)_{\mathcal{I}}\|_2 \le \gamma, \end{aligned} \tag{9}$$

for some suitable vector d and parameter $\gamma > 0$. The dual problem corresponding to (9) is then

$$\begin{aligned} &\text{maximize } -\delta\,\|(D^T u)_{\mathcal{I}_c}\|_2 + b_{\mathcal{I}_c}^T (D^T u)_{\mathcal{I}_c} - \gamma\,\|(D^T u)_{\mathcal{I}}\|_2 + d_{\mathcal{I}}^T (D^T u)_{\mathcal{I}} \\ &\text{subject to } \|u_{(ij)}\|_2 \le 1, \quad i = 1,\dots,m,\ j = 1,\dots,n, \end{aligned} \tag{10}$$

and now we have simple constraints (similar to the denoising problem).

It is important that d and γ in (9) are chosen such that $\|(x-d)_{\mathcal{I}}\|_2 < \gamma$ holds for the solution of the original problem. The pixel intensity in the inpainted region is always bounded by the intensity of the non-corrupted pixels, i.e., the vector of inpainted pixels satisfies

$$x_{\mathcal{I}}^* \in P = \left\{ z \mid \min_{i\in\mathcal{I}_c} b_i \le z_j \le \max_{i\in\mathcal{I}_c} b_i, \quad \forall j \in \mathcal{I} \right\}.$$

If we then set the elements of the vector d to

$$d_j = \tfrac{1}{2}\left(\max_{i\in\mathcal{I}_c} b_i + \min_{i\in\mathcal{I}_c} b_i \right) \quad \forall\, j \in \mathcal{I},$$

i.e., d is the midpoint in the set P, then we have

$$\|(x^* - d)_{\mathcal{I}}\|_2 \le \max_{x_{\mathcal{I}}\in P} \|x_{\mathcal{I}} - d_{\mathcal{I}}\|_2 = \frac{1}{2}\left(\max_{i\in\mathcal{I}_c} b_i - \min_{i\in\mathcal{I}_c} b_i \right) \sqrt{|I|} := \gamma,$$

which we then select as our γ. These settings guarantee that we have an artificial norm-constraint that is inactive at the solution. The primal set is now $Q_{\mathrm{p}}' = \{x \mid \|(x-b)_{\mathcal{I}_c}\|_2 \le \delta,\ \|(x-d)_{\mathcal{I}}\|_2 \le \gamma\}$, and as the prox-function for this set we use

$$f_{\mathrm{p}}'(x) = \tfrac{1}{2}\left\|(x-b)_{\mathcal{I}_c}\right\|_2^2 + \tfrac{1}{2}\left\|(x-d)_{\mathcal{I}}\right\|_2^2 \tag{11}$$

with upper bound $\Delta_{\mathrm{p}}' = \frac{1}{2}(\gamma^2 + \delta^2)$. As prox-function for Q_{d} (which is unchanged) we again use $f_{\mathrm{d}}(u) = \frac{1}{2}\|u\|_2^2$ and μ is chosen similarly as in Section 3.

Regarding the implementation issues, only step 2) and step 3) in the algorithm from Fig. 1 change in the TV inpainting algorithm. Note that the

two cone constraints in (9) are non-overlapping and that the norms in the prox-function (11) are partitioned in the same way as the constraints. Hence, the two index sets of $y^{[k]}$ in step 2) can be computed separately, and they are given by

$$y^{[k]}_{\mathcal{I}_c} = \left(\mathcal{L}_\mu\left(x^{[k]} - b\right) - g^{[k]}\right)_{\mathcal{I}_c} / \max\left\{\mathcal{L}_\mu,\ \left\|\left(\mathcal{L}_\mu\left(x^{[k]} - b\right) - g^{[k]}\right)_{\mathcal{I}_c}\right\|_2 / \delta\right\} + b_{\mathcal{I}_c}$$

$$y^{[k]}_{\mathcal{I}} = \left(\mathcal{L}_\mu\left(x^{[k]} - d\right) - g^{[k]}\right)_{\mathcal{I}} / \max\left\{\mathcal{L}_\mu,\ \left\|\left(\mathcal{L}_\mu\left(x^{[k]} - d\right) - g^{[k]}\right)_{\mathcal{I}}\right\|_2 / \gamma\right\} + d_{\mathcal{I}}\,.$$

Similarly in step 3) we have

$$z^{[k]}_{\mathcal{I}_c} = -w^{[k]}_{\mathcal{I}_c} / \max\left\{\mathcal{L}_\mu,\ \left\|w^{[k]}_{\mathcal{I}_c}\right\|_2 / \delta\right\} + b_{\mathcal{I}_c}\,,$$

$$z^{[k]}_{\mathcal{I}} = -w^{[k]}_{\mathcal{I}} / \max\left\{\mathcal{L}_\mu,\ \left\|w^{[k]}_{\mathcal{I}}\right\|_2 / \gamma\right\} + d_{\mathcal{I}}\,.$$

The upper bound for the number of iterations in the discrete TV inpainting algorithm becomes

$$\mathcal{N}_{\text{inpaint}} = 2\|D\|_2\sqrt{(\gamma^2 + \delta^2)mn}\cdot\frac{1}{\epsilon} \le \frac{4\sqrt{2mn}}{\epsilon}\sqrt{\gamma^2 + \delta^2}. \tag{12}$$

Note that γ enters the bound in the same way as δ. However, while δ is typically small—of the same size as the errors in the data—the parameter γ is of the same size as the norm of the inpainted pixels $x_{\mathcal{I}}$. This illustrates the difficulty of the inpainting problem, in terms of computational complexity—compared to the denoising problem—when using Nesterov's method with our choices of prox-functions.

Similarly to Section 3, the complexity of each of the subproblem is $O(mn)$ with the same memory requirement.

5 Deblurring for reflexive boundary conditions

In addition to denoising and inpainting, it is natural to consider TV *deblurring* of images, where the blurring is modelled by a linear operator, i.e., the blurred image is given by

$$b = K\,x^{\text{exact}} + \text{noise},$$

in which $K \in \mathbb{R}^{mn\times mn}$ is a known matrix that represents the linear blurring in the image B [15]. TV deblurring then amounts to computing a reconstruction which is, once again, "smooth" in the TV sense and fits the noisy data b within a tolerance δ that acts as the regularization parameter. Hence the discrete TV deblurring problem can be formulated as

$$\begin{aligned}&\text{minimize } \textstyle\sum_{i=1}^{m}\sum_{j=1}^{n}\|D_{(ij)}\,x\|_2\\ &\text{subject to } \|Kx - b\|_2 \le \delta.\end{aligned}$$

Here we only consider spatially invariant blurring with a doubly symmetric point spread function and reflexive boundary conditions, for which the matrix K can be diagonalized by a two-dimensional discrete cosine transform (DCT) [15]. The algorithm is easily extended to other matrices K that can be diagonalized efficiently by an orthogonal or unitary similarity transform (e.g., the discrete Fourier transform for general point spread functions and periodic boundary conditions), or by singular value decomposition of smaller matrices, such as is the case for separable blur where K is a Kronecker product.

We thus assume that K can be diagonalized by an orthogonal similarity transform,

$$CKC^T = \Lambda = \mathrm{diag}(\lambda_i), \tag{13}$$

where the matrix C represents the two-dimensional DCT, and Λ is a real diagonal matrix with the eigenvalues of K. Then by a change of variables $\bar{x} = Cx$ and $\bar{b} = Cb$ we obtain the equivalent TV deblurring problem in the DCT basis

$$\begin{aligned} &\text{minimize } \textstyle\sum_{i=1}^{m}\sum_{j=1}^{n} \|D_{(ij)}C^T\bar{x}\|_2 \\ &\text{subject to } \|\Lambda\bar{x} - \bar{b}\|_2 \le \delta. \end{aligned}$$

We note that multiplications with C and C^T are implemented very efficiently by means of the DCT algorithm with complexity $mn\log(\max\{m,n\})$. In our software we use the C package FTTW [10, 12], and it is needed only for TV deblurring. FFTW is known as the fastest free software implementation of the Fast Fourier Transform algorithm. It can compute transforms of real- and complex-valued arrays (including the DCT) of arbitrary size and dimension, and it does this by supporting a variety of algorithms and choosing the one it estimates or measures to be preferable in the particular circumstance.

5.1 Rank reduction

Often Λ is singular—either exactly or within the accuracy of the finite-precision computations—in which case the feasible set $\{x \mid \|\Lambda\bar{x} - \bar{b}\|_2 \le \delta\}$ is unbounded, and as such the problem cannot be solved using Nesterov's method. Moreover, when the condition number $\mathrm{cond}(\Lambda) = \max_i|\lambda_i| / \min_i|\lambda_i|$ is large (or infinite), we experience numerical difficulties and slow convergence of the algorithm.

To overcome these difficulties we apply the well-known approach of *rank reduction* and divide the eigenvalues into two partitions: One set with sufficiently large values indexed by $\mathcal{I} = \{i \mid |\lambda_i| > \rho\|K\|_2\}$, and the complementary set indexed by $\mathcal{I}_c$. Here, $\|K\|_2 = \max_j|\lambda_j|$, and ρ is a parameter satisfying $0 < \rho < 1$. We also define the diagonal matrix Λ^ρ whose diagonal elements are given by

$$(\Lambda^\rho)_{ii} = \begin{cases} \lambda_i & \text{if } i \in \mathcal{I} \\ 0 & \text{else,} \end{cases}$$

and we note that Λ^ρ is the closest rank-$|\mathcal{I}|$ approximation to Λ. The default value of ρ in our software is $\rho = 10^{-3}$

We then solve a slightly modified deblurring problem obtained by replacing the matrix K with the implicitly defined rank-deficient approximation

$$K^\rho = C^T \Lambda^\rho C\,.$$

The corresponding rank-reduced TV deblurring problem is thus

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^m \sum_{j=1}^n \|D_{(ij)} C^T \bar{x}\|_2 \\ \text{subject to} & \|(\Lambda \bar{x} - \bar{b})_{\mathcal{I}}\|_2 \le \delta \\ & \|\bar{x}_{\mathcal{I}_c}\|_2 \le \gamma, \end{array} \tag{14}$$

where $\bar{x}_{\mathcal{I}_c}$ should be considered as unconstrained variables. The parameter γ must therefore be chosen sufficiently large such that the constraint $\|\bar{x}_{\mathcal{I}_c}\|_2 \le \gamma$ is inactive at the solution. The extra constraint is added for the same reason as in the inpainting problem, namely, to keep the dual feasible set simple.

In addition to improving the numerical stability and reducing the number of iterations, rank-reduced deblurring can also be seen as another way of imposing regularization on the ill-posed problem by reducing the condition number for the problem from $\mathrm{cond}(\Lambda)$ to $\mathrm{cond}(\Lambda^\rho) \le 1/\rho$.

Choosing γ to guarantee that the γ-bound is inactive is difficult without making γ too large and thereby increasing the number of iterations. We assume without loss of generality that we can scale K such that $\|x^{\text{exact}}\|_2 \approx \|b\|_2$. This means that a solution which is properly regularized will also have $\|\bar{x}\|_2 = \|x\|_2 \approx \|\bar{b}\|_2 \approx \|b\|_2$. Our software therefore scales K and selects

$$\gamma = \sqrt{mn}\|b\|_\infty,$$

which guarantees that γ is sufficiently large. If the artificial γ-bound in (14) is active at the solution, then this is a sign that the problem might not be sufficiently regularized due to a too large value of δ.

We remark that the first inequality constraint in problem (14) is infeasible unless $\|(\Lambda \bar{x})_{\mathcal{I}} - \bar{b}_{\mathcal{I}}\|_2^2 + \|\bar{b}_{\mathcal{I}_c}\|_2^2 \le \delta^2$, i.e., δ must always be large enough to ensure that $\|\bar{b}_{\mathcal{I}_c}\|_2 \le \delta$, which is checked by our software. This is no practical difficulty, because δ must always be chosen to reflect the noise in the data. The requirement $\|\bar{b}_{\mathcal{I}_c}\|_2 \le \delta$ simply states that δ must be larger than the norm of the component of b in the null space of K^ρ, and according to the model (13) this component is dominated by the noise.

With the notation $\Lambda_{\mathcal{I}} = \mathrm{diag}(\lambda_i)_{i\in\mathcal{I}}$, the dual problem of (14) is

$$\begin{array}{ll} \text{maximize} & -\delta \left\| \Lambda_{\mathcal{I}}^{-1} \left(CD^T u\right)_{\mathcal{I}} \right\|_2 - \gamma \left\| \left(CD^T u\right)_{\mathcal{I}_c} \right\|_2 + \bar{b}_{\mathcal{I}}^T \Lambda_{\mathcal{I}}^{-1} \left(CD^T u\right)_{\mathcal{I}} \\ \text{subject to} & \|u_{(ij)}\|_2 \le 1, \quad i = 1, \ldots, m, \ j = 1, \ldots, n. \end{array} \tag{15}$$

As the prox-function for the primal set $Q_{\mathrm{p}}'' = \{\bar{x} \mid \|(\Lambda \bar{x} - \bar{b})_{\mathcal{I}}\|_2 \le \delta, \ \|\bar{x}_{\mathcal{I}_c}\|_2 \le \gamma\}$ we use

$$f_{\mathrm{p}}''(\bar{x}) = \tfrac{1}{2}\|\bar{x}\|_2^2.$$

The corresponding upper bound $\Delta_{\mathrm{p}}'' = \max_{\bar{x}\in Q_{\mathrm{p}}''} f_{\mathrm{p}}''(\bar{x})$ can be evaluated numerically as the solution to a trust-region subproblem discussed below. We can bound it as

$$\Delta_{\mathrm{p}}'' \le \tfrac{1}{2}\left(\|\Lambda_{\mathcal{I}}^{-1}\bar{b}_{\mathcal{I}}\|_2^2 + \gamma^2\right) \le \tfrac{1}{2}\left(\frac{\|b\|_2^2}{\rho^2\|K\|_2^2} + \gamma^2\right).$$

The upper bound for the number of iterations is

$$\mathcal{N}_{\mathrm{deblur}} = \sqrt{8}\|D\|_2\sqrt{\Delta_{\mathrm{p}}'' mn}\cdot\frac{1}{\epsilon} \le 4\sqrt{2mn}\left(\frac{\|b\|_2^2}{\rho\|K\|_2^2} + mn\|b\|_\infty^2\right)\cdot\frac{1}{\epsilon}. \tag{16}$$

5.2 Implementation

Compared to the TV denoising algorithm from Section 3 there are a few changes in the implementation. In step 1) the computation of $u_{(ij)}^{[k]}$ now takes the form

$$u_{(ij)}^{[k]} = D_{(ij)}C^T\bar{x}^{[k]} / \max\left\{\mu, \|D_{(ij)}C^T\bar{x}^{[k]}\|_2\right\},$$

which is computed in $mn\log(\max\{m,n\})$ complexity. For the computations in steps 2) and 3), first note that the two cones in Q_{p}'' are non-overlapping because $\mathcal{I}\cap\mathcal{I}_{\mathrm{c}} = \emptyset$, and the subproblems can therefore be treated as two separated cases as we had for the inpainting algorithm in Section 4. The minimizers $y_{\mathcal{I}}^{[k]}$ and $z_{\mathcal{I}}^{[k]}$ can be found (via simple changes of variables) as the solution to the well-studied *trust-region subproblem* [8], i.e., as the solution to a problem of the form

$$\begin{aligned} &\text{minimize } \tfrac{1}{2}\theta^T\theta - c^T\theta \\ &\text{subject to } \|L\theta - y\|_2 \le \eta \end{aligned} \tag{17}$$

where $L = \mathrm{diag}(\ell_i)$ is a diagonal matrix. We first check whether c satisfies the constraint, i.e., if $\|Lc - y\|_2 \le \eta$ then $\theta = c$. Otherwise, we find the global minimum of the (non-convex) problem, using Newton's method to compute the unique root $\lambda > -\min_i\{\ell_i\}$ of the so-called *secular equation* [8, §7.3.3]

$$q^T\left(L^{-2} + \lambda I\right)^{-2} q = \sum_{i=1}^{mn}\frac{q_i^2}{\left(\ell_i^{-2} + \lambda\right)^2} = \eta,$$

where I is the identity matrix and

$$q = L^{-1}c - L^{-2}y.$$

Once the root λ has been found, the solution to (17) is given by

$$\theta = L^{-1}\left(b + \left(L^{-2} + \lambda\right)^{-1} q\right).$$

As the staring value for λ in Newton's method, we can use the solution from the previous (outer) iteration in Nesterov's method. Our experience is that this limits the number of Newton iterations in the trust-region method to just a few

iterations each with complexity $O(mn)$, i.e., in practice the cost of computing the solution to steps 2) and 3) is still $O(mn)$.

The minimizers $y_{\mathcal{I}_c}^{[k]}$ and $z_{\mathcal{I}_c}^{[k]}$ are both computed as the solution to the quadratic constrained problems. For step 2) we obtain

$$y_{\mathcal{I}}^{[k]} = \theta \text{ in (17) with } c = x_{\mathcal{I}}^{[k]} - g_{\mathcal{I}}^{[k]} \mathcal{L}_\mu^{-1},\ L = \Lambda_{\mathcal{I}}, \text{ and } \eta = \delta,$$
$$y_{\mathcal{I}_c}^{[k]} = \left(\mathcal{L}_\mu\, x_{\mathcal{I}_c}^{[k]} - g_{\mathcal{I}_c}^{[k]}\right) / \max\left\{\mathcal{L}_\mu,\ \left\|\left(\mathcal{L}_\mu\, x_{\mathcal{I}_c}^{[k]} - g_{\mathcal{I}_c}^{[k]}\right)\right\|_2 / \gamma\right\},$$

and in step 3) we similarly have

$$z_{\mathcal{I}}^{[k]} = \theta \text{ in (17) with } c = -w_{\mathcal{I}}^{[k]} \mathcal{L}_\mu^{-1},\ L = \Lambda_{\mathcal{I}}, \text{ and } \eta = \delta,$$
$$z_{\mathcal{I}_c}^{[k]} = -w_{\mathcal{I}_c}^{[k]} / \max\left\{\mathcal{L}_\mu,\ \left\|w_{\mathcal{I}_c}^{[k]}\right\|_2 / \gamma\right\}.$$

The bound Δ_{p}'' on the primal set can be obtained a priori as

$$\Delta_{\mathrm{p}}'' = \tfrac{1}{2}\left(\|\theta\|_2^2 + \gamma^2\right),$$

where θ here is the solution to the problem

$$\begin{aligned} &\text{minimize } -\tfrac{1}{2}\theta^T \Lambda_{\mathcal{I}} \Lambda_{\mathcal{I}} \theta + b_{\mathcal{I}}^T \Lambda_{\mathcal{I}} \theta \\ &\text{subject to } \|\theta\|_2 \le \eta \end{aligned}$$

which can be solved using the same method as the previous trust region problem.

The complexity of step 1) in the TV deblurring algorithm increases, compared to the previous two algorithms, since we need to compute a two-dimensional DCT of the current iterate $x^{[k]}$ as well as an inverse two-dimensional DCT of $g^{[k]}$, i.e., the complexity per iteration of the algorithm is thus dominated by these $mn \log(\max\{m, n\})$ computations. The memory requirements of the algorithm is increased by the vectors holding q, q element-wise squared, and the diagonal elements of L^{-2} to avoid re-computation, plus and an extra temporary vector, leading to a total memory requirement of about $10mn$.

6 Numerical examples

In this section we give numerical examples that illustrate the three TV algorithms from the previous sections. All the algorithms are implemented in the C programming language, and the examples are run on a 2 GHz Intel Core 2 Duo computer with 2 GB of memory running the Linux operating system and using a single processor. We provide the three m-files `TVdenoise`, `TVinpaint`, and `TVdeblur` such that the C functions can be used from Matlab, and we also provide corresponding demo Matlab scripts that generate the examples in this section.

In the first example we consider the **TV denoising algorithm** from Section 3. The top images in Fig. 2 show the pure 512×512 image and the same image corrupted by additive white Gaussian noise with standard deviation $\sigma = 25$, leading to a signal-to-noise ratio $20 \log_{10}(\|X\|_F / \|X - B\|_F) = 15$ dB. For our TV reconstructions, we choose the parameter δ such that it reflects the noise level in the image [13],

$$\delta = \tau \sqrt{mn}\, \sigma \ , \tag{18}$$

where σ is the standard deviation of the noise, and τ is factor close to one. The two bottom images in Fig. 2 show TV reconstructions for $\tau = 0.85$ and 1.2; the

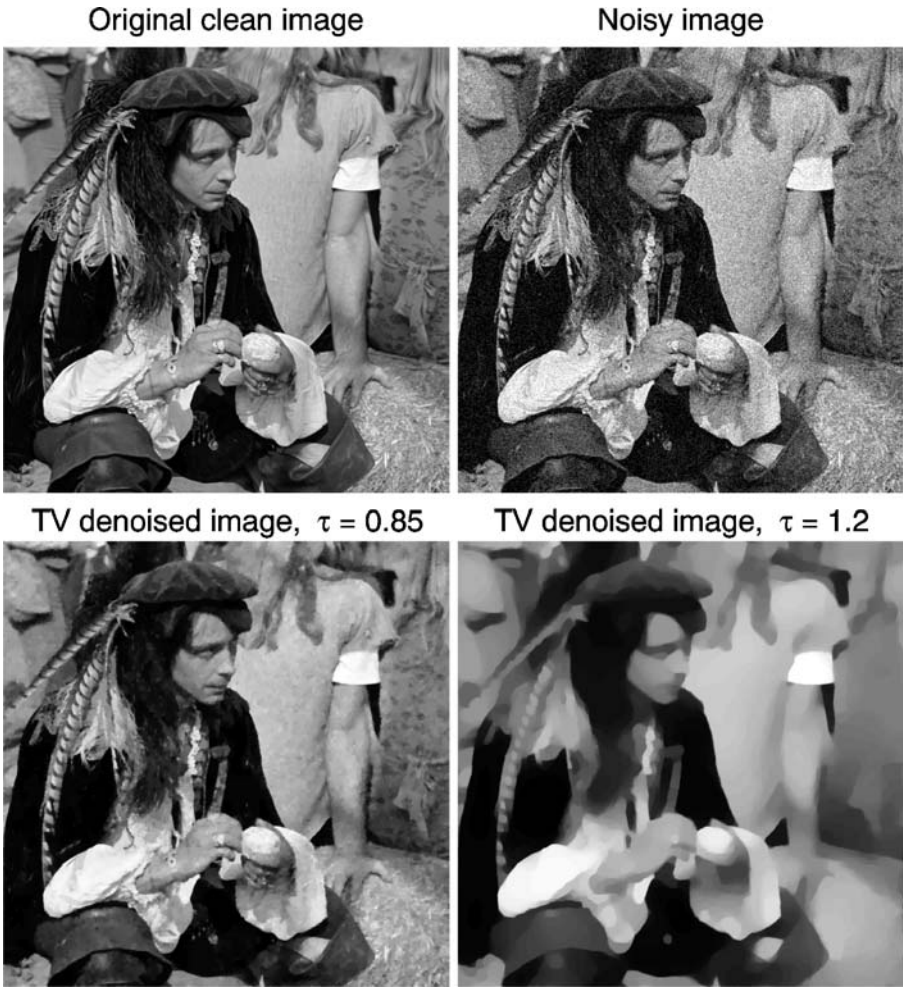

Fig. 2 Example of TV denoising. *Top*: clean and noisy images of size 512×512. *Bottom*: TV reconstructions for two different choices of the parameter τ in (18)

first choice leads to a good reconstruction, while the second choice is clearly too large, leading to a reconstruction that is too smooth in the TV sense (i.e., large domains with the same intensity, separated by sharp contours).

In the second example we illustrate the **TV inpainting algorithm** from Section 4, using the same clean image as above. Figure 3 shows the damaged image and the TV reconstruction. The white pixels in the corrupted image show the missing pixels, and we also added noise with standard deviation $\sigma = 15$ to the intact pixels. There is a total of $|\mathcal{I}| = 27,452$ damaged pixels, corresponding to about 10% of the total amount of pixels. In the reconstruction we used

$$\delta = \tau\sqrt{|\mathcal{I}_c|}\sigma, \tag{19}$$

which is a slight modification of (18) to reflect the presence of corrupted pixels. In the example we used $\tau = 0.85$.

The third example illustrates the **TV deblurring algorithm** from Section 5, again using the same clean image. Figure 4 shows the blurred and noise image and three TV reconstructions. We use Gaussian blur with standard deviation 3.0, leading to a coefficient matrix K with a numerically infinite condition number, and the standard deviation of the Gaussian noise is $\sigma = 3$. The regularization parameter δ is chosen by the same equation (18) as in denoising.

For $\tau = 0.2$, Fig. 4 shows that we obtain an under-regularized solution dominated by inverted noise. The choice $\tau = 0.45$ gives a sufficiently piecewise-smooth image with satisfactory reconstruction of details, while $\tau = 1.0$ leads to an over-regularized image with too few details.

Fig. 3 Example of TV inpainting: damaged and noisy 512 × 512 image (same clean image as in Fig. 2), and the TV reconstruction

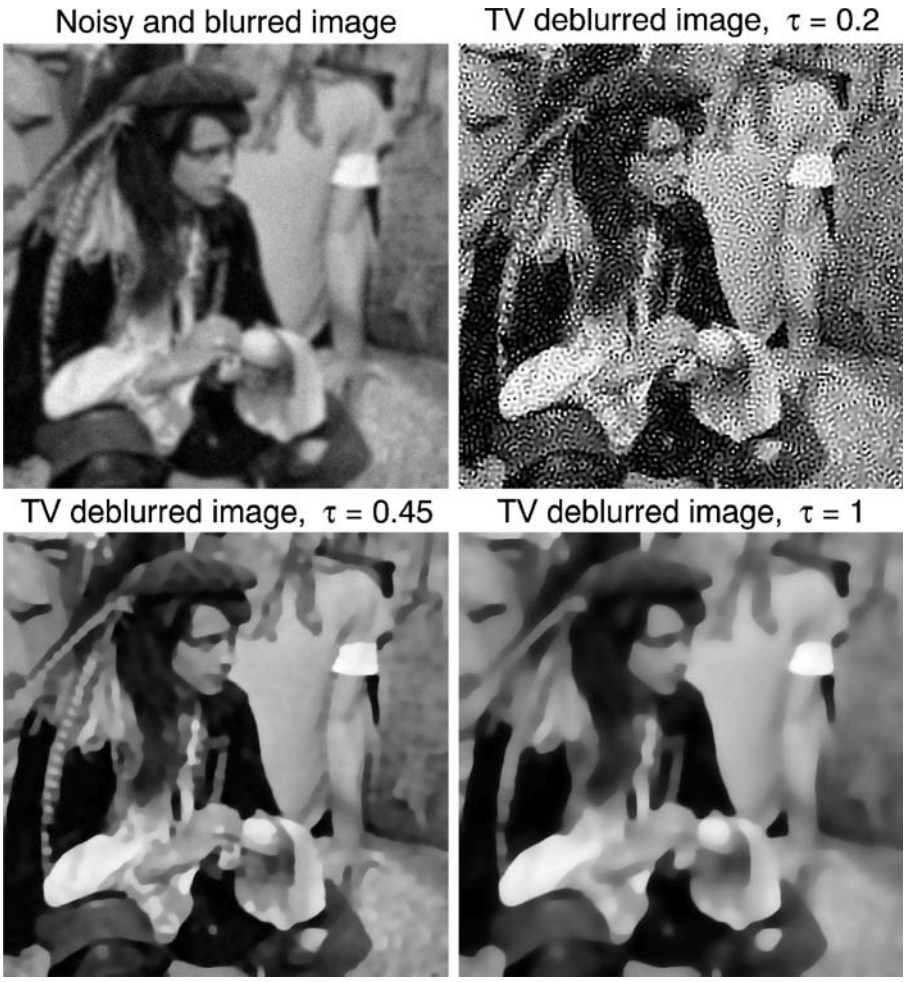

Fig. 4 Example of TV deblurring: blurred and noisy 512×512 image (same clean image as in Fig. 2), and TV reconstructions with three different values of τ

The computations associated with the blurring use the algorithm given in [15], and from the same source we use the Matlab functions `dcts2`, `idcts2`, and `dctshift` for the associated computations with the DCT.

7 Performance studies

The choice of ϵ obviously influences the computing time, and we choose to design our software such that the number of iterations remains unchanged when the image size is scaled—i.e., we want the bounds $\mathcal{N}_{\text{denoise}}$ (8), $\mathcal{N}_{\text{inpaint}}$ (12), and $\mathcal{N}_{\text{deblur}}$ (16) to be independent of the problem size mn. In order to achieve this, instead of setting an absolute ϵ in the stopping criterion we

use a *relative accuracy* ϵ_{rel} (with default value $\epsilon_{\text{rel}} = 10^{-3}$ for denoising and inpainting and $\epsilon_{\text{rel}} = 10^{-2}$ for deblurring), and then we set

$$\epsilon = \begin{cases} \|b\|_\infty \, mn \, \epsilon_{\text{rel}}, & \text{for denoising and deblurring} \\ \|b_{\mathcal{I}_c}\|_\infty \, mn \, \epsilon_{\text{rel}}, & \text{for inpainting.} \end{cases} \tag{20}$$

This choice, together with (18) and (19), leads to the bounds

$$\mathcal{N}_{\text{denoise}} \le \frac{4\sqrt{2}}{\epsilon_{\text{rel}}} \frac{\tau\,\sigma}{\|b\|_\infty}$$

$$\mathcal{N}_{\text{inpaint}} \le \frac{4\sqrt{2}}{\epsilon_{\text{rel}}} \sqrt{\left(\frac{\tau\,\sigma}{\|b_{\mathcal{I}_c}\|_\infty}\right)^2 \frac{|\mathcal{I}_c|}{mn} + \left(\frac{\max_{i\in\mathcal{I}_c} b_i - \min_{i\in\mathcal{I}_c} b_i}{2\,\|b_{\mathcal{I}_c}\|_\infty}\right)^2 \frac{|\mathcal{I}|}{mn}}$$

$$\mathcal{N}_{\text{deblur}} \le \frac{4\sqrt{2}}{\epsilon_{\text{rel}}} \sqrt{1 + \left(\frac{1}{\rho \max_i |\lambda_i|}\right)^2} \approx \frac{4\sqrt{2}}{\epsilon_{\text{rel}}} \frac{1}{\rho \max_i |\lambda_i|}.$$

For denoising, the bound is proportional to the relative noise level, as desired. For inpainting, the situation is more complex, but if the noise dominates then we have the same bound as in denoising, and otherwise the bound is proportional to the square root of the fraction of missing pixels. For deblurring, the bound is dominated by the term involving the smallest eigenvalue $\rho \max_i |\lambda_i|$ in the rank-deficient approximation.

To demonstrate the computational performance of our **TV denoising algorithm**, we created several smaller problems by extracting sub-images of the original clean image, and in each instance we added Gaussian white noise with standard deviation $\sigma = 25$ (similar to the previous section). We then solved these TV denoising problems using the default parameter $\epsilon_{\text{rel}} = 10^{-3}$ and for three different values of τ, and the actual number of iterations needed to solve the problem to ϵ-accuracy are shown in Fig. 5. We see that with the choice

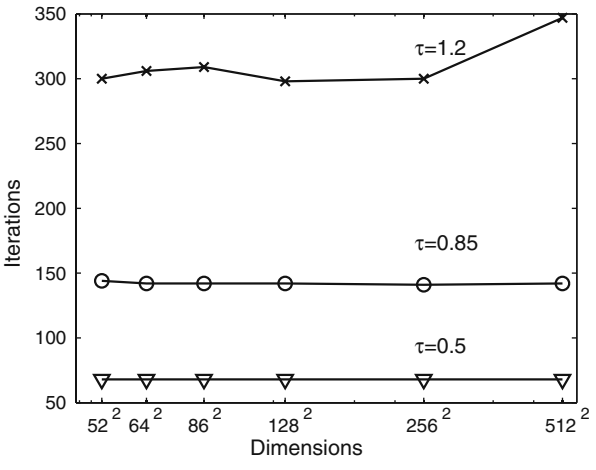

Fig. 5 The number of iterations in `TVdenoise` needed to compute an ϵ-accurate solution to the TV denoising problem, for varying image dimensions and three values of the parameter τ. The standard deviation of the image noise is $\sigma = 25$, and as stopping criterion we used the default value $\epsilon_{\text{rel}} = 10^{-3}$. For the three values of τ, the bounds for $\mathcal{N}_{\text{denoise}}$ are 278, 472, and 666, respectively

of ϵ in (20), the actual number of iterations is indeed almost independent of the problem size (except for unrealistic large τ). We also see that the actual number of iterations is approximately proportional to τ, and the bounds for $\mathcal{N}_{\text{denoise}}$ are somewhat pessimistic overestimates.

While the number of iterations is almost independent of the problem size, the computing time increases with the problem size because each iteration has $O(mn)$ complexity. Figure 6 shows the computing time for our TV denoising algorithm `TVdenoise`, and the dashed reference line confirms that the computing time is approximately linear in the problem size mn.

We compared our code with the codes `tvdenoise`, `perform_tv_denoising`, `TV_GPBBsafe` (from `TVGP`) and `SplitBregmanROF` from Table 1 (`TV_GPBBsafe` was chosen because it is the fastest method from `TVGP` for which convergence is guaranteed). These codes solves the Lagrange formulation of the TV denoising by minimizing problems on the form

$$\mathcal{T}(x) + \frac{1}{2\lambda}\|x - b\|_2^2. \tag{21}$$

There is equivalence between the regularized and the constrained TV denoising formulations. If we set

$$\delta = \lambda \|D^T u^*\|_2, \tag{22}$$

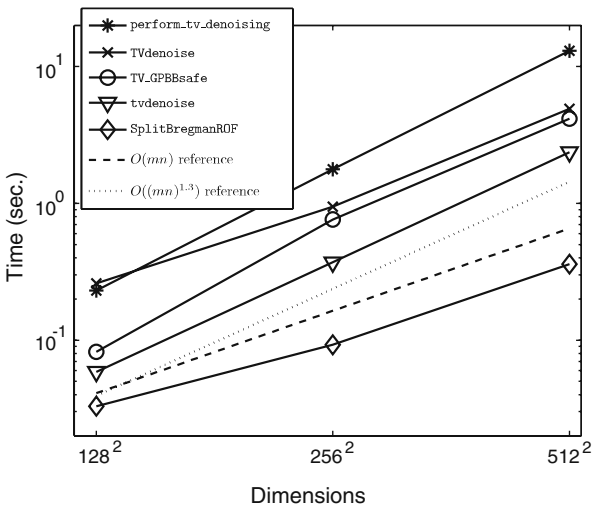

Fig. 6 The computing times (in seconds) for our TV denoising algorithm `TVdenoise` as a function of problem size mn, for the case $\sigma = 25$, $\tau = 0.85$, and $\epsilon_{\text{rel}} = 10^{-3}$. The *dashed reference line without markers* confirms that the computing time for `TVdenoise` is approximately linear in the problem size. We also show the computing times `tvdenoise`, `perform_tv_denoising`, `TV_GPBBsafe` (from `TVGP`) and `SplitBregmanROF` listed in Table 1. The *dotted reference line without markers* shows that the computing time for first two of the mentioned algorithms is approximately $O((mn)^{1.3})$, whereas `SplitBregmanROF` scales approximately linear

where u^* is the solution to the dual problem (5), then the two problems (4) and (21) are equivalent [13].

First we solved (5) to high accuracy with $\epsilon_{\text{rel}} = 10^{-6}$ for 100 different noise realizations, and then used (22) to obtain the corresponding Lagrange multiplier λ. We then picked the highest number of iterations for `tvdenoise`, `perform_tv_denoising`, and `TV_GPBBsafe` such that these codes returned a solution $x_{\mathcal{R}}$ slightly less accurate than the solution x from our code, i.e.,

$$\mathcal{R}_{\text{denoise}}(x) \le \mathcal{R}_{\text{denoise}}(x_{\mathcal{R}})$$

where

$$\mathcal{R}(x) = \sum_{i=2}^{m-1} \sum_{j=2}^{n-1} \| D_{(ij)}\, x \|_2 + \frac{1}{2\lambda} \| (x-b)_{\mathcal{J}} \|_2^2 , \tag{23}$$

where $\mathcal{J}$ is the index set of all inner pixels. The image boundaries are removed in (23) to reduce the effect of the boundary conditions imposed by the different algorithms.

The average computing times are shown in Fig. 6, and we see that the codes `tvdenoise`, `perform_tv_denoising`, and `TV_GPBBsafe` (for larger images) have a complexity of about $O((mn)^{1.3})$ as confirmed by the dotted reference line. For large images `perform_tv_denoising` is the slowest of these codes, while `tvdenoise` and `TV_GPBBsafe` are faster. The code `SplitBregmanROF` is the fastest and it scales with a complexity of about $O(mn)$. For the image dimensions shown, our code is faster than `perform_tv_denoising` but slower than `tvdenoise`, `TV_GPBBsafe`, and `SplitBregmanROF`. However, due to the lower complexity our algorithm scales as good as `SplitBregmanROF`.

For the **TV inpainting algorithm** the computing times depend on image dimensions and noise level as well as on the number and distribution of the missing pixels. We illustrate this with an example with noise level $\sigma = 15$ (similar to Fig. 3, and with the parameters $\tau = 0.85$ and $\epsilon_{\text{rel}} = 10^{-3}$). The problem shown in Fig. 3 (with the text mask) is solved in 28.1 s. However, if we generate a mask with same number of missing pixels located in a circle (of radius 93 pixels) in the middle of the image, then the computing time is only 6.8 s. Finally, with no missing pixels the problem reduces to the denoising problem, and it is solved in 3.2 s.

Table 2 Performance studies for inpainting, using our software `TVinpaint` and the script `tv_dode_2D` from Table 1

	Time	Its.	$\mathcal{N}_{\text{inpaint}}$
`TVdenoise`			
Inpaint text	28.1 s	751	954
Inpaint circle	6.8 s	190	958
Denoise	3.2 s	93	283
`tv_dode_2D`			
Inpaint text	729.5 s	142	

Table 3 Performance studies for deblurring of the image in Fig. 4

τ	Time	Its.	$\mathcal{N}_{\text{deblur}}$
0.20	15.7 s	174	1767
0.45	13.7 s	152	1766
1.00	19.4 s	222	1764

For comparison we also used the script `tv_dode_2D` from Table 1, which solves the problem in 729.5 s using default parameters. The Lagrange multiplier λ was selected such that the two components in (23) for `TVinpaint` were slightly smaller than those for `tv_dode_2D`.

Table 2 lists the computing times, the actual number of iterations, and the upper bound $\mathcal{N}_{\text{inpaint}}$ for the three variations of the inpainting problem. We see that $\mathcal{N}_{\text{inpaint}}$ is indeed an upper bound for the number of iterations, and that it can be very pessimistic if the problem is "easy."

For the **TV deblurring algorithm** the computing times depend on image dimensions, the noise level σ, and the parameters τ and ρ. The performance results for the examples in Fig. 4, obtained with the default $\rho = 10^{-3}$, are listed in Table 3. The bound $\mathcal{N}_{\text{deblur}}$ is extremely pessimistic, because it is independent of δ (and thus τ), and we see that the actual number of iterations depends on τ.

It follows from the complexity bound for $\mathcal{N}_{\text{deblur}}$ that the number of iterations also depends on the relative threshold ρ in our rank reduction. Table 4 reports the performance results for the same deblurring problem as above with varying ρ and fixed $\tau = 0.6$. As expected we see that the computing time depends on ρ. The smaller the ρ the more ill conditioned the problem and therefor the longer the computing time.

The last column shows the relative error $R_\rho = \|x_\rho - x_{10^{-7}}\|_2 / \|x_{10^{-7}}\|_2$ in the solutions for $\rho = 10^{-1}, 10^{-2}, \ldots, 10^{-6}$ compared to the solution for $\rho = 10^{-7}$. Interestingly, the relative error between the reconstructions computed for $\rho = 10^{-3}$ and 10^{-7} is only about 3% (the images are virtually identical to the eye), while there is a factor of almost 10 in computing time. Hence we choose the default value $\rho = 10^{-3}$ to allow fast experiments with the factor τ; when a suitable τ has been found the user may choose a smaller ρ to improve the accuracy of the solution. (For $\rho \geq 10^{-2}$ the rank reduction has a substantial and undesired regularizing effect on the solution.)

Table 4 Performance studies for deblurring when varying the rank reduction threshold ρ and using $\tau = 0.6$

ρ	Time	Its.	$\mathcal{N}_{\text{deblur}}$	R_ρ
10^{-1}	11.6 s	138	$6.2 \cdot 10^2$	0.042
10^{-2}	11.7 s	134	$6.4 \cdot 10^2$	0.037
10^{-3}	15.6 s	173	$1.7 \cdot 10^3$	0.033
10^{-4}	25.8 s	308	$1.6 \cdot 10^4$	0.028
10^{-5}	48.5 s	552	$1.6 \cdot 10^5$	0.021
10^{-6}	83.0 s	945	$1.7 \cdot 10^6$	0.012
10^{-7}	143.2 s	1574	$1.7 \cdot 10^7$	

Fig. 7 Example of TV deblurring of the noisy and blurred image from Fig. 4 using `FTVdG` (*left*) and `TVdeblur` (*right*) with $\rho = 10^{-3}$

We compared our code with the code `FTVdG` from Table 1, which solves the TV deblurring problem by minimizing

$$\mathcal{T}(x) + \frac{1}{2\lambda}\|\widetilde{K}x - b\|_2^2, \tag{24}$$

where the matrix $\widetilde{K}$ represents spatial invariant blurring with periodic boundary conditions. Using the default settings, we first select λ such that the TV – ignoring boundary elements – of the `FTVdG` solution $x_{\texttt{FTVdG}}$ is approximately the same as for our solution $x_{\texttt{TVdeblur}}$. The solutions are shown in Fig. 7 and the corresponding results are summarized in Table 5, where

$$\widetilde{\mathcal{R}}(x) = \sum_{i=2}^{m-1}\sum_{j=2}^{n-1}\|D_{(ij)}\,x\|_2 + \frac{1}{2\lambda}\|\widetilde{K}x - b\|_2^2.$$

These results demonstrate that although we can reproduce the value of the TV with the default settings of `FTVdG`, we are not able to obtain the same reconstruction, reflected in the fact that $\widetilde{\mathcal{R}}(x_{\texttt{FTVdG}}) > \widetilde{\mathcal{R}}(x_{\texttt{TVdeblur}})$.

Table 5 Comparison of the deblurring algorithms `TVdeblur` and `FTVdG`. The parameters are chosen such that the solutions have the same TV equal to 10^6

	$\|\widetilde{K}x - b\|_2^2$	$\widetilde{\mathcal{R}}(x)$	Time (s)
`TVdeblur`	$4.66 \cdot 10^3$	$1.63 \cdot 10^6$	13.7
`FTVdG` (default)	$5.81 \cdot 10^3$	$1.95 \cdot 10^6$	10.6
`FTVdG` (modified)	$5.79 \cdot 10^3$	$1.99 \cdot 10^6$	27.9

The table also shows results for a test with modified FTVdG settings $\beta = 2^{16}$ and $\epsilon = 10^{-5}$, cf. [23]. Here we needed to use a slightly different λ such that the above-mentioned TV requirement still holds. Table 5 shows that even with the modified settings, we are not able to obtain a much better solution as measured by $\widetilde{\mathcal{R}}(x_{\texttt{FTVdG}})$. In fact, it was not possible to adjust the settings for FTVdG such that $\widetilde{\mathcal{R}}(x_{\texttt{FTVdG}}) < 1.1\,\widetilde{\mathcal{R}}(x_{\texttt{TVdeblur}})$.

8 Conclusion

Total variation (TV) formulations provide a good basis for reconstruction of noisy, corrupted, and blurred images. In this paper we present easy-to-use public domain software for TV denoising, inpainting, and deblurring, using recently developed first-order optimization algorithms with complexity $O(1/\epsilon)$, where ϵ is the accuracy of the solution. Each iteration in our algorithms only requires moderate computation, of the order $O(mn)$ for denoising and inpainting, and $O(mn \log \max\{m, n\})$ for deblurring. Image deblurring often involves highly ill-conditioned matrices, and to improve both speed and numerical stability we use the technique of rank-reduction for such problems.

Our codes are written in C with Matlab interfaces, and they are available from www.netlib.org/numeralgo in the file na28. The codes are robust, user friendly (they require no extra parameters), and they are suited for large problems. The Matlab files have been tested on Matlab versions 7.5–7.8, and they require version 7.5 or later.

Acknowledgements We wish to thank Michela Redivo Zaglia, Giuseppe Rodriguez, and an anonymous referee for many valuable comments that helped to improve the paper and the software.

Appendix A: The matlab functions

TVdenoise

```
X = TVdenoise(B,delta)
[X,info] = TVdenoise(B,delta,eps_rel)
```

This function solves the TV denoising problem

$$\text{minimize } \texttt{TV(X)} \qquad \text{subject to} \qquad \|\texttt{X} - \texttt{B}\|_{\texttt{F}} \leq \texttt{delta}$$

where B is a noisy image, X is the reconstruction, and delta is an upper bound for the residual norm. The TV function is the 1-norm of the gradient magnitude, computed via neighbor pixel differences. At the image borders, we imposed reflexive boundary conditions for the gradient computations.

The parameter delta should be of the same size as the norm of the image noise. If the image is $m \times n$, and σ is the standard deviation of the image noise

in a pixel, then we recommend to use `delta` $= \tau\sqrt{mn}\,\sigma$, where τ is slightly smaller than one, say, $\tau = 0.85$.

The function returns an ϵ-optimal solution `X`, meaning that if `X`* is the exact solution, then our solution `X` satisfies

$$\texttt{TV(X)} - \texttt{TV(X}^*\texttt{)} \le \epsilon = \max(\texttt{B(:)})\, mn\, \texttt{eps\_rel},$$

where `eps_rel` is a specified relative accuracy (default `eps_rel` $= 10^{-3}$). The solution status is returned in the stuct `info`; write `help TVdenoise` for more information.

TVinpaint

```
X = TVinpaint(B,M,delta)
[X,info] = TVinpaint(B,M,delta,eps_rel)
```

This function solves the TV inpainting problem

$$\text{minimize } \texttt{TV(X)} \qquad \text{subject to} \qquad \|\texttt{X(Ic)} - \texttt{B(Ic)}\|_{\texttt{F}} \le \texttt{delta}$$

where `B` is a noisy image with missing pixels, `Ic` are the indices to the intact pixels, `X` is the reconstruction, and `delta` is an upper bound for the residual norm. The `TV` function is the 1-norm of the gradient magnitude, computed via neighbor pixel differences. At the image borders, we imposed reflexive boundary conditions for the gradient computations.

The information about the intact and missing pixels is given in the form of the mask `M`, which is a matrix of the same size as `B`, and whose nonzero elements indicate missing pixels.

The parameter `delta` should be of the same size as the norm of the image noise. If the image is $m \times n$, and σ is the standard deviation of the image noise in a pixel, then we recommend to use `delta` $= \tau\sqrt{mn}\,\sigma$, where τ is slightly smaller than one, say, $\tau = 0.85$.

The function returns an ϵ-optimal solution `X`, meaning that if `X`* is the exact solution, then our solution `X` satisfies

$$\texttt{TV(X)} - \texttt{TV(X}^*\texttt{)} \le \epsilon = \max(\texttt{B(Ic)})\, mn\, \texttt{eps\_rel},$$

where `eps_rel` is the specified relative accuracy (default `eps_rel` $= 10^{-3}$). The solution status is returned in the stuct `info`; write `help TVinpaint` for more information.

TVdeblur

```
X = TVdeblur(B,PSF,delta)
[X,info] = TVdeblur(B,PSF,delta,eps_rel,rho,gamma)
```

This function solves the TV deblurring problem

$$\text{minimize } \texttt{TV(X)} \qquad \text{subject to} \qquad \|\texttt{PSF} \star \texttt{X} - \texttt{B}\|_{\texttt{F}} \le \delta$$

where `B` is a blurred noisy image, `X` is the reconstruction, and `delta` is an upper bound for the residual norm. The `TV` function is the 1-norm of the gradient

magnitude, computed via neighbor pixel differences. At the image borders, we imposed reflexive boundary conditions for the gradient computations.

`PSF` $\star$ `X` is the image `X` convolved with the doubly symmetric point spread function `PSF` using reflexive boundary conditions. In the code, the blurring matrix that represents `PSF` is replaced by a rank-deficient well-conditioned approximation obtained by neglecting all eigenvalues smaller than `rho` times the largest eigenvalue (default `rho` $= 10^{-3}$).

The parameter `delta` should be of the same size as the norm of the image noise. If the image is $m \times n$, and σ is the standard deviation of the image noise in a pixel, then we recommend to use $\delta = \tau\sqrt{mn}\,\sigma$, where τ is smaller than one, say $\tau = 0.55$.

The parameter `gamma` is a an upper bound on the norm of the solution's component in the subspace corresponding to the neglected eigenvalues. The default value is `gamma` $= \sqrt{mn}$ max(`B(:)`) which should be sufficient for most problems.

The function returns an ϵ-optimal solution `X`, meaning that if `X`* is the exact solution, then our solution `X` satisfies

$$\texttt{TV(X)} - \texttt{TV(X}^*\texttt{)} \le \epsilon = \max(\texttt{B(:)})\, mn\, \texttt{eps\_rel},$$

where `eps_rel` is a specified relative accuracy (default `eps_rel` $= 10^{-2}$). The solution status is returned in the stuct `info`; write `help TVdeblur` for more information.

Appendix B: The norm of the derivative matrix

The matrix D defined in (2) can always be written as [15]

$$D = \Pi \begin{pmatrix} I_n \otimes L_m \\ L_n \otimes I_m \end{pmatrix},$$

where Π is a permutation matrix, I_p is the identity matrix of order p, and L_p is the chosen $p \times p$ first-derivative matrix with SVD $L_p = U_p \Sigma_p V_p^T$. We note that since L_p is a sparse matrix with 1 and -1 as the only two nonzero elements per row, it follows that $\|L_p\|_\infty = 2$. The 2-norm of D satisfies $\|D\|_2^2 = \lambda_{\max}(D^T D)$, the largest eigenvalue of $D^T D$, and hence we consider this matrix:

$$\begin{aligned} D^T D &= (I_n \otimes L_m)^T (I_n \otimes L_m) + (L_n \otimes I_m)^T (L_n \otimes I_m) \\ &= I_n \otimes L_m^T L_m + L_n^T L_n \otimes I_m \\ &= V_n V_n^T \otimes V_m \Sigma_m^2 V_m^T + V_n \Sigma_n^2 V_n^T \otimes V_m V_m^T \\ &= (V_n \otimes V_m)\left(I_n \otimes \Sigma_m^2 + \Sigma_n^2 \otimes I_m\right)(V_n \otimes V_m)^T \end{aligned}$$

Since the middle matrix is diagonal, it follows that

$$\lambda_{\max}\left(D^T D\right) = \lambda_{\max}\left(\Sigma_m^2\right) + \lambda_{\max}\left(\Sigma_n^2\right) = \|L_m\|_2^2 + \|L_n\|_2^2 \le \|L_m\|_\infty^2 + \|L_n\|_\infty^2 = 8.$$

We note that a completely different proof is given in [4, Thm. 3.1].

References

1. Alter, F., Durand, S., Froment, J.: Adapted total variation for artifact free decompression of JPEG images. J. Math. Imaging Vis. **23**, 199–211 (2005)
2. Aujol, J.-F.: Some first-order algorithms for total variation based image restoration. J. Math. Imaging Vis. **34**, 307–327 (2009)
3. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
4. Chambolle, A.: An algorithm for total variation minimization and applications. J. Math. Imaging Vis. **20**, 89–97 (2004)
5. Chan, T.F., Shen, J.: Image processing and analysis: variational, PDE, wavelet, and stochastic methods. SIAM, Philadelphia (2005)
6. Chan, T.F., Golub, G.H., Mulet, P.: A nonlinear primal-dual method for total variation-based image restoration. SIAM J. Sci. Comput. **20**, 1964–1977 (1999)
7. Combettes, P.L., Pennanen, T.: Generalized mann iterates for constructing fixed points in Hilbert spaces. J. Math. Anal. Appl. **275**, 521–536 (2002)
8. Conn, A.R., Gould, N.I.M., Toint, P.L.: Trust-region methods. SIAM, Philadelphia (2000)
9. Darbon, J., Sigelle, M.: Image restoration with discrete constrained total variation. Part I: fast and exact optimization. J. Math. Imaging Vis. **26**, 261–276 (2006)
10. FFTW: Freely available C subroutine library for computing the discrete Fourier transform (DFT) in one or more dimensions. http://www.fftw.org (2009)
11. Fornasier, M., Schönlieb, C.-B.: Subspace correction methods for total variation and ℓ_1-minimization. SIAM J. Numer. Anal. (2009, in press)
12. Frigo, M., Johnson, S.G.: The design and implementation of FFTW3. Proc. IEEE **93**, 216–231 (2005)
13. Goldfarb, D., Yin, W.: Second-order cone programming methods for total variation-based image restoration. SIAM J. Sci. Comput. **27**, 622–645 (2005)
14. Goldstein, T., Osher, S.: The split Bregman method for L_1 regularized problems. SIAM J. Imaging Sci. **2**, 323–343 (2009)
15. Hansen, P.C., Nagy, J.G., O'Leary, D.P.: Deblurring Images: Matrices, Spectra, and Filtering. SIAM, Philadelphia (2006)
16. Krishnan, D., Ping, L., Yip, A.M.: A primal-dual active-set methods for non-negativity constrained total variation deblurring problems. IEEE Trans. Image Process. **16**, 2766–2777 (2007)
17. Nesterov, Yu.: Introductory lectures on convex optimization. Kluwer, Dordrecht (2004)
18. Nesterov, Yu.: Smooth minimization of nonsmooth functions. Math. Program. Ser. A **103**, 127–152 (2005)
19. Nesterov, Yu.: Excessive gap technique in non-smooth convex optimization. SIAM J. Optim. **16**, 235–249 (2005)
20. Nesterov, Yu.: Gradient methods for minimizing composite objective functions. CORE Discussion Papers series, Université Catholique de Louvain, Center for Operations Research and Econometrics. http://www.uclouvain.be/en-44660.html (2007)
21. Vogel, C.R.: Computational methods for inverse problems. SIAM, Philadelphia (2002)
22. Weiss, P., Blanc-Féraud, L., Aubert, G.: Efficient schemes for total variation minimization under constraints in image processing. SIAM J. Sci. Comput. **31**, 2047–2080 (2009)
23. Yang, J., Zhang, Y., Yin, W., Wang, Y.: A new alternating minimization algorithm for total variation image reconstruction. SIAM J. Imaging Sci. **1**, 248–272 (2008)
24. Zhu, M., Wright, S., Chan, T.F.: Duality-based algorithms for total-variation-regularized image restoration. Comput. Optim. Appl. (2008). doi:10.1007/s10589-008-9225-2