Noname manuscript No.
(will be inserted by the editor)

Duality-Based Algorithms for Total Variation Image Restoration

Mingqiang Zhu · Stephen J. Wright · Tony F. Chan

Received: date / Accepted: date

Abstract Image restoration models based on total variation (TV) have become popular since their introduction by Rudin, Osher, and Fatemi (ROF) in 1992. The dual formulation of this model has a quadratic objective with separable constraints, making projections onto the feasible set easy to compute. This paper proposes application of gradient projection (GP) algorithms to the dual formulation. We test variants of GP with different step selection and line search strategies, including techniques based on the Barzilai-Borwein method. Global convergence can in some cases be proved by appealing to existing theory. We also propose a sequential quadratic programming (SQP) approach that takes account of the curvature of the boundary of the dual feasible set. Computational experiments show that the proposed approaches perform well in a wide range of applications and that some are significantly faster than previously proposed methods, particularly when only modest accuracy in the solution is required.

Keywords Image Denoising · Constrained Optimization · Gradient Projection

This work was supported by National Science Foundation grants DMS-0610079, DMS-0427689, CCF-0430504, CTS-0456694, and CNS-0540147, and Office of Naval Research grant N00014-06-1-0345. The work was performed while T. Chan was on leave at the National Science Foundation as Assistant Director of Mathematics and Physical Sciences.
Technical Report, May 19, 2008.

M. Zhu
Mathematics Department, UCLA, Box 951555, Los Angeles, CA 90095-1555, USA
E-mail: mqzhu@ucla.edu

S. J. Wright
Department of Computer Sciences, University of Wisconsin, 1210 W. Dayton Street, Madison, WI 53705, USA
E-mail: swright@cs.wisc.edu

T. F. Chan
Mathematics Department, UCLA, Box 951555, Los Angeles, CA 90095-1555, USA
E-mail: tfchan@nsf.gov

1 Introduction

1.1 Background

Variational models have been extremely successful in a wide variety of image restoration problems and remain one of the most active areas of research in mathematical image processing and computer vision. The most fundamental image restoration problem is perhaps denoising. It forms a significant preliminary step in many machine vision tasks such as object detection and recognition. Total variation (TV)-based image restoration models were first introduced by Rudin, Osher, and Fatemi (ROF) in their pioneering work [17]. It was designed with the explicit goal of preserving sharp discontinuities (edges) in an image while removing noise and other unwanted fine-scale detail. ROF formulated the following minimization problem:

$$\min_u \int_\Omega |\nabla u| \qquad \text{s.t.} \quad \|u - f\|_2^2 \leq \sigma^2. \tag{1}$$

Here, Ω denotes the image domain, which will be taken to be a bounded domain in $\mathbb{R}^n$ with Lipschitz boundary. Usually Ω is simply a rectangle in $\mathbb{R}^2$, modeling the computer screen. The function $f : \Omega \to \mathbb{R}$ represents the given observed image and σ^2 is an estimate of the variance of the noise in the image f. The notation $|\cdot|$ represents the Euclidean (ℓ_2) norm on $\mathbb{R}^2$. The objective function in the formulation (1) is the TV semi-norm of u.

Rather than solving the constrained minimization problem (1), ROF and subsequent researchers also formulated an unconstrained minimization problem which uses the TV term as a Tikhonov regularization:

$$\min_u \; P(u) := \int_\Omega |\nabla u| \; dx + \frac{\lambda}{2}\|u - f\|_2^2. \tag{2}$$

This above problem yields the same solution as (1) for a suitable choice of the Lagrange multiplier λ (see [6]).

Recently, many researchers have proposed algorithms that make use of the dual formulation of the ROF model; see, for example [8], [4], and [5]. To derive this form, we first notice the TV semi-norm has the following equivalent forms

$$\int_\Omega |\nabla u| = \max_{w \in C_0^1(\Omega),\, |w| \leq 1} \int_\Omega \nabla u \cdot w = \max_{|w| \leq 1} \int_\Omega -u \nabla \cdot w, \tag{3}$$

where $w : \Omega \to \mathbb{R}^2$. The rightmost definition of the TV semi-norm is more general since it requires the function u only to have bounded variation (BV), not necessarily to be smooth. In fact, this is the formal definition of TV semi-norm for the space of BV functions.

With this definition of TV, the ROF model becomes

$$\min_u \max_{w \in C_0^1(\Omega),\, |w| \leq 1} \int_\Omega -u \nabla \cdot w + \frac{\lambda}{2}\|u - f\|_2^2,$$

where u and w are the primal and dual variables, respectively. The min-max theorem (see e.g., [12, Chapter VI, Proposition 2.4]) allows us to interchange the min and max, to obtain

$$\max_{w \in C_0^1(\Omega),\, |w| \leq 1} \min_u \int_\Omega -u \nabla \cdot w + \frac{\lambda}{2} \|u - f\|_2^2.$$

The inner minimization problem can be solved exactly as follows:

$$u = f + \frac{1}{\lambda} \nabla \cdot w \tag{4}$$

leading to the following dual formulation:

$$\max_{w \in C_0^1(\Omega),\, |w| \leq 1} D(w) := \frac{\lambda}{2} \left[\|f\|_2^2 - \left\| \frac{1}{\lambda} \nabla \cdot w + f \right\|_2^2 \right], \tag{5}$$

or, equivalently,

$$\min_{w \in C_0^1(\Omega),\, |w| \leq 1} \frac{1}{2} \|\nabla \cdot w + \lambda f\|_2^2. \tag{6}$$

For a primal-dual feasible pair (u, w), the duality gap $G(u, w)$ is defined to be the difference between the primal and the dual objectives:

$$\begin{aligned} G(u, w) &= P(u) - D(w) \\ &= \int_\Omega |\nabla u| + \frac{\lambda}{2} ||u - f||_2^2 - \frac{\lambda}{2} \left[\|f\|_2^2 - \left\| \frac{1}{\lambda} \nabla \cdot w + f \right\|_2^2 \right] \\ &= \int_\Omega \Big(|\nabla u| - \nabla u \cdot w \Big) + \frac{\lambda}{2} \left\| \frac{1}{\lambda} \nabla \cdot w + f - u \right\|_2^2. \end{aligned} \tag{7}$$

The duality gap bounds the distance to optimality of the primal and dual objectives. Specifically, if u and w are feasible for the primal (2)) and dual (5) problems, respectively, we have

$$0 \leq P(u) - O^* \leq G(u, w), \tag{8a}$$
$$0 \leq O^* - D(w) \leq G(u, w), \tag{8b}$$

where O^* is the (common) primal-dual optimal objective value. In the dual-based algorithms proposed in this paper, the primal variable u is calculated using equation (4). This choice of u eliminates the second term in (7). We make use of the duality gap in the termination test of Section 4.

Over the years, the ROF model has been extended to many other image restoration tasks and has been modified in a variety of ways to improve its performance (see [7] and the references therein). However, in our paper, we will focus on the original TVL2 model (2) and more particularly on its dual formulation (6).

1.2 Notation and Discrete Formulation

Before describing the numerical algorithms, let us fix our main notational conventions.

Often in this paper we need to concatenate vectors and matrices, in both column-wise or row-wise fashion. We follow the MATLAB convention of using "," for adjoining vectors and matrices in a row, and ";" for adjoining them in a column. Thus, for any vectors x, y and z, the following are synonymous:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = (x^T,\, y^T,\, z^T)^T = (x; y; z)$$

For simplicity, we assume that the domain Ω is the unit square $[0,1]\times[0,1]$, and define a discretization via a regular $n \times n$ grid of pixels, indexed as (i,j), for $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, n$. The index (i,j) represents the point $(i/(n+1), j/(n+1)) \in \Omega$. We represent images images as two-dimensional matrices of dimension $n \times n$, where $u_{i,j}$ represents the value of the function u at the point indexed by (i,j). (Adaptation to less regular domains is not difficult in principle.) To define the discrete total variation, we introduce a discrete gradient operator, whose two components at each pixel (i,j) are defined as follows:

$$(\nabla u)^1_{i,j} = \begin{cases} u_{i+1,j} - u_{i,j} & \text{if } i < n \\ 0 & \text{if } i = n \end{cases} \tag{9a}$$

$$(\nabla u)^2_{i,j} = \begin{cases} u_{i,j+1} - u_{i,j} & \text{if } j < n \\ 0 & \text{if } j = n. \end{cases} \tag{9b}$$

(Thus $\nabla u \in \mathbb{R}^{n\times n\times 2}$.) The discrete TV of u is then defined by

$$\mathrm{TV}(u) = \sum_{1\le i,j,\le n} \|(\nabla u)_{i,j}\|.$$

Here and throughout the paper, we use $\|\cdot\|$ and $\|\cdot\|_2$ interchangeably, to denote the Euclidean (ℓ_2) norm of a vector of real numbers. Note that this norm is not a smooth function of its argument. It has the classic "ice-cream cone" shape, nondifferentiable when its argument vector is zero.

The discrete divergence operator is defined, by analogy with the continuous setting, as the negative adjoint of the gradient operator, that is, $\nabla\cdot = -\nabla^*$. Defining the inner product of two objects in $\mathbb{R}^{n\times n}$ as follows:

$$\langle u, v\rangle = \sum_{1\le i,j\le n} u_{i,j} v_{i,j},$$

(and similarly for objects in $\mathbb{R}^{n\times n\times 2}$), we have from definition of the discrete divergence operator that for any $u \in \mathbb{R}^{n\times n}$ and $w \in \mathbb{R}^{n\times n\times 2}$, that $\langle \nabla u, w\rangle =$

$\langle u, -\nabla \cdot w \rangle$. It is easy to check that the divergence operator can be defined explicitly as follows:

$$(\nabla \cdot w)_{i,j} = \begin{cases} w^1_{i,j} - w^1_{i-1,j} & \text{if } 1 < i < n \\ w^1_{i,j} & \text{if } i = 1 \\ -w^1_{i-1,j} & \text{if } i = n \end{cases} + \begin{cases} w^2_{i,j} - w^2_{i,j-1} & \text{if } 1 < j < n \\ w^2_{i,j} & \text{if } j = 1 \\ -w^2_{i,j-1} & \text{if } j = n. \end{cases}$$

To describe the problem in matrix algebra language, we reorder the image matrix u (resp. f) in row-wise fashion into a vector v (resp. g), associating the (i, j) element of the two-dimensional structure with the element $(j-1)n + i$ of the vector structure, as follows:

$$v_{(j-1)n+i} = u_{i,j}, \quad 1 \le i, j \le n.$$

We have $v \in \mathbb{R}^N$, where $N = n^2$. The (i, j) component of the gradient (9) can thus be represented as a multiplication of the vector $v \in \mathbb{R}^N$ by a matrix $A_l^T \in \mathbb{R}^{2\times N}$, for $l = 1, 2, \ldots, N$:

$$A_l^T v = \begin{cases} (v_{l+1} - v_l;\ v_{l+n} - v_l) & \text{if } l \bmod n \neq 0 \text{ and } l + n \le N \\ (0;\ v_{l+n} - v_l) & \text{if } l \bmod n = 0 \text{ and } l + n \le N \\ (v_{l+1} - v_l;\ 0) & \text{if } l \bmod n \neq 0 \text{ and } l + n > N \\ (0;\ 0) & \text{if } l \bmod n = 0 \text{ and } l + n > N. \end{cases} \tag{10}$$

Using this notation, the discrete version of the primal ROF model (2) can be written as follows:

$$\min_v \sum_{l=1}^{N} \|A_l^T v\|_2 + \frac{\lambda}{2}\|v - g\|_2^2 \tag{11}$$

Similarly, we restructure the dual variable w, using a row-wise ordering of the indices (i, j), into a collection of vectors $x_l \in \mathbb{R}^2$, $l = 1, 2, \ldots, N$, as follows:

$$x_{(j-1)n+i} = \begin{bmatrix} w^1_{i,j} \\ w^2_{i,j} \end{bmatrix}, \quad 1 \le i, j \le n.$$

The complete vector $x \in \mathbb{R}^{2N}$ of unknowns for the discretized dual problem is then obtained by concatenating these subvectors: $x = (x_1; x_2; \ldots; x_N)$. We also form the matrix A by concatenating the matrices A_l, $l = 1, 2, \ldots, N$ defined in (10), that is, $A = (A_1, \ldots, A_N) \in \mathbb{R}^{N\times 2N}$. In this notation, the divergence $\nabla \cdot w$ is simply $-Ax$, so the discretization of the dual ROF model (6) is

$$\min_{x\in X} \frac{1}{2}\|Ax - \lambda g\|_2^2 \tag{12}$$
$$\text{where} \quad X := \{(x_1; x_2; \ldots; x_N) \in \mathbb{R}^{2N} : x_l \in \mathbb{R}^2, \\ \|x_l\|_2 \le 1 \text{ for all } l = 1, 2, \ldots, N\}.$$

1.3 A Fundamental Convergence Result

Here we make several remarks on the discretized problems (11), (12) and prove a general convergence result. It is easy to verify that both problems can be obtained from the function $\ell : \mathbb{R}^N \times X \to \mathbb{R}$ defined as follows:

$$\ell(v, x) := x^T A^T v + \frac{\lambda}{2}\|v - g\|_2^2. \tag{13}$$

The primal problem (11) is simply

$$\min_{v \in \mathbb{R}^N} \max_{x \in X} \ell(v, x),$$

while the dual problem (12) is equivalent to

$$\max_{x \in X} \min_{v \in \mathbb{R}^N} \ell(v, x).$$

It is easy to verify that the conditions (H1), (H2), (H3), and (H4) of [15, pp. 333-334] are satisfied by this setting. Thus, it follows from [15, Chapter VII, Theorem 4.3.1] that ℓ has a nonempty convex set of saddle points $(\bar{v}, \bar{x}) \in \mathbb{R}^N \times X$. Moreover, from [15, Chapter IV, Theorem 4.2.5] and compactness of X, the point $(\bar{v}, \bar{x}) \in \mathbb{R}^N \times X$ is a saddle point if and only if $\bar{v}$ solves (11) and $\bar{x}$ solves (12).

Note that by strict convexity of the objective in (11), the solution $\bar{v}$ of (11) is in fact uniquely defined. For any saddle point $(\bar{v}, \bar{x})$, we have that $\ell(\bar{v}, \bar{x}) \leq \ell(v, \bar{x})$ for all $v \in \mathbb{R}^N$, that is, $\bar{v}$ is a minimizer of $\ell(\cdot, \bar{x})$. Thus, from optimality conditions for $\ell(\cdot, \bar{x})$, the following relationship is satisfied for the unique solution $\bar{v}$ of (11) and for any solution $\bar{x}$ of (12):

$$A\bar{x} + \lambda(\bar{v} - g) = 0. \tag{14}$$

By uniqueness of $\bar{v}$, it follows that $A\bar{x}$ is constant for all solutions $\bar{x}$ of (12).

The following general convergence result will be useful in our analysis of algorithms in Section 2.

Proposition 1 *Let $\{x^k\}$ be any sequence with $x^k \in X$ for all $k = 1, 2, \ldots$ such that all accumulation points of $\{x^k\}$ are stationary points of (12). Then the sequence $\{v^k\}$ defined by*

$$v^k = g - \frac{1}{\lambda} A x^k \tag{15}$$

converges to the unique solution $\bar{v}$ of (11).

Proof Note first that all stationary points of (12) are in fact (global) solutions of (12), by convexity.

Suppose for contradiction that $v^k \not\to \bar{v}$. Then we can choose $\epsilon > 0$ and a subsequence $\mathcal{S}$ such that $\|v^k - \bar{v}\|_2 \geq \epsilon$ for all $k \in \mathcal{S}$. Since all x^k belong to the bounded set X, the sequence $\{x^k\}$ is bounded, so $\{v^k\}$ is bounded also. In particular, the subsequence $\{v^k\}_{k \in \mathcal{S}}$ must have an accumulation point $\hat{v}$,

which must satisfy $\|\hat{v} - \bar{v}\|_2 \geq \epsilon > 0$. By restricting $\mathcal{S}$ if necessary, we can assume that $\lim_{k \in \mathcal{S}} v^k = \hat{v}$. By boundedness of $\{x^k\}$, we can further restrict $\mathcal{S}$ to identify a point $\hat{x} \in X$ such that $\lim_{k \in \mathcal{S}} x^k = \hat{x}$. By (15), we thus have

$$A\hat{x} + \lambda(\hat{v} - g) = 0 = \lim_{k \in \mathcal{S}} Ax^k + \lambda(v^k - g) = 0, \tag{16}$$

Since $\hat{x}$ is an accumulation point of the whole sequence, we have by assumption that $\hat{x}$ is a stationary point and hence a solution of (12). By our observation following (14), we thus have that $A\hat{x} + \lambda(\bar{v} - g) = 0$, where $\bar{v}$ is the unique solution of (11). By comparing this expression with (16), we obtain the contradiction $\hat{v} = \bar{v}$, proving the result.

1.4 Previous Algorithms

We briefly review here some of the many algorithms that have been proposed for solving the primal formulation (2) of the ROF model, the dual formulation (6), or both formulations simultaneously. We refer the interested readers to [9] for a more comprehensive survey.

In their original paper [17], ROF proposed a time-marching scheme that solves the associated Euler-Lagrange equation of (2) by seeking the steady-state solution of a parabolic PDE. The method is (asymptotically) slow due to the CFL stability constraints (see [16]), which puts a tight bound on the time step when the solution develop flat regions (where $|\nabla u| \approx 0$). Hence, this scheme is useful in practice only when low-accuracy solutions suffice. Even for an accuracy sufficient to yield a visually satisfactory result, the cost is often too great.

In [19], Vogel and Oman proposed to solve the same Euler-Lagrange equation of (2) via fixed-point iteration. Their main idea is to fix the diffusion coefficient $\frac{1}{|\nabla u|}$ in the Euler-Lagrange equation to its value at a previous step, thus obtaining the solution to the nonlinear equation by solving a sequence of linear systems. They prove global convergence and show that their method is asymptotically much faster than the explicit time-marching scheme.

Chan, Golub, and Mulet [8] (CGM) use Newton's method to solve a smoothed version of the primal-dual system for the ROF model, in which the gradient norm $|\nabla u|$ is replaced by a smoothed approximation $|\nabla u|_\beta = \sqrt{|\nabla u|^2 + \beta}$, for some smoothing parameter $\beta > 0$. Since this approach is based on Newton's method, it converges quadratically, but to a solution of the smoothed approximate model rather than to the solution of (2). Smaller values of β yield approximate solutions that are closer to the true solution, but more iterations are required before the onset of asymptotic quadratic convergence. The cost per iteration is similar to that of the fixed-point iteration scheme.

Hintermüller and Stadler [14] (HS) discuss an infeasible-interior-point method for a modification of (2) in which an term $\mu \int_\Omega |\nabla u|^2 \, dx$ is added, for some small but positive μ. By perturbing the dual of their problem with a regularization term, then applying a semismooth Newton method to the primal-dual

formulation of the resulting problem, they obtain superlinear convergence to a solution of the modified problem. In the implementation, the linear system at each iteration is symmetrized and solved with iterative approaches, such as preconditioned conjugate gradient. The overall approach is related to the CGM method of [8] in that both methods use a Newton-like scheme to solve a perturbation of the primal-dual optimality conditions. One difference is that the HS method does not require the dual variable w to satisfy the constraints $|w| \leq 1$ strictly at every iteration, thus allowing longer steps along the Newton direction to be taken at some iterations.

Goldfarb and Yin [13] reformulate the original ROF model (1) as a second-order cone program (SOCP) and solve it with a standard primal-dual interior-point software package (MOSEK). They replace MOSEK's default ordering of the variables with a nested dissection reordering strategy that is better tuned to their formulation. In contrast to the method of [8], the SOCP formulation converges to the true solution of the underlying ROF model (1).

There are also algorithms that tackle the dual formulation (6) explicitly. Chambolle's method [5] is the best known of this type. He invented the dual semi-implicit gradient descent algorithm based on an original observation he made concerning associated Lagrange multipliers. The method is globally convergent, with suitable restriction on the time step, and is much faster than the primal time-marching scheme.

1.5 Motivations and Proposed Approaches

Most existing numerical algorithms to solve ROF models (2) or (6) can be loosely divided into two categories: those that need to solve a linear system of equations at each iteration (implicit) and those that require only a matrix-vector multiplication in the discrete setting (explicit). Generally speaking, the implicit methods (e.g. CGM, HS, and SOCP) have fast asymptotic convergence rates and can provide highly accurate benchmark solutions. However, explicit methods are preferred in many situations for their simplicity and their convergence with relatively little computational effort to medium-accurate and visually satisfactory results. Their low memory requirements make them even more attractive for large-scale problems. To illustrate the high memory requirements of implicit schemes, we note that an image of size 512×512 is close to the limit of what the SOCP solver MOSEK can handle on a workstation with 2GB of memory.

In the remainder of this paper, we report on the development, implementation, and testing of some simple but fast explicit algorithms. These algorithms are based on the the dual formulation (6) so they do not require any numerical smoothing parameters that would prevent them from converging to the true optimizer. Our proposed approaches are for the most part gradient projection algorithms applied to (6), in which the search path from each iterate is obtained by projecting negative-gradient (steepest descent) directions onto the feasible set. Various enhancements involving different step-length rules and

different line-search strategies are important in making the method efficient. We also propose a sequential quadratic programming approach in which the curvature of the boundary of the feasible set is taken into account.

For the general problem of optimizing a smooth function over a closed convex set, that is,

$$\min_{x \in X} F(x) \tag{17}$$

(where $F : \mathbb{R}^m \to \mathbb{R}$ is smooth and X is a closed convex subset of $\mathbb{R}^m$), gradient projection methods set

$$x^{k+1} = x^k + \gamma_k (x^k(\alpha_k) - x^k), \tag{18}$$

for some parameter $\gamma_k \in [0, 1]$, where

$$x^k(\alpha_k) := P_X(x^k - \alpha_k \nabla F(x^k)), \tag{19}$$

for some $\alpha_k > 0$. Here, P_X denotes the projection onto the set X. Since X is closed and convex, the operator P_X is uniquely defined, but in order for the gradient projection approach to make practical sense, this operator must also be easy to compute. For this reason, gradient projection approaches have been applied most often to problems with separable constraints, where X can be expressed as a Cartesian product of low-dimensional sets. In our case (6), X is a cross product of unit balls in $\mathbb{R}^2$, so computation of P_X requires only $O(N)$ operations. Bertsekas [2] gives extensive background on gradient projection algorithms.

2 Gradient Projection Algorithms

From now on, we will focus on the solution of problem (12), which we restate here:

$$\min_{x \in X} F(x) := \frac{1}{2} \|Ax - \lambda g\|_2^2, \tag{20}$$

where the compact set $X \subset \mathbb{R}^{2N}$ is defined in (12). In this section, we discuss GP techniques for solving this problem. Our approaches move from iterate x^k to the next iterate x^{k+1} using the scheme (18)-(19). Projection P_X on the set X, a Cartesian product of unit Euclidean balls, can be computed straightforwardly as follows.

$$\Big(P_X(x)\Big)_l = \frac{x_l}{\max\{\|x_l\|,\ 1\}}, \qquad l = 1, 2, \ldots, N. \tag{21}$$

This operation projects each 2×1 subvector of x separately onto the unit ball in $\mathbb{R}^2$. It is worth pointing out here that this structure of the dual constraints, which makes the gradient projection approach practical, also enables Chambolle to develop an analytical formula for the Lagrange multipliers in [5].

Our approaches below differ in their rules for choosing the step parameters α_k and γ_k in (18) and (19).

2.1 Three Frameworks

We next consider three gradient projection frameworks that encompass our gradient projection algorithms, and present convergence results for methods in these frameworks.

Framework GP-NoLS chooses α_k in some predetermined range and sets $\gamma_k \equiv 1$.

Framework GP-NoLS

Step 0. *Initialization.* Choose parameters $\alpha_{\min}$, $\alpha_{\max}$ with $0 < \alpha_{\min} < \alpha_{\max}$. Choose x^0 and set $k \leftarrow 0$.
Step 1. Choose steplength $\alpha_k \in [\alpha_{\min},\ \alpha_{\max}]$.
Step 2. Set $x^{k+1} = x^k(\alpha_k)$.
Step 3. Terminate if a stopping criterion is satisfied; otherwise set $k \leftarrow k+1$ and go to Step 1.

Framework GP-ProjArc also sets $\gamma_k \equiv 1$, but chooses α_k by a backtracking line search to satisfy an Armijo criterion, which enforces monotonic decrease of F. This approach is referred to by Bertsekas [2, p. 236] as *Armijo Rule Along the Projection Arc.*

Framework GP-ProjArc

Step 0. *Initialization.* Choose parameters $\alpha_{\min}$, $\alpha_{\max}$ with $0 < \alpha_{\min} < \alpha_{\max}$, and choose $\rho \in (0,1)$ and $\mu \in (0, \frac{1}{2})$. Choose x^0 and set $k \leftarrow 0$.
Step 1. Choose initial steplength $\bar{\alpha}_k \in [\alpha_{\min},\ \alpha_{\max}]$.
Step 2. *Backtracking Line Search.* Choose m to be the smallest nonnegative integer such that

$$F(x^k(\rho^m \bar{\alpha}_k)) \leq F(x^k) - \mu \nabla F(x^k)^T (x^k - x^k(\rho^m \bar{\alpha}_k)),$$

where $x^k(\alpha)$ is defined as in (19);
Set $\alpha_k = \rho^m \bar{\alpha}_k$ and $x^{k+1} = x^k(\alpha_k)$.
Step 3. Terminate if a stopping criterion is satisfied; otherwise set $k \leftarrow k+1$ and go to Step 1.

Framework GP-LimMin fixes α_k at the start of each iteration (possibly using information gathered on previous steps), but then performs a "limited minimization" procedure to find γ_k, again ensuring decrease of F at every step.

Framework GP-LimMin

Step 0. *Initialization.* Choose parameters $\alpha_{\min}$, $\alpha_{\max}$ with $0 < \alpha_{\min} < \alpha_{\max}$. Choose x^0 and set $k \leftarrow 0$.

Step 1. Choose steplength $\alpha_k \in [\alpha_{\min},\ \alpha_{\max}]$. Compute $x^k(\alpha_k)$ and set $\delta^k := (x^k(\alpha_k) - x^k)$.

Step 2. *Limited Minimizing Line Search.* Set $x^{k+1} = x^k + \gamma_k \delta^k$, with $\gamma_k = \text{mid}(0,\ \gamma_{k,\text{opt}},\ 1)$ and

$$\gamma_{k,\text{opt}} = \arg\min\ F(x^k + \gamma\delta^k) = \frac{-(\delta^k)^T \nabla F(x^k)}{\|A\delta^k\|_2^2} \tag{22}$$

Step 3. Terminate if a stopping criterion is satisfied; otherwise set $k \leftarrow k + 1$ and go to Step 1.

The first algorithm we consider — Algorithm GPCL — is obtained from Framework GP-NoLS by setting α_k equal to the fixed value $\alpha > 0$ at every step. Convergence is obtained for all α sufficiently small, as we now show.

Theorem 1 *Let $\{x^k\}$ be a sequence generated by Algorithm GPCL. Then if $0 < \alpha < .25$, the sequence v^k obtained from (15) converges to the unique solution $\bar{v}$ of (11).*

Proof Given any two vectors x' and x'' we have that

$$\nabla F(x') - \nabla F(x'') = A^T A(x' - x''),$$

so the Lipschitz constant for ∇F is $\|A^T A\|_2$, which is bounded by 8 (see [5, p. 92]). It follows immediately from [2, Proposition 2.3.2] that every accumulation point of $\{x^k\}$ is stationary for (20) provided that $0 < \alpha < .25$. The result now follows immediately from Proposition 1.

The upper bound of .25 in Theorem 1 is tight; we observe in practice that the method is unstable evn for $\tau = .251$.

We consider other algorithms in Framework GP-NoLS below, in which α_k takes on different values at each iteration that may violate the bound of .25. These methods may be non-monotone (the function F may increase on some iterations) and convergence results cannot be proven in general without the addition of step acceptance criteria, such as the requiring a significant improvement over the worst function value at the last M points visited for some positive integer M; see [3]. Strategies are also required for modifying steps that fail these criteria.

For algorithms in Framework GP-ProjArc , we have the following convergence result.

Theorem 2 *Let $\{x^k\}$ be a sequence generated by an algorithm in Framework GP-ProjArc . Then the sequence v^k obtained from (15) converges to the unique solution $\bar{v}$ of (11).*

Proof Proposition 2.3.3 of Bertsekas [2], with minor modifications for the variable choice of $\bar{\alpha}_k$ within the range $[\alpha_{\min}, \alpha_{\max}]$, shows that all limit points of $\{x^k\}$ are stationary. The result then follows from Proposition 1.

An identical result holds for algorithms in Framework GP-LimMin .

Theorem 3 *Let $\{x^k\}$ be a sequence generated by an algorithm in Framework GP-LimMin . Then the sequence v^k obtained from (15) converges to the unique solution $\bar{v}$ of (11).*

Proof Proposition 2.3.1 of Bertsekas [2], with minor modifications for the variable choice of $\bar{\alpha}_k$ within the range $[\alpha_{\min}, \alpha_{\max}]$, shows that all limit points of $\{x^k\}$ are stationary. The result then follows from Proposition 1.

2.2 Barzilai-Borwein Strategies

We discuss strategies that choose α_k using approaches first proposed by Barzilai and Borwein [1] (BB) and subsequently elaborated by other authors. For the problem $\min_{x\in\mathbb{R}^{2N}} F(x)$, the basic BB strategy sets $x^{k+1} \leftarrow x^k - \alpha_k \nabla F(x^k)$, where α_k is chosen so that $\alpha_k^{-1} I$ mimics the behavior of the Hessian $\nabla^2 F$ over the previous step. By Taylor's theorem, we have

$$\nabla^2 F(x^k)\Delta x^{k-1} \approx \Delta g^{k-1}, \qquad \Delta x^{k-1} \approx (\nabla^2 F(x^k))^{-1}\Delta g^{k-1},$$

where

$$\Delta x^{k-1} := x^k - x^{k-1}, \qquad \Delta g^{k-1} := \nabla F(x^k) - \nabla F(x^{k-1}),$$

so our desired property on α is that $\alpha^{-1}\Delta x^{k-1} \approx \Delta g^{k-1}$. Note that for the F we consider here (20), we have $\Delta g^{k-1} = A^T A \Delta x^{k-1}$.

One formula for α is obtained by performing a least-squares fit in one variable, as follows:

$$\alpha_{k,1} = \left[\arg\min_{\tau\in\mathbb{R}} \|\tau\Delta x^{k-1} - \Delta g^{k-1}\|_2^2 \right]^{-1},$$

which yields

$$\alpha_{k,1} = \frac{\|\Delta x^{k-1}\|_2^2}{\langle \Delta x^{k-1}, \Delta g^{k-1}\rangle} = \frac{\|\Delta x^{k-1}\|_2^2}{\|A\Delta x^{k-1}\|_2^2}. \tag{23}$$

An alternative formula is obtained similarly, by doing a least-squares fit to α rather than α^{-1}, to obtain

$$\alpha_{k,2} = \arg\min_{\alpha\in\mathbb{R}} \|\Delta x^{k-1} - \alpha\Delta g^{k-1}\|_2^2 = \frac{\langle \Delta x^{k-1}, \Delta g^{k-1}\rangle}{\|\Delta g^{k-1}\|_2^2} = \frac{\|A\Delta x^{k-1}\|_2^2}{\|A^T A\Delta x^{k-1}\|_2^2}. \tag{24}$$

These step lengths were shown in [1] to be effective on simple problems; a partial analysis explaining the behavior was given. Numerous variants have been proposed recently, and subject to with theoretical and computational

evaluation. The BB stepsize rules have also been extended to constrained optimization, particularly to bound-constrained quadratic programming; see, for example [10] and [18]. The same formulae (23) and (24) can be used in these cases.

Other variants of Barzilai-Borwein schemes have been proposed by other authors in other contexts. The cyclic Barzilai-Borwein (CBB) method proves to have better performance than the standard BB in many cases (see for example [11] and the references therein). In this approach, we recalculate the BB stepsize from one of the formulae (23) or (24) at only every mth iteration, for some integer m. At intervening steps, we simply use the last calculated value of α_k. There are alternating Barzilai-Borwein (ABB) schemes that switch between the definitions (23) and (24), either adaptively or by following a fixed schedule.

2.3 Implemented Variants of Gradient Projection

We discuss here the variants of gradient projection that were implemented in our computational testing.

Algorithm GPLS. This algorithm falls into Framework GP-ProjArc , where we choose the initial steplength $\bar{\alpha}_k$ at each iteration by predicting what the steplength would be if no new constraints were to become active on this step. Specifically, we define the vector g^k by

$$g_i^k = \begin{cases} (\nabla F(x^k))_l, & \text{if } \|x_l^k\|_2 < 1 \text{ or } (\nabla F(x^k))_l^T x_l^k > 0, \\ \left[I - x_l^k (x_l^k)^T\right] (\nabla F(x^k))_l, & \text{otherwise.} \end{cases}$$

We then choose the initial guess to be

$$\bar{\alpha}_k = \arg\min_{\alpha} F(x^k - \alpha g^k),$$

which can be computed explicitly as

$$\bar{\alpha}_k = \frac{(g^k)^T \nabla F(x^k)}{\|Ag^k\|_2^2} = \frac{\|g^k\|_2^2}{\|Ag^k\|_2^2}.$$

In practice, we find that using $\frac{1}{2}\bar{\alpha}_k$ as the initial value gives better performance, and backtracking is not necessary in any of our numerical experiments.

Algorithm GPBB-NM. This is a nonmonotone Barzilai-Borwein method in Framework GP-NoLS , in which we obtain the step α_k via the formula (23), projected if necessary onto the interval $[\alpha_{\min}, \alpha_{\max}]$.

Algorithm GPBB-NM(m). A nonmonotone cyclic Barzilai-Borwein algorithm in Framework GP-NoLS , in which α_k is recalculated from (23) at every mth iteration. Formally, we set

$$\alpha_{ml+i} = \alpha_{ml+1}^{BB} \quad \text{for } l = 0, 1, 2, \ldots \text{ and } i = 1, 2, \ldots, m-1,$$

where α_{ml+1}^{BB} is obtained from (23) with $k = ml + 1$, restricted to the interval $[\alpha_{\min}, \alpha_{\max}]$.

Algorithm GPBB-M. A monotone Barzilai-Borwein method in Framework GP-LimMin , in which α_k is obtained as in Algorithm GPBB-NM.

Algorithm GPBB-M(m). A monotone cyclic Barzilai-Borwein algorithm in Framework GP-LimMin , in which α_k is recalculated from (23) at every mth iteration, similarly to Algorithm GPBB-NM(m).

Algorithm GPABB. A monotonic alternating Barzilai-Borwein method in Framework GP-LimMin , in which the technique of Serafini, Zanghirati, and Zanni [18, Section 2.2], is used to switch between the rules (23) and (24). This technique makes use of two positive integer parameters $n_{\min}$ and $n_{\max}$ with $0 < n_{\min} \leq n_{\max}$. Let n_α be the number of consecutive iterations that use the same steplength selection rule, (23) or (24). We switch from one rule to the other at the next iteration $k + 1$ if either (i) $n_\alpha \geq n_{\max}$ or (ii) $n_\alpha \geq n_{\min}$ and α_k is either a *separating steplength* or a *bad descent generator*. The current step α_k is a *separating steplength* if it lies between the values generated by the two rules at the next iteration, that is, $\alpha_{k+1,2} < \alpha_k < \alpha_{k+1,1}$. Given two constants γ_l and γ_u with $0 < \gamma_l \leq 1 \leq \gamma_u$, we say that α_k is a *bad descent generator* if one of the following conditions holds:

(a) $\gamma_{k,\text{opt}} < \gamma_l$ and $\alpha_k = \alpha_{k,1}$; or
(b) $\gamma_{k,\text{opt}} > \gamma_u$ and $\alpha_k = \alpha_{k,2}$.

where $\gamma_{k,\text{opt}}$ is obtained from the limited minimization rule (22). We refer interested readers to [18] for the rationale of the criterion. In any case, the chosen α_k is adjusted to ensure that it lies in the interval $[\alpha_{\min}, \alpha_{\max}]$.

3 A Sequential Quadratic Programming Algorithm

We describe here a variation on the techniques of the previous section in which the curvature of the boundary of the constraint set X is accounted for in computing the search direction. The method can be viewed as a sequential quadratic programming (SQP) method applied to the dual formulation (20). The KKT optimality conditions for this formulation can be written as follows:

$$\begin{aligned} A_l^T(Ax - \lambda g) + 2z_l x_l = 0, \qquad & l = 1, 2, \ldots, N, \\ 0 \leq z_l \ \perp \ \|x_l\|^2 - 1 \leq 0, \qquad & l = 1, 2, \ldots, N, \end{aligned}$$

where the scalars z_l are Lagrange multipliers for the constraints $\|x_l\|_2^2 \leq 1$, $l = 1, 2, \ldots, N$, and the operator $\perp$ indicates that at least one of its two operands must be zero. At iteration k, we compute an estimate of the active set $\mathcal{A}_k \subset \{1, 2, \ldots, N\}$, which are those indices for which we believe that $\|x_l\|_2^2 = 1$ at the solution. In our implementation, we choose this set as follows:

$$\begin{aligned}\mathcal{A}_k &= \{l \,|\, \|x_l^k\|_2 = 1 \;\text{ and }\; (x_l^k)^T[\nabla F(x^k)]_l \leq 0\} \\ &= \{l \,|\, \|x_l^k\|_2 = 1 \;\text{ and }\; (x_l^k)^T A_l^T(Ax^k - \lambda g) \leq 0\}. \end{aligned} \tag{25}$$

The SQP step is a Newton-like step for the following system of nonlinear equations, from the current estimates x^k and z_l^k, $l = 1, 2, \ldots, N$:

$$A_l^T(Ax - \lambda g) + 2x_l z_l = 0, \qquad l = 1, 2, \ldots, N, \tag{26a}$$

$$\|x_l\|_2^2 - 1 = 0, \qquad l \in \mathcal{A}_k, \tag{26b}$$

$$z_l = 0, \qquad l \notin \mathcal{A}_k. \tag{26c}$$

Using $\tilde{z}_l^{k+1}$ to denote the values of z_l at the next iterate, and $\tilde{d}^k$ to denote the step in x^k, a "second-order" step can be obtained from (26) by solving the following system for $\tilde{d}^k$ and $\tilde{z}_l^{k+1}$, $l = 1, 2, \ldots, N$:

$$A_l^T A\tilde{d}^k + 2\tilde{z}_l^{k+1}\tilde{d}_l^k = -A_l^T[Ax^k - \lambda g] - 2x_l^k \tilde{z}_l^{k+1}, \;\; l = 1, 2, \ldots, N, \tag{27a}$$

$$2(x_l^k)^T\tilde{d}_l^k = 0, \;\; l \in \mathcal{A}_k, \tag{27b}$$

$$\tilde{z}_l^{k+1} = 0, \;\; l \notin \mathcal{A}_k. \tag{27c}$$

We now define Newton-like steps d^k in x, and new iterates z^{k+1} in z, by replacing A^TA by $\alpha_k^{-1}I$ in (27a) and solving the following linear system:

$$\alpha_k^{-1}d_l^k + 2z_l^{k+1}d_l^k = -A_l^T[Ax^k - \lambda g] - 2x_l^k z_l^{k+1}, \;\; l = 1, 2, \ldots, N, \tag{28a}$$

$$2(x_l^k)^T d_l^k = 0, \;\; l \in \mathcal{A}_k, \tag{28b}$$

$$z_l^{k+1} = 0, \;\; l \notin \mathcal{A}_k. \tag{28c}$$

Considering indices $l \in \mathcal{A}_k$, we take the inner product of (28a) with x_l^k and use (28b) and (25) to obtain:

$$z_l^{k+1} = -(1/2)(x_l^k)^T A_l^T(Ax^k - \lambda g), \;\; l \in \mathcal{A}_k.$$

We obtain the steps d_l^k for these indices by substituting this expression in (28a):

$$d_l^k = -(\alpha_k^{-1} + 2z_l^{k+1})^{-1}\left[A_l^T(Ax^k - \lambda g) + 2x_l^k z_l^{k+1}\right], \;\; l \in \mathcal{A}_k.$$

In fact, because of (28c), this same formula holds for $l \notin \mathcal{A}_k$, when it reduces to the usual negative-gradient step

$$d_l^k = -\alpha_k A_l^T(Ax^k - \lambda g), \;\; l \notin \mathcal{A}_k.$$

We define the (nonmonotone) Algorithm SQPBB-NM by making an initial choice of α_k at each iteration according to the formula (23), and calculating

$x^{k+1} = x^k + d^k$ and z^{k+1} as described above. In the monotone variant of this method, known as SQPBB-M, we successively decrease α_k by a factor of $\rho \in (0, 1)$, as in Framework GP-ProjArc , and recalculate x^{k+1} and z^{k+1} as above, until a decrease in the objective function is obtained.

We also tried versions of these methods in which α_k was recalculated only on every mth iteration; these are referred to as SQPBB-NM(m) and SQPBB-M(m), respectively.

4 Termination

The decision about when an approximate solution is of sufficiently high quality to terminate the algorithm can be difficult for general constrained optimization problems. Often, we wish the approximate solution x to be close to a global minimizer x^* and/or the function value $F(x)$ be close to $F(x^*)$. In the case of (20), the duality gap (7) provides a reliable and easily calculated stopping criterion.

If (u, w) be a feasible primal-dual pair satisfying (4), we have from (7) that

$$G(u, w) = \int_\Omega \Big(|\nabla u| - \nabla u \cdot w \Big). \tag{29}$$

We terminate the algoithm when the current iterate $w = w^k$ satisfies the following stopping criterion:

$$\frac{G(u, w)}{D(w)} \leq \text{TOL}, \tag{30}$$

where u is obtained from (4) and TOL is a small positive tolerance. It follows from (8b) that the dual objective $D(w)$ is close to the optimal objective O^* when (30) is satisfied, in the sense that

$$0 \leq O^* - D(w) \leq (\text{TOL}) D(w).$$

We can show that the u obtained from (4) is also close to the optimal value u^* when this test is satisfied, by the following argument. From (4), we have

$$u - u^* = (f + \frac{1}{\lambda} \nabla \cdot w) - (f + \frac{1}{\lambda} \nabla \cdot w^*) = \frac{1}{\lambda} (\nabla \cdot w - \nabla \cdot w^*)$$

Since $|\nabla u^*| = \nabla u^* \cdot w^*$ and $|\nabla u| \geq \nabla u \cdot w$ for any feasible w (since $|w| \leq 1$), we have

$$\begin{aligned}
\lambda \|u - u^*\|_2^2 &= \int_\Omega (u - u^*)(\nabla \cdot w - \nabla \cdot w^*) \\
&= \int_\Omega u \nabla \cdot w - \int_\Omega u \nabla \cdot w^* - \int_\Omega u^* \nabla \cdot w + \int_\Omega u^* \nabla \cdot w^* \\
&= \int_\Omega -\nabla u \cdot w + \int_\Omega \nabla u \cdot w^* + \int_\Omega \nabla u^* \cdot w - \int_\Omega |\nabla u^*| \\
&\leq \int_\Omega -\nabla u \cdot w + \int_\Omega |\nabla u| + \int_\Omega |\nabla u^*|(|w| - 1) \\
&\leq \int_\Omega (|\nabla u| - \nabla u \cdot w) \\
&= G(u, w). \qquad (31)
\end{aligned}$$

Using this bound, we obtain the following bound when (u, w) satisfies (30):

$$\|u - u^*\|_2 \leq \sqrt{G(u, w)/\lambda} \leq \sqrt{D(w)(\text{TOL})/\lambda}.$$

5 Computational Experiments

We report on computational experiments for three test problems in image denoising. The original clean images and the noisy images used as input to the denoising codes are shown in Figures 1 and 2, respectively. The sizes of the discretizations for the three test problems are 128×128, 256×256, and 512×512, respectively. The noisy images are generated by adding Gaussian noise to the clean images using the MATLAB function `imnoise`, with variance parameter set to 0.01. The fidelity parameter λ is taken to be 0.045 throughout the experiments. This parameter is inversely related to the noise level σ and usually needs to be tuned for each individual image to get an optimal visual result.

We tested the following algorithms:

- Chambolle's semi-implicit gradient descent method [5];
- many variants of gradient projection proposed in Section 2;
- the SQP method of Section 3;
- the CGM method of [8].

We report on a subset of these tests here, including the gradient projection variants that gave consistently good results across the three test problems.

In Chambolle's method, we take the step to be 0.248 for near optimal performance, although global convergence is proved in [5] only for steps in the range $(0, .125)$. We use the same value $\alpha_k = 0.248$ in Algorithm GPCL, as it appears to be near optimal in this case as well.

For all gradient projection variants, we set $\alpha_{\min} = 10^{-5}$ and $\alpha_{\max} = 10^5$. (Performances are insensitive to these choices, as long as $\alpha_{\min}$ is sufficiently

Fig. 1 The original clean images for our test problems. Left: 128×128 "shape"; middle: 256×256 "cameraman"; right: 512×512 "Barbara".

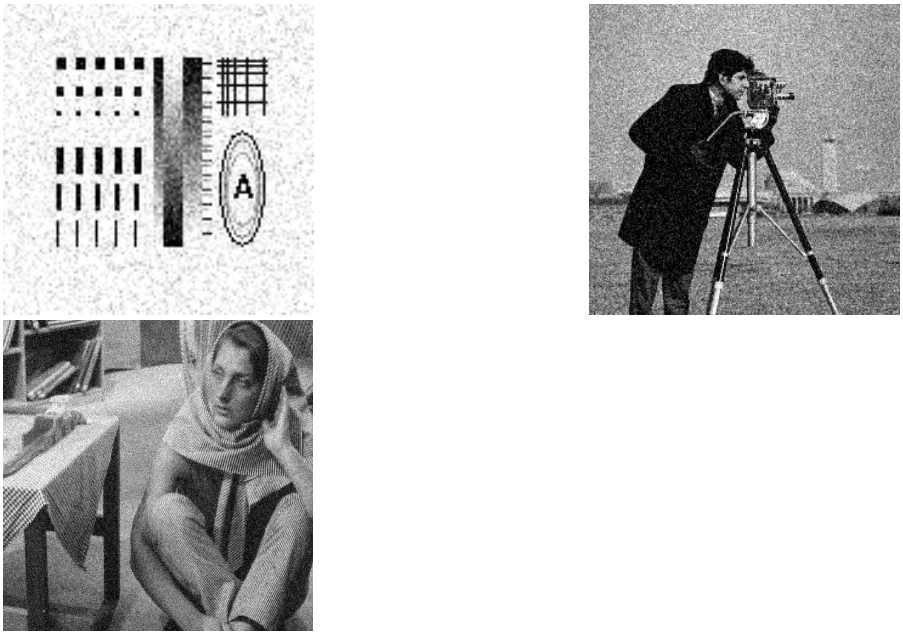

Fig. 2 The input noisy images for our test problems. Gaussian noise is added using MATLAB function `imnoise` with variance 0.01.

small and $\alpha_{\max}$ sufficiently large.) In Algorithm GPLS, we used $\rho = 0.5$ and $\mu = 10^{-4}$. In Algorithm GPABB, we set $\gamma_l = 0.1$ and $\gamma_u = 5$.

We also tried variants of the GPBB methods in which the initial choice of α_k was scaled by a factor of 0.5 at every iteration. We found that this variant often enhanced performance. This fact is not too surprising, as we can see from Section 3 that the curvature of the boundary of constraint set X suggests that it is appropriate to add positive diagonal elements to the Hessian approximation, which corresponds to decreasing the value of α_k.

In the CGM implementation, we used a direct solver for the linear system at each iteration, as the conjugate gradient iterative solver (which is an option in the CGM code) was slower on these examples. The smooth parameter β is dynamically updated based on duality gap from iteration to iteration. In particular, we take $\beta_0 = 100$ and let $\beta_k = \beta_{k-1} (G_k/G_{k-1})^2$, where G_k and G_{k-1} are the duality gaps for the past two iterations. This simple strategy for updating β, which is borrowed from interior-point methods, outperforms the classical CGM approach, producing faster decrease in the duality gap.

All methods are coded in MATLAB. It is likely the performance can be improved by recoding in C or C++, but we believe that improvements would be fairly uniform across all the algorithms.

Tables 1, 2, and 3 report number of iterations and average CPU times over ten runs, where each run adds a different random noise vector to the true image. In all codes, we used the starting point $x^0 = 0$ in each algorithm and the relative duality gap stopping criterion (30). We vary the threshold TOL from 10^{-2} to 10^{-6}, producing results of increasingly high accuracy as TOL is decreased.

Figure 3 shows the denoised images obtained at different values of TOL. Note that visually there is little difference between the results obtained with two tolerance values 10^{-2} and 10^{-4}. Smaller values of TOL do not produce further visual differences.

The tables show that on all problems, the proposed gradient projection algorithms are competitive to Chambolle's method, and that some variants are significantly faster, especially when moderate accuracy is required for the solutions. Two variants stood out as good performers: the GPBB-NM variant and the GPBB-M(3) variant in which the initial choice of α_k was scaled by 0.5 at each iteration. For all tests with TOL $= 10^{-2}$, TOL $= 10^{-3}$, and TOL $= 10^{-4}$, the winner was one of the gradient-projection Barzilai-Borwein strategies.

For these low-to-moderate accuracy requirements, CGM is generally slower than the gradient-based methods, particularly on the larger problems. The picture changes considerably, however, when high accuracy (TOL $= 10^{-6}$) is required. The rapid asymptotic convergence of CGM is seen to advantage in this situation, and it outperforms the gradient-based methods in all cases. These observations are illustrated in Figure 4, which plots the duality gap again the CPU time cost for Chambolle's method, CGM method and the GPBB-NM variant of the gradient projection algorithm for each of the three test problems.

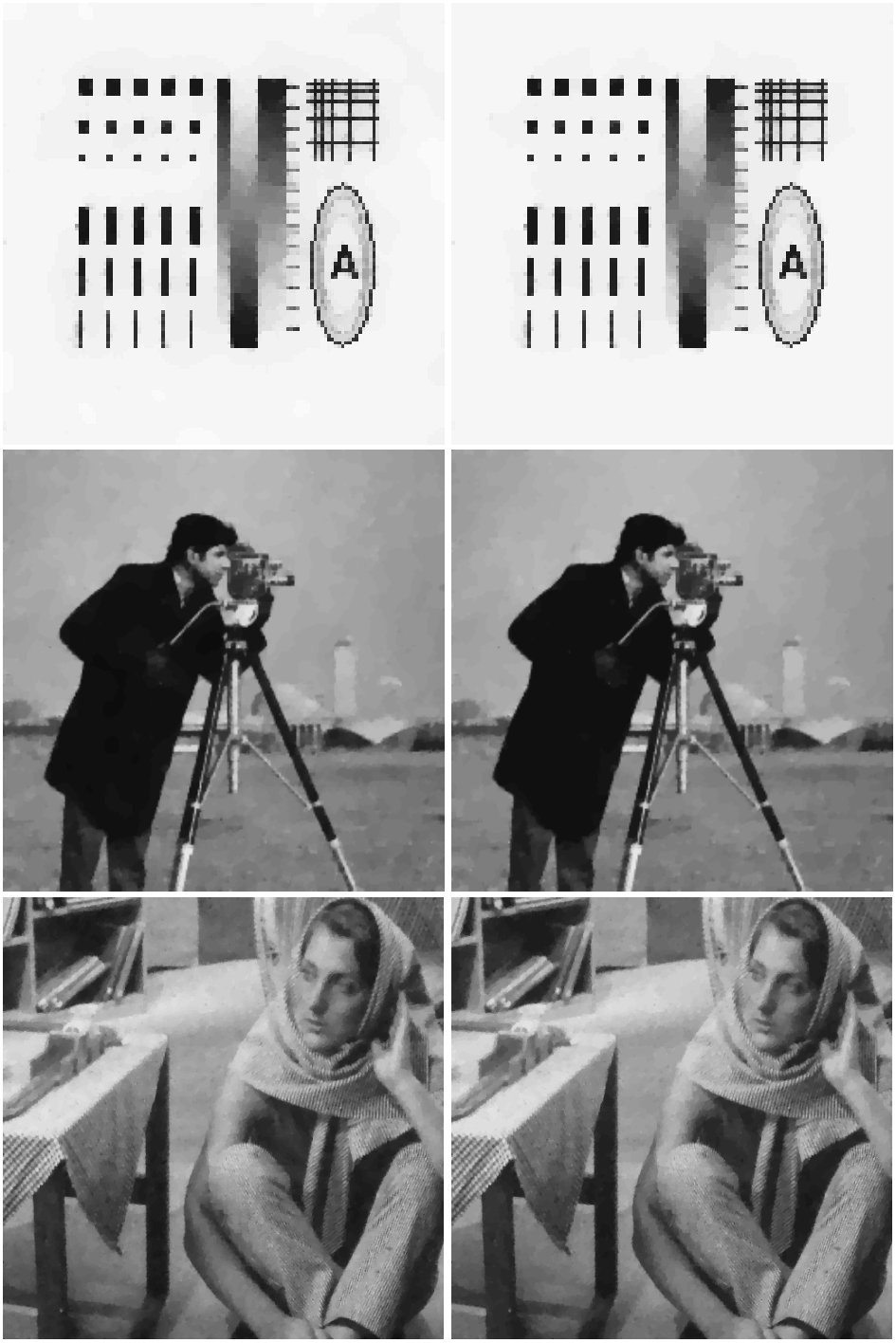

Fig. 3 The denoised images with different level of termination criterions. left column: TOL = 10^{-2}, right column: TOL = 10^{-4}.

Table 1 Number of iterations and CPU times (in seconds) for problem 1. $*$ = initial α_k scaled by 0.5 at each iteration.

	TOL $= 10^{-2}$		TOL $= 10^{-3}$		TOL $= 10^{-4}$		TOL $= 10^{-6}$	
Algorithms	Iter	Time	Iter	Time	Iter	Time	Iter	Time
Chambolle	18	0.12	169	1.15	1082	7.23	23884	154
GPCL	39	0.25	136	0.90	736	4.71	16196	110
GPLS	23	0.33	181	2.66	845	12.8	17056	260
GPBB-M	12	0.16	152	1.82	836	10.1	17464	167
GPBB-M (2)	18	0.17	187	1.80	941	9.00	21146	193
GPBB-M(3)	13	0.11	92	0.78	287	2.44	3749	32.2
GPBB-M(3)*	11	0.09	49	0.41	190	1.60	2298	19.7
GPBB-NM	10	0.09	48	0.46	217	2.10	3857	32.3
GPABB	13	0.16	58	0.66	245	2.80	2355	24.0
SQPBB-M	13	0.17	47	0.66	196	2.81	3983	61.0
CGM	6	4.05	9	5.90	12	8.00	18	12.1

Table 2 Number of iterations and CPU times (in seconds) for problem 2. $*$ = initial α_k scaled by 0.5 at each iteration.

	TOL $= 10^{-2}$		TOL $= 10^{-3}$		TOL $= 10^{-4}$		TOL $= 10^{-6}$	
Algorithms	Iter	Time	Iter	Time	Iter	Time	Iter	Time
Chambolle	27	1.05	164	6.40	815	31.80	15911	628
GPCL	32	1.26	112	4.31	540	21.0	11434	452
GPLS	20	1.85	132	14.8	575	66	12892	1531
GPBB-M	20	1.14	124	7.01	576	32.7	11776	674
GPBB-M (2)	20	1.12	72	4.05	245	13.9	4377	251
GPBB-M(3)	20	1.12	77	4.33	345	19.5	3522	200
GPBB-M(3) fudge	17	0.95	47	2.65	162	9.17	1766	100
GPBB-NM	16	0.85	48	2.53	178	9.52	2802	150
GPABB	16	1.06	47	3.16	168	11.4	1865	127
SQPBB-M	14	1.10	41	3.44	152	13.5	2653	245
CGM	6	22.30	10	37.5	13	48.8	19	71.0

Table 3 Number of iterations and CPU times (in seconds) for problem 3. $*$ = initial α_k scaled by 0.5 at each iteration.

	TOL $= 10^{-2}$		TOL $= 10^{-3}$		TOL $= 10^{-4}$		TOL $= 10^{-6}$	
Algorithms	Iter	Time	Iter	Time	Iter	Time	Iter	Time
Chambolle	27	5.46	131	28.3	534	112	8314	1781
GPCL	24	4.65	80	17.2	328	69.1	5650	1212
GPLS	34	10.9	86	32.5	322	128	5473	2258
GPBB-M	20	5.50	84	24.9	332	98.2	5312	1576
GPBB-M (2)	20	5.40	56	16.3	160	46.6	4408	1290
GPBB-M(3)	20	5.38	67	19.5	174	50.5	2533	738
GPBB-M(3) fudge	17	4.58	41	11.9	131	38.0	1104	321
GPBB-NM	15	3.92	40	11.3	115	32.4	1371	388
GPABB	14	4.57	35	12.3	122	42.8	1118	394
SQPBB-M	14	4.91	40	15.8	109	44.6	1556	646
CGM	7	168	10	216	14	302	21	441

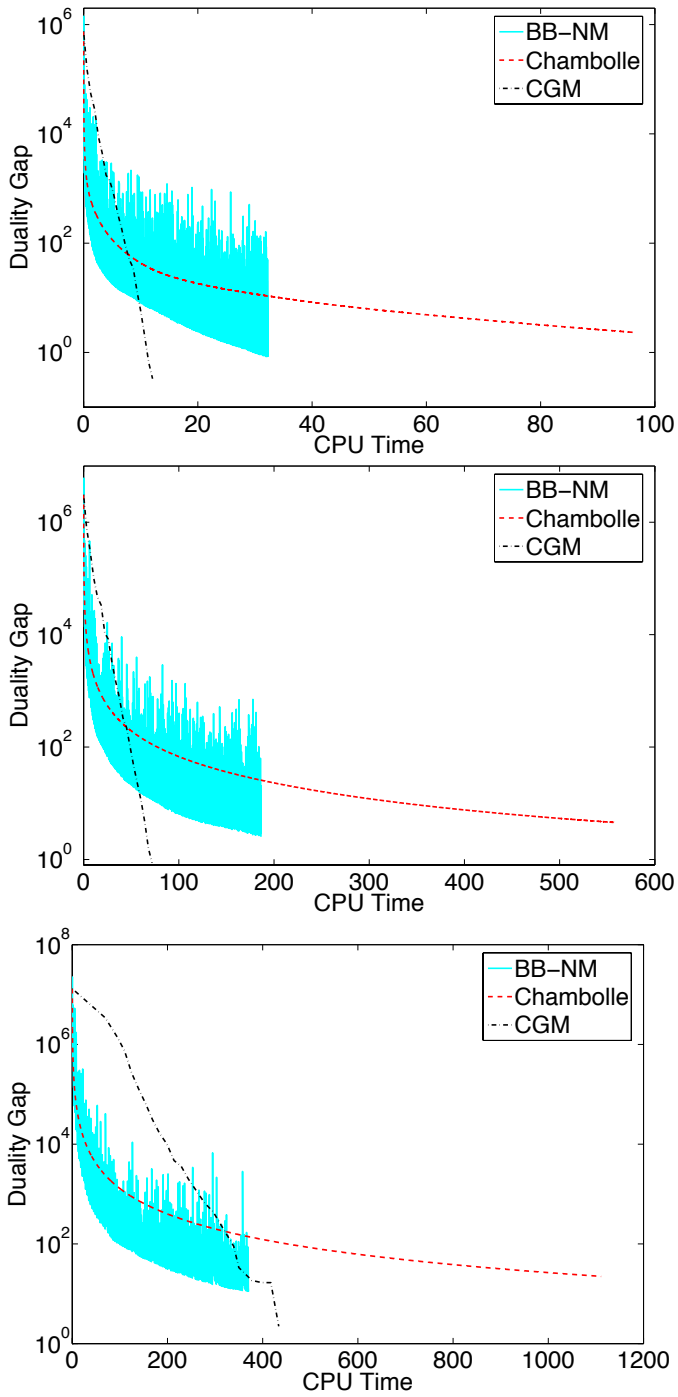

Fig. 4 Duality gap vs. CPU time for GPBB-NM, Chambolle, and CGT codes, for problems Prolems 1, 2, and 3, respectively.

6 Conclusions and Final Remarks

We have proposed gradient projection algorithms for solving the discretized dual formulation of the total variation image restoration model of Rudin, Osher, and Fatemi [17]. The problem has a convex quadratic objective with separable convex constraints, a problem structure that makes gradient projection schemes practical and simple to implement. We tried different variants of gradient projection (including non-monotone Barzilai-Borwein spectral variants) that make use of different steplength strategies. We compare these methods to two popular existing approaches proposed by Chambolle [5] and Chan, Golub, and Mulet [8], and show that when low to moderate solution accuracy is required, several of the gradient projection variants are faster than earlier approaches.

Besides giving evidence of improved strategies for obtaining lower-accuracy solutions to the ROF model, our results also suggest several strategies for obtaining higher-accuracy solutions. First, it would seem appealing to use a GP approach to obtain a low-accuracy solution, then "cross over" to CGM, using the solution obtained from the GP approach to generate a starting point for CGM. However, we found that the primal-dual starting point $(v, x) = (g - Ax/\lambda, x)$ obtained from the GP solution x was not generally better than using a "cold start" for CGM. As in interior-point primal-dual methods, it seems difficult to find good warm starts. Second, we experimented with a multi-level strategy, using computed solutions for a coarser discretization (smaller n and N) to construct a starting point for a finer discretization. We have not yet been able to find an interpolation strategy for which the constructed starting point is better than a cold start.

MATLAB implementations of the algorithms discussed in this paper, along with data sets, are available at `www.cs.wisc.edu/~swright/TVdenoising/`.

References

1. Barzilai, J., Borwein, J.M.: Two-point step size gradient methods. IMA Journal of Numerical Analysis **8**, 141–148 (1988)
2. Bertsekas, D.P.: Nonlinear Programming, second edn. Athena Scientific (1999)
3. Birgin, E.G., Martínez, J.M., Raydan, M.: Nonmonotone spectral projected gradient methods on convex sets. SIAM Journal on Optimization **10**(4), 1196–1211 (2000)
4. Carter, J.L.: Dual method for total variation-based image restoration. Report 02-13, UCLA CAM (2002)
5. Chambolle, A.: An algorithm for total variation minimization and applications. Journal of Mathematical Imaging and Visualization **20**, 89–97 (2004)
6. Chambolle, A., Lions, P.L.: Image recovery via total variation minimization and related problmes. Numerische Mathematik **76**, 167–188 (1997)
7. Chan, T.F., Esedoglu, S., Park, F., Yip, A.: Total variation image restoration: Overview and recent developments. In: N. Paragios, Y. Chen, O. Faugeras (eds.) Handbook of Mathematical Models in Computer Vision. Springer (2005)
8. Chan, T.F., Golub, G.H., Mulet, P.: A nonlinear primal-dual method for total variation based image restoration. SIAM Journal of Scientific Computing **20**, 1964–1977 (1999)
9. Chan, T.F., Zhu, M.: Fast algorithms for total variation-based image processing. In: Proceedings of the 4th ICCM. Hangzhuo, China (2007)

10. Dai, Y.H., Fletcher, R.: Projected barzilai-borwein methods for large-scale box-constrained quadratic programming. Numerische Mathematik **100**, 21–47 (2005)
11. Dai, Y.H., Hager, W.W., Schittkowski, K., Zhang, H.: The cyclic Barzilai-Borwein method for unconstrained optimization. IMA Journal of Numerical Analysis **26**, 604–627 (2006)
12. Ekeland, I., Témam: Convex Analysis and Variational Problems. SIAM Classics in Applied Mathematics. SIAM (1999)
13. Goldfarb, D., Yin, W.: Second-order cone programming methods for total variation-based image restoration. SIAM Journal on Scientific Computing **27**, 622–645 (2005)
14. Hintermüller, M., Stadler, G.: An infeasible primal-dual algorithm for TV-based inf-convolution-type image restoration. SIAM Journal on Scientific Computing **28**, 1–23 (2006)
15. Hiriart-Urruty, J., Lemaréchal, C.: Convex Analysis and Minimization Algorithms, vol. I. Springer-Verlag, Berlin (1993)
16. Osher, S., Marquina, A.: Explicit algorithms for a new time dependent model based on level set motion for nonlinear deblurring and noise removal. SIAM Journal of Scientific Computing **22**, 387–405 (2000)
17. Rudin, L., Osher, S., Fatemi, E.: Nonlinear total variation based noise removal algorithms. Physica D **60**, 259–268 (1992)
18. Serafini, T., Zanghirati, G., Zanni, L.: Gradient projection methods for large quadratic programs and applications in training support vector machines. Optimization Methods and Software **20**(2–3), 353–378 (2004)
19. Vogel, C.R., Oman, M.E.: Iterative methods for total variation denoising. SIAM Journal of Scientific Computing **17**, 227–238 (1996)