A unified approach for a 1D generalized total variation problem

Cheng Lu · Dorit S. Hochbaum

Received: 18 July 2019 / Accepted: 17 February 2021
© The Author(s) 2021

Abstract
We study a 1-dimensional discrete signal denoising problem that consists of mini-
mizing a sum of separable convex fidelity terms and convex regularization terms, the
latter penalize the differences of adjacent signal values. This problem generalizes the
total variation regularization problem. We provide here a unified approach to solve
the problem for general convex fidelity and regularization functions that is based on
the Karush–Kuhn–Tucker optimality conditions. This approach is shown here to lead
to a fast algorithm for the problem with general convex fidelity and regularization
functions, and a faster algorithm if, in addition, the fidelity functions are differentiable
and the regularization functions are strictly convex. Both algorithms achieve the best
theoretical worst case complexity over existing algorithms for the classes of objective
functions studied here. Also in practice, our C++ implementation of the method is
considerably faster than popular C++ nonlinear optimization solvers for the problem.

Mathematics Subject Classification 90-08 · 90-10 · 90C25 · 90C30 · 90C46 · 49M05 · 65K05

1 Introduction

The problem addressed here is the 1-dimensional discrete signal denoising problem
which is formulated as:

\[(1\text{-D-Denoise}) \min_{x_1, \ldots, x_n} \sum_{i=1}^{n} f_i(x_i) + \sum_{i=1}^{n-1} h_i(x_i - x_{i+1}). \]  

Dorit S. Hochbaum: This author’s research was supported in part by NSF award No. CMMI-1760102.

Cheng Lu
chengu@berkeley.edu

Dorit S. Hochbaum
hochbaum@ieor.berkeley.edu

1 Department of Industrial Engineering and Operations Research, University of California, Berkeley, USA

Published online: 13 March 2021
Here the $x_i$’s are the denoised values over a 1-dimensional discrete signal. Each $f_i(x_i)$ is a convex data fidelity function that penalizes the difference between the estimated signal value $x_i$ and an observed value. Each $h_i(x_i - x_{i+1})$ is a convex regularization function that penalizes the difference between neighboring estimated signal values, $x_i$ and $x_{i+1}$. To guarantee bounded minimum for the problem, it is assumed that all variable values are within an interval $[-U, U]$. Note that boundedness is an unavoidable requirement for solving general nonlinear optimization problems [12].

We present here a unified approach for solving 1D-Denoise problem (1) for two broad classes of objective functions. In the first class, the fidelity functions $f_i(x_i)$ are convex differentiable and the regularization functions $h_i(x_i - x_{i+1})$ are strictly convex. We name this class as the differentiable-strictly-convex class, or dsc-class for short. The second class consists of both fidelity and regularization functions general convex, and it is named here the convex-convex class, or cc-class for short. Note that the second class is more general and contains the first class. Our unified approach, which is based on the KKT (Karush–Kuhn–Tucker) optimality conditions, results in an $O(n \log^2 \frac{U}{\epsilon})$ algorithm for the dsc class, the dsc-algorithm, and an $O(n^2 \log^2 \frac{U}{\epsilon})$ algorithm for the cc class, the cc-algorithm. In the complexity expressions, $\epsilon$ refers to the solution accuracy.

The state-of-the-art algorithm that solves the 1D-Denoise problem for the two classes of objective functions considered here, solves a more general problem, the Markov Random Field (MRF) problem. From a graph perspective, the 1D-Denoise problem can be viewed as defined on a path graph consisting of $n$ nodes $\{1, \ldots, n\}$ and $n - 1$ edges $\{(i, i+1)\}_{i=1,\ldots,n-1}$. The MRF problem generalizes the 1D-Denoise problem in that it is not restricted to a path graph, but rather an arbitrary graph $G = (V, E)$ of $n$ nodes in $V$ and $m$ edges in $E$ [1,9,10]. The MRF problem is formulated as:

$$\text{(MRF)} \min_{x_i, i \in V} \sum_{i: i \in V} f_i(x_i) + \sum_{(i, j) \in E} h_{i, j}(x_i - x_j)$$

$$\text{s.t. } \ell_i \leq x_i \leq u_i, \; \forall i \in V. \tag{2}$$

The fastest algorithm for MRF of general convex fidelity functions $f_i$ and convex regularization functions $h_{i, j}$ has complexity $O(nm \log \frac{n^2}{m} \log \frac{nU}{\epsilon})$ ($U = \max_i \{u_i - \ell_i\}$) [1]. The complexity model used here is the oracle unit cost complexity model that assumes $O(1)$ time to evaluate the function value on any argument input of polynomial length. This complexity model has no restriction on the structure of the convex functions. Since for the 1D-Denoise, $m = n - 1$, applying the algorithm of [1] to 1D-Denoise yields complexity of $O(n^2 \log n \log \frac{nU}{\epsilon})$. Other than the $O(\log \frac{U}{\epsilon})$ factor which our algorithm adds, it improves the complexity for the dsc-class of objective functions as compared to the best existing algorithm by a factor of $O(n \log n)$, and for the cc-class of objective functions, our algorithm improves on the complexity of the best existing algorithm by a $O(\log n)$ factor.

The paper is organized as follows. Section 1.1 reviews literature on special cases of 1D-Denoise and the associated ad-hoc algorithms. Notations and preliminaries are introduced in Sect. 2. We present an overview of our unified KKT approach in Sect. 3,
followed by the discussion of the dsc-algorithm in Sect. 4 and the cc-algorithm in Sect. 5. In Sect. 6, we present a numerically improved implementation of the KKT approach that combines the advantages of both the dsc-algorithm and the cc-algorithm. Numerical experimental comparison results are reported in Sect. 7. Conclusion and future research directions are provided in Sect. 8.

1.1 Literature review

One of the most studied special cases of the 1D-Denoise problem has $\ell_2$ fidelity functions and $\ell_1$ regularization functions, a.k.a. the total variation (TV) regularization functions:

$$
(\ell_2 - \text{TV-unweighted}) \min_{x_1, \ldots, x_n} \frac{1}{2} \sum_{i=1}^{n} (x_i - a_i)^2 + \lambda \sum_{i=1}^{n-1} |x_i - x_{i+1}|.
$$

The suffix “-unweighted” refers to the identical coefficient $\lambda$ of all the regularization terms. The use of TV-regularized terms have been shown to lead to piecewise constant recovered signals with sharp jumps/edges. The $\ell_2$-TV-unweighted model is widely applied in signal processing, statistics, machine learning, and many fields of science and engineering [7,17–19].

Rudin et al. in [19] first advocated the use of $\ell_1$ regularization in image signal processing, and since then popularized total variation (TV) denoising models. In statistics contexts $\ell_2$-TV-unweighted is known as a least squares penalized regression model with total variation penalties [18]. The use of efficient taut string algorithm for the problem is well-known in the statistics community [3,6–8,18]. The classic taut string algorithm solves the $\ell_2$-TV-unweighted problem in $O(n)$ time.

A number of other algorithms were proposed to solve $\ell_2$-TV-unweighted in the signal processing community. Condat in [4] proposed an algorithm with theoretical complexity of $O(n^2)$ which is fast in practice with observed performance of $O(n)$. Johnson in [14] proposed an efficient dynamic programming algorithm with complexity $O(n)$. Kolmogorov et al. in [16] proposed a message passing algorithm framework such that, when applied to the $\ell_2$-TV-unweighted problem, has complexity $O(n)$. Barbero and Sra in [2] proposed three algorithms to solve the problem. The first one is a Projected Newton method. It is an iterative algorithm where for this case the authors reduce the cost of each iteration from a standard $O(n^3)$ complexity to $O(n)$ by exploiting the structure of the problem. The second one is a linearized taut string algorithm, which they showed is equivalent to Condat’s algorithm in [4] and hence has worst case complexity of $O(n^2)$. The third one is a hybrid taut string algorithm that combines the advantage of the classic taut string algorithm and the proposed linearized taut string algorithm, with complexity $O(n^5)$ for some $S \in (1, 2)$.

Some of the aforementioned algorithms can be applied to the weighted variant of $\ell_2$-TV-unweighted, namely the $\ell_2$-TV-weighted problem:
\[(\ell_2\text{-TV-weighted}) \quad \min_{x_1, \ldots, x_n} \frac{1}{2} \sum_{i=1}^{n} (x_i - a_i)^2 + \sum_{i=1}^{n-1} c_{i,i+1} |x_i - x_{i+1}|. \quad (4)\]

The difference between the weighted and unweighted problems is that the regularized coefficients \(c_{i,i+1}\) can be different for each regularization term. The classic taut string algorithm can be easily adapted for this case while maintaining the \(O(n)\) complexity \([2,6] \). The Projected Newton method and the linearized taut string algorithm are adapted in \([2]\) to solve this case as well, with no change in the respective complexities. Kolmogorov et al.’s message passing algorithm in \([16]\) has \(O(n)\) complexity for the weighted case as well.

The use of absolute value (\(\ell_1\)) loss functions in data fidelity terms is known to be more robust to heavy-tailed noises and to the presence of outliers than the quadratic (\(\ell_2\)) loss functions \([11,20]\). As such, the following \(\ell_1\text{-TV}\) problem has been investigated in the literature:

\[(\ell_1\text{-TV}) \quad \min_{x_1, \ldots, x_n} \sum_{i=1}^{n} c_i |x_i - a_i| + \sum_{i=1}^{n-1} c_{i,i+1} |x_i - x_{i+1}|. \quad (5)\]

Storath et al. \([20]\) studied an unweighted version of \(\ell_1\text{-TV}\), where \(c_{i,i+1} = \lambda\) for all \(i\), for real-valued and circle-valued data, and provided an \(O(nK)\) algorithm for \(K\) denoting the number of different values in \(\{a_i\}_{i=1}^{n}\). If the data is quantized to finitely many levels, the algorithm’s complexity is \(O(n)\). In the worst case, this complexity can be \(O(n^2)\). Kolmogorov et al. \([16]\) studied a generalized version of \(\ell_1\text{-TV}\) where each fidelity function is a convex piecewise linear function with \(O(1)\) breakpoints. They provided two efficient algorithms of complexities \(O(n \log n)\) and \(O(n \log \log n)\) respectively. Hochbaum and Lu \([11]\) studied a further generalized version where each fidelity function is a convex piecewise linear function with an arbitrary number of breakpoints. They provided an \(O(q \log n)\) algorithm for \(q\) the total number of breakpoints of the \(n\) convex piecewise linear fidelity functions. Applying Hochbaum and Lu’s algorithm to \(\ell_1\text{-TV}\) gives an \(O(n \log n)\) algorithm since \((q = n)\).

The Tikhonov-regularized terms impose global smoothing over the denoised signals, rather than preserving sharp jumps/edges using the TV-regularized terms. With \(\ell_2\) fidelity functions the problem formulation is:

\[(\ell_2\text{-Tikhonov}) \quad \min_{x_1, \ldots, x_n} \frac{1}{2} \sum_{i=1}^{n} c_i (x_i - a_i)^2 + \frac{1}{2} \sum_{i=1}^{n-1} c_{i,i+1} (x_i - x_{i+1})^2. \quad (6)\]

The optimal solution to \(\ell_2\text{-Tikhonov}\) can be obtained by solving a system of linear equations. It is easy to verify that this system of equations is a tridiagonal system of equations (i.e., the matrix \(A\) is a tridiagonal matrix for the equations \(Ax = b\)). It is well-known that the Thomas algorithm \([5]\) can solve this special form of system of linear equations in \(O(n)\) time.
All the aforementioned special cases can be cast under the umbrella of a more general $\ell_p-\ell_q$ problem, for any $p, q \geq 1$:

$$\min_{x_1, \ldots, x_n} \frac{1}{p} \sum_{i=1}^{n} c_i |x_i - a_i|^p + \frac{1}{q} \sum_{i=1}^{n-1} c_{i,i+1} |x_i - x_{i+1}|^q. \quad (7)$$

Weinmann et al. [21] studied the unweighted version of $\ell_p-\ell_q$ where all $c_{i,i+1} = \lambda$ for a fixed $\lambda$ value:

$$\min_{x_1, \ldots, x_n} \frac{1}{p} \sum_{i=1}^{n} |x_i - a_i|^p + \frac{\lambda}{q} \sum_{i=1}^{n-1} |x_i - x_{i+1}|^q. \quad (8)$$

Weinmann et al. solved problem (8) for manifold-valued data, which is more general than a real line considered here, with an $O(nkT)$ algorithm, where $T$ is the number of iterations in order to converge to certain stopping criterion and $k$ is the complexity of computation at each iteration. For the cases $q = 1$ (TV) and $q = 2$ (Tikhonov), they proved that $k = O(1)$. The convergence of their algorithm is proved in [21] but there is no bound provided on the value of $T$. In addition, it was also studied in [21] the models where the fidelity terms and/or the regularization terms are replaced by the Huber function [13]:

$$\rho_k(x) = \begin{cases} \frac{1}{2}x^2, & \text{for } |x| < k \\ k|x| - \frac{1}{2}k^2, & \text{for } |x| \geq k. \end{cases} \quad (9)$$

The Huber function is known to be more robust to outliers than the quadratic ($\ell_2$) functions [13]. Weinmann et al. showed that the computational complexity of every iteration of their algorithm for Huber functions is $k = O(1)$. In this paper, we conduct the experimental study on two problems with Huber objective functions.

### 2 Notation and preliminaries

In this paper, unless specified otherwise, we do not assume any restriction on the functions $f_i$ and $h_i$ except for convexity and adopt the same unit cost complexity model as in [1] for MRF (2). Therefore, each function can be evaluated $O(1)$ time for a given input. In addition, our algorithms provide solutions to the $\epsilon$-accuracy of the optimal solution, in other words, our solution has the first $\log \frac{1}{\epsilon}$ digits same as the optimal solution.

When we consider the case where a function $f$ is differentiable, we assume that its derivative can be evaluated in $O(1)$ time as well for any input. Otherwise, we can compute the function’s gradients on the $\epsilon$ solution accuracy granularity, where the the left and right subgradients of function $f$ on input $x$ is:

$$f'_L(x) = (f(x) - f(x - \epsilon))/\epsilon,$$

$$f'_R(x) = (f(x + \epsilon) - f(x))/\epsilon. \quad (10)$$
Thus both the left and right subgradients can be computed in $O(1)$ time. The range from the left subgradient to the right subgradient of function $f$ at $x$ forms the subdifferential of $f$ at $x$, $\partial f(x) = [f'_L(x), f'_R(x)]$.

By the convexity of function $f$, for any $x_1 < x_2$, we have

$$f'_L(x_1) \leq f'_R(x_1) \leq f'_L(x_2) \leq f'_R(x_2). \quad (11)$$

A function $f(x)$ is said to be strictly convex if for any $x_1 < x_2$ and $\lambda \in (0,1)$:

$$f(\lambda x_1 + (1-\lambda)x_2) < \lambda f(x_1) + (1-\lambda)f(x_2).$$

Given a strictly convex function $f(x)$, for any $x_1 < x_2$, the second inequality of (11) hold strictly:

$$f'_L(x_1) \leq f'_R(x_1) < f'_L(x_2) \leq f'_R(x_2). \quad (12)$$

We introduce the following subgradient-to-argument inverse operations. For a convex function $h$ and a given subgradient value $g$, we would like to compute the maximal interval of argument $z$, $[z_L, z_R]$, such that for any $z \in [z_L, z_R]$, $g \in \partial h(z) = [h'_L(z), h'_R(z)]$. The maximality of the interval implies that $z_L$ satisfies $g \in \partial h(z_L)$ and $\forall z < z_L, h'_R(z) < g$; similarly, $z_R$ satisfies that $g \in \partial h(z_R)$ and $\forall z > z_R, h'_L(z) > g$. We denote the two inverse operations to compute $z_L$ and $z_R$ as $z_L := (\partial h)_L^{-1}(g)$ and $z_R := (\partial h)_R^{-1}(g)$ respectively. Our algorithms will use these two inverse operations.

By the convexity of function $h$, for any two subgradient values $g_1 < g_2$, we have

$$(\partial h)_L^{-1}(g_1) \leq (\partial h)_R^{-1}(g_1) \leq (\partial h)_L^{-1}(g_2) \leq (\partial h)_R^{-1}(g_2).$$

In addition, if function $h$ is strictly convex, as the subgradients are strictly increasing (from inequalities (12)), we have $(\partial h)_L^{-1}(g) = (\partial h)_R^{-1}(g)$. In this case $(\partial h)_L^{-1}(g) = (\partial h)_R^{-1}(g)$ and we denote this value as $g^{-1}(g)$.

As for the complexity of the inverse operations, it is possible to find the values of $g^{-1}(g)$ or $(\partial h)_R^{-1}(g)$ for a given $g$ by binary search on an interval of length $O(U)$. This is because function $h$ is convex, thus finding these values reduces to finding the zeros of monotone subgradient functions. The complexity of computing a subgradient-to-argument inverse is thus $O(\log \frac{U}{\epsilon})$ to the $\epsilon$-accuracy. Note that if $h$ has a special structure, the complexity of the inverse can be improved to $O(1)$. Some examples are $h$ being quadratic ($\ell_2$) or piecewise linear with few pieces, including absolute value ($\ell_1$) functions.

### 2.1 An equivalent formulation of 1D-Denoise

We now present an equivalent formulation of 1D-Denoise (1). Consider each regularization function $h_i(x_i - x_{i+1})$. Since each $x_i$ is bounded in $[-U, U]$, it follows that $x_i - x_{i+1}$ is bounded in the interval $[-2U, 2U]$. Let $c_i$ be the $\epsilon$-accurate minimizer
of the regularization function $h_i$ in $[-2U, 2U]$. This can be done by binary search over the interval $[-2U, 2U]$, which takes time $O(\log \frac{U}{\epsilon})$. Then we split function $h_i(z_i)(z_i = x_i - x_{i+1})$ into two convex functions, $h_{i,i+1}(z_{i,i+1})$ and $h_{i+1,i}(z_{i+1,i})$:

$$
\begin{align*}
  h_{i,i+1}(z_{i,i+1}) &= h_i(z_i + c_i) - h_i(c_i), \\
  h_{i+1,i}(z_{i+1,i}) &= h_i(c_i - z_{i+1,i}) - h_i(c_i).
\end{align*}
$$

Without loss of generality, we have $0 \in \partial h_i(c_i)$.\footnote{This is also true even if $c_i$ attains at the boundary, $-2U$ or $2U$. If $c_i = -2U$, implying that $h_i$ is non-decreasing in $[-2U, 2U]$, then we can transform $h_i$ in the interval $(\infty, -2U)$ to a decreasing function. This transformation does not change the optimal solution to the original problem while keeping $h_i$ convex. It also guarantees that $c_i = -2U$ is a global minimizer so $0 \in \partial h_i(c_i)$. The case of $c_i = 2U$ is similar.}

The split functions $h_{i,i+1}$ and $h_{i+1,i}$ have the following properties:

**Proposition 1** The (strict) convexity of function $h_i$ implies the (strict) convexity of functions $h_{i,i+1}$ and $h_{i+1,i}$. Both functions $h_{i,i+1}(z_{i,i+1})$ and $h_{i+1,i}(z_{i+1,i})$ are non-decreasing for $z_{i,i+1}, z_{i+1,i} \geq 0$, and $h_{i,i+1;R}(0), h_{i+1,i;R}(0) \geq 0$.

**Proof** We prove the properties for $h_{i,i+1}$. The case of $h_{i+1,i}$ can be proved similarly.

If function $h_i$ is (strictly) convex, so is function $h_i(z_{i,i+1} + c_i) - h_i(c_i)$, hence by definition, $h_{i,i+1}(z_{i,i+1})$ is also (strictly) convex.

By definition, we have $h_{i,i+1}'(z_{i,i+1}) = h_i'(z_{i,i+1} + c_i)$. Since $0 \in \partial h_i(c_i)$, by definition we have $h_{i,i+1;'L}(c_i) \leq 0 \leq h_{i,i+1;'R}(c_i)$. As a result, plugging $z_{i,i+1} = 0$ into $h_{i,i+1}'(z_{i,i+1})$, we have $h_{i,i+1;'R}(0) = h_{i,i+1;'R}(c_i) \geq 0$. Combining the convexity of $h_{i,i+1}(z_{i,i+1})$ and $h_{i,i+1;'R}(0) \geq 0$, we have that for any $z_{i,i+1} > 0$, $h_{i,i+1;'R}(z_{i,i+1}) \geq h_{i,i+1;'R}(0) \geq 0$. This implies that $h_{i,i+1}(z_{i,i+1})$ is nondecreasing for $z_{i,i+1} \geq 0$.

By splitting the regularization functions, we introduce the following equivalent formulation of 1D-Denoise (1):

\begin{equation}
\begin{align*}
\text{(1D-Denoise)} \quad \min_{\{x_i\}_{i=1,\ldots,n}, \{z_{i,i+1}, z_{i+1,i}\}_{i=1,\ldots,n-1}} \quad & \sum_{i=1}^{n} f_i(x_i) + \sum_{i=1}^{n-1} h_{i,i+1}(z_{i,i+1}) + \sum_{i=1}^{n-1} h_{i+1,i}(z_{i+1,i}) \\
\text{s.t.} \quad & x_i - x_{i+1} \leq z_{i,i+1} + c_i \quad i = 1, \ldots, n-1 \\
& x_{i+1} - x_{i} \leq z_{i+1,i} - c_i \quad i = 1, \ldots, n-1 \\
& z_{i,i+1}, z_{i+1,i} \geq 0 \quad i = 1, \ldots, n-1.
\end{align*}
\end{equation}

(13)

The formulation (13) and formulation (1) are equivalent:

**Lemma 2** Formulations (1) and (13) share the same optimal solution.

**Proof** Let $P(\{x_i\}_{i=1,\ldots,n})$ be the objective value of 1D-Denoise (1) for any given values $x_1, \ldots, x_n$. Let $P(\{z_{i,i+1}, z_{i+1,i}\}_{i=1,\ldots,n-1}|\{x_i\}_{i=1,\ldots,n})$ be the objective value of problem (13) such that, given the values of $\{x_i\}_{i=1,\ldots,n}$, the values of the $z$ variables are selected to minimize the objective function. Note that given $x_i$ and $x_{i+1}$, either
We illustrate the KKT optimality conditions of 1D-Denoise (13). Let the dual variable for each constraint $x_i - x_{i+1} - c_i$ or $x_{i+1} - x_i + c_i$ be $\lambda_{i,i+1}$; let the dual variable for each constraint $x_{i+1} - x_i - c_i$ be $\lambda_{i+1,i}$; let the dual variable for each constraint $z_{i,i+1}$ be $\mu_{i,i+1}$; and let the dual variable for each constraint $z_{i+1,i}$ be $\mu_{i+1,i}$. The KKT optimality conditions state that, for all $i$, $(x_i^*, z_{i,i+1}^*, z_{i+1,i}^*)$ is an optimal solution to 1D-Denoise (13) if and only if there exist subgradients \( \left( \left( f_i'(x_i^*) \right)_i, \left( h_{i,i+1}'(z_{i,i+1}^*) \right)_i, \left( h_{i+1,i}'(z_{i+1,i}^*) \right)_i \right) \), and optimal values of the dual variables \( \left( \lambda_{i,i+1}^*, \lambda_{i+1,i}^* \right), \left( \mu_{i,i+1}^*, \mu_{i+1,i}^* \right) \) such that the following conditions hold:

\[
\begin{align*}
\text{(Stationarity)} & \quad f_i'(x_i^*) + \lambda_{i,i+1}^* - \lambda_{i+1,i}^* = 0, \\
& \quad f_i'(x_i^*) - \lambda_{i-1,i}^* + \lambda_{i+1,i+1}^* + \lambda_{i,i+1}^* - \lambda_{i+1,i}^* = 0, \quad i = 2, \ldots, n - 1 \\
& \quad f_{n}'(x_{n}^*) - \lambda_{n-1,n}^* + \lambda_{n,n-1}^* = 0.
\end{align*}
\]

\[
\begin{align*}
\text{(Stationarity)} & \quad h_{i,i+1}'(z_{i,i+1}^*) - \lambda_{i,i+1}^* - \mu_{i,i+1}^* = 0, \quad i = 1, \ldots, n - 1 \\
& \quad h_{i+1,i}'(z_{i+1,i}^*) - \lambda_{i+1,i}^* - \mu_{i+1,i}^* = 0, \quad i = 1, \ldots, n - 1
\end{align*}
\]

\[
\begin{align*}
\text{(Primal feasibility)} & \quad x_i^* - x_i^{*+} \leq z_{i,i+1}^* + c_i, \quad i = 1, \ldots, n - 1 \\
& \quad x_{i+1}^* - x_i^{*+} \leq z_{i,i+1}^* - c_i, \quad i = 1, \ldots, n - 1 \\
& \quad z_{i,i+1}^* - z_{i+1,i}^* \geq 0, \quad i = 1, \ldots, n - 1
\end{align*}
\]
Equations (14) and (15) are stationarity conditions for $x$ and $z$ variables respectively, inequalities (16) and (17) are primal and dual feasibility conditions for the primal $x$, $z$ variables and the dual $\lambda$, $\mu$ variables respectively, and equations (18) are complementary slackness (C-S) conditions.

Our approach directly solves the above KKT optimality conditions. We derive two key results from the KKT conditions. The first is a propagation lemma. For the complementary slackness (C-S) conditions.

\begin{equation}
\begin{aligned}
(Dual \ feasibility) \quad & \lambda_{i, i+1}^*, \lambda_{i, i+1}^* \geq 0, \quad i = 1, \ldots, n - 1 \\
& \mu_{i, i+1}, \mu_{i, i+1}^* \geq 0, \quad i = 1, \ldots, n - 1 \\
(C-S) \quad & (x_i^* - x_{i+1}^* - z_i^* - c_i - c_i)\lambda_{i, i+1}^* = 0, \quad i = 1, \ldots, n - 1 \\
& (x_i^* - x_{i+1}^* - z_i^* + c_i)\lambda_{i, i+1}^* = 0, \quad i = 1, \ldots, n - 1 \\
& z_{i, i+1}^* \mu_{i, i+1}^* = 0, \quad i = 1, \ldots, n - 1 \\
& z_{i, i+1}^* \mu_{i, i+1}^* = 0, \quad i = 1, \ldots, n - 1.
\end{aligned}
\end{equation}

Equations (14) and (15) are stationarity conditions for $x$ and $z$ variables respectively, inequalities (16) and (17) are primal and dual feasibility conditions for the primal $x$, $z$ variables and the dual $\lambda$, $\mu$ variables respectively, and equations (18) are complementary slackness (C-S) conditions.

Our approach directly solves the above KKT optimality conditions. We derive two key results from the KKT conditions. The first is a propagation lemma. For the differentiable-strictly-convex class, we show that, for any value of $x_1$, one can follow the KKT conditions to uniquely determine the values of $x_2, \ldots, x_n$. The only leftover condition in the list of KKT conditions for which the generated sequence $(x_1, x_2, \ldots, x_n)$ may not satisfy, i.e., prevents the generated sequence from being optimal, is an equation that requires $\sum_{i=1}^n f_i'(x_i)$ equal to 0 (achieved by summing up the equations in (14)). If not, then thanks to the convexity of $f_i$ and $h_i', i+1$, we next have a monotonicity lemma that shows if $\sum_{i=1}^n f_i'(x_i) > 0$, we should search for a smaller $x_1$; otherwise we should search for a larger $x_1$. Combining both the propagation lemma and the monotonicity lemma naturally yields a binary search algorithm to the optimal $x_1$, which further leads to optimal $x_2$ to $x_n$ based on the propagation result. This is the dsc-algorithm we shall present next. For the convex-convex class, the propagation result is extended from a unique value to a unique range, i.e., for any value of $x_1$, one can follow the KKT conditions to uniquely determine the ranges of $x_2, \ldots, x_n$. Those ranges finally determine a lower bound and an upper bound for the sum $\sum_{i=1}^n f_i'(x_i)$. If value 0 is within the bounds, then $x_1$ is the optimal value. Otherwise, we have an extended monotonicity result showing that if the lower bound is above 0, then we should search for a smaller $x_1$, otherwise (the upper bound is below 0), we should search for a larger $x_1$. By the dichotomy, the optimal value of $x_1$ is found. This process is then repeated from $x_2$ to $x_n$ to find the respective optimal values. This is the cc-algorithm we shall present in Sect. 5.

4 KKT approach for differentiable-strictly-convex class: dsc-algorithm

With convex differentiable fidelity functions and strictly convex regularization functions, we show that if we fix the value of $x_1$, then all the values of $x_2$ to $x_n$ can be uniquely determined by the KKT optimality conditions. This is proved in the following propagation lemma:

**Lemma 3** (Propagation Lemma) Given any value of $x_1$, the KKT optimality conditions (14), (15), (16), (17) and (18) uniquely determine the other values of $x_2, \ldots, x_n$. 

© Springer
Proof We prove the lemma by induction on \( i \) from 1 to \( n \). The case of \( i = 1 \) is trivial. Suppose the values of \( x_1, \ldots, x_l \) are uniquely determined by \( x_1 \) for some \( i \) \((1 \leq i \leq n - 1)\), we show that the value of \( x_{i+1} \) is also uniquely determined.

Adding up the equations of \( j = 1, \ldots, i \) in the stationarity conditions (14), we have

\[
\sum_{j=1}^{i} f'_j(x_j) = -\lambda_{i,i+1} + \lambda_{i+1,i}.
\]

(19)

There are 5 different cases about Eq. (19), depending on the value of \( \sum_{j=1}^{i} f'_j(x_j) \):

1. \( \sum_{j=1}^{i} f'_j(x_j) > h'_{i+1,i}; R(0) \geq 0 \):

By the dual feasibility conditions (17), we have \( \lambda_{i,i+1}, \lambda_{i+1,i} \geq 0 \). Hence \( \lambda_{i+1,i} \geq \sum_{j=1}^{i} f'_j(x_j) > 0 \). If \( \lambda_{i+1,i} = \sum_{j=1}^{i} f'_j(x_j) \), then \( \lambda_{i+1,i} > 0 \) as well. Then by the complementary slackness conditions (18), we have \( x_i - x_{i+1} - z_{i,i+1} - c_i = 0 \) \text{ and } \( x_{i+1} - x_i - z_{i,i+1} + c_i = 0 \). These two equations imply that \( z_{i,i+1} \geq 0 \). On the other hand, by the primal feasibility conditions (16), we have \( z_{i,i+1}, z_{i+1,i} \geq 0 \). As a result, it must be \( z_{i,i+1} = z_{i+1,i} = 0 \). Then we plug \( z_{i+1,i} = 0 \) into the stationarity condition on \( z_{i+1,i} \) in (15), we have that there exists a subgradient \( h'_{i+1,i}(0) \) such that \( h'_{i+1,i}(0) = \lambda_{i+1,i} + \mu_{i+1,i} \geq \lambda_{i+1,i} > \sum_{j=1}^{i} f'_j(x_j) > h'_{i+1,i}; R(0) (\mu_{i+1,i} \geq 0 \) is due to the dual feasibility conditions (17)), which is a contradiction. Therefore we have \( \lambda_{i+1,i} = \sum_{j=1}^{i} f'_j(x_j) \) and \( \lambda_{i,i+1} = 0 \). Then by the stationarity condition on \( z_{i+1,i} \) in (15), we have that there exists a subgradient \( h'_{i+1,i}(z_{i+1,i}) \) such that \( h'_{i+1,i}(z_{i+1,i}) = \lambda_{i+1,i} + \mu_{i+1,i} \geq \lambda_{i+1,i} = \sum_{j=1}^{i} f'_j(x_j) > h'_{i+1,i}; R(0) \). As a result, we have \( z_{i+1,i} > 0 \), and this implies that \( \mu_{i+1,i} = 0 \) by the complementary slackness conditions (18). And since \( h_{i+1,i}(z_{i+1,i}) \) is a strictly convex function, \( z_{i+1,i} \) is thus uniquely determined by

\[
z_{i+1,i} = (\partial h_{i+1,i})^{-1}(\lambda_{i+1,i}) = (\partial h_{i+1,i})^{-1}\left(\sum_{j=1}^{i} f'_j(x_j)\right).
\]

Since \( \lambda_{i+1,i} > 0 \), by the complementary slackness conditions (18), we have \( x_{i+1} \) is uniquely determined by the equation \( x_{i+1} = x_i + z_{i+1,i} - c_i \).

2. \( 0 < \sum_{j=1}^{i} f'_j(x_j) \leq h'_{i+1,i}; R(0) \geq 0 \):

This case exists only if \( h'_{i+1,i}; R(0) > 0 \). By the dual feasibility conditions (17), we still have \( \lambda_{i,i+1}, \lambda_{i+1,i} \geq 0 \), and thus \( \lambda_{i+1,i} \geq \sum_{j=1}^{i} f'_j(x_j) \). If \( \lambda_{i+1,i} > h'_{i+1,i}; R(0) \), we can derive the same contradiction as Case 1. As a result, it must be \( 0 < \sum_{j=1}^{i} f'_j(x_j) \leq \lambda_{i+1,i} \leq h'_{i+1,i}; R(0) \). Then we consider the stationarity conditions (15) on \( z_{i+1,i} ; h'_{i+1,i}(z_{i+1,i}) = \lambda_{i+1,i} + \mu_{i+1,i} \). If \( \mu_{i+1,i} > 0 \), then by the complementary slackness conditions (18), we have \( z_{i+1,i} = 0 \). If \( \mu_{i+1,i} = 0 \), then \( h'_{i+1,i}(z_{i+1,i}) = \lambda_{i+1,i} \leq h'_{i+1,i}; R(0) \), which still implies that \( z_{i+1,i} = 0 \) by the strict convexity of \( h_{i+1,i} \). In either case, by the complementary slackness
conditions (18) with \( \lambda_{i+1,i} > 0 \), we have \( x_{i+1} \) is uniquely determined by the equation \( x_{i+1} = x_i + z_{i+1,i} - c_i = x_i - c_i \).

3. \( \sum_{j=1}^{i} f_j'(x_j) = 0 \):
   In this case, we have \( \lambda_{i,i+1} = \lambda_{i+1,i} \). If both \( \lambda_{i,i+1} \) and \( \lambda_{i+1,i} \) are positive, then by the complementary slackness conditions (18) on \( \lambda_{i,i+1} \) and \( \lambda_{i+1,i} \), and the primal feasibility conditions (16) on \( z_{i+1} \) and \( z_{i,i+1} \), we have \( z_{i,i+1} = z_{i+1,i} = 0 \).
   As a result, \( x_{i+1} = x_i - c_i \). If \( \lambda_{i,i+1} = \lambda_{i+1,i} = 0 \), we consider the stationarity conditions (15) on both \( z_{i,i+1} \) and \( z_{i+1,i} \). For \( z_{i,i+1} \), we have that there exists a subgradient \( h_{i,i+1}'(z_{i,i+1}) \) such that \( h_{i,i+1}'(z_{i,i+1}) = \mu_{i,i+1} \geq 0 \). If \( \mu_{i,i+1} \leq h_{i,i+1,i}:R(0) \), by the strict convexity of \( h_{i,i+1} \), we have \( z_{i,i+1} = 0 \); otherwise \( \mu_{i,i+1} > h_{i,i+1,i}:R(0) \), then the complementary slackness condition (18) also implies that \( z_{i,i+1} = 0 \). Therefore we always have \( z_{i,i+1} = 0 \). The same analysis shows that \( z_{i+1,i} = 0 \). Then by the primal feasibility conditions (16) on \( x_i \) and \( x_{i+1} \), we have \( x_{i+1} = x_i - c_i \) being uniquely determined.

4. \(-h_{i,i+1}:R(0) \leq \sum_{j=1}^{i} f_j'(x_j) < 0\):
   This case exists only if \( h_{i,i+1}:R(0) > 0 \), it is symmetric to case 2. By the dual feasibility conditions (17), we have \( \lambda_{i+1,i} \geq 0 \) and \( \lambda_{i,i+1} \geq -\sum_{j=1}^{i} f_j'(x_j) > 0 \). If \( \lambda_{i,i+1} > h_{i,i+1,i}:R(0) \), then \( \lambda_{i+1,i} > 0 \). Same as Case 1, it implies that \( z_{i,i+1} = z_{i+1,i} = 0 \). However, this violates the stationarity conditions (15) on \( z_{i,i+1} \) because \( h_{i,i+1}(0) = \lambda_{i,i+1} + \mu_{i,i+1} \geq \lambda_{i,i+1} > h_{i,i+1,i}:R(0) \). Therefore we have \( 0 < \lambda_{i,i+1} \leq h_{i,i+1,i}:R(0) \). Then we consider the stationarity conditions (15) on \( z_{i,i+1} \):
   \( h_{i,i+1}'(z_{i,i+1}) = \lambda_{i,i+1} + \mu_{i,i+1} \). If \( \mu_{i,i+1} > 0 \), then by the complementary slackness conditions (18), we have \( z_{i,i+1} = 0 \). If \( \mu_{i,i+1} = 0 \), then \( h_{i,i+1}'(z_{i,i+1}) = \lambda_{i,i+1} \leq h_{i,i+1,i}:R(0) \), which still implies that \( z_{i,i+1} = 0 \) by the strict convexity of \( h_{i,i+1} \). In either case, by the complementary slackness conditions (18) with \( \lambda_{i,i+1} > 0 \), we have \( x_{i+1} \) is uniquely determined by the equation \( x_{i+1} = x_i - z_{i,i+1} - c_i = x_i - c_i \).

5. \( \sum_{j=1}^{i} f_j'(x_j) < -h_{i,i+1,i}:R(0) \leq 0\):
   This case is symmetric to Case 1. Following the same reasoning in Case 1, we can show that \( \lambda_{i,i+1} = -\sum_{j=1}^{i} f_j'(x_j) > h_{i,i+1,i}:R(0) \geq 0 \) and \( \lambda_{i+1,i} = 0 \). As a result, we have \( z_{i+1,i} > 0 \), thus \( \mu_{i,i+1} = 0 \) by the complementary slackness conditions (18). And since \( h_{i,i+1}(z_{i,i+1}) \) is a strictly convex function, \( z_{i,i+1} \) is uniquely determined by
   \[
   z_{i,i+1} = (\partial h_{i,i+1}^{-1})(\lambda_{i,i+1}) = (\partial h_{i,i+1}^{-1}) \left( -\sum_{j=1}^{i} f_j'(x_j) \right).
   \]
   Since \( \lambda_{i,i+1} > 0 \), by the complementary slackness conditions (18), we have \( x_{i+1} \) is uniquely determined by the equation \( x_{i+1} = x_i - z_{i,i+1} - c_i \).

This completes the proof for the case of \( i + 1 \).
Lemma 3 implies that, given a value of $x_1$, the values of $x_2$ to $x_n$ are uniquely determined by the following iterative equations:

$$x_{i+1} = x_i + z_i - c_i, \quad i = 1, \ldots, n - 1$$

(20)

where

$$z_i = \begin{cases} (\partial h_{i+1,i})^{-1}(\sum_{j=1}^i f'_j(x_j)), & \text{if } \sum_{j=1}^i f'_j(x_j) > h'_{i+1,i};R(0), \\ 0, & \text{if } -h'_{i+1,i};R(0) \leq \sum_{j=1}^i f'_j(x_j) \leq h'_{i+1,i};R(0), \\ -(\partial h_{i,i+1})^{-1}(-\sum_{j=1}^i f'_j(x_j)), & \text{if } \sum_{j=1}^i f'_j(x_j) < -h'_{i,i+1};R(0). \end{cases}$$

(21)

Based on the convexity of $f_i(x)$, $h_{i,i+1}(z_i,i+1)$, and $h_{i+1,i}(z_{i+1},i)$, we have the following monotonicity property for any two sequences of $x_1, \ldots, x_n$ generated by Eqs. (20) and (21):

**Corollary 4 (Monotonicity property)** Let $x_1^{(1)} < x_1^{(2)}$ be any two given values of variable $x_1$. Let $(x_1^{(1)}, x_2^{(1)}, \ldots, x_n^{(1)})$ and $(x_1^{(2)}, x_2^{(2)}, \ldots, x_n^{(2)})$ be the respective sequence of $x$ values determined by the value of $x_1$ and Eqs. (20) and (21). Then we have $x_i^{(1)} < x_i^{(2)}$ for all $i = 1, \ldots, n$.

**Proof** The proof is by induction on $i$ ($i = 1, \ldots, n$). For $i = 1$, the values of $x_1^{(1)}$ and $x_1^{(2)}$ are given and satisfy $x_1^{(1)} < x_1^{(2)}$. Suppose $x_i^{(1)} < x_i^{(2)}$ for all $j = 1, \ldots, i$ for some $i$ ($1 \leq i \leq n - 1$), we show that $x_{i+1}^{(1)} < x_{i+1}^{(2)}$.

Due to the convexity of $h_{i,i+1}$ and $h_{i+1,i}$, Eq. (21) implies that $z_i$ is a nondecreasing function of $\sum_{j=1}^i f'_j(x_j)$. On the other hand, since all $f_j$ ($j = 1, \ldots, i$) functions are convex, by the induction hypothesis, we have $f'_j(x_j^{(1)}) \leq f'_j(x_j^{(2)})$ for $j = 1, \ldots, i$. Hence $\sum_{j=1}^i f'_j(x_j^{(1)}) \leq \sum_{j=1}^i f'_j(x_j^{(2)})$. As a result, we have $z_i^{(1)} \leq z_i^{(2)}$. Since we have $x_i^{(1)} < x_i^{(2)}$ by the induction hypothesis, we have $x_i^{(1)} = x_i^{(1)} + z_i^{(1)} - c_i < x_i^{(2)} + z_i^{(2)} - c_i = x_i^{(2)}$. \hfill □

The only equation in the KKT optimality conditions that a given sequence of $x_1, \ldots, x_n$ determined by Eqs. (20) and (21) may violate is the last stationarity condition (14) for $x_n$. This is because $x_n$, $\lambda_{n-1,n}$ and $\lambda_{n,n-1}$ are determined in the step of computing $x_n$ from $x_{n-1}$, based on the equation $\sum_{j=1}^{n-1} f'_j(x_j) = \lambda_{n,n-1} - \lambda_{n-1,n}$. However, the generated values of $x_n$, $\lambda_{n-1,n}$ and $\lambda_{n,n-1}$ do not necessarily satisfy the last stationarity condition for $x_n$, $f_n(x_n) - \lambda_{n-1,n} + \lambda_{n,n-1} = 0$. On the other hand, we observe that if we sum up all the stationarity conditions (14) for the $x$ variables, we have:

$$\sum_{i=1}^n f'_i(x_i^n) = 0.$$ 

(22)
The equation for $x_n$ in the stationarity conditions (14) can be equivalently replaced by Eq. (22). Hence a sequence of $x_1, \ldots, x_n$ determined by Eqs. (20) and (21) satisfy the KKT optimality conditions if and only if Eq. (22) also holds.

The above analysis implies a binary search algorithm to solve the KKT optimality conditions. In every iteration, we try a value of $x_1$, and compute the values of $x_2$ to $x_n$ based on Eqs. (20) and (21). Then we check whether Eq. (22) holds for the generated sequence of $x_1, \ldots, x_n$. If yes, then the generated sequence of $x_1, \ldots, x_n$ satisfies the KKT optimality conditions, thus it is an optimal solution to 1D-Denoise. Otherwise, we determine the next value of $x_1$ to try based on the sign of $\sum_{i=1}^nf_i'(x_i)$ of the currently generated sequence of $x_1, \ldots, x_n$: If $\sum_{i=1}^nf_i'(x_i) > 0$, we try a smaller value of $x_1$; If $\sum_{i=1}^nf_i'(x_i) < 0$, we try a larger value of $x_1$. The binary search efficiently determines the optimal value of $x_1$, so are the optimal values of $x_2$ to $x_n$ by Eqs. (20) and (21). The complete dsc-algorithm is presented in Box 1.

dsc-algorithm for 1D-Denoise of convex differentiable fidelity functions and strictly convex regularization functions:

**Step 0:** Solve $c_i := \arg\min_{-U \leq z_i \leq U} h_i(z_i)$ for $i = 1, \ldots, n - 1$. Initialize the lower and upper bounds of the search region for $x_1$ as $l := -U$ and $u := U$ respectively.

**Step 1:** Set $x_1 := \frac{l+u}{2}$. Compute the values of $x_2, \ldots, x_n$ based on iterative equations (20) and (21):

$$x_{i+1} := x_i + z_i - c_i, \quad i = 1, \ldots, n - 1$$

where

$$z_i := \begin{cases} (\partial h_{i+1,i})^{-1}(\sum_{j=1}^if_j'(x_j)), & \text{if } \sum_{j=1}^if_j'(x_j) > h_{i+1,i;R}(0), \\ 0, & \text{if } -h_{i+1,i;R}(0) \leq \sum_{j=1}^if_j'(x_j) \leq h_{i+1,i;R}(0), \\ -(\partial h_{i,i+1})^{-1}(\sum_{j=1}^if_j'(x_j)), & \text{if } \sum_{j=1}^if_j'(x_j) < -h_{i,i+1;R}(0). \end{cases}$$

**Step 2:** If $u - l < \epsilon$ or $\sum_{i=1}^nf_i'(x_i) = 0$, return $(x_1, x_2, \ldots, x_n)$ and stop.

**Step 3:** If $\sum_{i=1}^nf_i'(x_i) < 0$, set $l := x_1$; otherwise set $u := x_1$. Go to **Step 1**.

AlgoBox 1: dsc-algorithm for 1D-Denoise of convex differentiable fidelity functions and strictly convex regularization functions.

The number of different $x_1$ values we need to check in the algorithm is $O(\log \frac{U}{\epsilon})$. For each $x_1$ value fixed, the complexity to compute the values of $x_2, \ldots, x_n$ by Eqs. (20) and (21) is $O(n \log \frac{U}{\epsilon})$, where the $O(\log \frac{U}{\epsilon})$ term corresponds to the complexity of computing subgradient-to-argument inverse of $h_{i+1,i}$ or $h_{i,i+1}$ functions to compute $z_i$. Hence the complexity of the Step 1 to Step 3 is $O(n \log^2 \frac{U}{\epsilon})$. At Step 0, it takes $O(n \log \frac{U}{\epsilon})$ time to compute the $\epsilon$-accurate $c_i$ values for all regularization functions $\{h_i(z_i)\}_{i=1,\ldots,n-1}$. Thus we have:

**Theorem 5** The 1D-Denoise problem of convex differentiable fidelity functions and strictly convex regularization functions is solved in $O(n \log^2 \frac{U}{\epsilon})$ time.
Note that if the regularization functions $h_i$ are quadratic ($\ell_2$), then the complexity of subgradient-to-argument inverse is $O(1)$, thus the dsc-algorithm can be sped-up to $O(n \log \frac{U}{\varepsilon})$ complexity.

**Remark 6** One may ask if transforming the unconstrained optimization form of 1D-Denoise (1) to a more complicated constrained optimization form (13) is necessary. Actually, the KKT conditions, as shown in the following, with respect to the original formulation (1) are much simpler:

\[
\begin{align*}
0 & \in f'_i(x_1) + \partial h_1(x_1 - x_2), \\
0 & \in f'_i(x_i) - \partial h_{i-1}(x_{i-1} - x_i) + \partial h_i(x_i - x_{i+1}), \quad i = 2, \ldots, n-1 \\
0 & \in f'_n(x_n) - \partial h_{n-1}(x_{n-1} - x_n).
\end{align*}
\]

However, we argue that the above simpler form of KKT conditions are not sufficient to provide a rigorous proof to the unique value propagation result (Lemma 3) for the differentiable-strictly-convex class. For a fixed value of $x_1$, since $h_1$ is strictly convex, we can change the 0-inclusion formula $0 \in f'_i(x_1) + \partial h_1(x_1 - x_2)$ to 0-equation $0 = f'_i(x_1) + \partial h_1(x_1 - x_2)$, and thus uniquely determine the value of $x_2$ as $x_2 = x_1 - (\partial h_1)^{-1}(-f'_i(x_1))$. However, even $x_1$ and $x_2$ are fixed, the subdifferential of $h_1$ at $x_1 - x_2$, $\partial h_1(x_1 - x_2)$, may be a set containing more than one subgradient. As a result, the value of $x_3$ can not be uniquely determined by the 0-inclusion formula $0 \in f'_2(x_2) - \partial h_1(x_1 - x_2) + \partial h_2(x_2 - x_3)$, nor does it help even if we add the two 0-inclusion formulas regarding $f'_i(x_1)$ and $f'_2(x_2)$ because the subdifferential of $h_1$ at $x_1 - x_2$, $\partial h_1(x_1 - x_2)$, may not be able to cancel each other in the two formulas.

For the general convex-convex class, however, as we shall show in the next section, we can follow the simplified KKT conditions to present our algorithm in a concise way.

## 5 KKT approach for convex-convex class: cc-algorithm

We extend the ideas developed from the dsc-algorithm to solve 1D-Denoise of arbitrary convex fidelity and regularization functions, leading to the cc-algorithm.

The impact of removing the differentiability and strict convexity assumptions are two-fold: the non-differentiability of $f_i(x_i)$ implies that a given $x_i$ value corresponds to a non-singleton subdifferential $\partial f_i(x_i) = [f^i_{i,L}(x_i), f^i_{i,R}(x_i)]$, instead of a unique derivative $f'(x_i)$ in the differentiable case; the non-strict convexity of $h_{i,i+1}$ (and $h_{i+1,i}$) implies that a subgradient value $g$ can be inversely mapped to a non-singleton interval of arguments $[(\partial h_{i,i+1})^{-1}_L(g), (\partial h_{i,i+1})^{-1}_R(g)]$ (and $[(\partial h_{i+1,i})^{-1}_L(g), (\partial h_{i+1,i})^{-1}_R(g)]$), instead of a unique argument $(\partial h_{i,i+1})^{-1}(g)$ (and $(\partial h_{i+1,i})^{-1}(g)$) in the strictly convex case. Both observations imply that, based on the KKT optimality conditions, a given value of $x_1$ does not uniquely determine the other variables $x_2, \ldots, x_n$ to values, but to ranges.

For this general convex-convex class, instead of working on the KKT conditions of the reformulation (13), we directly solve the KKT conditions of the original uncon-
strained formulation (1) of the 1D-Denoise problem, shown as follows:

\[
\begin{cases}
0 \in f'_1(x_1) + \partial h_1(x_1 - x_2), \\
0 \in f'_j(x_i) - \partial h_{i-1}(x_{i-1} - x_i) + \partial h_i(x_i - x_{i+1}), \ i = 2, \ldots, n - 1 \\
0 \in f'_n(x_n) - \partial h_{n-1}(x_{n-1} - x_n).
\end{cases}
\] (23)

We first have the following range propagation result which can be viewed as an extension of the propagation lemma in the dsc-algorithm.

**Lemma 7** (Extended Propagation Lemma) Given any value of \( x_1 \), the KKT optimality conditions (23) uniquely determine the ranges of other variables \( x_2, \ldots, x_n \). Specifically, let \([l_{kkt,i}, u_{kkt,i}]\) be the range of variable \( x_i \), we have:

\[
l_{kkt,1} = u_{kkt,1} = x_1,
\]

\[
l_{kkt,i+1} = l_{kkt,i} - (\partial h_i)^{-1}_R \left( - \sum_{j=1}^{i} f'_j:R(l_{kkt,j}) \right), \ \ i = 1, \ldots, n - 1.
\] (24)

**Proof** We proof the lemma by induction on \( i \). Together with proving Eq. (24), we prove the following series of inequalities:

\[
\partial h_i(x_i - x_{i+1}) \in \left[ - \sum_{j=1}^{i} f'_j:R(u_{kkt,j}), - \sum_{j=1}^{i} f'_j:R(l_{kkt,j}) \right], \ i = 1, \ldots, n - 1.
\] (25)

Let’s first prove Eq. (24) and inequality (25) hold for \( i = 1 \). In order to make the first 0-inclusion formula, \( 0 \in f'_1(x_1) + \partial h_1(x_1 - x_2) \), in the KKT conditions (23) hold, we have:

\[
\partial h_1(x_1 - x_2) \in [-f'_1:R(x_1), -f'_1:L(x_1)] \subseteq [-f'_1:R(u_{kkt,1}), -f'_1:R(l_{kkt,1})].
\]

Hence inequality (25) holds for \( i = 1 \). Further, due to the convexity of \( h_1 \), we have

\[
x_2 \in \left[ x_1 - (\partial h_1)^{-1}_R \left( - f'_1:R(l_{kkt,1}) \right), x_1 - (\partial h_1)^{-1}_L \left( - f'_1:R(u_{kkt,1}) \right) \right] \subseteq \left[ l_{kkt,1} - (\partial h_1)^{-1}_R \left( - f'_1:R(l_{kkt,1}) \right), u_{kkt,1} - (\partial h_1)^{-1}_L \left( - f'_1:R(u_{kkt,1}) \right) \right].
\]

Therefore Eq. (24) holds for \( i = 1 \).

The induction from \( i - 1 \) to \( i \) is straightforward by leveraging the \( i \)th 0-inclusion formula, \( 0 \in f'_i(x_i) - \partial h_{i-1}(x_{i-1} - x_i) + \partial h_i(x_i - x_{i+1}) \), in the KKT conditions (23). By the induction hypothesis on inequality (25) on \( i - 1 \), we have:

\[
\partial h_{i-1}(x_{i-1} - x_i) \in \left[ - \sum_{j=1}^{i-1} f'_j:R(u_{kkt,j}), - \sum_{j=1}^{i-1} f'_j:R(l_{kkt,j}) \right] .
\]
As a result, according to the $i$th 0-inclusion formula, we have

$$\partial h_i(x_i - x_{i+1}) \in \left[ \partial h_{i-1}(x_{i-1} - x_i) - f'_i;R(u_{kkt,i}), \partial h_{i-1}(x_{i-1} - x_i) - f'_i;L(l_{kkt,i}) \right]$$

$$\subseteq \left[ -\sum_{j=1}^{i} f'_j;R(u_{kkt,j}), -\sum_{j=1}^{i} f'_j;L(l_{kkt,j}) \right].$$

Hence inequality (25) holds for $i$. Further, due to the convexity of $h_i$, we have

$$x_{i+1} \in \left[ x_i - (\partial h_i)_{R}^{-1} \left( -\sum_{j=1}^{i} f'_j;L(l_{kkt,j}) \right), x_i - (\partial h_i)_{L}^{-1} \left( -\sum_{j=1}^{i} f'_j;R(u_{kkt,j}) \right) \right]$$

$$\subseteq \left[ l_{kkt,i} - (\partial h_i)_{R}^{-1} \left( -\sum_{j=1}^{i} f'_j;L(l_{kkt,j}) \right), u_{kkt,i} - (\partial h_i)_{L}^{-1} \left( -\sum_{j=1}^{i} f'_j;R(u_{kkt,j}) \right) \right].$$

Therefore Eq. (24) holds for $i$ as well. \qed

Similar to the analysis of the dsc-algorithm, the last 0-inclusion formula yet to satisfy is the $n$th one: $0 \in f'_n(x_n) - \partial h_{n-1}(x_{n-1} - x_n)$. By the proof in Lemma 7, we have inequality (25) holds for $n - 1$:

$$\partial h_{n-1}(x_{n-1} - x_n) \in \left[ -\sum_{j=1}^{n-1} f'_j;R(u_{kkt,j}), -\sum_{j=1}^{n-1} f'_j;L(l_{kkt,j}) \right].$$

Hence

$$f'_n(x_n) - \partial h_{n-1}(x_{n-1} - x_n) \in \left[ f'_n(x_n) + \sum_{j=1}^{n-1} f'_j;L(l_{kkt,j}), f'_n(x_n) + \sum_{j=1}^{n-1} f'_j;R(u_{kkt,j}) \right]$$

$$\subseteq \left[ \sum_{j=1}^{n} f'_j;L(l_{kkt,j}), \sum_{j=1}^{n} f'_j;R(u_{kkt,j}) \right].$$

Therefore, to check whether the last 0-inclusion formula holds, it is equivalent to check whether

$$0 \in \left[ \sum_{j=1}^{n} f'_j;L(l_{kkt,j}), \sum_{j=1}^{n} f'_j;R(u_{kkt,j}) \right].$$

(26)

If inequality (26) holds, then we conclude that the given value of $x_1$ is an optimal value of $x_1$. Otherwise, similar to Corollary 4 for the dsc-algorithm, thanks to convexity, we have the following extended monotonicity property.
Corollary 8 (Extended Monotonicity Property) Let \( x_1^{(1)} < x_1^{(2)} \) be any two given values of variable \( x_1 \). Let

\[
\left( [l_{kkt,1}^{(1)}, u_{kkt,1}^{(1)}], [l_{kkt,2}^{(1)}, u_{kkt,2}^{(1)}], \ldots, [l_{kkt,n}^{(1)}, u_{kkt,n}^{(1)}], \right) \\
\left( [l_{kkt,1}^{(2)}, u_{kkt,1}^{(2)}], [l_{kkt,2}^{(2)}, u_{kkt,2}^{(2)}], \ldots, [l_{kkt,n}^{(2)}, u_{kkt,n}^{(2)}], \right),
\]

be the respective sequence of ranges of \( x_i \) values determined by \( x_1^{(1)} \) and \( x_1^{(2)} \) according to Eq. (24). Then we have, for all \( i = 1, \ldots, n \):

\[
l_{kkt,i}^{(1)} < l_{kkt,i}^{(2)}, \\
u_{kkt,i}^{(1)} < u_{kkt,i}^{(2)},
\]

(27)

(28)

Proof The proof is by a straightforward induction on \( i \). We prove for inequalities (27). Inequalities (28) are proved in a similar way.

The inequality is true for \( i = 1 \) because \( l_{kkt,1}^{(1)} = x_1^{(1)} < x_1^{(2)} = l_{kkt,1}^{(2)} \). Suppose the result is true for all \( j = 1, \ldots, i \) for some \( i (1 \leq i \leq n - 1) \). We prove that it is also true for \( i + 1 \). By the induction hypothesis and the convexity of functions \( f_j \), we have

\[
\sum_{j=1}^{i} f_{j;L}(l_{kkt,j}^{(1)}) \leq \sum_{j=1}^{i} f_{j;L}(l_{kkt,j}^{(2)}).
\]

Then by the convexity of function \( h_i \), we have

\[
-(\partial h_i)_R^{-1} \left( -\sum_{j=1}^{i} f_{j;L}(l_{kkt,j}^{(1)}) \right) \leq -(\partial h_i)_R^{-1} \left( -\sum_{j=1}^{i} f_{j;L}(l_{kkt,j}^{(2)}) \right).
\]

Therefore,

\[
l_{kkt,i+1}^{(1)} = l_{kkt,i}^{(1)} - (\partial h_i)_R^{-1} \left( -\sum_{j=1}^{i} f_{j;L}(l_{kkt,j}^{(1)}) \right) \\
< l_{kkt,i}^{(2)} - (\partial h_i)_R^{-1} \left( -\sum_{j=1}^{i} f_{j;L}(l_{kkt,j}^{(2)}) \right) = l_{kkt,i+1}^{(2)}.
\]

\( \Box \)

Combining Lemma 7 and Corollary 8 gives a binary search algorithm to find an optimal value of \( x_1 \). In every step, we try a value of \( x_1 \) and compute the endpoints of the ranges \([l_{kkt,i}, u_{kkt,i}]\) for \( i = 1, \ldots, n \) based on Eq. (24). Then we compute the two quantities \( \sum_{i=1}^{n} f'_{i;L}(l_{kkt,i}) \) and \( \sum_{i=1}^{n} f'_{i;R}(u_{kkt,i}) \). If \( \sum_{i=1}^{n} f'_{i;L}(l_{kkt,i}) \leq 0 \leq \sum_{i=1}^{n} f'_{i;R}(u_{kkt,i}) \), then the current value of \( x_1 \) is optimal to 1D-Denoise. Otherwise,
if $\sum_{i=1}^n f'_{i;R}(ukkt,i) < 0$, we check a larger value of $x_1$, or if $\sum_{i=1}^n f'_{i;L}(ukkt,i) > 0$, we check a smaller value of $x_1$. The binary search efficiently determines an optimal value of $x_1$, in time complexity of $O(n \log^2 \frac{U}{\epsilon})$, where one $O(\log \frac{U}{\epsilon})$ term indicates the number of iterations in the binary search, and the other $O(\log \frac{U}{\epsilon})$ term indicates the complexity of subgradient-to-argument inverse on regularization functions. After $x_1^*$ is found, we plug it into 1D-Denoise (1) to reduce the problem from $n$ variables to $n - 1$ variables of the same form. In the reduced 1D-Denoise problem, $f_1(x_1^*)$ is removed since it is a constant, and the deviation function of $x_2$ becomes $f_2(x_2) + h_1(x_1^* - x_2)$. Thus we can repeat the above process to find an optimal solution of $x_2, x_3^*$. As a result, it requires $n$ iterations to solve an optimal solution $(x_1^*, \ldots, x_n^*)$ for 1D-Denoise. In the $i$th iteration, $x_i^*$ is solved and the problem 1D-Denoise (1) is reduced to a smaller problem, of the same form, with one less variable.

We first present a subroutine in Box 2 that solves an optimal value of $x_i$ on the reduced 1D-Denoise problem of $n - i + 1$ variables, $x_i, x_{i+1}, \ldots, x_n$, with fidelity functions $f_i(x_i), \ldots, f_n(x_n)$, and regularization functions $h_i(x_i - x_{i+1}), \ldots, h_{n-1}(x_{n-1} - x_n)$. The values of $x_1, \ldots, x_{i-1}$ are assumed fixed.

$$
x_i^* := \text{SOLVE_REDUCE_1D-Denoise}((f_i, \ldots, f_n), (h_i, \ldots, h_{n-1}))
$$

**Step 0:** If $i = n$, solve $x_n^* := \arg \min_{x_n} -U \leq x_n \leq U f_n(x_n)$ to $\epsilon$-accuracy by binary search and return. Otherwise initialize the lower and upper bounds of the search region for $x_i$ as $l := -U$ and $u := U$ respectively.

**Step 1:** Set $x_i := \frac{l+u}{2}$. Set $l_{kk}, i := u_{kk}, i := x_i$. Compute the intervals $[l_{kk}, j+1, u_{kk}, j+1]$ for $j = i, \ldots, n - 1$ based on the following equations:

$$
\begin{align*}
\{ l_{kk}, j+1 & := l_{kk}, j + z_j; L, \\
u_{kk}, j+1 & := u_{kk}, j + z_j; R,
\end{align*}
$$

where

$$
\begin{align*}
z_j; L & := -(\partial h_j)_R^{-1} \left( -\sum_{p=i}^j f'_{p;L}(l_{kk}, p) \right), \\
z_j; R & := -(\partial h_j)_L^{-1} \left( -\sum_{p=i}^j f'_{p;R}(u_{kk}, p) \right).
\end{align*}
$$

(29)

**Step 2:** If $u - l < \epsilon$ or $\sum_{j=i}^n f'_{j;L}(l_{kk}, j) \leq 0 \leq \sum_{j=i}^n f'_{j;R}(u_{kk}, j)$, return $x_i^* := x_i$ and stop.

**Step 3:** If $\sum_{j=i}^n f'_{j;R}(u_{kk}, j) < 0$, set $l := x_i$; otherwise set $u := x_i$. Go to **Step 1**.

AlgoBox 2: Subroutine to solve a reduced 1D-Denoise problem of variables $x_i, x_{i+1}, \ldots, x_n$.

With the above subroutine, the complete cc-algorithm is in Box 3.
cc-algorithm for 1D-Denoise of arbitrary convex fidelity and regularization functions:

**Step 0:** Set $i := 1$.

**Step 1:** $x_i^* := \text{SOLVE\_REDUCED\_1D-Denoise}(\{f_i, f_{i+1}, \ldots, f_n\}, \{h_i, h_{i+1}, \ldots, h_{n-1}\})$.

**Step 2:** Set $f_{i+1}(x_{i+1}) := f_{i+1}(x_i) + h_i(x_i^* - x_i), \ i := i + 1$.

**Step 3:** If $i \leq n$, go to **Step 1**. Otherwise, return $(x_1^*, x_2^*, \ldots, x_n^*)$ and stop.

AlgoBox 3: cc-algorithm for 1D-Denoise of arbitrary convex fidelity and regularization functions.

The complexity of subroutine SOLVE\_REDUCED\_1D-Denoise is $O(n \log^2 \frac{U}{\epsilon})$, hence the total complexity of the cc-algorithm is $O(n^2 \log^2 \frac{U}{\epsilon})$. Therefore,

**Theorem 9** The 1D-Denoise problem of arbitrary convex fidelity and regularization functions is solved in $O(n^2 \log^2 \frac{U}{\epsilon})$ time.

Note that if the regularization functions $h_i$ are quadratic ($\ell_2$) or piecewise linear with few pieces, including absolute value ($\ell_1$) functions, then the complexity of subgradient-to-argument inverse is $O(1)$. As a result, the cc-algorithm can save an $O(\log \frac{U}{\epsilon})$ factor, with complexity speed-up to $O(n^2 \log \frac{U}{\epsilon})$.

6 A numerically improved implementation

In this section, we will present a numerically improved implementation of the KKT approach that combines the advantages of both the dsc-algorithm and the cc-algorithm. In addition, for special cases of $\ell_2$-TV-unweighted, $\ell_2$-TV-weighted and 1D-Laplacian, we specialize the numerically improved implementation to make it empirically faster.

While the dsc-algorithm is of attractive complexity $O(n \log^2 \frac{U}{\epsilon})$, in practice it suffers from numerical instability. This is because the algorithm sums the derivatives of $f'_1(x_1)$ to $f'_n(x_n)$, where the numerical errors resulting from the calculation of each derivative accumulate. In contrast, the cc-algorithm, of complexity $O(n^2 \log^2 \frac{U}{\epsilon})$, does not suffer from numerical instability because it repeats the binary search for every variable $x_i$, at the expense of an additional $O(n)$ complexity factor.

On the other hand, however, in the cc-algorithm, at every propagation, two sequences of quantities, $(l_{kkt,j})_j$ and $(u_{kkt,j})_j$ are computed, which incurs two times the computation over the propagation in the dsc-algorithm.

As a result, we propose a numerical implementation that combines the advantages of both the dsc-algorithm and the cc-algorithm: We repeat the binary search for every variable $x_i$, while in every propagation, we only compute one sequence of quantities — for the convex-convex class, we only compute the upper bounds $(u_{kkt,j})_j$ (one may as well compute only the lower bounds $(l_{kkt,j})_j$). This implies the following changes to SOLVE\_REDUCED\_1D\_Denoise in Box 2: In Step 1, the computations on $(z_j; L)_j$ and $(l_{kkt,j})_j$ are saved; In Step 2, the stopping criterion is left with only $u - l < \epsilon$. It’s
easy to verify that these changes do not affect the correctness of the subroutine. The
trade-off is that we save the computation in Step 1, while pay for the potential cost
of more iterations because we remove one stopping criterion in Step 2. There is one
more earning with these changes, though, that is we have one succinct code to cover
both the differentiable-strictly-convex class and the general convex-convex class.

Besides, to eliminate redundant computation, we conduct a “bound check” as fol-
lows: A lower bound and an upper bound of the optimal solution of each variable \( x_i \) are
maintained. The lower bounds are all initialized to \(-U\), and the upper bounds are all
initialized to \( U \), for all variables \( x_i \). The bounds are dynamically updated through the
propagation computations. And during each propagation, we check if each propagated
value \( x_i \) violates its latest bounds – if violation happens, the current propagation stops
(saving the remaining propagation computation on variables after \( x_i \) ) and the bounds
of some variables are updated. Meanwhile, the type of violation guides the next trial
value to propagate.

The pseudo-code of the numerically improved implementation, named as the KKT-
algorithm, is presented in Box 4.

---

KKT-algorithm for 1D-Denoise problem:

**input**: \( \{f_i\}_{i=1,...,n}, \{h_i\}_{i=1,...,n-1}, U, \epsilon. \)

**output**: \( \{x^*_i\}_{i=1,...,n} \), optimal solution to 1D-Denoise.

**begin**

Initialize lower bound \( l_i := -U \) and upper bound \( u_i := U \) of \( x_i \) for
\( i = 1, \ldots, n. \)

for \( i := 1, \ldots, n: \)

\{Solve \( x^*_i \}\)

Set \( f_i(x_i) := f_i(x_i) + h_{i-1}(x^*_i-1 - x_i) \) if \( i > 1; \)

\( x_i := (l_i + u_i)/2; \)

\( state := \text{propagate}(\{f_j\}_{j=i,...,n}, \{h_j\}_{j=i,...,n-1}, x_i, \{l_j, u_j\}_{j=i,...,n}); \)

while \( (u_i - l_i \geq \epsilon) \) do:

if \( state = 0 \) then

\{\( \sum_{j=i}^n f'_j(x_j) = 0, \) optimal \( x_i \) found\}

break;

else if \( state < 0 \) then

\{Try larger \( x_i \}\}

\( l_i := x_i; \)

else \( state > 0, \) try smaller \( x_i \) \}

\( u_i := x_i; \)

end if

\( x_i := (l_i + u_i)/2; \)

\( state := \text{propagate}(\{f_j\}_{j=i,...,n}, \{h_j\}_{j=i,...,n-1}, x_i, \{l_j, u_j\}_{j=i,...,n}); \)

end while

\( x^*_i := x_i; \)

end for

return \( \{x^*_i\}_{i=1,...,n}; \)

**end**
A unified approach for a 1D generalized total variation…

AlgoBox 4: KKT-algorithm for 1D-Denoise problem. This numerical implementation applies to both the differentiable-strictly-convex class and the general convex-convex class. The pseudo-code of the subroutine \textit{propagate} is shown in Box 5.

\begin{verbatim}
state := propagate([f_j]_{j=i,...,n}, [h_j]_{j=i,...,n-1}, x_i, [l_j, u_j]_{j=i,...,n})
begin
    Compute \( f' := f'_{i,R}(x_i); \)
    for \( j := i, \ldots, n - 1 \):
        Compute \( z_j;R \) from \( f' \) according to equation (29);
        \( x_{j+1} := x_j + z_j;R; \)
        if \( x_{j+1} < l_{j+1} \) then
            \{ Violate lower bound \}
            \{ Update lower bounds of \( x_i \) to \( x_j \) \}
            \( l_p := x_p, \) for \( p \in [i, j]; \)
            return \(-1; \)
        end if
        if \( x_{j+1} > u_{j+1} \) then
            \{ Violate upper bound \}
            \{ Update upper bounds of \( x_i \) to \( x_j \) \}
            \( u_p := x_p, \) for \( p \in [i, j]; \)
            return \(1; \)
        end if
        Compute \( f'_{j+1;R}(x_{j+1}); \)
        \( f' := f' + f'_{j+1;R}(x_{j+1}); \)
    end for
    if \( f' = 0 \) then
        return 0;
    else if \( f' < 0 \) then
        \{ Update lower bounds of \( x_i \) to \( x_n \) \}
        \( l_p := x_p, \) for \( p \in [i, n]; \)
        return \(-1; \)
    else \( f' > 0 \)
        \{ Update upper bounds of \( x_i \) to \( x_n \) \}
        \( u_p := x_p, \) for \( p \in [i, n]; \)
        return \(1; \)
    end if
end
\end{verbatim}

AlgoBox 5: Pseudo-code of subroutine \textit{propagate} for the KKT-algorithm displayed in Box 4. The subroutine returns a \textit{state} indicating the direction to try next \( x_i \) value. Values of the bounds, \([l_j, u_j]_{j=i,...,n}\), will be updated.
7 Experimental study

We implement in C++ the KKT-algorithm in Box 4 and compare our implementation with existing solvers for the special cases of 1D-Denoise problem (1) discussed in Sect. 1.1. For $\ell_2$-TV-unweighted problem (3), $\ell_2$-TV-weighted problem (4), $\ell_1$-TV problem (5) and $\ell_2$-Tikhonov problem (6), we compared our implementation with efficient specialized C++ solvers [2,4,5,14,16] and the experimental results showed that our algorithm is not advantageous. For $\ell_p-\ell_q$ problem (7) and problems with Huber objective functions (9), the work of [21] does not provide C++ implementation of their algorithm, nor are we aware of other publicly available specialized C++ solvers for these classes of problems. As a result, we compare our implementation with popular C++ nonlinear optimization solver softwares in solving the $\ell_p-\ell_q$ problem (7) and two problems with Huber objective functions. Those solvers solve the three problems using first-order methods, where we feed in objective values and gradient values of the problems to the solvers. We find out that our solver is much faster than the nonlinear optimization solvers in solving the three problems.

The two problems with Huber objective functions we compare in the experiment are:

$$\begin{align*}
(\text{Huber-TV}) \quad & \min_{x_1, \ldots, x_n} \sum_{i=1}^{n} c_i \rho_{k_i} (x_i - a_i) + \sum_{i=1}^{n-1} c_{i,i+1}|x_i - x_{i+1}|, \\
(\ell_2\text{-Huber}) \quad & \min_{x_1, \ldots, x_n} \frac{1}{2} \sum_{i=1}^{n} c_i (x_i - a_i)^2 + \sum_{i=1}^{n-1} c_{i,i+1} \rho_{k_i,i+1} (x_i - x_{i+1}).
\end{align*}$$

Huber-TV problem (30) has Huber functions in the fidelity terms and $\ell_2$-Huber problem (31) has Huber functions in the regularization terms.

We compare the KKT-algorithm with the following three popular C++ nonlinear optimization solvers:

1. Ceres solver\(^2\): This solver is provided by Google and has been used in production at Google since 2010. It is a popular optimization solver for robotics and other areas in industry. It provides API for modeling and solving general unconstrained minimization problems, which suits for our cases here.
2. NLopt solver\(^3\): This solver is an open source library for nonlinear optimization that provides many different algorithms and low-level code optimization.
3. dlib solver\(^4\): This solver is an open source library for machine learning algorithms and tools. It contains general purpose unconstrained nonlinear optimization algorithms that are suitable for our cases here.

The three solvers provide different first-order algorithms to solve unconstrained nonlinear optimization problems, which require the objective value and gradient value of a problem to be fed in. For each of the three solvers, we ran multiple different

---

\(^2\) https://ceres-solver.googlesource.com/ceres-solver.
\(^3\) https://github.com/stevengj/nlopt.
\(^4\) https://github.com/davisking/dlib.
first-order methods they provide and recorded each method’s running time and output objective value upon the method stops. These records provide a range of running times and objective values for each solver. We compare the running times and objective values output from the KKT-algorithm with those ranges of each solver respectively. For objective value comparison, rather than reporting the raw objective values, we report the relative objective value gap, which uses the objective value of the compared solver minus the objective value of the KKT-algorithm, then divided by the objective value of the KKT-algorithm. For the three problems considered here, as the objective values are always positive, hence a positive relative objective value gap implies that the KKT-algorithm gives a better solution upon stopping, while a negative relative objective value gap implies that, upon the algorithms stop, the nonlinear optimization solver that we compare to gives a better solution.

In nonlinear optimization, stopping criteria need to be specified for each numerical algorithm, including ours. The stopping criteria of the KKT-algorithm and the three solvers are set as follows:

1. KKT-algorithm: The stopping criterion is that the gap between the lower bound \(\ell_i\) and \(u_i\) for each variable \(x_i\) is less than \(\epsilon = 10^{-6}\) (See the algorithm pseudo-code in Box 4).
2. Ceres solver: The stopping criteria are that either (i) the maximum difference between two consecutive solution vectors is less than \(10^{-6}\), or (ii) a running time limit of 5 minutes, whichever reaches first.
3. NLopt solver: Same as Ceres solver.
4. dlib solver: dlib solver does not provide the above two stopping criteria. Instead, it provides the objective value change stopping criterion: we set that if in successive iterations, the objective value change is less than \(10^{-2}\), the algorithm stops and output results.

In all the three problems studied, each value of \(c_i\) and \(c_{i,i+1}\) is sampled uniformly from \((0, 1)\), each value of \(a_i\) is sampled uniformly from \((-1, 1)\). For Huber-TV problem (30), each \(k_i\) value is determined as \(k_i = b_i |a_i|\), where coefficient \(b_i\) is sampled uniformly from \((0.5, 1.0)\). Similarly, for \(\ell_2\)-Huber problem (31), each \(k_{i,i+1}\) value is determined as \(k_{i,i+1} = b_{i,i+1} |a_i - a_{i+1}|\), where coefficient \(b_{i,i+1}\) is sampled uniformly from \((0.5, 1.0)\) as well.

We run the experiment on a MacBook Pro laptop with Intel Core i7 2.2 GHz processor. For each of the three problems, we run the KKT-algorithm and the three solvers with the number of variables, \(n\), increasing from \(10^2\) to \(10^7\) by a factor of 10. For each \(n\), we randomly generate 5 problem instances following the above parameter generation scheme, and record the running times and output objective values upon stopping, for the KKT-algorithm and the three solvers respectively. We find out that in each run, the KKT-algorithm always gives a smaller objective value in a much shorter time, over all compared methods provided by the three solvers. In the following tables, we report averaged results, including averaged running time of the KKT-algorithm, averaged range of running times of the three solvers, and averaged range of relative objective value gaps of the three solvers to the KKT-algorithm. We note that, for all the problems tested, for the three solvers, the algorithm achieving the shortest running time and the algorithm achieving the best objective value are often not the same.
The experimental results for solving the $\ell_p - \ell_q$ problem (7) with different solvers are shown in Table 1 for running times and Table 2 for objective value comparison.

For the $\ell_p - \ell_q$ problem, one can see that all solvers have similar running times for $n = 10^2$. For $n$ increasing from $10^3$ to $10^7$, the KKT-algorithm is from 3 to 7 times faster than Ceres and from 10 to 25 times faster than NLopt. For dlib, except for the case of $n = 10^3$, the KKT-algorithm is from 1.4 to 8 times faster. The gaps between objective values are not significant.

The experimental results for solving the Huber-TV problem (30) with different solvers are shown in Table 3 for running times and Table 4 for objective value comparison.
A unified approach for a 1D generalized total variation...

Table 3  Average running time (in cpu seconds) of the KKT-algorithm and the three nonlinear optimization solvers for Huber-TV problem (30) of increasing number of variables, n

| Huber-TV | $10^2$ | $10^3$ | $10^4$ | $10^5$ | $10^6$ | $10^7$ |
|----------|--------|--------|--------|--------|--------|--------|
| KKT      | 0      | 0      | 0.0040 | 0.050  | 0.48   | 4.78   |
| Ceres    |        |        |        |        |        |        |
| Min      | 0      | 0.0024 | 0.032  | 0.38   | 5.05   | 56.23  |
| Max      | 0.0066 | 0.219  | 3.71   | 28.75  | 300.18 | 302.04 |
| NLopt    |        |        |        |        |        |        |
| Min      | 0.0012 | 0.018  | 0.074  | 0.71   | 6.45   | 60.71  |
| Max      | 0.68   | 60.70  | 180.74 | 234.20 | 296.67 | 301.32 |
| dlib     |        |        |        |        |        |        |
| Min      | 0      | 0.012  | 0.56   | 8.23   | 116.06 | 1551.07|
| Max      | 0.0064 | 0.036  | 0.65   | 10.66  | 218.69 | 3830.23|

For the three solvers, the averaged minimum and maximum running times over each solver’s different algorithms are reported. All nonzero numbers are rounded to two significant digits after the decimal points.

Table 4  Average minimum and maximum relative objective value gaps from the three nonlinear optimization solvers to the KKT-algorithm for Huber-TV problem (30) of increasing number of variables, n

| Huber-TV | $10^2$ (%) | $10^3$ (%) | $10^4$ (%) | $10^5$ (%) | $10^6$ (%) | $10^7$ (%) |
|----------|------------|------------|------------|------------|------------|------------|
| KKT      | –          | –          | –          | –          | –          | –          |
| Ceres    |            |            |            |            |            |            |
| Min      | 5.73       | 2.11       | 1.17       | 2.43       | 2.09       | 9.40       |
| Max      | 54.21      | 39.51      | 39.08      | 38.84      | 38.98      | 38.98      |
| NLopt    |            |            |            |            |            |            |
| Min      | 3.03       | 0.57       | 0.27       | 0.78       | 3.76       | 11.77      |
| Max      | 32.60      | 35.12      | 44.06      | 44.15      | 44.47      | 44.43      |
| dlib     |            |            |            |            |            |            |
| Min      | 27.42      | 11.13      | 2.10       | 1.68       | 1.59       | 1.55       |
| Max      | 36.26      | 23.97      | 23.53      | 23.76      | 23.92      | 23.91      |

All numbers are rounded to two significant digits after the decimal points. We use “–” in the second row of the table to denote that the relative objective value gaps are computed against the objective values of the KKT-algorithm.

For the Huber-TV problem (30), the four solvers all have negligible running times for $n = 10^2$. For $n = 10^3$, the running time of the KKT-algorithm and the minimum running time of Ceres remain negligible, but the minimum running times of NLopt and dlib are several times slower than Ceres. From $n = 10^4$ to $n = 10^7$, the KKT-algorithm is from 8 to 10 times faster than Ceres, from 12 to 18 times faster than NLopt, and from 160 to 324 times faster than dlib. The relative objective value gaps are overall much more significant for the Huber-TV problem than for the $\ell_p-\ell_q$ problem. One reason might be that the TV regularization terms are not strictly convex and not differentiable at the point 0. Ceres and NLopt both have the largest relative objective value gaps at
Table 5 Average running time (in cpu seconds) of the KKT-algorithm and the three nonlinear optimization solvers for $\ell_2$-Huber problem (31) of increasing number of variables, $n$

| $\ell_2$-Huber | $10^2$ | $10^3$ | $10^4$ | $10^5$ | $10^6$ | $10^7$ |
|----------------|--------|--------|--------|--------|--------|--------|
| KKT           | 0      | 0.0012 | 0.010  | 0.11   | 1.01   | 10.12  |
| Ceres         | Min    | 0      | 0.0020 | 0.025  | 0.30   | 4.20   |
|               | Max    | 0      | 0.0056 | 0.037  | 0.397  | 5.32   |
| NLopt         | Min    | 0.00040| 0.0094 | 0.19   | 1.75   | 24.59  |
|               | Max    | 0.0052 | 0.032  | 0.43   | 6.25   | 76.17  |
| dlib          | Min    | 0      | 0      | 0.013  | 0.23   | 3.57   |
|               | Max    | 0      | 0.0038 | 0.068  | 0.84   | 12.48  |

For the three solvers, the averaged minimum and maximum running times over each solver’s different algorithms are reported. All nonzero numbers are rounded to two effective digits after the decimal points.

Table 6 Average minimum and maximum relative objective value gaps from the three nonlinear optimization solvers to the KKT-algorithm for $\ell_2$-Huber problem (31) of increasing number of variables, $n$

| $\ell_2$-Huber | $10^2$ (%) | $10^3$ (%) | $10^4$ (%) | $10^5$ (%) | $10^6$ (%) | $10^7$ (%) |
|----------------|------------|------------|------------|------------|------------|------------|
| KKT           | –          | –          | –          | –          | –          | –          |
| Ceres         | Min        | 1.98       | 0.047      | 0.00076    | 0.00084    | 0.00043    | 0.00035    |
|               | Max        | 1.98       | 0.049      | 0.0016     | 0.0023     | 0.0022     | 0.0017     |
| NLopt         | Min        | 1.98       | 0.047      | 0.00051    | 0.00050    | 0.000071   | 0.0000059  |
|               | Max        | 1.98       | 0.047      | 0.00051    | 0.00050    | 0.000071   | 0.0000074  |
| dlib          | Min        | 2.10       | 0.072      | 0.0045     | 0.0012     | 0.00018    | 0.000022   |
|               | Max        | 2.20       | 0.097      | 0.0082     | 0.0012     | 0.00019    | 0.000024   |

All numbers are rounded to two effective digits after the decimal points. We use “–” in the second row of the table to denote that the relative objective value gaps are computed against the objective values of the KKT-algorithm.

The experimental results for solving the $\ell_2$-Huber problem (31) with different solvers are shown in Table 5 for running times and Table 6 for objective value comparison.

For the $\ell_2$-Huber problem (31), the four solvers all have negligible running times for $n = 10^2$ and $n = 10^3$. From $n = 10^4$ to $n = 10^7$, the KKT-algorithm is from 2.5 to 5 times faster than Ceres, from 16 to 30 times faster than NLopt, and from 1.3 to 0.000024.
5.7 times faster than dlib. Similar for the $\ell_p-\ell_q$ problem, the gaps between objective values are negligible.

8 Conclusion and future directions

In this paper we present two efficient algorithms to solve the 1D-Denoise problem (1) for different classes of objective functions. Both algorithms follow a unified approach that directly solves the KKT optimality conditions of the 1D-Denoise problem. For convex differentiable fidelity functions and strictly convex regularization functions, our dsc-algorithm has $O(n \log^2 \frac{U}{\epsilon})$ time complexity; for arbitrary convex fidelity and regularization functions, our cc-algorithm has $O(n^2 \log^2 \frac{U}{\epsilon})$ time complexity. The numerically improved algorithm, KKT-algorithm, that combines the advantages of both the dsc-algorithm and the cc-algorithm, is presented and implemented in C++. For complicated objective functions, such as higher order $\ell_p$ functions of $p > 2$ and Huber functions, the KKT-algorithm has much better performance over existing popular nonlinear optimization solvers.

There are many directions that could potentially make use of the results or push further research based on the ideas presented here. The 1D-Denoise problem considered here only penalizes the first order differences, $x_i - x_{i+1}$. It would be interesting to study if a similar technique can be applied to solve problems that penalize higher order differences, such as second order difference $x_{i-1} - 2x_i + x_{i+1}$, which appears in some trend filtering applications [15]. On the other hand, it would be interesting to study if we can use the algorithms here as subroutines to solve higher dimensional denoising problems, such as 2D denoising problems in imaging [2,16,21]. From a graph-theoretic perspective, the 1D-Denoise problem is defined on a path graph. It would be interesting to study whether we can adopt the algorithms here to provide a faster algorithm to solve problems defined on more general graphs, such as the MRF problem (2). Inspired by Weinmann et al.’s work in [21] on manifold-valued data, we are very interested in investigating whether our methods can be generalized from the Euclidean space to manifolds as well.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Ahuja, R.K., Hochbaum, D.S., Orlin, J.B.: Solving the convex cost integer dual network flow problem. Manag. Sci. 49(7), 950–964 (2003)
2. Barbero, Á., Sra, S.: Modular proximal optimization for multidimensional total-variation regularization. J. Mach. Learn. Res. 19(1), 2233–2313 (2018)
3. Barlow, R.E., Bartholomew, D.J., Brenner, J.M., Brunk, H.D.: Statistical Inference Under Order Restrictions: The Theory and Application of Isotonic Regression. Wiley, New York (1972)
4. Condat, L.: A direct algorithm for 1-D total variation denoising. IEEE Signal Process. Lett. 20(11), 1054–1057 (2013)
5. Conte, S.D., de Boor, C.: Elementary Numerical Analysis. McGraw-Hill, New York (1972)
6. Davies, P.L., Kovac, A.: Local extremes, runs, strings and multiresolution. Ann. Stat. 29(1), 1–65 (2001)
7. Dümbgen, L., Kovac, A.: Extensions of smoothing via taut strings. Electron. J. Stat. 3, 41–75 (2009)
8. Grasmair, M.: The equivalence of the taut string algorithm and BV-regularization. J. Math. Imag. Vis. 27(1), 59–66 (2007)
9. Hochbaum, D.S.: An efficient algorithm for image segmentation, Markov random fields and related problems. J. ACM 48(4), 686–701 (2001)
10. Hochbaum, D.S.: Multi-label Markov random fields as an efficient and effective tool for image segmentation, total variations and regularization. Numer. Math. Theory Methods Appl. 6(1), 169–198 (2013)
11. Hochbaum, D.S., Lu, C.: A faster algorithm solving a generalization of isotonic median regression and a class of fused lasso problems. SIAM J. Optim. 27(4), 2563–2596 (2017)
12. Hochbaum, D.S., Shanthikumar, J.G.: Nonlinear separable optimization is not much harder than linear optimization. J. ACM 37(4), 843–862 (1990)
13. Huber, P.: Robust estimation of a location parameter. Ann. Math. Stat. 35(1), 73–101 (1964)
14. Johnson, N.A.: A dynamic programming algorithm for the fused lasso and $L_0$-segmentation. J. Comput. Graph. Stat. 22(2), 246–260 (2013)
15. Kim, S.-J., Koh, K., Boyd, S., Gorinevsky, D.: $\ell_1$ trend filtering. SIAM Rev. 51(2), 339–360 (2009)
16. Kolmogorov, V., Pock, T., Rolinek, M.: Total variation on a tree. SIAM J. Imag. Sci. 9(2), 605–636 (2016)
17. Little, M.A., Jones, N.S.: Generalized methods and solvers for noise removal from piecewise constant signals I Background theory. Proc. R. Soc. A Math. Phys. Eng. Sci. 467(2135), 3088–3114 (2011)
18. Mammen, E., van de Geer, S.: Locally adaptive regression splines. Ann. Stat. 25(1), 387–413 (1997)
19. Rudin, L.I., Osher, S., Fatemi, E.: Nonlinear total variation based noise removal algorithms. Physica D 60(1–4), 259–268 (1992)
20. Storath, M., Weinmann, A., Unser, M.: Exact algorithms for $L^1$-TV regularization of real-valued or circle-valued signals. SIAM J. Sci. Comput. 38(4), A614–A630 (2016)
21. Weinmann, A., Demaret, L., Storath, M.: Total variation regularization for manifold-valued data. SIAM J. Imag. Sci. 7(4), 2226–2257 (2014)

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.