Yokohama Publishers
ISSN 1349-8169 ONLINE JOURNAL

Pacific Journal of Optimization
Vol. 19, No. 2, pp. 187–204, 2023

SPARSE HESSIAN BASED SEMISMOOTH NEWTON AUGMENTED LAGRANGIAN ALGORITHM FOR GENERAL ℓ_1 TREND FILTERING*

Yong-Jin Liu† and Tiqi Zhang

Abstract: This paper investigates a semismooth Newton based augmented Lagrangian (Ssnal) algorithm for solving equivalent formulation of the general ℓ_1 trend filtering problem. The computational costs of a semismooth Newton (Ssn) algorithm for solving the subproblem in the Ssnal algorithm can be substantially reduced by exploiting the second order sparsity of Hessian matrix and some efficient techniques. The global convergence and the asymptotically superlinear local convergence of the Ssnal algorithm are given under mild conditions. Numerical comparisons between the Ssnal algorithm and other state-of-the-art algorithms on real and synthetic data sets validate that our algorithm has superior performance in both robustness and efficiency.

Key words: *general ℓ_1 trend filtering, semismooth Newton algorithm, augmented Lagrangian algorithm, sparse Hessian*

Mathematics Subject Classification: *90C06, 90C25, 90C31*

1 Introduction

For a given (noisy) signal $y = (y_1, ..., y_n) \in \mathbb{R}^n$ and a constant $k \geq 1$, this paper intends to propose an efficient algorithm to solve the general ℓ_1 trend filtering problem as follows:

$$x = \arg\min_{x \in \mathbb{R}^n} \frac{1}{2}\|x - y\|^2 + \lambda \|D^{(k,n)} x\|_1, \tag{1.1}$$

where $\|\cdot\|_1$ denotes the ℓ_1 norm, the tuning parameter $\lambda > 0$ provides a tradeoff between fidelity to measurements and noise sensitivity, and $D^{(k,n)} \in \mathbb{R}^{(n-k)\times n}$ is the kth order difference matrix on $\mathbb{R}^n$, which can be defined recursively as

$$D^{(k,n)} = D^{(1,n-k+1)} D^{(k-1,n)}, \quad k \geq 2,$$

where $D^{(1,p)} \in \mathbb{R}^{(p-1)\times p}$ is the first order difference matrix on $\mathbb{R}^p$

$$D^{(1,p)} = \begin{bmatrix} 1 & -1 & & & \\ & 1 & -1 & & \\ & & & \ddots & \\ & & & 1 & -1 \end{bmatrix}.$$

*This work was supported by the National Natural Science Foundation of China (Grant No. 11871153) and the Natural Science Foundation of Fujian Province of China (Grant No. 2019J01644).

†Corresponding author

Nonparametric regression is a popular field with many celebrated tools. Trend filtering, a recently proposed tool for nonparametric regression, is defined as the minimizer of a penalized least squares criterion. Trend filtering arises in a wide of applications including macroeconomics [10, 33], social sciences [14], financial time series analysis [36], revenue management [34], and so on [1, 9]. For solving some practical economic problems, a number of linear trend filtering methods have been presented, see, e.g., the exponential smoothing method [20], the moving average filtering method [25] and the Hodrick-Prescott(H-P) filtering method [10]. Kim et al. [12] proposed the ℓ_1 trend filtering, i.e., a variation on H-P filtering which substitutes a sum of absolute values (i.e., an ℓ_1 norm) for the sum of squares used in H-P filtering to penalize variations in the estimated trend. Furthermore, this idea can be extended to fitting a piecewise polynomial of degree $k-1$ to the data, i.e., general ℓ_1 trend filtering.

When $k = 1$, general ℓ_1 trend filtering becomes 1-dimensional total variation denoising [32], also called the 1-dimensional fused lasso problem [35], which admits an explicit solution. There are some direct, linear time algorithms for 1-dimensional total variation denoising problem, such as a taut string principle [5], an entirely different dynamic programming approach [11] and 1D total variation denoising algorithm [4]. However, these algorithms cannot be directly extended to the higher order cases $k \geq 2$. Therefore, some iterative algorithms have been developed to solve higher order ℓ_1 trend filtering, including the alternating direction method of multipliers (ADMM) [26] and the primal dual interior point (PDIP) algorithm [12].

In this paper, we intend to design a sparse second order information based efficient algorithm to solve general ℓ_1 trend filtering (1.1). The superior performances of the SSNAL algorithm applied to solve some large-scale problems including support vector machines [24], Lasso problems [17, 19, 15, 16, 37], Dantzig selector [6], SLBoxLSR problem [18], OSCAR and SLOPE models [21] have been demonstrated. The highlights of the SSNAL algorithm lie in the attractive convergence property and the low computational costs in the SSN algorithm by exploiting the second order sparsity and some efficient techniques. Based on the success of previous work, we aim to develop the SSNAL algorithm to find the optimal solution of general ℓ_1 trend filtering.

We summarize main contributions of this paper as follows. Firstly, we reformulate the general ℓ_1 trend filtering (1.1) and then apply the SSNAL algorithm to solve it. Secondly, we validate the global and local convergence property of the SSNAL algorithm under very mild conditions. In addition, the computational costs of the SSN algorithm for solving the subproblem involved in the SSNAL algorithm can be cheap by exploiting the second-order sparsity of the Hessian matrix and some efficient computation techniques. Thirdly, we demonstrate the numerical results on synthetic and real data sets by comparing our algorithm against other algorithms including ADMM and PDIP algorithms.

The rest of this paper is organized as follows. Section 2 develops the SSNAL algorithm for solving the reformulation of the general ℓ_1 trend filtering problem (1.1) and shows its global and local convergence. We then employ the second order sparsity of Hessian and some efficient techniques to solve inner problem of the SSNAL algorithm. In Section 3, we conduct extensive numerical experiments on the synthetic and real data sets by comparing our proposed algorithm with other state-of-the-art algorithms and report their performances. In Section 4, we make final conclusions to close this paper.

Notation. We define the l_∞ norm unit ball by $B_\infty := \{x \in \mathbb{R}^m |\ \|x\|_\infty \leq 1\}$. For any positive integer n, I_n is the $n \times n$ identity matrix, and $\mathbf{1}_m$ is the vector of all ones. For any $x \in \mathbb{R}^n$, " Diag(x) " denotes the diagonal matrix whose diagonal is the vector x, " $|x|$ " denotes the absolute vector whose i-th entry is $|x_i|$, and " sign(x) " denotes the sign vector

whose i-th entry is 1 if $x_i > 0$, -1 if $x_i < 0$, and 0 otherwise. The Fenchel conjugate of a proper convex function f is defined as $f^*(s) = \sup_{x\in\mathbb{R}^n}\{\langle x, s\rangle - f(x)\}, \forall s \in \mathbb{R}^n$ and $\odot$ denotes the Hadamard product.

2 A semismooth Newton based augmented Lagrangian algorithm

In this section, we employ an efficient semismooth Newton based augmented Lagrangian (SSNAL) algorithm to solve the equivalent form of general ℓ_1 trend filtering problem (1.1).

We first reformulate problem (1.1) as the following form:

$$\begin{aligned} \min_{x,z} \quad & \frac{1}{2}\|x-y\|^2 + \lambda\|z\|_1 \\ \text{s.t.} \quad & D^{(k,n)}x - z = 0. \end{aligned} \tag{P}$$

The Lagrangian function of problem (P) is given by

$$l(x, z; \mu) := \frac{1}{2}\|x-y\|^2 + \lambda\|z\|_1 + \langle \mu, D^{(k,n)}x - z\rangle.$$

The Karush-Kuhn-Tucker (KKT) optimality conditions for problem (P) are given as follows:

$$x - y + {D^{(k,n)}}^T\mu = 0,\ 0 \in \lambda\partial\|z\|_1 - \mu,\ D^{(k,n)}x - z = 0. \tag{2.1}$$

The dual of problem (P) has the following form:

$$\begin{aligned} \min_{\mu} \quad & \frac{1}{2}\|{D^{(k,n)}}^T\mu\|^2 - \langle {D^{(k,n)}}^T\mu, y\rangle \\ \text{s.t.} \quad & \|\mu\|_\infty \le \lambda. \end{aligned} \tag{D}$$

For given $\sigma > 0$, we obtain the augmented Lagrangian function of problem (P) :

$$\mathcal{L}_\sigma(x, z; \mu) := \frac{1}{2}\|x-y\|^2 + \lambda\|z\|_1 + \langle \mu, D^{(k,n)}x - z\rangle + \frac{\sigma}{2}\|D^{(k,n)}x - z\|^2. \tag{2.2}$$

2.1 A semismooth Newton based augmented Lagrangian algorithm for problem (P)

Since the proposed algorithm combines a semismooth Newton algorithm and an inexact augmented Lagrangian algorithm, we are able to call it as SSNAL. The framework of SSNAL algorithm is given as follows:

Algorithm 1 (SSNAL)A semismooth Newton based augmented Lagrangian algorithm for problem (P)

Input: $\sigma_0 > 0$, $\lambda > 0$, $(x^0, z^0, \mu^0) \in \mathbb{R}^n \times \mathbb{R}^{n-k} \times \mathbb{R}^{n-k}$ and $i = 0$.

1: Compute

$$x^{i+1} \approx \operatorname*{argmin}_{x\in\mathbb{R}^n}\{\Phi_i(x) := \inf_z \mathcal{L}_{\sigma_i}(x, z; \mu^i)\} \tag{2.3}$$

to satisfy the conditions (A1) and (A2) below.

2: Compute

$$z^{i+1} = \mathrm{Prox}_{\sigma_i^{-1}\lambda\|\cdot\|_1}(D^{(k,n)}x^{i+1} + \sigma_i^{-1}\mu^i).$$

3: Compute

$$\mu^{i+1} = \mu^i + \sigma_i(D^{(k,n)}x^{i+1} - z^{i+1}).$$

4: Update $\sigma_{i+1} \uparrow \sigma_\infty \le +\infty$, $i \leftarrow i+1$, and go to step 1.

We get the approximate solution x^{i+1} of problem (2.3) under the following stopping criteria discussed in [29, 30]:

$$\|\nabla\Phi_i(x^{i+1})\| \le \frac{\epsilon_i}{\sqrt{\sigma_i}}, \sum_{i=0}^{\infty}\epsilon_i < \infty, \tag{A1}$$

$$\|\nabla\Phi_i(x^{i+1})\| \le c_i\sqrt{\sigma_i}\|D^{(k,n)}x^{i+1} - z^{i+1}\|, \sum_{i=0}^{\infty} c_i < \infty, \tag{A2}$$

$$\|\nabla\Phi_i(x^{i+1})\| \le c_i'\|D^{(k,n)}x^{i+1} - z^{i+1}\|, 0 \le c_i' \to 0, \tag{A2'}$$

where $\{c_i\}$ and $\{\epsilon_i\}$ are given nonnegative error tolerance sequences.

For the objective function g of problem (D) and the Lagrangian function l of problem (P), we define the maximal monotone operators $\mathcal{T}_g$ and $\mathcal{T}_l$ [28, 29] by

$$\mathcal{T}_g(\mu) := \partial g(\mu),\ \mathcal{T}_l(x,z;\mu) := \{(x',z';\mu')|(x',z';-\mu') \in \partial l(x,z;\mu)\}.$$

Their inverse are given as the following form:

$$\mathcal{T}_g^{-1}(\mu) = \partial g^*(\mu),\ \mathcal{T}_l^{-1}(x',z';\mu') := \arg\ \underset{x,z,\mu}{\text{minimax}}\{l(x,z;\mu) - \langle x',x\rangle - \langle z',z\rangle + \langle \mu',\mu\rangle\}.$$

Now, we shall state the global and local convergence results of the SSNAL algorithm. Since problem (P) is feasible, one knows from [29, 30] that if the stopping criterion (A1) is satisfied, the global convergence of the SSNAL algorithm for problem (P) can be guaranteed.

Theorem 2.1 (Global convergence)**.** *If $\{(x^i,z^i,\mu^i)\}$ is the infinite sequence generated by Algorithm 1 with stopping criterion* (A1)*. Then, the sequence $\{\mu^i\}$ is bounded and converges to an optimal solution of problem* (D)*. In addition, the sequence $\{(x^i,z^i)\}$ is also bounded and converges to the optimal solution (x^*,z^*) of problem* (P)*.*

In order to characterize the local convergence of the SSNAL algorithm, we give some results on error bound conditions. According to the definition of piecewise linear-quadratic functions [31], one obtains that g and l are piecewise linear-quadratic functions. One can further obtain from [31, Proposition 12.30] that $\mathcal{T}_g$, $\mathcal{T}_l$ are piecewise polyhedral multifunctions. According to [27], if F is a polyhedral multifunction, then F is locally upper Lipschitz continuous and it satisfies the error bound condition. We can conclude that $\mathcal{T}_g$ satisfies the error bound condition with a common modulus, say r_g. Especially, since the optimal solution set of problem (D), denoted by S_D, is exactly $\mathcal{T}_g^{-1}(0)$, there exists $\epsilon_g > 0$ such that if $\text{dist}(0,\mathcal{T}_g(\mu)) \le \epsilon_g$, then

$$\text{dist}(\mu, S_D) \le r_g\text{dist}(0,\mathcal{T}_g(\mu)), \tag{2.4}$$

Similarly, for the polyhedral multifunction $\mathcal{T}_l$, there exists $\epsilon_l > 0$ and $r_l > 0$ such that if $\text{dist}(0,\mathcal{T}_l(x,z;\mu)) \le \epsilon_l$, then

$$\text{dist}((x,z;\mu), S_l) \le r_l\text{dist}(0,\mathcal{T}_l(x,z;\mu)), \tag{2.5}$$

where $S_l := \{x^*,z^*\} \times S_D$ and (x^*,z^*) is the unique optimal solution of problem (P).

By virture of above analysis, combining with [22, 29, 30], we can get the result on local convergence of the SSNAL algorithm.

Theorem 2.2 (Local convergence)**.** *Let the sequence $\{(x^i,z^i,\mu^i)\}$ be the infinite sequence generated by Algorithm 1 with stopping criteria* (A1) *and* (A2)*. Then, for sufficiently large i,*

$$\text{dist}(\mu^{i+1}, S_D) \le \theta_i\text{dist}(\mu^i, S_D),$$

where $\theta_i = [r_g/\sqrt{r_g^2+\sigma_i^2}+2c_i]/(1-c_i) \to \theta_\infty = r_g/\sqrt{r_g^2+\sigma_\infty^2} < 1$ *as* $i \to \infty$.

If in addition to (A1) *and* (A2), *the stopping criterion* (A2') *is also satisfied, it holds that for sufficiently large* i,

$$\|(x^{i+1}, z^{i+1}) - (x^*, z^*)\| \le \theta_i' \|\mu^{i+1} - \mu^i\|,$$

where $\theta_i' = r_l(1+c_i')/\sigma_i \to \theta_\infty' = r_l/\sigma_\infty$.

Proof. The result of the first part is valid due to [22, Theorem 2.1]. With the above analysis, one knows that $\mathcal{T}_\ell$ satisfies the error bound condition with modulus r_ℓ. It then holds that for sufficiently large i,

$$\|(x^{i+1}, z^{i+1}) - (x^*, z^*)\| + \mathrm{dist}(\mu^{i+1}, \mathcal{S}_D) \le r_\ell \mathrm{dist}(0, \mathcal{T}_\ell(x^{i+1}, z^{i+1}; \mu^{i+1})).$$

Combining with the stopping criterion (A2') and [29, Corollary], we get the desired result of the second part. The proof is complete. □

2.2 A semismooth Newton method for the subproblem

In this subsection, we focus on an efficient semismooth Newton (SSN) algorithm to solve the inner subproblem (2.3) in the augmented Lagrangian algorithm.

For the purpose of our subsequent analysis, we first give some relevant preliminaries. Given a closed proper convex function $f : \mathbb{R} \to (-\infty, \infty]$ and any scalar $\gamma > 0$, the proximal mapping and the Moreau-Yosida regularization of γf (cf. [23]) are defined respectively by

$$\mathrm{Prox}_{\gamma f}(s) := \arg\min_{x\in\mathbb{R}^n}\{f(x) + \frac{1}{2\gamma}\|x-s\|^2\}, \forall s \in \mathbb{R}^n,$$
$$M_{\gamma f}(s) := \min_{x\in\mathbb{R}^n}\{f(x) + \frac{1}{2\gamma}\|x-s\|^2\}, \forall s \in \mathbb{R}^n.$$

The Moreau identity [28] holds, i.e.,

$$\mathrm{Prox}_{\gamma f}(s) + \gamma \mathrm{Prox}_{f^*/\gamma}(s/\gamma) = s, \forall s \in \mathbb{R}^n.$$

From [13], we know that $M_{\gamma f}(\cdot)$ is convex and continuously differentiable with its gradient given by

$$\nabla M_{\gamma f}(s) = (s - \mathrm{Prox}_{\gamma f}(s))/\gamma, \forall s \in \mathbb{R}^n.$$

Moreover, $\mathrm{Prox}_{\gamma f}(\cdot)$ and $M_{\gamma f}(\cdot)$ are globally Lipschitz continuous with modulus 1. For given $\kappa > 0$ and a closed set $\Lambda \subseteq \mathbb{R}^n$, the proximal mapping of ℓ_1 norm, also called soft-thresholding operator, is given by

$$\mathrm{Prox}_{\kappa\|\cdot\|_1}(s) = \mathrm{sign}(s) \odot \max\{|s| - \kappa \mathbf{1}_n, 0\}, \forall s \in \mathbb{R}^n.$$

In particular, if f is the indicator function of Λ, the proximal mapping of f at $s \in \mathbb{R}^n$ reduces to the projection $\Pi_\Lambda(s)$, i.e.,

$$\Pi_\Lambda(s) = \arg\min_{x\in\Lambda}\{\frac{1}{2}\|x-s\|^2\}.$$

Furthermore, the projection $\Pi_{\kappa B_\infty}(s)$ can be further expressed as

$$\Pi_{\kappa B_\infty}(s) = \mathrm{sign}(s) \odot \min\{|s|, \kappa \mathbf{1}_n\}, \forall s \in \mathbb{R}^n.$$

Now, we are ready to consider solving the subproblem (2.3). Given $\tilde{\mu} \in \mathbb{R}^n$ and $\sigma > 0$, the minimization problem is given by

$$\min_{x\in\mathbb{R}^n}\{\Phi(x) := \inf_z \mathcal{L}_\sigma(x, z; \tilde{\mu})\}, \tag{2.6}$$

where $\Phi(\cdot)$ is given by

$$\begin{aligned}\Phi(x) &= \inf_z \mathcal{L}_\sigma(x, z; \tilde{\mu}) \\ &= \frac{1}{2}\|x-y\|^2 + \inf_z\{\lambda\|z\|_1 + \langle \mu, D^{(k,n)}x - z\rangle + \frac{\sigma}{2}\|D^{(k,n)}x - z\|^2\} \\ &= \frac{1}{2}\|x-y\|^2 + \inf_z\{\lambda\|z\|_1 + \frac{\sigma}{2}\|z - (D^{(k,n)}x + \sigma^{-1}\tilde{\mu})\|^2\} - \frac{1}{2\sigma}\|\tilde{\mu}\|^2 \\ &= \frac{1}{2}\|x-y\|^2 + \sigma M_{\sigma^{-1}\lambda\|\cdot\|_1}(D^{(k,n)}x + \sigma^{-1}\tilde{\mu}) - \frac{1}{2\sigma}\|\tilde{\mu}\|^2.\end{aligned}$$

Since $M_{\sigma^{-1}\lambda\|\cdot\|_1}(\cdot)$ is strongly convex and continuously differentiable, $\Phi(\cdot)$ is continuously differentiable and convex. Hence, the problem (2.6) is equivalent to solving the following nonsmooth equations:

$$\begin{aligned}0 = \nabla\Phi(x) &= x - y + \sigma {D^{(k,n)}}^T \nabla M_{\sigma^{-1}\lambda\|\cdot\|_1}(s)\mid_{s=D^{(k,n)}x+\sigma^{-1}\tilde{\mu}} \\ &= x - y + \sigma {D^{(k,n)}}^T \mathrm{Prox}_{(\sigma^{-1}\lambda\|\cdot\|_1)^*}(D^{(k,n)}x + \sigma^{-1}\tilde{\mu}) \\ &= x - y + \sigma {D^{(k,n)}}^T \Pi_{\sigma^{-1}\lambda B_\infty}(D^{(k,n)}x + \sigma^{-1}\tilde{\mu}),\end{aligned}$$

where the second equation follows from the chain rule, the third equation holds from the Moreau identity, and the last equation is valid because the conjugate function of $\sigma^{-1}\lambda\|\cdot\|_1$ is the indicator function of $\sigma^{-1}\lambda B_\infty$. Since $\Pi_{\sigma^{-1}\lambda B_\infty}(\cdot)$ is Lipschitz continuous, the following function is well defined:

$$\hat{\partial}^2\Phi(x) := I_n + \sigma {D^{(k,n)}}^T \partial\Pi_{\sigma^{-1}\lambda B_\infty}(D^{(k,n)}x + \sigma^{-1}\tilde{\mu})D^{(k,n)}.$$

Let

$$V \in \partial\Pi_{\sigma^{-1}\lambda B_\infty}(D^{(k,n)}x + \sigma^{-1}\tilde{\mu}).$$

Then, we can easily obtain that $V = \mathrm{Diag}(v_1, ..., v_{n-k})$ with

$$v_i = \begin{cases} 1, & \text{if } |D^{(k,n)}x + \sigma^{-1}\tilde{\mu}|_i < \sigma^{-1}\lambda, \\ 0, & \text{if } |D^{(k,n)}x + \sigma^{-1}\tilde{\mu}|_i > \sigma^{-1}\lambda, \\ \in [0,1], & \text{if } |D^{(k,n)}x + \sigma^{-1}\tilde{\mu}|_i = \sigma^{-1}\lambda, \end{cases} \quad i = 1, 2, ..., n-k.$$

We know that the generalized Jacobian of $\nabla\Phi(\cdot)$ is the key to apply the SSN algorithm to solve the nonsmooth equations $\nabla\Phi(x) = 0$, however it is difficult to express it exactly. From [3, Proposition 2.3.3 and Theorem 2.6.6], we know that $\partial^2\Phi(x) \subseteq \hat{\partial}^2\Phi(x)$, $\forall x \in \mathbb{R}^n$, where $\partial^2\Phi(x)$ is the generalized Hessian of Φ at x. Therefore, we can define the following alternative for $\partial^2\Phi(x)$:

$$W := I_n + \sigma {D^{(k,n)}}^T V D^{(k,n)},$$

with $V \in \partial\Pi_{\sigma^{-1}\lambda B_\infty}(D^{(k,n)}x + \sigma^{-1}\tilde{\mu})$.

Since $\Pi_{\sigma^{-1}\lambda B_\infty}(\cdot)$ is strongly semismooth, we obtain that $\nabla\Phi(\cdot)$ is strongly semismooth. We employ a superlinearly convergent semismooth Newton algorithm to solve the nonsmooth equations $\nabla\Phi(x) = 0$. The process is described as follows:

Algorithm 2 (SSN) A semismooth Newton algorithm for subproblem (2.3)

Input: $\iota \in (0, 1/2)$, $\bar{\delta} \in (0, 1)$, $\tau \in (0, 1]$ and $\zeta \in (0, 1)$. Choose $x^0 \in \mathbb{R}^n$. Iterate the following steps for j=0,1,...

1: Choose $W_j \in \hat{\partial}^2\Phi(x^j)$. Apply the conjugate gradient (CG) algorithm to find an approximate direction d^j to satisfy the following linear system

$$W_j d = -\nabla\Phi(x^j) \tag{2.7}$$

such that

$$\|W_j d^j + \nabla\Phi(x^j)\| \le \min(\bar{\delta}, \|\nabla\Phi(x^j)\|^{1+\tau}).$$

2: Set $\alpha_j = \zeta^{m_j}$, where m_j is the smallest nonnegative integer m satisfying

$$\Phi(x^j + \zeta^m d^j) \le \Phi(x^j) + \iota\zeta^m\langle\nabla\Phi(x^j), d^j\rangle.$$

3: Set $x^{j+1} = x^j + \alpha_j d^j$.

From [38, Theorem 3.4 and 3.5], we can easily state results on the global and local convergence of the above SSN algorithm without detailed proof here.

Theorem 2.3. *Let the sequence $\{x^j\}$ be generated by the* SSN *algorithm. Then $\{x^j\}$ converges to the unique optimal solution $\hat{x}$ of problem* (2.3) *and the rate of convergence is at least superlinear*

$$\|x^{j+1} - \hat{x}\| = O(\|x^j - \hat{x}\|^{1+\tau}),$$

where τ is the parameter used in the SSN *algorithm.*

2.3 Efficient techniques for implementation of the semismooth Newton algorithm

As we know, it is the most crucial step to find the appropriate search direction d^j in the SSN Algorithm, so we focus on solving the linear system (2.7) efficiently. We intend to further analyze the special structure of the Hessian matrix W to reduce the computational costs of linear system (2.7).

For given $\sigma > 0$, $x \in \mathbb{R}^n$ and $\tilde{\mu} \in \mathbb{R}^{n-k}$, the linear system (2.7) can be written as follows:

$$(I_n + \sigma {D^{(k,n)}}^T V D^{(k,n)})d = -\nabla\Phi(x), \tag{2.8}$$

where $V \in \partial\Pi_{\sigma^{-1}\lambda B_\infty}(D^{(k,n)}x + \sigma^{-1}\tilde{\mu})$. The costs of computing ${D^{(k,n)}}^T V D^{(k,n)}$ and ${D^{(k,n)}}^T V D^{(k,n)}d$ are $O(n(n-k)(2n-k))$ and $O((n-k)(2n-k))$, respectively. The computational costs of the Hessian matrix $I_n+\sigma {D^{(k,n)}}^T V D^{(k,n)}$ and the matrix-vector multiplication $(I_n+\sigma {D^{(k,n)}}^T V D^{(k,n)})d$ are so expensive that the frequently-used methods like the Cholesky factorization and the conjugate gradient method are not efficient for solving some large-scale problems.

Fortunately, the computational costs of solving the linear system (2.7) can be reduced by applying the second order sparsity of V. Next, we tend to show how this can be done by fully utilizing the sparsity of V. By observing the structure of matrix V, we can let $v_i = 0$ when $|D^{(k,n)}x + \sigma^{-1}\tilde{\mu}|_i = \sigma^{-1}\lambda$ in the equation (2.7). Therefore, V is a diagonal matrix with ith diagonal elements being either 0 or 1 and hence the sparsity of V can substantially reduce these unfavorable computational costs to a low level.

The diagonal matrix V can be written as the following form:

$$v_i = \begin{cases} 1, & \text{if } |D^{(k,n)}x + \sigma^{-1}\tilde{\mu}|_i < \sigma^{-1}\lambda, \\ 0, & \text{otherwise,} \end{cases} \quad i = 1, ..., n-k.$$

Let $\mathcal{J} := \{j | v_j = 1, j = 1, ..., n-k\}$ be the index set that corresponds to the indices of nonzero diagonal elements in V and $r = |\mathcal{J}|$ denotes the cardinality of $\mathcal{J}$. Then, we take the special 0-1 structure of V into account, i.e.,

$$D^{(k,n)^T} V D^{(k,n)} = (V D^{(k,n)})^T (V D^{(k,n)}) = D_{\mathcal{J}}^{(k,n)^T} D_{\mathcal{J}}^{(k,n)}, \tag{2.9}$$

where $D_{\mathcal{J}}^{(k,n)} \in \mathbb{R}^{r \times n}$ is the sub-matrix of $D^{(k,n)}$ with those rows not in $\mathcal{J}$ being removed from $D^{(k,n)}$. Therefore, the cost of computing $D^{(k,n)^T} V D^{(k,n)}$ can be reduced from $O(n(n-k)(2n-k))$ to $O(n^2 r)$ by using (2.9). We reduce the computational costs of the linear system (2.7) by applying the Cholesky factorization and the total computational costs are reduced from $O(n(n-k)(2n-k)) + O(n^3)$ to $O(n^2 r) + O(n^3)$.

In addition, we use Sherman-Morrison-Woodbury formula [8] to get the inverse of $I_n + \sigma D^{(k,n)^T} V D^{(k,n)}$:

$$\begin{aligned}(I_n + \sigma D^{(k,n)^T} V D^{(k,n)})^{-1} &= (I_n + \sigma D_{\mathcal{J}}^{(k,n)^T} D_{\mathcal{J}}^{(k,n)})^{-1} \\ &= I_n - D_{\mathcal{J}}^{(k,n)^T} (\sigma^{-1} I_r + D_{\mathcal{J}}^{(k,n)} D_{\mathcal{J}}^{(k,n)^T})^{-1} D_{\mathcal{J}}^{(k,n)}.\end{aligned}$$

In this case, we only need to factorize a $r \times r$ matrix instead of a $n \times n$ matrix. Thus, the total computational costs can be reduced from $O(n^2 r) + O(n^3)$ to $O(n^2 r) + O(r^3)$ when combining the Cholesky factorization with Sherman-Morrison-Woodbury formula.

From the above arguments, we know that the computational costs would be cheap when r is small. When r is large, we only need to use the conjugate gradient method to solve the linear system (2.7), where the pivotal step is to compute the formula $D_{\mathcal{J}}^{(k,n)^T} D_{\mathcal{J}}^{(k,n)} d$ efficiently. We can apply the fast computation $D_{\mathcal{J}}^{(k,n)^T} (D_{\mathcal{J}}^{(k,n)} d)$ for any given d to make $D_{\mathcal{J}}^{(k,n)^T} D_{\mathcal{J}}^{(k,n)} d$ consist only of matrix-vector multiplication rather than matrix-matrix multiplication. Since the matrix $D_{\mathcal{J}}^{(k,n)}$ has a special structure that $D_{\mathcal{J}}^{(k,n)}$ only has k nonzero elements in each row, the effective multiplication numbers of matrix-vector multiplication $D_{\mathcal{J}}^{(k,n)} d$ and $D_{\mathcal{J}}^{(k,n)^T} (D_{\mathcal{J}}^{(k,n)} d)$ are both $(k * r)$. So the total computational cost of $D_{\mathcal{J}}^{(k,n)^T} (D_{\mathcal{J}}^{(k,n)} d)$ is $O(kr)$.

In order to employ the SSN algorithm to solve the subproblem involved in the SSNAL algorithm, we need to compute the matrix-vector multiplications $D^{(k,n)} x$ and $D^{(k,n)^T} x$. When n or k is large, direct matrix-vector multiplications may have expensive computational costs. To reduce computational costs and increase the efficiency of the algorithm, we can compute the matrix-vector multiplications $D^{(k,n)} x$ and $D^{(k,n)^T} x$ recursively. The function of computing the matrix-vector multiplication $D^{(k,n)} x$ in Matlab is given blow:

```
function Dx = ComputDx(x,k)
  if k==1
     Dx = -diff(x);
  else
     y = ComputDx(x,k-1);
     Dx = -diff(y);
  end
```

Similarly, the function of computing the matrix-vector multiplication $D^{(k,n)^T} x$ in Matlab is given by

```
function DTx = ComputDTx(x,k)
  if k==1
     DTx = x;
     DTx(2:end) = diff(x);
     DTx(end+1) = -x(end);
  else
     DTx = ComputDTx(ComputDTx(x,1),k-1);
  end
```

3 Numerical Experiments

In this section, we conduct numerical experiments on synthetic and real data sets to demonstrate excellent performance of the SSNAL algorithm for solving general ℓ_1 trend filtering (1.1). In addition, we tend to compare it with other algorithms including ADMM and PDIP. All our numerical experiments are executed in MATLAB on Dell desktop computer with Inter(R) Core(TM) i5-8500 CPU@3.00GHz and 8GB RAM.

Alternating direction method of multipliers (ADMM) algorithm is an important method for solving convex optimization problems with separable structures, which was first proposed by Gabay and Mercier [7]. The process is stated as follows:

Algorithm 3 (ADMM) Alternating direction method of multipliers for problem (P)

Input: $\omega \in (0, (1+\sqrt{5})/2)$, $\sigma > 0$, $(x^0, z^0, \mu^0) \in \mathbb{R}^n \times \mathbb{R}^{n-k} \times \mathbb{R}^{n-k}$ and $j = 0$.

1: Compute
$$x^{j+1} \in \arg\min_x \mathcal{L}_\sigma(x, z^j; \mu^j). \tag{3.1}$$

2: Compute
$$z^{j+1} \in \arg\min_z \mathcal{L}_\sigma(x^{j+1}, z; \mu^j). \tag{3.2}$$

3: Set $\mu^{j+1} = \mu^j + \omega\sigma(D^{(k,n)}x^{j+1} - z^{j+1})$.

4: Set $j \leftarrow j+1$, and go to step 1.

From equation (3.1), we know that the subproblem can be solved by the following equivalent form

$$(I_n + \sigma {D^{(k,n)}}^T D^{(k,n)})x = y + {D^{(k,n)}}^T(\sigma z^j - \mu^j). \tag{3.3}$$

We can apply the conjugate gradient method or the Cholesky factorization method to obtain the solution of the linear system (3.3). We note that the subproblem (3.2) has a closed-form solution:

$$z^{j+1} = \mathrm{Prox}_{\sigma^{-1}\lambda\|\cdot\|_1}(D^{(k,n)}x^{j+1} + \sigma^{-1}\mu^j). \tag{3.4}$$

A primal-dual interior-point (PDIP) algorithm can solve some quadratic problems in finite number of interations, almost independent of the problem size or data. The details of PDIP algorithm for solving ℓ_1 trend filtering has been discussed in [12]. We summarize the algorithmic framework of PDIP algorithm as follows:

Algorithm 4 (PDIP) Primal-dual interior-point algorithm

Input: $t > 0$, $\eta \in (0, 0.5]$, $\xi \in (0, 1)$, $(\mu^0, \mu_1^0, \mu_2^0) \in \mathbb{R}^{n-k} \times \mathbb{R}^{n-k} \times \mathbb{R}^{n-k}$ and $j = 0$.

1: Compute the Newton steps $(\Delta\mu^j, \Delta\mu_1^j, \Delta\mu_2^j)$ for the system of nonlinear equations

$$r_t(\mu^j, \mu_1^j, \mu_2^j) = \begin{bmatrix} \nabla g(\mu^j) + D^{(k,n)}(\mu^j - \lambda \mathbf{1}_{n-k})^T \mu_1^j - D^{(k,n)}(\mu^j + \lambda \mathbf{1}_{n-k})^T \mu_2^j \\ -\mu_1^j(\mu^j - \lambda \mathbf{1}_{n-k}) + \mu_2^j(\mu^j + \lambda \mathbf{1}_{n-k}) - (1/t)\mathbf{1}_{n-k} \end{bmatrix} = 0,$$

where $r_t(\mu^j, \mu_1^j, \mu_2^j)$ is the residual.

2: Set $\alpha^j = \xi^{m_j}$, where m_j is the smallest nonnegative integer m satisfying

$$\|r(\mu^j + \xi^m \Delta\mu, \mu_1^j + \xi^m \Delta\mu_1, \mu_2^j + \xi^m \Delta\mu_2)\|_2 \le (1 - \eta\xi^m)\|(r_t(\mu^j, \mu_1^j, \mu_2^j)\|_2.$$

3: Update

$$\mu^{j+1} = \mu^j + \alpha^j \Delta\mu^j,\ \mu_1^{j+1} = \mu_1^j + \alpha^j \Delta\mu_1^j,\ \mu_2^{j+1} = \mu_2^j + \alpha^j \Delta\mu_2^j.$$

4: Compute

$$x^{j+1} = y - {D^{(k,n)}}^T \mu^{j+1}.$$

5: Set $j \leftarrow j + 1$, and go to step 1.

3.1 Stopping criteria

We use the following relative residuals to measure the accuracy of approximate optimal solutions obtained by all the tested algorithms:

$$\mathrm{Res}_1 := \frac{\|x - y + {D^{(k,n)}}^T \mu\|}{1 + \|x\| + \|y\| + \|{D^{(k,n)}}^T \mu\|},$$
$$\mathrm{Res}_2 := \frac{\|D^{(k,n)} x - \mathrm{Prox}_{\lambda\|\cdot\|_1}(D^{(k,n)} x + \mu)\|}{1 + \|D^{(k,n)} x\| + \|\mu\|}.$$

The primal infeasibility, dual infeasibility are denoted by R_p, R_d, namely

$$R_p = \frac{\|Dx - z\|}{1 + \|Dx\| + \|z\|},\ R_d = \frac{\|x - y + {D^{(k,n)}}^T \mu\|}{1 + \|x\| + \|y\| + \|{D^{(k,n)}}^T \mu\|}.$$

In our experiments, for the SSNAL algorithm, we initialize the SSNAL algorithm with $(x^0, z^0, \mu^0) = (0, 0, 0)$ and terminate the algorithm when $R_{kkt} := \max\{\mathrm{Res}_1, \mathrm{Res}_2\} \le \mathrm{tol}$ or the number of outer iterations exceeds 50. The penalty parameter σ_{i+1} in the SSNAL algorithm is adjusted dynamically: starting from the initial value 0.1, we adjust σ_{i+1} as follows such that σ_{i+1} cannot go to extremely large or small, and the primal and dual infeasibilities are well balanced [2]:

$$\sigma_{i+1} = \begin{cases} \min\{10^{-4}, 0.1\sigma_i\}, & \text{if } \mathrm{Res}_1^i > 10\mathrm{Res}_2^i, \\ \max\{5 \times 10^3, 1.3\sigma_i\}, & \text{if } \mathrm{Res}_1^i < 0.1\mathrm{Res}_2^i, \\ \sigma_i, & \text{otherwise.} \end{cases}$$

We initialize the ADMM algorithm with $(x^0, z^0, \mu^0) = (0, 0, 0)$ and terminate the algorithm when $R_{kkt} := \max\{\mathrm{Res}_1, \mathrm{Res}_2\} \le \mathrm{tol}$ or the maximum number of iterations 20000 is reached. We start the PDIP algorithm from the initial point $(\mu^0, \mu_1^0, \mu_2^0) = (0, 0, 0)$ and terminate the algorithm when $R_{kkt} := \max\{\mathrm{Res}_1, \mathrm{Res}_2\} \le \mathrm{tol}$ or the number of outer iterations exceeds 20000.

For the parameters of the SSN algorithm, we set $\iota = 10^{-12}$ and $\zeta = 0.5$. We terminate the CG algorithm at j-th SSN iteration when $\|W_j d^j + \nabla\Phi(x^j)\| \le \mathrm{tol}_{\mathrm{cg}}^{\mathrm{j}}$ with

$$\mathrm{tol}_{\mathrm{cg}}^{\mathrm{j}} = \begin{cases} \min\{0.1, 0.1\nabla\Phi(x^j)\}, & i < 2 \text{ and } j < 1, \\ \min\{0.5, 0.5\nabla\Phi(x^j)\}, & \text{otherwise.} \end{cases}$$

3.2 Numerical results for synthetic data

In this subsection, we compare the SSNAL algorithm against other algorithms (PDIP, ADMM) for the general ℓ_1 trend filtering problem (1.1) on synthetic data sets, which are generated as

$$y_t = x_t + z_t,\ t = 1, ..., n,\ x_{t+1} = x_t + v_t,\ t = 1, ..., n-1, \tag{3.5}$$

where v_t is the trend slope, x_t is the "true" underlying trend, and z_t is the irregular component or noise. The initial condition is $x_1 = 0$ and the noises z_t are independent and identically distributed $\mathcal{N}(0, \beta^2)$. The trend slopes v_t are selected from a simple Markov process (independent of z). With probability p, we have $v_{t+1} = v_t$, i.e., no slope change in the underlying trend. With probability $1 - p$, we choose v_{t+1} from a uniform distribution on $[-b, b]$. We choose the initial slope v_1 from a uniform distribution on $[-b, b]$. In our experiments, we set $p = 0.01, \beta = 1, b = 0.5$. We test three cases of the tuning parameter with $\lambda = 0.001, 0.005, 0.01$. We let $n = 200000i, i = 1, ..., 5$ and the accuracy tol $= 10^{-6}$. We selected the average of the data from five experiments as our experimental results and time is measured in seconds.

Table 1: The numerical results of SSNAL, PDIP and ADMM on synthetic data sets when $k = 2$, tol $= 10^{-6}$.

n	solver	λ=0.001		λ=0.005		λ=0.01	
		Time	R_{kkt}	Time	R_{kkt}	Time	R_{kkt}
2e+05	SSNAL	0.206	3.003e-07	0.488	3.025e-07	0.543	2.908e-07
	PDIP	1.147	3.561e-07	1.427	2.523e-07	1.472	1.904e-07
	ADMM	0.575	3.614e-07	2.536	3.940e-07	3.293	3.702e-07
4e+05	SSNAL	0.281	2.963e-07	0.956	3.035e-07	1.197	2.863e-07
	PDIP	2.269	3.630e-07	2.812	2.458e-07	2.955	1.955e-07
	ADMM	1.239	3.580e-07	5.092	3.766e-07	7.077	3.796e-07
6e+05	SSNAL	0.423	2.962e-07	1.501	3.056e-07	1.906	2.893e-07
	PDIP	3.406	3.659e-07	4.277	2.530e-07	4.424	2.012e-07
	ADMM	1.873	3.579e-07	8.127	3.776e-07	11.093	3.785e-07
8e+05	SSNAL	0.585	2.902e-07	2.046	3.036e-07	2.602	2.925e-07
	PDIP	4.547	3.654e-07	5.697	2.535e-07	5.962	2.004e-07
	ADMM	2.568	3.574e-07	11.019	3.757e-07	15.231	3.778e-07
1e+06	SSNAL	0.735	2.910e-07	2.567	3.027e-07	3.334	2.976e-07
	PDIP	5.719	3.666e-07	7.011	2.582e-07	7.501	2.018e-07
	ADMM	3.275	3.569e-07	13.926	3.734e-07	19.406	3.788e-07

From Table 1, we can see the comparison results of SSNAL, PDIP and ADMM on the computational time (Time) and the relative KKT residual (R_{kkt}) under $k = 2$, tol $= 10^{-6}$. Although all the algorithms can successfully solve the problem within the required accuracy, it is obvious that SSNAL algorithm has better performance than other algorithms. For the computational time, the SSNAL algorithm is 3 to 10 times faster than PDIP algorithm, 5 to 10 times faster than ADMM algorithm.

Table 2: The numerical results of SSNAL, PDIP and ADMM on synthetic data sets when $k = 3$, tol $= 10^{-6}$.

n	solver	λ=0.001		λ=0.005		λ=0.01	
		Time	R_{kkt}	Time	R_{kkt}	Time	R_{kkt}
2e+05	SSNAL	0.107	1.977e-07	0.471	3.081e-07	0.591	2.399e-07
	PDIP	1.377	2.963e-07	1.854	2.810e-07	1.850	3.241e-07
	ADMM	1.359	1.472e-07	2.566	3.469e-07	3.001	2.952e-07
4e+05	SSNAL	0.238	1.982e-07	1.021	2.963e-07	1.382	2.628e-07
	PDIP	2.990	2.589e-07	3.454	3.643e-07	3.971	1.699e-07
	ADMM	2.797	1.907e-07	5.455	3.348e-07	6.394	3.199e-07
6e+05	SSNAL	0.340	1.943e-07	1.660	2.973e-07	2.191	2.796e-07
	PDIP	4.169	2.704e-07	5.275	3.365e-07	6.025	2.403e-07
	ADMM	4.012	2.831e-07	8.459	3.433e-07	9.953	3.301e-07
8e+05	SSNAL	0.466	1.970e-07	2.326	2.991e-07	2.990	2.740e-07
	PDIP	5.691	2.720e-07	7.703	3.122e-07	7.621	3.735e-07
	ADMM	5.475	2.015e-07	11.631	3.469e-07	13.706	3.262e-07
1e+06	SSNAL	0.594	1.944e-07	2.902	3.036e-07	3.994	2.704e-07
	PDIP	11.413	2.326e-07	14.905	2.729e-07	14.870	2.571e-07
	ADMM	6.134	3.007e-07	14.901	3.533e-07	17.488	3.262e-07

Table 2 shows the numerical results of the tested algorithms under $k = 3$, tol $= 10^{-6}$. Compared with the results of $k = 2$, it can be seen that SSNAL algorithm still maintains the advantages of shorter time. The SSNAL algorithm is 10 to 20 times faster than PDIP algorithm and ADMM algorithm with $\lambda = 0.001$. For the instance $n =$ 1e+06 and $\lambda = 0.001$, it takes 0.594 seconds for SSNAL algorithm to get the high-precision solution, while ADMM algorithm needs 6.134 seconds to achieve the required accuracy, and PDIP algorithm even costs 11.413 seconds to solve the problem.

Table 3: The numerical results of SSNAL, PDIP and ADMM on synthetic data sets when $k = 4$, tol $= 10^{-6}$.

n	solver	λ=0.001		λ=0.005		λ=0.01	
		Time	R_{kkt}	Time	R_{kkt}	Time	R_{kkt}
2e+05	SSNAL	0.125	9.181e-08	0.931	3.408e-07	1.335	3.642e-07
	PDIP	1.709	2.511e-07	2.234	2.679e-07	2.569	2.144e-07
	ADMM	7.435	3.243e-07	7.419	3.169e-07	7.425	3.854e-07
4e+05	SSNAL	0.257	9.379e-08	1.957	3.530e-07	2.716	3.651e-07
	PDIP	3.384	2.554e-07	4.424	2.901e-07	4.842	3.690e-07
	ADMM	13.452	3.708e-07	14.800	3.903e-07	15.447	3.874e-07
6e+05	SSNAL	0.406	9.311e-08	3.078	3.451e-07	4.525	3.570e-07
	PDIP	4.819	3.096e-07	6.654	2.995e-07	7.297	3.960e-07
	ADMM	22.062	3.934e-07	23.106	3.950e-07	25.201	2.891e-07
8e+05	SSNAL	0.550	9.432e-08	4.185	3.472e-07	7.449	3.687e-07
	PDIP	7.156	2.904e-07	9.183	2.984e-07	9.738	2.848e-07
	ADMM	28.981	3.942e-07	31.848	3.975e-07	34.749	2.887e-07
1e+06	SSNAL	0.706	9.322e-08	5.615	3.497e-07	10.385	3.740e-07
	PDIP	9.212	2.828e-07	11.832	2.991e-07	12.737	2.995e-07
	ADMM	35.269	3.894e-07	42.733	3.023e-07	44.589	2.968e-07

Table 3 demonstrates the performances of all algorithms on Time and R_{kkt} under $k = 4$, tol $= 10^{-6}$. The SSNAL algorithm still performs very well, while the performances of PDIP algorithm and ADMM algorithm are quite poor. For the instance $\lambda = 0.001$ and $n = 1\text{e}+6$, SSNAL algorithm takes 0.706 seconds to reach the high accuracy $R_{kkt} = 9.322\text{e}-08$, while PDIP algorithm needs 9.212 seconds to achieve the lower accuracy $R_{kkt} = 2.828\text{e}-07$, and ADMM algorithm even costs 35.269 seconds to get the lowest accuracy $R_{kkt} = 3.894\text{e}-07$. Therefore, SSNAL algorithm is superior to other two algorithms on synthetic data when $R_{kkt} \leq 10^{-6}$.

3.3 Numerical results for real data

In this subsection, we compare the SSNAL algorithm with PDIP algorithm and ADMM algorithm for solving general ℓ_1 trend filtering (1.1) on real data sets collected from PJM dataset. PJM Interconnection LLC (PJM) is a regional transmission organization (RTO) in the United States. It is part of the Eastern Interconnection grid operating an electric transmission system serving all or parts of Delaware, Illinois, Indiana, Kentucky and so on. In this database, we pick out four datasets to test all the algorithms. In our experiments, we still choose three tuning parameters with $\lambda = 0.001, 0.005, 0.01$. We set $k = 2, 3, 4$ and different n depending on the size of the data sets. In addition, the statistics of all tested instances are shown in Table 4.

https://www.kaggle.com/robikscube/hourly-energy-consumption

Table 4: Summary of tested data sets.

abbreviation	proname	source	n
data1	PJM-Load-hourly	PJM	32896
data2	NI-hourly	PJM	58450
data3	PJMW-hourly	PJM	143206
data4	PJM-hourly-est	PJM	803000

Table 5: The numerical results of SSNAL, PDIP and ADMM on real data sets when $k = 2$, tol $= 10^{-6}$.

n	solver	λ=0.001		λ=0.005		λ=0.01	
		Time	R_{kkt}	Time	R_{kkt}	Time	R_{kkt}
32896	SSNAL	0.006	7.617e-08	0.003	3.809e-07	0.006	5.777e-08
	PDIP	0.023	1.641e-07	0.110	3.513e-07	0.188	2.626e-07
	ADMM	0.021	2.950e-07	0.018	3.086e-07	0.019	3.302e-07
58450	SSNAL	0.006	3.283e-07	0.010	1.343e-07	0.011	2.842e-07
	PDIP	0.104	3.882e-07	0.307	3.423e-07	0.381	2.670e-07
	ADMM	0.034	3.296e-07	0.029	3.730e-07	0.041	2.900e-07
143206	SSNAL	0.076	8.355e-08	0.093	3.814e-07	0.249	3.722e-07
	PDIP	1.088	3.099e-07	1.247	2.660e-07	1.385	3.104e-07
	ADMM	0.257	3.370e-07	0.421	3.291e-07	0.867	3.901e-07
803000	SSNAL	0.430	2.192e-07	2.027	3.112e-07	2.856	1.564e-07
	PDIP	6.146	2.936e-07	7.426	2.540e-07	7.848	2.282e-07
	ADMM	4.242	3.396e-07	7.924	3.814e-07	13.781	3.979e-07

From Table 5, we obtain comparison results of several algorithms on Time and R_{kkt} under $k = 2$, tol $= 10^{-6}$. The SSNAL algorithm can solve the problem in a shorter time, while the other two algorithms perform worse. For example, when $n = 32896$ and $\lambda = 0.01$, the SSNAL algorithm only takes 0.006 seconds to solve the problem with high accuracy $R_{kkt} = 5.777\text{e}-08$, but the ADMM algorithm costs 0.019 seconds to get the solution with low accuracy $R_{kkt} = 3.302\text{e}-07$, and the PDIP algorithm even needs 0.188 seconds to obtain the solution with low accuracy $R_{kkt} = 2.626\text{e}-07$.

Table 6: The numerical results of SSNAL, PDIP and ADMM on real data sets when $k = 3$, tol $= 10^{-6}$.

n	solver	λ=0.001		λ=0.005		λ=0.01	
		Time	R_{kkt}	Time	R_{kkt}	Time	R_{kkt}
32896	SSNAL	0.003	5.355e-08	0.003	2.677e-07	0.007	5.220e-08
	PDIP	0.030	1.135e-07	0.131	3.339e-07	0.241	2.248e-07
	ADMM	0.063	3.551e-07	0.062	3.575e-07	0.064	3.610e-07
58450	SSNAL	0.006	2.258e-07	0.012	8.894e-08	0.015	1.996e-07
	PDIP	0.049	2.753e-07	0.406	3.896e-07	0.460	3.061e-07
	ADMM	0.101	3.909e-07	0.100	3.975e-07	0.114	1.575e-07
143206	SSNAL	0.084	4.869e-08	0.081	2.310e-07	0.168	3.985e-07
	PDIP	0.982	3.645e-07	1.748	2.583e-07	1.732	2.516e-07
	ADMM	1.020	3.906e-07	1.182	1.992e-07	1.144	3.921e-07
803000	SSNAL	0.538	9.862e-08	2.061	3.588e-07	3.223	2.567e-07
	PDIP	7.827	2.306e-07	8.574	3.492e-07	9.734	2.646e-07
	ADMM	9.989	3.458e-07	11.637	3.936e-07	14.954	3.257e-07

As revealed in Table 6, the SSNAL algorithm has more obvious advantages over PDIP algorithm and ADMM algorithm with the increase of k. For the computational time, the SSNAL algorithm is 3 to 20 times faster than PDIP algorithm, 5 to 20 times faster than ADMM algorithm. For accuracy, the SSNAL algorithm is higher than other two algorithms in most cases.

Table 7: The numerical results of SSNAL, PDIP and ADMM on real data sets when $k = 4$, tol $= 10^{-6}$.

n	solver	λ=0.001		λ=0.005		λ=0.01	
		Time	R_{kkt}	Time	R_{kkt}	Time	R_{kkt}
32896	SSNAL	0.004	6.628e-08	0.004	3.314e-07	0.007	2.402e-07
	PDIP	0.039	6.561e-08	0.038	3.279e-07	0.174	3.756e-07
	ADMM	0.373	2.896e-07	0.372	2.892e-07	0.380	2.887e-07
58450	SSNAL	0.007	1.664e-07	0.019	3.584e-08	0.018	9.836e-08
	PDIP	0.059	1.563e-07	0.372	3.908e-07	0.579	3.154e-07
	ADMM	0.585	3.160e-07	0.590	3.144e-07	0.597	3.125e-07
143206	SSNAL	0.052	3.578e-07	0.142	9.146e-08	0.150	2.526e-07
	PDIP	0.176	3.376e-07	2.015	2.921e-07	2.224	3.616e-07
	ADMM	6.145	3.161e-07	6.232	3.137e-07	6.179	3.111e-07
803000	SSNAL	1.502	6.818e-08	4.472	1.958e-07	6.115	2.541e-07
	PDIP	8.125	3.304e-07	11.120	3.652e-07	13.128	3.867e-07
	ADMM	55.464	3.593e-07	55.791	3.539e-07	55.514	3.481e-07

From Table 7, we can intuitively observe that performance of the SSNAL algorithm is much faster than that of other two algorithms under $k = 4$, tol $= 10^{-6}$. For example, when $n = 803000$ and $\lambda = 0.001$, it only takes 1.502 seconds for SSNAL algorithm to achieve the high accuracy $R_{kkt} = 6.818\text{e} - 08$, while the PDIP algorithm costs 8.125 seconds with a lower accuracy $R_{kkt} = 3.304\text{e} - 07$, and the ADMM algorithm even needs 55.464 seconds to get the lowest accuracy $R_{kkt} = 3.593\text{e} - 07$. To sum up, the SSNAL algorithm performs better than other two algorithms on real datasets when $R_{kkt} \leq 10^{-6}$.

4 Conclusion

In this paper, we have presented an efficient and robust semismooth Newton based augmented Lagrangian (SSNAL) algorithm for solving general ℓ_1 trend filtering. The theoretical results on the global and local convergence property of the SSNAL algorithm has been illustrated. We utilized the second order sparsity and some efficient techniques to reduce the computational costs of the SSN algorithm for solving the subproblem of the SSNAL algorithm. In numerical experiments, the remarkable performance of the SSNAL algorithm has been demonstrated by comparing the SSNAL algorithm with other state-of-the-art algorithms.

References

[1] R.T. Baillie and S.K. Chung, Modeling and forecasting from trend stationary long memory models with applications to climatology, *Int. J. Forecast.* 18 (2002) 215–226.

[2] C.H. Chen, Y.-J. Liu, D.F. Sun and K.-C. Toh, A semismooth Newton-CG based dual PPA for matrix spectral norm approximation problems, *Math. Program.* 155 (2016) 435–470.

[3] F.H. Clarke, *Optimization and Nonsmooth Analysis*, John Wiley and Sons, New York, 1983.

[4] L. Condat, A direct algorithm for 1D total variation denoising, *IEEE Signal Proc. Let.* 20 (2013) 1054–1057.

[5] P.L. Davies and A. Kovac, Local extremes, runs, strings and multiresolution, *Ann. Stat.* 29 (2001) 61–65.

[6] S. Fang, Y.-J. Liu and X.Z. Xiong, Efficient sparse Heissian based semismooth Newton algorithms for Dantzig selector, *SIAM J. Sci. Comput.* in press.

[7] D. Gabay and B. Mercier, A dual algorithm for the solution of nonlinear variational problems via finite element approximation, *Comput. Math. Appl.* 2 (1976) 17–40.

[8] G.H. Golub and C.F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, 1996.

[9] S. Greenland and M.P. Longnecker, Methods for trend estimation from summarized dose-response data, with applications to meta-analysis, *Amer. J. Epidemiol.* 135 (1992) 1301–1309.

[10] R.J. Hodrick and E.C. Prescott, Postwar U.S. business cycles: an empirical investigation, *J. Money Credit Bank.* 29 (1997) 1–16.

[11] N.A. Johnson, A dynamic programming algorithm for the fused lasso and L_0-segmentation, *J. Comput. Graph. Stat.* 22 (2013) 246–260.

[12] S.J. Kim, K. Koh, S. Boyd and D. Gorinevsky, ℓ_1 Trend Filtering, *SIAM Rev.* 51 (2009) 339–360.

[13] C. Lemaréchal and C. Sagastizábal, Practical aspects of the Moreau-Yosida regularization: theoretical preliminaries, *SIAM J. Optim.* 7 (1997) 367–385.

[14] S.D. Levitt, Understanding why crime fell in the 1990s: four factors that explain the decline and six that do not, *J. Econ. Perspect.* 18 (2004) 163–190.

[15] X.D. Li, D.F. Sun and K.-C. Toh, A highly efficient semismooth Newton augmented Lagrangian method for solving Lasso problems, *SIAM J. Optim.* 28 (2016) 433–458.

[16] X.D. Li, D.F. Sun and K.-C. Toh, On efficiently solving the subproblems of a level-set method for fused lasso problems, *SIAM J. Optim.* 28 (2018) 1842–1866.

[17] M.X. Lin, Y.-J. Liu, D.F. Sun and K.-C. Toh, Efficient sparse semismooth Newton methods for the clustered Lasso problem, *SIAM J. Optim.* 29 (2019) 2026–2052.

[18] L.Y. Lin and Y.-J. Liu, An efficient Hessian based algorithm for singly linearly and box constrained least squares regression, *J. Sci. Comput.* 88 (2021) 1–21.

[19] J. Liu, L. Yuan and J.P. Ye, An efficient algorithm for a class of fused Lasso problems, in: *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2010, pp. 323–332.

[20] R.E. Lucas, Two illustrations of the quantity theory of money, *Amer. Econom. Rev.* 70 (1980) 1005–1014.

[21] Z.Y. Luo, D.F. Sun, K.-C. Toh and N.H. Xiu, Solving the OSCAR and SLOPE models using a semismooth Newton-based augmented Lagrangian method, *J. Mach. Learn. Res.* 20 (2019) 1–5.

[22] F.J. Luque, Asymptotic convergence analysis of the proximal point algorithm, *SIAM J. Control Optim.* 22 (1984) 277–293.

[23] J.J. Moreau, Proximité et dualité dans un espace hilbertien, *Bull. Soc. Math. France* 93 (1965) 273–299.

[24] D.B. Niu, C.J. Wang, P.P. Tang, Q.S. Wang and E.B. Song, A sparse semismooth Newton based augmented Lagrangian method for large-scale support vector machines, *arXiv preprint arXiv:1910.01312*, (2019).

[25] D.R. Osborne, Moving average detrending and the analysis of business cycles, *Oxford B. Econom. Statist.* 57 (1995) 547–558.

[26] A. Ramdas and R.J. Tibshirani, Fast and flexible ADMM algorithms for trend filtering, *J. Comput. Graph. Statist.* 25 (2016) 839–858.

[27] S.M. Robinson, Some continuity properties of polyhedral multifunctions, *Math. Program. Stud.* 14 (1981) 206–214.

[28] R.T. Rockafellar, *Convex Analysis*, Princeton University Press, Princeton, 1970.

[29] R.T. Rockafellar, Augmented Lagrangians and applications of the proximal point algorithm in convex programming, *Math. Oper. Res.* 1 (1976), 97–116.

[30] R.T. Rockafellar, Monotone operators and the proximal point algorithm, *SIAM J. Control Optim.* 14 (1976) 877–898.

[31] R.T. Rockafellar and J.B. Wets, *Variational Analysis*, Springer, Berlin, 1998.

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

[33] K.J. Singleton, Econometric issues in the analysis of equilibrium business cycle models, *J. Monetary Econ.* 21 (1988) 361–386.

[34] K.T. Talluri and G.J. Ryzin, *The Theory and Practice of Revenue Management*, Kluwer Academic, Boston, 2004.

[35] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu and K. Knight, Sparsity and smoothness via the fused lasso, *J. R. Stat. Soc. Ser. B. Stat. Methodol.* 67 (2005) 91–108.

[36] R.S. Tsay, *Analysis of Financial Time Series*, 2nd Edition, Wiley-Interscience, Hoboken, 2005.

[37] Y.J. Zhang, N. Zhang and D.F. Sun, An efficient Hessian based algorithm for solving large-scale sparse group Lasso problems, *Math. Program.* 179 (2020) 223–263.

[38] X.Y. Zhao, D.F. Sun and K.-C. Toh, A Newton-CG augmented Lagrangian method for semidefinite programming, *SIAM J. Optim.* 20 (2010) 1737–1765.

Manuscript received 5 August 2021
revised 25 October 2021
accepted for publication 6 December 2021

Yong-Jin Liu
School of Mathematics and Statistics, Fuzhou University
Fuzhou 350108, People's Republic of China
E-mail address: yjliu@fzu.edu.cn

Tiqi Zhang
School of Mathematics and Statistics, Fuzhou University
Fuzhou 350108, People's Republic of China
E-mail address: tiqizhang1018@163.com