# Implementation of SuSiE trend filtering

Kaiqian Zhang

June 26, 2022

## 1  Notation

We first describe the notation used in this text. We denote matrices by boldface uppercase letters ($\mathbf{A}$), vectors are denoted by boldface lowercase letters ($\mathbf{a}$), and scalars are denoted by non-boldface letters ($a$ or $A$). All vectors are column-vectors. Lowercase letters may represent elements of a vector or matrix if they have subscripts. For example, $a_{ij}$ is the $(i, j)$th element of $\mathbf{A}$, $a_i$ is the $i$th element of $\mathbf{a}$, and $\mathbf{a}_i$ is either the $i$th row or $i$th column of $\mathbf{A}$. For indexing, we will generally use capital non-boldface letters to denote the total number of elements and their lowercase non-boldface versions to denote the index. For example, $i = 1, \ldots, I$. We let $\mathbf{A}_{n \times p}$ denote that $\mathbf{A} \in \mathbb{R}^{n \times p}$. We denote the matrix transpose by $\mathbf{A}^T$, the matrix inverse by $\mathbf{A}^{-1}$, and the matrix determinant by $\det(\mathbf{A})$. Finally, sets will be denoted by calligraphic letters ($\mathcal{A}$).

## 2  Overview

Trend filtering is a useful statistical tool for nonparametric regression. Kim et al. [2009] first proposed $\ell_1$ trend filtering for estimating underlying piecewise linear trends in time series data. This idea can be further extended to fit piecewise polynomial of degree $k$ to the data. In their paper, Kim et al. showed the equivalence between the $\ell_1$ trend filtering and the $\ell_1$-regularized least squares problem. This motivates us to think about the connection between trend filtering and sparse approximation in general.

## 3  Trend filtering and sparse regression

Trend filtering problem is defined mathematically as follows. For a given integer $k \geqslant 0$, the kth order trend filtering is defined by a penalized least squares optimization problem,

$$\hat{\boldsymbol{b}} = \operatorname*{argmin}_{\boldsymbol{b}} \frac{1}{2} ||\boldsymbol{y} - \boldsymbol{b}||_2^2 + \frac{n^k}{k!} \lambda ||D_{k+1} \boldsymbol{b}||_1, \tag{1}$$

where $\boldsymbol{y} = [y_1 \ldots y_n]^T$ is an n vector of observations, $\lambda$ is a tuning parameter, and $D_{k+1}$ is the discrete difference operator of order $k$. When order $k = 0$, $D$ is defined

$$D_1 = \begin{bmatrix} -1 & 1 & 0 & \dots & 0 & 0 \\ 0 & -1 & 1 & \dots & 0 & 0 \\ \vdots & & \ddots & & & \\ 0 & 0 & 0 & \dots & -1 & 1 \end{bmatrix} \in \mathbb{R}^{(n-1)\times n}. \tag{2}$$

In this case, the components of the trend filtering estimate form a piecewise constant structure, with break points corresponding to the nonzero entries of $D_1\hat{\boldsymbol{b}} = (\hat{b}_2 - \hat{b}_1, \dots, \hat{b}_n - \hat{b}_{n-1})$ [Tibshirani, 2014]. And when $k \geqslant 1$, the operator $D_{k+1}$ is defined recursively,

$$D_{k+1} = D_1 \cdot D_k \in \mathbb{R}^{(n-k-1)\times n}, \tag{3}$$

where the dot product is matrix multiplication. Notice that $D_1$ here in 3 is the $(n-k-1) \times (n-k)$ version of $D_1$ described in 2.

Now we want to transform the trend filtering problem into a sparse regression problem. Let $\boldsymbol{\beta} = D_{k+1}\boldsymbol{b}$. Then if $D_{k+1}$ were invertibe, we could write $\boldsymbol{b} = (D_{k+1})^{-1}\boldsymbol{\beta}$ and the above problem would become

$$\hat{\boldsymbol{\beta}} = \operatorname*{argmin}_{\boldsymbol{\beta}} \frac{1}{2}||\boldsymbol{y} - (D_{k+1})^{-1}\boldsymbol{\beta}||_2^2 + \frac{n^k}{k!}\lambda||\boldsymbol{\beta}||_1. \tag{4}$$

We can consider this a sparse regression with $\ell_1$ regularization problem, where the design matrix is $X_{k+1} = (D_{k+1})^{-1}$.

## 4  Modification of $D$

As we have seen, the trend filtering problem becomes a sparse regression with $\ell_1$ regularization if we consider the design matrix $X_{k+1} = (D_{k+1})^{-1}$. However, $D_1 \in \mathbb{R}^{(n-1)\times n}$ is not invertible, so is $D_{k+1}$ for $k = 1, 2, \dots$ By observation, we complete $D_1$ as a square and symmetric matrix

$$\hat{D}_1 = \begin{bmatrix} -1 & 1 & 0 & \dots & 0 & 0 \\ 0 & -1 & 1 & \dots & 0 & 0 \\ \vdots & & \ddots & & & \\ 0 & 0 & 0 & \dots & -1 & 1 \\ 0 & 0 & 0 & \dots & 0 & -1 \end{bmatrix} \in \mathbb{R}^{n\times n}. \tag{5}$$

And for $k \geqslant 1$, we obtain

$$\hat{D}_{k+1} = \hat{D}_1 \cdot \hat{D}_k \in \mathbb{R}^{n\times n}. \tag{6}$$

We notice that, by this modification, $\hat{D}_{k+1}$ has $k$ more rows added at the bottom without changing any previous entry. With this modification, we are able to invert $\hat{D}_{k+1}$ and consider the inverse matrix as a design matrix in the sparse regression.

# 5 Special structure of $\hat{D}^{-1}$

After determining the design matrix $X$ in the sparse regression problem, we could apply *SuSiE* algorithm to help us find a possible fit. Here, we denote $X_{k+1} = (\hat{D}_{k+1})^{-1}$, where $k$ is the order of trend filtering. Rather than generating $\hat{D}^{-1}$ and set this as an $X$ input, we exploit the special structure of $\hat{D}^{-1}$ and perform *SuSiE* on this specific trend filtering problem with $O(n)$ complexity. We will talk about how to make different computations linear in complexity by utilizing the special structure respectively.

# 6 Computation of $Xb$

In the trend filtering application, since $X_{k+1} = (\hat{D}_{k+1})^{-1}$, we obtain

$$X_{k+1}\boldsymbol{b} = (\hat{D}_{k+1})^{-1}\boldsymbol{b} = (\underbrace{\hat{D}_1 \dots \hat{D}_1}_{k+1})^{-1}\boldsymbol{b} \tag{7}$$

$$= \underbrace{(\hat{D}_1)^{-1} \dots (\hat{D}_1)^{-1}}_{k+1}\boldsymbol{b} \tag{8}$$

$$= \underbrace{X_1 \dots X_1}_{k+1}\boldsymbol{b}. \tag{9}$$

We notice that since

$$X_1 = \begin{bmatrix} -1 & -1 & -1 & \dots & -1 & -1 \\ 0 & -1 & -1 & \dots & -1 & -1 \\ \vdots & & \ddots & & & \\ 0 & 0 & 0 & \dots & -1 & -1 \\ 0 & 0 & 0 & \dots & 0 & -1 \end{bmatrix} \in \mathbb{R}^{n \times n}, \tag{10}$$

$$X_1\boldsymbol{b} = -1 \cdot [b_1 + b_2 + \dots + b_n, b_2 + \dots + b_n, \dots, b_{n-1} + b_n, b_n]^T \tag{11}$$

$$= -1 \cdot \text{cumsum}(\text{reverse}(\boldsymbol{b})). \tag{12}$$

Let $f : \mathbb{R}^n \to \mathbb{R}^n$ such that $f(\boldsymbol{x}) = -\text{cumsum}(\text{reverse}(\boldsymbol{x}))$ for any $\boldsymbol{x} \in \mathbb{R}^n$. Then

$$X_{k+1}\boldsymbol{b} = \underbrace{X_1 \dots X_1}_{k+1}\boldsymbol{b} = f^{(k+1)}(\boldsymbol{b}), \tag{13}$$

where $k$ is the order of trend filtering.

# 7 Computation of $X^T y$

We consider $X_{k+1}^T \boldsymbol{y}$. Here $X_{k+1} = (\hat{D}_{k+1})^{-1}$ in the trend filtering problem, and $\boldsymbol{y}$ is an n vector. We have

$$X_{k+1}^T \boldsymbol{y} = ((\hat{D}_{k+1})^{-1})^T \boldsymbol{y} = ((\underbrace{\hat{D}_1 \dots \hat{D}_1}_{k+1})^{-1})^T \boldsymbol{y} \tag{14}$$

$$= \underbrace{((\hat{D}_1)^{-1})^T \dots ((\hat{D}_1)^{-1})^T}_{k+1} \boldsymbol{y} \tag{15}$$

$$= \underbrace{X_1^T \dots X_1^T}_{k+1} \boldsymbol{y}. \tag{16}$$

Similarly, we observe that since

$$X_1^T = \begin{bmatrix} -1 & 0 & 0 & \dots & 0 & 0 \\ -1 & -1 & 0 & \dots & 0 & 0 \\ \vdots & & \ddots & & & \\ -1 & -1 & -1 & \dots & -1 & 0 \\ -1 & -1 & -1 & \dots & -1 & -1 \end{bmatrix} \in \mathbb{R}^{n \times n}, \tag{17}$$

$$X_1^T \boldsymbol{y} = -1 \cdot [y_1, y_1 + y_2, \dots, y_1 + y_2 + \dots + y_n]^T \tag{18}$$

$$= -1 \cdot \text{cumsum}(\boldsymbol{y}). \tag{19}$$

Let $g : \mathbb{R}^n \to \mathbb{R}^n$ such that $g(\boldsymbol{x}) = -\text{cumsum}(\boldsymbol{x})$ for any $\boldsymbol{x} \in \mathbb{R}^n$. Then

$$X_{k+1}^T \boldsymbol{y} = \underbrace{X_1^T \dots X_1^T}_{k+1} \boldsymbol{y} = g^{(k+1)}(\boldsymbol{y}), \tag{20}$$

where $k$ is the order of trend filtering.

# 8 Computation on $(X^2)^T \mathbf{1}$; i.e., colSums$(X^2)$

To compute $(X_{k+1}^2)^T \mathbf{1}$, let's first explore the special structure of $X_{k+1} = (\hat{D}^{(k+1)})^{-1}$ for $k = 0, 1, 2$.

$$X_1 = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 & \dots \\ 0 & -1 & -1 & -1 & -1 & \dots \\ 0 & 0 & -1 & -1 & -1 & \dots \\ 0 & 0 & 0 & -1 & -1 & \dots \\ \vdots & & \ddots & & & \end{bmatrix} \in \mathbb{R}^{n \times n}, \tag{21}$$

$$X_2 = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & \dots \\ 0 & 1 & 2 & 3 & 4 & 5 & \dots \\ 0 & 0 & 1 & 2 & 3 & 4 & \dots \\ 0 & 0 & 0 & 1 & 2 & 3 & \dots \\ \vdots & & \ddots & & & \end{bmatrix} \in \mathbb{R}^{n \times n}, \tag{22}$$

4

$$X_3 = \begin{bmatrix} -1 & -3 & -6 & -10 & -15 & \cdots \\ 0 & -1 & -3 & -6 & -10 & \cdots \\ 0 & 0 & -1 & -3 & -6 & \cdots \\ 0 & 0 & 0 & -1 & -3 & \cdots \\ \vdots & & \ddots & & & \end{bmatrix} \in \mathbb{R}^{n \times n}, \tag{23}$$

Define a triangular rotate matrix $Q \in \mathbb{R}^{n \times n}$ such that
(i) For any $i, j \leqslant n$, $Q_{ij} = 0$ if $i > j$.
(ii) For any $k < n$, $Q_{ab} = Q_{cd}$ if $b - a = d - c = k$.
We observe that if $X$ is a triangular rotate matrix, then

$$X^T \mathbf{1} = \text{cumsum}(X_{1.}). \tag{24}$$

Since $X^2$ is still a triangular rotate matrix, we obtain

$$(X^2)^T \mathbf{1} = \text{cumsum}(X_{1.}^2). \tag{25}$$

Since $X_{k+1} = (\hat{D}_{k+1})^{-1}$ is a triangular rotate matrix,

$$(X_{k+1}^2)^T \mathbf{1} = \text{cumsum}((X_{k+1})_{1.}^2). \tag{26}$$

And, obviously, the first row of $X_{k+1}$ is

$$(X_{k+1})_{1.} = \begin{cases} -\mathbf{1} & \text{if } k = 0 \\ g^{(k)}(\mathbf{1}) & \text{if } k > 0. \end{cases} \tag{27}$$

# 9  Computation of $\mu$; i.e., column means

For each column $j = 1, 2, \ldots, n$,
$$\mu_j = E[X_{.j}], \tag{28}$$

and $\boldsymbol{\mu} = [\mu_1, \mu_2, \ldots, \mu_n]^T \in \mathbb{R}^n$. Hence we get

$$\boldsymbol{\mu} = \frac{1}{n} X_{k+1}^T \mathbf{1} = \frac{1}{n} \text{cumsum}((X_{k+1})_{1.}), \tag{29}$$

where $(X_{k+1})_{1.}$ is defined above in 8.

# 10 Computation of $\sigma$; i.e., column standard deviations

For each column $j = 1, 2, \ldots, n$,

$$\sigma_j = \sqrt{E[X_{.j}^2] - E[X_{.j}]^2} \tag{30}$$

$$= \sqrt{\frac{1}{n} \sum_{i=1}^{n} X_{ij}^2 - (\frac{1}{n} \sum_{i=1}^{n} X_{ij})^2}. \tag{31}$$

Hence, $\boldsymbol{\sigma} = [\sigma_1, \sigma_2, \ldots, \sigma_n]^T \in \mathbb{R}^n$ becomes

$$\boldsymbol{\sigma} = \sqrt{E[X^2] - E[X]^2} \tag{32}$$

$$= \sqrt{\frac{1}{n} \text{colSums}(X^2) - (\frac{1}{n} \text{colSums}(X))^2} \tag{33}$$

$$= \sqrt{\frac{1}{n} (X^2)^T \mathbf{1} + (\frac{1}{n} X^T \mathbf{1})^2}, \tag{34}$$

where the first term involves 8 and the second term is computed in 9. Note that in the algorithm, we set the column standard deviation 1 when the column has variance 0 for computation convenience.

# 11 Computation on $(\hat{X}^2)^T 1$; i.e., colSums$(\hat{X}^2)$

We want to compute colSums$(\hat{X}^2)$, where $\hat{X}$ is scaled by both column means $\boldsymbol{\mu} = [\mu_1, \mu_2, \ldots, \mu_n]^T \in \mathbb{R}^n$ and column standard deviations $\boldsymbol{\sigma} = [\sigma_1, \sigma_2, \ldots, \sigma_n]^T \in \mathbb{R}^n$. We define $\hat{X} \in \mathbb{R}^{n \times n}$ such that for each $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, n$,

$$\hat{X}_{ij} = (X_{ij} - \mu_j)/\sigma_j,$$

where $X = X_{k+1} = (\hat{D}_{k+1})^{-1}$ if the order is $k$.

# 12 Conclusion

Computation details from section 6 to section 11 explain how we can benefit from the unique structure of matrices from trend filtering problem. As shown by our formula, we do not need to form any matrix and complete *SuSiE* algorithm with $O(n)$ complexity.

# References

S.-J. Kim, K. Koh, S. Boyd, and D. Gorinevsky. $\ell_1$ trend filtering. *SIAM Review*, 51(2):339–360, 2009.

R. J. Tibshirani. Adaptive piecewise polynomial estimation via trend filtering. *Annals of Statistics*, 42(1):285–323, 2014.