Inverse Problems in Science and Engineering

ISSN: 1741-5977 (Print) 1741-5985 (Online) Journal homepage: http://www.tandfonline.com/loi/gipe20

Recursive SURE for iterative reweighted least square algorithms

Feng Xue, Anatoly G. Yagola, Jiaqi Liu & Gang Meng

To cite this article: Feng Xue, Anatoly G. Yagola, Jiaqi Liu & Gang Meng (2016) Recursive SURE for iterative reweighted least square algorithms, Inverse Problems in Science and Engineering, 24:4, 625-646, DOI: 10.1080/17415977.2015.1054822

To link to this article: http://dx.doi.org/10.1080/17415977.2015.1054822

Published online: 18 Jun 2015.

Inverse Problems in Science and Engineering, 2016
Vol. 24, No. 4, 625–646, http://dx.doi.org/10.1080/17415977.2015.1054822

Recursive SURE for iterative reweighted least square algorithms

Feng Xue[a]*, Anatoly G. Yagola[b], Jiaqi Liu[a] and Gang Meng[a]

[a]*National Key Laboratory of Science and Technology on Test Physics and Numerical Mathematics, Beijing, China;* [b]*Faculty of Physics, Department of Mathematics, Moscow State University, Moscow, Russia*

(*Received 7 November 2014; final version received 4 April 2015*)

Iterative re-weighted least square (IRLS) algorithms for ℓ_1-minimization problems require to select proper value of regularization parameter, for which Stein's unbiased risk estimate (SURE) – an unbiased estimate of prediction error – is often used as a criterion for this selection. In this paper, we propose a recursive SURE to estimate the prediction error during the IRLS iterations. Particularly, we derive the recursion of Jacobian matrix by incorporating matrix splitting scheme into IRLS algorithms. Numerical examples demonstrate that minimizing SURE consistently leads to the nearly optimal reconstructions in terms of prediction error. Theoretical derivations in this work related to the evaluation of Jacobian matrix can be extended, in principle, to other types of regularizers and regularized iterative reconstruction algorithms.

Keywords: Stein's unbiased risk estimate (SURE); iterative re-weighted least squares (IRLS); matrix-splitting

AMS Subject Classifications: 49N45; 49N60; 62J07

1. Introduction

1.1. *Problem description*

We consider the following linear model [1]:

$$y = Ax + \epsilon, \quad \mu = Ax \tag{1}$$

where $y \in \mathbb{R}^n$ is the observed data or the response vector, $A = [A_1, A_2, \ldots, A_p] \in \mathbb{R}^{n\times p}$ is a deterministic design matrix with the column vectors A_i representing predictors or features, $x \in \mathbb{R}^p$ is the vector of unknown coefficients and ϵ is a vector of i.i.d. centered Gaussian random variable with variance $\sigma^2 > 0$. The regression problem is to estimate the value x and μ.[2] To improve the prediction accuracy of the ordinary least square and ridge regression,[3,4] it is typical to promote the sparsity of the unknown vector x, which is often formulated as the ℓ_1-penalized linear regression [4–6]:

$$P(\lambda): \quad \min_{x\in\mathbb{R}^p} \underbrace{\frac{1}{2}\|y - Ax\|_2^2 + \lambda\|x\|_1}_{\mathcal{L}(x)} \tag{2}$$

*Corresponding author. Email: fxue2012@gmail.com

where $\lambda > 0$ is regularization parameter and $\|\cdot\|_2$ (resp. $\|\cdot\|_1$) denotes the ℓ_2 (resp. ℓ_1) norm. An important feature of the problem (2) is that, depending on the regularization parameter λ, some coefficients of x are exactly set to zero.[7] The sparsity-promoting formulation (2) can be found in many applications of inverse problems, the typical examples are listed as follows.

- **model selection:** one often would like to determine a small subset within a large number of predictors that exhibits the strongest effects [4];
- **pattern recognition:** it is preferable to select only a few features that could represent a wide range of object classes [3,8];
- **signal recovery:** natural images typically have sparse representation in some transform domain (e.g. wavelet),[9] which gave rise to a great many sparse reconstruction algorithms, see [5,10–12] for example;
- **compressed sensing:** the sparsity of a signal can be exploited to recover it from far fewer samples than required by the Shannon–Nyquist sampling theorem.[13–15]

Notice that the problem (2) can be solved via standard convex optimization algorithms, such as *interior point* (IP) method [16] and *Least angle regression* (LARS) algorithm.[17] In signal processing, *iterative shrinkage thresholding* (IST) [18] and its variants (e.g. thresholded-Landweber,[19] FISTA [10] and NESTA [20]) are probably the most popular algorithms to solve (2) because of its simplicity. The basic iterative scheme of IST is [5,10]:

$$x^{(i+1)} = \mathcal{T}_{\lambda\tau}\left(x^{(i)} + \tau A^{\mathrm{T}}(y - Ax^{(i)})\right) \tag{3}$$

where $\mathcal{T}_T(\cdot) = \mathrm{sign}(\cdot)(|\cdot| - T)_+$ is a soft-threshold function with the step-size τ. The convergence analysis has been well studied in [12,18]. However, it is also recognized that IST-family algorithms have slow convergence speed.[5,10,21]

Another class of algorithms – *iterative re-weighted least square* (IRLS) – has been proposed for solving (2). Not limited to the ℓ_1-minimization, IRLS can also be used for computing local minima of the non-convex problems.[22] This algorithm consists of solving a sequence of weighted ℓ_1-minimization problems where the weights used for the next iteration are computed from the value of the current solution.[23] The convergence analysis of IRLS tailored for ℓ_1-minimization has been provided in [24]. It is also observed that the IRLS-type algorithms exhibit superior performance in terms of convergence speed among various experiments.[5,24] For this reason, we choose IRLS to solve (2) in this paper, which will be described in Section 2.2.

These regularized iterative reconstruction algorithms (e.g. IST and IRLS) often require selection of appropriate value for regularization parameter λ. For problem (2), typically, the estimation accuracy is governed by the parameter λ that controls the bias-variance trade-off (or equivalently, the balance between data smoothing and noise amplification). A number of criteria for this selection have been proposed, e.g. discrepancy principle,[25] the L-curve method,[26] generalized cross validation (GCV).[27] However, all of them are either computationally expensive or difficult to evaluate. Squared-error-based criterion can be a promising alternative to others, since the prediction accuracy is often quantified in terms of *expected prediction error* (EPE for short) [3,6,28]:

$$\mathrm{EPE} = \mathbb{E}\left\{\left\|\widehat{\mu}_\lambda - \mu\right\|_2^2\right\} \tag{4}$$

where $\widehat{\mu}_\lambda$ denotes the estimate of μ.

To this end, in the present paper, rather than proposing more efficient algorithm to solve (2), we will focus on selecting an appropriate value of λ for IRLS algorithm, such that the predicted value $\widehat{\mu}_\lambda$ minimizes the EPE (4).

1.2. *Related works*

Notice that we cannot directly minimize EPE (4) to find optimal λ, since the true value μ is unknown. Efron et al. [17] shows that the prediction risk (4) can be expressed as:

$$\mathrm{EPE} = \mathbb{E}\left\{\left\|\widehat{\mu}_\lambda - y\right\|_2^2 - n\sigma^2 + 2\sigma^2 \mathrm{df}(\widehat{\mu}_\lambda)\right\}$$

where $\mathrm{df}(\cdot)$ denotes the degrees of freedom, defined by $\mathrm{df}(\widehat{\mu}_\lambda) = \frac{1}{\sigma^2}\sum_{m=1}^{n} \mathrm{cov}\big((\widehat{\mu}_\lambda)_m, y_m\big)$.[29,30] This estimate is also termed as C_p-type statistics by Mallows [31]. However, this procedure is difficult to apply to evaluate the degrees of freedom.

CM Stein proposed an extremely useful formula to simplify the degrees of freedom, which can be replaced by a statistically unbiased estimate [28,30]:

$$\mathrm{df}(\widehat{\mu}_\lambda) = \sum_{m=1}^{n} \frac{\partial(\widehat{\mu}_\lambda)_m}{\partial y_m} \triangleq \mathrm{div}_y \widehat{\mu}_\lambda$$

It was first proposed by CM Stein, phrased in the famous *Stein's lemma*.[32] It gave rise to *Stein's unbiased risk estimate* (SURE), expressed as [7,30,33–35]:

$$\mathrm{SURE}(\widehat{\mu}_\lambda) = \left\|\widehat{\mu}_\lambda - y\right\|_2^2 - n\sigma^2 + 2\sigma^2 \mathrm{div}_y \widehat{\mu}_\lambda \tag{5}$$

The SURE depends solely on y, without prior knowledge of x. This can prove to be very useful as a basis to automatically choose the parameter λ of the estimator.[35] The SURE has been widely used in the statistics and signal processing communities, as a principled and efficient way for parameter selection with a variety of non-linear estimators, e.g. wavelet denoising,[36] image deconvolution [37,38] and non-local filtering.[39]

Essentially, the IST and IRLS algorithms can be regarded as the non-linear estimators as well. For the IST-family algorithms, in [40,41], the parameter λ is optimized by SURE minimization. Particularly, the iteration number is selected at the highest SNR improvement, which is monitored by the SURE computation at each iteration. The optimal threshold is also optimized for a given number of iterations. For the IRLS algorithms, in [42], λ is updated at each iteration by minimizing SURE. However, the derivation of SURE in [42] is problematic, since the estimate $\mu^{(i)}$ at ith iteration is NOT a linear function of the observed data y. Hence, SURE cannot be readily computed by a closed-form solution, as described in [42]. We will point out the incorrect derivation of [42] in Section 2, and show the results of [42] in Section 4.3 as our comparisons.

The difficulty of computing the last divergence term of (5) has been mentioned in [3,43]. Though *Bootstrap* [3] and *Monte–Carlo* methods [43] can be used to compute, they are always computationally expensive. In addition, for the iterative algorithms (e.g. IST or IRLS), it will be more difficult to evaluate the divergence than non-iterative methods (e.g. [36,37]).

1.3. *Key contributions of this work*

In this paper, we first consider the problem (2) with given λ, and propose a new approach to evaluate the SURE for IRLS (see Algorithm 1). Then, we repeatedly perform this algorithm for various values of λ, and finally select the optimal value that yields the minimum SURE (see Algorithm 2).

In particular, we apply a matrix-splitting strategy to solve each iteration of IRLS, which decouples the data-dependent term (i.e. depending on y) and linearizes the function of the observed data y. Thus, the SURE computation is dramatically simplified: it can be performed in a linearly recursive manner during the IRLS iterations.

1.4. *Paper organization*

In Section 2, we present the preliminary knowledge of Jacobian matrix, IRLS algorithms and basic SURE formulation; we also explain why the SURE derivation of [42] is incorrect, which necessitates the proposed recursive SURE for the valid evaluation. Section 3 derives a recursive SURE for IRLS, by which one can monitor the prediction error at each iteration, without referencing the unknown true data μ. Section 4 shows many numerical behaviours of the algorithm and presents the experimental results of the numerical examples. Some concluding remarks are summarized in Section 5.

To keep consistency throughout the paper, we use lowercase letter to denote vector, and uppercase for matrix. The superscript T denotes the transpose of vector or matrix. We denote the data dimension by n and p, use subscripts m and k as the entry index of vector or matrix. The iteration is indexed by i and j.

2. Basic ingredients

2.1. *Basic properties of Jacobian matrix*

For the estimate $\widehat{\mu}_\lambda$ of $\mu \in \mathbb{R}^n$, we define the Jacobian matrix $J\big(\widehat{\mu}_\lambda, y\big) \in \mathbb{R}^{n\times n}$ as:

$$\Big[J(\widehat{\mu}_\lambda, y)\Big]_{m,k} = \frac{\partial\big(\widehat{\mu}_\lambda\big)_m}{\partial y_k}$$

for its (m, k)th element. Hence, the SURE of (5) can be rewritten as:

$$\mathrm{SURE}(\widehat{\mu}_\lambda) = \big\|\widehat{\mu}_\lambda - y\big\|_2^2 - n\sigma^2 + 2\sigma^2\mathrm{Tr}\big(J(\widehat{\mu}_\lambda, y)\big) \tag{6}$$

where Tr denotes trace of matrix. Recalling that $\widehat{\mu}_\lambda = A\widehat{x}_\lambda$, we have:

$$\Big[J\big(\widehat{\mu}_\lambda, y\big)\Big]_{m,k} = \frac{\partial\sum_{s=1}^p A_{m,s}\big(\widehat{x}_\lambda\big)_s}{\partial y_k} = \sum_{s=1}^p A_{m,s}\frac{\partial\big(\widehat{x}_\lambda\big)_s}{\partial y_k} = \sum_{s=1}^p A_{m,s}\Big[J\big(\widehat{x}_\lambda, y\big)\Big]_{s,k}$$

which implies a basic property $J\big(\widehat{\mu}_\lambda, y\big) = AJ\big(\widehat{x}_\lambda, y\big)$, where $J\big(\widehat{x}_\lambda, y\big) \in \mathbb{R}^{p\times n}$. Finally, the SURE of $\widehat{\mu}_\lambda$ can be expressed in terms of $\widehat{x}_\lambda$, i.e.

$$\mathrm{SURE}(\widehat{\mu}_\lambda) = \big\|A\widehat{x}_\lambda - y\big\|_2^2 - n\sigma^2 + 2\sigma^2\mathrm{Tr}\big(AJ(\widehat{x}_\lambda, y)\big) \tag{7}$$

Therefore, in the following sections, we will focus on the estimation of $\widehat{x}_\lambda$ and the corresponding Jacobian matrix $J(\widehat{x}_\lambda, y)$, from which SURE of $\widehat{\mu}_\lambda$ can be readily obtained by (7).

2.2. *IRLS algorithms*

To solve the problem (2), the IRLS algorithm updates x by minimizing [5,24,42]:

$$x^{(i)} = \arg\min_x \frac{1}{2}\|y - Ax\|_2^2 + \frac{\lambda}{2}x^{\mathrm{T}} W_{(i-1)} x \tag{8}$$

at ith iteration, where $W \in \mathbb{R}^{p\times p}$ is a diagonal matrix and $W_{m,m} = 1/\left|x_m^{(i)}\right|$ for $m = 1, 2, \ldots, p$.[1]

The derivation of IRLS can be interpreted from the viewpoint of *majorization minimization* (MM) framework,[44] which majorizes the ℓ_1-penalty term $\|x\|_1$. Consider $\frac{1}{2}|\tilde{x}| + \frac{1}{2|\tilde{x}|}x^2$, which is an arithmetic average of $|\tilde{x}|$ and $\frac{1}{|\tilde{x}|}x^2$; it is lower bounded by the geometric average $|x|$; hence, $\frac{1}{2|\tilde{x}|}x^2$ is a majorizer of $|x|$ (up to a constant term independent of x). Likewise, a majorization function for the ℓ_1-term $\|x\|_1$ is $\frac{1}{2}x^{\mathrm{T}}Wx$, where W is given in (8).

Equation (8) is equivalent to solving iteration-dependent linear systems of the form:

$$\left(A^{\mathrm{T}}A + \lambda W_{(i-1)}\right)x^{(i)} = A^{\mathrm{T}}y \tag{9}$$

which leads to the following update of $x^{(i)}$:

$$x^{(i)} = \underbrace{\left(A^{\mathrm{T}}A + \lambda W_{(i-1)}\right)^{-1}A^{\mathrm{T}}}_{B_{(i-1)}}y \tag{10}$$

which is the basic iterative step of IRLS. From (10), we can see that $x^{(i)}$ is NOT a linear transformation of y, since $W_{(i-1)}$ (and $B_{(i-1)}$) is constructed by $x_m^{(i-1)}$, which is dependent on data y.

2.3. *SURE formulation for ridge regression*

Considering the estimation of x in linear model (1), the ridge regression is given as [3,27]:

$$\widehat{x}_\lambda = \underbrace{\left(A^{\mathrm{T}}A + \lambda I\right)^{-1}A^{\mathrm{T}}}_{B_\lambda}y \tag{11}$$

for ridge parameter λ, where I denotes identity matrix. In this case where the matrix B_λ is independent of y, the Jacobian matrix of $\widehat{x}_\lambda$ is $J(\widehat{x}_\lambda, y) = B_\lambda$ by definition, and the SURE of $\widehat{\mu}_\lambda$ becomes:

$$\mathrm{SURE}(\widehat{\mu}_\lambda) = \left\|A\widehat{x}_\lambda - y\right\|_2^2 - n\sigma^2 + 2\sigma^2\mathrm{Tr}\left(AB_\lambda\right) \tag{12}$$

from (7).

In [3,38], the parameter λ is optimized by minimizing (12). For the ith update $\mu^{(i)} = Ax^{(i)} = AB_{(i-1)}y$ of IRLS (10), the authors of [42] made a similar derivation:

$$\mathrm{SURE}(\mu^{(i)}) = \left\|Ax^{(i)} - y\right\|_2^2 - n\sigma^2 + 2\sigma^2\mathrm{Tr}\left(AB_{(i-1)}\right)$$

Unfortunately, this formula is incorrect, due to the non-linearity between $x^{(i)}$ and y, as pointed out in Section 2.2. Thus, $J(x^{(i)}, y) \neq B_{(i-1)}$, and therefore, $J(\mu^{(i)}, y) \neq AB_{(i-1)}$.

3. Recursive SURE for IRLS algorithms

3.1. *Matrix-splitting strategy to IRLS*

3.1.1. *Analysis of IRLS iteration*

Considering the update Equation (9) of IRLS, for the 1st iteration of IRLS (i.e. $i = 1$), if we initialize $W_{(0)}$ as identity matrix: $W_{(0)} = I$, the solution to (9) is given as:

$$x^{(1)} = \underbrace{\left(A^{\mathrm{T}}A + \lambda W_{(0)}\right)^{-1} A^{\mathrm{T}}}_{B_{(0)}} y$$

which is exactly the ridge regression (11), where matrix $B_{(0)}$ is independent of data y. Hence, the Jacobian matrix is $J(x^{(1)}, y) = B_{(0)}$. The SURE of this step can be easily computed by (12).

In the second and the successive iterations (i.e. $i \geq 2$), $x^{(i)}$ is updated by (10), however, $J(x^{(i)}, y) \neq B_{(i-1)}$ for $i \geq 2$, as emphasized in Section 2.3. Since the data-dependent term $\lambda W_{(i-1)}$ is deeply involved in the matrix $B_{(i-1)}$, the Jacobian matrix $J(x^{(i)}, y)$ is not easily derived from the direct solution (10). So we need to use alternative strategy to solve (9), by decoupling diagonal matrix $W_{(i-1)}$ from $B_{(i-1)}$, in order to conveniently compute $J(x^{(i)}, y)$.

3.1.2. *Matrix splitting to solve IRLS iteration*

Now, consider Equation (9) for fixed iteration step i. To compute $J(x^{(i)}, y)$, instead of the closed-form solution (10), we apply the matrix-splitting scheme [45] to solve (9), which splits the matrix $A^{\mathrm{T}}A + \lambda W_{(i-1)}$ as:

$$A^{\mathrm{T}}A + \lambda W_{(i-1)} = \underbrace{(D + \lambda W_{(i-1)})}_{M} - \underbrace{(D - A^{\mathrm{T}}A)}_{N} \tag{13}$$

for any matrix D, and the approximate solution $x^{(i,j)}$ is generated as follows (indexed by j):

$$Mx^{(i,j)} = Nx^{(i,j-1)} + A^{\mathrm{T}}y$$

or equivalently,

$$x^{(i,j)} = M^{-1}\left(Nx^{(i,j-1)} + A^{\mathrm{T}}y\right) \tag{14}$$

provided that matrix M is invertible. The iterative method (14) is convergent to the unique solution of (10) for any initial $x^{(i,0)}$, if and only if the spectral radius $\rho(M^{-1}N) < 1$. Refer to [45,46] for the convergence analysis.

3.1.3. *Convergence analysis of matrix splitting*

For efficient computation and decoupling of $W_{(i-1)}$, M should be chosen as an easily invertible matrix. Hence, we choose $D = \alpha I \in \mathbb{R}^{p\times p}$ here, such that M is diagonal. The following theorem states that for a certain value of α, $\rho(M^{-1}N)$ is guaranteed to be smaller than 1.

THEOREM 3.1 *If $\alpha > \gamma_{\max}(A^{\mathrm{T}}A)$, where $\gamma_{\max}(A^{\mathrm{T}}A)$ denotes the maximum eigenvalue of $A^{\mathrm{T}}A$, then, matrices M and N in (13) always satisfy $\rho(M^{-1}N) < 1$.*

Proof Under the above conditions, it is easy to verify that both matrices M and N are positive definite. Then, by the definition of spectral radius, we have:

$$\begin{aligned}\rho(M^{-1}N) &\triangleq \sup_{\|x\|_2=1} \|M^{-1}Nx\|_2 \\ &\leq \sup_{\|x\|_2=1} \|M^{-1}\|_2 \cdot \|Nx\|_2 \\ &= \|M^{-1}\|_2 \cdot \sup_{\|x\|_2=1} \|Nx\|_2 \\ &= \|M^{-1}\|_2 \cdot \|N\|_2 \\ &= \frac{\alpha - \gamma_{\min}(A^{\mathrm{T}}A)}{\alpha + \lambda w} < 1 \end{aligned} \tag{15}$$

where $\|N\|_2 = \alpha - \gamma_{\min}(A^{\mathrm{T}}A)$ provided that $\alpha > \gamma_{\max}(A^{\mathrm{T}}A)$, w is the smallest diagonal element (also minimum eigenvalue) of $W_{(i-1)}$. □

Theorem 3.1 shows that if we choose $D = \alpha I$ with $\alpha > \gamma_{\max}(A^{\mathrm{T}}A)$, the matrix-splitting iterative algorithm (14) is bound to converge to the true solution (10).

It is also observed in [45,46] that for large values of iteration number j, the solution error decreases in magnitude approximately by a factor of $\rho(M^{-1}N)$ at each iteration step: the smaller is the $\rho(M^{-1}N)$, the quicker is the convergence of (14). Thus, $\rho(M^{-1}N)$ should be as small as possible for fast convergence speed.

For this particular problem setting, the value of $\rho(M^{-1}N)$ is determined by the two parameters α and λ. There is no direct link (e.g. monotonicity) between $\rho(M^{-1}N)$ and the parameters. However, (15) in the proof provides an upper bound of $\rho(M^{-1}N)$, and thus, gives a strong hint of how α and λ roughly affect $\rho(M^{-1}N)$. From (15), we can see that the upper bound of $\rho(M^{-1}N)$ is increasing of α, but decreasing w.r.t. λ. Therefore, for faster convergence of matrix splitting, we suggest to choose α as small as possible with the constraint $\alpha > \gamma_{\max}(A^{\mathrm{T}}A)$. Considering the problem (2) with fixed λ: one may expect the convergence to be slower for smaller value of λ.

3.1.4. *Numerical examples of matrix splitting*

In this part, we will implement the matrix splitting algorithm to solve (9) and discuss its numerical behaviours under various values of α and λ.

To specify Equation (9), we randomly generate matrix A, diagonal matrix $W \succ 0$ and vector $b = A^{\mathrm{T}}y$. The true solution is given in a closed form:

$$x_{\text{true}} = \left(A^{\mathrm{T}}A + \lambda W\right)^{-1} b$$

The matrix-splitting scheme gives the following iterates (indexed by j):

$$x^{(j+1)} = \underbrace{(\alpha I + \lambda W)}_{M}{}^{-1} \Big(\underbrace{(\alpha I - A^{\mathrm{T}}A)}_{N} x^{(j)} + b \Big)$$

with the randomly initialized $x^{(0)}$.

As suggested in Section 3.1.3, we evaluate α by a factor k_α of $\gamma_{\max}(A^{\mathrm{T}}A)$, i.e. $\alpha = k_\alpha \cdot \gamma_{\max}(A^{\mathrm{T}}A)$. We use the relative error $e_1^{(j)}$ (difference between two successive steps) and true error $e_2^{(j)}$ (difference between current step and true solution) as the stopping criteria:

$$e_1^{(j)} = \frac{\|x^{(j)} - x^{(j-1)}\|_2^2}{\|x^{(j)}\|_2^2}; \quad e_2^{(j)} = \frac{\|x^{(j)} - x_{\text{true}}\|_2^2}{\|x_{\text{true}}\|_2^2}$$

The iteration is terminated when the tolerance $e_1^{(j)} \leq 10^{-15}$.

Figure 1(a) shows that the spectral radius $\rho(M^{-1}N)$ is increasing w.r.t. α in this example, thus, the algorithm converges slower for larger value of α, as shown in Figure 1(b) and (c). We can see that it needs more than 1000 steps to converge for $\alpha = 100 \cdot \gamma_{\max}(A^{\mathrm{T}}A)$ and less than 100 steps for $\alpha = 1.1 \cdot \gamma_{\max}(A^{\mathrm{T}}A)$. Hence, we choose $\alpha = 1.1 \cdot \gamma_{\max}(A^{\mathrm{T}}A)$ in the remainder of the paper.

Figure 2(a) shows that $\rho(M^{-1}N)$ decreases with larger λ in this example. Figure 2(b) and (c) demonstrates that the algorithm needs only 6–7 steps to converge for $\lambda = 0.1$ and more than 100 steps for $\lambda = 10^{-5}$. It indicates that the algorithm converges slower for smaller λ.

Figures 1 and 2 also show that the true error $e_2^{(j)}$ is always consistent with the relative error $e_1^{(j)}$, which means that the algorithm finally converges to the true solution x_{true}. Hence, it is sufficient to use only $e_1^{(j)}$ as the stopping criterion.

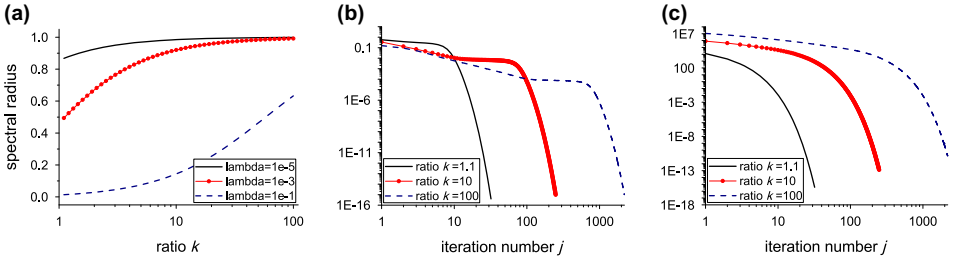

Figure 1. The convergence speed of matrix-splitting w.r.t. the value of α. (a) $\rho(M^{-1}N)$ as a function of α. (b) Relative error $e_1^{(j)}$, $\lambda = 1 \times 10^{-3}$. (c) True error $e_2^{(j)}$, $\lambda = 1 \times 10^{-3}$.

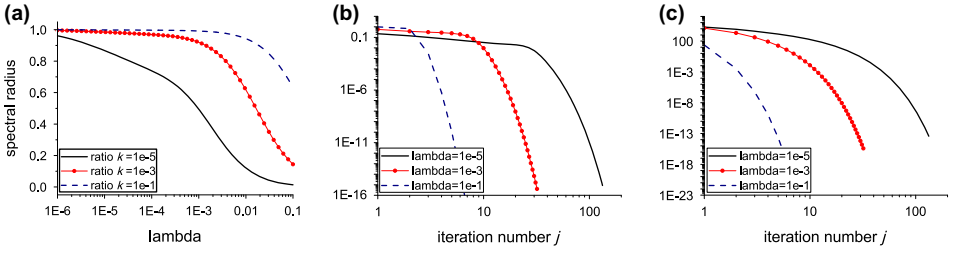

Figure 2. The convergence speed of matrix-splitting w.r.t. the value of λ. (a) $\rho(M^{-1}N)$ as a function of λ. (b) Relative error $e_1^{(j)}$, $k_\alpha = 1.1$. (c) True error $e_2^{(j)}$, $k_\alpha = 1.1$.

Table 1. $\rho(M^{-1}N)$ and iteration number for various values of λ and α.

λ	1×10^{-5}		1×10^{-1}	
k_α	$\rho(M^{-1}N)$	# of iterations	$\rho(M^{-1}N)$	# of iterations
1.1	0.8674	135	0.0141	9
10	0.9846	1081	0.1440	17
100	0.9985	9320	0.6331	61

Table 1 shows $\rho(M^{-1}N)$ and iteration number for other values of λ and α, where we can also see that the larger $\rho(M^{-1}N)$ is, the more iterations are required to reach convergence.

In the above experiments, we use $e_1^{(j)} = 10^{-15}$ as the stopping criterion for the strict convergence. In practice, there is no need to set such rigorous tolerance for efficient computation. When it is embedded in IRLS, this tolerance will affect the convergence speed of IRLS. It will be further investigated in Section 4.3.

3.2. *Recursion of Jacobian matrix*

We have verified that for a certain value of α, the matrix splitting is guaranteed to converge to the true solution. In this part, we will see that the computation of Jacobian matrix becomes much easier by the matrix splitting.

Rewrite (14) as $v = Hb$ for brevity, where $v = x^{(i,j)}$, $H = M^{-1}$ and $b = A^{\mathrm{T}}y + Nx^{(i,j-1)}$. The Jacobian matrix of (14) is derived as:

$$\big[J(v, y)\big]_{m,k} = \frac{\partial v_m}{\partial y_k} = \frac{\partial \sum_{s=1}^{p} H_{m,s} b_s}{\partial y_k} = \sum_{s=1}^{p} H_{m,s} \frac{\partial b_s}{\partial y_k} + \sum_{s=1}^{p} b_s \frac{\partial H_{m,s}}{\partial y_k} \tag{16}$$

for the (m, k)th entry of Jacobian matrix. The first term of (16) is:

$$\sum_{s=1}^{p} H_{m,s} \frac{\partial b_s}{\partial y_k} = \sum_{s=1}^{p} H_{m,s} \big[J(b, y)\big]_{s,k}$$

where $J(b, y) = A^{\mathrm{T}} + NJ(x^{(i,j-1)}, y)$ by the property of Jacobian matrix (see Section 2.1).

Considering the diagonal matrix H:

$$\frac{\partial H_{m,s}}{\partial y_k} = \frac{\partial \left[\left(\lambda W_{(i-1)} + \alpha \right)^{-1} \right]_{m,s}}{\partial y_k} = \begin{cases} \frac{\partial \left(\lambda \left| x_m^{(i-1)} \right|^{-1} + \alpha \right)^{-1}}{\partial y_k} & \text{if } m = s \\ 0 & \text{otherwise} \end{cases}$$

where

$$\begin{aligned} \frac{\partial \left(\lambda \left| x_m^{(i-1)} \right|^{-1} + \alpha \right)^{-1}}{\partial y_k} &= \frac{\partial \left(\lambda \left| x_m^{(i-1)} \right|^{-1} + \alpha \right)^{-1}}{\partial \left| x_m^{(i-1)} \right|} \cdot \frac{\partial |x_m^{(i-1)}|}{\partial x_m^{(i-1)}} \cdot \frac{\partial x_m^{(i-1)}}{\partial y_k} \\ &= \underbrace{\left(\lambda \left| x_m^{(i-1)} \right|^{-1} + \alpha \right)^{-2} \cdot \frac{\lambda}{\left| x_m^{(i-1)} \right|^2} \cdot \operatorname{sign}\left(x_m^{(i-1)} \right)}_{E_{m,m}} \cdot \frac{\partial x_m^{(i-1)}}{\partial y_k} \end{aligned}$$

the second term of (16) becomes:

$$\sum_{m=1}^{p} b_m \frac{\partial H_{m,m}}{\partial y_k} = \sum_{m=1}^{p} b_m E_{m,m} \frac{\partial x_m^{(i-1)}}{\partial y_k} = \sum_{m=1}^{p} b_m E_{m,m} \left[J(x_m^{(i-1)}, y) \right]_{m,k}$$

Finally, (16) becomes:

$$J\left(x^{(i,j)}, y\right) = M^{-1}A^{\mathrm{T}} + M^{-1}NJ\left(x^{(i,j-1)}, y\right) + CJ\left(x^{(i-1)}, y\right) \tag{17}$$

where diagonal matrix $C = \mathrm{diag}(b_m)\mathrm{diag}(E_{m,m}) \in \mathbb{R}^{p\times p}$, and its mth diagonal element is given as:

$$[C]_{m,m} = \left(A^{\mathrm{T}}y + Nx^{(i,j-1)}\right)_m \cdot \left(\frac{\lambda}{\left|x_m^{(i-1)}\right|} + \alpha\right)^{-2} \cdot \frac{\lambda}{\left|x_m^{(i-1)}\right|^2} \cdot \mathrm{sign}\left(x_m^{(i-1)}\right)$$

for $m = 1, 2, \ldots, p$. The key Equation (17) expresses the recursion of Jacobian matrix.

Thus, we can compute the SURE by (7) and (17) during the IRLS iterations:

$$\mathrm{SURE}(\mu^{(i)}) = \left\| Ax^{(i)} - y \right\|_2^2 - n\sigma^2 + 2\sigma^2 \mathrm{Tr}\left(AJ(x^{(i)}, y)\right)$$

and

$$\mathrm{SURE}(\mu^{(i,j)}) = \left\| Ax^{(i,j)} - y \right\|_2^2 - n\sigma^2 + 2\sigma^2 \mathrm{Tr}\left(AJ(x^{(i,j)}, y)\right)$$

where $x^{(i+1)} = x^{(i,\infty)}$ and $J(x^{(i+1)}, y) = J(x^{(i,\infty)}, y)$, assuming that matrix splitting is converged at $j = \infty$.

3.3. *Summary of IRLS algorithm with matrix-splitting strategy*

Finally, we summarize the proposed IRLS-MS-SURE algorithm as Algorithm 1, where the matrix splitting ('MS' for short) is embedded in every IRLS iteration. This algorithm enables us to solve the problem (2) with a prescribed value of λ, and simultaneously evaluate the SURE during the IRLS iterations.

For the embedded matrix splitting algorithm (i.e. inner iteration on j), we use $e^{(j)}$ as the stopping criterion:

$$e^{(j)} = \frac{\|x^{(i,j+1)} - x^{(i,j)}\|_2^2}{\|x^{(i,j)}\|_2^2} \leq \text{some tolerance} \qquad \text{for } j\text{th iteration with fixed } i$$

Since the IRLS algorithm (i.e. outer iteration on i) is to minimize the objective functional (2). We use the relative error of $\mathcal{L}(x)$:

$$e^{(i)} = \frac{\left|\mathcal{L}(x^{(i+1)}) - \mathcal{L}(x^{(i)})\right|}{\mathcal{L}(x^{(i)})} \leq \text{some tolerance} \qquad \text{for } i\text{th iteration}$$

as the stopping criterion.

To find the optimal λ with minimum SURE, Algorithm 2 uses a simple exhaustive search over a number of tentative values of λ, for each of them Algorithm 1 is repeatedly applied to compute the SURE.

Algorithm 1: IRLS-MS-SURE Algorithm

Input: objective function given as (2), observed data y, design matrix A, parameter λ
Output: IRLS estimate $\widehat{x}_\lambda$, $\widehat{\mu}_\lambda$, and SURE($\widehat{\mu}_\lambda$).
begin
  1 Initialize $W_{(0)} = I$;
  2 For 1st iteration ($i = 1$): compute $x^{(1)}$ and $J(x^{(1)}, y)$ by (11) and (12);
  while *stopping criterion not met (iterations on i)* **do**
    (1) set $i = 2$, initialize $x^{(i,0)} = x^{(i-1)}$ and $J(x^{(i,0)}, y) = J(x^{(i-1)}, y)$ for $j = 0$, compute $W_{(i-1)}$ by $x_{(i-1)}$;
    while *stopping criterion not met (iterations on j)* **do**
      ① compute $x^{(i,j)}$ by (14);
      ② update $J(x^{(i,j)}, y)$ by (17);
      ③ $j := j + 1$;
    end
    (2) $x^{(i+1)} = x^{(i,j+1)}$, $J(x^{(i+1)}, y) = J(x^{(i,j+1)}, y)$;
    (3) $i := i + 1$;
  end
  3 $\widehat{x}_\lambda = x^{(i+1)}$, $\widehat{\mu}_\lambda = A\widehat{x}_\lambda$, compute SURE($\widehat{\mu}_\lambda$) by (7).
end

Algorithm 2: Optimization of λ for IRLS Algorithm

Input: objective function given as (2), observed data y, design matrix A
Output: optimal λ_{opt}, the corresponding IRLS estimates $\widehat{x}_{\lambda_{\text{opt}}}$ and $\widehat{\mu}_{\lambda_{\text{opt}}}$
begin
  1 Set K sample values of λ: $\lambda_1, \lambda_2, \ldots, \lambda_K$;
  for $k = 1, 2, \ldots, K$ **do**
    perform **Algorithm 1** with given λ_k, to obtain $\widehat{x}_{\lambda_k}$, $\widehat{\mu}_{\lambda_k}$ and SURE($\widehat{\mu}_{\lambda_k}$)
  end
  2 find minimum value of SURE, and optimal $\lambda_{\text{opt}} = \arg\min_{\lambda_k} \text{SURE}(\widehat{\mu}_{\lambda_k})$.
  3 compute $\widehat{x}_{\lambda_{\text{opt}}}$ and $\widehat{\mu}_{\lambda_{\text{opt}}}$ by **Algorithm 1** with λ_{opt}.
end

4. Experimental results and discussions

4.1. *Experimental setting*

In this section, we randomly generate the matrix $A \in \mathbb{R}^{300\times 500}$, and set $x \in \mathbb{R}^{500}$ as a sparse vector with very few non-zeros (in this example, 10 non-zeros elements). Then, we add the noise ϵ with noise variance σ^2 to obtain the observed data $y = Ax + \epsilon$, such that the input SNR is 10 dB.[2] Now, we are going to apply Algorithms 1 and 2 to solve (2).

4.2. *Matrix-splitting scheme in IRLS*

In this part, we will show the results of matrix-splitting algorithm (14) during the IRLS. As suggested in Section 3.1.3, we set $\alpha = 1.1 \cdot \gamma_{\max}(A^{\mathrm{T}}A)$ for fast convergence and $e^{(j)} \leq 10^{-10}$ as the stopping criterion.

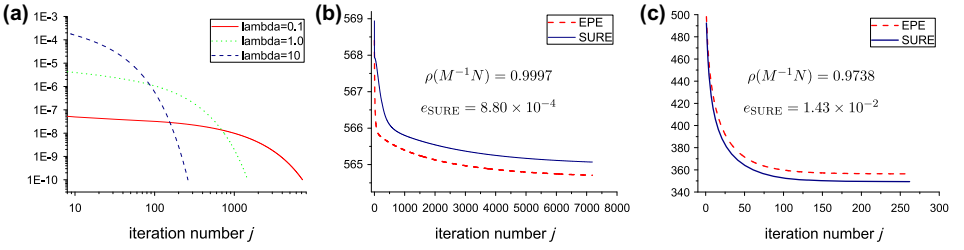

Figure 3. The convergence of matrix splitting in second iteration of IRLS ($i = 2$). (a) Relative error $e^{(j)}$. (b) SURE and *oracle* EPE for $\lambda = 0.1$. (c) SURE and *oracle* EPE for $\lambda = 10$.

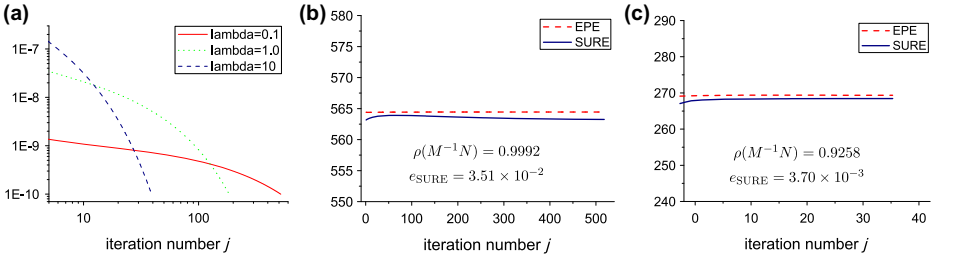

Figure 4. The convergence of matrix splitting in 10th iteration of IRLS ($i = 10$). (a) Relative error $e^{(j)}$. (b) SURE and *oracle* EPE for $\lambda = 0.1$. (c) SURE and *oracle* EPE for $\lambda = 10$.

Figure 3 shows the convergence of matrix splitting during the second outer iteration of IRLS (i.e. $i = 2$) for $\lambda = 0.1, 1.0$ and 10. Figure 3(a) shows the relative error $e^{(j)}$ decreasing to below 10^{-10}. We can see that the smaller $\rho(M^{-1}N)$ (i.e. larger λ) leads to faster convergence of matrix splitting. Figure 3(b) and (c) shows the SURE and corresponding EPE for $\lambda = 0.1$ and 10, respectively. We use maximum error $e_{\mathrm{SURE}} = \max_j \frac{|\mathrm{EPE}^{(j)} - \mathrm{SURE}^{(j)}|}{|\mathrm{EPE}^{(j)}|}$ among all the iterations j, to quantify the closeness of SURE to EPE. The small errors indicate that the SURE always coincides with EPE very well.

Figure 4 shows the case of 10th iteration of IRLS (i.e. $i = 10$). Comparing Figure 4 with Figure 3, we can see that the matrix splitting converges significantly faster as the IRLS iteration i proceeds. The reason behind it is probably that $x^{(i)}$ tends to be sparser with the IRLS iteration i: the increments of the diagonal elements of $W^{(i)}$ result in the reduction of $\rho(M^{-1}N)$.

It is not advised to discuss the tolerance sensitivity of matrix splitting for the moment, since this tolerance has great influence on the convergence speed and computational time of the IRLS, when matrix splitting is embedded in IRLS. We will give a thorough discussion on this problem in Section 4.3.

4.3. *Convergence of IRLS algorithm and comparison with IST*

In this part, we will show the convergence of IRLS algorithm and the comparisons with IST. We perform the standard IST algorithm iterated as Equation (3), where we choose the step-size $\tau = 10^{-4}$. We set $e^{(i)} \leq 10^{-7}$ as the stopping criterion for both IRLS and IST algorithms.

We compare the following iterative algorithms in terms of both convergence speed and computational time:

- IST: standard IST with iterates (3);
- IRLS: standard IRLS with direct solution (10), rather than matrix-splitting (14);
- IRLS-MS: use matrix-splitting (14) to solve each IRLS iteration;
- IRLS-MS-SURE: perform matrix-splitting (14), and compute SURE (17) simultaneously, during each IRLS iteration.

To study the influence of the MS tolerance $e^{(j)}$ upon the IRLS convergence, we test the following four tolerances of $e^{(j)}$:

- IRLS-MS-1 and IRLS-MS-SURE-1: $e^{(j)} \leq 10^{-1}$;
- IRLS-MS-2 and IRLS-MS-SURE-2: $e^{(j)} \leq 10^{-5}$;
- IRLS-MS-3 and IRLS-MS-SURE-3: $e^{(j)} \leq 10^{-10}$;
- IRLS-MS-4 and IRLS-MS-SURE-4: $e^{(j)} \leq 10^{-15}$;

From Figure 5(a) and (b), we can see that IRLS converges substantially faster than IST by two orders of magnitude in terms of the outer iteration number. When the direct solution (10) is replaced by matrix-splitting scheme, the IRLS converges much slower:

(1) IRLS has slower convergence for looser tolerance $e^{(j)}$;
(2) The stricter tolerance is $e^{(j)}$, the less iterations IRLS needs to converge. The direct solution (10) can also be regarded as an extreme case, where the tolerance $e^{(j)} = 0$ for the exact solution.

On the other hand, the stricter tolerance $e^{(j)}$ requires more iterations of matrix splitting, as shown in Figures 3 and 4. It reflects that a strict tolerance $e^{(j)}$ will save a great many IRLS iterations at the cost of considerably increasing iterations of matrix splitting. It needs much more MS iterations to solve each IRLS iteration, if less steps are required to converge for IRLS. Hence, there is a trade-off between the inner matrix splitting and outer IRLS in terms of whole computational complexity and time. We will see it later.

Figure 5(c) shows the SURE (and the *oracle* EPE) of the estimate $\mu^{(i)}$ during the IRLS iterations (the case of $e^{(j)} \leq 10^{-10}$). The error of SURE is 7.13×10^{-3}, which demonstrates that the SURE is a very accurate estimator of true prediction error. We also present the result of [42]: the SURE computed by [42] is not a valid estimator of the true prediction error, as we pointed out in Section 2.

Figures 6 and 7 show the IRLS convergence for $\lambda = 1.0$ and 10. In addition, Figures 5–7 also show the sensitivities of IRLS to the tolerance $e^{(i)}$. For efficient implementation, we often do not need to use the strict tolerance $e^{(i)} \leq 10^{-7}$. Typically, we use $e^{(i)} \leq 10^{-4}$ instead, since Figures 5–7 show that 10^{-4} – tolerance could greatly reduce the iterations to converge, while keeping the similar solution accuracy. Comparing the two solutions: $\widehat{x}_1$ (terminated when $e^{(i)} \leq 10^{-7}$) and $\widehat{x}_2$ (terminated when $e^{(i)} \leq 10^{-4}$), we calculated that the relative difference $\frac{\|\widehat{x}_1 - \widehat{x}_2\|_2^2}{\|\widehat{x}_1\|_2^2} \approx 10^{-9}$, which implies the negligible difference between $\widehat{x}_1$ and $\widehat{x}_2$. Hence, we choose $e^{(i)} \leq 10^{-4}$ in the following experiments for computational efficiency.

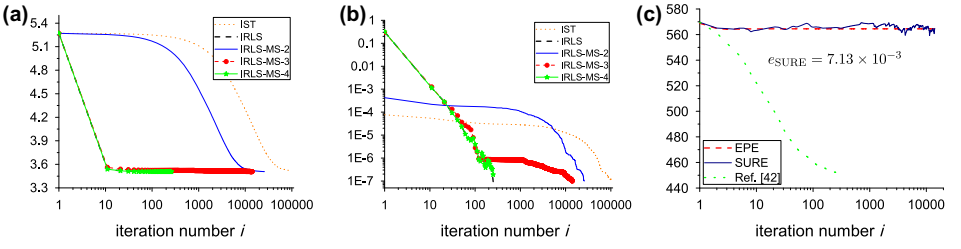

Figure 5. The convergence of IRLS for $\lambda = 0.1$. (a) Objective functional $\mathcal{L}(x^{(i)})$. (b) Relative error $e^{(i)}$. (c) SURE and *oracle* EPE of $\mu^{(i)}$.

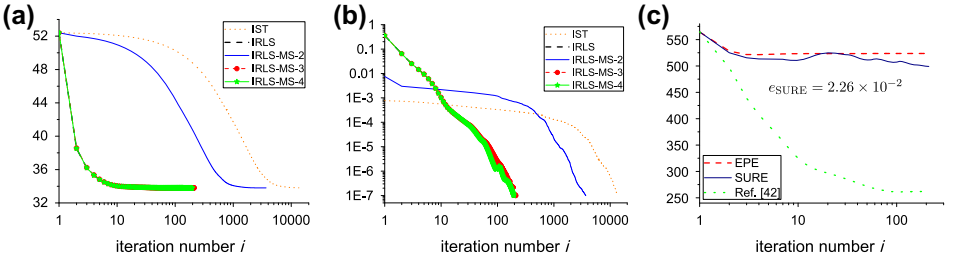

Figure 6. The convergence of IRLS for $\lambda = 1.0$. (a) Objective functional $\mathcal{L}(x^{(i)})$. (b) Relative error $e^{(i)}$. (c) SURE and *oracle* EPE of $\mu^{(i)}$.

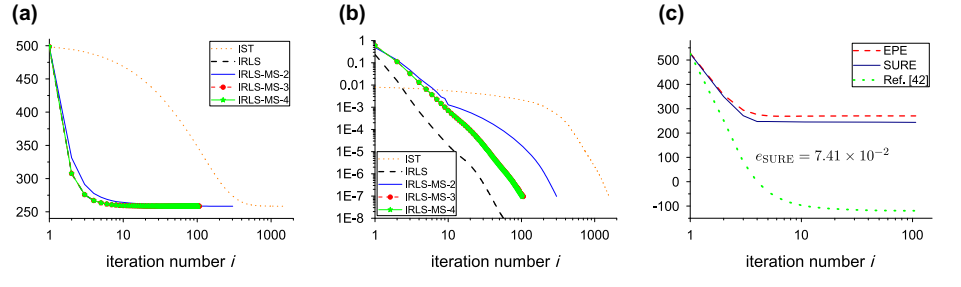

Figure 7. The convergence of IRLS for $\lambda = 10$. (a) Objective functional $\mathcal{L}(x^{(i)})$. (b) Relative error $e^{(i)}$. (c) SURE and *oracle* EPE of $\mu^{(i)}$.

Table 2 lists a complete comparison of IST and various versions of IRLS in terms of computational time. The standard IRLS is faster than standard IST. However, IRLS may become slower if the direct solution (10) is replaced by matrix splitting (14), especially when the matrix splitting has very slow convergence speed (e.g. $\lambda = 0.1$ and 1.0). For IRLS-MS, the computational time is very sensitive to the tolerance $e^{(j)}$. The shortest time is highlighted in Table 2. Comparing different tolerances $e^{(j)}$ of matrix splitting, we can see that despite of requiring much more steps to converge of IRLS, the looser tolerance $e^{(j)}$ (e.g. 10^{-1} and 10^{-5}) is more computationally efficient than strict one (e.g. 10^{-10} and 10^{-15}). It implies that the slow MS convergence cannot be compensated by fast convergence of IRLS, if $e^{(j)} \leq 10^{-10}$ or 10^{-15}, i.e. the MS convergence is more dominant than IRLS convergence in terms of computational time. We choose $e^{(j)} \leq 10^{-5}$ in the following experiments, since it leads to the fastest computation generally.

Table 2. Computational time of various algorithms (units: seconds) (tolerance: $e^{(i)} \leq 10^{-4}$).

Algorithms	IST	IRLS	IRLS-MS				IRLS-MS-SURE			
			1	2	3	4	1	2	3	4
$\lambda = 0.01$	0.05	0.12	**0.006**	**0.006**	3.38	164.40	0.04	**0.03**	310.20	12219.98
$\lambda = 0.1$	8.59	0.18	**2.17**	2.18	3.36	24.48	37.84	**35.36**	269.08	1989.05
$\lambda = 1.0$	2.95	0.15	1.38	1.43	**1.01**	3.51	24.50	**22.23**	72.47	299.91
$\lambda = 10$	0.47	0.11	0.25	**0.14**	0.22	0.47	4.33	**3.02**	14.55	34.52
$\lambda = 100$	0.11	0.07	0.06	**0.04**	0.09	0.15	**0.98**	1.13	4.21	8.16

However, notice that the convergence speed of IRLS algorithms also crucially depends on the initialization of W. The running time reported in Table 2 is based on $W^{(0)} = I$. We will propose another initialization of W to remarkably accelerate the algorithm in Section 4.4. In addition, comparing IRLS-MS and IRLS-MS-SURE, we can see that most time is consumed by computing Jacobian matrix and recursive SURE.

4.4. *Selection of parameter λ and the optimal solution of IRLS*

In this part, we are going to implement Algorithm 2, to select λ with minimum SURE. We empirically set the search range of λ as $[10^{-2}, 10^{4}]$, and pick up $K = 50$ logarithmically spaced values of λ. We implement IRLS-MS-SURE algorithm 50 times (for each λ_k), with the tolerances as $e^{(j)} \leq 10^{-5}$ and $e^{(i)} \leq 10^{-4}$. The result is shown in Figure 8(a): the optimal value of λ is 29.47. We can also see that the SURE is very close to the true prediction error.

We compare our result with IST and ridge regression. For standard IST, we update λ according to the discrepancy principle $\|y - Ax^{(i)}\|_2^2 = n\sigma^2$ (refer to [5] for more details). For ridge regression (11), it is easy to compute the SURE for each tentative λ_k by the closed form (12). Figure 8(b) shows that the optimal value of λ for ridge regression is 59.64.

Figures 9 and 10 show the estimations of x and μ by various algorithms. For better legibility and visual comparison, we only present the first three small fractions of the whole data: $1 \sim 50$, $50 \sim 100$ and $100 \sim 150$. Figure 10(d)–(f) shows the difference $\widehat{\mu} - \mu$ between estimate $\widehat{\mu}$ and true μ. We can see that the proposed IRLS-MS-SURE algorithm yields better estimates than others.

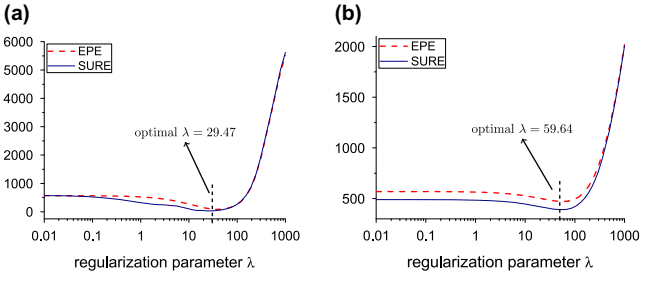

Figure 8. SURE and *oracle* EPE of $\widehat{\mu}_\lambda$ as the functions of parameter λ. (a) IRLS-MS-SURE algorithm. (b) Ridge regression.

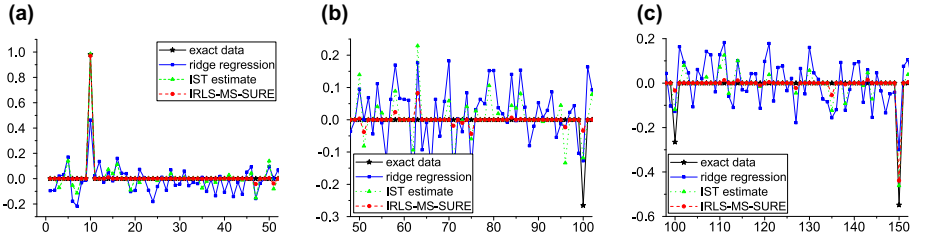

Figure 9. The estimations of x by various methods. (a) First fraction of x. (b) Second fraction of x. (c) Third fraction of x.

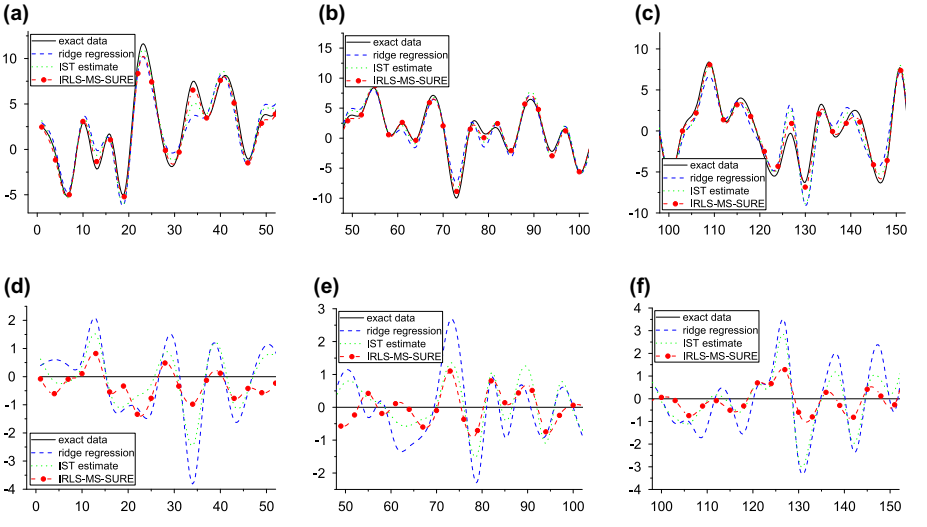

Figure 10. The estimations of μ by various methods. (a) First fraction of μ. (b) Second fraction of μ. (c) Third fraction of μ. (d) Difference of first fraction of μ. (e) Difference of second fraction of μ. (f) Difference of third fraction of μ.

Table 3. Estimation errors and total running time by various algorithms.

Algorithms	ridge	IST	IRLS-MS-SURE
e_x	1.84×10^{-2}	2.30×10^{-3}	7.17×10^{-4}
e_μ	1.57	1.38	0.30
Time (in seconds)	1.92	5.63	12.51

Table 3 reports the estimation errors and total running time of the algorithms. The errors are defined as $e_x = \frac{1}{p}\|\widehat{x} - x\|_2^2$ and $e_\mu = \frac{1}{n}\|\widehat{\mu} - \mu\|_2^2$, where $\widehat{x}$ and $\widehat{\mu}$ are the estimates of x and μ by these algorithms. The errors of IRLS are smaller than IST and ridge regression by one order of magnitude. The total computational time of IRLS and ridge includes $K = 50$ times of implementations for each λ_k to optimize λ. It is no surprise that ridge regression is the fastest due to its non-iterative nature, but worth noting that IRLS is slightly slower than IST, despite of 50 times of implementations of IRLS-MS-SURE. An important reason

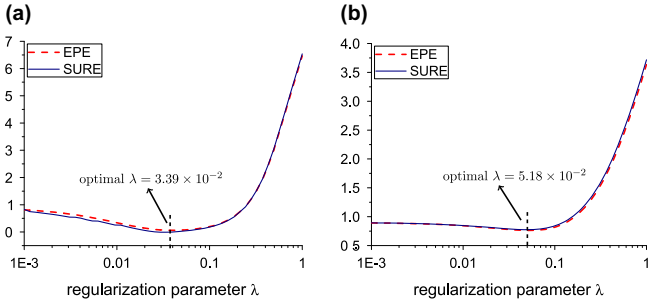

Figure 11. SURE and *oracle* EPE of $\widehat{\mu}_\lambda$ as the functions of parameter λ. (a) IRLS-MS-SURE algorithm. (b) Ridge regression.

for the fast speed is that for $(k+1)$th value λ_{k+1}, $W^{(0)}$ is initialized by the solution $\widehat{x}_k$ of the previous value λ_k: $W^{(0)}_{m,m} = 1/\left|(\widehat{x}_k)_m\right|$ for $m = 1, 2, \ldots, p$. This initialization would greatly accelerate the IRLS convergence, comparing to the initialization of $W^{(0)}$ as identity.

4.5. *Some real applications*

We have discussed the proposed IRLS-MS-SURE algorithm from many perspectives, taking random data for example. Now, we are going to show some real applications.

4.5.1. *Spline interpolation*

In this part, we consider spline interpolation. Suppose we have a signal $\mu(t)$ that can be expressed as a linear combination of the basis functions $\{\phi(t-k)\}_{k\in\mathbb{Z}}$: $\mu(t) = \sum_{k=0}^{p-1} x_k \phi(t-k)$, where $\{\phi(t-k), k \in \mathbb{Z}\}$ form a Riesz basis. Here, $\phi(t-k)$ is assumed as shifted cubic B-splines. We also assume that $\mu(t)$ is sparse in $\{\phi(t-k), k \in \mathbb{Z}\}$ basis, and hence the vector $x = [x_0, x_1, \ldots, x_{p-1}]^{\mathrm{T}} \in \mathbb{R}^p$ has few non-zero entries. We have measurements $\{y_m\}_{m=1}^n$ of the form $y_m = \mu_m + \epsilon_m$, where $\{\epsilon_m\}_{m=1}^n$ are i.i.d. realizations of a Gaussian distributed random variable with zero mean and variance σ^2, and $\mu_m = \sum_{k=0}^{p-1} x_k \phi(m-k)$ are the samples of $\mu(t)$. We set the data dimension as $n = p = 1000$, and there are only 10 non-zeros coefficients in vector x. We then add a large value of noise variance σ^2, such that the input SNR is 10 dB. The linear model (1) exactly suits the problem, when $A \in \mathbb{R}^{n\times p}$ is a matrix whose (m,k)th entry is $A_{m,k} = \phi(m-k)$. The sparsity is enhanced, as the regularization parameter λ in (2) increases. Now, we are to find the optimal λ, to achieve the minimum prediction error.

Figure 11 shows the SURE as a function of λ for IRLS and ridge regression. By the optimal values of λ, Figures 12 and 13 show the signal reconstruction by various algorithms. For better visual comparison, we only present a number of small segments of the high-dimensional data x and μ.

Table 4 presents the estimation errors and computational time of the algorithms. We can see the superior performance of the proposed algorithm.

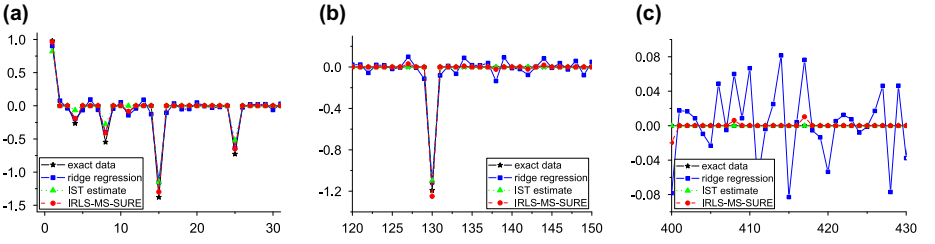

Figure 12. The estimations of x by various methods. (a) First fraction of x. (b) Second fraction of x. (c) Third fraction of x.

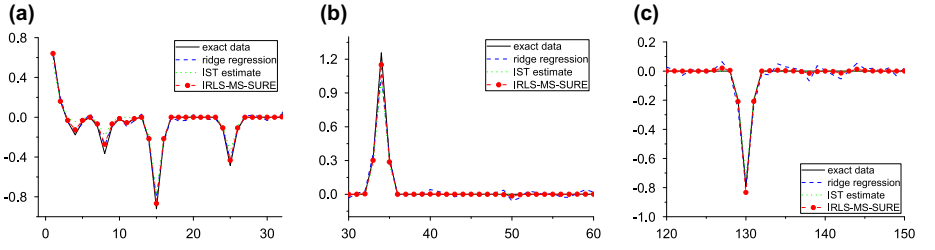

Figure 13. The estimations of μ by various methods. (a) First fraction of μ. (b) Second fraction of μ. (c) Third fraction of μ.

4.5.2. *Deconvolution of 1-D signal*

Let us now treat the linear model (1) as the convolutional observation. Suppose we have a sparse 1-D signal $x \in \mathbb{R}^p$ with very few non-zero elements, which is blurred by a Gaussian kernel a, and then corrupted by the Gaussian noise ϵ, s.t. the input SNR is 10 dB. Then, we have the distorted data $y = Ax + \epsilon$ where the convolution matrix A is constructed by the Gaussian kernel a, and becomes circulant under the periodic boundary condition being considered.

Table 4. Estimation errors and running time by various algorithms.

Algorithms	Ridge	IST	IRLS-MS-SURE
e_x	2.59×10^{-3}	4.59×10^{-4}	1.25×10^{-4}
e_μ	7.62×10^{-4}	2.26×10^{-4}	6.09×10^{-5}
Time (in seconds)	2.13	15.84	21.66

Figure 14 shows the SURE as a function of λ for IRLS and ridge regression. By the optimal values of λ, Figures 15 and 16 show the deconvolution results by various algorithms. For better visual comparison, we only present several small segments of the signals x and μ.

Table 5 presents the estimation errors and computational time of the algorithms. The proposed algorithm achieves higher estimation accuracy than others.

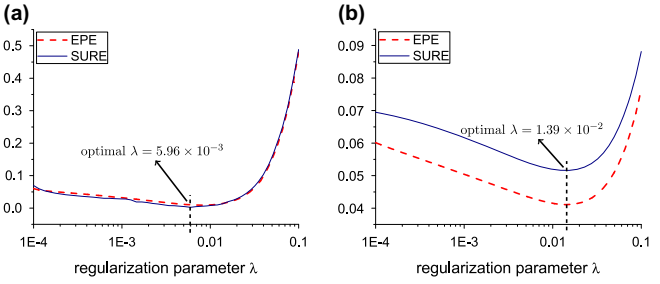

Figure 14. SURE and *oracle* EPE of $\widehat{\mu}_\lambda$ as the functions of parameter λ. (a) IRLS-MS-SURE algorithm. (b) Ridge regression.

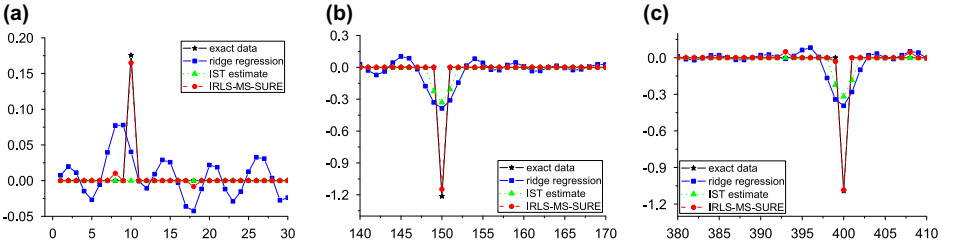

Figure 15. The estimations of x by various methods. (a) First fraction of x. (b) Second fraction of x. (c) Third fraction of x.

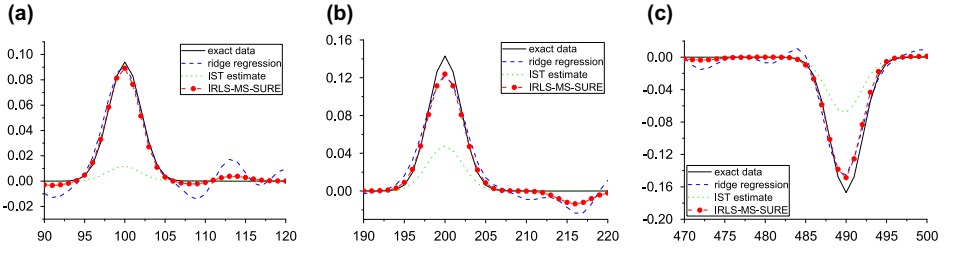

Figure 16. The estimations of μ by various methods. (a) First fraction of μ. (b) Second fraction of μ. (c) Third fraction of μ.

Table 5. Estimation errors and running time by various algorithms.

Algorithms	ridge	IST	IRLS-MS-SURE
e_x	1.40×10^{-2}	1.26×10^{-2}	9.55×10^{-4}
e_μ	8.22×10^{-5}	4.96×10^{-4}	1.95×10^{-5}
Time (in seconds)	2.15	20.11	18.12

5. Conclusion

SURE has been proven to be a powerful tool to select regularization parameter. In this paper, to solve ℓ_1-minimization problem, we proposed a recursive SURE for IRLS algorithms based on matrix splitting scheme. It enables us to monitor the prediction loss (actually, an unbiased estimator of prediction loss – SURE) during the iterations of IRLS, without referring to the true unknown data.

Numerical results showed that the selected regularization parameter by SURE minimization consistently yields optimal sparse recovery in terms of prediction loss. We would also like to emphasize that the philosophy underlying theoretical developments in this work can also be extended, in principle, to handle other regularizers and reconstruction algorithms, e.g. total-variation regularization and IST-family algorithms.

Acknowledgements

The authors would like to thank the anonymous reviewers for their insightful comments. We are also obliged to Hanjie PAN for many useful discussions on this article.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

The work was supported by the National Natural Science Foundation of China [grant number 61401013].

Notes

1. To avoid numerical instability, we use $\max\left\{|x_m^{(i)}|, 10^{-15}\right\}$ for the denominator in $W_{(i-1)}$ in the implementation, see [5,23,24] for the similar treatment.
2. Input signal-to-noise ratio (SNR) is defined as: $10\log_{10}\left(\frac{\|\mu\|_2^2}{\|y-\mu\|_2^2}\right) = 10\log_{10}\left(\frac{\|\mu\|_2^2}{N\sigma^2}\right)$ in dB.

ORCID

Feng Xue http://orcid.org/0000-0003-4671-591X

References

[1] Rao C, Toutenburg H, Shalabh H, et al. Linear models and generalizations: least squares and alternatives. 3rd ed. Berlin: Springer-Verlag; 2008.
[2] Rao C. Linear statistical inference and its applications. 2nd ed. Wiley series in probability and statistics. New York (NY): Wiley; 2001.
[3] Hastie T, Tibshirani R, Friedman J, et al. The elements of statistical learning. Vol. 2. New York (NY): Springer-Verlag; 2009.
[4] Tibshirani R. Regression shrinkage and selection via the lasso. J. R. Stat. Soc. Ser. B (Methodological). 1996;58:267–288.

[5] Pan H, Blu T. An iterative linear expansion of thresholds for ℓ_1-based image restoration. IEEE Trans. Image Process. 2013;22:3715–3728.
[6] Bühlmann P, Yu B. Boosting, model selection, Lasso and nonnegative Garrote. Research report No. 127; Jan 2005.
[7] Dossal C, Kachour M, Fadili J, et al. The degrees of freedom of penalized ℓ_1 minimization. Stat. Sinica. 2012;23:809–828.
[8] Wright J, Yang A, Ganesh A, et al. Robust face recognition via sparse representation. IEEE Trans. Pattern Anal. Mach. Intelligence. 2008;31:210–227.
[9] Mallat S. A wavelet tour of signal processing: the sparse way. 3rd ed. Burlington (MA): Academic Press; 2008.
[10] Beck A, Teboulle M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM J. Imaging Sci. 2009;2:183–202.
[11] Yin W, Osher S, Goldfarb D, et al. Bregman iterative algorithms for ℓ_1-minimization with applications to compressed sensing. SIAM J. Imaging Sci. 2008;1:143–168.
[12] Combettes P, Wajs V. Signal recovery by proximal forward-backward splitting. Multiscale Model. Simul. 2005;4:1168–1200.
[13] Candès EJ, Romberg JK, Tao T. Stable signal recovery from incomplete and inaccurate measurements. Commun. Pure Appl. Math. 2006;59:1207–1223.
[14] Donoho D. Compressed sensing. IEEE Trans. Inform. Theory. 2006;52:1289–1306.
[15] Candès EJ, Wakin MB. An introduction to compressive sampling. IEEE Signal Process. Mag. 2008;25:21–30.
[16] Boyd SP, Vandenberghe L. Convex optimization. Cambridge University Press; 2004.
[17] Efron B, Hastie T, Johnstone I, et al. Least angle regression. Ann. Stat. 2004;32:407–499.
[18] Daubechies I, Defrise M, Mol CD. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. Commun. Pure Appl. Math. 2004;57:1413–1457.
[19] Vonesch C, Unser M. A fast thresholded Landweber algorithm for wavelet-regularized multidimensional deconvolution. IEEE Trans. Image Process. 2008;17:539–549.
[20] Nesterov Y. Smooth minimization of non-smooth functions. Math. Program. 2005;103: 127–152.
[21] Bredies K, Lorenz DA. Linear convergence of iterative soft-thresholding. J. Fourier Anal. Appl. 2008;14:813–837.
[22] Chartrand R, Yin W. Iteratively reweighted algorithms for compressive sensing. In: IEEE International Conference on Acoustics, Speech and Signal Processing. Las Vegas (NV); 2008. p. 3869–3872.
[23] Candès E, Wakin M, Boyd S. Enhancing sparsity by reweighted ℓ_1 minimization. J. Fourier Anal. Appl. 2008;14:877–905.
[24] Daubechies I, DeVore R, Fornasier M, et al. Iteratively reweighted least squares minimization for sparse recovery. Commun. Pure Appl. Math. 2010;63:1–38.
[25] Morozov V. Methods for solving incorrectly posed problems. New York (NY): Springer-Verlag; 1984.
[26] Hansen PC. Analysis of discrete ill-posed problems by means of the L-curve. SIAM Rev. 1992;34:561–580.
[27] Golub G, Heath M, Wahba G. Generalized cross-validation as a method for choosing a good ridge parameter. Technometrics. 1979;21:215–223.
[28] Zou H. Some perspectives of sparse statistical modeling [PhD thesis]. Stanford (CA): Stanford University; 2005.
[29] Tibshirani RJ, Taylor J. Degrees of freedom in lasso problems. Ann. Stat. 2012;40:1198–1232.
[30] Zou H, Hastie T, Tibshirani R, et al. On the "degrees of freedom" of the lasso. Ann. Stat. 2007;35:2173–2192.
[31] Mallows C. Some comments on C_p. Technometrics. 1973;15:661–675.

[32] Stein CM. Estimation of the mean of a multivariate normal distribution. Ann. Stat. 1981;9:1135–1151.
[33] Xue F, Blu T. A novel SURE-based criterion for parametric PSF estimation. IEEE Trans. Image Process. 2015;24:595–607.
[34] Leung G, Barron A. Information theory and mixing least-squares regressions. IEEE Trans. Inform. Theory. 2006;52:3396–3410.
[35] Vaiter S, Deledalle C, Peyré G, et al. Local behavior of sparse analysis regularization: applications to risk estimation. Appl. Comput. Harmon. Anal. 2013;35:433–451.
[36] Blu T, Luisier F. The SURE-LET approach to image denoising. IEEE Trans. Image Process. 2007;16:2778–2786.
[37] Xue F, Luisier F, Blu T. Multi-Wiener SURE-LET deconvolution. IEEE Trans. Image Process. 2013;22:1954–1968.
[38] Eldar Y. Generalized SURE for exponential families: applications to regularization. IEEE Trans. Signal Process. 2009;57:471–481.
[39] Van De Ville D, Kocher M. SURE-based non-local means. IEEE Signal Process. Lett. 2009;16:973–976.
[40] Vonesch C, Ramani S, Unser M. Recursive risk estimation for non-linear image deconvolution with a wavelet-domain sparsity constraint. In: 15th IEEE International Conference on Image Processing. San Diego (CA); 2008. p. 665–668.
[41] Giryes R, Elad M, Eldar Y. The projected GSURE for automatic parameter tuning in iterative shrinkage methods. Appl. Comput. Harmon. Anal. 2010;30:407–422.
[42] Satish M, Shenoy A, Chandra S. Sparse signal reconstruction in shift-invariant spaces. In: Signal Processing with Adaptive Sparse Structured Representations, July 8–11, 2013. Lausanne: EPFL; 2013. p. 665–668.
[43] Ramani S, Blu T, Unser M. Monte-Carlo SURE: a black-box optimization of regularization parameters for general denoising algorithms. IEEE Trans. Image Process. 2008;17:1540–1554.
[44] Figueiredo M, Bioucas-Dias J, Nowak R. Majorization-minimization algorithms for wavelet-based image restoration. IEEE Trans. Image Process. 2007;16:2980–2991.
[45] Varga RS. Matrix iterative analysis. Vol. 27, Springer series in computational mathematics. Berlin: Springer; 2000.
[46] Woźnicki ZI. Matrix splitting principles. Int. J. Math. Math. Sci. 2001;28:251–284.