Received: 3 January 2019 | Revised: 24 July 2019 | Accepted: 27 October 2019
DOI: 10.1002/nla.2278

RESEARCH ARTICLE

Structured FISTA for image restoration

Zixuan Chen[1] | **James G. Nagy**[2] | **Yuanzhe Xi**[2] | **Bo Yu**[1]

[1]School of Mathematical Sciences, Dalian University of Technology, Dalian, China

[2]Department of Mathematics, Emory University, Atlanta, Georgia

Correspondence
James G. Nagy, Department of Mathematics, Emory University, Atlanta, GA 30322.
Email: jnagy@emory.edu

Funding information
Directorate for Mathematical and Physical Sciences; US National Science Foundation, Grant/Award Number: DMS-1819042

Summary

In this paper, we propose an efficient numerical scheme for solving some large-scale ill-posed linear inverse problems arising from image restoration. In order to accelerate the computation, two different hidden structures are exploited. First, the coefficient matrix is approximated as the sum of a small number of Kronecker products. This procedure not only introduces one more level of parallelism into the computation but also enables the usage of computationally intensive matrix–matrix multiplications in the subsequent optimization procedure. We then derive the corresponding Tikhonov regularized minimization model and extend the fast iterative shrinkage-thresholding algorithm (FISTA) to solve the resulting optimization problem. Because the matrices appearing in the Kronecker product approximation are all structured matrices (Toeplitz, Hankel, etc.), we can further exploit their fast matrix–vector multiplication algorithms at each iteration. The proposed algorithm is thus called *structured* FISTA (sFISTA). In particular, we show that the approximation error introduced by sFISTA is well under control and sFISTA can reach the same image restoration accuracy level as FISTA. Finally, both the theoretical complexity analysis and some numerical results are provided to demonstrate the efficiency of sFISTA.

KEYWORDS

image restoration, Kronecker product approximation, linear inverse problem, structured FISTA

1 | INTRODUCTION

Image restoration problems have a wide range of important applications, such as digital camera and video, microscopy, and medical imaging. Image restoration is the process of reconstructing an image of an unknown scene from an observed image, where the distortion can arise from many sources, such as motion blurs, out of focus lens, or atmospheric turbulence. Suppose there is an exact image of being all black except for a single bright pixel. If we take a picture of this image, then the distortion operation will cause the single bright pixel to be spread over its neighboring pixels. This single bright pixel is called a point source, and the function that describes the distortion and the resulting image of the point source is called the point spread function (PSF).[1] Mathematically, the distortion can be represented by a PSF. If the PSF is the same regardless of the location of the point source, it is called spatially invariant. Throughout this paper, we assume the PSF under consideration is always spatially invariant.

A spatially invariant image restoration problem can be modeled by a linear inverse problem of the following form:

$$b = Ax + e, \tag{1}$$

where $A \in \mathbb{R}^{N\times N}$ is a blurring matrix constructed from the PSF, $e \in \mathbb{R}^N$ is a vector representing additive noise, $b \in \mathbb{R}^N$ represents the distorted image, and $x \in \mathbb{R}^N$ denotes the unknown true image to be estimated. The matrix A is usually very ill-conditioned in these image restoration problems.

A classical way to solve (1) is by the least squares (LS) approach,[2] whose solution takes the following form:

$$\hat{x}_{\text{LS}} = \arg\min_x \frac{1}{2}\|Ax - b\|_2^2.$$

However, when A is ill-conditioned, the LS solution usually has a huge norm and is thus meaningless.[1] In order to compute a decent approximation to x, it is necessary to employ some form of regularization. The basic idea of regularization is to replace the original ill-conditioned problem with a "nearby" well-conditioned problem whose solution is close to the exact solution. Tikhonov regularization[3] is one of the most popular regularization techniques, where a quadratic penalty is added to the object function.

$$\hat{x}_{\text{TIK}} = \arg\min_x \frac{1}{2}\|Ax - b\|_2^2 + \frac{\lambda^2}{2}\|Rx\|_2^2$$

The second term in the above equation is a regularization term, which controls the norm (or seminorm) of the solution. The regularization parameter $\lambda > 0$ controls "smoothness" of the regularized solution. Typical choices of R include the identity matrix and a matrix approximating the first or second-order derivative operator.[4–6]

In this paper, we choose R as an identity matrix and consider the following minimization model:

$$\min_x \Phi(x) = \frac{1}{2}\|Ax - b\|_2^2 + \frac{\lambda^2}{2}\|x\|_2^2. \qquad (\mathcal{P})$$

In many applications, such as image restoration, it may also be important to include convex constraints (e.g., $x \geq 0$) on the solution.

Numerous algorithms proposed in the literature can be used to solve $(\mathcal{P})$ with convex constraints. One of them is the interior point method.[7,8] However, image restoration problems often involve dense matrix data, which will hamper the effectiveness of the interior point method. Another popular class of methods for solving $(\mathcal{P})$ are gradient-based algorithms.[9–11] Although these algorithms are relatively inexpensive at each iteration, they often suffer from slow convergence. One recent development is the fast iterative shrinkage-thresholding algorithm (FISTA),[12] which was proposed to solve nonsmooth convex optimization problems. FISTA preserves the computational simplicity and has a fast global convergence rate. Thus, FISTA becomes quite attractive for solving large-scale problems. Although problem $(\mathcal{P})$ does not involve any nonsmooth term, incorporating convex constraints is important in image deblurring applications. Moreover, in some situations l_1-based regularization has to be exploited to enforce sparsity in the solution. We plan to apply the proposed method to solve this class of nonsmooth optimization problems in the future. In this paper, we will first fully take advantage of the hidden structures of the blurring matrix A and improve the efficiency of the FISTA framework for solving the smooth optimization problem $(\mathcal{P})$.

Due to the 2D nature of the blurring model, the first structure to be exploited is the Kronecker product structure. Assume $K \in \mathbb{R}^{n\times n}$ and $H \in \mathbb{R}^{m\times m}$, the Kronecker product of these two matrices is defined as

$$K \otimes H = \begin{bmatrix} k_{11}H & \cdots & k_{1n}H \\ \vdots & & \vdots \\ k_{n1}H & \cdots & k_{nn}H \end{bmatrix}. \qquad (2)$$

For the blurring operator A in (1), it has been shown that A can be approximated by a matrix A_s as follows[13,14]:

$$A \approx A_s = \sum_{i=1}^{s} K_i \otimes H_i, \qquad (3)$$

where $K_i \in \mathbb{R}^{n\times n}$, $H_i \in \mathbb{R}^{m\times m}$ with $N = mn$. The error between the blurring matrix and the Kronecker product approximation can be easily controlled. In addition, these K_i and H_i are not general dense matrices but structured matrices (Toeplitz, Hankel, etc.[15–17]). We will give more details on the error between A and A_s and the structures of K_i and H_i in Section 2.

Consequently, the solution of (1) can be approximated by the problem

$$b = A_s x_s + e, \tag{4}$$

and equivalently, the solution of $(\mathcal{P})$ can be approximated by solving the optimization problem

$$\min_{x_s} \ \Phi_s(x) = \frac{1}{2}\|A_s x_s - b\|_2^2 + \frac{\lambda^2}{2}\|x_s\|_2^2. \tag{$\hat{\mathcal{P}}$}$$

From the numerical examples in Section 4, we can see that x_s from $(\hat{\mathcal{P}})$ and x from $(\mathcal{P})$ can provide indistinguishable image restoration results. This is because the original ill-posed problem $(\mathcal{P})$ only requires a numerical solution x with relatively low accuracy. As long as the difference between A_s and A falls below a certain level, which can be easily met with only a small value of s in (3), x_s from $(\hat{\mathcal{P}})$ and x from $(\mathcal{P})$ can reach the same level of accuracy. This phenomenon is analyzed in Theorem 4 in Section 3 and verified by the numerical experiments in Section 4.

If $b = \text{vec}(B), x_s = \text{vec}(X)$ and $e = \text{vec}(E)$, where vec(X) represents a column vector obtained from vectorizing a matrix X (i.e, columns of X are stacked one after the other), then (4) can be rewritten equivalently as

$$B = \sum_{i=1}^{s} H_i X K_i^T + E. \tag{5}$$

It is straightforward to derive the corresponding Tikhonov regularized minimization model as follows:

$$\min_X \frac{1}{2}\left\|\sum_{i=1}^{s} H_i X K_i^T - B\right\|_F^2 + \frac{\lambda^2}{2}\|X\|_F^2, \tag{$\tilde{\mathcal{P}}$}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. $(\tilde{\mathcal{P}})$ has several advantages over the original optimization problem $(\mathcal{P})$. First, $(\tilde{\mathcal{P}})$ benefits from the Kronecker product structure of A_s and can exploit more computationally intensive matrix–matrix operations. In addition, all the matrices H_i and K_i are structured matrices, which enables fast matrix–vector multiplications at each iteration. Second, the summation of s terms in $(\tilde{\mathcal{P}})$ can be performed independently and enables $(\tilde{\mathcal{P}})$ to reach superior parallel efficiency when implemented on modern high performance computing architectures. Some work has been done to exploit matrix equation structures for iterative methods to solve inverse problems of the form (5); see, for example, other works.[18–21] In this paper, we propose the structured FISTA (sFISTA) method. It gains its efficiency by exploiting both the Kronecker product structure of A and the structures from K_i and H_i. The convergence rate of sFISTA can be of the same order as FISTA under mild conditions.

The remaining sections are organized as follows. In Section 2, we describe how to approximate the blurring matrix A into the sum of a few of Kronecker products. In Section 3, we first briefly review the FISTA framework and then propose the sFISTA method. We also show that sFISTA for $(\tilde{\mathcal{P}})$ is equivalent to FISTA for $(\hat{\mathcal{P}})$ and derive the convergence and complexity analysis of sFISTA for $(\tilde{\mathcal{P}})$. Some numerical examples are provided in Section 4, and the concluding remarks are drawn in Section 5.

2 | KRONECKER DECOMPOSITION

Consider a 2D spatially invariant image restoration problem. It was shown in the work of Ng et al.[22] that three different structures of the blurring matrix A commonly occur. If the zero boundary conditions (corresponding to assuming the values of x outside the domain of consideration are zero) are applied, A will be a block-Toeplitz-Toeplitz-block (BTTB) matrix. On the other hand, if the periodic boundary conditions (corresponding to the case that the image outside the domain of consideration is a copy of the image inside in all directions) are used, A becomes a block-circulant-circulant-block (BCCB) matrix. Finally, A would be block-Toeplitz-plus-Hankel with Toeplitz-plus-Hankel-blocks (BTHTHB) if the reflective boundary conditions (corresponding to a reflection of the original scene at the boundary) are utilized. In any case, the matrix A can always be approximated as the sum of a few Kronecker products. Because the periodic boundary conditions often cause severe ringing artifacts near image borders, only the other two cases are considered in the remaining sections.

In practice, the PSF for images with $m \times n$ pixels is often stored as an $m \times n$ array P. When P represents the image of a single bright pixel, the process of taking a picture of such an image is equivalent to computing one column of matrix A with column index t, where t depends on the location of the point source. Thus, the structure of A is completely determined by that of P. More specifically, suppose P has the SVD decomposition $P = U\Sigma V^T$. Let u_i and v_i be the ith columns of the matrices U and V, respectively and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min(m,n)}$ be the singular values of P. It has been shown that A then admits the following Kronecker decomposition[13,14]:

$$A = \sum_{i=1}^{\min(m,n)} K_i \otimes H_i, \tag{6}$$

where K_i and H_i are matrices defined based on u_i, v_i, σ_i, and boundary conditions. More details on the structure of K_i and H_i will be provided at the end of this section. Because the singular values of P decay quickly in realistic applications, (6) can be further truncated by keeping only the first s terms

$$A \approx A_s = \sum_{i=1}^{s} K_i \otimes H_i. \tag{7}$$

The approximation error introduced in (7) has been well studied in other works.[13,14] The analysis in other works[13,14] shows that the distance between A and A_s is related to the approximation error of a truncated SVD decomposition of a matrix $\bar{P}$, which is summarized in the following theorem for the square PSF case.

Theorem 1 (See theorem 3.1 in the work of Nagy et al.[14]). *Assume the blurring matrix A is constructed from a PSF P with center p_{lq} located at (l,q), then for both zero boundary conditions and reflective boundary conditions, we have*

$$\left\| A - \sum_{i=1}^{s} K_i \otimes H_i \right\|_F = \left\| \bar{P} - \sum_{i=1}^{s} \sigma_i u_i {v_i}^T \right\|_F, \tag{8}$$

where $\bar{P} = W_a P W_b$ with $W_a = \text{diag}\left([\sqrt{n-l+1} \dots \sqrt{n-1}\sqrt{n}\sqrt{n-1} \dots \sqrt{l}]^T\right)$, $W_b = \text{diag}\Big([\sqrt{n-q+1} \dots \sqrt{n-1}$ $\sqrt{n}\sqrt{n-1} \dots \sqrt{q}]^T\Big)$ for the zero boundary conditions case and $\bar{P} = RPR^T$ with R is the Cholesky factor of the symmetric Toeplitz matrix with its first row as $[n, 1, 0, 1, 0, 1, \dots]$ for the reflective boundary conditions case. Here, $\sum_{i=1}^{s} \sigma_i u_i {v_i}^T$ is the summation of the first s terms in the SVD decomposition of $\bar{P}$.

Because the singular values of $\bar{P}$ (as well as P) decay quickly to zero for most PSFs, Theorem 1 guarantees that even a small s in (7) could lead to a very accurate approximation. Numerical experiments in Section 4 show that taking s as small as 5 is enough for the image restoration applications under consideration.

At the end of this section, let us take a look at the structure of K_i and H_i. If the zero boundary conditions are used, K_i and H_i have the following Toeplitz structure.

$$K_i = \text{toep}(k_i, l) \quad \text{and} \quad H_i = \text{toep}(h_i, q) \tag{9}$$

In the above equations, $k_i = (\sqrt{\sigma_i} u_i)./\text{diag}(W_a)$, $h_i = (\sqrt{\sigma_i} v_i)./\text{diag}(W_b)$, where ./ denotes point-wise division, and toep(c, j) denotes a banded Toeplitz matrix whose jth column is equal to c. For example,

$$\text{toep}(c,4) = \begin{bmatrix} c_4 & c_3 & c_2 & c_1 & 0 \\ c_5 & c_4 & c_3 & c_2 & c_1 \\ 0 & c_5 & c_4 & c_3 & c_2 \\ 0 & 0 & c_5 & c_4 & c_3 \\ 0 & 0 & 0 & c_5 & c_4 \end{bmatrix} \quad \text{with} \quad c = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix}. \tag{10}$$

On the other hand, if the reflective boundary conditions are applied, K_i and H_i are equal to the linear combinations of a Toeplitz matrix and a Hankel matrix:

$$K_i = \text{toep}(k_i, l) + \text{hank}(k_i, l) \quad \text{and} \quad H_i = \text{toep}(h_i, q) + \text{hank}(h_i, q), \tag{11}$$

where $k_i = \sqrt{\sigma_i} R^{-1} u_i$, $h_i = \sqrt{\sigma_i} R^{-1} v_i$ and hank(c,j) denotes a banded Hankel matrix whose first row and last column are defined by $[c_{j+1}, \dots, c_n, 0, \dots, 0]$ and $[0, \dots, 0, c_1, \dots, c_{j-1}]^T$, respectively. For example,

$$\text{hank}(c,3) = \begin{bmatrix} c_4 & c_5 & 0 & 0 & 0 \\ c_5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & c_1 \\ 0 & 0 & 0 & c_1 & c_2 \end{bmatrix} \quad \text{with} \quad c = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix}. \tag{12}$$

The sFISTA to be introduced in the next section will benefit from the fast Toeplitz/Hankel matrix–vector product algorithms when multiplying K_i and H_i with vectors at each iteration.

3 | STRUCTURED FISTA

In this section, we will first review the FISTA framework for solving $(\mathcal{P})$ and then propose sFISTA for solving $(\tilde{\mathcal{P}})$. We can prove that the proposed sFISTA for solving $(\tilde{\mathcal{P}})$ is equivalent to FISTA for solving $(\hat{\mathcal{P}})$. A detailed error analysis has also been conducted to show that the computational accuracy of sFISTA can reach the same level as that of FISTA under mild conditions. Finally, we compare the computational complexity of sFISTA for solving $(\tilde{\mathcal{P}})$ and FISTA for solving $(\mathcal{P})$ and show that sFISTA is more efficient in both serial and parallel computing environments.

3.1 | FISTA: a fast iterative shrinkage-thresholding algorithm

FISTA was first proposed in the work of Beck et al.[12] to solve the following general nonsmooth convex optimization model:

$$\min_x \{F(x) = f(x) + g(x)\}, \tag{13}$$

where $f : \mathbb{R}^N \to \mathbb{R}$ is a smooth convex function of the type $C^{1,1}$ and $g : \mathbb{R}^N \to \mathbb{R}$ is a continuous convex function, which is possibly nonsmooth. The basic idea of FISTA is that at each iteration, after getting the current iteration point x_k, an additional point y_{k+1} is chosen as the linear combination of the current iteration point x_k and the previous iteration point x_{k-1}. The next iteration point x_{k+1} is then set as the unique minimizer $p_{L_f}(y_{k+1})$ of the quadratic approximation $Q_{L_f}(x, y_{k+1})$ of $F(x)$ at y_{k+1} with

$$Q_{L_f}(x,y) := f(y) + \langle x - y, \nabla f(y) \rangle + \frac{L_f}{2} \|x - y\|_2^2 + g(x) \tag{14}$$

and L_f being the Lipschitz constant of ∇f. For more details about FISTA, see the work of Beck et al.[12]

Obviously, $(\mathcal{P})$ is a special instance of problem (13) if we let $f(x) = \frac{1}{2}\|Ax - b\|_2^2$ and $g(x) = \frac{\lambda^2}{2}\|x\|_2^2$. In this case, the (smallest) Lipschitz constant of the gradient ∇f is $L_f = \lambda_{\max}(A^T A)$. Simple calculations lead to

$$\begin{aligned} x_k = p_{L_f}(y_k) &= \arg\min_x \{Q_{L_f}(x, y_k) : x \in \mathbb{R}^{mn}\}. \\ &= \arg\min_x \left\{ \langle x, A^T(Ay_k - b)\rangle + \frac{L_f}{2}\|x\|_2^2 - L_f\langle x, y_k\rangle + \frac{\lambda^2}{2}\|x\|_2^2 \right\}, \\ &= \arg\min_x \left\{ \frac{L_f + \lambda^2}{2} \left\| x - \frac{1}{L_f + \lambda^2}\left(L_f y_k - A^T(Ay_k - b)\right) \right\|_2^2 \right\}, \\ &= \frac{1}{L_f + \lambda^2}\left(L_f y_k - A^T(Ay_k - b)\right). \end{aligned}$$

See Algorithm 1 for a description of FISTA for $(\mathcal{P})$.

```
Algorithm 1 FISTA for $(\mathcal{P})$
Initialization: Set initial point $y_1 = x_0 \in \mathbb{R}^N$, $L_f = \lambda_{\max}(A^TA)$, $k = 1$, $t_1 = 1$.
Step 1 Compute $x_k$ as follows
    $x_k = \frac{1}{L_f + \lambda^2}\left(L_f y_k - A^T(Ay_k - b)\right).$
Step 2 Compute $t_{k+1}$ as follows
    $t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2}.$
Step 3 Compute $y_{k+1}$ as follows
    $y_{k+1} = x_k + \frac{t_k - 1}{t_{k+1}}(x_k - x_{k-1}).$
Step 4 If a termination criterion is met, Stop; else, set $k := k + 1$ and go to Step 1.
```

As can be seen from Algorithm 1, the total computational cost of FISTA is dominated by matrix–vector multiplications associated with A and A^T at Step 1. Other steps only involve inexpensive vector and scalar operators. Despite its simplicity, FISTA enjoys a fast global convergence rate, which is summarized in Theorem 2.

Theorem 2 (See theorem 4.4 in the work of Beck et al.[12]). *Let $\{x_k\}$, $\{y_k\}$ be generated by FISTA. Then, for any $k \geq 1$,*

$$F(x_k) - F\left(x_F^*\right) \leq \frac{2L_f\left\|x_0 - x_F^*\right\|_2^2}{(k+1)^2},$$

where x_F^* is the solution of (13).

It is well known that first-order algorithms such as ISTAs,[23] generally only enjoy $O(1/k)$ convergence rate for solving $(\mathcal{P})$. From Theorem 2, we can see that FISTA is different from classical first-order methods in the sense that it preserves a fast global convergence rate $O(1/k^2)$. That is, in order to obtain a numerical solution x such that $F(x) - F(x_F^*) \leq \epsilon$, the number of iterations required by FISTA is at most $\frac{\sqrt{2L_f}\|x_0 - x_F^*\|_2}{\sqrt{\epsilon}} - 1$. In the next section, we will propose the sFISTA, which is more efficient for solving $(\tilde{\mathcal{P}})$.

3.2 | Accelerating FISTA by exploiting structures

In this section, we will show how to adapt the FISTA framework to solve $(\tilde{\mathcal{P}})$ by exploiting the two hidden structures. We first use a Kronecker product approximation A_s of the coefficient matrix A to introduce problem $(\hat{\mathcal{P}})$, which can be equivalently transformed into a matrix problem $(\tilde{\mathcal{P}})$. Consider the following quadratic approximation of the objective function of $(\tilde{\mathcal{P}})$ at a given point Y:

$$\begin{aligned} Q_L(X, Y) := &\frac{1}{2}\left\|\sum_{i=1}^{s} H_i Y K_i^T - B\right\|_F^2 + \left\langle X - Y, \sum_{j=1}^{s} H_j^T\left(\sum_{i=1}^{s} H_i Y K_i^T - B\right)K_j\right\rangle_F \\ &+ \frac{L}{2}\|X - Y\|_F^2 + \frac{\lambda^2}{2}\|X\|_F^2, \end{aligned} \tag{15}$$

where $L = \lambda_{\max}(A_s^T A_s)$ is the Lipschitz constant of the gradient of the first term in the object function of $(\tilde{\mathcal{P}})$. Similar to FISTA, we choose the unique minimizer of the quadratic approximation at point Y_{k+1}, which is the linear combination of X_k and X_{k-1}, as the new iteration point X_{k+1}. Mathematically, we set

$$Y_{k+1} = X_k + \frac{t_k - 1}{t_{k+1}}(X_k - X_{k-1}),$$

where t_k and t_{k+1} are parameters updated in the same way as FISTA to make sFISTA maintain the same convergence rate as FISTA for solving $(\tilde{\mathcal{P}})$ and compute X_{k+1} as

$$\begin{aligned}
X_{k+1} &= p_L(Y_{k+1}) = \arg\min_X \{Q_L(X, Y_{k+1}) : X \in \mathbb{R}^{m\times n}\}. \\
&= \arg\min_X \left\{ \left\langle X, \sum_{j=1}^{s} H_j^T \left(\sum_{i=1}^{s} H_i Y_{k+1} K_i^T - B \right) K_j \right\rangle_F + \frac{L}{2}\|X\|_F^2 + L\langle X, Y_{k+1}\rangle_F + \frac{\lambda^2}{2}\|X\|_F^2 \right\}, \\
&= \arg\min_X \left\{ \frac{L+\lambda^2}{2} \left\| X - \frac{1}{L+\lambda^2}\left(LY_{k+1} - \sum_{j=1}^{s} H_j^T \left(\sum_{i=1}^{s} H_i Y_{k+1} K_i^T - B \right) K_j \right) \right\|_F^2 \right\}, \\
&= \frac{1}{L+\lambda^2}\left(LY_{k+1} - \sum_{j=1}^{s} H_j^T \left(\sum_{i=1}^{s} H_i Y_{k+1} K_i^T - B \right) K_j \right).
\end{aligned}$$

Basic steps of sFISTA for $(\tilde{\mathcal{P}})$ are summarized in Algorithm 2.

Algorithm 2 sFISTA for $(\tilde{\mathcal{P}})$

```
Initialization: Compute a Kronecker product approximation $\sum_{i=1}^{s} K_i \otimes H_i$ of the coefficient matrix $A$.
Give initial point $Y_1 = X_0 \in \mathbb{R}^m \times \mathbb{R}^n$ and a Lipschitz constant $L$. Set $k = 1$, $t_1 = 1$, $b = \mathrm{vec}(B)$.
Step 1 Compute $X_k$ as follows
    $X_k = \frac{1}{L+\lambda^2}\left( LY_k - \sum_{j=1}^{s} H_j^T \left( \sum_{i=1}^{s} H_i Y_k K_i^T - B \right) K_j \right).$
Step 2 Compute $t_{k+1}$ as follows
    $t_{k+1} = \frac{1+\sqrt{1+4{t_k}^2}}{2}.$
Step 3 Compute $Y_{k+1}$ as follows
    $Y_{k+1} = X_k + \frac{t_k - 1}{t_{k+1}}(X_k - X_{k-1}).$
Step 4 If a termination criterion is met, Stop; else, set $k := k+1$ and go to Step 1.
Step 5 Return $x = \mathrm{vec}(X_k)$
```

Compared with Algorithm 1, there are several major differences between sFISTA and FISTA. First of all, the computational cost of Algorithm 1 is dominated by matrix–vector multiplications whereas Algorithm 2 can benefit from more computationally intensive matrix–matrix multiplications. Moreover, because H_i and K_i are all structured matrices (Toeplitz, Hankel, etc.), we can further exploit their fast matrix–vector multiplications at Step 1 in Algorithm 2. Second, Algorithm 2 decomposes the computation of X_k as the summation of s terms, which can be computed independently. Therefore, we can easily explore two levels of parallelism at each iteration in Algorithm 2. The first level corresponds to the structured matrix–vector multiplications with multiple vectors and the second level comes from the summation of s terms. This property enables Algorithm 2 to reach superior parallel performance when implemented on modern high performance architectures. Finally, we can prove that sFISTA for $(\tilde{\mathcal{P}})$ is equivalent to FISTA for $(\hat{\mathcal{P}})$, which guarantees the fast convergence.

Theorem 3. *sFISTA for $(\tilde{\mathcal{P}})$ and FISTA for $(\hat{\mathcal{P}})$ provide the same output as long as their initial points satisfy $x_0 = \mathrm{vec}(X_0)$. Mathematically, suppose $\{X_k\}$, $\{Y_k\}$ are generated by sFISTA for $(\tilde{\mathcal{P}})$ and $\{x_k\}$, $\{y_k\}$ are obtained by FISTA for $(\hat{\mathcal{P}})$, then we have $x_k = \mathrm{vec}(X_k)$ and $y_k = \mathrm{vec}(Y_k)$.*

Proof. To prove the desired results, we first review two important properties of Kronecker products, which will be used in the proof below.

$$(H \otimes K)\text{vec}(Z) = \text{vec}(KZH^T),$$
$$(H \otimes K)^T = H^T \otimes K^T,$$

where H, K, and Z are matrices of appropriate dimensions. Recall the frameworks of the two algorithms; to prove they provide the same output, we only have to show that both algorithms are equivalent at Step 1. Specifically, we just have to prove

$$\text{vec}\left(\sum_{j=1}^{s} H_j^T \left(\sum_{i=1}^{s} H_i Y_k K_i^T - B\right) K_j\right) = A_s^T \left(A_s y_k - b\right). \tag{16}$$

Utilizing the Kronecker product properties mentioned above, we can get

$$\begin{aligned}
&\text{vec}\left(\sum_{j=1}^{s} H_j^T \left(\sum_{i=1}^{s} H_i Y_k K_i^T - B\right) K_j\right), \\
&= \sum_{j=1}^{s} \left\{ \text{vec}\left(\sum_{i=1}^{s} H_j^T H_i Y_k K_i^T K_j\right) - \text{vec}\left(H_j^T B K_j\right) \right\}, \\
&= \sum_{j=1}^{s} \left\{ \sum_{i=1}^{s} \text{vec}\left(H_j^T H_i Y_k K_i^T K_j\right) - \text{vec}\left(H_j^T B K_j\right) \right\}, \\
&= \sum_{j=1}^{s} \left\{ \sum_{i=1}^{s} \left(\left(K_j^T K_i\right) \otimes \left(H_j^T H_i\right)\right) \text{vec}(Y_k) - \left(K_j^T \otimes H_j^T\right) \text{vec}(B) \right\}, \\
&= \sum_{j=1}^{s} \sum_{i=1}^{s} \left(K_j^T \otimes H_j^T\right) (K_i \otimes H_i) \text{vec}(Y_k) - \sum_{j=1}^{s} \left(K_j^T \otimes H_j^T\right) \text{vec}(B), \\
&= \left(\sum_{j=1}^{s} K_j^T \otimes H_j^T\right) \left(\sum_{i=1}^{s} K_i \otimes H_i\right) \text{vec}(Y_k) - \left(\sum_{j=1}^{s} K_j^T \otimes H_j^T\right) \text{vec}(B), \\
&= A_s^T \left(A_s y_k - b\right),
\end{aligned} \tag{17}$$

from which we can derive that $x_k = \text{vec}(X_k)$ and $y_k = \text{vec}(Y_k)$. □

It is worth pointing out that sFISTA for $(\tilde{\mathcal{P}})$ is only equivalent to FISTA for $(\hat{\mathcal{P}})$ due to the Kronecker product approximation error. The total computational error of sFISTA for solving $(\tilde{\mathcal{P}})$ comes from two places: the Kronecker product approximation to A and the iterative procedure of sFISTA. The following theorem analyzes the effect of these two kinds of errors on the accuracy of the final computed result.

Theorem 4. *Assume x^* and x_s^* are the exact solutions of $(\mathcal{P})$ and $(\hat{\mathcal{P}})$ respectively, $\{X_k, Y_k\}$ is the sequence obtained by sFISTA for $(\tilde{\mathcal{P}})$. Denote $\tilde{x}_k = \text{vec}(X_k)$. If the singular values of $A^T A_s + \lambda^2 I$ have a positive lower bound and the Kronecker product approximation A_s satisfies $\|A - A_s\|_F = \epsilon_s$, then we have for any $k \geq 1$,*

$$|\Phi(\tilde{x}_k) - \Phi(x^*)| \leq \frac{2L \left\|\tilde{x}_0 - x_s^*\right\|_2^2}{(k+1)^2} + c_0 \epsilon_s, \tag{18}$$

where c_0 is a positive constant independent of k.

Proof. Because x^* and x_s^* are the exact solutions of $(\mathcal{P})$ and $(\hat{\mathcal{P}})$ respectively, from their optimality conditions, we have

$$A^T(Ax^* - b) + \lambda^2 x^* = 0 \quad \text{and} \quad A_s^T \left(A_s x_s^* - b\right) + \lambda^2 x_s^* = 0, \tag{19}$$

which implies that

$$A^T A x^* = A^T b - \lambda^2 x^* \quad \text{and} \quad A_s^T A_s x_s^* = A_s^T b - \lambda^2 x_s^*. \tag{20}$$

It is easy to see

$$|\Phi(\tilde{x}_k) - \Phi(x^*)| \le \underbrace{|\Phi(\tilde{x}_k) - \Phi_s(\tilde{x}_k)|}_{I} + \underbrace{|\Phi_s(\tilde{x}_k) - \Phi_s(x_s^*)|}_{II} + \underbrace{|\Phi_s(x_s^*) - \Phi(x^*)|}_{III}. \tag{21}$$

For the first term, we have

$$\begin{aligned} |\Phi(\tilde{x}_k) - \Phi_s(\tilde{x}_k)| &= \left| \frac{1}{2}\|A\tilde{x}_k - b\|_2^2 - \frac{1}{2}\|A_s\tilde{x}_k - b\|_2^2 \right|, \\ &= \left| \frac{1}{2}\left(\tilde{x}_k^T A^T - b^T + \tilde{x}_k^T A_s^T - b^T\right)(A\tilde{x}_k - A_s\tilde{x}_k) \right|, \\ &\le \left\| \frac{1}{2}\tilde{x}_k^T A^T + \frac{1}{2}\tilde{x}_k^T A_s^T - b^T \right\|_2 \cdot \|\tilde{x}_k\|_2 \cdot \|A - A_s\|_2, \\ &\le c_1 \epsilon_s, \end{aligned} \tag{22}$$

where $c_1 = \frac{1}{2}\|\tilde{x}_k\|_2^2(\|A\|_2 + \|A_s\|_2) + \|b\|_2\|\tilde{x}_k\|_2$, and we use the fact that $\|A\|_2$, $\|A_s\|_2$, $\|\tilde{x}_k\|_2$ are bounded.

From Theorem 3, we know that sFISTA for $(\tilde{\mathcal{P}})$ is equivalent to FISTA for $(\hat{\mathcal{P}})$, which implies that the second term satisfies

$$|\Phi_s(\tilde{x}_k) - \Phi_s(x_s^*)| \le \frac{2L\left\|\tilde{x}_0 - x_s^*\right\|_2^2}{(k+1)^2}. \tag{23}$$

To estimate the last term, we first prove the following fact. From (19), we get

$$\begin{aligned} \lambda^2(x_s^* - x^*) &= A^T(Ax^* - b) - A_s^T(A_s x_s^* - b), \\ &= \left(A_s^T - A^T\right)b + A^T A x^* - A_s^T A_s x_s^*, \\ &= \left(A_s^T - A^T\right)b + \left(A^T A x^* - A^T A_s x^*\right) + \left(A^T A_s x^* - A^T A_s x_s^*\right) + \left(A^T A_s x_s^* - A_s^T A_s x_s^*\right), \\ &= \left(A_s^T - A^T\right)b + A^T(A - A_s)x^* + A^T A_s(x^* - x_s^*) + \left(A^T - A_s^T\right)A_s x_s^*, \end{aligned}$$

which implies that

$$\left(A^T A_s + \lambda^2 I\right)(x_s^* - x^*) = \left(A_s^T - A^T\right)b + A^T(A - A_s)x^* + \left(A^T - A_s^T\right)A_s x_s^*.$$

Then, we have

$$\begin{aligned} \|x_s^* - x^*\|_2 &= \left\|(A^T A_s + \lambda^2 I)^{-1} \cdot \left(\left(A_s^T - A^T\right)b + A^T(A - A_s)x^* + \left(A^T - A_s^T\right)A_s x_s^*\right)\right\|_2, \\ &\le \|(A^T A_s + \lambda^2 I)^{-1}\|_2 \cdot \left(\|b\|_2 + \|A\|_2\|x^*\|_2 + \|A_s\|_2\|x_s^*\|_2\right)\|A - A_s\|_2, \\ &\le \tilde{c}\epsilon_s, \end{aligned}$$

where $\tilde{c} = \|(A^T A_s + \lambda^2 I)^{-1}\|_2 \cdot (\|b\|_2 + \|A\|_2\|x^*\|_2 + \|A_s\|_2\|x_s^*\|_2)$ and we utilize Equation (20), the boundedness of $\|A\|_2$, $\|A_s\|_2$, $\|b\|_2$, $\|x^*\|_2$, $\|x_s^*\|_2$ and the assumption that the singular values of $A^T A_s + \lambda^2 I$ have a positive lower bound.

Then, for the last term, we have

$$\begin{aligned}
|\Phi_s(x_s^*) - \Phi(x^*)| &= \left| \frac{1}{2} \|A_s x_s^* - b\|_2^2 + \frac{\lambda^2}{2} \|x_s^*\|_2^2 - \frac{1}{2}\|Ax^* - b\|_2^2 - \frac{\lambda^2}{2}\|x^*\|_2^2 \right|, \\
&= \left| \frac{1}{2} x_s^{*^T} A_s^T A_s x_s^* - b^T A_s x_s^* + \frac{\lambda^2}{2} \|x_s^*\|_2^2 - \frac{1}{2} x^{*^T} A^T A x^* + b^T A x^* - \frac{\lambda^2}{2}\|x^*\|_2^2 \right|, \\
&= \left| \frac{1}{2} x_s^{*^T} \left(A_s^T b - \lambda^2 x_s^*\right) - b^T A_s x_s^* + \frac{\lambda^2}{2} \|x_s^*\|_2^2 - \frac{1}{2} x^{*^T} (A^T b - \lambda^2 x^*) + b^T A x^* - \frac{\lambda^2}{2}\|x^*\|_2^2 \right|, \\
&= \left| -\frac{1}{2} b^T A_s x_s^* + \frac{1}{2} b^T A x^* \right|, \\
&= \left| \frac{1}{2} b^T \left(A x^* - A_s x_s^*\right) \right|, \\
&= \left| \frac{1}{2} b^T \left(A x^* - A x_s^* + A x_s^* - A_s x_s^*\right) \right|, \\
&\le \frac{1}{2} \|b\|_2 \left(\|A\|_2 \|x^* - x_s^*\|_2 + \|A - A_s\|_2 \|x_s^*\|_2 \right), \\
&\le c_2 \epsilon_s,
\end{aligned}$$

where $c_2 = \frac{1}{2}\|b\|_2(\|A\|_2\tilde{c} + \|x_s^*\|_2)\epsilon_s$, and we utilize the boundedness of $\|A\|_2$, $\|b\|_2$, $\|x_s^*\|_2$ and the fact that $\|x_s^* - x^*\|_2 \le \tilde{c}\epsilon_s$.

Based on the analysis above for the three terms in (21), it follows that

$$|\Phi(\tilde{x}_k) - \Phi(x^*)| \le \frac{2L \left\|\tilde{x}_0 - x_s^*\right\|_2^2}{(k+1)^2} + (c_1 + c_2)\epsilon_s. \tag{24}$$

Let $c_0 = c_1 + c_2$, where c_1 and c_2 are defined above, then the desired result (18) follows. □

Theorem 4 shows that the error $|\Phi(\tilde{x}_k) - \Phi(x^*)|$ from sFISTA is bounded by two terms: $\frac{2L\|\tilde{x}_0 - x_s^*\|_2^2}{(k+1)^2}$ and $c_0\epsilon_s$. The first term decreases as the iteration proceeds whereas the second term remains constant during the iteration. In order to let the total error $|\Phi(\tilde{x}_k) - \Phi(x^*)|$ fall below a threshold ϵ, we need to make both terms smaller than ϵ. As discussed before, because only a relatively large ϵ is necessary in these ill-posed inverse problems, a small s would be enough to guarantee $c_0\epsilon_s < \epsilon$. In this sense, the convergence of sFISTA is dominated by the first term $\frac{2L\|\tilde{x}_0 - x_s^*\|_2^2}{(k+1)^2}$ and behaves in a similar way as FISTA.

As an example, we plot the singular values of the matrices P and $\bar{P}$ from the test image "hst" (see Example 1 in Section 4 for more details about this image) in Figure 1. It is easy to see that the singular values of both matrices decay quickly to zero. For example, the ratio of the sixth largest singular value of P to the largest one is only $6.47e-2$ and the

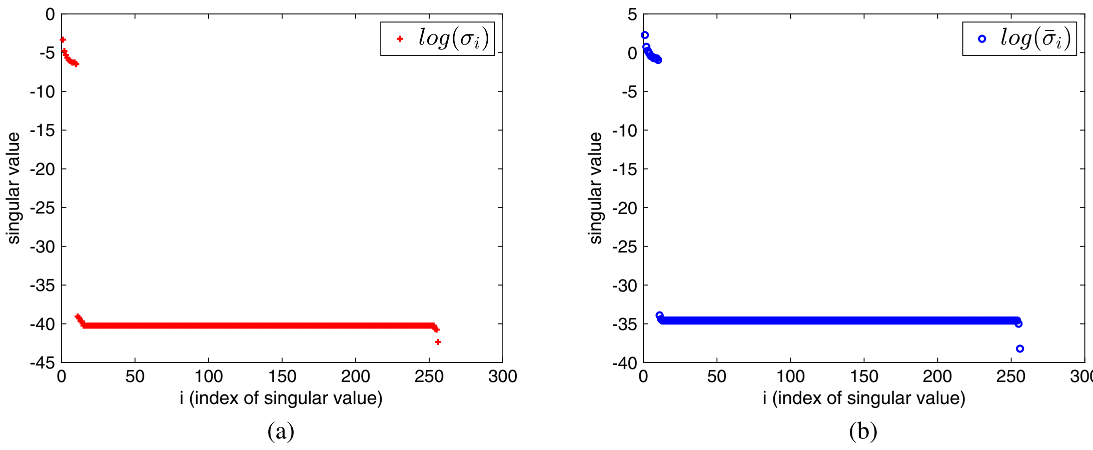

FIGURE 1 Singular values of P and $\bar{P}$ for test image "hst" in Example 1. (a) Singular values of P (b) Singular values of $\bar{P}$

ratio of the tenth largest singular value of P to the largest one reduces to $4.27e-2$. These patterns can also be observed in other test examples.

3.3 | Complexity analysis

In this section, we consider the computational complexity of sFISTA for $(\tilde{\mathcal{P}})$ (Algorithm 2) and FISTA for $(\mathcal{P})$ (Algorithm 1). If we ignore the structures in $A \in \mathcal{R}^{mn\times mn}$, $K_i \in \mathcal{R}^{n\times n}$, $H_i \in \mathcal{R}^{m\times m}$, and assume that they are all general dense matrices, then the cost of Step 1 in Algorithms 1 and 2 would be $O\left(m^2n^2\right)$ and $O\left(s(nm^2+mn^2)\right)$, respectively. When s is much smaller than m and n, which is the case for the applications under consideration in this paper, Algorithm 2 is definitely faster than Algorithm 1.

Recall that the blurring matrix A and matrices K_i and H_i from the Kronecker product approximation of A all have specific structures. As the matrix size becomes big enough, these structures will enable us to use fast Fourier transforms (FFTs) to accelerate matrix–vector multiplications encountered in both algorithms. For example, when zero boundary conditions are used, A is a BTTB matrix and K_i, H_i are Toeplitz matrices. In this case, the matrix–vector multiplication at Step 1 in Algorithm 1 can be performed in $O(mn\log(mn))$ with 2D FFTs, whereas Step 1 in Algorithm 2 can be done with 1D FFTs in $O(smn\log(mn))$. When reflective boundary conditions are utilized, A is a BTHTHB matrix and K_i, H_i can be represented as the sum of a Toeplitz matrix and a Hankel matrix. In this case, the computational complexities of Step 1 in both algorithms are still of the same order as in the zero boundary conditions case. Although Algorithm 2 has the same complexity as Algorithm 1, it is important to notice that Algorithm 2 is actually much more attractive when implemented on high performance architectures for a number of reasons. First of all, as discussed in the previous section, Algorithm 2 can easily exploit two levels of parallelism, which is crucial for fully taking advantage of the multilevel parallelism offered by the current architectures. Second, parallel 1D FFTs are known to scale better than parallel 2D FFTs. Thus, Algorithm 2 is more computationally efficient than Algorithm 1 for solving large-scale problems.

4 | NUMERICAL RESULTS

In this section, we provide some numerical examples to demonstrate the performance of sFISTA for solving $(\tilde{\mathcal{P}})$. All the algorithms were implemented with MATLAB and the experiments were performed on a MacBook Air with Intel Core i7 CPU (2.2 GHz). The following notations will be used throughout this section:

- s: the number of terms in the Kronecker product approximation;
- b: the data vector;
- noise: the vector of perturbations;
- b_n: the noisy data $b_n = b + \text{noise}$;
- NoiseLevel: relative level of noise defined as $\|\text{noise}\|_2/\|b\|_2$
- BlurLevel: an indicator used to set the severity of the blur to one of the following: "mild," "medium," and "severe";
- η: the relative error $\|x-x^*\|/\|x^*\|$;
- γ: the relative residual $\|r\|/\|b\|$, where $r = Ax - b$;
- *iter*: the iteration number of one algorithm;
- t(FISTA) and t(sFISTA): the CPU time (seconds) of FISTA and sFISTA, respectively;
- *tratio*: an indicator defined as $tratio = \frac{t(\text{sFISTA})}{t(\text{FISTA})}$ to compare the efficiency of FISTA and sFISTA.

Example 1. In this example, four 256×256 simple test images were extracted based on functions `PRblurdefocus` and `PRblurshake` from the regularization toolbox.[24] The four test images in this example are represented by "hst" (image of the Hubble space telescope), "satellite" (satellite test image), "pattern1" (geometrical image) and "ppower" (random image with patterns of nonzero pixels) respectively, which used reflective (Neumann) boundary conditions.[1] `PRblurdefocus` and `PRblurshake` are functions simulating a spatially invariant, out-of-focus blur, and spatially invariant motion blur caused by shaking of a camera, respectively. The BlurLevel was set to be "medium" in these four tests. In addition, function `PRnoise` was used to add Gaussian noise with NoiseLevel $= 0.01$ in this example. The regularization parameters were chosen automatically by `IRhybrid_lsqr` from the work of Gazzola et al.,[24] which is based on the hybrid bidiagonalization method presented in the work of Chung et al.[25] The Lipschitz constant was computed as an estimation of the two-norm of the matrix A, which was realized by a few iterations of Lanczos bidiagonalization as implemented in HyBR.[24]

We then tested FISTA for $(\mathcal{P})$ and sFISTA for $(\tilde{\mathcal{P}})$ on these four images. To show how the number of terms in the Kronecker product approximation affects the performance of sFISTA, s was set to range from 1 to 5 in these four tests. Both algorithms were stopped when either the relative residual γ satisfied $\gamma \leq \beta \cdot \text{NoiseLevel}$ or a maximum iteration number 50 was reached, where $\beta = 1.05$ is a "safety factor." To compare the performance of FISTA and sFISTA, we report the CPU time (seconds), the relative error η and the relative residual γ returned by both algorithms. Their values on these four tests are tabulated in Tables 1–4. To show how the FISTA and sFISTA iterations evolve, we plot η and γ versus iteration number for FISTA and sFISTA (with s = 1 and 5) in Figures 2, 3, 4, and 5.

TABLE 1 Numerical results for fast iterative shrinkage-thresholding algorithm (FISTA) and structured FISTA (sFISTA) for "hst"

	FISTA	sFISTA ($s = 1$)	sFISTA ($s = 2$)	sFISTA ($s = 3$)	sFISTA ($s = 4$)	sFISTA ($s = 5$)
Time	6.8225	0.5255	0.7784	1.0089	1.4913	1.4991
Iter	50	50	50	50	50	50
η	0.2184	0.2649	0.2329	0.2212	0.2186	0.2182
γ	0.0115	0.0180	0.0123	0.0116	0.0115	0.0115

TABLE 2 Numerical results for fast iterative shrinkage-thresholding algorithm (FISTA) and structured FISTA (sFISTA) for "satellite"

	FISTA	sFISTA ($s = 1$)	sFISTA ($s = 2$)	sFISTA ($s = 3$)	sFISTA ($s = 4$)	sFISTA ($s = 5$)
Time	6.8618	0.5593	0.7870	0.9975	1.2309	1.6005
Iter	50	50	50	50	50	50
η	0.2740	0.3551	0.2979	0.2783	0.2757	0.2744
γ	0.0129	0.0256	0.0152	0.0129	0.0130	0.0129

TABLE 3 Numerical results for fast iterative shrinkage-thresholding algorithm (FISTA) and structured FISTA (sFISTA) for "pattern1"

	FISTA	sFISTA ($s = 1$)	sFISTA ($s = 2$)	sFISTA ($s = 3$)	sFISTA ($s = 4$)	sFISTA ($s = 5$)
Time	7.5562	0.6701	0.8669	1.0097	1.2027	1.5385
Iter	50	50	50	50	50	50
η	0.0607	0.4781	0.2171	0.1417	0.0813	0.0689
γ	0.0087	0.0336	0.0137	0.0135	0.0108	0.0089

TABLE 4 Numerical results for fast iterative shrinkage-thresholding algorithm (FISTA) and structured FISTA (sFISTA) for "ppower"

	FISTA	sFISTA ($s = 1$)	sFISTA ($s = 2$)	sFISTA ($s = 3$)	sFISTA ($s = 4$)	sFISTA ($s = 5$)
Time	7.1086	0.5863	0.8070	1.0227	1.2676	1.4446
Iter	50	50	50	50	50	50
η	0.0968	0.2725	0.1478	0.1160	0.1097	0.0985
γ	0.0096	0.0226	0.0207	0.0139	0.0132	0.0096

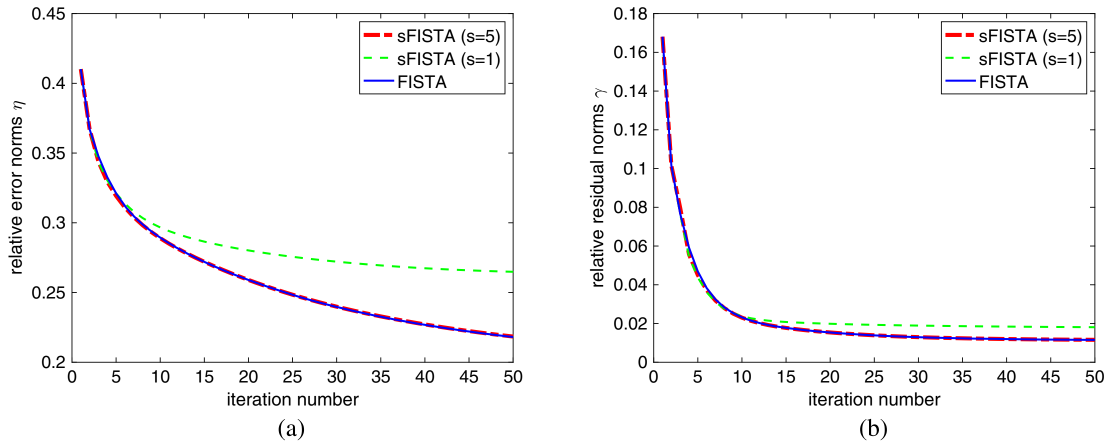

FIGURE 2 η and γ versus iteration number for "hst" extracted from PRblurdefocus. (a) Relative error norms η (b) Relative residual norms γ

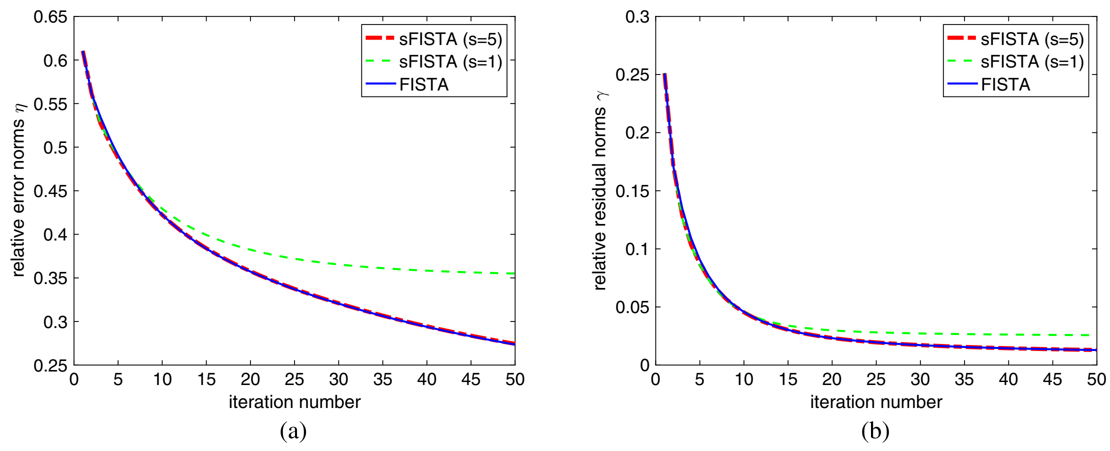

FIGURE 3 η and γ versus iteration number for "satellite" extracted from PRblurdefocus. (a) Relative error norms η (b) Relative residual norms γ

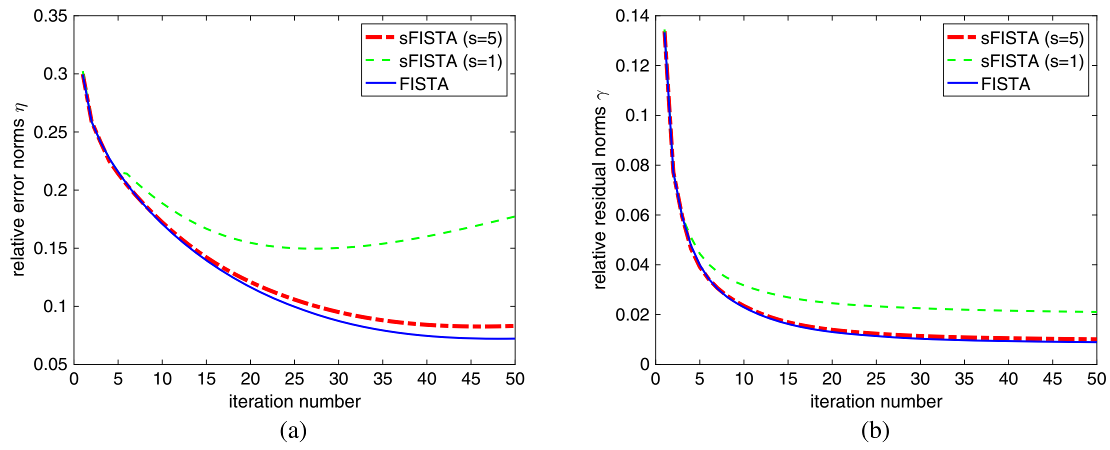

FIGURE 4 η and γ versus iteration number for "pattern1" extracted from PRblurshake. (a) Relative error norms η (b) Relative residual norms γ

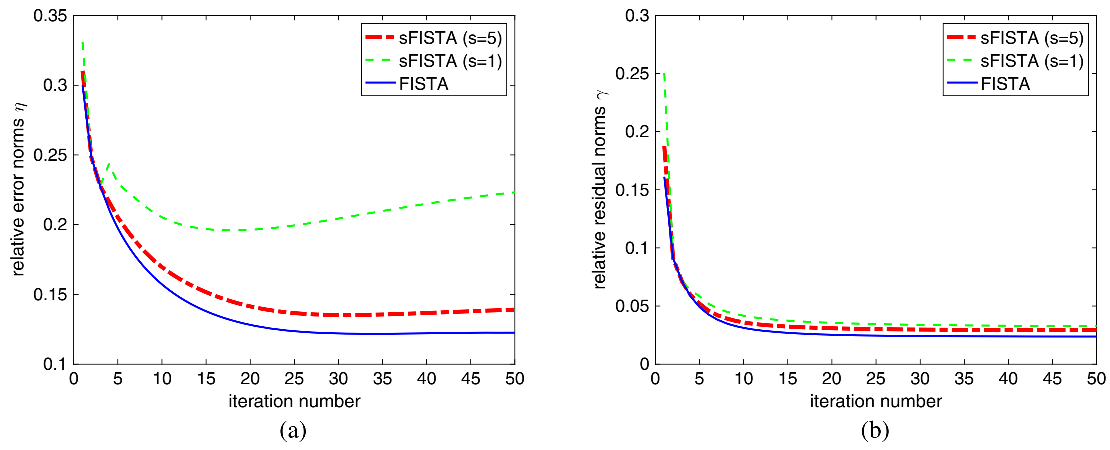

FIGURE 5 η and γ versus iteration number for "ppower" extracted from PRblurshake. (a) Relative error norms η (b) Relative residual norms γ

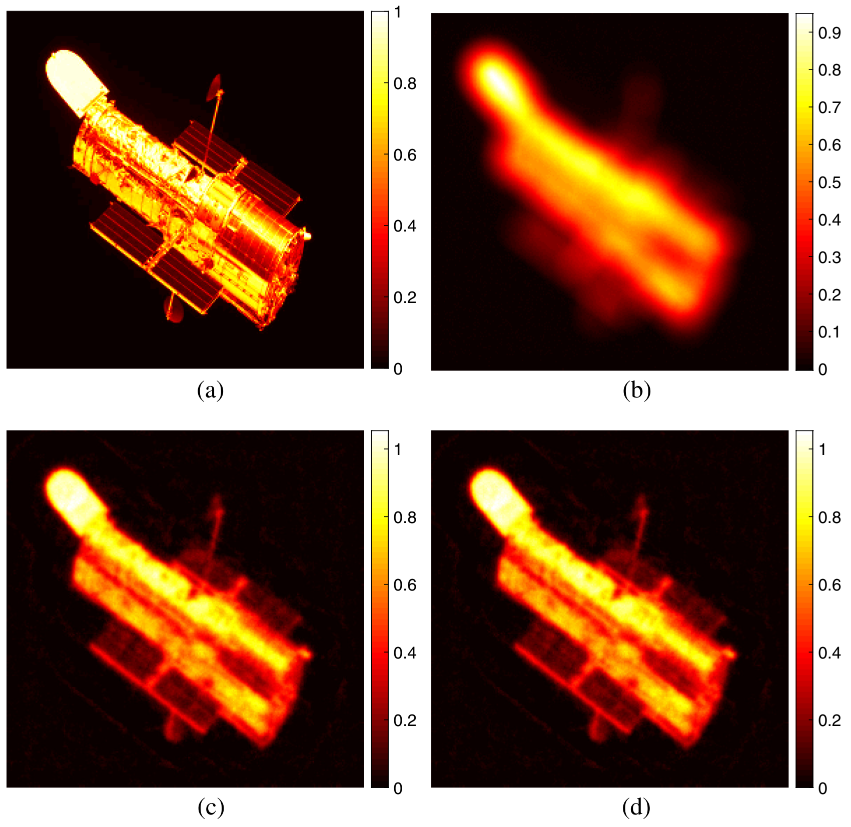

FIGURE 6 Figures for “hst” extracted from PRblurdefocus. (a) True image (b) Blurred and noisy image (c) Image obtained by fast iterative shrinkage-thresholding algorithm (FISTA) (d) Image obtained by structured FISTA (sFISTA; $s = 5$)

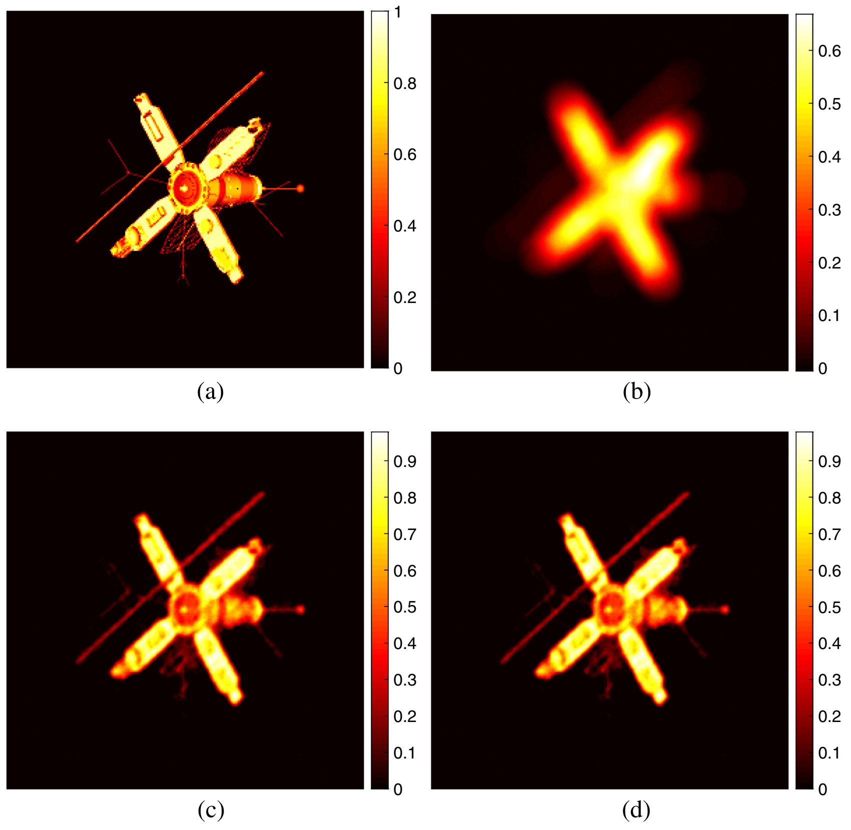

FIGURE 7 Figures for “satellite” extracted from PRblurdefocus. (a) True image (b) Blurred and noisy image (c) Image obtained by fast iterative shrinkage-thresholding algorithm (FISTA) (d) Image obtained by structured FISTA (sFISTA; $s = 5$)

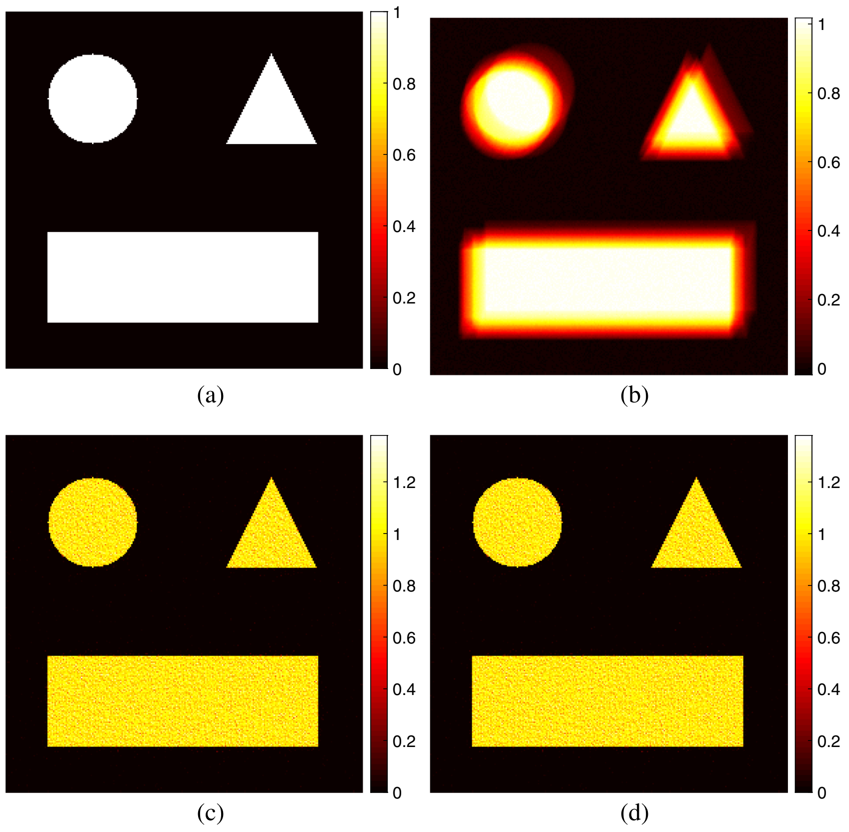

FIGURE 8 Figures for "pattern1" extracted from PRblurshake. (a) True image (b) Blurred and noisy image (c) Image obtained by fast iterative shrinkage-thresholding algorithm (FISTA) (d) Image obtained by structured FISTA (sFISTA; $s = 5$)

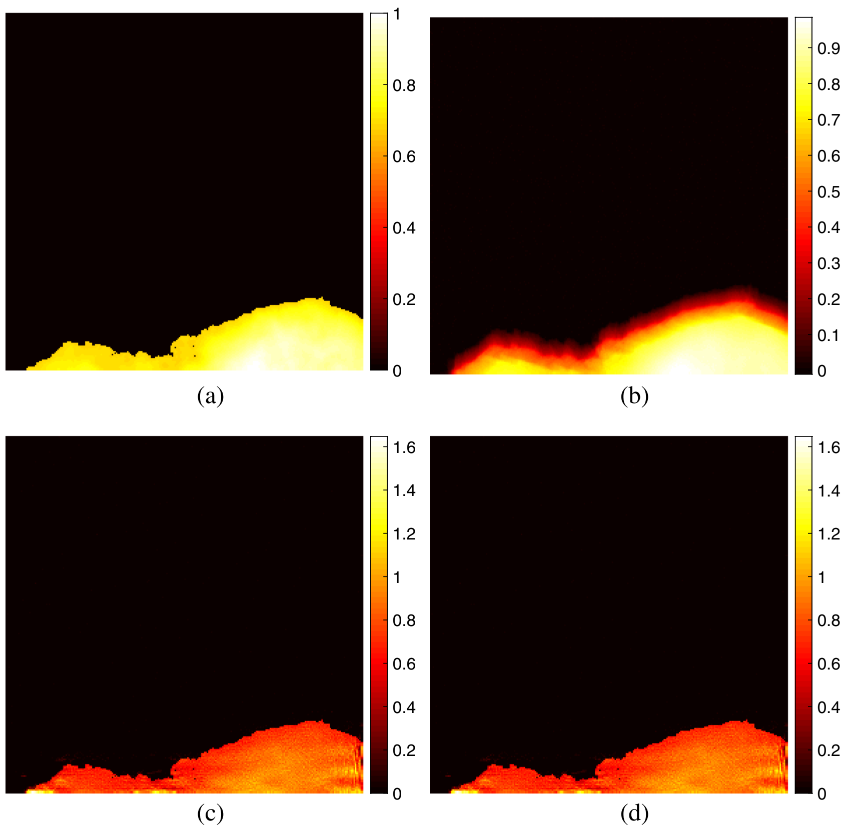

FIGURE 9 Figures for "ppower" extracted from PRblurshake. (a) True image (b) Blurred and noisy image (c) Image obtained by fast iterative shrinkage-thresholding algorithm (FISTA) (d) Image obtained by structured FISTA (sFISTA; $s = 5$)

As can be seen from Tables 1–4, sFISTA is much faster than FISTA in all test problems. As s increases from 1 to 5, the computational time of sFISTA increases monotonically whereas both the relative error η and the relative residual γ keep decreasing. When s reaches 5, the errors of sFISTA are close enough to those of FISTA and sFISTA is still about five times faster than FISTA. We would like to emphasize that sFISTA was only implemented as a serial code, and we expect to see a larger speedup with a parallel implementation in the future. As can be seen from Figures 2, 3, 4, and 5, η and γ for sFISTA with $s = 1$ are worse than those for sFISTA with $s = 5$ and FISTA. Those values even increase as the iteration proceeds for the last two test images. However, η and γ for sFISTA with $s = 5$ and FISTA are quite close during the iteration. This implies that 5 is a good choice for s.

We also plot the four images obtained by sFISTA in Figures 6, 7, 8, and 9. As a comparison, the true, blurred and noisy image and the image obtained by FISTA are also provided. It is easy to see that the images obtained by FISTA and sFISTA ($s = 5$) seem very similar to each other.

Example 2. In this set of test problems, we compare the performance of sFISTA and FISTA on eight images with different blur levels and noise levels. These problems are all extracted from functions `PRblurdefocus` and `PRblurshake`. The eight test images are represented by "pattern1" (geometrical image), "pattern2" (geometrical image), "ppower" (random image with patterns of nonzero pixels), "smooth" (very smooth image), "dot2" (two small Gaussian shaped dots), "dotk" ($N/2$ small Gaussian shaped dots), "satellite" (satellite test image), and "hst" (image of the Hubble space telescope), respectively. Each test image in this example undergoes blurring and Gaussian white noise-adding procedure with three different blurring levels: "mild," "medium," and "severe" and three different noise levels: 10^{-3}, 10^{-2}, and 10^{-1}. Because we have seen from Example 1 that $s = 5$ is a good choice for sFISTA, s is fixed to be 5 in this example.

We then tested sFISTA and FISTA on these eight images to compare their performance. The computational results are tabulated in Tables 5 and 6, respectively. In both tables, there are nine cases corresponding to the three different blurring levels and three different noise levels. Because the indicator *tratio* measures the ratio of the computational time of sFISTA to that of FISTA for solving the same problem to the same accuracy. Therefore, the smaller *tratio* is, the more efficient sFISTA is than FISTA.

The results in Tables 5 and 6 show that sFISTA is more efficient than FISTA in all test problems. For example, when we focus on one row or one column of Table 5 or Table 6, it is easy to see that all the values of *tratio* are less than 1.

TABLE 5 Values of *tratio* for `PRblurdefocus` with different blurring levels and noise levels

tratio	**case1**	**case2**	**case3**	**case4**	**case5**	**case6**	**case7**	**case8**	**case9**
BlurLevel	**"mild"**	**"mild"**	**"mild"**	**"medium"**	**"medium"**	**"medium"**	**"severe"**	**"severe"**	**"severe"**
NoiseLevel	10^{-3}	10^{-2}	10^{-1}	10^{-3}	10^{-2}	10^{-1}	10^{-3}	10^{-2}	10^{-1}
"pattern1"	0.2542	0.2654	0.2351	0.2534	0.2989	0.2937	0.2661	0.2874	0.2746
"pattern2"	0.2577	0.2493	0.2471	0.2515	0.2724	0.2953	0.2553	0.2607	0.2721
"ppower"	0.2449	0.2530	0.2601	0.2552	0.2753	0.2537	0.2601	0.2808	0.2531
"smooth"	0.2869	0.2562	0.2567	0.2554	0.2511	0.2735	0.2544	0.2913	0.2519
"dot2"	0.2559	0.2595	0.2545	0.2559	0.2545	0.2789	0.2911	0.2530	0.2905
"dotk"	0.2508	0.2769	0.2507	0.2620	0.2802	0.2560	0.2540	0.2618	0.2573
"satellite"	0.2643	0.2509	0.2560	0.2717	0.2708	0.2856	0.2539	0.2467	0.2944
"hst"	0.2501	0.2569	0.2618	0.2555	0.2599	0.2974	0.2542	0.2547	0.2821

TABLE 6 Values of *tratio* for `PRblurshake` with different blurring levels and noise levels

tratio	**case1**	**case2**	**case3**	**case4**	**case5**	**case6**	**case7**	**case8**	**case9**
BlurLevel	**"mild"**	**"mild"**	**"mild"**	**"medium"**	**"medium"**	**"medium"**	**"severe"**	**"severe"**	**"severe"**
NoiseLevel	10^{-3}	10^{-2}	10^{-1}	10^{-3}	10^{-2}	10^{-1}	10^{-3}	10^{-2}	10^{-1}
"pattern1"	0.2948	0.2723	0.2965	0.2859	0.2571	0.2885	0.2520	0.2758	0.2497
"pattern2"	0.2518	0.2842	0.2510	0.2528	0.2565	0.2559	0.2459	0.2478	0.2400
"ppower"	0.2828	0.2551	0.2593	0.2638	0.2539	0.2605	0.2520	0.2587	0.2873
"smooth"	0.2538	0.2487	0.2931	0.2522	0.2559	0.2476	0.2949	0.2650	0.2841
"dot2"	0.2496	0.2695	0.2590	0.2543	0.2536	0.2908	0.2374	0.2451	0.2329
"dotk"	0.2586	0.2596	0.2641	0.2522	0.2606	0.2564	0.2809	0.2568	0.2487
"satellite"	0.2411	0.2549	0.2600	0.2880	0.2542	0.2686	0.2493	0.2829	0.2803
"hst"	0.2541	0.2627	0.2501	0.2620	0.2721	0.2591	0.2744	0.2729	0.2714

Moreover, *tratio* in both tables are quite close to 0.25, which indicates that the efficiency of sFISTA does not depend on the blurring type, the test images, the blurring levels or the noise levels. These results further indicate that sFISTA is not only fast but also very robust for solving the image restoration problems considered in this paper.

5 | CONCLUSION

In this paper, we propose the structured FISTA (sFISTA) for solving large-scale ill-posed linear inverse problems arising from image restoration. By exploiting both the Kronecker product structure of the coefficient matrix and the pattern structure of the matrices in the Kronecker product approximation, sFISTA can significantly accelerate the computation compared to FISTA. A theoretical error analysis has been conducted to show that sFISTA can reach the same level of computational accuracy as FISTA under certain conditions. Finally, the efficiency of sFISTA is demonstrated with both a theoretical computational complexity analysis and various numerical examples coming from different applications.

The proposed sFISTA framework provides the possibility of developing new solvers for other imaging deblurring problems. For example, it is possible to adapt sFISTA to solve nonsmooth optimization problems with sparsity constraints such as those using l_1-based regularization. In additional, the structured matrix computations may also benefit other iterative regularization methods, such as Krylov methods and Krylov–Tikhonov methods.[26] We also plan to exploit preconditioning techniques to further reduce the iteration number and iteration time in our future work.

ACKNOWLEDGEMENT

This research was supported in part by the U.S. National Science Foundation under Grant DMS-1819042.

ORCID

James G. Nagy https://orcid.org/0000-0002-0451-3853
Yuanzhe Xi http://orcid.org/0000-0003-0361-0931

REFERENCES

1. Hansen PC, Nagy JG, O'Leary DP. Deblurring images. Philadelphia, PA: SIAM; 2006.
2. Björck A. Numerical methods for least squares problems. Philadelphia, PA: SIAM; 1996.
3. Tikhonov AN, Arsenin VY. Solution of ill-posed problems. Washington, DC: V.H. Winston; 1977.
4. Golub GH, Hansen PC, O'Leary DP. Tikhonov regularization and total least squares. SIAM J Matrix Anal Appl. 1999;21:185–194.
5. Hansen PC. Rank-deficient and discrete ill-posed problems. Philadelphia, PA: SIAM; 1997.
6. Hansen PC, O'Leary DP. The use of the L-curve in the regularization of discrete ill-posed problems. SIAM J Sci Comput. 1993;14:1487–1503.
7. Ben-Tal A, Nemirovski A. Lectures on modern convex optimization: Analysis algorithms, and engineering applications. Philadelphia, PA: SIAM; 2001.
8. Nocedal J, Wright S. Numerical optimization. New York, NY: Springer; 2006.
9. Beck A, Teboulle M. Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems. IEEE Trans Image Process. 2009;18:2419–2434.
10. Beck A, Teboulle M. Gradient-based algorithms with applications to signal-recovery problems. In: Palomar DP, Eldar YC, editors. Convex optimization in signal processing and communications. Cambridge, UK: Cambridge University Press, 2009; p. 42–88.
11. Zhang J, Hu Y, Nagy JG. A scaled gradient method for digital tomographic image reconstruction. Inverse Probl Imaging. 2018;18:239–259.
12. Beck A, Teboulle M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM J Imaging Sci. 2009;2:183–202.
13. Kamm J, Nagy JG. Optimal Kronecker product approximation of block Toeplitz matrices. SIAM J Matrix Anal Appl. 2000;22:155–172.
14. Nagy JG, Ng MK, Perrone L. Kronecker product approximations for image restoration with reflexive boundary conditions. SIAM J Matrix Anal Appl. 2004;25:829–841.
15. Xi Y, Xia J, Chan R. A fast randomized eigensolver with structured LDL factorization update. SIAM J Matrix Anal Appl. 2014;35:974–996.
16. Xi Y, Xia J, Cauley S, Balakrishnan V. Superfast and stable structured solvers for Toeplitz least squares via randomized sampling. SIAM J Matrix Anal Appl. 2014;35:44–72.
17. Xia J, Xi Y, Gu M. A superfast structured solver for Toeplitz linear systems via randomized sampling. SIAM J Matrix Anal Appl. 2012;33:837–858.
18. Bentbib A, Guide ME, Jbilou K. A generalized matrix Krylov subspace method for TV regularization. arXiv preprint arXiv:1802.03527. 2018.
19. Bouhamidi A, Jbilou K. Sylvester Tikhonov-regularization methods in image restoration. J Comput Appl Math. 2007;206:86–98.
20. Calvetti D, Reichel L. Application of ADI iterative methods to the image restoration of noisy images. SIAM J Matrix Anal Appl. 1996;17:165–174.

21. Zhang J, Nagy JG. An alternating direction method of multipliers for the solution of matrix equations arising in inverse problems. Numer Linear Algebra Appl. 2018. https://doi.org/10.1002/nla.2123
22. Ng MK, Chan RH, Tang W-C. A fast algorithm for deblurring models with Neumann boundary conditions. SIAM J Sci Comput. 1999;21:851–866.
23. Daubechies I, Defrise M, De Mol C. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. Commun Pure Appl Math. 2004;57:1413–1457.
24. Gazzola S, Hansen PC, Nagy JG. IR tools: a MATLAB package of iterative regularization methods and large-scale test problems. Numerical Algorithms. 2018. https://doi.org/10.1007/s11075-018-0570-7
25. Chung J, Nagy JG, O'Leary DP. A weighted-GCV method for Lanczos-hybrid regularization. Electron Trans Numer Anal. 2008;28:149–167.
26. Gazzola S, Novati P, Russo MR. On Krylov projection methods and Tikhonov regularization. Electron Trans Numer Anal. 2015;44:83–123.

How to cite this article: Chen Z, Nagy JG, Xi Y, Yu B. Structured FISTA for image restoration. *Numer Linear Algebra Appl.* 2020;27:e2278. https://doi.org/10.1002/nla.2278