A new trust region algorithm for image restoration

WEN Zaiwen[1] & WANG Yanfei[2]

1. State Key Laboratory of Scientific Engineering Computing, Institute of Computational Mathematics and Scientific/Engineering Computing, Academy of Mathematics and System Sciences, Chinese Academy of Sciences, Beijing 100080, China;
2. National Key Laboratory on Remote Sensing Science, Institute of Remote Sensing Applications, Chinese Academy of Sciences, Beijing 100101, China

Correspondence should be addressed to Wang Yanfei (email: yfwang@irsa.ac.cn)

Received June 13, 2003; revised April 5, 2004

Abstract The image restoration problems play an important role in remote sensing and astronomical image analysis. One common method for the recovery of a true image from corrupted or blurred image is the least squares error (LSE) method. But the LSE method is unstable in practical applications. A popular way to overcome instability is the Tikhonov regularization. However, difficulties will encounter when adjusting the so-called regularization parameter α. Moreover, how to truncate the iteration at appropriate steps is also challenging. In this paper we use the trust region method to deal with the image restoration problem, meanwhile, the trust region subproblem is solved by the truncated Lanczos method and the preconditioned truncated Lanczos method. We also develop a fast algorithm for evaluating the Kronecker matrix-vector product when the matrix is banded. The trust region method is very stable and robust, and it has the nice property of updating the trust region automatically. This releases us from tedious finding the regularization parameters and truncation levels. Some numerical tests on remotely sensed images are given to show that the trust region method is promising.

Keywords: trust region algorithm, image restoration, Lanczos method, Kronecker matrix-vector product, preconditioning.

DOI: 10.1360/03ys0178

1 Introduction

In remote sensing and astronomical imaging, the images obtained are usually corrupted or distorted by blurring due to atmospheric turbulence and noise. A forward model simulating such a process is usually in the form

$$(Kf)(x,y) := \int_a^b \int_c^d k(x-\xi, y-\eta) f(\xi,\eta) d\xi d\eta = h(x,y). \tag{1}$$

The problem of image restoration is the determination of the original object distribution f according to the recorded image h and knowledge about the PSF k (point spread function[1−3]). (1) is a convolution process, K represents the convolution operator. By operator theory, K maps a function f from space $D(f)$ to a function g into space $D(h)$. The image restoration problem is then to find the inverse transformation K^{-1} such that

$$K^{-1}h \longrightarrow f. \tag{2}$$

Essentially, the image restoration problem is an ill-posed problem. In other words, such problem concerns with the existence and uniqueness of an inverse transformation and particularly, the stability of the transformation[4]. The existence and uniqueness are usually easy to overcome, while the stability is the key problem deserves much more considering. Since in remote sensing and astronomical imaging, the function h is the observation, a trivial perturbation in h may produce significant perturbation in f. For example, there may exist δ_{out} which can be made arbitrarily small such that

$$K^{-1}(h+\delta_{\text{out}}) = f + \delta_{\text{in}}, \tag{3}$$

where $\delta_{\text{in}} \gg \delta_{\text{out}}$, but δ_{in} may not be arbitrarily small and hence cannot be neglected.

For digital image restoration, it is useful to develop a discrete-discrete model for image and object. The discrete-discrete model corresponding to (1) is a linear relation as the following matrix-vector equation:

$$\mathcal{K}\boldsymbol{f} = \boldsymbol{h}, \tag{4}$$

where $\boldsymbol{f}$ and $\boldsymbol{h}$ are lexicographically ordered vectors created from sampled object and image and $\mathcal{K}$ is the matrix resulting from the sampling point spread function. Because error is always involved in the experiment or sampling, noise cannot be ignored in the model above, so the matrix-vector equation can be expressed as

$$\mathcal{K}\boldsymbol{f} = \boldsymbol{h} + \boldsymbol{n}, \tag{5}$$

$\boldsymbol{n}$ is the noise which is also in lexicographically ordered vectors. To be simple, the matrix $\mathcal{K}$ is of order $N^2 \times N^2$, which is usually a tensor; $\boldsymbol{f}, \boldsymbol{h}, \boldsymbol{n}$ are vectors of order N^2. Since the ill-posed property of model (1), the problem (5) is ill-conditioned. It is very difficult to solve this linear equation by direct methods, such as the LU decomposition.

One way to solve (5) is the least square error (LSE) method, i.e. one minimizes the problem

$$J_1[\boldsymbol{f}] := \|\mathcal{K}\boldsymbol{f} - \boldsymbol{h}\|^2 \tag{6}$$

such that $\|\boldsymbol{n}\|$ goes to minimization. But we must note that the LSE is unstable. Because the minimization of J_1 is equivalent to

$$\mathcal{K}^T\mathcal{K}\boldsymbol{f} = \mathcal{K}^T\boldsymbol{h},$$

which is called the "normal equation". Since

$$\text{cond}(\mathcal{K}^T\mathcal{K}) \gg \text{cond}(\mathcal{K}),$$

so it is more ill-conditioned.

Tikhonov regularization is a popular way to overcome the instability of LSE, which can be considered as a penalized least square error (PLSE) problem

$$J_2[\boldsymbol{f}] := \|\mathcal{K}\boldsymbol{f} - \boldsymbol{h}\|^2 + \alpha\|\mathbf{f}\|_L^2, \tag{7}$$

where $\|f\|_L \overset{\text{def}}{=} \sqrt{(Lf,f)}$, L is the scale operator, for example, L can be chosen as a semi-definite or definite matrix. If we choose $L \equiv I$ (I is the identity matrix), then it leads to the standard Tikhonov regularization. α is the regularization parameter which is greater than zero. For the Tikhonov regularization, the choice of the parameter α is not an easy thing. It must make a trade-off between the noise and the regulation error. If α is chosen too large, then (7) is over-regularized; if α is chosen too small, (7) is still unstable.

Trust region method is another way to overcome the instability in solving the ill-conditioned problem (5). In refs. $[5, 6]$, the authors have developed trust region-cg method for such problem. This paper will develop a truncated Lanczos method in the trust region framework. We will see, for digital image restoration problem, this method is fast, robust and useful. This paper is organized in the following way: in section 2, the relationship between conjugate gradient method and Lanczos method is presented; in section 3, a truncated Lanczos method for trust region subproblem is introduced; in section 4, a preconditioned Lanczos algorithm is developed and a specific algorithm for the matrix-vector multiplication is designed; finally in section 5, some numerical simulations are given.

2 Relationship between conjugate gradient method and Lanczos method

The conjugate gradient method is one way of Krylov subspace method for solving the minimization problem

$$\min J_1[\boldsymbol{f}] = \frac{1}{2}\|\mathcal{K}\boldsymbol{f} - \boldsymbol{h}\|^2 = \frac{1}{2}\boldsymbol{f}^T\mathcal{K}^T\mathcal{K}\boldsymbol{f} - \boldsymbol{h}^T\mathcal{K}\boldsymbol{f} + \frac{1}{2}\boldsymbol{h}^T\boldsymbol{h}, \tag{8}$$

The general framework of conjugate gradient method reads as follows:

Algorithm 2.1. The conjugate gradient (CG) method

Step 0: Given $\boldsymbol{f}_0$; set $g_0 = grad(J_1[\boldsymbol{f}_0])$ and let $p_0 = -g_0$; set $j = 0$ and $\epsilon > 0$ (tolerance), perform the iteration:

Step 1: Calculate $a_j = (g_j, g_j)/(p_j, \mathcal{K}^T\mathcal{K}p_j)$;

Step 2: Set $\boldsymbol{f}_{j+1} = \boldsymbol{f}_j + a_j p_j$ and compute $g_{j+1} = g_j + a_j\mathcal{K}^T\mathcal{K}p_j$. If $\|g_{j+1}\| < \epsilon$, STOP;

Step 3: Calculate $\beta_j = (g_{j+1}, g_{j+1})/(g_j, g_j)$;

Step 4: Set $p_{j+1} = -g_{j+1} + \beta_j p_j,\ j := j + 1$, go to Step 1.

In Algorithm 2.1, $\boldsymbol{f}_0$ is the initial trial step, $grad(\cdot)$ denotes the gradient of a given functional, a_j is the step size in each iteration, p_j is the search direction in each iteration, β_j is the Fletcher-Reeves correction. This method generates a basis for $\mathbb{K}(\mathcal{K}^T\mathcal{K}, g_0, k)$ as

$$\mathbb{K}_k = \text{span}\{p_0, p_1, \cdots, p_k\} = \text{span}\{g_0, g_1, \cdots, g_k\}.$$

There are also other ways to generate the basis for the Krylov space $\mathbb{K}_k$, say, the Lanczos method (Algorithm 2.2). The algorithm reads as follows:

Algorithm 2.2. The Lanczos method

Step 0: Given $\boldsymbol{f}_0$; set $g_0 = grad(J_1[\boldsymbol{f}_0])$ and let $y_0 = g_0,\ q_{-1} = 0$; for $j = 0, 1, \cdots$, perform the following iteration:

Step 1: Calculate $\gamma_j = (y_j, y_j)$;

Step 2: Calculate $q_j = y_j/\gamma_j$;

Step 3: Calculate $\delta_j = (q_j, \mathcal{K}^T\mathcal{K}q_j)$;

Step 4: Set $y_{j+1} = \mathcal{K}^T\mathcal{K}q_j - \delta_j q_j - \gamma_j q_{j-1}$.

The Lanczos method is constructive, which is based on a particular decomposition of the matrix $\mathcal{K}^T\mathcal{K}$ to a tridiagonal form. This method generates an orthonormal basis $\{q_0, q_1, ..., q_k\}$ for the same Krylov space $\mathbb{K}_k$. If we define Q_k by the matrix $[q_0, ..., q_k]$, where each q_i is in the vector form, and take into account that $q_{-1} = 0$, then the Lanczos method can be written in a matrix form directly:

$$\mathcal{K}^T\mathcal{K}Q_k - Q_kT_k = \gamma_{k+1}q_{k+1}e_{k+1}^T, \tag{9}$$

$$Q_k^TQ_k = I_{k+1}, \tag{10}$$

where the tridiagonal matrix T_k is in the form

$$T_k = \begin{pmatrix} \delta_0 & \gamma_1 & & & \\ \gamma_1 & \delta_1 & \cdot & & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \delta_{k-1} & \gamma_k \\ & & & \gamma_k & \delta_k \end{pmatrix}. \tag{11}$$

Some useful properties can be deduced from Algorithm 2.2:

$$Q_k^T\mathcal{K}^T\mathcal{K}Q_k = T_k, \tag{12}$$

$$Q_k^Tg_0 = \gamma_0 e_1, \tag{13}$$

$$g_0 = y_0 = \gamma_0 q_0. \tag{14}$$

The equivalence between the two methods is that the elements of T_k can be expressed explicitly by the parameters generated by Algorithm 2.1 in the following fashion[11]:

$$\gamma_k = \sqrt{\beta_{k-1}}/\mid \alpha_{k-1} \mid, \quad \delta_k = \frac{1}{\alpha_k} + \frac{\beta_{k-1}}{\alpha_{k-1}}. \tag{15}$$

Therefore the Lanczos tridiagonal matrix T_k is available as a trivial byproduct of the conjugate gradient method.

3 Lanczos method for trust region subproblem

Trust region methods are a group of methods for ensuring global convergence while retaining fast local convergence in optimization algorithms[7,8]. In trust region method, one can solve the unconstrained minimization problem

$$\min J_1[\boldsymbol{f}] := \|\mathcal{K}\boldsymbol{f} - \boldsymbol{h}\|^2. \tag{16}$$

This problem can be approximated by a trust region subproblem (TRS):

$$\min_{s\in\mathcal{S}} \phi(s) = (grad(J_1[\boldsymbol{f}]), s) + \frac{1}{2}(Hess(J_1[\boldsymbol{f}])s, s), \tag{17}$$

$$\text{s.t. } \|s\| \leqslant \Delta, \tag{18}$$

where, $\mathcal{S}$ is some subspace in $\mathbb{R}^{N^2}$. The gradient and Hessian of the objective function $J_1[\boldsymbol{f}]$ can be computed explicitly as

$$grad(J_1[\boldsymbol{f}]) = \mathcal{K}^T\mathcal{K}\boldsymbol{f} - \mathcal{K}^T\boldsymbol{h}, \quad Hess(J_1[\boldsymbol{f}]) = \mathcal{K}^T\mathcal{K}. \tag{19}$$

Δ is the trust region radius which is an estimate of how far we trust the quadratic approximate model.

The key of trust region method is that the more reduction of the functional value $J_1[\boldsymbol{f}]$, the bigger increasing of the trust region radius Δ. By the minimization property, the functional values are always decreasing when solving the trust region subproblem during the iterations. Let s_k be a solution of (17) and (18) at the k-th iteration, we then define a ratio by the functional value

$$r_k = \frac{J_1[\boldsymbol{f}_k + s_k]}{J_1[\boldsymbol{f}_k]} \tag{20}$$

or by the norm of the gradient, i.e.

$$r_k = \frac{\|grad(J_1[\boldsymbol{f}_k + s_k])\|}{\|grad(J_1[\boldsymbol{f}_k])\|} \tag{21}$$

to decide whether the trial step s_k is acceptable or not and to adjust the new trust region radius.

Based on the standard trust region method and the analysis above, we outline a realistic algorithm as follows:

Algorithm 3.1. Trust region algorithm

Step 0: Initialization. Given the initial guess value f_0 and an initial trust-region radius Δ_0; choose parameters $\eta_1, \eta_2, \gamma_1, \gamma_2$ and γ_3 such that $0 < \eta_1 \leqslant \eta_2 < 1$ and $0 < \gamma_1 \leqslant 1 \leqslant \gamma_2 \leqslant \gamma_3$; compute $J_1[\boldsymbol{f}_0]$ and set $k := 0$.

Step 1: Step calculation. If the stopping rule is satisfied then STOP; else, solve (17) and (18) to get a step s_k that "sufficiently reduce the model".

Step 2: Acceptance of the trial point. Compute $J_1[\boldsymbol{f}_k + s_k]$ and compute r_k by (20) or (21), set

$$\boldsymbol{f}_{k+1} = \begin{cases} \boldsymbol{f}_k + s_k, & \text{if } r_k \leqslant \eta_1, \\ \boldsymbol{f}_k, & \text{otherwise.} \end{cases}$$

Step 3: Update of the trust-region radius. Set

$$\Delta_{k+1} \in \begin{cases} [\gamma_2\|s_k\|, \gamma_3\Delta_k], & \text{if } r_k \leqslant \eta_2, \\ [\gamma_1\Delta_k, \gamma_2\Delta_k], & \text{otherwise.} \end{cases}$$

Step 4: Evaluate $grad(J_1[\boldsymbol{f}_{k+1}])$; set $k := k + 1$; go to Step 1.

In Step 1, the stopping rule can be based on the values of $J_1[\boldsymbol{f}_k]$ or $grad(J_1[\boldsymbol{f}_k])$. In our numerical tests, the iteration is terminated if $J_1[\boldsymbol{f}_k] \leqslant \epsilon$ is satisfied, where ϵ is the tolerance which can be adjusted by users.

In refs. $[9,10]$, the authors consider the solution of the TRS by conjugate gradient method, we call it the Steihaug-Toint method. This method is very technical. If the trial step cross the boundary, it simply truncated the step at the boundary. That is, the Steihaug-Toint method is unconcerned with the trust region until it encounters its boundary and stops. Although this strategy is simple, it is somewhat practical. We can explore the property of the method more deeply. Utilizing the conjugate directions generated by CG, and noting the relationship between conjugate gradient method and Lanczos method, it is easy to reduce the system to a tridiagonal system. The cost of solving a tridiagonal system will far less than that of the original system. Thus our new approach is developed by integrating Lanczos method with Steihuag-Toint's CG method. When the trial step constrains is in the trust region, the iteration follows the Steihaug-Toint method; when the trial step blunders into its boundary, we solve the approximate tridiagonal system derived from the original system iteratively until the approximation satisfy the terminal condition.

Now, let

$$s \in \mathcal{S} = \left\{ s \in \mathbb{R}^{N^2} | s = Q_k u \right\}$$

and seek

$$s_k = Q_k u_k, \tag{22}$$

where s_k solves (17) and (18). It then follows directly from (12)—(14) that u_k solves the problem

$$\min_{u \in \mathbb{R}^{k+1}} \quad J_3[u] := (u, \gamma_0 e_1) + \frac{1}{2}(u, T_k u), \tag{23}$$

$$\text{s.t.} \quad \|u\| \leqslant \Delta. \tag{24}$$

The KKT conditions for a feasible point u_k to be a solution of problem (23) and (24) with corresponding Lagrangian parameter λ_k are

$$\begin{aligned} T_k(\lambda_k) := (T_k + \lambda_k I)u_k = -\gamma_0 e_1, \\ \lambda_k(\|u_k\| - \Delta) = 0, \\ \|u_k\| \leqslant \Delta, \ \lambda_k \geqslant 0. \end{aligned}$$

To be simple of notation, in the following, we denote the gradient of the function $J_1[\boldsymbol{f}]$ by g. The following theorem tells us how good the approximation is.

Theorem 3.1[11]**.**

$$(\mathcal{K}^T\mathcal{K} + \lambda_k I)s_k + g \ = \ \gamma_{k+1}(e_{k+1}, u_k)q_{k+1}, \tag{25}$$

$$\|(\mathcal{K}^T\mathcal{K} + \lambda_k I)s_k + g\| \ = \ \gamma_{k+1}\|(e_{k+1}, u_k)\|. \tag{26}$$

Proof. We have

$$\begin{aligned} \mathcal{K}^T\mathcal{K}s_k &= \mathcal{K}^T\mathcal{K}Q_k u_k \\ &= Q_k T_k u_k + \gamma_{k+1}(e_{k+1}, u_k)q_{k+1} \qquad \text{from } (9) \\ &= -Q_k(\lambda_k u_k + \gamma_0 e_1) + \gamma_{k+1}(e_{k+1}, u_k)q_{k+1} \\ &= -\lambda_k Q_k u_k - \gamma_0 Q_k e_1 + \gamma_{k+1}(e_{k+1}, u_k)q_{k+1} \end{aligned}$$

$$\begin{aligned} &= -\lambda_k s_k - \gamma_0 q_0 + \gamma_{k+1}(e_{k+1}, u_k) q_{k+1} \\ &= -\lambda_k s_k - g + \gamma_{k+1}(e_{k+1}, u_k) q_{k+1}. \quad \text{from (14)} \end{aligned}$$

This then directly gives (25), and (26) follows from the orthonormality of q_{k+1}. Q.E.D.

Therefore once we know γ_{k+1} and the last component of u_k, we can measure the error directly. This is convenient as we even do not need to know s_k or Q_k at all. Now, we outline our algorithm as follows:

Algorithm 3.2. The Lanczos method for TRS (TRLan)

Step 0: Let $s_0 = 0,\ g_0 = grad(J_1[\boldsymbol{f}]),\ \gamma_0 = \sqrt{(g_0, g_0)}$, and $p_0 = -g_0$; set ϵ(tolerance); set the flag INTERIOR as true and set $k := 0$.

Step 1: Set $\alpha_k = (g_k, g_k)/(p_k, \mathrm{Hess}(J_1[\boldsymbol{f}])p_k)$, obtain T_k from T_{k-1} using (11).

Step 2: If INTERIOR is true, but $\alpha_k < 0$ or $\|s_k + \alpha_k p_k\| \geqslant \Delta$, reset INTERIOR to false.

Step 3: If INTERIOR is true, set $s_{k+1} = s_k + \alpha_k p_k$; otherwise, solve the tridiagonal trust-region subproblem (23) and (24) to obtain u_k.

Step 4: Set $g_{k+1} = g_k + \alpha_k \mathrm{Hess}(J_1[\boldsymbol{f}])p_k$.

Step 5: If INTERIOR is true and $\|g_{k+1}\| \leqslant \epsilon$, STOP. If INTERIOR is false and $\gamma_{k+1}|(e_{k+1}, u_k)| \leqslant \epsilon$, go to Step 7.

Step 6: Set $\beta_k = (g_{k+1}, g_{k+1})/(g_k, g_k)$ and $p_{k+1} = -g_{k+1} + \beta_k p_k$, $k := k+1$, go to Step 1.

Stem 7: Recover $s_k = Q_k u_k$ by recurrences or obtaining Q_k from backing store, STOP.

In the algorithm above, we need to recover s_k from u_k by Q_k, so the Lanczos vectors will either need to be saved on backing store or regenerated. For example, economies can be made by saving the values of α_i and β_i during the first pass, and reusing them during the second.

Now another question arises: how to solve the tridiagonal system efficiently? After carefully investigating of the system, we find that Newton's method seems appropriate here. Since the Hessian is tridiagonal, it is not very expensive to factorize, and so does computing its leftmost eigenvalue. Another reason is that the hard case will not occur[11], although the so called "almost" hard case may still happen (But this will not affect us much thanks to the specialties of the application). So the following algorithm will be adopted:

Algorithm 3.3. Find model minimizer when Eigen solution is cheap[7]

Step 0: Let $\kappa_{\text{easy}} \in (0, 1)$.

Step 1: If T_k is positive definite, set $\lambda = 0$; otherwise, compute its leftmost eigenvalue λ_1, and set $\lambda = -\lambda_1^+$.

Step 2: Factorize $T_k(\lambda) = LL^T$, and solve $LL^T u = -g$.

Step 3: If $\|u\|_2 \leqslant \Delta$

Step 3a: If $\lambda = 0$ or $\|u\|_2 = \Delta$, STOP;

Step 3b: otherwise, compute an eigenvector v_1 corresponding to λ_1, find the root "α" of the equation $\|u+\alpha v_1\|_2 = \Delta$ which makes the model $J_3(u+\alpha v_1)$ be smallest, replace u by $u + \alpha v_1$, and STOP.

Step 4: If

$$||u\| - \Delta| \leqslant \kappa_{\text{easy}}\, \Delta,$$

STOP.

Step 5: Solve $Lw = u$ and replace λ by

$$\lambda + \left(\frac{\|u\| - \Delta}{\Delta}\right)\left(\frac{\|u\|_2^2}{\|w\|_2^2}\right).$$

Step 6: Factorize $T_k(\lambda) = LL^T$, solve $LL^T u = -g$ and go to Step 4.

In Step 1, λ^+ is barely smaller than λ_1. In practice, as long as $|\lambda_1^+ - \lambda_1|$ is small, the solution is acceptable.

4 Some numerical extensions

In many applications, the blurring process is assumed to be spatially invariant, that is, the blur is independent of position and a blurred object will look the same regardless of its position in the image. Thus the PSF is represented by the image of a single point source. In this case, the structure of the discrete operator $\mathcal{K}$ will be highly structured such as BCCB, BTTB and so on. Those structures have dominative importance in order to develop high performance algorithm.

4.1 Preconditioned Lanczos method

Preconditioning is often used with conjugate gradient methods, to accelerate the rate of convergence, that is, to reduce the number of iterations which are needed to compute a good approximation of the solution. Recalling that the relationships (12)—(15) also hold for preconditioned conjugate gradient method and preconditioned Lanczos method, therefore, we apply the preconditioning technique to Lanczos method.

We outline the preconditioned Lanczos method for trust region subproblem as follows:

Algorithm 4.1. The preconditioned Lanczos method for TRS (PTRLan)

Step 0: Let $s_0 = 0$, $g_0 = grad(J_1[\boldsymbol{f}])$, $v_0 = M^{-1}g_0$, $\gamma_0 = \sqrt{(v_0, g_0)}$ and $p_0 = -v_0$. Set ϵ(tolerance), set the flag INTERIOR as true and set $k := 0$.

Step 1: Set $\alpha_k = (g_k, v_k)/(p_k, Hess(J_1[\boldsymbol{f}])p_k)$, obtain T_k from T_{k-1}.

Step 2: If INTERIOR is true, but $\alpha_k < 0$ or $\|s_k + \alpha_k p_k\|_M \geqslant \Delta$, reset INTERIOR to false.

Step 3: If INTERIOR is true, set $s_{k+1} = s_k + \alpha_k p_k$; otherwise, solve the tridiagonal trust-region subproblem (23) and (24) to obtain u_k.

Step 4: Set $g_{k+1} = g_k + \alpha_k Hess(J_1[\boldsymbol{f}])p_k$.

Step 5: If INTERIOR is true and $\|g_{k+1}\|_{M^{-1}} \leqslant \epsilon$, STOP. If INTERIOR is false and $\gamma_{k+1}|(e_{k+1}, u_k)| \leqslant \epsilon$, go to Step 7.

Step 6: Set $\beta_k = (g_{k+1}, v_{k+1})/(g_k, g_k)$ and $p_{k+1} = -v_{k+1} + \beta_k p_k$, $k := k+1$, go to Step 1.

Step 7: Recover $s_k = Q_k u_k$ by recurrences or obtaining Q_k from backing store, STOP.

The choice of an appropriate preconditioner in PTRLan depends on the trade-off between the gain in the convergence rate and the increased cost that results from applying the preconditioner. As we see, the matrices that arise in image restoration are highly structured, involving Circulant and Toeplitz matrices and so on. Preconditioning such matrices has been thoroughly investigated in the literature, see the survey paper[12] and the references therein. Moreover, many of these approaches have been applied to image restoration.

Although a variety of approaches to preconditioning have been proposed, the simplest preconditioner is the diagonal preconditioner which can be computed easily, and that is the approach we consider in this paper. But it should be noted that, in principle, many of the other fast transforms based on preconditioners can be used as well.

4.2 Matrix-vector multiplication

For iterative methods, such as conjugate gradient method and Lanczos method, matrix-vector multiplication is the bottle-neck of computation. But utilizing the special structure of $\mathcal{K}$, matrix-vector multiplication can be done by using the two dimensional discrete Fourier transform. That is to say, it will be done as fast as FFTs.

All BCCB matrices can be written as

$$\mathcal{K} = \mathcal{F}^* \Lambda \mathcal{F},$$

where $\mathcal{F}$ is the two-dimensional discrete Fourier transform matrix, $\mathcal{F}^*$ is the complex conjugate transpose of $\mathcal{F}$ and Λ is a diagonal matrix containing the eigenvalues of $\mathcal{K}$. Moreover, using properties of the matrix $\mathcal{F}$, it is not difficult to show that the eigenvalues of $\mathcal{K}$ can be obtained by computing a two-dimensional FFT of the first column of $\mathcal{K}$. Thus, for computing $y = \mathcal{K}x$, we use

$$y = \mathcal{F}^* \Lambda \mathcal{F} x.$$

If $\mathcal{K}$ is a BTTB matrix, matrix-vector multiplication can be done by embedding $\mathcal{K}$ into a larger BCCB matrix, padding outside the borders of the image with an appropriate number of zeros, and then using FFTs as BCCB matrix.

In some cases, for example, the Gaussian point spread function (PSF), $\mathcal{K}$ can also be represented by a kronecker tensor product of two low order matrices A and B as $\mathcal{K} = A \otimes B$. If $\mathcal{K}$ is taken as a band matrix, say, if A and B are taken as band matrices, the performance of the matrix-vector multiplication can even be enhanced.

In the following, we will demonstrate computing $y = \mathcal{K}x$ by an economic algorithm for Gaussian PSF, where $\mathcal{K}$ is of order $mn \times mn$. Suppose A, B are tridiagonal matrices, so the different elements of $\mathcal{K}$ are only $C = A(1:2,1) \otimes B(1:2,1)$. We will give this algorithm in a MATLAB code languages. In Algorithm 4.2, to evaluate y by the function subroutine $y = MatVecTri(m,n,C,x)$, we need to call another subroutine $b = blockmulti(m,n,D,x)$. Some MATLAB notations are introduced: $y(l_1 : l_2)$ represents a new vector whose elements consist of row l_1 to row l_2 of vector y, where $l_1,\ l_j$ are two integers. $b(l_1 : h : l_2)$ represents a new vector whose elements consist of row l_1 to row l_2 of vector b with step size h. Also note that C is in the vector form and only possesses 4 numbers, therefore $C(l_1 : l_2)$ represents choosing elements of C from row l_1 to l_2.

Algorithm 4.2. Block matrix-vector multiplication for Gaussian PSF

```
$function\ y = MatVecTri(m,n,C,x)$
$y = blockmulti(m,n,C(1:2),x);$
$x = blockmulti(m,n,C(3:4),x);$
$y(1:n*(m-1)) = y(1:n*(m-1)) + x(n+1:m*n);$
$y(n+1:m*n) = y(n+1:m*n) + x(1:n*(m-1));$

$function\ b = blockmulti(m,n,D,x)$
$b = zeros(m*n,1);$
$tmp = D(2)*x;$
$b(1:m*n-1) = tmp(2:m*n);$
$b(n:n:m*n) = 0;$
$b = D(1)*x + b;$
$tmp(2:m*n) = tmp(1:m*n-1);$
$tmp(1:n:m*n) = 0;$
$b = b + tmp.$
```

For our algorithm, the cost of the matrix-vector multiplication is only $4mn$ multiplications. On the other hand, if the matrix-vector multiplication is performed by the FFT, then its amount of work is $O(mn\log_2(mn))$. When the band width of $A,\ B$ are set to be 5, the same trick can also be effective. Therefore our algorithm is very effective for some special problems.

5 Numerical simulation

In this section, we give examples on the restoration of remotely sensed images. The blurring process is modelled by a Gaussian point spread function:

$$k(x,y,\xi,\eta)=\frac{1}{2\pi\rho\bar{\rho}}\exp\left(-\frac{1}{2}\left(\frac{x-\xi}{\rho}\right)^2-\frac{1}{2}\left(\frac{y-\eta}{\bar{\rho}}\right)^2\right), \tag{27}$$

where ρ and $\bar{\rho}$ are two constants that characterize the blurring in the x and y directions, respectively. In our test, we choose $\rho=\bar{\rho}=0.7$.

Clearly, (27) can be written in the form

$$k(x,y,\xi,\eta)=k_{1,\rho}(x,\xi)k_{2,\bar{\rho}}(y,\eta), \tag{28}$$

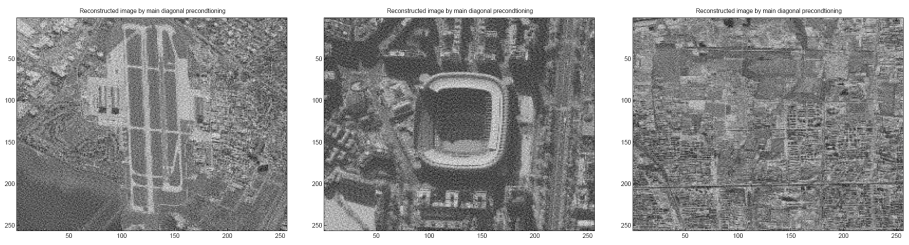
where $k_{i,\zeta}(x,\xi)=\frac{1}{\sqrt{2\pi}\zeta}\exp(-\frac{1}{2}(\frac{x-\xi}{\zeta})^2), i=1,2$. The discretization of the k can be realized by the discretization of k_1 and k_2, both k_1 and k_2 are in Toeplitz form. The tensor product of k_1 and k_2 constitute k. We choose k_1 and k_2 as banded matrix, the band-width is five. Hence the discretization of k is a block Toeplitz matrix with Toeplitz blocks. Note that $k_1,\ k_2$ have narrow bandwidths, the calculation of matrix-vector multiplication and the factorization of $T_k(\lambda)$ in Algorithm 3.3 are extremely cheap.

To simulate the blurring process, we assume that the noise satisfies Gaussian distribution with zero mean and standard deviation σ. So, we add Gaussian white noise to the image data, i.e. instead of $\boldsymbol{f}$, we would have

$$\boldsymbol{f}_n=\boldsymbol{f}+\delta\cdot\text{rand}(mn,1),$$

where $\text{rand}(\cdot,1)$ means the collum vector of Gaussian white noise, δ is the error level which is specified by the users.

We apply our methods to three different remotely sensed images: the first is a 256×256 image of an airfield of California, the second is a 256×256 image of Madrid Stadium, and the third is a 256×256 photogrammetric airborne image of Beijing City. Fig. 1 lists the true images. To test the efficiency of our algorithm, we add different error level noise to the true images. First, we add Gaussian white noise to these images with the error level $\delta=0.001$. Fig. 2 lists the blurred images. Fig. 6 lists the images restored by trust region method whose subproblem is solved by Lanczos method (denoted by LTRS). Fig. 10 lists the images restored by trust region method whose subproblem is solved by preconditioned Lanczos method (denoted by PLTRS). In fig. 10, the preconditioner is taken as main diagonal of $\mathcal{K}^T\mathcal{K}$. The same preconditioning trick is employed to other tests with different error level. Next we add Gaussian white noise to the true images with the error level $\delta=0.005$. Fig. 3 lists the blurred images. Fig. 7 lists the images restored by trust region method whose subproblem is solved by LTRS. Fig. 11 lists the images restored by trust region method whose subproblem is solved by PLTRS. Then we add large Gaussian white noise to the true images with the error level $\delta=0.01$. Fig. 4 lists the blurred images. Fig. 8 lists the images restored by trust region method whose subproblem is solved by LTRS. Fig. 12 lists the images restored by trust region method whose subproblem solved by PLTRS. Finally, we add larger Gaussian white noise to the true images with the error level $\delta=0.02$. Fig. 5 lists the blurred images. Fig. 9 lists the restored images by LTRS. Fig. 13 lists the restored images by PLTRS.


Reconstructed image by main diagonal preconditioning
Reconstructed image by main diagonal preconditioning
Reconstructed image by main diagonal preconditioning
50
100
150
200
250

(a) (b) (c)

Fig. 13. The restoration images by preconditioned Lanczos method for error level $\delta = 0.02$.

Table 1 The iteration results for error level $\delta = 0.001$

Images	$Initer$		$Outiter$		$Error$	
	LTRS	PLTRS	LTRS	PLTRS	LTRS	PLTRS
Airfield	41	35	3	5	4379.5589	4873.0837
Stadium	27	27	3	5	3117.8386	3335.4046
Airborne	43	37	3	5	4640.2131	5196.2899

Table 2 The iteration results for error level $\delta = 0.005$

Images	$Initer$		$Outiter$		$Error$	
	LTRS	PLTRS	LTRS	PLTRS	LTRS	PLTRS
Airfield	44	36	3	5	4627.4995	4937.3513
Stadium	39	27	3	5	3000.2422	3392.6917
Airborne	45	38	3	5	4854.0366	5251.5561

Table 3 The iteration results for error level $\delta = 0.01$

Images	$Initer$		$Outiter$		$Error$	
	LTRS	PLTRS	LTRS	PLTRS	LTRS	PLTRS
Airfield	45	37	3	5	5972.9118	5443.6936
Stadium	47	37	4	5	5884.7627	5709.6846
Airborne	44	27	3	5	4774.5657	3607.3968

Table 4 The iteration results for error level $\delta = 0.02$

Images	$Initer$		$Outiter$		$Error$	
	LTRS	PLTRS	LTRS	PLTRS	LTRS	PLTRS
Airfield	69	48	5	6	11364.7184	8572.6459
Stadium	67	44	5	6	10652.6523	7542.3392
Airborne	67	53	5	6	10838.7183	8687.0703

Acknowledgements We would like to thank the referee's useful suggestions on the numerical experiments. This work was supported by the National Natural Science Foundation of China (Grant Nos. 19731010 and 10231060), and the Knowledge Innovation Program of CAS, and was supported by SRF for ROSS, SEM and also partially supported by the Special Innovation Fund for graduate students of CAS.

References

1. Roggemann, M., Welsh, B., Imaging Through Turbulence, Boca Raton: CRC Press, 1996.

2. Hanke, M., Iterative Regularization Techniques in Image Reconstruction, in Proceedings of the Conference Mathematical Methods in Inverse Problems for Partial Differential Equations, Mt. Holyoke: Springer-Verlag, 1998.
3. Vogel, C. R., Oman, M. E., Fast Robust Total Variation-based Reconstruction of Noisy, Blurred Images, IEEE Transactions on Image Processing, 1998, 7: 813—824.
4. Tikhonov, A. N., Arsenin, V. Y., Solutions of Ill-Posed Problems, New York: Wiley, 1977.
5. Wang, Y. F., Yuan, Y. X., Zhang, H. C. et al., A Trust Region-CG Algorithm for Deblurring Problem in Atmospheric Image Reconstruction, Science in China, Ser. A, 2002, 45: 731—740.
6. Wang, Y. F., On the Regularity of Trust Region-CG Algorithm: with Application to Image Deconvlution Problem, Science in China, Ser. A, 2003, 46: 312—325.
7. Conn, A. R., Gould, N., Toint, Ph. L. et al., Trust Region Methods, CGT publications, 1999.
8. Yuan, Y. X., Sun, W. Y., On the Theory and Methods for Optimization, Beijing: Science Press, 2001.
9. Steihaug, T., The Conjugate Gradient Method and Trust Regions in Large Scale Optimization, SIAM J. Numer. Anal., 1983, 20: 626—637.
10. Toint, Ph. L., Towards an Efficient Sparsity Exploiting Newton Method for Minimization, in Sparse Matrices and Their Uses (ed. Duff, I.), Berlin: Academic Press, 1981, 57—88.
11. Gould, N., Lucidi, S., Roma, M. et al., Solving the trust region subproblem using the Lanczos method, SIAM Journal on Optimization, 1999, 9: 504—525.
12. Chan, R. H., Michael, K. N. et al., Conjugate gradient methods for Toeplitz systems, SIAM Review, 1996, 38(3): 427—482.