Multigrid for Differential-Convolution Problems Arising from Image Processing *

Raymond H. Chan[1], Tony F. Chan[2] and W. L. Wan[2]

[1] Department of Mathematics, Chinese University of Hong Kong, Shatin, Hong Kong. Email: rchan@math.cuhk.edu.hk.

[2] Department of Mathematics, University of Califonia at Los Angeles, Los Angeles, CA 90095-1555. Email: chan@math.ucla.edu, wlwan@math.ucla.edu.

Abstract. We consider the use of multigrid methods for solving certain differential-convolution equations which arise in regularized image deconvolution problems. We first point out that the usual smoothing procedures (e.g. relaxation smoothers) do not work well for these types of problems because the high frequency error components are not smoothed out. To overcome this problem, we propose to use optimal fast-transform preconditioned conjugate gradient smoothers. The motivation is to combine the advantages of multigrid (mesh independence) and fast transform based methods (clustering of eigenvalues for the convolution operator). Numerical results for Tikhonov regularization with the identity and the Laplacian operators show that the resulting method is effective. However, preliminary results for total variation regularization show that this case is much more difficult and further analysis is required.

1 Introduction

In PDE based image processing, we often need to solve differential-convolution equations of the form:

$$\alpha R(u)(x) + \int_\Omega k(x-y)u(y)dy = f(x), \quad x \text{ in } \Omega, \tag{1}$$

where $u(x)$ is the recovered image, $k(x)$ is the kernel convolution function, $R(u)$ is a regularization functional and α is a positive parameter. Typical forms of $R(u)$ are:

$$R(u) = \begin{cases} u & \text{Tikhonov} \\ -\Delta u & \text{Isotropic Diffusion (ID)} \\ -\nabla \cdot (\nabla u/|\nabla u|) & \text{Total Variation (TV).} \end{cases}$$

The discretization of (1) gives rise to a linear system of the form:

$$(\alpha A + K)u = f, \tag{2}$$

* To appear in *Proceedings of the Workshop on Scientific Computing 97*, Springer-Verlag, 1997.

with the following properties. The matrix A, corresponding to the regularization part, is typically sparse, symmetric and positive-definite (positive semi-definite for ID and TV because the boundary condition is Neumann). The matrix K, corresponding to the convolution part, is typically ill-conditioned, symmetric and dense but with a Toeplitz structure. In this paper, we are interested in using iterative methods to solve a large system of the form (2).

The effectiveness of iterative methods depends on the choice of preconditioners. For matrix A, the commonly used preconditioners include [9]: multigrid (MG), domain decomposition (DD), incomplete LU factorization (ILU), successive over-relaxation (SOR) etc. MG or DD type preconditioners have a characteristics of optimal convergence in the sense that its convergence rate is independent of the mesh size.

For matrix K, various preconditioners have been proposed, for example, circulant preconditioners [10, 6, 5], sine transform preconditioners [4], cosine transform preconditioners [2] etc. For these types of preconditioners, the eigenvalues of the preconditioned system typically clustered around one which is a very desirable condition for the conjugate gradient method. Recently, a MG preconditioner [3] has also been proposed and optimal convergence is proved for a class of Toeplitz systems.

The construction of preconditioners for the sum of operators $L = \alpha A + K$, however, is difficult. Suppose M_A and M_K are two efficient preconditioners for A and K respectively. Then $M_L = \alpha M_A + M_K$ would be a good approximation to L. Unfortunately, M_L is not easily invertible in general even if M_A and M_K are.

A simple strategy is to use either M_A or M_K alone to precondition L. In [8, 12], a MG preconditioner is constructed for $\tilde{L} = \alpha A + \gamma I$ which in turn is used to precondition L, hoping that the matrix K is well approximated by γI. A potential drawback is that γI may be a poor approximation to K.

In such situations, the operator splitting method of Vogel and Oman [13] may be more effective. This preconditioner approximates the inverse of L by a product of factors each involving only either A or K:

$$M = (K + \gamma I)^{1/2}(\alpha A + \gamma I)(K + \gamma I)^{1/2},$$

where γ is an appropriately chosen constant. This preconditioner is very effective for both very large and very small values of α but the performance can deteriorate for intermediate values of α.

To alleviate this problem, Chan-Chan-Wong [2] proposed a class of optimal fast-transform based preconditioners to precondition L. The main idea is to select as preconditioner the best approximation to L from a fast-transform invertible class of matrices by solving the following optimization problem:

$$\min_{M \in C} ||M - L||_F,$$

where C is the class of matrices diagonalizable by the cosine-transform. Such optimal fast-transform based preconditioners have proven to be very effective for convolution type problems [5] and they have also been extended to elliptic problems [1]. It turns out that the optimal M for L can be computed very efficiently by exploiting the Toeplitz structure

of K and the banded structure of A. Since L is not "split" in arriving at a preconditioner, the performance is not sensitive to the value of α. However, even though the performance is very satisfactory for Tikhonov and ID regularization, the convergence behavior for the TV regularization case may still depend on the mesh size. This is caused by the highly varying coefficient in the TV operator.

In view of the effectiveness of MG for A and the fast transform preconditioners for K, our idea is to combine the benefits of both. Specifically, we use fast-transform based preconditioned conjugate gradient as a smoother for MG. Our analysis and numerical results show that this is an effective smoother, whereas the standard relaxation type preconditioners are totally ineffective for convolution type problems. In this paper, we shall focus on two 1D cases: (1) $A = I$ (identity) (2) $A = -\Delta$ (Laplacian operator). In sections 2 and 3, we discuss the difficulties of using MG for $L = \alpha I + K$ and $L = -\alpha\Delta + K$ and how we tackle it through the use of fast transform based smoothers. In section 4, we discuss the total variation case. It turns out that this case is much more difficult and although we have some encouraging results, we still have not arrived at an effective method. In 5, we shall estimate the complexity of some of the methods discussed. Finally, some conclusions are made in section 6.

We remark that this paper is only a preliminary report of on-going work and much further investigation remains to be done.

2 The Case $A = I$

In this section, we shall consider operators of the form $L = \alpha I + K$, where K arises from the discretization of an integral operator of the first kind. It is well-known that K is very ill-conditioned and MG with traditional smoothers does not work well for K. The regularization term αI improves the conditioning by shifting the spectrum a distance α away from zero. It turns out that this is not enough to make MG work well. The reason is that the set of eigenvectors remains the same independent of α. We shall explain this phenomenon next.

Our observation is that common relaxation methods, for instance, Richardson, Jacobi or Gauss-Seidel method, fail to smooth the error in the geometric sense. The reason is that, unlike in the elliptic case, eigenvectors of $\alpha I + K$ corresponding to small eigenvalues are highly oscillatory while those corresponding to large eigenvalues are smooth. It is known that relaxation methods reduce the error components corresponding to large eigenvalues only and therefore they in fact remove the smooth error components. We illustrate this using Richardson iteration as an example. Let A be a symmetric positive definite matrix and let $0 < \lambda_1 \leq \cdots \leq \lambda_n$ be its eigenvalues and $\{v_k\}$ the corresponding eigenvectors. The error e^{m+1} in the $m+1$st iteration step of the Richardson method is given by

$$e^{m+1} = (I - \frac{1}{\lambda_n}A)e^m.$$

Let the eigendecomposition of e^m be $e^m = \sum_{k=1}^{n} \xi_k v_k$. Since $\{v_k\}$ are

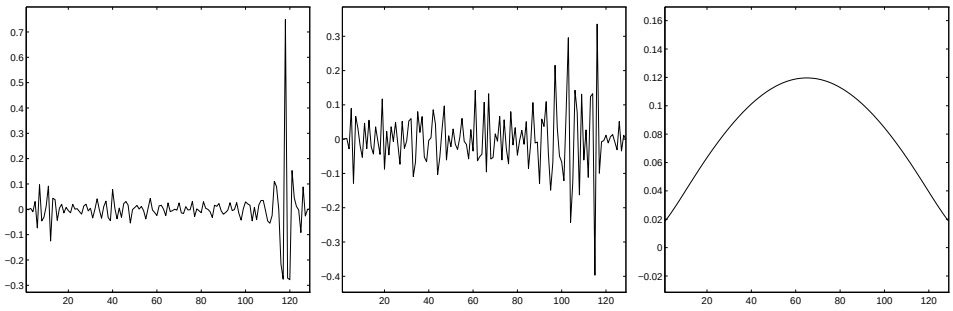

Fig. 1. Eigenvectors corresponding to (a) the smallest (b) the middle (c) the largest eigenvalue of $L = 10^{-4}I + K$. The oscillatory eigenvectors corresponding to the small eigenvalues.

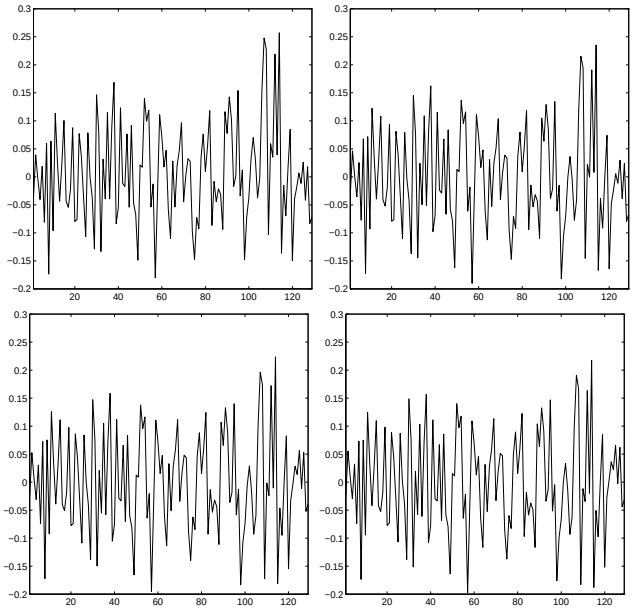

Fig. 2. Error vectors after 0 iteration (top left), 1 iteration (top right), 5 iterations (bottom left) and 10 iterations (bottom right) of Richardson smoothing applied to $L = 10^{-4}I + K$. Note that there is no smoothing effect.

orthogonal by the symmetry of A, we have

$$\|e^{m+1}\|_2^2 = \sum_{k=1}^{n}(1 - \frac{\lambda_k}{\lambda_n})^2\xi^2.$$

Note that $(1 - \lambda_k/\lambda_n) \approx 0$ when k is close to n and $(1 - \lambda_k/\lambda_n) \approx 1$ when k is close to 1. Hence, the components corresponding to large

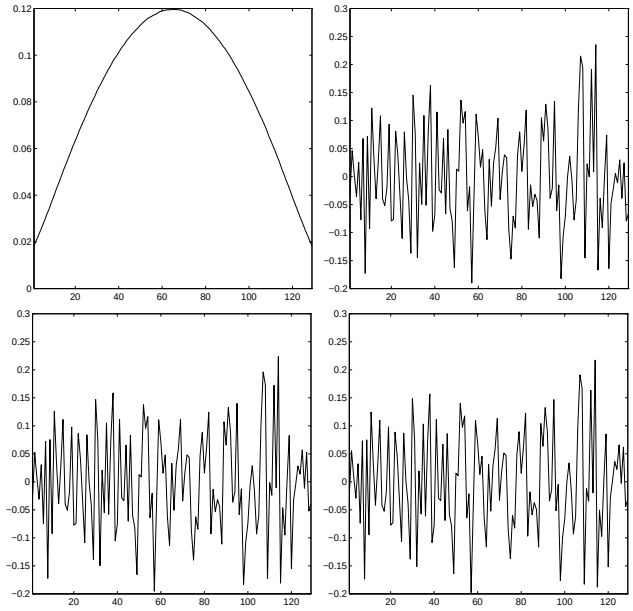

Fig. 3. Error vectors after 0 iteration (top left), 1 iteration (top right), 5 iterations (bottom left) and 10 iterations (bottom right) of Richardson smoothing applied to $L = 10^{-4}I + K$. The smooth component is removed completetly after only 1 iteration whereas the oscillatory components persist. All the plots are scaled so that the l_2-norm of the vector is equal to 1.

eigenvalues are reduced while those corresponding to small eigenvalues remain essentially unchanged.

We illustrate the smoothing phenomenon of the Richardson iteration applied to $L = \alpha I + K$ by a simple example. Choose $\alpha = 10^{-4}$ and $k(x) = \frac{1}{C}\exp(-x^2/0.01)$ which is known as the Gaussian blurring operator in image processing. Here $C = \int_0^1 \exp(-x^2/0.01)dx$ is the normalization constant. Let $0 < \lambda_1 \leq \cdots \leq \lambda_n$ be the eigenvalues of L and $v_1, \ldots, v_n$ be the corresponding eigenvectors. Figure 1 shows the plots of v_1, $v_{n/2}$ and v_n for $n = 128$. Relaxation methods, for example, the Richardson method, essentially reduces the error components corresponding to large eigenvalues, not necessary the high frequencies. Because of the special spectrum of L, these methods do not reduce the high frequency errors. Figure 2 shows the plots of the initial (oscillatory) error and the errors after 1, 5, 10 number of Richardson iterations. No smoothing effect can be seen. In fact, as shown in Figure 3, if the initial error consists of low frequency and a small perturbation of high frequency vectors, after one Richardson iteration, the low frequency components will be removed and the error is left with high frequency only.

In contrast, MG converges rapidly for integral operators of the second kind of the form $L = I - K$ and this can also be explained by the smooth-

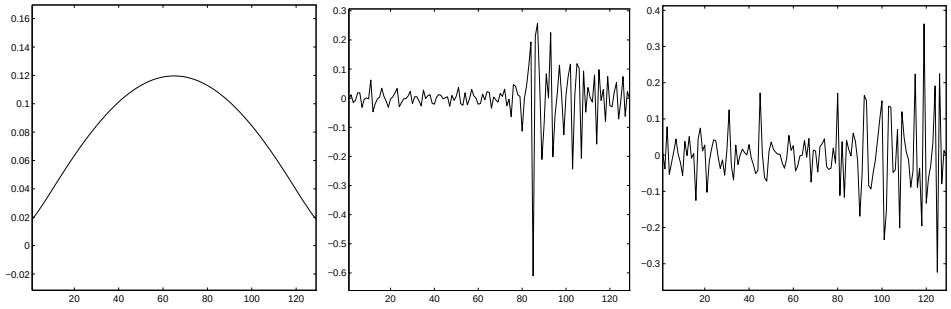

Fig. 4. Eigenvectors corresponding to (a) the smallest (b) the middle (c) the largest eigenvalue of $L = I - K$. The oscillatory eigenvectors correspond to the largest eigenvalues.

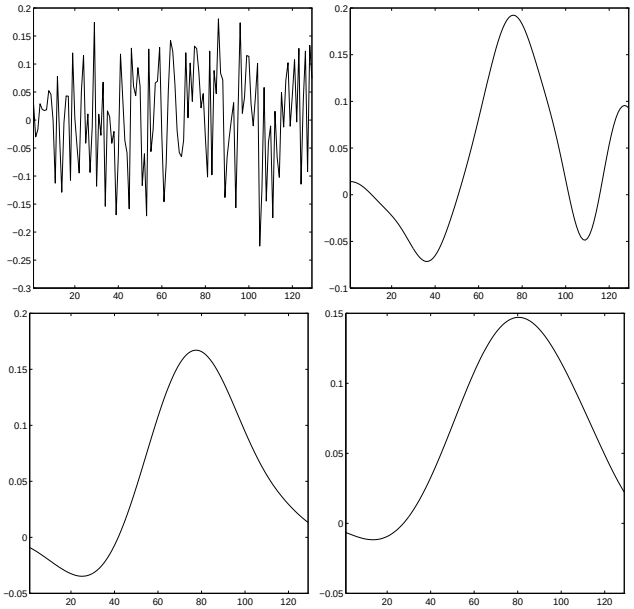

Fig. 5. Error vectors after 0 iteration (top left), 1 iteration (top right), 5 iterations (bottom left) and 10 iterations (bottom right) of Richardson smoothing applied to $L = I - K$. The oscillatory components are quickly smoothed out.

ing argument. Figure 4 shows the eigenvectors of $L = I - K$ with K as before. Because of the minus sign, we see that eigenvectors corresponding to small eigenvalues are smooth while those of large eigenvalues are oscillatory as in the standard elliptic case. Thus the Richardson iteration has no trouble removing high frequency errors as shown in Figures 5 and 6. We note that in Figure 6, the initial error consists of small perturbation of high frequency vectors.

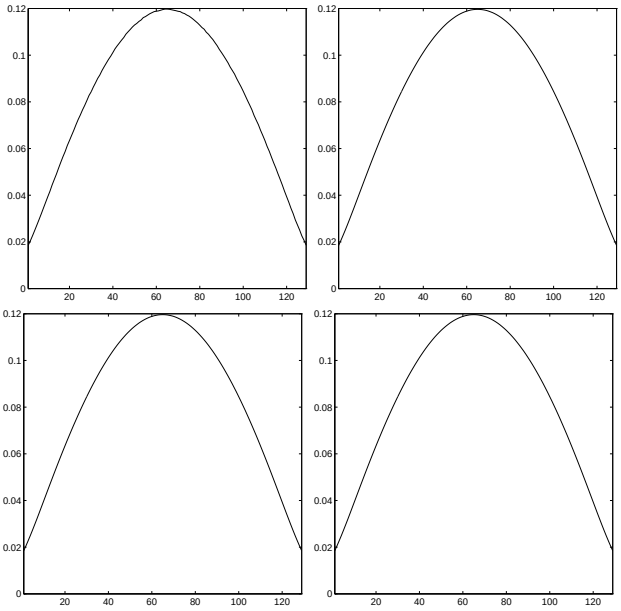

Fig. 6. Error vectors after 0 iteration (top left), 1 iteration (top right), 5 iterations (bottom left) and 10 iterations (bottom right) of Richardson smoothing applied to $L = I - K$. The smooth components remain after many iterations.

With the above understanding, it is clear that MG does not work well for $L = \alpha I + K$ because the standard smoothers are not effective and we need to devise smoothers which can remove high frequency error components more effectively. Our approach is based on two observations. First, fast-transform preconditioners are effective for clustering the eigenvalues of L around one. Second, conjugate gradient annihilates efficiently error components corresponding to clusters of eigenvalues, in addition to those at both ends of the spectrum. Hence we propose to use PCG with fast transform preconditioners as smoother in the MG cycle.

Figure 7 shows the eigenvectors of the preconditioned system using the cosine transform preconditioner. It is interesting to note that low frequency vectors are located at both ends of the spectrum while high frequency vectors concentrate at the cluster. Figure 8 shows the smoothing effect of PCG using the cosine-transform preconditioner (PCG(Cos)). We remark that MG with the optimal circulant preconditioner also produces similar plots and hence we do not show it. Table 1 shows the MG convergence (MG(*)) of different smoothers specified in the brackets. The Richardson smoother is denoted by R and the PCG smoother with the cosine-transform preconditioner is denoted by PCG(Cos). The convergence of PCG(Cos) alone is also given for comparison. Here we use two pre-smoothing and no post-smoothing step. The iteration is stopped when the relative residual is less than 10^{-10}. The matrix K is the Gaussian blurring operator as before. From the table, we see that PCG as

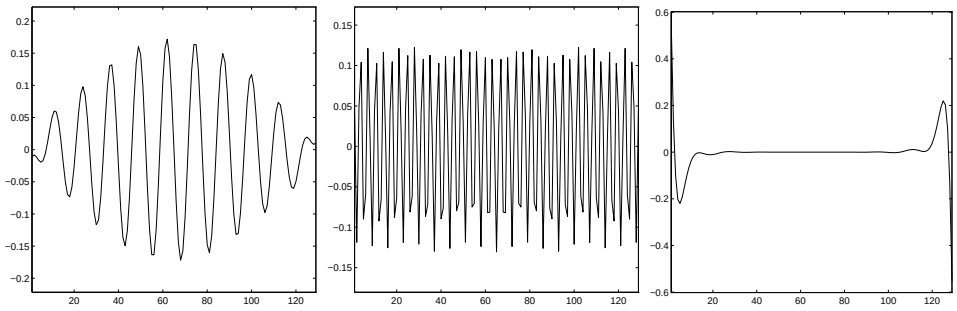

Fig. 7. Eigenvectors corresponding to (a) the smallest (b) the middle (c) the largest eigenvalue of the cosine transform preconditioned system of $L = 10^{-4}I + K$. The oscillatory eigenvectors are clustered in the middle of the spectrum.

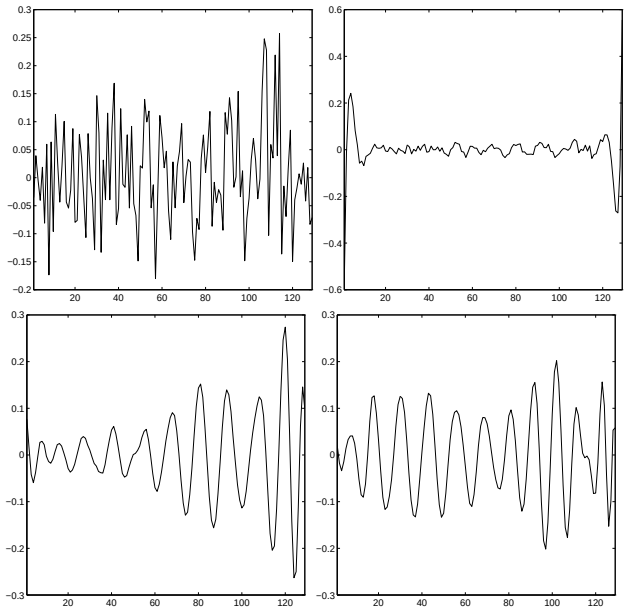

Fig. 8. Error vectors after 0 iteration (top left), 1 iteration (top right), 5 iterations (bottom left) and 10 iterations (bottom right) of PCG(Cos) smoothing applied to $L = 10^{-4}I + K$. The smoothing effect is much improved over Richardson in Figure 2.

smoother is much more efficient than standard relaxation methods in all cases. For large α, MG with PCG as smoother is about as efficient as PCG alone, taking into the account of two smoothing steps in each MG iteration. But for small α, MG is significantly better. In fact, its performance improves as the mesh size approaches zero whereas that of PCG alone remains constant.

α	h	1/64	1/128	1/256	1/512
10^{-2}	MG(R)	*	*	*	*
	MG(PCG(Cos))	5	4	4	4
	PCG(Cos)	8	8	8	8
10^{-3}	MG(R)	*	*	*	*
	MG(PCG(Cos))	6	5	4	4
	PCG(Cos)	11	11	11	11
10^{-4}	MG(R)	*	*	*	*
	MG(PCG(Cos))	11	7	6	6
	PCG(Cos)	18	18	18	18
10^{-5}	MG(R)	*	*	*	*
	MG(PCG(Cos))	40	18	14	11
	PCG(Cos)	33	37	36	38

Table 1. Convergence of different MG and PCG with varying α and mesh size h. $L = \alpha I + K$. $*$ indicates more than 100 iterations. The results show that PCG(Cos) is an effective smoother.

3 The Case $A = -\Delta$

In the following, we shall assume Neumann boundary condition for the Laplacian operator. The situation of $L = -\alpha\Delta + K$ is much more complicated. First of all, the regularization term $\alpha\Delta$ does not simply shift the spectrum; it actually alters the spectrum. For large α, the eigenvectors of L resemble those of Δ and for small α, they resemble those of K, where the high and low frequency vectors are flipped over each other. For α in between, it is a mixture but the precise nature of the mixing is not known. We pick three different size of α to illustrate the changing spectrum of L in Figures 9, 10 and 11.

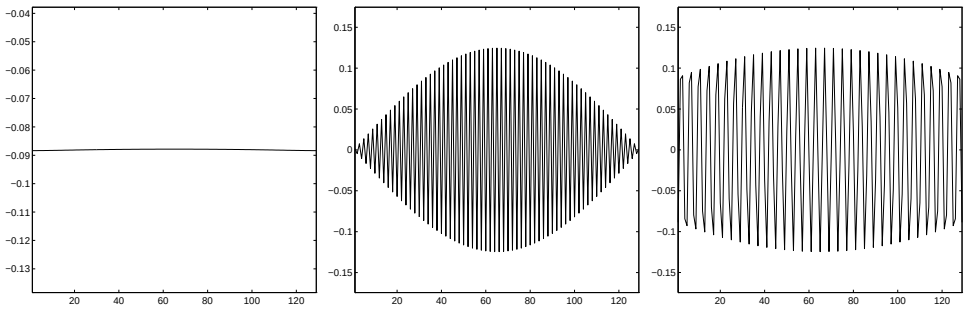

Fig. 9. Eigenvectors corresponding to (a) the smallest (b) the middle (c) the largest eigenvalue of $L = -\Delta + K$. When α is large, the eigenvectors of L resemble those of $-\Delta$.

The numerical results for this case is given in Table 2. As expected,

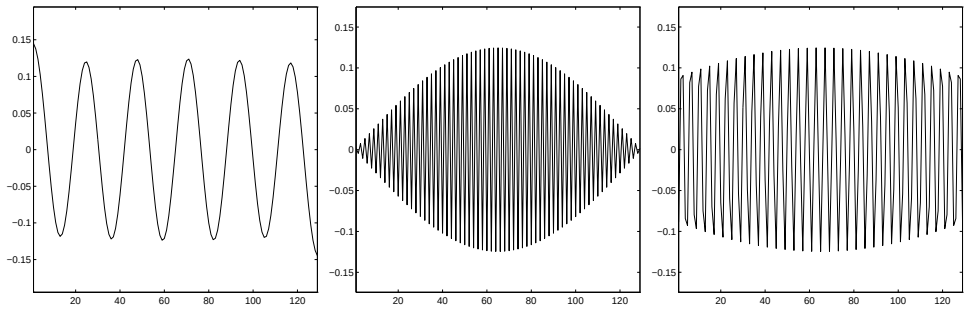

Fig. 10. Eigenvectors corresponding to (a) the smallest (b) the middle (c) the largest eigenvalue of $L = -10^{-4}\Delta + K$. For intermediate value of α, the eigenvectors corresponding to large eigenvalues resemble those of $-\Delta$ and the eigenvectors corresponding to small eigenvalues are oscillatory and resemble those of K.

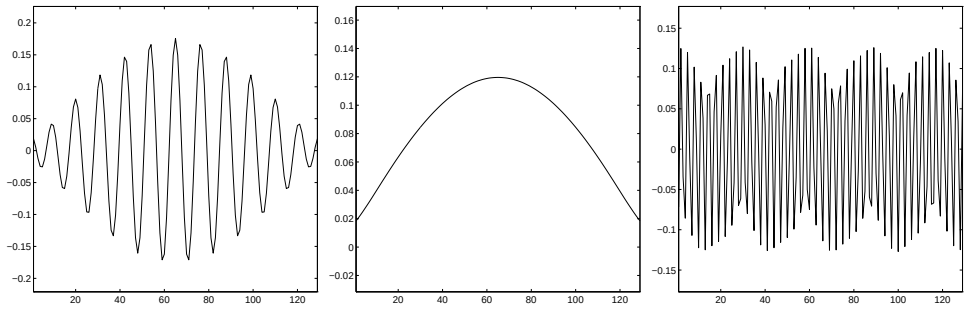

Fig. 11. Eigenvectors corresponding to (a) the smallest (b) the middle (c) the largest eigenvalue of $L = -10^{-8}\Delta + K$. When α is small, the eigenvectors of L resemble those of K.

MG with standard relaxation methods as smoother deteriorates when α decreases because L approaches the convolution operator K which we have shown in section 2 that standard smoothers do not work well. Again, MG(PCG(Cos)) shows better performance over PCG(Cos) alone for small values of α and h.

4 MG for TV deblurring

In this section, we shall discuss our preliminary experience in solving the TV deblurring problem [7] by MG. The governing differential-convolution equation is slightly different from (1) and is given here:

$$\alpha R(u)(x) + \mathcal{K}^*\mathcal{K}(u) = \mathcal{K}^* z,$$

where $R(u) = -\nabla \cdot (1/|\nabla u|)\nabla u$, $\mathcal{K}u = \int_\Omega k(x-y)u(y)dx$, $\mathcal{K}^*$ is the adjoint operator of $\mathcal{K}$ and z is the observed blurred and noisy image.

α	h	1/64	1/128	1/256	1/512
10^{-2}	MG(R)	18	19	20	21
	MG(PCG(Cos))	3	3	3	3
	PCG(Cos)	6	6	6	6
10^{-3}	MG(R)	17	18	19	20
	MG(PCG(Cos))	3	3	3	3
	PCG(Cos)	7	7	7	7
10^{-4}	MG(R)	32	32	32	32
	MG(PCG(Cos))	4	4	3	3
	PCG(Cos)	8	8	8	8
10^{-5}	MG(R)	*	*	*	*
	MG(PCG(Cos))	4	4	3	3
	PCG(Cos)	9	9	9	9

Table 2. Convergence of different MG and PCG with varying α and mesh size h. $L = -\alpha\Delta + K$. The results show that PCG(Cos) is an effective smoother.

Basically, the convolution operator is replaced by a product of itself with its adjoint. The corresponding linear system is:

$$(\alpha A + K^T K)u = f, \tag{3}$$

which is similar to (2) with K replaced by $K^T K$. The additional challenges of solving (3) are two fold. First, the matrix A now comes from an elliptic operator with highly varying coefficient (which is $1/|\nabla u|$). It is not known if MG can handle this case efficiently. Second, the product $K^T K$ is no longer Toeplitz which complicates the implementation issue. For instance, while it is trivial to construct the Jacobi preconditioner for K, it is not so for $K^T K$ at first glance, although it turns out that it can also be done in $O(n)$ operations. Moreover, the conditioning of $K^T K$ is worse than K alone.

It turns out MG even with PCG as smoother does not work well in this case. A natural way to improve its performance is to use it as a preconditioner for conjugate gradient. However, this is not feasible as MG with PCG as smoother gives rise to a nonstationary preconditioner. One solution to this problem is based on the following observation. The success of PCG as smoother is that CG takes advantage of the clustered eigenvalues of the cosine-transform preconditioned system. We notice that it is probably advantageous but not necessary to apply CG (which gives rise to a nonstationary preconditioner) to the preconditioned system. An alternative is to use standard relaxation methods on the cosine-transform preconditioned system.

We have tried out several possibilities and the results are shown in Tables 3, 4 and 5 for a TV deblurring example. The original and the blurred noisy 1D image together with the recovered image are shown in Figure 12. The signal-to-noise ratio SNR=13. Here we have used the Gaussian blur again. For each grid size h, we use the *optimal* α_{opt} for L which is chosen so that the recovered image has the specified SNR. We test three cases: $\alpha = 10 * \alpha_{opt}$, $\alpha = \alpha_{opt}$ and $\alpha = 0.1 * \alpha_{opt}$, corresponding

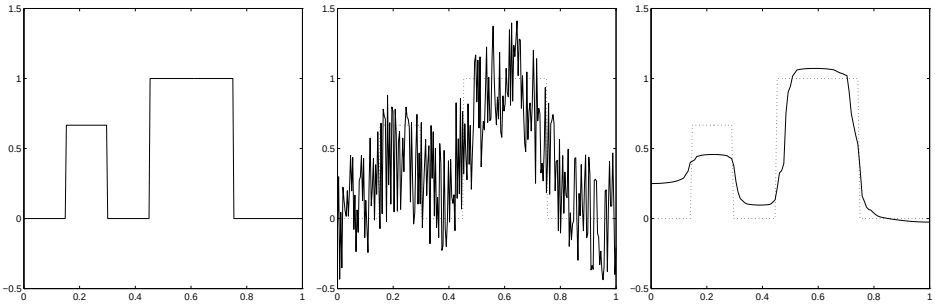

Fig. 12. (a) Original image (b) Blurred and noisy image (c) Recovered image. Gaussian blur is used and SNR=13.

to Tables 3-5 respectively. In each table, the second to fourth column show the convergence in the first fixed point iteration and the fifth to seventh ones show the convergence at the 11th fixed point iteration. (For our examples, the fixed point iteration has already converged at the 11th iteration.) We show these two sets of results because the coefficient $1/|\nabla u|$ is quite different for the two cases; see Figure 12. In the first fixed point iteration, the coefficient is very oscillatory whereas at the eleventh iteration, it is almost piecewise constant. With the same notation as before, the bracket right after PCG specifies the preconditioner used for CG and the bracket right after MG specifies the smoother. Here GS+Cos denotes the Gauss-Seidel (GS) method applied to the cosine-transform preconditioned system. Similarly for J+Cos where J denotes the Jacobi method.

	1st fixed pt. iter.			11th fixed pt. iter.		
$10 * \alpha_{opt}$	1/64	1/128	1/256	1/64	1/128	1/256
PCG(Cos)	42	85	108	13	49	75
PCG(MG(GS))	20	29	35	12	17	21
PCG(MG(GS+Cos))	14	28	37	4	11	12
PCG(MG(J+Cos))	17	51	79	13	17	22

Table 3. Convergence of PCG with varying h. $\alpha = 10 * \alpha_{opt}$.

We see that PCG(MG(GS)) and PCG(MG(GS+Cos)) are the best. They are not sensitive to α and deterioration with smaller h is slow. Besides, PCG(MG(GS+Cos)) is better than PCG(MG(GS)) for smaller α which shows that cosine transform is effective in dealing with K. However, we have not come up with an efficient implementation for these two methods. For PCG(MG(GS)), Vogel [11] has also made this observation independently. PCG(MG(J+Cos)) shows a degradation over PCG(MG(GS+Cos)), similar to that of the ordinary GS over Jacobi. We should also remark that PCG(Cos) is quite effective among the methods

	1st fixed pt. iter.			11th fixed pt. iter.		
α_{opt}	1/64	1/128	1/256	1/64	1/128	1/256
PCG(Cos)	38	81	98	42	92	106
PCG(MG(GS))	17	28	37	16	21	24
PCG(MG(GS+Cos))	14	26	34	13	15	14
PCG(MG(J+Cos))	17	45	73	20	28	28

Table 4. Convergence of PCG with varying h. $\alpha = \alpha_{opt}$.

	1st fixed pt. iter.			11th fixed pt. iter.		
$0.1 * \alpha_{opt}$	1/64	1/128	1/256	1/64	1/128	1/256
PCG(Cos)	35	75	93	55	90	128
PCG(MG(GS))	19	35	54	15	21	25
PCG(MG(GS+Cos))	13	24	33	15	19	17
PCG(MG(J+Cos))	16	43	67	22	35	36

Table 5. Convergence of PCG with varying h. $\alpha = 0.1 * \alpha_{opt}$.

we have tried.

5 Computation complexity

Here we estimate the complexity of one iteration of some of the methods that we have described in Sections 2 and 3.

PCG(Cos): This method has been estimated in [2]. The construction of the preconditioner is $O(n)$ and the cost of the preconditioning is $O(n \log_2 n)$.

MG(R): On each level, the cost of a Richardson smoothing is essentially the cost of matrix-vector multiply. For the sparse matrix A, it can be done in $O(n_l)$ operations and for the Toeplitz matrix, it can be done in $O(n_l \log_2 n_l)$, where n_l is the size of the matrix at level l. Here we assume that K is Toeplitz at all levels. In fact, this is proved to be true in [3] if linear interpolation is used. The construction of the coarse grid matrices can also be done in $O(n)$. Thus the overall complexity of an iteration of MG(R) is $O(n \log_2 n)$.

MG(PCG(Cos)): The method is almost the same as MG(R) but with different smoother. The cost of applying the PCG(Cos) is $O(n \log_2 n)$ and hence the overall complexity is $O(n \log_2 n)$.

We remark that we have not come up with an efficient implementation of the methods in the TV case and so we do not discuss the complexity issue of those methods here.

6 Conclusions

We have shown in section 2 that standard smoothers do not work for matrices of the form $\alpha I + K$ arising from convolution operators. We have proposed to use PCG as smoother and demonstrated numerically that it is effective to reduce oscillatory errors. We have also tested the matrices of the form $-\alpha\Delta + K$ and the PCG smoother works as well. For the TV image deblurring, the situation is complicated by the highly varying coefficient and the product of convolution operators. We have proposed several multigrid preconditioners and the numerical results are satisfactory. However the implementation issue is still left open. Further investigation is needed to devise a practical and efficient multigrid preconditioner in this case.

7 Acknowledgment

Research of R. Chan has been partially supported by HKRGC Research Grant CUHK 178/93E. Research of Tony F. Chan has been partially supported by the ONR under Contract N00014-96-1-0277 and the NSF under contract DMS-9626755. Research of W. L. Wan has been partially supported by the grants listed under the second author and the Alfred P. Sloan Foundation as a Doctoral Dissertation Fellow.

References

1. R. Chan and T. Chan. Circulant preconditioners for elliptic problems. *Numer. Linear Algebra Appl.*, 1:77–101, 1992.
2. R. Chan, T. Chan, and C. Wong. Cosine transform based preconditioners for total variation minimization problems in image processing. Technical Report 95-23, Dept. of Mathematics, UCLA, 1995.
3. R. Chan, Q. Chang, and H. Sun. Multigrid method for ill-conditioned symmetric toeplitz systems. Technical Report 95-12, The Chinese University of Hong Kong, 1995.
4. R. Chan, K. Ng, and C. Wong. Sine transform based preconditioners for symmetric toeplitz systems. *Linear Algebra Appls.*, 232:237–260, 1996.
5. R. Chan and M. Ng. Conjugate gradient methods for toeplitz systems. *SIAM Review*, 38:427–482, 1996.
6. T. Chan. An optimal circulant preconditioner for toeplitz systems. *SIAM J. Sci. Stat. Comput.*, 9:766–771, 1988.
7. T. Chan and P. Mulet. Iterative methods for total variation image restoration. Technical Report 96-38, Dept. of Mathematics, UCLA, 1996.
8. M. Omen. Fast multigrid techniques in total variation-based image reconstruction. In *Proceedings of the 1995 Copper Mountain Conference on Multigrid Methods*, 1995.
9. Y. Saad. *Iterative Methods for Sparse Linear Systems*. PWS Kent Publishing Co., 1995.

10. G. Strang. A proposal for toeplitz matrix calculations. *Stud. Appl. Math.*, 74:171–176, 1986.
11. C. R. Vogel. Private communication. March 97.
12. C. R. Vogel. A multigrid method for total variation-based image denoising. In K. Bowers and J. Lund, editors, *Computation and Control IV*. Birkhauser, 1995.
13. C. R. Vogel and M. Oman. Fast, robust total variation-based reconstruction of noisy, blurred images. 1996. Submitted to IEEE Transactions on Image Processing.

This article was processed using the LaTeX macro package with LLNCS style