Cosine Transform Based Preconditioners for Total Variation Deblurring

Raymond H. Chan, Tony F. Chan, Chiu-Kwong Wong

Abstract— **Image reconstruction is a mathematically ill-posed problem and regularization methods are often used to obtain a reasonable solution. Recently, the total variation (TV) regularization, proposed by Rudin, Osher and Fatemi (1992), has become very popular for this purpose. In a typical iterative solution of the nonlinear regularization problem, such as the fixed point iteration of Vogel or Newton's method, one has to invert linear operators consisting of the sum of two distinct parts. One part corresponds to the blurring operator and is often a convolution; the other part corresponds to the TV regularization and resembles an elliptic operator with highly varying coefficients. In this paper, we present a preconditioner for operators of this kind which can be used in conjunction with the conjugate gradient method. It is derived from combining fast transform (e.g. cosine-transform based) preconditioners which the authors had earlier proposed for Toeplitz matrices and for elliptic operators separately. Some numerical results will be presented. In particular, we will compare our preconditioner with a variant of the product preconditioner proposed by Vogel and Oman [28].**

I. Introduction

In this paper, we apply conjugate gradient preconditioners to the iterative solution of some large-scale image processing problems. The quality of the recorded image is usually degraded by blurring and noise. The recorded image z and the original image u are often related by the equation,

$$\begin{aligned} z(x,y) &= \mathcal{H}u(x,y) + \eta(x,y) \\ &\equiv \int_\Omega h(x-s,y-t)u(s,t)\,dt\,ds + \eta(x,y), \end{aligned} \tag{1}$$

see [12]. Here $\mathcal{H}$ denotes the blurring operator for the blurring function h, and η denotes the noise function. The image restoration problem is to obtain a reasonable approximation of the original image.

Note that the problem $\mathcal{H}u = z$ is ill-posed and the discretization matrix of $\mathcal{H}$ is usually ill-condition and hence the problem is extremely sensitive to noise. Thus, we cannot neglect the effect of noise and simply solve $\mathcal{H}u = z$. To remedy this ill-conditioning, several approaches of regularization are used.

The *Total Variation* (TV) regularization method of Rudin, Osher and Fatemi [25], [24] is extremely effective approach for restoring edges of the original image. They consider solving the following constrained minimization problem:

$$\min_u \int_\Omega |\nabla u|\,dx\,dy \quad \text{subject to} \quad \|\mathcal{H}u - z\|_{L^2(\Omega)} = \sigma \tag{2}$$

where $|\cdot|$ denotes the Euclidean norm and σ is the noise level. The quantity $\int_\Omega |\nabla u|\,dx\,dy$ is called the total variational norm of u.

Instead of solving the constrained problem, Vogel considered the following closely-related regularization problem:

$$\min_u f(u) = \min_u \frac{1}{2}\|\mathcal{H}u - z\|^2_{L^2(\Omega)} + \alpha \int_\Omega |\nabla u|\,dx\,dy, \tag{3}$$

see [1], [27]. Here α is a positive parameter which measures the trade off between a good fit and an oscillatory solution. At a stationary point of (3), the gradient of f vanishes, giving:

$$\begin{aligned} g(u) \equiv \mathcal{H}^*(\mathcal{H}u - z) - \alpha\nabla\cdot\left(\frac{\nabla u}{|\nabla u|}\right) = 0, \quad (x,y)\in\Omega, \\ \frac{\partial u}{\partial n} = 0, \quad (x,y)\in\partial\Omega. \end{aligned} \tag{4}$$

The second term in g is obtained by taking the gradient of $\alpha\int_\Omega |\nabla u|\,dx\,dy$ and then applying integration by parts from which Neumann boundary condition results. We remark that the Euler-Lagrange equation for (2) also has a form similar to (4).

Due to the term $1/|\nabla u|$, (4) is a degenerate nonlinear second order diffusion equation. The degeneracy can be removed by adding a positive parameter β to $|\nabla u|$; see [27]. More precisely, if we let

$$\kappa_\beta(u) = \frac{1}{\sqrt{|\nabla u|^2 + \beta}}, \qquad \mathcal{L}_u v = -\nabla\cdot(\kappa_\beta(u)\nabla v) \tag{5}$$

and

$$\mathcal{A}_u v \equiv (\mathcal{H}^*\mathcal{H} + \alpha\mathcal{L}_u)v,$$

then (4) becomes the following non-degenerate system

$$\begin{aligned} \mathcal{A}_u u = \mathcal{H}^* z, \quad (x,y)\in\Omega, \\ \text{with} \qquad \frac{\partial u}{\partial n} = 0, \quad (x,y)\in\partial\Omega. \end{aligned} \tag{6}$$

A recent survey of related PDE approach to image analysis can be found in [2].

Raymond H. Chan is with Department of Mathematics, The Chinese University of Hong Kong, Shatin, Hong Kong (http://www.math.cuhk.edu.hk/~ rchan). Research supported in part by CUHK DAG 2060110 and Summer Research Grant.

Tony F. Chan is with the Department of Mathematics, University of California, Los Angeles, 405 Hilgard Avenue, Los Angeles, CA 90095-1555 (http://www.math.ucla.edu/~ chan). Supported by grants ONR-N00017-96-1-0277 and NSF DMS-9626755. Part of this work was performed during a visit to the Department of Mathematics at the CUHK.

Chiu-Kwong Wong is with Department of Mathematics, University of California, Los Angeles, 405 Hilgard Avenue, Los Angeles, CA 90095-1555 (http://www.math.ucla.edu/~ ckwong).

In [27], Vogel introduced the "lagged diffusivity fixed point iteration", which we denote by FP, to solve the system (6). If A_{u^k}, H and L_{u^k} denote respectively the discretization matrices of $\mathcal{A}_{u^k}$, $\mathcal{H}$ and $\mathcal{L}_{u^k}$, then the FP iteration will produce a sequence of discrete approximations $\{u^k\}$ to the solution u and can be expressed as

$$A_{u^k}u^{k+1} \equiv (H^*H+\alpha L_{u^k})u^{k+1} = H^*z, \quad k = 0, 1, \ldots . \quad (7)$$

Note that obtaining u^{k+1} from the solution of u^k requires solving a linear system with coefficient matrix $H^*H+\alpha L_{u^k}$.

For the image restoration problem in (1), H corresponds to a discretization of the convolution operator, and often H will be a Toeplitz matrix. Thus, the coefficient matrix in (7) corresponds to a sum of a convolution operator and an elliptic operator. We emphasize that it is not easy to devise fast iterative algorithms to solve this linear system. For example, the technique of applying multigrid method to solve such linear system is not yet well developed, see [9]. Vogel and Oman [28] has recently proposed using a "product" preconditioner for (7) which allows the deblurring part H^*H and the PDE part L_{u^k} to be preconditioned separately. An alternative approach to solving the gradient equation (7) is to directly solve the minimization problem (3) by non-smooth optimization techniques; see for example [21], [22].

In this work, we apply the preconditioned conjugate gradient (PCG) method to solve (7) and we concentrate on finding a good preconditioner for (7). Given a matrix A, there are two criteria for choosing a preconditioner for A; see [18]. First, a preconditioner should be a "good" approximation to A. Secondly, it must be easily invertible. Recall that A_{u^k} corresponds to a sum of a convolution operator and an elliptic operator. For matrices arising from elliptic boundary value problem, a "good" preconditioner must retain the boundary condition of the given operator [23]. Based on this idea, optimal sine transform preconditioners were constructed [15] for elliptic problems with Dirichlet boundary condition. If the boundary is rectangular, it was proved [15], [29] that the convergence rate of the PCG method with this preconditioner is independent of the grid size. In our present problem, Neumann boundary condition is imposed, see (4). Since the discrete Laplacian on a unit square with Neumann boundary conditions can be diagonalized by the discrete cosine transform matrix, this motivates us to use the optimal cosine transform approximation [11] to L_{u^k} as a preconditioner for the elliptic part in (7).

In addition, R. Chan, Ng and Wong [14] applied the sine transform approximation to construct preconditioners for Toeplitz systems. It gives rise to fast convergence of the PCG method. It has been shown in [20] that the PCG method with the optimal cosine transform approximation can also produce the same good convergence result.

By combining the results mentioned in the previous two paragraphs, we here propose a preconditioner for the system (7) by taking the sum of the cosine transform approximations to the matrices H^*H and L_{u^k} separately. The resulting approximation can still be diagonalized by the discrete cosine transform matrix and therefore easily invertible. This preconditioner was originally proposed in [10] where preliminary results and numerical experiments were given. In this paper, we will give a detailed discussion and analysis of the preconditioner, including a comparison of our preconditioner with the product preconditioner proposed by Vogel and Oman [28].

The outline of the paper is as follows. In the next section, we will define and construct the optimal cosine transform approximation for a general matrix. In Section III, we will use the approximation to construct a preconditioner for the system (7). In Section IV, we will introduce Vogel and Oman's product preconditioner and some of its possible variants. In the final section, numerical performance of the preconditioner will be presented.

II. Optimal Discrete Cosine Transform Preconditioner

The concept of optimal transform approximation was first introduced by T. Chan [16]. Since preconditioners can be viewed as approximations to the given matrix A_n, it is reasonable to consider preconditioners which minimize $||B_n - A_n||$ over all B_n belonging to some class of matrices and for some matrix norm $||\cdot||$. T. Chan [16] proposed *optimal* circulant preconditioner that is the minimizer of the Frobenius norm $\|B_n - A_n\|_F$ over the class of all circulant matrices B_n. These preconditioners have been proved to be very efficient preconditioners for solving Toeplitz systems with the PCG method; see [5].

Analogously, R. Chan, Ng and Wong [14] defined the optimal sine transform preconditioner to be the minimizer of $\|B_n - A_n\|_F$ over all matrices B_n which can be diagonalized by the discrete sine transform. They proved that for a large class of Toeplitz system, the PCG method with the sine transform preconditioner converges at the same rate as the optimal circulant one. Following the same approach, we will construct in Section II-A the optimal cosine transform preconditioner for general matrices. We remark that although the derivation is almost the same as that of the sine transform preconditioner, we present it here for the seek of charity. An alternative derivation using displacement structure can be found in [20]. The preconditioner will be applied to precondition both H^*H and L_{u^k} in (7) separately. For a survey on fast transform type preconditioners, we refer the reader to [8].

A. Construction of One-dimensional Preconditioner

Let us denote C_n to be the n-by-n discrete cosine transform matrix. If δ_{ij} is the Kronecker delta, then the (i,j)th entry of C_n is given by

$$\sqrt{\frac{2-\delta_{j1}}{n}}\cos\left(\frac{(2i-1)(j-1)\pi}{2n}\right), \quad 1 \le i,j \le n,$$

see Sorensen and Burrus [26, p.557]. We note that the C_n's are orthogonal, i.e. $C_nC_n^t = I_n$. Also, for any n-vector v, the matrices-vector multiplication C_nv and C_n^tv can be computed in $O(n\log n)$ real operations; see [30].

A_n	cost of constructing $c(A_n)$
general	$O(n^2)$
Toeplitz	$O(n)$
banded	$O((b_l + b_u)n)$

TABLE I
COST OF CONSTRUCTING $c(A_n)$.

Let $\mathcal{B}_{n\times n}$ be the vector space containing all matrices that can be diagonalized by C_n. More precisely,

$$\mathcal{B}_{n\times n} = \{C_n\Lambda_n C_n^t \mid \Lambda_n \text{ is an } n\text{−by−}n \text{ real diagonal matrix}\}.$$

For an n-by-n matrix A_n, we choose our preconditioner $c(A_n)$ to be the minimizer of $||B_n - A_n||_F$ in the Frobenius norm in the space $\mathcal{B}_{n\times n}$. Following the terminology used in T. Chan, the approximation is called the *optimal cosine transform preconditioner* for A_n and denoted by $c(A_n)$. It can be shown that $c(A_n)$ is linear and preserves positive definiteness; see [19].

We will show in Appendix A that $c(A_n)$ can be obtained optimally in $O(n^2)$ operations for general matrices. The cost can be reduced to $O(n)$ operations when A_n is a banded matrix or a Toeplitz matrix. This is the same cost as that for constructing the optimal circulant preconditioners. We recall that in our case, L_{u^k} and H are banded matrix and Toeplitz matrix respectively. We summarize the construction cost of $c(A_n)$ in Table I. In Table I, we denote b_l and b_u the lower and upper band width of A_n.

B. Construction of Two-dimensional Preconditioner

For 2D $n \times n$ images, the matrices H^*H and L_u in (7) are block matrices of the following form:

$$A_{nn} = \begin{pmatrix} A_{1,1} & A_{1,2} & \dots & A_{1,n} \\ A_{2,1} & A_{2,2} & \dots & A_{2,n} \\ \vdots & \ddots & \ddots & \vdots \\ A_{n,1} & A_{n,2} & \dots & A_{n,n} \end{pmatrix}.$$

Here $A_{i,j}$ are square matrices of order n.

In [17], T. Chan and Olkin proposed the Level-1 and Level-2 circulant preconditioners for such block matrices. Following their approach, we define the Level-1 and Level-2 cosine transform preconditioners for A_{nn}. The idea of the Level-1 and Level-2 preconditioners is to approximate the matrix A_{nn} in one direction and two directions respectively. Specifically, the Level-1 preconditioner $c_1(A_{nn})$ is defined by

$$c_1(A_{nn}) = \begin{pmatrix} c(A_{1,1}) & c(A_{1,2}) & \dots & c(A_{1,n}) \\ c(A_{2,1}) & c(A_{2,2}) & \dots & c(A_{2,n}) \\ \vdots & \ddots & \ddots & \vdots \\ c(A_{n,1}) & c(A_{n,2}) & \dots & c(A_{n,n}) \end{pmatrix}.$$

To define the Level-2 cosine transform preconditioner, let us first give some notations. For any n^2-by-n^2 block matrix A_{nn}, we denote $(A_{nn})_{i,j;k,l}$ to be the (i,j)th entry of the (k,l)th block of A_{nn}. Let R be a permutation matrix which simply reorders A_{nn} in another coordinate direction. More precisely, R satisfies

$$(R^t A_{nn} R)_{i,j;k,l} = (A_{nn})_{k,l;i,j}, \quad 1 \le i,j \le n, \ \ 1 \le k,l \le n.$$

Then the Level-2 cosine transform preconditioner $c_2(A_{nn})$ for A_{nn} is defined by

$$c_2(A_{nn}) = Rc_1(R^t c_1(A_{nn})R)R^t.$$

It is easy to show that the preconditioner $c_2(A_{nn})$ can be diagonalized by $C_n \otimes C_n$:

$$c_2(A_{nn}) = (C_n \otimes C_n)\, \mathrm{diag}((C_n \otimes C_n)^t A_{nn}(C_n \otimes C_n))\, (C_n \otimes C_n)^t.$$

Hence, $c_2(A_{nn})$ can be inverted easily. Note that in the construction of our preconditioner, we will use the Level-2 approximation instead of the Level-1. It is because the Level-1 preconditioner has a relatively expensive initialization cost for block Toeplitz matrices with Toeplitz block, see [7].

For elliptic problem, it can be proved that the Level-2 optimal cosine transform preconditioner $c_2(A_{nn})$ is a "good" preconditioner.

Theorem 1: Let A_{nn} be the 5-point centered discretization matrix of

$$-(a(x,y)u_x)_x - (b(x,y)u_y)_y + \omega u = f(x,y) \quad \text{on} \quad [0,1]^2$$

with homogeneous Neumann boundary condition. Assume $\omega > 0$ and the mesh is uniform with size $1/n$. Then we have

$$\kappa(c_2(A_{nn})^{-1}A_{nn}) \le (\frac{c_{\max}}{c_{\min}})^2$$

where $0 < c_{\min} \le a(x,y), b(x,y) \le c_{\max}$.

Proof: We refer the reader to Appendix B. ■

Optimal cosine transform preconditioner can also be shown to be good for solving Toeplitz system. For one-dimensional cases, i.e. point-Toeplitz matrix systems, the convergence proof has been established in [20]. For 2-dimensional cases, we can prove the following result.

Theorem 2: Let A_{nn} be a block Toeplitz matrix with Toeplitz blocks. Assume the underlying generating sequence $a_k^{(j)}$ of A_{nn} is absolutely summable, i.e.,

$$\sum_{j=0}^{\infty}\sum_{k=0}^{\infty} |a_k^{(j)}| \le G < \infty$$

and

$$a_k^{(j)} = (A_{nn})_{(j-1)n+k,1} \qquad \text{for} \quad 1 \le j,k \le n.$$

If $\lambda_{\min}(A_{nn}) \ge \delta > 0$ where δ is independence on n, then for all $\epsilon > 0$, there exists $N > 0$ such that for all $n > N$, at most $O(n)$ of the eigenvalue values of the preconditioned matrix $c_2^{-1}(A_{nn})A_{nn}$ have absolute values larger than ϵ. Thus the PCG method converges in at most $O(n)$ steps.

Proof: The proof is similar to that of Corollary 1 in [7] and will be omitted. ■

We remark that the numerical results in [7] for the Level-2 optimal circulant preconditioner is better than the theoretical result and the numbers of iterations do not grow with n. With Theorems 1 and 2, it seems reasonable to use the Level-2 cosine approximation to construct preconditioners for the linear system (7).

III. Cosine Transform Preconditioner for TV denoising and deblurring

A straightforward preconditioner for A_{u^k} is

$$c_2(A_{u^k}) = c_2(H^*H + \alpha L_{u^k}) = c_2(H^*H) + \alpha c_2(L_{u^k}).$$

However, computing $c_2(H^*H)$ according to the formula in Corollary 1 in Appendix A, requires computing all the entries of H^*H and is costly. Another way is to approximate $c_2(H^*H)$ by $c_2(H)^*c_2(H)$. More precisely, a preconditioner for A_{u^k} in (7) can be defined as

$$M = c_2(H)^*c_2(H) + \alpha c_2(L_{u^k}). \tag{8}$$

One problem with the preconditioner M is that it does not capture the possibly large variation in the coefficient of the elliptic operator in (7) caused by the vanishing of $|\nabla u|$ in (4). To cure this problem, we apply the technique of diagonal scaling. More precisely, if we denote $\rho(\cdot)$ to be the spectral radius of a matrix and we define

$$\Delta \equiv \rho(H^*H)I + \alpha \ \mathrm{diag}(L_{u^k})$$

then we consider solving the equivalent system

$$\tilde{A}_{u^k}\tilde{u}^{k+1} = \tilde{H}^*z$$

where $\tilde{A}_{u^k} = \tilde{H}^*\tilde{H} + \alpha\tilde{L}_{u^k}$, $\tilde{H} = H\Delta^{-1/2}$, $\tilde{L}_{u^k} = \Delta^{-1/2}L_{u^k}\Delta^{-1/2}$ and $\tilde{u}^k = \Delta^{1/2}u^k$. In summary, the Level-2 cosine preconditioner with diagonal scaling is given by

$$M_D = \hat{H}^*\hat{H} + \alpha c_2(\tilde{L}_{u^k}) \tag{9}$$

where $\hat{H} = c_2(H)c_2(\Delta^{-1/2})$. We note further that if Λ_1, Λ_2 and Λ_3 are respectively the eigenvalue matrices of $c_2(H)$, $c_2(\Delta^{-1/2})$ and $c_2(\tilde{L}_{u^k})$, then the preconditioner can be expressed as

$$M_D = (C_n \otimes C_n)\ (\Lambda_1^*\Lambda_1\Lambda_2^*\Lambda_2 + \alpha\Lambda_3)\ (C_n \otimes C_n)^t.$$

Hence, the preconditioner can be easily invertible.

Finally, we comment on the cost of constructing M_D and of each PCG iteration. We note that L_{u^k} is a sparse matrix with only five nonzero bands. Also since H corresponds to a discretization of the convolution operator (1), H often is a block Toeplitz matrix with Toeplitz blocks. By using Table I, the construction cost of $c_2(H)$ and $c_2(L_{u^k})$ can be shown to be $O(n^2 \log n)$ operations; see [7], [17]. The cost of one PCG iteration is bounded by the cost of the matrix vector multiplication $\tilde{A}_{u^k}v = \Delta^{-1/2}(H^*H + \alpha L_{u^k})\Delta^{-1/2}v$ and the cost of solving the system $M_Dy = b$. The matrix vector multiplication $\Delta^{-1/2}v$ can be computed in $O(n^2)$ operations because $\Delta^{-1/2}$ is a diagonal matrix. Since L_{u^k} is banded, $L_{u^k}v$ can also be done in $O(n^2)$ operations. For H being a block Toeplitz matrix with Toeplitz blocks, Hv can be calculated in $O(n^2 \log n)$ operations; see [7]. Therefore, the matrix vector multiplication can be done in $O(n^2 \log n)$ operations. The system $M_Dy = b$ can be solved in $O(n^2 \log n)$ operations by exploiting the Fast Cosine Transform. Therefore, the total cost of each PCG iteration is bounded by of $O(n^2 \log n)$.

IV. The Product Preconditioner of Vogel and Oman

In this section, we introduce the product preconditioner proposed in Vogel and Oman in [28] and discuss some of its possible variants. The product preconditioner for the system (7) is defined as

$$\tilde{P} = \frac{1}{\gamma}(H^*H + \gamma I)^{1/2}(\alpha L_{u^k} + \gamma I)(H^*H + \gamma I)^{1/2}. \tag{10}$$

Here, γ is a parameter and will be chosen to be $\sqrt{\alpha}$ as suggested in [28]. Note that computing $\tilde{P}^{-1}v$ in each PCG iteration requires solving a convolution problem $(H^*H + \gamma I)^{-1/2}v$ and an elliptic problem $(\alpha L_{u^k} + \gamma I)^{-1}v$ which are not straightforward. To make the product preconditioner $\tilde{P}$ more practical, Vogel and Oman assumed that the blurring function h is periodic and hence $(H^*H + \gamma I)^{-1/2}v$ can be computed by the FFT. Moreover, when solving $(\alpha L_{u^k} + \gamma I)^{-1}v$, it requires an inner PCG iteration in which a multigrid preconditioner is used, see [28]. More precisely, there are three nested iterations in their method: the outermost FP iterations, the inner PCG iterations with preconditioner $\tilde{P}$ and the innermost PCG iterations to invert $(\alpha L_{u^k} + \gamma I)$.

Since we do not make any assumptions on h and it would not be fair to compare only the inner PCG iteration numbers and ignore the work required in the innermost PCG iterations (i.e. the work on solving $(\alpha L_{u^k} + \gamma I)^{-1}v$ which is substantial), this makes the comparison between the two preconditioners difficult. In the experiments below, we will instead use the following preconditioner for comparison:

$$\begin{aligned} P = \ & \frac{1}{\gamma}(c_2(H)^*c_2(H) + \gamma I)^{1/2} \cdot \\ & (\alpha c_2(L_{u^k}) + \gamma I) \cdot (c_2(H)^*c_2(H) + \gamma I)^{1/2}, \end{aligned} \tag{11}$$

where γ is again set to $\sqrt{\alpha}$. This preconditioner is obtained by taking cosine transform approximations of the three factors in $\tilde{P}$. The resulting preconditioner P can be diagonalized by the 2-D cosine transform and hence solving $P^{-1}v$ at each PCG step requires just about the same cost as our preconditioner M. Moreover, there is no need for the innermost PCG iterations.

We note that there is a connection between the product preconditioner P and the preconditioner M given in (8). First, note that the product preconditioner P in (11) can be viewed as an operator-splitted approximation of our preconditioner M in (8). Since the three factors in the right

hand side of (11) are commutative, we have

$$P - M = \frac{\alpha}{\gamma} c_2(H)^* c_2(H) c_2(L_{u^k}) + \gamma I. \qquad (12)$$

The right hand side matrix will be called the *splitting error* of P. If we again take $\gamma = \sqrt{\alpha}$, the splitting error becomes

$$\sqrt{\alpha}(c_2(H)^* c_2(H) c_2(L_{u^k}) + I).$$

Now, let us investigate the contribution from this splitting operator by using Fourier analysis. Denote $H(f)$ and $L(f)$ to be the spectrum of the blurring function h and the differential operator $\mathcal{L}_{u^k}$ respectively. Then the splitting error in (12) will have spectrum

$$\sqrt{\alpha}(|H(f)|^2 L(f) + 1), \qquad (13)$$

where f denotes the frequency variable.

In general, the second order differential operator L_{u^k} has $L(f)$ proportional to f^2. In fact L_{u^k}, being a high pass filter, has large eigenvalues corresponding to the large frequency modes and small eigenvalues corresponding to the low frequency modes (c.f. the case when L_{u^k} is the Laplacian operator). On the other hand, the blurring operator H being a low pass filter has the large eigenvalues corresponding to the low frequency modes and the small eigenvalues corresponding to the high frequency modes. Hence, we expect that the large eigenvalues of the differential operator will be damped by the small eigenvalues of the blurring operator when they are multiplied together as in (13). In particular, we expect $c_2(H)^* c_2(H) c_2(L_{u^k})$ to be a bounded operator (Note that the cosine transform approximation $c_2(\cdot)$ only changes the boundary condition of the operators, and in general will not change the ordering of the eigenvalues). Thus, for small α, the splitting error in (12) will be small and hence the performance of P and M should be about the same.

However, since $L(f)$ is proportional to f^2, when the spectrum of the blurring function $H(f)$ decays slowly, $L(f)$ may not be sufficiently damped. As a result, the splitting error in (12) will be large. Therefore, in this case, we expect M to outperform P for α's that are not very small. In particular, let us consider the limiting case, the denoising problem with $H(f) \equiv 1$ (i.e. $h(x, y)$ is the delta function). In this case,

$$M = \alpha c_2(L_{u^k}) + I,$$

$$P = \frac{1+\gamma}{\gamma}(\alpha c_2(L_{u^k}) + \gamma I) = (1 + \sqrt{\alpha})(\sqrt{\alpha} c_2(L_{u^k}) + I),$$

and hence

$$P - M = \sqrt{\alpha}(c_2(L_{u^k}) + I),$$

which is not small. We remark that even in this denoising case, $P \neq M$. The reason is due to the choice of $\gamma = \sqrt{\alpha}$. Of course, when $\gamma = 1$, the performance of the preconditioners P and M are exactly the same (since $P = 2M$). However, $\gamma = \sqrt{\alpha}$ is the best choice over a wide range of problems, see [28].

In conclusion, our preconditioner M is good and robust for solving deblurring problems and we expect it to outperform the product preconditioner P especially when the splitting error in (12) is large. That is the case when h is not a very significant blur or when h tends to the delta function (i.e., H tends to I). This will be demonstrated in Section V-B when we compare the performance of P and M for various blurring functions h and α's.

As in the case for the preconditioner M, we can apply a diagonal scaling to the matrix P to capture the possible large variation in the coefficient matrix. One possible way is to define it analogous to (9), i.e.

$$P_D = \frac{1}{\gamma}(\hat{H}^* \hat{H} + \gamma I)^{1/2}(\alpha c_2(\tilde{L}_{u^k}) + \gamma I)(\hat{H}^* \hat{H} + \gamma I)^{1/2}.$$

Similarly, we have

$$P_D - M_D = \frac{\alpha}{\gamma} \hat{H}^* \hat{H} c_2(\tilde{L}_{u^k}) + \gamma I,$$

which will be called the splitting error of P_D.

V. Numerical Results

In this section, we compare the numerical performance of different preconditioners in solving the linear system (7) for different images.

A. Test Problem 1

In this test, the original image is given by

$$u(x, y) = \chi_{[1/5,2/5]\times[1/4,3/4]} + \chi_{[3/5,4/5]\times[1/4,3/4]}$$

where $(x, y) \in [0, 1] \times [0, 1]$ and $\chi_{[a,b]}$ denotes the characteristics function for the interval $[a, b]$. The blurring function h in (1) is chosen to be a truncated Gaussian, given by

$$h(x, y) = \begin{cases} e^{-\tau(x^2+y^2)} & \text{if } |x|, |y| \leq 1/4 \\ 0 & \text{otherwise} \end{cases}. \qquad (14)$$

Here τ is a parameter which controls the severity of the blurring. More precisely, the smaller τ is, the more significant the blurring will be. In this experiment, we choose $\tau = 200$ so that h is a moderate blurring function. The noise function η has normal distribution and is scaled such that the noise-to-signal ratio (NSR), $\|\eta\|_{L^2}/\|\mathcal{H}u\|_{L^2}$, is 0.5. The true image and the observed image are shown in Figure 3.

We will perform the FP iterations, starting with $u^0 = z$, until the gradient g in (4) satisfies $\|g(u^k)\|_2 / \|g(u^0)\|_2 < 10^{-3}$. We will apply the CG method to solve the linear system (7) and the initial guess for the CG method in the kth FP iteration is chosen to be the $(k-1)$th FP iterate. The iteration is stopped when the residual vector r_k of the linear system (7) at the kth CG iteration satisfies $||r_k||_2/||r_0||_2 < 10^{-3}$. In our numerical experiment, we will focus on the performance of different choices of preconditioners for various of parameters n, α and β. Here n is the number of pixels in one direction, i.e. the matrix A_{u^k} is of size n^2-by-n^2.

Tables II and III show the number of iterations required for convergence of the FP iteration and the CG iteration for different choices of preconditioners and parameters. Note

that the CG iteration numbers shown in Tables II and III are the average number of CG iterations per FP step. The symbol "#" denotes the number of iterations for FP. The notations I, Δ denote respectively no preconditioner and the diagonal scaling preconditioner. Some of the data are plotted in Figures 1 and 2.

We observe from Figure 1 that the M_D and P_D preconditioners require significantly fewer iterations than other preconditioners for all values of β and α. Moreover, we can observe that the smaller β is, the more ill-conditioned the system is.

From Figure 2, we observe that the number of iterations corresponding to I grows like $O(n^{0.9})$, which from standard convergence theory for CG implies that $\kappa(A_{u^k}) \approx O(n^{1.8})$. If the preconditioner M_D or P_D is used, the number of iterations grows like $O(n^{0.22})$ which implies that $\kappa(M_D^{-1}A_{u^k})$ $\approx O(n^{0.44})$. However, the preconditioners Δ, M and P reduce the growth of the number of iterations only to $O(n^{0.67})$, $O(n^{0.58})$ and $O(n^{0.54})$ respectively. Therefore, M_D and P_D as preconditioners are much more effective than other three preconditioners.

From Table III, we observe that the performance of the preconditioners M and P (resp. M_D and P_D) are almost the same. This is in agreement with our observation in Section IV that for a smooth kernal function h (as in our case, the Gaussian function in (14)), the splitting error $P - M$ will be small for small α (e.g. $< 10^{-2}$) and hence we expect the performance of the preconditioners M and P to be close.

In Figure 4, we show the recovered images for various β. The smaller β is, the closer the recovered image is to the true image. Figure 5 shows how the recovered images depend on the value of α. In this case, the FP method produces the best image when $\alpha = 10^{-3}$.

B. Test Problem 2

We will now perform two experiments to illustrate that there are situations where our preconditioner M is better and more robust than the product preconditioner P. In this test problem, we will basically repeat the experiment in Section V-A with two different type of blurring functions.

In the first experiment, we will choose the Gaussian blurring function in (14) with several parameters τ, see Figure 6. We remark that the larger the τ is, the less significant is the blurring (see Figure 7), and we expect that our preconditioner to give a better performance than the product preconditioner P. The parameter n is fixed to 63, β to 0.1 and NSR=0.5. We report the number of PCG iterations required for various preconditioners and α's in Table IV.

We observe from Table IV that when $\tau = 200$, the performance of M and P are almost identical, as expected. When τ is increased to 2000, the number of PCG iterations required by P are much more than that by M for $\alpha \in [10^{-2}, 10^{-4}]$. As we mentioned in Section IV, when the blurring is not very significant (i.e., when h is close to the delta function), the splitting error in (12) becomes large unless α is very small (e.g. 10^{-5} when $\tau = 2000$). We remark that the performance of M_D and P_D is still close

	$n = 15$						
β	#	I	Δ	M	P	M_D	P_D
10^{-3}	93	44	26	43	52	24	27
10^{-2}	70	27	16	25	32	15	18
10^{-1}	49	16	11	15	20	10	12
10^{0}	27	10	8	9	13	7	10

	$n = 31$						
β	#	I	Δ	M	P	M_D	P_D
10^{-3}	112	108	42	72	77	23	21
10^{-2}	101	62	26	41	44	16	14
10^{-1}	20	36	18	24	25	11	10
10^{0}	52	21	13	13	14	8	8

	$n = 63$						
β	#	I	Δ	M	P	M_D	P_D
10^{-3}	266	197	59	112	114	25	24
10^{-2}	211	113	42	64	65	18	16
10^{-1}	164	64	29	35	36	13	11
10^{0}	115	37	21	18	18	10	8

	$n = 127$						
β	#	I	Δ	M	P	M_D	P_D
10^{-3}	420	> 400	99	169	169	32	29
10^{-2}	340	213	67	93	94	22	19
10^{-1}	257	118	47	49	50	15	13
10^{0}	177	69	34	25	25	9	8

TABLE II
$\alpha = 10^{-3}$

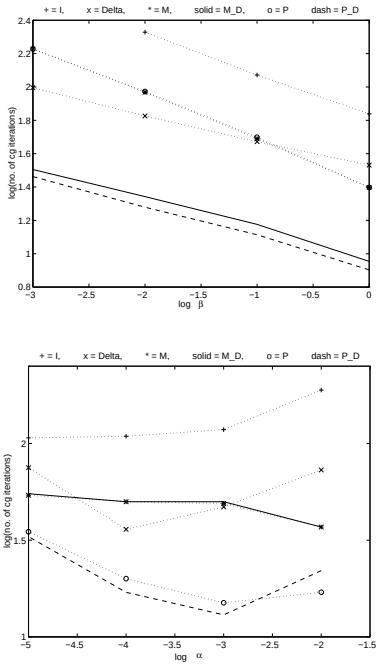

Fig. 1. Observation (top figure): the smaller the β is, the more ill-condition of A_{u^k}. Here $n = 127$ and $\alpha = 10^{-3}$. Observation (bottom figure): For various α, preconditioners M_D and P_D requires significantly less PCG iterations compared to the others preconditioners. Here $n = 127$ and $\beta = 0.1$.

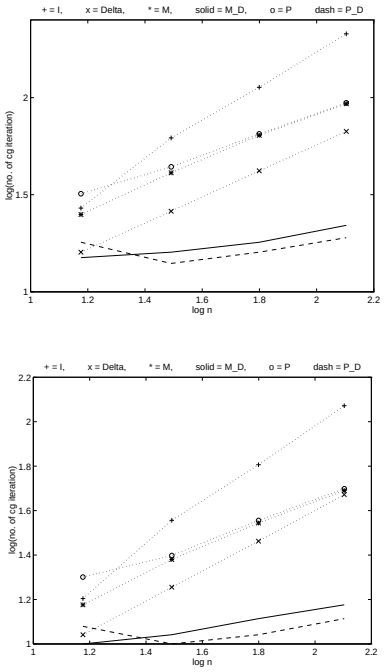

Fig. 2. We plot the no. of PCG iterations versus n in log scale for various preconditioners and β. Here $\alpha = 10^{-3}$ is fixed. Top : $\beta = 0.01$, slopes of I, Δ, M, P, M_D, P_D are 0.88, 0.67, 0.58, 0.54, 0.22, 0.22 respectively. Bottom: $\beta = 0.1$, slopes of I, Δ, M, P, M_D, P_D are 0.84, 0.68, 0.51, 0.49, 0.22, 0.19 respectively. Observation: Preconditioning by M_D or P_D require the smallest number of PCG iterations. The lines correspond to M_D and P_D have the smallest slope and therefore PCG has faster convergence rate.

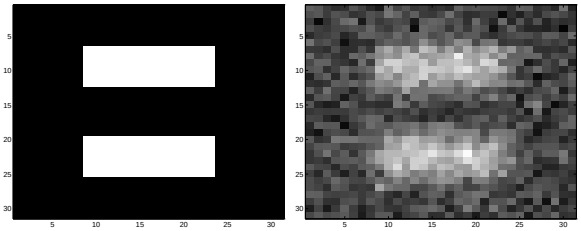

Fig. 3. True image (left) and the blurred noisy image, NSR=0.5 (right).

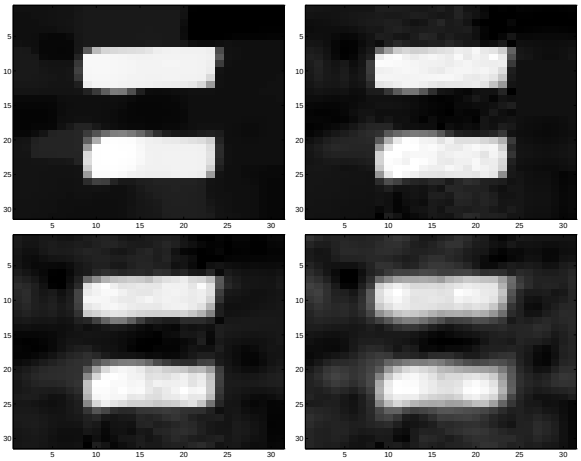

Fig. 4. Deblurred images for various β: 0.01 (left top), 0.1 (right top), 1 (bottom left), 10 (bottom right). In this experiment, $\alpha = 10^{-3}$, $n = 31$. Observation: the smaller the β is, the shaper the recovered image is .

	$n = 15$						
α	#	I	Δ	M	P	M_D	P_D
10^{-2}	36	25	19	14	20	10	10
10^{-3}	49	16	11	15	20	10	12
10^{-4}	32	37	36	21	22	19	22
10^{-5}	20	123	125	15	17	15	17

	$n = 31$						
α	#	I	Δ	M	P	M_D	P_D
10^{-2}	76	52	31	20	25	13	12
10^{-3}	80	36	18	24	25	11	10
10^{-4}	92	48	36	35	40	27	30
10^{-5}	94	183	183	78	79	74	76

	$n = 63$						
α	#	I	Δ	M	P	M_D	P_D
10^{-2}	137	98	48	29	30	16	17
10^{-3}	164	64	29	35	36	13	11
10^{-4}	104	64	38	35	36	21	20
10^{-5}	214	115	102	72	73	60	60

	$n = 127$						
α	#	I	Δ	M	P	M_D	P_D
10^{-2}	211	189	73	37	37	17	22
10^{-3}	257	118	47	49	50	15	13
10^{-4}	181	109	36	50	50	20	17
10^{-5}	136	107	75	54	55	35	33

TABLE III
$\beta = 0.1$

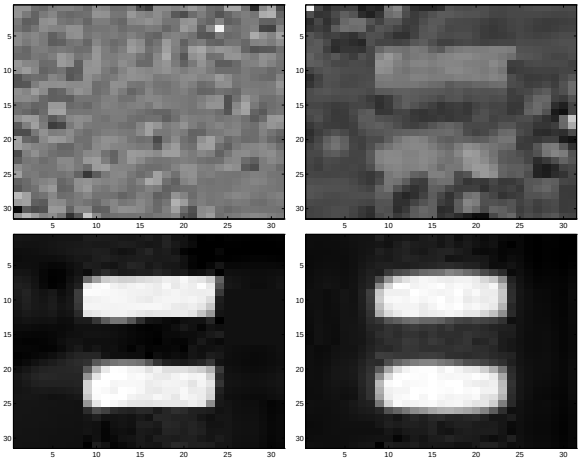

Fig. 5. Deblurred images for various α: 10^{-5} (top left), 10^{-4} (top right), 10^{-3} (bottom left), 10^{-2} (bottom right). In this experiment, $\beta = 0.1$, $n = 31$. Observation: when α is too small (e.g 10^{-4}), the recovered images do not get enough regularization and become irregular. When α is large (e.g. 10^{-2}), edges in the recovered images seem to be smooth out. In this case, the FP iteration produces the best image when $\alpha = 10^{-3}$.

	$\tau = 200$				
α	#	M	P	M_D	P_D
10^{-2}	137	29	30	16	17
10^{-3}	164	35	36	13	11
10^{-4}	104	35	36	21	20
10^{-5}	214	72	73	60	60

	$\tau = 2000$				
α	#	M	P	M_D	P_D
10^{-2}	111	31	42	18	18
10^{-3}	61	31	47	17	14
10^{-4}	136	46	55	29	30
10^{-5}	82	96	92	83	80

	$\tau = 20000$				
α	#	M	P	M_D	P_D
10^{-2}	25	23	37	17	16
10^{-3}	19	14	36	9	12
10^{-4}	6	5	22	4	10
10^{-5}	3	2	13	2	6

TABLE IV

GAUSSIAN BLUR, $n = 63$ AND $\beta = 0.1$

	$\tau = 0.1$				
α	#	M	P	M_D	P_D
10^{-2}	141	28	29	16	17
10^{-3}	146	36	36	13	12
10^{-4}	90	38	38	24	23
10^{-5}	90	118	110	100	99

	$\tau = 0.05$				
α	#	M	P	M_D	P_D
10^{-2}	126	32	35	18	19
10^{-3}	86	35	37	14	12
10^{-4}	101	56	66	34	34
10^{-5}	44	78	80	69	72

	$\tau = 0.01$				
α	#	M	P	M_D	P_D
10^{-2}	31	26	34	19	19
10^{-3}	31	18	35	13	15
10^{-4}	10	7	23	5	17
10^{-5}	5	3	14	3	13

TABLE V

OUT OF FOCUS BLUR, $n = 63$ AND $\beta = 0.1$

in this case. When we further increase τ to 20000, both preconditioners M and M_D outperform P and P_D for a wide range of α (e.g. $[10^{-2}, 10^{-5}]$).

In the second experiment, we choose the out of focus blurring functions. More precisely,

$$h(x,y) = \begin{cases} 1/(\pi\tau^2) & \sqrt{x^2+y^2} < \tau \\ 0 & elsewhere \end{cases} .$$

In this case, the smaller τ is, the less significant is the blurring, see Figures 9 and 10. From Table V, we can make the same observation as for the Gaussian blurring function. Namely, when the blurring is very significant (e.g. when $\tau = 0.1$), the performance of M and P are almost the same. However when the severity of blur decreases (e.g. $\tau = 0.05$ or 0.01), our preconditioners M and M_D perform better. In conclusion, the cosine transform preconditioners M and M_D are more robust than the product preconditioners P and P_D over a wide range of blurring functions.

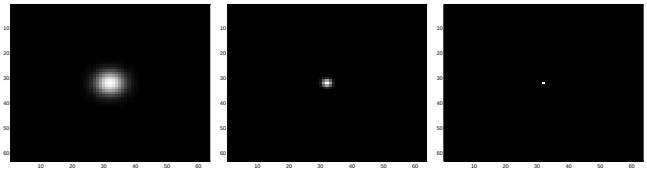

Fig. 6. Gaussian PSF's when $\tau = 200, 2000, 20000$ (left to right).

C. Test Problem 3

We consider a 2D image restoration problem arising in ground-based atmospheric imaging. In this test, we will compare the quality of the recovered images for several

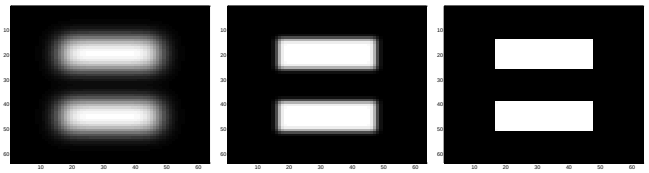

Fig. 7. Blurred images when $\tau = 200, 2000, 20000$ (left to right).

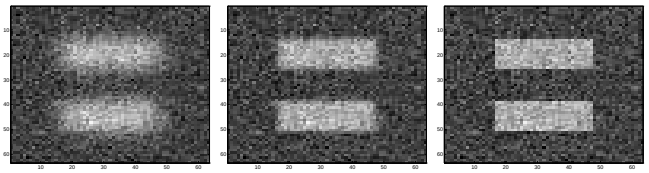

Fig. 8. Blurred and noisy images (observed images) when $\tau = 200, 2000, 20000$ (left to right) .

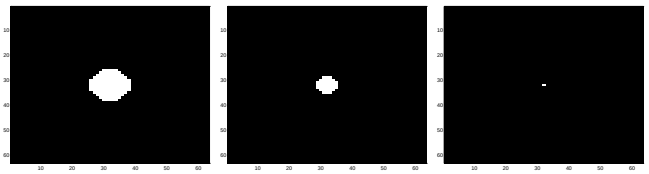

Fig. 9. Out of focus PSF's when $\tau = 0.1, 0.05, 0.01$ (left to right).

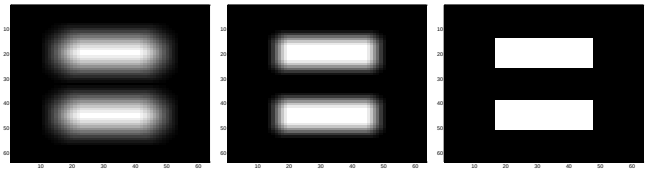

Fig. 10. Blurred images when $\tau = 0.1, 0.05, 0.01$ (left to right).

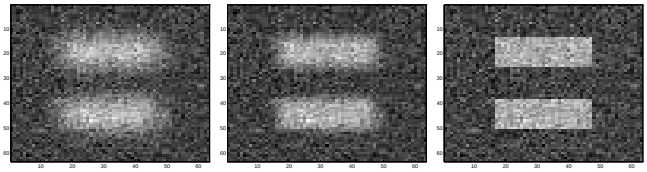

Fig. 11. Observed images when $\tau = 0.1, 0.05, 0.01$ (left to right).

numbers of FP iterations. The problem consists of a 256-by-256 image of an ocean reconnaissance satellite observed by a simulated ground-based imaging system together with a 256-by-256 image of a guide star observed under similar circumstances, see Figure 13. The data are provided by the Phillips Air Force Laboratory at Kirkland AFB, NM [4]. The imaging system detects the atmospheric distortions using a natural guide star image, see Figure 12 (right). A wavefront sensor measures the optical distortions which can then be digitized into a blurred image of the guide star pixel. We refer the reader to [13] on how to form the blurring matrix H. In Figures 14-17, we present restored images for various values of the parameter α after one, three and five FP iteration(s). Here, we fix $\beta = 0.1$ and use the M_D preconditioner when solving the linear system (7). We perform the PCG iterations until the relative residual less than 10^{-2}. The value $\|g(u)\|$ after the FP iterations are presented.

In Table VI, we compare the number of CG iterations with and without applying our preconditioner M_D for various of α's. We note that the preconditioner M_D can signficiantly speed up the convergence rate of the CG method. We observe from Figures 14-17 that after performing only 3 FP iterations, we obtain very good recovered images and most of the noise in the observed image are removed. Also, we note that when α is too large (e.g. $\alpha = 10^{-4}$), the recovered image looks "flat" and lost most of the features in the original image. When α got smaller (e.g. $\alpha = 10^{-5}$ or 10^{-6}), an antenna appears in the recovered images. We remark that the nonsmoothless of the images appearing in Figure 17 is due to insufficient regularization.

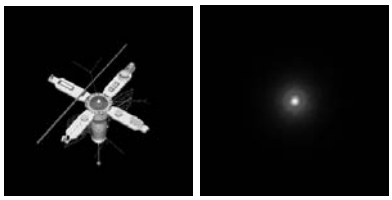

Fig. 12. Original image (left) and guide star image (right)

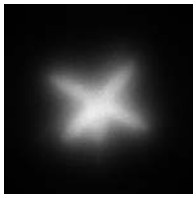

Fig. 13. Observed image

Fig. 14. Restored images for various FP iterations. Left: 1st FP with $\|g(u)\| = 1.1 \times 10^{-2}$. Middle: 3rd FP with $\|g(u)\| = 3.3 \times 10^{-3}$. Right: 5th FP with $\|g(u)\| = 1.6 \times 10^{-3}$. Here $\alpha = 10^{-4}$.

Fig. 15. Restored images for various FP iterations. Left: 1st FP with $\|g(u)\| = 1.4 \times 10^{-3}$. Middle: 3rd FP with $\|g(u)\| = 3.4 \times 10^{-4}$. Right: 5th FP with $\|g(u)\| = 1.7 \times 10^{-4}$. Here $\alpha = 10^{-5}$.

Fig. 16. Restored images for various FP iterations. Left: 1st FP with $\|g(u)\| = 2.7 \times 10^{-4}$. Middle: 3rd FP with $\|g(u)\| = 5.4 \times 10^{-5}$. Right: 5th FP with $\|g(u)\| = 2.9 \times 10^{-5}$. Here $\alpha = 10^{-6}$.

Fig. 17. Restored images for various FP iterations: Left: 1st FP with $\|g(u)\| = 1.9 \times 10^{-4}$. Middle: 3rd FP with $\|g(u)\| = 1.1 \times 10^{-5}$. Right: 5th FP with $\|g(u)\| = 6.8 \times 10^{-6}$. Here $\alpha = 10^{-7}$.

α	I	M_D
10^{-4}	176	19
10^{-5}	152	21
10^{-6}	105	30
10^{-7}	154	40

TABLE VI

Average number of PCG iterations in the first five FP steps. The column "I" denotes no preconditioning.

VI. Conclusion Remarks

In this paper, we propose cosine transform preconditioners for solving linear systems arising from total variation image deblurring problem. Our analysis and the numerical results indicate that the cosine transform preconditioner is an efficient and robust preconditioner over a wide class of image deblurring problems.

VII. Appendix A

In this appendix, we present how to construct $c(A_n)$ efficiently, as stated in Table I in Section II-A. The approach is basically the same as that for the sine transform preconditioner, see [14]; we present it here for the sake of clairty. An alternative derivation can be found in [20]. An essential idea is to make use of a sparse and structured basis for $\mathcal{B}_{n\times n}$, see [14].

Lemma 1: **(Boman and Koltracht [3])** Let Q_k, $k = 1,\ldots,n$, be n-by-n matrices with the (i,j)th entry given by

$$Q_k(i,j) = \begin{cases} 1 & \text{if } |i-j| = k-1, \\ 1 & \text{if } i+j = 2n-k+2, \\ 1 & \text{if } i+j = k, \\ 0 & \text{otherwise.} \end{cases}$$

Then $\{Q_k\}_{k=1}^n$ is a basis for $\mathcal{B}_{n\times n}$.

In other words, every matrix in $\mathcal{B}_{n\times n}$ can be expressed as a linear combination of the n matrices $\{Q_k\}_{k=1}^n$. We display the basis for the case $n = 6$.

$$Q_1 = \begin{pmatrix} 1&0&0&0&0&0\\ 0&1&0&0&0&0\\ 0&0&1&0&0&0\\ 0&0&0&1&0&0\\ 0&0&0&0&1&0\\ 0&0&0&0&0&1 \end{pmatrix}, \quad Q_2 = \begin{pmatrix} 1&1&0&0&0&0\\ 1&0&1&0&0&0\\ 0&1&0&1&0&0\\ 0&0&1&0&1&0\\ 0&0&0&1&0&1\\ 0&0&0&0&1&1 \end{pmatrix},$$

$$Q_3 = \begin{pmatrix} 0&1&1&0&0&0\\ 1&0&0&1&0&0\\ 1&0&0&0&1&0\\ 0&1&0&0&0&1\\ 0&0&1&0&0&1\\ 0&0&0&1&1&0 \end{pmatrix}, \quad Q_4 = \begin{pmatrix} 0&0&1&1&0&0\\ 0&1&0&0&1&0\\ 1&0&0&0&0&1\\ 1&0&0&0&0&1\\ 0&1&0&0&1&0\\ 0&0&1&1&0&0 \end{pmatrix},$$

$$Q_5 = \begin{pmatrix} 0&0&0&1&1&0\\ 0&0&1&0&0&1\\ 0&1&0&0&0&1\\ 1&0&0&0&1&0\\ 1&0&0&1&0&0\\ 0&1&1&0&0&0 \end{pmatrix}, \quad Q_6 = \begin{pmatrix} 0&0&0&0&1&1\\ 0&0&0&1&0&1\\ 0&0&1&0&1&0\\ 0&1&0&1&0&0\\ 1&0&1&0&0&0\\ 1&1&0&0&0&0 \end{pmatrix}. \tag{15}$$

In general, each Q_k has at most $2n$ non-zero entries.

In order to give a precise description of $\mathcal{B}_{n\times n}$, we introduce the following notations.

Definition 1: Let $w = (w_1,\ldots,w_n)^t$ be an n-vector. Define the shift of w as $\sigma(w) \equiv (w_2, w_3, \ldots, w_n, 0)^t$. Define $\mathcal{T}_n(w)$ to be the n-by-n symmetric Toeplitz matrix with w as the first column and $\mathcal{H}_n(w)$ to be the n-by-n Hankel matrix with w as the first column and $(w_n,\ldots,w_1)^t$ as the last column.

Lemma 2: $\mathcal{B}_{n\times n} = \{\mathcal{T}_n(w) + \mathcal{H}_n(\sigma(w)) \mid w = (w_1,\ldots,w_n)^t \in I\!R^n\}$.

Proof: From (15), we observe that every Q_i $(i = 1,\cdots,6)$ is a sum of a Toeplitz matrix and a Hankel matrix. In fact, by using Lemma 1, it is not difficult to prove that

$$Q_i = \mathcal{T}_n(e_i) + \mathcal{H}_n(\sigma(e_i)).$$

Here e_i denotes the ith unit vector in $I\!R^n$. Note that $\mathcal{T}_n(\cdot)$ and $\mathcal{H}_n(\sigma(\cdot))$ are linear operators. Therefore, an n-by-n matrix B_n belongs to $\mathcal{B}_{n\times n}$ if and only if there exist $w_1,\ldots,w_n \in I\!R$ such that

$$\begin{aligned} B_n = \sum_{j=1}^n w_j Q_j &= \sum_{j=1}^n w_j[\mathcal{T}_n(e_j) + \mathcal{H}_n(\sigma(e_j))] \\ &= \mathcal{T}_n(\sum_{j=1}^n w_j e_j) + \mathcal{H}_n(\sigma(\sum_{j=1}^n w_j e_j)) \\ &= \mathcal{T}_n(w) + \mathcal{H}_n(\sigma(w)) \end{aligned}$$

with $w = \sum_{j=1}^n w_j e_j$. ■

Now computing the optimal cosine transform approximation can be reformulated as solving the n-dimensional minimization problem,

$$\min_{w=(w_1,\ldots,w_n)\in I\!R^n} \|\mathcal{T}_n(w) + \mathcal{H}_n(\sigma(w)) - A_n\|_F.$$

The minimum can be calculated by setting

$$\frac{\partial}{\partial w_i}\|\mathcal{T}_n(w) + \mathcal{H}_n(\sigma(w)) - A_n\|_F^2 = 0,$$

for $i = 1,\ldots,n$. For the sake of presentation, let us illustrate the procedure of computing the minimum by considering the simple case $n = 6$. By the definition of the Frobenius norm, we express $\|\mathcal{T}_n(w) + \mathcal{H}_n(\sigma(w)) - A_n\|_F^2$ in terms of the matrix entries and then carry out the partial derivative with respect to w_i. Then, we see that w satisfies the following linear system

$$\begin{aligned} &\begin{pmatrix} 6&2&0&2&0&2\\ 2&12&4&0&4&0\\ 0&4&12&4&0&4\\ 2&0&4&12&4&0\\ 0&4&0&4&12&4\\ 2&0&4&0&4&12 \end{pmatrix} w \\ &= \begin{pmatrix} a_{11}+a_{22}+a_{33} \\ a_{12}+a_{23}+a_{34}+a_{45}+a_{56}+a_{21} \\ a_{13}+a_{24}+a_{35}+a_{46}+a_{31}+a_{42} \\ a_{14}+a_{25}+a_{36}+a_{41}+a_{52}+a_{63} \\ a_{15}+a_{26}+a_{51}+a_{62}+a_{14}+a_{23} \\ a_{16}+a_{61}+a_{15}+a_{24}+a_{33}+a_{42} \\ +a_{44}+a_{55}+a_{66} \\ +a_{32}+a_{43}+a_{54}+a_{65}+a_{11}+a_{66} \\ +a_{53}+a_{64}+a_{12}+a_{21}+a_{56}+a_{65} \\ +a_{13}+a_{22}+a_{31}+a_{46}+a_{55}+a_{64} \\ +a_{32}+a_{41}+a_{36}+a_{45}+a_{54}+a_{63} \\ +a_{51}+a_{26}+a_{35}+a_{44}+a_{53}+a_{62} \end{pmatrix}. \end{aligned} \tag{16}$$

We observe that the kth entry of the right hand side vector in (16) is obtained by adding those a_{ij} for which the (i,j)th position of Q_k is nonzero. For general n, let r_n be an n-vector with the k-th component given by

$$[r_n]_k = \sum_{(Q_k)_{i,j}\neq 0} a_{i,j}. \tag{17}$$

A_n	cost of constructing r_n
general	$O(n^2)$
Toeplitz	$O(n)$
banded	$O((b_l+b_u)n)$

TABLE VII
COST OF CONSTRUCTING r_n.

If A_n has no special structure, then clearly by (17), r_n can be computed in $O(n^2)$ operations because each Q_i has only $O(n)$ non-zero entries. If $A_n = [a_{i,j}]$ is a Toeplitz matrix (correspond to H in (7)), then the sum in (17) can be computed without explicit addition because summing $a_{i,j}$ for constant value of $|j-i|$ can be reduced to a scalar multiplication. Similarly, for banded matrix A_n with lower and upper band width b_l and b_u, the cost of forming r_n can be reduced to $O((b_l+b_u)n)$. We summarize the construction cost of r_n in Table VII.

We now go back to the solution of the linear system (16). We first reorder the unknowns w_i of w in such a way that the odd index entries and even index entries appear respectively in the upper half and lower half of the resulting vector. For simplicity, this leads to the following definition.

Definition 2: Let R_n be the n-by-n permutation matrix with the (i,j)th entry given by

$$[R_n]_{i,j} = \begin{cases} 1 & \text{if } 1 \le i \le \lceil \frac{n}{2} \rceil \text{ and } j = 2i-1, \\ 1 & \text{if } \lceil \frac{n}{2} \rceil < i \le n \text{ and } j = 2i - 2\lceil \frac{n}{2} \rceil, \\ 0 & \text{otherwise.} \end{cases}$$

After permutation, (16) becomes a block system,

$$\begin{pmatrix} 6 & 0 & 0 & 2 & 2 & 2 \\ 0 & 12 & 0 & 4 & 4 & 4 \\ 0 & 0 & 12 & 4 & 4 & 4 \\ 2 & 4 & 4 & 12 & 0 & 0 \\ 2 & 4 & 4 & 0 & 12 & 0 \\ 2 & 4 & 4 & 0 & 0 & 12 \end{pmatrix} \begin{pmatrix} w_1 \\ w_3 \\ w_5 \\ w_2 \\ w_4 \\ w_6 \end{pmatrix} = R_6 r_6.$$

The following theorem and corollary prove that in general if r_n is known in advance, then the block system can be solved in $O(n)$ operations.

Theorem 3: Let $A_n = [a_{i,j}]$ be an n-by-n matrix and $c(A_n)$ be the minimizer of $\|B_n - A_n\|_F$ over all $B_n \in \mathcal{B}_{n\times n}$. Denote $U_{i,j}$ to be the i-by-j matrix with all its entries being one and if i is equal to j, then the matrix is just denoted by U_i. Let e_1 to be the first unit vector of length $\lceil \frac{n}{2} \rceil$. Then

$$c(A_n) = \mathcal{T}_n(w) + \mathcal{H}_n(\sigma(w))$$

with

$$w = \tfrac{1}{2n^2} R_n^t \begin{pmatrix} 2nU_{\frac{n}{2}} + nI_{\frac{n}{2}} + ne_1e_1^t & -2nU_{\frac{n}{2}} \\ -2nU_{\frac{n}{2}} & 2(n-1)U_{\frac{n}{2}} + nI_{\frac{n}{2}} \end{pmatrix} R_n r_n \tag{18}$$

if n is even; and

$$w = \tfrac{1}{2n^2} R_n^t \begin{pmatrix} 2(n-1)U_{\frac{n+1}{2}} + nI_{\frac{n+1}{2}} + ne_1e_1^t & -2nU_{\frac{n+1}{2},\frac{n-1}{2}} \\ -2nU_{\frac{n-1}{2},\frac{n+1}{2}} & 2nU_{\frac{n-1}{2}} + I_{\frac{n-1}{2}} \end{pmatrix} R_n r_n$$

if n is odd.

Proof: Here we just give the proof for the case n is even. The proof for odd n is similar. To minimize $\|B_n - A_n\|_F^2$ over $\mathcal{B}_{n\times n}$, we set

$$\frac{\partial}{\partial w_i} \|c(A_n) - A_n\|_F^2 = 0, \qquad \text{for } i = 1, \ldots, n.$$

We obtain a linear system that has the same structure as that in (16). Permutating the system by R_n yields

$$\begin{pmatrix} nV & 2VU_{\frac{n}{2}} \\ 2U_{\frac{n}{2}}V^t & 2nI_{\frac{n}{2}} \end{pmatrix} R_n w = R_n r_n.$$

Here V is an $\frac{n}{2}$-by-$\frac{n}{2}$ matrix given by

$$V = 2I_{\frac{n}{2}} - e_1e_1^t.$$

By direct verification,

$$nV\left[2nU_{\frac{n}{2}} + nI_{\frac{n}{2}} + ne_1e_1^t\right] + 2VU_{\frac{n}{2}}\left[-2nU_{\frac{n}{2}}\right] = 2n^2 I_{\frac{n}{2}},$$

$$\left[2U_{\frac{n}{2}}V^t\right]\left[-2nU_{\frac{n}{2}}\right] + 2nI_{\frac{n}{2}}\left[2(n-1)U_{\frac{n}{2}} + nI_{\frac{n}{2}}\right] = 2n^2 I_{\frac{n}{2}},$$

$$nV\left[-2nU_{\frac{n}{2}}\right] + 2VU_{\frac{n}{2}}\left[2(n-1)U_{\frac{n}{2}} + nI_{\frac{n}{2}}\right] = 0,$$

and

$$2U_{\frac{n}{2}}V^t\left[2nU_{\frac{n}{2}} + nI_{\frac{n}{2}} + ne_1e_1^t\right] + 2nI_{\frac{n}{2}}\left[-2nU_{\frac{n}{2}}\right] = 0.$$

Therefore, we get

$$\begin{pmatrix} nV & 2VU_{\frac{n}{2}} \\ 2U_{\frac{n}{2}}V^t & 2nI_{\frac{n}{2}} \end{pmatrix} \begin{pmatrix} 2nU_{\frac{n}{2}} + nI_{\frac{n}{2}} + ne_1e_1^t & -2nU_{\frac{n}{2}} \\ -2nU_{\frac{n}{2}} & 2(n-1)U_{\frac{n}{2}} + nI_{\frac{n}{2}} \end{pmatrix} = 2n^2 I_n.$$

Combining together with the fact that R_n is orthogonal, (18) follows. ∎

Before going on, let us first emphasize the relationship between the first column of matrices $B \in \mathcal{B}_{n\times n}$ and their eigenvalues. For any matrix $B \in \mathcal{B}_{n\times n}$, we have $B = C_n \Lambda C_n^t$ where Λ is the eigenvalue matrix of B. If D denotes the diagonal matrix whose diagonal is equal to the first column of C_n^t and 1_n denotes the n-vector of all ones, then we have $C_n^t e_1 = D1_n$. Therefore, the relation is given by

$$D^{-1}C_n^t B e_1 = \Lambda 1_n.$$

In particular if the first column of $c(A_n)$ is known, we can obtain the eigenvalue matrix Λ of $c(A_n)$ in $O(n\log n)$ operations. Hence, the matrix-vector multiplication $c(A_n)^{-1}v$ can be computed as $v \longleftarrow C_n^t v$, $v \longleftarrow \Lambda^{-1}v$, $v \longleftarrow C_n v$ which costs $O(n\log n)$ operations. The following corollary gives the explicit formula for the entries of the first column

of $c(A_n)$. The proof follows directly from the expressions (18) and therefore we omit it.

Corollary 1: Let A_n be an n-by-n matrix and $c(A_n)$ be the minimizer of $\|B_n - A_n\|_F$ over all $B_n \in \mathcal{B}_{n\times n}$. Denote by q the first column of $c(A_n)$. If s_o and s_e are defined respectively to be the sum of the odd and even index entries of r_n, then we have, for n even,

$$\begin{aligned}
[q]_1 &= \frac{1}{2n^2}(2n[r_n]_1 + n[r_n]_2 - 2s_e) \\
[q]_i &= \frac{1}{2n^2}(n[r_n]_i + n[r_n]_{i+1} - 2s_e) \qquad i = 2,\ldots,n-1 \\
[q]_n &= \frac{1}{2n^2}(-2ns_o + (2n-2)s_e + n[r_n]_n)
\end{aligned}$$

and for n odd,

$$\begin{aligned}
[q]_1 &= \frac{1}{2n^2}(2n[r_n]_1 + n[r_n]_2 - 2s_o) \\
[q]_i &= \frac{1}{2n^2}(n[r_n]_i + n[r_n]_{i+1} - 2s_o) \qquad i = 2,\ldots,n-1 \\
[q]_n &= \frac{1}{2n^2}(-2ns_e + (2n-2)s_o + n[r_n]_n).
\end{aligned}$$

From Corollary 1 and Table VII, we see that $c(A_n)$ can be obtained in $O(n^2)$ operations for general matrix A_n and $O(n)$ operations for band matrix A_n and Toeplitz matrix A_n.

VIII. Appendix B: Proof of Theorem 1

Let A and A_L be the 5-point discretization matrices of $-(a(x,y)u_x)_x - (b(x,y)u_y)_y$ and the negation of the Laplacian respectively, both with homogeneous Neumann boundary condition. It follows immediately that $A_{nn} = A + \omega I$. Similar to [6, (4.2)], we can prove that

$$c_{\min} A_L \le A \le c_{\max} A_L.$$

By noting that A_L can be diagonalized by $C_n \otimes C_n$, we can prove similar to [15, equation (17)] that

$$c_{\min} A_L \le c_2(A) \le c_{\max} A_L.$$

Now for any $0 < \epsilon < \frac{\omega}{c_{\max}}$ and $x \in I\!R^{n^2}$, we have

$$\begin{aligned}
& x^t(A + c_{\min}\epsilon I)x + (\omega - c_{\min}\epsilon)x^t x \\
= \;& x^t A_{nn} x \\
= \;& x^t(A + c_{\max}\epsilon I)x + (\omega - c_{\max}\epsilon)x^t x
\end{aligned} \tag{19}$$

and

$$\begin{aligned}
& x^t(c_2(A) + c_{\max}\epsilon I)x + (\omega - c_{\max}\epsilon)x^t x \\
= \;& x^t c_2(A_{nn}) x \\
= \;& x^t(c_2(A) + c_{\min}\epsilon I)x + (\omega - c_{\min}\epsilon)x^t x.
\end{aligned} \tag{20}$$

Since the inequality

$$\min\left\{\frac{a}{c}, \frac{b}{d}\right\} \le \frac{a+b}{c+d} \le \max\left\{\frac{a}{c}, \frac{b}{d}\right\}$$

holds for any $a, b, c, d > 0$ and the matrices $A + c_{\min}\epsilon I$, $A + c_{\max}\epsilon I$, $c_2(A) + c_{\max}\epsilon I$, $c_2(A) + c_{\min}\epsilon I$, $(\omega - c_{\min}\epsilon)I$ and $(\omega - c_{\max}\epsilon)I$ are positive definite, using (19) and (20) we have

$$\begin{aligned}
\frac{x^t A_{nn} x}{x^t c_2(A_{nn}) x} &\le \max\left\{\frac{x^t(A + c_{\max}\epsilon I)x}{x^t(c_2(A) + c_{\min}\epsilon I)x}, \frac{\omega - c_{\max}\epsilon}{\omega - c_{\min}\epsilon}\right\} \\
&\le \max\left\{\frac{c_{\max} x^t(A_L + \epsilon I)x}{c_{\min} x^t(A_L + \epsilon I)x}, \frac{\omega - c_{\max}\epsilon}{\omega - c_{\min}\epsilon}\right\} \\
&= \max\left\{\frac{c_{\max}}{c_{\min}}, \frac{\omega - c_{\max}\epsilon}{\omega - c_{\min}\epsilon}\right\} \\
&\longrightarrow \frac{c_{\max}}{c_{\min}} \quad \text{as} \quad \epsilon \longrightarrow 0
\end{aligned}$$

and

$$\begin{aligned}
\frac{x^t A_{nn} x}{x^t c_2(A_{nn}) x} &\ge \min\left\{\frac{x^t(A + c_{\min}\epsilon I)x}{x^t(c_2(A) + c_{\max}\epsilon I)x}, \frac{\omega - c_{\min}\epsilon}{\omega - c_{\max}\epsilon}\right\} \\
&\ge \min\left\{\frac{c_{\min} x^t(A_L + \epsilon I)x}{c_{\max} x^t(A_L + \epsilon I)x}, \frac{\omega - c_{\min}\epsilon}{\omega - c_{\max}\epsilon}\right\} \\
&= \min\left\{\frac{c_{\min}}{c_{\max}}, \frac{\omega - c_{\min}\epsilon}{\omega - c_{\max}\epsilon}\right\} \\
&\longrightarrow \frac{c_{\min}}{c_{\max}} \quad \text{as} \quad \epsilon \longrightarrow 0.
\end{aligned}$$

Hence, Theorem 1 is proved.

References

[1] R. Acar and C. Vogel, *Analysis of Bounded Variation Penalty Methods*, Inverse Problems, 10 (1994), pp. 1217–1229.

[2] L. Alvarez and J. Morel, *Formalization and Computational Aspects of Image Analysis*, Acta Numerica (1994), pp. 1–59.

[3] E. Boman and I. Koltracht, *Fast Transform Based Preconditioners for Toeplitz Equations*, SIAM J. Matrix Anal. Appl., 16, 628–645, 1995.

[4] R. Carreras, *Personal Correspondence*, 1993.

[5] R. Chan, *The Spectrum of a Family of Circulant Preconditioned Toeplitz Systems*, SIAM J. Numer. Anal., 26 (1989), pp. 503–506.

[6] R. Chan and T. Chan, *Circulant Preconditioners for Elliptic Problems* J. Numer. Linear Algebra Appls., 1 (1992), pp. 77–101.

[7] R. Chan and X. Jin, *A Family of Block Preconditioners for Block Systems*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 1218–1235.

[8] R. Chan and M. Ng, *Conjugate Gradient Methods for Toeplitz Systems*, SIAM Review, 38 (1996), pp. 427-482.

[9] R. Chan, T. Chan and W. Wan, *Multigrid for Differential-Convolution Problems Arising from Image Processing,* Proceedings of the Workshop on Scientific Computing, Hong Kong, 97, Springer-Verlag, Ed: G. Golub, S.H. Lui, F. Luk and R. Plemmons.

[10] R. Chan, T.F. Chan, and C.K. Wong, *Cosine Transform Based Preconditioners for Total Variation Minimization Problems in Image Processing*, Iterative Methods in Linear Algebra, II, V3, IMACS Series in Computational and Applied Mathematics, Proceedings of the Second IMACS International Symposium on Iterative Methods in Linear Algebra, Bulgaria, June, 1995, pp. 311–329, Ed: S. Margenov and P. Vassilevski.

[11] R. Chan, W. Ching and C. Wong, *Optimal Trigonometric Preconditioners for Elliptic and Queueing Problems*, SEA Bull. Math., 20 (1996), pp. 117–124.

[12] R. Chan, J. Nagy and R. Plemmons, *FFT-Based Preconditioners for Toeplitz-Block Least Squares Problems*, SIAM J. Numer. Anal., 30 (1993), pp. 1740–1768.

[13] R. Chan, M. Ng and R. Plemmons, *Generalization of Strang's preconditioner with applications to iterative deconvolution*, Numer. Linear Algebra Appls., 3 (1996), pp. 45–64.

[14] R. Chan, M. Ng, and C. Wong, *Sine Transform Based Preconditioners for Symmetric Toeplitz Systems*, Linear Algebra Appls., 232 (1996), 237-260.

[15] R. Chan and C. Wong, *Sine Transform Based Preconditioners for Elliptic Problems*, accepted for publication by Numerical Linear Algebra and Applications.

[16] T. Chan, *An Optimal Circulant Preconditioner for Toeplitz Systems*, SIAM J. Sci. Stat. Comput., 9 (1988), pp. 766–771.

[17] T. Chan and J. Olkin, *Circulant Preconditioners for Toeplitz-block Matrices*, Numer. Algorithms, 6 (1994) pp. 89–101.
[18] G. Golub and C. Van Loan, *Matrix Computations* , 2nd Ed., The Johns Hopkins University Press, Maryland, 1989.
[19] T. Huckle, *Circulant/skewcirculant Matrices for Solving Toeplitz Matrix Problems*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 767 – 777.
[20] T. Kailath and V. Olshevsky, *Displacement structure approach to discrete-trigonometric-transform based preconditioners of G. Strang and T. Chan types*, appears in the Italian Journal Calcolo, devoted to the Proceedings of the International Workshop on "Toeplitz matrices, algorithms and applications", Cortona, Italy, September 1996.
[21] I. Ito and K. Kunisch, *An Active Set Strategy for Image Restoration Based on the Augmented Lagrangian Formulation*, Preprint, Center for Research in Scientific Computation, North Carolina State University.
[22] Y. Li and F. Santosa, *An Affine Scaling Algorithm for Minimizing Total Variation in Image Enhancement*, Technical Report CTC94TR201, Cornell Theory Center, Cornell University.
[23] T. Manteuffel and S. Parter, *Preconditioning and Boundary Conditions*, SIAM J. Numer. Anal., 27 (1990), pp. 656–694.
[24] L. Rudin, S. Osher, *Total Variation Based Image Restoration with Free Local Constraints*, IEEE International Conference on Image Processing, Austin, TX, 1994.
[25] L. Rudin, S. Osher and E. Fatemi, *Nonlinear Total Variation Based Noise Removal Algorithms*, Physica D., 60 (1992), pp. 259–268.
[26] H. Sorensen and C. Burrus, *Fast DFT and Convolution Algorithms*, Handbook of Signal Processing edited by S. Mitra and J. Kaiser, New York, Wiley.
[27] C. Vogel and M. Oman, *Iterative Methods for Total Variation Denoising*, SIAM J. Sci. Stat. Comput., 17 (1996) pp. 227–238.
[28] C. Vogel and M. Oman, *Fast Total Variation-Based Image Reconstruction*, in the Proceedings of the 1995 ASME Design Engineering Conferences, V3, part C, pp. 1009-1015.
[29] C. Wong, *Block Toeplitz Type Preconditioners for Elliptic Problem*, M.Phil. thesis, The University of Hong Kong, 1994.
[30] P. Yip and K. Rao, *Fast Decimation-in-time Algorithms for a Family of Discrete Sine and Cosine Transforms*, Circuits, Syst., Signal Process, 3 (1984), pp. 387–408.