Journal of Computational and Applied Mathematics 123 (2000) 85–100

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

www.elsevier.nl/locate/cam

The ubiquitous Kronecker product

Charles F. Van Loan [1]

Department of Computer Science, Cornell University, Ithaca, New York 14853, USA

Received 26 October 1999

Abstract

The Kronecker product has a rich and very pleasing algebra that supports a wide range of fast, elegant, and practical algorithms. Several trends in scientific computing suggest that this important matrix operation will have an increasingly greater role to play in the future. First, the application areas where Kronecker products abound are all thriving. These include signal processing, image processing, semidefinite programming, and quantum computing. Second, sparse factorizations and Kronecker products are proving to be a very effective way to look at fast linear transforms. Researchers have taken the Kronecker methodology as developed for the fast Fourier transform and used it to build exciting alternatives. Third, as computers get more powerful, researchers are more willing to entertain problems of high dimension and this leads to Kronecker products whenever low-dimension techniques are "tensored" together.

1. Basic properties

If $B \in \mathbb{R}^{m_1 \times n_1}$ and $C \in \mathbb{R}^{m_2 \times n_2}$, then their *Kronecker product* $B \otimes C$ is an $m_1 \times n_1$ block matrix whose (i,j) block is the $m_2 \times n_2$ matrix $b_{ij}C$. Thus,

$$\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \otimes \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} = \left[\begin{array}{ccc|ccc} b_{11}c_{11} & b_{11}c_{12} & b_{11}c_{13} & b_{12}c_{11} & b_{12}c_{12} & b_{12}c_{13} \\ b_{11}c_{21} & b_{11}c_{22} & b_{11}c_{23} & b_{12}c_{21} & b_{12}c_{22} & b_{12}c_{23} \\ b_{11}c_{31} & b_{11}c_{32} & b_{11}c_{33} & b_{12}c_{31} & b_{12}c_{32} & b_{12}c_{33} \\ \hline b_{21}c_{11} & b_{21}c_{12} & b_{21}c_{13} & b_{22}c_{11} & b_{22}c_{12} & b_{22}c_{13} \\ b_{21}c_{21} & b_{21}c_{22} & b_{21}c_{23} & b_{22}c_{21} & b_{22}c_{22} & b_{22}c_{23} \\ b_{21}c_{31} & b_{21}c_{32} & b_{21}c_{33} & b_{22}c_{31} & b_{22}c_{32} & b_{22}c_{33} \end{array}\right].$$

The basic properties of the Kronecker product are quite predictable:

$$(B \otimes C)^{\mathrm{T}} = B^{\mathrm{T}} \otimes C^{\mathrm{T}},$$

$$(B \otimes C)^{-1} = B^{-1} \otimes C^{-1},$$

[1] Supported in part by the NSF contract CCR-9901988.

E-mail address: cv@cs.cornell.edu (C.F. Van Loan).

0377-0427/00/\$ - see front matter
PII: S0377-0427(00)00393-9

$$(B \otimes C)(D \otimes F) = BD \otimes CF,$$

$$B \otimes (C \otimes D) = (B \otimes C) \otimes D.$$

Of course, the indicated products and inverses must exist for the second and third identities to hold.

The entries of $B \otimes C$ and $C \otimes B$ consist of all possible products of a B-matrix entry with a C-matrix entry and this raises the possibility that these two Kronecker products are related by a permutation. The permutation involved is in fact the *perfect shuffle*. If p and q are positive integers and $r = pq$, then the (p, q) perfect shuffle is the $r \times r$ matrix

$$S_{p,q} = \begin{bmatrix} I_r(1:q:r,:) \\ I_r(2:q:r,:) \\ \vdots \\ I_r(q:q:r,:) \end{bmatrix} \tag{1}$$

where I_r is the $r \times r$ identity. (The well-known "colon notation" used in MATLAB to designate submatrices is being used here.) In effect, the matrix–vector product $S_{p,q}x$ takes the "card deck" x, splits it into p piles of length-q each, and then takes one card from each pile in turn until the deck is reassembled. It can be shown that if $B \in \mathbb{R}^{m_1 \times n_1}$ and $C \in \mathbb{R}^{m_2 \times n_2}$, then

$$C \otimes B = S_{m_1,m_2}(B \otimes C)S^{\mathrm{T}}_{n_1,n_2}.$$

Henderson and Searle [36] survey the numerous connections between the Kronecker product and the perfect shuffle. Additional observations about the Kronecker product may be found in [30,14,37,63]. Henderson et al. [35] look at the operation from the historical point of view.

Particularly important in computational work are the issues that surround the exploitation of structure and the application of matrix factorizations. By and large, a Kronecker product inherits structure from its factors. For example,

$$\text{if } B \text{ and } C \text{ are} \left\{ \begin{array}{l} \text{nonsingular} \\ \text{lower(upper) triangular} \\ \text{banded} \\ \text{symmetric} \\ \text{positive definite} \\ \text{stochastic} \\ \text{Toeplitz} \\ \text{permutations} \\ \text{orthogonal} \end{array} \right\}, \text{ then } B \otimes C \text{ is} \left\{ \begin{array}{l} \text{nonsingular} \\ \text{lower(upper) triangular} \\ \text{block banded} \\ \text{symmetric} \\ \text{positive definite} \\ \text{stochastic} \\ \text{block Toeplitz} \\ \text{a permutation} \\ \text{orthogonal} \end{array} \right\}.$$

With respect to factorizations, the LU-with-partial-pivoting, Cholesky, and QR factorizations of $B \otimes C$ merely require the corresponding factorizations of B and C:

$$B \otimes C = (P_B^{\mathrm{T}} L_B U_A) \otimes (P_C^{\mathrm{T}} L_C U_C) = (P_B \otimes P_C)^{\mathrm{T}} (L_B \otimes L_C)(U_B \otimes U_C),$$
$$B \otimes C = (G_B G_B^{\mathrm{T}}) \otimes (G_C G_C^{\mathrm{T}}) = (G_B \otimes G_C)(G_B \otimes G_C)^{\mathrm{T}},$$
$$B \otimes C = (Q_B R_B) \otimes (Q_C R_C) = (Q_B \otimes Q_C)(R_B \otimes R_C).$$

The same is true for the singular value and Schur decompositions if we disregard ordering issues. In contrast, the CS and QR-with-column pivoting factorizations of $B \otimes C$ do not have simple relationships to the corresponding factorizations of B and C. (The matrix factorizations and decompositions mentioned in this paper are all described in [29]).

In Kronecker product work, matrices are sometimes regarded as vectors and vectors are sometimes "made into" into matrices. To be precise about these reshapings we use the vec operation. If $X \in \mathbb{R}^{m\times n}$, then vec$(X)$ is an $nm \times 1$ vector obtained by "stacking" X's columns. If C, X, and B are matrices and the product CXB^T is defined, then it is not hard to establish the following equivalence:

$$Y = CXB^T \equiv y = (B \otimes C)x \tag{2}$$

where $x = \text{vec}(X)$ and $y = \text{vec}(Y)$. Henderson and Searle [36] thoroughly discuss the vec/Kronecker product connection.

A consequence of the above properties is that linear systems of the form $(B \otimes C)x = f$ can be solved *fast*. For example, if $B, C \in \mathbb{R}^{m\times m}$, then x can be obtained in $O(m^3)$ flops via the *LU* factorizations of B and C. Without the exploitation of structure, an $m^2 \times m^2$ system would normally require $O(m^6)$ flops to solve.

As with any important mathematical operation, the Kronecker product has been specialized and modified to address new and interesting applications. Rauhala [58] presents a theory of "array algebra" that applies to certain photogrammetric problems. See also [62]. Regalia and Mitra [60] have used the Kronecker product and various generalizations of it to describe a range of fast unitary transforms. A sample generalization that figures in their presentation is

$$\{A_1, \ldots, A_m\} \text{``}\otimes\text{''}\ B = \begin{bmatrix} A_1 \otimes B(1,:) \\ A_2 \otimes B(2,:) \\ \vdots \\ A_m \otimes B(m,:) \end{bmatrix}$$

where $A_1, \ldots, A_m$ are given matrices of the same size and B has m rows.

Another generalization, the *strong Kronecker product*, is developed in [61] and supports the analysis of certain orthogonal matrix multiplication problems. The strong Kronecker product of an $m \times p$ block matrix $B = (B_{ij})$ and a $p \times n$ block matrix $C = (C_{ij})$ is an $m \times n$ block matrix $A = (A_{ij})$ where $A_{ij} = B_{i1} \otimes C_{1j} + \cdots + B_{ip} \otimes C_{pj}$.

Kronecker product problems arise in photogrammetry [59], image processing [34], computer vision [47], and system theory [6]. They surface in the analysis of generalized spectra [2], stochastic models [57], and operator theory [64]. They have even found their way into the analysis of chess endgames [65].

To make sense of the "spread" of the Kronecker product, we have organized this paper around a few important families of applications. These include the linear matrix equation problem, fast linear transforms, various optimization problems, and the idea of preconditioning with Kronecker products.

2. Matrix equations

To researchers in numerical linear algebra, the most familiar problem where Kronecker products arise is the *Sylvester* matrix equation problem. Here we are given $F \in \mathbb{R}^{m\times m}$, $G \in \mathbb{R}^{n\times n}$, and $C \in \mathbb{R}^{m\times n}$ and seek $X \in \mathbb{R}^{m\times n}$ so that $FX + XG^T = C$. Linear systems of this variety play a central role in control theory [13], Poisson equation solving [18], and invariant subspace computation [29]. In light of Eq. (2), the act of finding X is equivalent to solving the $mn \times mn$ linear system

$$(I_n \otimes F + G \otimes I_m)\ \mathrm{vec}(X) = \mathrm{vec}(C).$$

The *Lyapunov* problem results if $F = G$, a very important special case.

One family of effective methods for these problems involve reducing F and G to Hessenberg or triangular form via orthogonal similarity transformations [5,27].

The more general matrix equation $F_1XG_1^{\mathrm{T}} + F_2XG_2^{\mathrm{T}} = C$ can be handled using the generalized Schur decomposition as discussed by Gardiner et al. [24]. However, these factorization approaches break down for the general Sylvester matrix equation problem [49,74].

$$F_1XG_1^{\mathrm{T}} + \cdots + F_pXG_p^{\mathrm{T}} = C \equiv (G_1 \otimes F_1 + \cdots + G_p \otimes F_p)\mathrm{vec}(X) = \mathrm{vec}(C). \tag{3}$$

Related to this are linear systems where the matrix of coefficients has blocks that are themselves Kronecker products, e.g.,

$$\begin{bmatrix} F_{11} \otimes G_{11} & \cdots & F_{1p} \otimes G_{1p} \\ \vdots & \ddots & \vdots \\ F_{p1} \otimes G_{p1} & \cdots & F_{pp} \otimes G_{pp} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_p \end{bmatrix} = \begin{bmatrix} c_1 \\ \vdots \\ c_p \end{bmatrix}. \tag{4}$$

(Clearly the dimensions of the matrices F_{ij} and G_{ij} have to "make sense" when compared to the dimensions of the vectors x_i and c_i.) This is equivalent to a system of generalized Sylvester equations:

$$\sum_{j=1}^{p} F_{ij}X_jG_{ij}^{\mathrm{T}} = C_i, \quad i = 1:p$$

where $\mathrm{vec}(X_i) = x_i$ and $\mathrm{vec}(C_i) = c_i$ for $i = 1:p$. We can solve a linear system $Ax = b$ fast if A is a Kronecker product. But fast solutions seem problematical for (4).

A problem of this variety arises in conjunction with the generalized eigenproblem $M - \lambda N$ where important subspace calculations require the solution of a system of the form

$$\begin{bmatrix} I_n \otimes A & -B^{\mathrm{T}} \otimes I_m \\ I_n \otimes D & -E^{\mathrm{T}} \otimes I_m \end{bmatrix} \begin{bmatrix} \mathrm{vec}(R) \\ \mathrm{vec}(L) \end{bmatrix} = \begin{bmatrix} \mathrm{vec}(C) \\ \mathrm{vec}(F) \end{bmatrix}.$$

Here the matrices $A, D \in \mathbb{R}^{m\times m}$, $B, E \in \mathbb{R}^{n\times n}$, and $C, F \in \mathbb{R}^{m\times n}$ are given and the matrices $L, R \in \mathbb{R}^{m\times n}$ are sought [42].

Another area where block systems arise with Kronecker product blocks is semidefinite programming. There has been an explosion of interest in this area during the last few years due largely to the applicability of interior point methods; see [69,71]. An important feature of these methods is that they frequently require the solution of linear systems that involve the symmetric Kronecker product $\circledast$. For symmetric $X \in \mathbb{R}^{n\times n}$ and arbitrary $B, C \in \mathbb{R}^{n\times n}$ this operation is defined by

$$(B \circledast C)\mathrm{svec}(X) = \mathrm{svec}\left(\frac{1}{2}(CXB^{\mathrm{T}} + BXC^{\mathrm{T}})\right)$$

where the "svec" operation is a normalized stacking of X's subdiagonal columns, e.g.,

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} \Rightarrow \mathrm{svec}(X) = [x_{11}, \sqrt{2}x_{21}, \sqrt{2}x_{31}, x_{22}, \sqrt{2}x_{32}\ x_{33}]^{\mathrm{T}};$$

See the work of Alizadeh et al. Among other things they discuss the efficient solution of systems of the form

$$\begin{bmatrix} 0 & A^{\mathrm{T}} & I \\ A & 0 & 0 \\ Z \otimes I & 0 & X \otimes I \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} = \begin{bmatrix} r_d \\ r_p \\ r_c \end{bmatrix}.$$

3. Least squares

Least squares problems of the form

$$\min \|(B \otimes C)x - b\|$$

can be efficiently solved by computing the QR factorizations (or SVDs) of B and C; [21,22]. Barrlund [4] shows how to minimize $\|(A_1 \otimes A_2)x - f\|$ subject to the constraint that $(B_1 \otimes B_2)x = g$, a problem that comes up in surface fitting with certain kinds of splines.

Coleman et al. [11] describe an interesting least-squares problem that arises in segmentation analysis. It is the minimization of

$$\left\| W \left(\begin{bmatrix} I_n \otimes D_m \\ D_n \otimes I_m \\ \lambda I \end{bmatrix} x - \begin{bmatrix} b_1 \\ b_2 \\ 0 \end{bmatrix} \right) \right\|_2 \tag{5}$$

where W is diagonal, and the D matrices are upper bidiagonal. The scaling matrix W and the λI block seem to rule out obvious fast factorization approaches.

On the other hand, LS problems of the form

$$\min \left\| \begin{bmatrix} B_1 \otimes C_1 \\ B_2 \otimes C_2 \end{bmatrix} x - b \right\|_2 \tag{6}$$

can be solved fast by computing the generalized singular value decomposition of the pairs (B_1, B_2) and (C_1, C_2):

$$\begin{aligned} B_1 &= U_{1B} D_{1B} X_B^{\mathrm{T}} \qquad & B_2 &= U_{2B} D_{2B} X_B^{\mathrm{T}}, \\ C_1 &= U_{1C} D_{1C} X_C^{\mathrm{T}} \qquad & C_2 &= U_{2C} D_{2C} X_C^{\mathrm{T}}. \end{aligned}$$

Here, the U's are orthogonal, the D's are diagonal, and the X's are nonsingular. With these decompositions the matrices in (6) can be transformed to diagonal form since

$$\begin{bmatrix} B_1 \otimes C_1 \\ B_2 \otimes C_2 \end{bmatrix} = \begin{bmatrix} U_{1B} \otimes U_{2B} & 0 \\ 0 & U_{1C} \otimes U_{2C} \end{bmatrix} \begin{bmatrix} D_{1B} \otimes D_{2B} \\ D_{1C} \otimes D_{2C} \end{bmatrix} X_B^{\mathrm{T}} \otimes X_C^{\mathrm{T}}.$$

The solution of the converted problem is straightforward and the overall cost of the procedure is essentially the cost of the two generalized SVDs.

The total least squares (TLS) problem is another example of just how little it takes to stifle the easy exploitation Kronecker products in a matrix computation. A TLS solution to $(B \otimes C)x \approx b$ requires the computation of the smallest singular value and the associated left and right singular vectors of the augmented matrix

$$M = [B \otimes C \mid b].$$

If $B \in \mathbb{R}^{m_1 \times n_1}$ and $C \in \mathbb{R}^{m_2 \times n_2}$, then the SVD of $B \otimes C$ costs $O(m_1 n_1^2 + m_2 n_2^2)$ while the SVD of M appears to require $O((m_1 m_2)(n_1 n_2)^2)$. However, in this case the special structure of M permits the fast calculation of the required minimum singular triple via the coupling of condition estimation ideas with the QR factorization of $B \otimes C$.

4. Tensoring low-dimension ideas

Tensor product "ideas" in approximation and interpolation also lead to Kronecker product problems. In these multidimensional situations the "overall" method involves repetition of the same 1-dimensional idea in each coordinate direction. For example, if 1-dimensional quadrature rules of the form $\int f(x)\,\mathrm{d}x \approx \sum w_i f(x_i) \equiv w^{\mathrm{T}} f(x)$ are applied in the x, y, and z directions to the triple integral

$$I = \int_{a_1}^{b_1} \int_{a_2}^{b_2} \int_{a_3}^{b_3} g(x, y, z)\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z,$$

then we obtain

$$I \approx \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} \sum_{k=1}^{n_z} w_i^{(x)} w_j^{(y)} w_k^{(z)} g(x_i, y_j, z_k) = (w^{(x)} \otimes w^{(y)} \otimes w^{(z)})^{\mathrm{T}} g(x \otimes y \otimes z)$$

where $x \in \mathbb{R}^{n_x}$, $y \in \mathbb{R}^{n_y}$, and $z \in \mathbb{R}^{n_z}$ are vectors of abscissa values and $g(x \otimes y \otimes z)$ designates the vector of values obtained by evaluating g at each component of $x \otimes y \otimes z$. For further details about this kind of multidimensional problem [15,16]. The computationally oriented papers by Pereyra and Scherer [55] and de Boor [17] are motivated by these tensor product applications and are among the earliest references that discuss how to organize a Kronecker product calculation.

The ability to solve problems with increasingly high dimension because of very powerful computers partially explains the heightened profile of the Kronecker product in scientific computing. Interest in higher-order statistics is a good example. Roughly speaking, second-order statistics revolve around the expected value of xx^{T} where x is a random vector. In higher-order statistics the calculations involve the "cumulants" $x \otimes x \otimes \cdots \otimes x$. (Note that $\mathrm{vec}(xx^{\mathrm{T}}) = x \otimes x$.) [67,1]. Related Kronecker product computations arise in Volterra filtering as presented in [54,53]. A collection of very high dimensional Kronecker product problems that arise in statistical mechanics and quantum mechanics is discussed [63].

5. Fast transforms

Kronecker products and various generalizations are what "drive" many fast transform algorithms. Consider the fast Fourier transform (FFT) with $n = 2^t$. If P_n is the *bit reversal permutation*

$$P_n = S_{2,n/2}(I_2 \otimes S_{2,n/4}) \cdots (I_{n/4} \otimes S_{2,2})$$

and $\omega_n = \exp(-2\pi \mathrm{i}/n)$, then the discrete Fourier transform (DFT) matrix $F_n = (\omega_n^{pq}) \in \mathbb{C}^{n \times n}$ can be factored as $F_n = A_t \cdots A_1 P_n$ where

$$A_q = I_r \otimes \begin{bmatrix} I_{L/2} & \Omega_{L/2} \\ I_{L/2} & -\Omega_{L/2} \end{bmatrix},$$

with $L=2^q$, $r=n/L$, $\Omega_{L/2}=\mathrm{diag}(1,\omega_L,\ldots,\omega_L^{L/2-1})$, and $\omega_L=\exp(-2\pi i/L)$. Based on this "sparse" factorization of the DFT matrix we obtain the Cooley–Tukey FFT framework for computing the DFT $y=F_nx=A_t\cdots A_1P_nx$:

$$
\begin{array}{l}
x \leftarrow P_n x \\
\text{for } k=1:t \\
\qquad x \leftarrow A_q x \\
\text{end} \\
y \leftarrow x
\end{array}
$$

Tolimieri et al. [70] and Van Loan [72] have shown how the organization of the FFT is clarified through the "language" of Kronecker products. Different FFT algorithms correspond to different factorizations of F_n. The value of this point of view is that it unifies the literature and exposes simple connections between seemingly disparate algorithms. For example, the Gentleman–Sande FFT framework results by taking transposes in $F_n=A_t\cdots A_1P_n$ and noting that F_n and P_n are symmetric:

$$
\begin{array}{l}
\text{for } k=1:t \\
\qquad x \leftarrow A_q^{\mathrm{T}} x \\
\text{end} \\
y \leftarrow P_n x
\end{array}
$$

The evolution of FFT ideas and algorithms would have proceeded much more rapidly and with greater clarity from the famous 1965 Cooley–Tukey paper onwards had the Kronecker notation been more actively employed.

Huang et al. [39] have developed a parallel programming methodology that revolves around the Kronecker product. Related papers include Johnson et al. [41], and Granata et al. [31,32]. Pitsianis [56] built a "Kronecker compiler" that permits the user to specify algorithms in a Kronecker product language. The compiler is based on a set of term rewriting rules that translate high-level, Kronecker-based matrix descriptions of an algorithm into any imperative language such as C, MATLAB or Fortran. The efficiency of the automatically generated code is shown to be excellent.

The Kronecker product methodology extends beyond the FFT and the related sine/cosine transforms. Indeed, Regalia and Mitra [60] have used the Kronecker product and various generalizations of it to describe a range of fast unitary transforms; See also [40] for Kronecker presentations of the Walsh–Hadamard, slant, and Hartley transforms. Strohmer [66] uses a Kronecker product framework to develop factorizations for the Gabor frame operator while Fijany and Williams [23] do the same thing with quantum wavelet transforms. Kumar et al. [48] use Kronecker product ideas to develop a memory-efficient implementation of the Strassen matrix multiply.

Kronecker products and sparse factorizations are as central to fast wavelet transforms as they are to the FFT. Consider the Haar wavelet transform $y=W_nx$ where $n=2^t$. The transform matrix W_n is defined by

$$W_n=\left[W_m\otimes\begin{pmatrix}1\\1\end{pmatrix}\;\middle|\;I_m\otimes\begin{pmatrix}1\\-1\end{pmatrix}\right]\qquad n=2m$$

with $W_1=[1]$. It can be shown that

$$W_n = S_{2,m}(W_2 \otimes I_m)\begin{bmatrix} W_m & 0 \\ 0 & I_m \end{bmatrix} \quad n=2m$$

and from this "splitting" it is possible to show that $W_n = H_t \cdots H_1$ where

$$H_q = \begin{bmatrix} S_{2,L_*} & 0 \\ 0 & I_{n-L} \end{bmatrix}\begin{bmatrix} W_2 \otimes I_{L_*} & 0 \\ 0 & I_{n-L} \end{bmatrix} \quad L=2^q,\ L_*=L/2.$$

The fast Haar transform then proceeds as a sequence of matrix–vector products, i.e., $x \leftarrow H_1x$, $x \leftarrow H_2x, \ldots, x \leftarrow H_tx$.

More complicated wavelets require more sophisticated Kronecker manipulations in order to derive the underlying sparse factorization. For example, the $n=4$ transform matrix for the Daubechies wavelet is given by

$$D_4 = \left[\begin{array}{cc|cc} c_0 & c_1 & c_2 & c_3 \\ c_3 & -c_2 & c_1 & -c_0 \\ \hline c_2 & c_3 & c_0 & c_1 \\ c_1 & -c_0 & c_3 & -c_2 \end{array}\right] \quad \text{where} \quad \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \frac{1}{4\sqrt{2}}\begin{bmatrix} 1+\sqrt{3} \\ 3+\sqrt{3} \\ 3-\sqrt{3} \\ 1-\sqrt{3} \end{bmatrix}.$$

It is easy to verify that D_4 is orthogonal. The $n=8$ version is given as follows:

$$D_8 = \left[\begin{array}{cc|cc|cc|cc} c_0 & c_1 & c_2 & c_3 & 0 & 0 & 0 & 0 \\ c_3 & -c_2 & c_1 & -c_0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & c_0 & c_1 & c_2 & c_3 & 0 & 0 \\ 0 & 0 & c_3 & -c_2 & c_1 & -c_0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & c_0 & c_1 & c_2 & c_3 \\ 0 & 0 & 0 & 0 & c_3 & -c_2 & c_1 & -c_0 \\ \hline c_2 & c_3 & 0 & 0 & 0 & 0 & c_0 & c_1 \\ c_1 & -c_0 & 0 & 0 & 0 & 0 & c_3 & -c_2 \end{array}\right]$$

which clearly has a replicated block structure. It is possible to describe this structure quite elegantly with a generalized Kronecker product. It is then a straightforward exercise to obtain a splitting that relates D_8 to D_4 and more generally, D_n to $D_{n/2}$. From the splitting one can then derive the sparse factorization associated with the underlying fast transform [23].

It is almost always the case that behind every fast linear transform is a sparse, Kronecker-based, factorization of the transform matrix. Notable exceptions are the recent and very interesting fast algorithms whose complexity has the form cn or $cn\log n$ where c is a constant that depends on the precision required. Examples include the fast Gauss transform of Greengard and Strain [33] and the non-uniformly spaced FFT of Dutt and Rokhlin [19]. It is interesting to conjecture whether these algorithms can be described in terms of some approximate sparse, Kronecker-based factorization of the underlying transform matrix.

Fast transforms often require various matrix transpositions of the data and these operations also submit to Kronecker product descriptions. For example, it follows from (1) that if $A \in \mathbb{R}^{m\times n}$ and $B=A^{\mathrm{T}}$, then $\mathrm{vec}(B) = S_{n,m} \cdot \mathrm{vec}(A)$. It turns out that different "multi-pass" transposition algorithms correspond to different factorizations of the underlying perfect shuffle. If

$$S_{n,m} = \Gamma_t \cdots \Gamma_1 \tag{7}$$

then $B = A^T$ can be computed with t passes through the data as follows:

$a = \text{vec}(A)$
for $k = 1 : t$
  $a \leftarrow \Gamma_k a$
end
Define $B \in \mathbb{R}^{n\times m}$ by $\text{vec}(B) = a$.

The idea is to choose a factorization (7) so that the data motion behind the operation kth pass, i.e., $a \leftarrow \Gamma_k a$, is in harmony with the architecture of the underlying memory hierarchy.

To illustrate this factorization idea, it can be shown that if $m = pn$, then $S_{n,m} = \Gamma_2\Gamma_1$ where

$$\Gamma_1 = S_{n,p} \otimes I_n,$$
$$\Gamma_2 = I_p \otimes S_{n,n}.$$

The first pass $b^{(1)} = \Gamma_1 \text{vec}(A)$ corresponds to a block transposition while $b^{(2)} = \Gamma_2 b^{(1)}$ carries out the transposition of the blocks. For example, if $p = 4$ and

$$A = \begin{bmatrix} A_1 \\ A_2 \\ A_3 \\ A_4 \end{bmatrix}, \quad A_i \in \mathbb{R}^{n\times n}$$

then the Γ_1 update leaves us with

$$B^{(1)} = [\, A_1 | A_2 | A_3 | A_4 \,].$$

During the Γ_2 update the individual blocks are transposed yielding

$$B = B^{(2)} = [\, A_1^T | A_2^T | A_3^T | A_4^T \,].$$

See [72] for more details about factorizations and matrix transposition.

6. The nearest Kronecker product problem

Suppose $A \in \mathbb{R}^{m\times n}$ is given with $m = m_1 m_2$ and $n = n_1 n_2$. For these integer factorizations the nearest Kronecker product (NKP) problem involves minimizing

$$\phi(B, C) = ||A - B \otimes C||_F \tag{8}$$

where $B \in \mathbb{R}^{m_1\times n_1}$ and $C \in \mathbb{R}^{m_2\times n_2}$. Van Loan and Pitsianis [73] show how to solve the NKP problem using the singular value decomposition of a permuted version of A. This result is central to much of the research proposal and so we use a small example to communicate the main idea. Suppose $m_1 = 3$ and $n_1 = m_2 = n_2 = 2$. By carefully thinking about the sum of squares that define ϕ we see that

$$\phi(B,C) = \left\| \left[\begin{array}{cc|cc} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \\ \hline a_{51} & a_{52} & a_{53} & a_{54} \\ a_{61} & a_{62} & a_{63} & a_{64} \end{array}\right] - \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} \otimes \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \right\|_F$$

$$= \left\| \begin{bmatrix} a_{11} & a_{21} & a_{12} & a_{22} \\ a_{31} & a_{41} & a_{32} & a_{42} \\ a_{51} & a_{61} & a_{52} & a_{62} \\ a_{13} & a_{23} & a_{14} & a_{24} \\ a_{33} & a_{43} & a_{34} & a_{44} \\ a_{53} & a_{63} & a_{54} & a_{64} \end{bmatrix} - \begin{bmatrix} b_{11} \\ b_{21} \\ b_{31} \\ b_{12} \\ b_{22} \\ b_{32} \end{bmatrix} [\, c_{11} \;\; c_{21} \;\; c_{12} \;\; c_{22} \,] \right\|_F .$$

Denote the preceeding 6×4 matrix by $\mathscr{R}(A)$ and observe that

$$\mathscr{R}(A) = \begin{bmatrix} \text{vec}(A_{11})^{\mathrm{T}} \\ \text{vec}(A_{21})^{\mathrm{T}} \\ \text{vec}(A_{31})^{\mathrm{T}} \\ \text{vec}(A_{12})^{\mathrm{T}} \\ \text{vec}(A_{22})^{\mathrm{T}} \\ \text{vec}(A_{32})^{\mathrm{T}} \end{bmatrix} .$$

It follows that

$$\phi(B,C) = ||\mathscr{R}(A) - \text{vec}(B)\text{vec}(C)^{\mathrm{T}}||_F$$

and so the act of minimizing ϕ is equivalent to finding a nearest rank-1 matrix to $\mathscr{R}(A)$. The nearest rank-1 matrix problem has a well-known SVD solution [29]. In particular, if

$$U^{\mathrm{T}}\mathscr{R}(A)V = \Sigma \tag{9}$$

is the SVD of $\mathscr{R}(A)$, then optimum B and C are defined by

$$\text{vec}(B_{\text{opt}}) = \sqrt{\sigma_1}U(:,1) \quad \text{vec}(C_{\text{opt}}) = \sqrt{\sigma_1}V(:,1).$$

The scalings are arbitrary. Indeed, if B_{opt} and C_{opt} solve the NKP problem and $\alpha \neq 0$, then $\alpha \cdot B_{\text{opt}}$ and $(1/\alpha) \cdot C_{\text{opt}}$ are also optimal.

In general, if $A = (A_{ij})$ is an $m_1 \times n_1$ block matrix with $m_2 \times n_2$ blocks, then

$$\tilde{A} = \mathscr{R}(A) \in \mathbb{R}^{m_1 n_1 \times m_2 n_2} \Rightarrow \tilde{A}(i + (j-1)m_1, :) = \text{vec}(A_{ij})^{\mathrm{T}}, \quad i = 1 : m_1,\ j = 1 : n_1.$$

If $\mathscr{R}(A))$ has rank $\tilde{r}$ and SVD (9), then

$$A = \sum_{k=1}^{\tilde{r}} \sigma_k U_k \otimes V_k$$

where $\mathrm{vec}(U_k) = U(:,k)$ and $\mathrm{vec}(V_k) = V(:,k)$ for $k = 1 : \tilde{r}$. We refer to this as the *Kronecker Product SVD* (KPSVD) of A associated with the integer factorizations $m = m_1 m_2$ and $n = n_1 n_2$. Note that for these integers,

$$A_r = \sum_{k=1}^{r} \sigma_k U_k \otimes V_k \quad r \leqslant \tilde{r} \tag{10}$$

is the closest matrix to A (in the Frobenius norm) that is the sum of r Kronecker products.

If A is large and sparse and r is small, then the Lanzcos SVD iteration of Golub, Luk, and Overton [26] can effectively be used to compute the singular vectors of $\mathscr{R}(A)$ from which we can build the optimal Kronecker product factors. An implementation is available in [12] and some preliminary experience with the method is discussed in [73,56].

Certain situations permit one to "cut corners" in the above SVD computation when solving the NKP problem:

- If A is the sum of p Kronecker products as in the generalized Sylvester equation problem (3), then $\mathrm{rank}(\mathscr{R}(A)) \leqslant p$.
- If A is an $n_1 \times n_1$ block Toeplitz matrix with $n_2 \times n_2$ Toeplitz blocks, then it is not hard to show that the rank of $\mathscr{R}(A)$ is less than $\min\{2n_1 + 1, 2n_2 + 1\}$ [51].

In each of these situations the matrix $\mathscr{R}(A)$ is rank deficient.

7. Other NKP problems

For $A \in \mathbb{R}^{n \times n}$ with $n = n_1 n_2$ we refer to the problem of minimizing

$$\psi(B,C) = ||A(B \otimes C) - I_n||_F \quad B \in \mathbb{R}^{n_1 \times n_1},\ C \in \mathbb{R}^{n_2 \times n_2} \tag{11}$$

as the *inverse nearest Kronecker product problem*. This problem does not have an explicit SVD solution and so we must approach it as a structured nonlinear least squares problem. It can be shown that

$$\mathrm{vec}(A(B \otimes C)) = (I_n \otimes A)\mathrm{vec}(B \otimes C) = (I_n \otimes A)P(\mathrm{vec}(B) \otimes \mathrm{vec}(C))$$

where P is the $n^2 \times n^2$ permutation matrix defined by

$$P = I_{n_1} \otimes S_{n_1,n_2} \otimes I_{n_2}$$

Thus, minimizing ψ is equivalent to minimizing the 2-norm of

$$F(B,C) = (I_n \otimes A)P(\mathrm{vec}(B) \otimes \mathrm{vec}(C)) - \mathrm{vec}(I_n).$$

The Jacobian of this function is $(I_n \otimes A)P[(I_n \otimes \mathrm{vec}(C))\ \mathrm{vec}(B) \otimes I_n)]$. Having exposed these structures we see that there are several ways to approach the inverse NKP problem. Since it is a separable least-squares problem, the variable projection methods discussed in [28,46] or the ideas in [3] are applicable.

The NKP problem has a multiple factor analog in which we try to approximate $A \in \mathbb{R}^{m \times n}$ with a matrix of the form $C_1 \otimes \cdots \otimes C_p$. In particular, if $m = m_1 \cdots m_p$ and $n = n_1 \cdots n_p$, then we seek $C_i \in \mathbb{R}^{m_i \times n_i}$, $i = 1 : p$ so that $\phi(C_1, \ldots, C_p) = ||A - C_1 \otimes \cdots \otimes C_p||_F$ is minimized. Closed-form SVD solutions do not appear to be possible if $p > 2$. The inverse NKP problem also has a multiple factor generalization.

If A is structured, then it is sometimes the case that the B and C matrices that solve the NKP problem are similarly structured. For example, if A is symmetric and positive definite, then the same can be said of B_{opt} and C_{opt} (if properly normalized). Likewise, if A is nonnegative, then the optimal B and C can be chosen to be nonnegative. These and other structured NKP problems are discussed in [73,56], but a number of interesting open questions remain.

Suppose $A \in \mathbb{R}^{n\times n}$ and that $n = m^2$. A trace minimization problem that we are aware of requires the minimization of $||A - \alpha B \otimes B||_F$ where $B \in \mathbb{R}^{m\times m}$ and $\alpha = \pm 1$. This leads to a nearest symmetric rank-1 problem of the form

$$\min_{\alpha,b} ||\mathscr{R}(A) - \alpha bb^{\mathrm{T}}||_F.$$

A related problem arises in neural networks [38]. Given $A \in \mathbb{R}^{n\times n}$ with $n = m^2$, find $B \in \mathbb{R}^{m\times m}$ so that $||A - B \otimes B^{\mathrm{T}}||_F$ is minimized. This leads to the minimization of

$$||\mathscr{R}(A) - \operatorname{vec}(B)\operatorname{vec}(B^{\mathrm{T}})^{\mathrm{T}}||_F = ||\mathscr{R}(A)S_{m,m} - \operatorname{vec}(B)\operatorname{vec}(B)^{\mathrm{T}}||_F.$$

The linearly constrained NKP problem

$$\min_{\substack{F^{\mathrm{T}}\operatorname{vec}(B)=r\\ G^{\mathrm{T}}\operatorname{vec}(C)=t}} ||A - B \otimes C||_F$$

leads to a linearly constrained nearest rank-1 problem

$$\min_{\substack{F^{\mathrm{T}}b=r\\ G^{\mathrm{T}}c=t}} ||\tilde{A} - bc^{\mathrm{T}}||_F$$

where $\tilde{A} = \mathscr{R}(A)$, $b = \operatorname{vec}(B)$ and $c = \operatorname{vec}(C)$. We suppress the dimensions of the matrices involved and just assume that the linear constraint equations are underdetermined. Following Golub [25] we compute the QR factorizations

$$F = Q_F \begin{bmatrix} R_F \\ 0 \end{bmatrix} \qquad G = Q_G \begin{bmatrix} R_G \\ 0 \end{bmatrix}$$

for then the problem transforms to

$$\min_{b_2,c_2} \left|\left| \begin{bmatrix} \tilde{A}_{11} - b_1c_1^{\mathrm{T}} & \tilde{A}_{12} - b_1c_2^{\mathrm{T}} \\ \tilde{A}_{21} - b_2c_1^{\mathrm{T}} & \tilde{A}_{22} - b_2c_2^{\mathrm{T}} \end{bmatrix} \right|\right|_F$$

where

$$Q_F^{\mathrm{T}}b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \quad Q_G^{\mathrm{T}}c = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}, \quad Q_F^{\mathrm{T}}\tilde{A}Q_G = \begin{bmatrix} \tilde{A}_{11} & \tilde{A}_{12} \\ \tilde{A}_{21} & \tilde{A}_{22} \end{bmatrix}$$

and $R_F^{\mathrm{T}}b_1 = r$ and $R_G^{\mathrm{T}}c_1 = t$. Thus, we are led to the minimization of the function

$$\tau(b_2, c_2) = ||\tilde{A}_{22} - b_2c_2^{\mathrm{T}}||_F^2 + ||\tilde{A}_{12} - b_1c_2^{\mathrm{T}}||_F^2 + ||\tilde{A}_{21} - b_2c_1^{\mathrm{T}}||_F^2.$$

If r and t are zero, then b_1 and c_1 are zero and we are left with a reduced version of the nearest rank-1 matrix problem. This homogeneous situation arises when we wish to impose sparsity constraints on the B and C matrices or when we require the optimizing B and/or C to have circulant, Toeplitz, Hankel, or some other structure of that type. In the nonhomogeneous case we are again confronted with a bilinear least-square problem. (The inhomogeneous problem would arise, for example, if we required the B and C matrices to have columns that sum to one.)

8. Preconditioners

In recent years the "culture" of fast transforms and Kronecker products has found its way into the linear system solving area through the design of effective preconditioners. Chan [8] proposed solving large Toeplitz systems $Tx = b$ using a circulant preconditioner C that minimizes $||C - T||_F$. The product of a circulant matrix and a vector can be carried out in $O(n \log n)$ flops using the FFT; See also [9] and [10].

Similar in spirit is the design of Kronecker product preconditioners. The idea is to approximate the matrix of coefficients A with a Kronecker product $B \otimes C$ and to use $B \otimes C$ as a preconditioner noting that linear systems of the form $(B \otimes C)z = r$ can be solved fast. One method for generating the Kronecker factors B and C is to minimize $||A - B \otimes C||_F$. Other approaches tailored to applications in image restoration have been offered by Nagy [51], Kamm and Nagy [43–45] and Thirumalai [68]; See also [20,52,7].

The success of many numerical methods hinge on efficient linear equation solving and this in turn often requires finding the "right" preconditioner for the coefficient matrix A. To be effective, the preconditioner M must "capture the essence" of A and have the property that systems of the form $Mz = r$ are easily solved.

The idea of setting M to be the nearest Kronecker product to A is studied in [73]. When applied to a model problem (Poisson's equation on a rectangle) the results compared favorably with the best alternatives, e.g., the incomplete Cholesky preconditioner. This work can be extended by (a) looking at 3D problems where the resulting linear system is the sum of three Kronecker products and (b) considering non-uniform mesh settings where the linear system is the sum of a few "near" Kronecker products.

9. Conclusion

Our goal in this paper is to point to the widening use of the Kronecker product in numerical linear algebra. Research in this area will heighten the profile of the Kronecker product throughout the field of matrix computations and will make it easier for researchers to spot Kronecker "opportunities" in their work. This phenomena is not without precedent. The development of effective algorithms for the QR and SVD factorizations turned many "A^TA" problems into least-square/singular-value calculations. Likewise, with the development of the QZ algorithm [50] engineers with "standard" eigenproblems of the form $B^{-1}Ax = \lambda x$ came to approach them as a generalized eigenproblems of the form $Ax = \lambda Bx$. The point we are making is that if an infrastructure of effective Kronecker-product algorithms is built, then Kronecker product problems will "come out of the woodwork."

References

[1] T.F. Andre, R.D. Nowak, B.D. Van Veen, Low rank estimation of higher order statistics, IEEE Trans. Signal Process. 45 (1997) 673–685.

[2] H.C. Andrews, J. Kane, Kronecker matrices, computer implementation, and generalized spectra, J. Assoc. Comput. Mach. 17 (1970) 260–268.

[3] R.H. Barham, W. Drane, An algorithm for least squares estimation of nonlinear parameters when some of the parameters are linear, Technometrics 14 (1972) 757–766.

[4] A. Barrlund, Efficient solution of constrained least squares problems with Kronecker product structure, SIAM J. Matrix Anal. Appl. 19 (1998) 154–160.
[5] R.H. Bartels, G.W. Stewart, Solution of the equation $AX + XB = C$, Comm. ACM 15 (1972) 820–826.
[6] J.W. Brewer, Kronecker products and matrix calculus in system theory, IEEE Trans. Circuits Systems 25 (1978) 772–781.
[7] D. Calvetti, L. Reichel, Application of ADI iterative methods to the image restoration of noisy images, SIAM J. Matrix Anal. Appl. 17 (1996) 165–174.
[8] T.F. Chan, An optimal circulant preconditioner for Toeplitz systems, SIAM J. Sci. Statist. Comput. 9 (1988) 766–771.
[9] T.F. Chan, J.A. Olkin, Preconditioners for Toeplitz-block matrices, Numer. Algorithms 6 (1993) 89–101.
[10] R. Chan, X.-Q. Jin, A family of block preconditioners for block systems, SIAM J. Sci. Statist. Comput. 13 (1992) 1218–1235.
[11] T.F. Coleman, Y. Li, A. Mariano, Separation of pulmonary nodule images using total variation minimization, Technical Report, Cornell Theory Center, Ithaca, New York, 1998.
[12] J. Cullum, R.A. Willoughby, Lanczos Algorithms for Large Sparse Symmetric Eigenvalue Computations, Vol. I (Theory), II (Programs), Birkhauser, Boston, 1985.
[13] K. Datta, The matrix equation $XA - BX = R$ and its applications, Linear Algebra Appl. 109 (1988) 91–105.
[14] M. Davio, Kronecker products and shuffle algebra, IEEE Trans. Comput. c-30 (1981) 116–125.
[15] P. Davis, P. Rabinowitz, Numerical Integration, Blaisdell, Waltham, MA, 1967.
[16] C. de Boor, A Practical Guide to Splines, Springer, New York, 1978.
[17] C. de Boor, Efficient computer manipulation of tensor products, ACM Trans. Math. Software 5 (1979) 173–182.
[18] F.W. Dorr, The direct solution of the discrete poisson equation on a rectangle, SIAM Rev. 12 (1970) 248–263.
[19] A. Dutt, V. Rokhlin, Fast fourier transforms for nonequispaced data, SIAM J. Sci. Comput. 14 (1993) 1368–1398.
[20] L. Eldin, I. Skoglund, Algorithms for the regularization of Ill-conditioned least squares problems with tensor product structure and applications to space-variant image restoration, Technical Report LiTH-Mat-R-82-48, Department of Mathematics, Linkoping University, Sweden, 1982.
[21] D.W. Fausett, C.T. Fulton, Large least squares problems involving Kronecker products, SIAM J. Matrix Anal. 15 (1994) 219–227.
[22] D.W. Fausett, C.T. Fulton, H. Hashish, Improved parallel QR method for large least squares problems involving Kronecker products, J. Comput. Appl. Math. (1997).
[23] A. Fijany, C.P. Williams, Quantum wavelet transforms: fast algorithms and complete circuits, Technical Report 9809004, Los Alamos National Laboratory, Los Alamos, New Mexico, 1998.
[24] J. Gardiner, M.R. Wette, A.J. Laub, J.J. Amato, C.B. Moler, Algorithm 705: a FORTRAN-77 software package for solving the Sylvester matrix equation $AXB^T + CXD^T = E$, ACM Trans. Math. Software 18 (1992) 232–238.
[25] G.H. Golub, Some modified eigenvalue problems, SIAM Rev. 15 (1973) 318–344.
[26] G.H. Golub, F. Luk, M. Overton, A block Lanzcos method for computing the singular values and corresponding singular vectors of a matrix, ACM Trans. Math. Software 7 (1981) 149–169.
[27] G.H. Golub, S. Nash, C. Van Loan, A Hessenberg-Schur method for the matrix problem $AX + XB = C$, IEEE Trans. Automat. Control AC-24 (1979) 909–913.
[28] G.H. Golub, V. Pereya, The differentiation of pseudoinverses and nonlinear least squares problems whose variables separate, SIAM J. Numer. Anal. 10 (1973) 413–432.
[29] G.H. Golub, C. Van Loan, Matrix Computations, 3rd Edition, Johns Hopkins University Press, Baltimore, MD, 1996.
[30] A. Graham, Kronecker Products and Matrix Calculus with Applications, Ellis Horwood, Chichester, England, 1981.
[31] J. Granata, M. Conner, R. Tolimieri, Recursive fast algorithms and the role of the tensor product, IEEE Trans. Signal Process. 40 (1992) 2921–2930.
[32] J. Granata, M. Conner, R. Tolimieri, The tensor product: a mathematical programming language for FFTs and other fast DSP operations, IEEE SP Mag. (January 1992) 40–48.
[33] L. Greengard, J. Strain, The fast Gauss transform, SIAM J. Sci. Statist. Comput. 12 (1991) 79–94.
[34] S.R Heap, D.J. Lindler, Block iterative restoration of astronomical images with the massively parallel processor Proceedings of the First Aerospace Symposium on Massively Parallel Scientific Computation, 1986, pp. 99–109.
[35] H.V. Henderson, F. Pukelsheim, S.R. Searle, On the history of the Kronecker product, Linear Multilinear Algebra 14 (1983) 113–120.

[36] H.V. Henderson, S.R. Searle, The vec-permutation matrix, the vec operator and Kronecker products: a review, Linear Multilinear Algebra 9 (1981) 271–288.
[37] R.A. Horn, C.A. Johnson, Topics in Matrix Analysis, Cambridge University Press, New York, 1991.
[38] R.M. Hristev, private communication, 1998.
[39] C-H. Huang, J.R. Johnson, R.W. Johnson, Multilinear algebra and parallel programming, J. Supercomput. 5 (1991) 189–217.
[40] A.K. Jain, Fundamentals of Digital Image Processing, Prentice-Hall, Englewood Cliffs, NJ, 1989.
[41] J. Johnson, R.W. Johnson, D. Rodriguez, R. Tolimieri, A methodology for designing, modifying, and implementing fourier transform algorithms on various architectures, Circuits Systems Signal Process. 9 (1990) 449–500.
[42] B. Kagstrom, Perturbation analysis of the generalized Sylvester equation $(AR - LB, DR - LE) = (C, F)$, SIAM J. Matrix Anal. Appl. 15 (1994) 1045–1060.
[43] J. Kamm, J.G. Nagy, Kronecker product and SVD approximations in image restoration, Linear Algebra Appli. (1998a), to appear.
[44] J. Kamm, J.G. Nagy, Kronecker product and SVD approximations for separable spatially variant blurs, SMU Mathematics Technical Report 98-04, Dallas, TX, 1998b.
[45] J. Kamm, J.G. Nagy, Optimal kronecker product approximation of block Toeplitz matrices, SMU Mathematics Technical Report 98-05, Dallas, TX, 1998c.
[46] L. Kaufman, A variable projection method for solving separable nonlinear least squares problems, BIT 15 (1975) 49–57.
[47] B. Klaus, P. Horn, Robot Vision, MIT Press, Cambridge, MA, 1990.
[48] B. Kumar, C.H. Huang, J. Johnson, R.W. Johnson, P. Sadayappan, A tensor product formulation of Strassen's matrix multiplication algorithm with memory reduction, Seventh International Parallel Processing Symposium, 1993, pp. 582–588.
[49] P. Lancaster, Explicit solution of linear matrix equations, SIAM Rev. 12 (1970) 544–566.
[50] C.B. Moler, G.W. Stewart, An algorithm for generalized matrix eigenvalue problems, SIAM J. Numer. Anal. 10 (1973) 241–256.
[51] J.G. Nagy, Decomposition of block Toeplitz matrices into a sum of Kronecker products with applications in image processing, SMU Mathematics Technical Report 96-01, Dallas, TX, 1996.
[52] J.G. Nagy, D.P. O'Leary, Restoring images degraded by spatially-variant blur, SIAM J. Sci. Comput. 19 (1998) 1063–1082.
[53] R.D. Nowak, Penalized least squares estimation of higher order statistics, IEEE Trans. Signal Process. 46 (1998) 419–428.
[54] R.D. Nowak, B. Van Veen, Tensor product basis approximations for Volterra filters, IEEE Trans Signal Process. 44 (1996) 36–50.
[55] V. Pereyra, G. Scherer, Efficient computer manipulation of tensor products with applications to multidimensional approximation, Math. Comp. 27 (1973) 595–604.
[56] N.P. Pitsianis, The Kronecker product in approximation, fast transform generation, Ph.D. Thesis, Department of Computer Science, Cornell University, 1997.
[57] J.H. Pollard, On the use of the direct matrix product in analyzing certain stochastic population models, Biometrika 53 (1966) 397–415.
[58] U.A. Rauhala, Introduction to array algebra, Photogrammetric Eng. Remote Sensing 46 (2) (1980) 177–182.
[59] U.A. Rauhala, D. Davis, K. Baker, Automated DTM validation and progressive sampling algorithm of finite element array relaxation, Photogrammetric Eng. Remote Sensing 55 (1989) 449–465.
[60] P.A. Regalia, S. Mitra, Kronecker products, unitary matrices, and signal processing applications, SIAM Rev. 31 (1989) 586–613.
[61] J. Seberry, X-M. Zhang, Some orthogonal matrices constructed by strong Kronecker product multiplication, Austral. J. Combin. 7 (1993) 213–224.
[62] R.A. Snay, Applicability of array algebra, Rev. Geophys. Space Phys. 16 (1978) 459–464.
[63] W-H. Steeb, Matrix Calculus and Kronecker Product with Applications and C++ Programs, World Scientific Publishing, Singapore, 1997.
[64] F. Stenger, Kronecker product extensions of linear operators, SIAM J. Numer. Anal. 5 (1968) 422–435.
[65] L. Stiller, Multilinear algebra and chess endgames, in: Computational Games, Vol. 29, MSRI Publications.

[66] T. Strohmer, Numerical algorithms for discrete gabor expansions, in: H.G. Feichtinger, T. Strohmer (Eds.), Gabor Analysis and Algorithms, Birkhauser, Basel, 1998, pp. 267–294.
[67] A. Swami, J. Mendel, Time and lag recursive computation of cumulants from a state-space model, IEEE Trans. Automat. Control 35 (1990) 4–17.
[68] S. Thirumalai, High performance algorithms to solve Toeplitz and block Toeplitz matrices, Ph.D. Thesis, University of Illinois, Urbana, IL, 1996.
[69] M.J. Todd, On search directions in interior point methods for semidefinite programming, Optim. Methods Software, (1998), to appear.
[70] R. Tolimieri, M. An, C. Lu, Algorithms for Discrete Fourier Transform and Convolution, Springer, New York, 1989.
[71] L. Vandenberghe, S. Boyd, Semidefinite Programming, SIAM Rev. 38 (1996) 27–48.
[72] C. Van Loan, Computational Frameworks for the Fast Fourier Transform, SIAM Publications, Philadelphia, PA, 1992.
[73] C. Van Loan, N.P. Pitsianis, Approximation with Kronecker products, in: M.S. Moonen, G.H. Golub (Eds.), Linear Algebra for Large Scale and Real Time Applications, Kluwer Publications, Dordrecht, 1992, pp. 293–314.
[74] W.J. Vetter, Vector structures, solutions of linear matrix equations, Linear Algebra Appl. 10 (1975) 181–188.