Combining Kronecker product approximation with discrete wavelet transforms to solve dense, function-related linear systems[*]

Judith M. Ford[†] Eugene E. Tyrtyshnikov[‡]

May 8, 2003

Abstract

A new solution technique is proposed for linear systems with large dense matrices of a certain class including those that come from typical integral equations of potential theory. This technique combines Kronecker product approximation and wavelet sparsification for the Kronecker product factors. The user is only required to supply a procedure for computation of each entry of the given matrix.

The main sources of efficiency are the incomplete cross approximation procedure adapted from the mosaic-skeleton method of the second author and data-sparse preconditioners (the incomplete LU decomposition with dynamic choice of the fill-in structure with a prescribed threshold and the inverse Kronecker product preconditioner) constructed for the sum of Kronecker products of sparsified finger-like matrices computed by the Discrete Wavelet Transform. In some model, but quite representative, examples the new technique allowed us to solve dense systems with more than 1 million unknowns in a few minutes on a personal computer with 1 Gbyte operative memory.

Kronecker product, wavelets, dense matrices, preconditioning
65F10, 65T60, 65F30

1 Introduction

Dense matrices arise during numerical solution of problems in a variety of applications in science and engineering. For example, integral operators, interface

[*]This work was supported by EPSRC Postdoctoral Research Fellowship ref: GR/R95982/01 and by the Russian Fund of Basic Research (grant 02-01-00590) and Science Support Foundation.

[†]Mathematics Department, UMIST, PO Box 88, Manchester M60 1QD, UK (`j.ford@umist.ac.uk`).

[‡]Institute of Numerical Mathematics, Russian Academy of Sciences, Gubkina Street, 8, Moscow(`tee@inm.ras.ru`).

preconditioners in the domain decomposition method, Schur complements in multilevel solution strategies, and Jacobians in the Newton method can all be sources of large dense matrices. Also, the inverses of sparse matrices in PDE applications are usually dense.

In many practical cases the matrix entries are not pronouncedly different in magnitude and almost none of them are negligibly small. More often than not, the matrices possess no special structure like that of Toeplitz and related matrices. In these same applications, the sizes of the matrices may need to be as much as several hundreds of thousands or even millions. Tackling problems on this scale has become feasible only since the 1980's when the multipole and panel clustering approaches appeared [**?**, **?**] and later on when wavelet-based techniques were adapted to numerical analysis needs [**?**, **?**]. Since then a large amount of research has been aimed at gaining a better insight into the essentials of these techniques and at the design of better-performing algorithms for many important practical purposes. A distinctive line of research has been concerned with the development of a matrix view of the established approaches [**?**, **?**, **?**, **?**, **?**, **?**] and with user-friendly matrix approximation methods [**?**, **?**, **?**, **?**] whose input is simply a procedure by which any specified entry can be computed.

Despite the breakthrough in solution strategies for large dense matrices provided by the above-mentioned techniques, they still require two things. First, we need a large amount of operative memory when matrix dimensions approach and go beyond 1 million. Second, a lot of descriptive information pertaining to quite intricate hierarchical (mosaic) block partitionings of the given matrix must be given.

In this paper, we present a new technique that can manage with a modest operative memory (up to 1 Gbyte for sizes beyond 1 million, as we observed for typical model examples) and enjoys a very simple logic.

The technique formally applies to matrices that can be associated (at least virtually) with a function of two variables in the following way:

$$A = [f(z^i, z^j)], \quad 1 \le i, j \le n, \tag{1}$$

where $\{z^i\}$ and $\{z^j\}$ are the nodes of some grids logically equivalent to the Cartesian product of some one-dimensional grids.

Given a linear system $Ax = b$ (which should read "given b and a procedure enabling us to pick up any requested entry of A") and an allowed bound ε on the relative perturbation error (in the Frobenius norm), we proceed with the following steps:

(A) Assuming $n = pq$, approximate A by a sum of Kronecker products [1]

$$B = \sum_{k=1}^{r} U_k \otimes V_k \;\approx\; A, \tag{2}$$

[1] Recall that $[u_{kl}] \otimes V$ is a block matrix of the form $[u_{kl}V]$.

with U_k and V_k of size $p \times p$ and $q \times q$, respectively, so that

$$||B - A||_F \leq \varepsilon ||A||_F. \tag{3}$$

To simplify presentation, from now on we assume that $n = p^2$ and $p = q$.

(B) Apply the Daubechies Discrete Wavelet Transforms (DWT) with a prescribed number μ of vanishing moments to U_k and V_k:

$$P_k = W U_k W^T, \quad Q_k = W V_k W^T, \quad 1 \leq k \leq r. \tag{4}$$

Here, $W = W(\mu)$ is an orthogonal matrix of the DWT of degree μ and P_k and Q_k are pseudo-sparse matrices with the well-known finger-like pattern for the significant entries. Choosing an appropriate threshold $\tau = \tau(\varepsilon, \{P_k\}, \{Q_k\})$ and setting to zero any entry that is less than τ, we get from P_k and Q_k to their sparsified counterparts P_k^τ and Q_k^τ and finally approximate B by

$$C = \left(W^T \otimes W^T\right) \; D \; \left(W \otimes W\right) \; \approx \; B, \qquad D = \left(\sum_{k=1}^{r} P_k^\tau \otimes Q_k^\tau \right), \tag{5}$$

so that

$$||C - B||_F \leq \varepsilon ||B||_F. \tag{6}$$

(C) Take $\delta > \tau$ and discard more entries from P_k and Q_k to obtain $P_k^\delta \approx P_k$ and $Q_k^\delta \approx Q_k$ with greater sparsity. Then, construct a new sparse matrix

$$E = \sum_{k=1}^{r} P_k^\delta \otimes Q_k^\delta \; \approx \; D \tag{7}$$

and compute its incomplete LU decomposition $LU \approx E$ with dynamic decision on the fill-in structure for a properly chosen threshold.

(D) Apply GMRES to solve

$$C F^{-1} y = b, \tag{8}$$

where

$$F = (W^T \otimes W^T) \; LU \; (W \otimes W) \tag{9}$$

is a preconditioner (usually termed *implicit*) for C . Finally, output $F^{-1}y$ as an approximation to the exact solution x.

Step (A) is key to the others. The coefficient matrix A cannot be stored as a full array of entries because of its size. Nevertheless, after this step, A appears in the computer memory as a matrix object represented by the sum of Kronecker products. Assuming that $r \ll n$, we need to store only $2rp^2 = 2rn \ll n^2$ numbers.

However, the matrix-vector multiplication with B requires $O(n^{3/2})$ operations. This is already better than the $\mathcal{O}(n^2)$ operations of the standard rule but is still expensive. To improve performance, we use the Discrete Wavelet Transform because it is capable of producing matrices with a bulk of relatively small entries that can be neglected with little reduction in accuracy. As soon as step (B) is completed, we are ready to employ a suitable iterative method such as GMRES using C in place of A. However, the number of iterations may be large and in this case we need a suitable preconditioner. Step (C) serves to construct one based on the incomplete LU decomposition.

Application of the proposed technique certainly has its limits, but we are nevertheless confident of its usefulness. First, we present very promising numerical results for model cases pertaining to a wide class of matrices coming from integral equations. Second, we are able to outline the limits of the method through discussion of the mathematical grounds on which the approach has grown. Third, the algebraic nature of the approach allows one to apply it in cases that may lie beyond the established grounds.

In what follows, we present a detailed description and discussion of the above steps and numerical verification of the proposed technique. To make our presentation clearer, we illustrate all the steps on one characteristic example.

Example 1.1 *Define the matrix* $A = [a_{ij}]$ *of order* $n = p^2$ *by*

$$a_{ij} = \begin{cases} 2p, & i = j, \\ 1/||z^i - z^j||, & i \neq j, \end{cases} \tag{10}$$

where

$$z^i = (x_{k(i)}, y_{l(i)}), \quad i = (k(i) - 1)p + l(i), \tag{11}$$

with uniquely defined integer $k(i)$ *and* $l(i)$ *in the range from* 1 *to* p. *The nodes* $z^1, \dots, z^n$ *belong to the unit square* $\Omega = [0,1] \times [0,1]$ *and match the Cartesian product of same one-dimensional grids with*

$$\begin{aligned} x_\alpha &= (\alpha - 0.5)/p, \quad \alpha = 1, \dots, p, \\ y_\beta &= (\beta - 0.5)/p, \quad \beta = 1, \dots, p. \end{aligned} \tag{12}$$

We have used uniform grids in our introductory example for the sake of simplicity, with the consequence that the matrix A in Example 1.1 is doubly Toeplitz[2]. This structure means that it could be stored compactly in a straightforward manner. In this paper we use Example 1.1 purely to illustrate our proposed technique, which does not rely on uniformity of the grids, so in what follows we ignore the Toeplitz property of A. Real-life applications typically use non-uniform grids and so give rise to matrices that do not have such a convenient structure. Numerical experiments in the last section confirm that performance of our new technique is similar, regardless of the choice of grids.

[2]By "doubly Toeplitz" we mean that A is a block Toeplitz matrix whose blocks are themselves Toeplitz.

2 Approximation by the sum of Kronecker products

The idea of separation of variables is pervasive in approximation theory and analytical and numerical methods for operator equations. In the language of matrices, this idea converts to one of using low-rank approximations which are equivalent, as noted in [?], to Kronecker product approximations of some related matrices, and matrix approximations by sums of Kronecker products (especially in the case of multidimensional matrices) have become one of the major research topics in the numerical analysis and linear algebra communities [?, ?, ?, ?, ?].

In the context of this article, the work of Kamm and Nagy [?] on the preconditioning of block Toeplitz matrices is of particular interest since it can be viewed as following steps (A) - (D) above but using the singular vectors of U_1, V_1 in place of the wavelet basis and adopting the compression criterion of retaining only diagonal entries. We are concerned here with a wider class of dense problems, which makes it necessary to look for a more general approach. In particular, we are concerned not only with constructing approximations suitable for *preconditioning* but also in finding a way of representing dense matrices that are too large to be stored and manipulated directly.

Given a matrix A of order $n = p^2$, we want to approximate it by a matrix B of the form (2) with r as small as possible while the error satisfies (3) with a prescribed relative error bound ε. We will refer to B as a matrix of low Kronecker rank.

Prior to construction of algorithms, one would like to have some *existence theorems* stating which classes of matrices are guaranteed to possess approximations of low Kronecker rank. The first existence theorems of this kind are proposed in [?] and then extended to the case of three and more Kronecker factors in [?].

These results assume that matrices are associated with a function $f(z', z)$ of two vectors $z' = (x', y')$ and $z = (x, y)$ with real coordinates, and it holds that

$$f(z', z) = F(u, v), \qquad u = x' - x, \quad v = y' - y, \tag{13}$$

where $F(u, v)$ is such that any mixed derivative

$$D^m F = \frac{\partial^k \, \partial^l}{(\partial u)^k \, (\partial v)^l} \, F, \quad m = k + l,$$

satisfies the inequality

$$|D^m F| \le c \, d^m \, m! \, \rho^{g-m}, \quad \rho = \sqrt{u^2 + v^2} \ne 0, \tag{14}$$

with some real constants $c, d > 0$ and g. Such an F is called a *complete asymptotically smooth* function.

[?] Let $A = [f(z^i, \, z^j)]$ be a matrix of order $n = pq$, where $z^i = (x_{k(i)}, \, y_{l(i)})$ with integer $k(i), l(i)$ defined by

$$i = (k(i) - 1)q + l(i), \quad 1 \le k(i) \le p, \quad 1 \le l(i) \le q$$

and one-dimensional grids

$$0 \le x_1 < \ldots < x_p \le 1, \quad 0 \le y_1 < \ldots < y_q \le 1.$$

Assume that f satisfies (13), where F is a complete asymptotically smooth function. Define the minimal step size for the one-dimensional grids

$$h = \min\{ \min_{\substack{1 \le k,k' \le p \\ k \ne k'}} |x_k - x_{k'}|, \ \min_{\substack{1 \le l,l' \le q \\ l \ne l'}} |y_l - y_{l'}| \}, \tag{15}$$

and let γ be an arbitrary number such that $0 < \gamma < 1$.

Then, for any $m = 1,\ 2,\ \ldots$, there exists a matrix B of the form (2) with the following estimates on the number of summands r and approximation error:

$$r \le \left(c_0 + c_1 \ \log h^{-1}\right) \ m, \tag{16}$$

$$|\{A - A_r\}_{ij}| \le c_2 \ \gamma^m \ ||z^i - z^j||^g, \quad 1 \le i,j \le n, \tag{17}$$

where c_0, c_1 and c_2 are positive constants depending on γ, and 0^g is to be set to 0 for any g.

Note that the matrix A from Example1.1 satisfies the hypotheses of this theorem [**?**]. Below we return to the assumption that $p = q$.

Given A, minimization of r for a given ε can be done via SVD applied to a matrix $\mathcal{P}(A)$ defined as follows:

$$\mathcal{P}(A) = [\mathcal{V}(A_{11}), \mathcal{V}(A_{21}), \ \ldots \ , \mathcal{V}(A_{pp})]^T ,$$

provided that

$$A = \begin{bmatrix} A_{11} & \ldots & A_{1p} \\ \ldots & \ldots & \ldots \\ A_{p1} & \ldots & A_{pp} \end{bmatrix},$$

and $\mathcal{V}$ maps a matrix to a vector of its entries taken column by column. We make use of the following observation [**?**]:

$$\mathcal{P}\left(\sum_{k=1}^{r} U_k \otimes V_k\right) = \sum_{k=1}^{r} (\mathcal{V}(U_k))(\mathcal{V}(V_k))^T, \tag{18}$$

$$\left\| A - \sum_{k=1}^{r} U_k \otimes V_k \right\|_F = \left\| \mathcal{P}(A) - \sum_{k=1}^{r} (\mathcal{V}(U_k))(\mathcal{V}(V_k))^T \right\|_F . \tag{19}$$

Consequently, the best Frobenius norm approximation of the form (2) can be computed by the standard SVD method applied to $\mathcal{P}(A)$.

In Fig. 1 one can see the portraits of “large” entries of A and $\mathcal{P}(A)$. The original matrix A is from Example 1.1. Its large entries are located around the main diagonal whereas those of $\mathcal{P}(A)$ are "uniformly distributed" over the whole matrix suggesting that it could be close to a low-rank matrix.

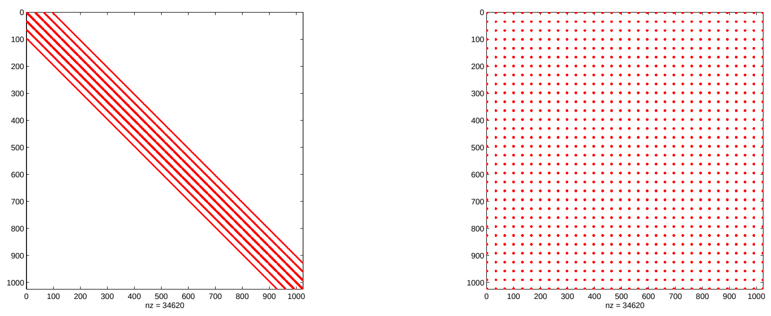

Figure 2.1 Portraits of A and $\mathcal{P}(A)$ for $p = 32, n = 1024$.

The application of SVD to $\mathcal{P}(A)$ shows that it can be approximated by a matrix of rank $r = 9$ with the relative Frobenius norm error $4.5 \cdot 10^{-6}$. It follows that A can be approximated by the sum of $r = 9$ Kronecker products with the same error.

Since we obviously cannot afford SVD for large matrices, we have to seek an alternative. One possibility is the Lanczos bidiagonalization algorithm (see [**?**]). It is considerably less expensive than SVD and almost equally reliable, but, as matrix dimensions increase, it also becomes time-consuming because we have to recompute all the entries of A on each Lanczos iteration.

As a matter of fact, the very computation of all the entries of A becomes expensive for large dimensions and so we cannot afford this either.

Again, we need a kind of "existence theorem" that claims that a low-rank approximation to a matrix of order n can be *reliably* obtained if we use only $O(n)$ entries in some, appropriately chosen, positions. Such a claim is proved in [**?**]. The case would be closed if we knew which positions to take. A useful precise answer to this question is as follows.

[**?**] Let M be of order n and with the singular values $\sigma_1 \geq \ldots \geq \sigma_n$. Suppose that M is a block matrix of the form

$$\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix},$$

where M_{11} is nonsingular, $r \times r$, and of maximal volume (determinant in modulus) among all $r \times r$ submatrices. Then

$$|\{M_{22} - M_{21} M_{11}^{-1} M_{12}\}_{ij}| \leq (r+1)\,\sigma_{r+1}, \quad 1 \leq i, j \leq n.$$

Let the distance between M and matrices of rank r in the spectral norm be equal to or less than ε. Then a reliable rank r approximation to M can be obtained from the entries of r columns and rows whose intersection contains the submatrix of maximal volume among all $r \times r$ submatrices in M. The entry-wise error of this approximation is estimated from above by $(r+1)\varepsilon$.

In our case $M = \mathcal{P}(A)$. Thus, we need to choose an appropriate cross of r columns and rows from $\mathcal{P}(A)$. In this enterprise, the maximal-volume principle serves as an ultimate target that should be approached yet might be never achieved. One possible strategy for selecting this cross is proposed in [?]. However, it turns out that in many cases of practical interest it can be simplified and reduced, in effect, to the classical LU decomposition with selection of pivots in certain subsets of the so-called 'active submatrices'. We compute only those columns and rows that correspond to pivots and stop when the next pivot becomes smaller than a preset dropping tolerance. Admittedly, this is not a completely reliable method since we do not compute all the entries of M. However, some justification for a similar algorithm is given in [?], and in all our experiments it has been found to work well for function-related matrices such as those of Theorem 2.

Algorithm 2.1 (Incomplete Cross Approximation Algorithm)
Given a function $\mathcal{M}(i,j)$ of two indices $1 \le i,j \le n$ and a dropping tolerance ε, compute the vectors $u_1, v_1, \ \dots \ , u_r, v_r$ of size n with r close to the smallest possible one such that

$$||M - \sum_{k=1}^{r} u_k v_k^T||_F \ \le \ \tilde{\varepsilon} \ ||M||_F,$$

where

$$M = [m_{ij}], \quad m_{ij} = \mathcal{M}(i,j), \quad 1 \le i,j \le n,$$

and $\tilde{\varepsilon} \approx \varepsilon$.

1. Set $\delta = \varepsilon$, $k = 1$ and $\mathcal{I}[1:n] = \mathcal{J}[1:n] = [1, 2, \dots, n]$.

2. Compute $m_{ij} = \mathcal{M}(i,j)$ for

$$(i,j) \in \mathcal{P}_k = \{(\mathcal{I}(k), \mathcal{J}(k)), (\mathcal{I}(k+1), \mathcal{J}(k+1)), \ \dots \ , (\mathcal{I}(n), \mathcal{J}(n)\}$$

   and set

$$m'_{ij} = m_{ij} - \sum_{l=1}^{k-1} u_l(i) v_l(j).$$

3. Find (i'_k, j_k) such that

$$|m'_{i'_k j_k}| = \max_{(i,j) \in \mathcal{P}_k} |m'_{i,j}|.$$

4. Compute $m_{i,j_k} = \mathcal{M}(i, j_k)$ for all

$$i \in \mathcal{I}_k = \{\mathcal{I}(k), \mathcal{I}(k+1), \ \dots \ , \mathcal{I}(n)\}$$

   and set

$$m'_{ij_k} = m_{ij_k} - \sum_{l=1}^{k-1} u_l(i) v_l(j_k).$$

5. Find i_k such that
$$d_k \equiv |m'_{i_k j_k}| = \max_{i \in \mathcal{I}_k} |m'_{i j_k}|.$$
(Now, the pivot on the kth step is to be in position (i_k, j_k) of M).

6. If d_k is less than the machine precision then stop. Otherwise, compute
$$\tilde{\varepsilon} = d_k(n-k)/||\sum_{l=1}^{k-1} u_l v_l^T||_F$$
and if $\tilde{\varepsilon} \le \varepsilon$, set $r = k-1$ and quit.

7. Compute
$$m_{i_k j} = \mathcal{M}(i_k, j) \text{ for } j = 1, \ldots, n,$$
$$\alpha = m_{i_k j_k}/\sqrt{|m_{i_k j_k}|}, \quad \beta = \sqrt{|m_{i_k j_k}|},$$
$$u_k(i) = m_{i j_k}/\alpha, \quad i = 1, \ldots, n,$$
$$v_k(j) = m_{i_k j}/\beta, \quad j = 1, \ldots, n.$$

8. Swap the entries in positions k and l in $\mathcal{I}$, where l is such that $\mathcal{I}(l) = i_k$. Swap the entries in positions k and l in $\mathcal{J}$, where l is such that $\mathcal{J}(l) = j_k$.

9. If $k < n$, set $k \leftarrow k+1$ and go to Step 2.

In this algorithm, one can recognize the familiar LU decomposition in which the pivots are sought in a subset of entries of the "active submatrix". This subset is small compared to the whole set but larger than the single column used in the column pivoting strategy. If we set $\epsilon = 0$ the algorithm becomes equivalent to LU factorization with column pivoting and, provided M is non-singular, it terminates in n steps to yield an exact LU decomposition.

Unlike the classical LU, we make the corresponding diagonal entries in L and U be equal in modulus (Step 7). Thus, for symmetric positive definite matrices we come up with the Cholesky decomposition. In the general case, we find it useful to equalize somewhat the norms of u_k and v_k.

The most important difference between the above algorithm and the classical LU is that the columns and rows are not assumed to be stored at the outset. Each new row and column pair is computed and stored when the corresponding pivot has been chosen from some precomputed entries. The choice of the latter entries is based on a heuristic rule. This rule can be provided by the user and may differ from the one we use above. Note that each new pivot is chosen with a view to maximizing the volume of the updated cross intersection submatrix under certain restrictions on the update data.

Concerning Step 6, note that the norm $||\sum_{l=1}^{k} u_l v_l^T||_F$ is computed recursively in successive summands without having to compute all the entries of the corresponding matrix.

Table 2.1 confirms the robustness of the Incomplete Cross Approximation Algorithm. The input matrix A is from Example 1.1 and ε is set to 10^{-5}. In the table, $\tilde{\varepsilon}$ is the relative accuracy estimate computed by the algorithm. It should approximate $||A - B||_F/||A||_F$ and, as we see, it does so fairly reliably.

n	256	1 024	4 096	16 384	65 536
r	8	10	11	14	15
$\tilde{\varepsilon}$	$3.3 \cdot 10^{-6}$	$3.3 \cdot 10^{-6}$	$3.3 \cdot 10^{-6}$	$1.8 \cdot 10^{-6}$	$6.5 \cdot 10^{-6}$
$\|\|A - B\|\|_F/\|\|A\|\|_F$	$2.9 \cdot 10^{-6}$	$2.6 \cdot 10^{-6}$	$6.4 \cdot 10^{-6}$	$2.2 \cdot 10^{-6}$	$4.7 \cdot 10^{-6}$

Table 2.1 Verification of the Incomplete Cross Approximation Algorithm.

Note that the approximation time for $n = 65\ 536$ was 3.6 sec. whereas a single computation of all the entries of A took 565.5 sec. (on a Pentium 1600 notebook).

The time required by Algorithm 2.1 depends on r. As we observe from typical examples, $r \leq 20$ even for n of the order of 1 million. Consequently, the time for Step (A) is next to negligible in comparison with Steps (C) and (D). Table 2.2 shows the times for the same matrix of Example 1.1 (on an AMD 1000 computer with 1 Gbyte operative memory).

n	16 384	65 536	262 144	1 048 576
TIME (sec.)	0.6	2.7	14.6	61.5

Table 2.2 Timings for the Incomplete Cross Approximation Algorithm.

Thus, from the practical point of view, construction of the approximation $B \approx A$ by Algorithm 2.1 is fast and reliable (at least for the test matrices of Example 1.1). However, the matrix-vector multiplication procedure using the Kronecker-product structure of B is still time-consuming. Also, we need a preconditioner to reduce the number of GMRES iterations with A or B. Therefore, we have to consider further approximation steps, (B) and (C).

3 Approximation by sparse matrices in a wavelet basis

From now on we concentrate on the solution procedure for $Bx = b$, where B is a well-structured matrix of the form $B = \sum\limits_{k=1}^{r} U_k \otimes V_k$ with U_k and V_k being stored in the computer memory. Matrices U_k and V_k are of order p which can be of order of several thousands (however, for $n \leq 1\ 000\ 000$ we have $p \leq 1\ 000$). We propose that these U_k and V_k undergo what is called *wavelet compression*. The purpose of this is two-fold:

- to reduce the matrix-vector multiplication costs;
- to set the stage for construction of data-sparse preconditioners.

In typical applications, the U_k and V_k cannot easily be approximated by simple thresholding (i.e. setting to zero 'small' entries) because most of the entries are of similar magnitude. However, the divided differences of the entries (considered as the values of a function on the integral grid) may differ in magnitudes and those of certain orders may become sufficiently small. For such cases, a discrete Wavelet Transform (DWT) offers a way of separating the information contained in a 'smooth' matrix into blocks of entries corresponding to weighted averages and weighted differences of entries in the original matrix. The weighted average entries will be large in magnitude compared with the weighted difference entries enabling a sparse approximation to the transformed matrix to be obtained by thresholding. This technique has been used to provide preconditioners both for dense matrices and for some sparse matrices (see, for example, [?, ?, ?, ?, ?, ?, ?, ?]). A typical scenario is that of large divided differences close to the main diagonal with rapid decay as the distance from the diagonal increases. It is this 'diagonal singularity' that produces the familiar 'finger' pattern of pseudo-sparsity in the transformed matrix.

Let us apply the Daubechies wavelet transform with μ vanishing moments [?]. It is defined by its 'low-pass' filter coefficients h_i, $i = 0, \ldots, 2\mu - 1$, chosen so that

$$\sum_{i=0}^{2\mu-2l-1} h_i h_{i+2l} = \begin{cases} 1, & l = 0, \\ 0, & l \geq 1. \end{cases} \tag{20}$$

Then, the 'high-pass' filter coefficients are of the form

$$g_i = (-1)^i h_{2\mu-1-i}, \; i = 0, \ldots, 2\mu - 1. \tag{21}$$

The requirement of μ vanishing moments means that

$$\sum_{i=0}^{2\mu-1} g_i i^s = 0, \quad s = 0, \ldots, \mu. \tag{22}$$

In order to apply the transform to a *finite* length vector we *periodize* the transform. That is, for a vector x of length n we extend x to be an infinite periodic vector X of period n and compute the transformed vector $\tilde{x}$ as the first n components of the infinite transformed vector $\tilde{X}$. Set $m = 2\mu$ and, taking an even

$N \geq m$, define an $N \times N$ matrix of the following form:

$$\Phi_N = \left(\begin{array}{cccccccccccccc}
h_0 & h_1 & h_2 & \cdots & h_{m-1} & 0 & \cdots & & & & & & \cdots & 0 \\
0 & 0 & h_0 & h_1 & \cdots & \cdots & h_{m-1} & 0 & \cdots & & & & \cdots & 0 \\
\vdots & & & & \ddots & \ddots & \ddots & \ddots & & & & & & \vdots \\
 & & & & & & \ddots & \ddots & \ddots & \ddots & & & & \\
\vdots & & & & & & & & \ddots & \ddots & \ddots & & & \vdots \\
h_4 & h_5 & \cdots & h_{m-1} & 0 & \cdots & & & \cdots & 0 & h_0 & h_1 & h_2 & h_3 \\
h_2 & h_3 & h_4 & h_5 & \cdots & h_{m-1} & 0 & \cdots & \cdots & & \cdots & 0 & h_0 & h_1 \\
g_0 & g_1 & g_2 & \cdots & g_{m-1} & 0 & \cdots & & & & & & \cdots & 0 \\
0 & 0 & g_0 & g_1 & \cdots & \cdots & g_{m-1} & 0 & \cdots & & & & \cdots & 0 \\
\vdots & & & & \ddots & \ddots & \ddots & \ddots & & & & & & \vdots \\
 & & & & & & \ddots & \ddots & \ddots & \ddots & & & & \\
\vdots & & & & & & & & \ddots & \ddots & \ddots & & & \vdots \\
g_4 & g_5 & \cdots & g_{m-1} & 0 & \cdots & & & \cdots & 0 & g_0 & g_1 & g_2 & g_3 \\
g_2 & g_3 & g_4 & g_5 & \cdots & g_{m-1} & 0 & \cdots & & & \cdots & 0 & g_0 & g_1
\end{array} \right).$$

Notice the wrap-round of the filter coefficients as a result of periodization. By virtue of (20) and (21), Φ_N is an orthogonal matrix. When Φ_N is multiplied by a vector, the rows with h_i correspond to the weighted averages of its components while those with g_i correspond to the weighted differences of the same components. Then, the k level transform $W^{(k)}$ is defined as the product

$$W^{(k)} = \Psi^{(k)} \dots \Psi^{(1)} \tag{23}$$

with

$$\Psi^{(k)} = \begin{pmatrix} \Phi_{2[p/2^k]} & 0 \\ 0 & I_{p-2[p/2^k]} \end{pmatrix}. \tag{24}$$

Note that the above definition does not assume that p is a power of 2.

If $x_i = f(i)$ for a polynomial f of order μ, then, due to (22),

$$\sum_{i=0}^{m-1} g_i f(j+li) = 0, \quad j = 0, \dots, p - l(m-1).$$

Consequently, most of the weighted differences (except those that come from the wrap-round effect) are equal to zero. In this case, the divided differences of f of order μ are equal to zero (see, for example, [?]). In general, small divided differences of order $k \leq \mu$ signal that the kth derivatives at certain points are small and, under some assumptions, f can be proved to be close to a polynomial of order k. This results in the pseudo-sparsity of the transformed vector.

The matrix level k wavelet transform of a $p \times p$ matrix Z is defined to be

$$\tilde{Z} = WZW^T, \quad W = W^{(k)}. \tag{25}$$

This is equivalent to performing the level k vector DWT on each of the rows and the columns of Z. In what follows, we replace W with the same matrix with the columns in the reverse order. Clearly it makes no difference for the matrix compression purposes but turns out to be useful for construction of sparse factorized preconditioners.

Now, consider $B = \sum\limits_{k=1}^{r} U_k \otimes V_k$ with dense $p \times p$ matrices U_k and V_k. Applying an orthogonal discrete wavelet transform (DWT) to the U_k and V_k we obtain $P_k = W U_k W^T$ and $Q_k = W V_k W^T$. Clearly,

$$\sum_{k=1}^{r} P_k \otimes Q_k = (W \otimes W)\, B \left(W^T \otimes W^T\right).$$

Choosing a threshold $\tau > 0$, we approximate P_k and Q_k by sparse matrices P_k^τ and Q_k^τ. It can be verified that

$$\|\sum_{k=1}^{r} P_k^\tau \otimes Q_k^\tau - \sum_{k=1}^{r} P_k \otimes Q_k\|_F \le \varepsilon_W \|\sum_{k=1}^{r} P_k \otimes Q_k\|_F, \tag{26}$$

where

$$\varepsilon_W = \varepsilon_W(\tau) = \frac{\sum\limits_{k=1}^{r} \left(\|P_k - P_k^\tau\|_F \|Q_k\|_F + \|P_k\|_F \|Q_k - Q_k^\tau\|_F\right)}{\|\sum\limits_{k=1}^{r} P_k \otimes Q_k\|_F} \tag{27}$$

is easy to compute. By orthogonality of the DWT, we have

$$\|C - B\|_F \le \varepsilon_W \|B\|_F.$$

If B is an approximation of A such that

$$\|B - A\|_F \le \varepsilon_K \|A\|_F, \tag{28}$$

it is easy to see that

$$\|C - A\|_F \le (\varepsilon_K + \varepsilon_W + \varepsilon_k \varepsilon_W)\, \|A\|_F. \tag{29}$$

Hence, by appropriate choices of r and τ, steps (A) and (B) of Section 1 enable us to approximate A to any required degree of accuracy.

In the wavelet compression of Step (B) we begin with some $\tau = \tau_0$ and then diminish it (currently by the rule $\tau_k = \tau_{k-1}/4$) until $\varepsilon_W(\tau)$ becomes of the same level as the error estimate of Step (A). The initial threshold is set to be $\tau_0 = \gamma a_{\max}$ where $a_{\max}$ is the maximal in modulus entry in the U_k and V_k and $0 < \gamma \le 1$ is a preset control parameter. The important outcome of this procedure is the compression factor (the ratio of the total number of nonzero entries in the U_k^τ and V_k^τ and the total number of entries in A). In Table 3.1 we show the wavelet compression results for the matrix of Example 1.1 of order $n = 65\ 536$, using the Daubechies DWT of order 8.

In this example, the compression factor of Step (A) was `0.0003967` and the relative accuracy estimate was $\varepsilon_K = 6.4 \cdot 10^{-5}$. On the output of Step (B) we have the compression factor `0.00007169`, which is a significant improvement of the one from Step (A). The final accuracy estimate $\varepsilon_W = 5.8 \cdot 10^{-5}$ is of the same order as the estimate of approximation by the sum of the Kronecker products (ε_W).

However, it is not the compression factor itself that is the principal target to consider in Step (B). The main purpose here is to reduce the matrix-vector multiplication costs. Let the number of nonzero entries in all the P_k^τ and Q_k^τ matrices be ν. Then D (and hence C) can be multiplied by a vector in $O(\nu\sqrt{n})$ operations. In Table 3.1 we also give the complexity reduction factor which is the ratio of the reduced matrix-vector multiplication costs and the same costs for the original matrix A.

τ	Compression factor	Complexity reduction factor	ε_W
1.513253	0.00000179	0.00045836	0.14160909
0.378313	0.00000600	0.00153661	0.04432408
0.094578	0.00001376	0.00352216	0.01161883
0.023645	0.00002444	0.00625551	0.00331343
0.005911	0.00003830	0.00980413	0.00089616
0.001478	0.00005413	0.01385605	0.00021871
0.000369	0.00007169	0.01835191	0.00005751

Table 3.1 Wavelet compression of Step (B).

Instead of using the same τ for each of the P_k and Q_k we could have opted for choosing individual thresholds. However, since the norms of P_k and Q_k differ considerably for different k, one has to be careful in this choice lest the sparsity for the matrices of smaller norms is lost.

4 Data-sparse preconditioners

Once we have computed the P_k^τ and Q_k^τ, we are in a position to solve $Ax = b$ by using an iterative method, such as GMRES, applied to the approximate equation $Cx = b$. This requires only that we are able to perform matrix-vector multiplication with $C = (W^T \otimes W^T)D(W \otimes W)$, where $D = \sum\limits_{k=1}^{r} P_k^\tau \otimes Q_k^\tau$. The 2D DWT matrices $W \otimes W$ and $W^T \otimes W^T$ are multiplied by a vector in $O(n)$ operations. It implies that the matrix-vector multiplication costs for C are determined by the same costs for D. Overall we multiply C by a vector in $O(\nu\sqrt{n})$ operations, where ν is the total number of nonzero entries in the P_k^τ and Q_k^τ matrices.

Let A be from Example 1.1 of order $n = 1\ 048\ 576$. Application of PCG to solve $Cx = b$ took 186 iterations and 199.58 minutes (on an AMD-1000 computer). The spectral condition number of C was found (from the corresponding Ritz values) to be 6737.4. Thus, we are interested to reduce this condition number, and hence the number of iterations, by choosing an appropriate preconditioner. We consider the following two options:

- Incomplete LU preconditioner.
- Inverse Kronecker product preconditioner.

4.1 Incomplete LU preconditioner

Although D has a fairly large proportion of zero entries, it is not sparse enough to be stored in memory in any conventional storage scheme for sparse matrices (for example, the Compressed Row Format or the Compressed Column Format [?]). To form a preconditioner, we first increase the threshold $\delta > \tau$ for P_k^δ and Q_k^δ to increase the number of zeroes so that the matrix

$$E = \sum_{k=1}^{r} P_k^\delta \otimes Q_k^\delta$$

can be stored as an explicit sparse matrix.

The choice of the threshold is now different from the one we had at the wavelet compression stage of Step (B). Then we had to maintain the approximation error at the level of that obtained at Step (A). Now we begin with some $\delta_0 = \gamma a_{\max}$, where $0 < \gamma < 1$ is a preset control parameter and a_{max} is the maximal in modulus entry in the P_k and Q_k matrices. Then we *increase* δ (currently by the rule $\delta_k = 2\delta_{k-1}$) until the compression factor reaches the prescribed level. In the example below we use the Daubechies DWT of order 8.

In Table 4.1 we show the results of the above thresholding for A of order $n = 65\ 536$ from Example 1.1. The compression factor of Step (A) is $f_A = 3.9673 \cdot 10^{-4}$ and we agree to satisfy with the compression factor f_E for E when it becomes less than $\tilde{f} = c_E f_A$ with some preset $c_E \geq 1$. In this particular case we take $c_E = 2.5$ and obtain $\tilde{f} = 9.9182 \cdot 10^{-4}$ In the last column we output the estimate for the relative Frobenius-norm approximation error of this sparsification procedure.

δ	f_E	Error estimate
0.015133	0.01850058	0.00257633
0.030265	0.01384041	0.00494958
0.060530	0.00978269	0.00971694
0.121060	0.00669412	0.01966431
0.242121	0.00385682	0.03909217
0.484241	0.00241708	0.07117478
0.968482	0.00096515	0.14700328

Table 4.1 Sparsification preliminaries for the incomplete LU decomposition.

Once E has been stored, we apply the incomplete LU decomposition algorithm with the dynamic choice of fill-in depending on the preset threshold ε_{LU} (the so-called ILUT, in the terminology of [?]).

For the above example, the ILUT threshold $\varepsilon_{\rm LU}$ was set to 0.01. In this case we use the symmetry and positive definiteness of the matrix and apply the incomplete Cholesky algorithm, a special variant of the LU decomposition with $L = U^T$. The compression factor for L versus A was found to be $4.902 \cdot 10^{-4}$. On Fig. 4.1 we show the sparsity portraits of E and the incomplete Cholesky factor U.

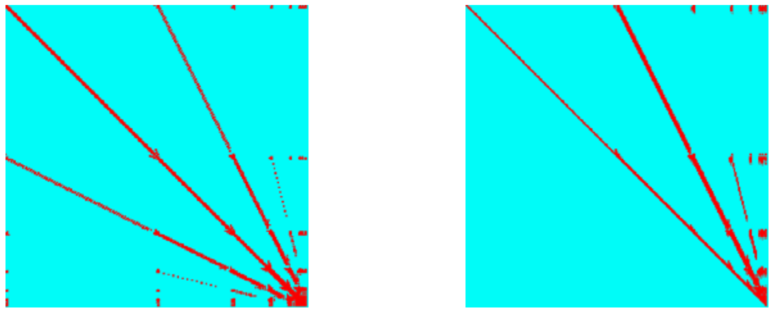

Figure 4.1 Matrix and its incomplete Cholesky factor ($n = 65\,563$).

In Table 4.2 we can see the performance of CG with no preconditioner. The timing was done on an AMD-1000 computer with 1 Gbyte operative memory.

n	16 384	65 536	262 144
r	12	13	16
CG time	14.7 sec	106.7 sec	1968.0 sec
Number of iterations	61	90	129
$\mathrm{cond}_2(C)$	824.455	1667.53	3344.93

Table 4.2 CG with no preconditioner for matrices of Example 1.1.

In Table 4.3 we record the performance of the PCG with the ILUT preconditioner for matrices of different orders for the same Example 1.1. Note that we were not able to proceed with larger sizes because the memory was insufficient for construction of the ILUT preconditioner.

n	16 384	65 536	262 144
ILUT construction time	2.3 sec	37.7 sec	86.1 sec
PCG time	2.5 sec	10.0 sec	163.6 sec
Number of iterations	8	6	9
$\mathrm{cond}_2(CF^{-1})$	3.852	2.435	5.638

Table 4.3 Performance of the ILUT preconditioner for matrices of Example 1.1.

4.2 Inverse Kronecker product preconditioner

One may expect that a good preconditioner could be built up from approximations to A of smaller Kronecker rank than B. To this end, we may consider matrices

$$B_l = \sum_{k=1}^{l} U_k \otimes V_k, \quad 1 \le l < r, \tag{30}$$

and especially the simplest case corresponding to $l = 1$. A remarkable advantage of the simplest case is that B_1 can be inverted explicitly at low costs as

$$B_1^{-1} = U_1^{-1} \otimes V_1^{-1}. \tag{31}$$

Applying the DWT to the U_1^{-1} and V_1^{-1} we obtain

$$S = WU_1^{-1}W^T, \quad T = WV_1^{-1}W^T,$$

then approximate S and T by their sparsified counterparts S^δ and T^δ, respectively, and finally come up with an *explicit* preconditioner of the form

$$F^{-1} = (W^T \otimes W^T)(S^\delta \otimes T^\delta)(W \otimes W). \tag{32}$$

The threshold δ can be chosen as $\delta = \gamma a_{\max}$ where $0 < \gamma < 1$ is a preset control parameter and $a_{\max}$ is the maximal in modulus entry in the S and T.

In Table 4.4 we present the results for matrices of different orders from the same Example 1.1 on an AMD-1000 computer with 1 Gbyte operative memory.

n	16 384	65 536	262 144	1 048 576
r	12	13	16	16
$\tilde{\varepsilon}$	$3.8 \cdot 10^{-5}$	$6.4 \cdot 10^{-5}$	$3.1 \cdot 10^{-5}$	$6.8 \cdot 10^{-5}$
Matrix-by-vector time	0.2 sec	1.2 sec	15.4 sec	48.3 sec
Number of iterations	18	22	26	35
IKP construction time	0.2 sec	1.3 sec	11.1 sec	77.4 sec
Time for the PCG	4.7 sec	28.8 sec	7.14 min	30.33 min
$\mathrm{cond}_2(CF^{-1})$	32.59	53.21	90.44	159.77
Relative solution error	$1.2 \cdot 10^{-4}$	$1.8 \cdot 10^{-4}$	$9.6 \cdot 10^{-5}$	$3 \cdot 10^{-4}$

Table 4.4 Numerical results for matrices of Example 1.1 using IKP preconditioner.

The right-hand side b was set to be the sum of the first, fifth and tenth columns of A, so we know the exact solution of $Ax = b$ and can report on the resulting accuracy in the most reliable way. The accuracy control parameter ε was set to 10^{-4}, the same as the residual reduction parameter for PCG. The threshold control parameter γ for sparsification of S and T was set to 0.04.

If we apply IKP without wavelet sparsification of the Kronecker factors, the number of iterations is smaller but the overall time becomes considerably higher, because of the multiplication complexity (see Table 4.5).

n	16 384	65 536	262 144
Number of iterations	15	20	26
IKP construction time	0.1 sec	1.2 sec	10.6 sec
Time for the PCG	5.1 sec	70.5 sec	32.35 min

Table 4.5 IKP preconditioner without wavelet sparsification.

4.3 Implementation remarks

Our solution method for $Ax = b$, as set out above, requires that we store the approximate matrix D (in the form of a sum of Kronecker products) and use it to solve the approximate equation

$$\left(W^T \otimes W^T\right) D \left(W \otimes W\right) x = b.$$

At each iteration, this involves transforming a vector into the wavelet basis and transforming another vector back to the standard basis. Although the cost is not high ($\mathcal{O}(n)$ operations), it is not negligible and is unnecessary. The cost of each iterative step can be reduced by transforming the entire equation into the wavelet basis, solving it there and then transforming the solution vector back to the standard basis. In other words, we solve the transformed equation

$$D\tilde{x} = \tilde{b}, \quad \tilde{x} = \left(W \otimes W\right) x, \quad \tilde{b} = \left(W \otimes W\right) b, \tag{33}$$

and then obtain $x = \left(W^T \otimes W^T\right) \tilde{x}$ by applying an inverse DWT. Thus we now only perform one vector DWT at the start of the iteration process and one inverse DWT at the end. Similarly, when a preconditioner $\tilde{F}$ (in the wavelet basis) is used, we solve

$$D\tilde{F}^{-1}\tilde{y} = \tilde{b}, \quad \tilde{y} = \tilde{F}\tilde{x}, \quad \tilde{x} = \left(W \otimes W\right) x, \quad \tilde{b} = \left(W \otimes W\right) b. \tag{34}$$

The orthogonality of the DWT ensures that the residual norm is the same in both the wavelet and the standard basis.

5 More examples and discussion

In several places above, we have already reported on very promising numerical results obtained for matrices of Example 1.1. Consistent with theory proposed in [**?**, **?**], the Kronecker product approximation method works well for these matrices. Moreover, both of the data-sparse preconditioners proposed in this paper perform well. Construction of the ILUT preconditioner takes longer, but it leads to a smaller number of iterations. The IKP preconditioner is easier to construct but the number of iterations in this case is larger. The overall time is also greater than for ILUT, but only slightly. However, the main drawback of the ILUT preconditioner is that it *requires more memory* than the IKP preconditioner.

To give more examples, consider a somewhat artificial extension of Example 1.1 as follows.

Example 5.1 *For any $\alpha \geq 0$, define $A(\alpha) = [a_{ij}(\alpha)]$ of order $n = p^2$ by*

$$a_{ij}(\alpha) = \begin{cases} 2p^{\alpha}, & i = j, \\ 1/||z^i - z^j||^{\alpha}, & i \neq j, \end{cases} \tag{35}$$

where the nodes z^i are the same as those in Example 1.1. For brevity, we refer to $A(\alpha)$ as the α-matrices.

All the α-matrices are symmetric. For $n = 4\ 096$, we have observed numerically that they are positive definite for $0 \leq \alpha \leq 1.6$ and for $\alpha \geq 1.7$ they become indefinite. [3] Thus, for $0 \leq \alpha \leq 1.6$ we may apply CG.

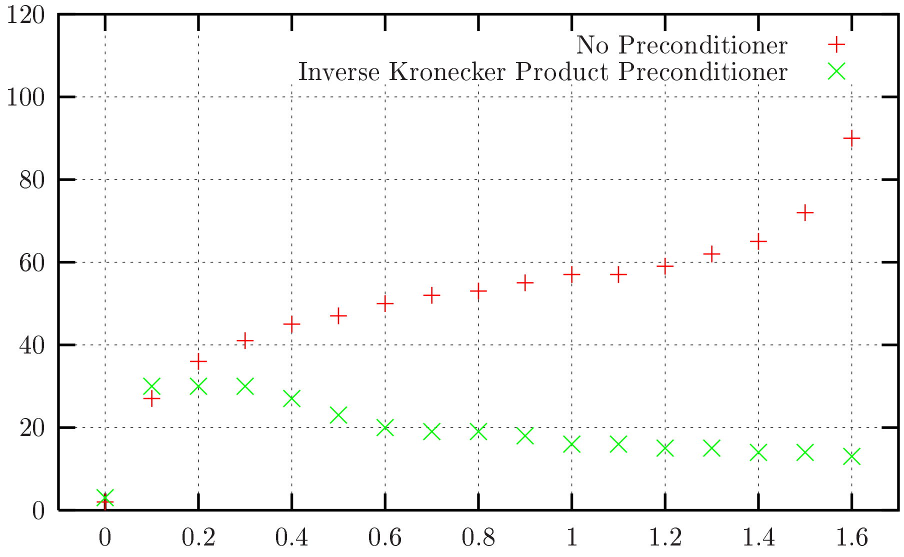

Figure 5.1 The number of CG iterations against α ($n = 4\ 096$).

The approximation error control and residual reduction parameters were set to 10^{-5}. The relative accuracy of the computed solution was found to be of order 10^{-5}. The Kronecker rank was 2 for $\alpha = 2$ and from 10 to 12 for other values of α.

In Fig. 5.1. we plot the number of unpreconditioned and IKP preconditioned CG iterations against α. In Fig. 5.2 we show the estimates for the spectral condition number computed from the Ritz values. Obviously, the IKP preconditioner considerably reduces the condition number. However, the smallest number of iterations (just 2) is observed with the largest condition number! This is no surprise, in fact, as it is not only the condition number that counts, and in the case $\alpha = 0$ we have only two different eigenvalues. For small α a relatively small number of iterations is best effected by a specific distribution of the eigenvalues, the eigenvalue cluster around 1 (see [**?**]).

[3] A proof and a general theorem on this phenomenon would be interesting.

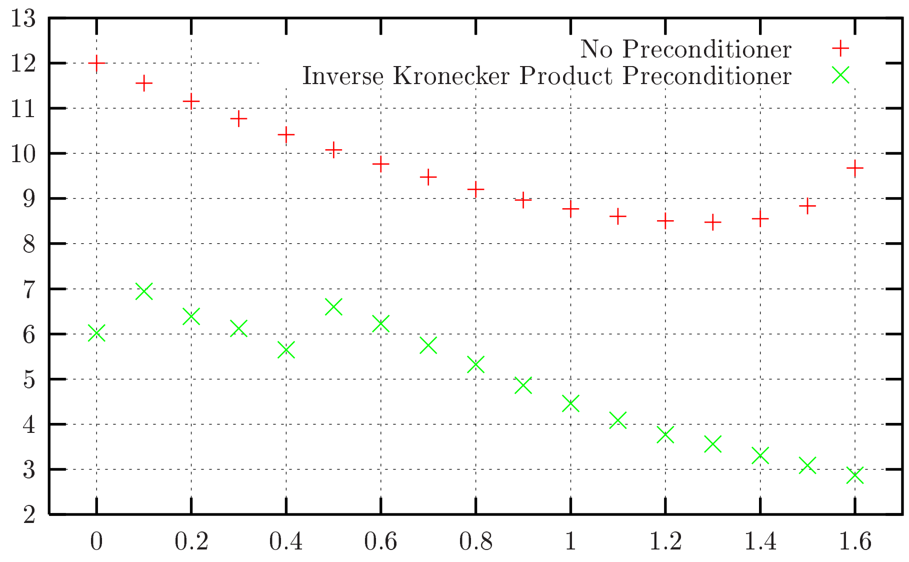

Figure 5.2 Spectral condition number in $\log_2$ scale against α.

For $\alpha \geq 1.7$ we apply GMRES with a maximum of 600 iterations. In Table 5.1 we put the results for the IKP and ILUT preconditioners. As above, the approximation control and residual reduction parameters were set to 10^{-5}. Despite the previous results, in this range of α the solution accuracy falls by three orders and finally is of order of 10^{-2}.

α	1.7	1.8	1.9	2.0	2.1
No prec.	198	> 600	> 600	> 600	> 600
IKP	27	141	194	346	444
ILUT	231	125	154	182	202

Table 5.1 Number of GMRES iterations for the α-matrices ($n = 4\ 096$).

Example 5.2 *Define a_{ij} by (10) with the nodes (11) constructed from the nonuniform one-dimensional grids of the following form:*

$$\begin{aligned} x_\alpha &= \frac{1}{2}\left(1 - \cos\left(\frac{\pi}{p}(\alpha - \frac{1}{2})\right)\right), \quad \alpha = 1, \ldots, p, \\ y_\beta &= \frac{1}{2}\left(1 - \cos\left(\frac{\pi}{p}(\beta - \frac{1}{2})\right)\right), \quad \beta = 1, \ldots, p. \end{aligned} \tag{36}$$

The Kronecker ranks for matrices of Example 5.2 are found to be approximately the same as they were in the case of uniform grids. Now the matrices are not positive definite, so we apply GMRES. For the solution accuracy to be of order of 10^{-4} we have to set the residual reduction parameter to 10^{-6} (note that it was 10^{-4} in case of Example 1.1). The number of unpreconditioned iterations was larger than it was with uniform grids. Also, we found the ILUT preconditioner less impressive in this new case but still very useful. For construction of E we took $c_{\rm E} = 4.5$ and $\varepsilon_{\rm LU} = 0.005$.

n	2 025	3 600	5 625	8 100
r	14	16	18	17
$\tilde{\varepsilon}$	$7 \cdot 10^{-5}$	$4 \cdot 10^{-5}$	$4 \cdot 10^{-5}$	$7 \cdot 10^{-5}$
No prec.	353	571	> 900	> 900
ILUT	50	54	94	109

Table 5.2 GMRES iterations for matrices of Example 5.2. Wavelet order is 8.

In the case of nonuniform grids we have no difficulties at the stage of approximation by the sum of Kronecker products, even for very large matrices. However, the ILUT preconditioner now requires much more memory than it did with the uniform grids. In further research we need to find out if we can do with lesser memory in the ILUT and also look for alternatives. (The current version of IKP appeared inefficient.) We think that the currently observed difficulties may come from the application of the Daubechies wavelets that do not very well suit the case of nonuniform grids.

In Table 5.3 we compare the performance of ILUT with wavelets of different orders. The timing is made on a Pentium-1600 notebook (the first number in the brackets is time for the construction of preconditioner and the second is for GMRES iterations).

Wavelet order	$n = 2\ 025$	$n = 3\ 600$	$n = 5\ 625$	$n = 8\ 100$
2	49 (4.1+1.9)	68 (11.4+5.8)	157 (71.7+32.7)	129 (161.1+41.3)
4	29 (6.8+1.0)	54 (26.9+4.7)	110 (153.9+22.5)	90 (256.6+26.4)
8	50 (12.2+2.0)	54 (22.1+4.4)	94 (109.6+17.0)	109 (256.8+31.1)

Table 5.3 Number of iterations and timings (in sec.) for matrices of Example 5.2.

In conclusion, let us make a summary of the results and outline directions for future research.

We have presented a new way in which very large dense matrices derived from radial functions can be stored (in an approximate form) very compactly, by expressing them as a sum of the Kronecker products of much smaller matrices in a wavelet basis. This enables us to store and manipulate such matrices without the need for prohibitively large memory resources.

We have further demonstrated that solving linear systems of equations involving such matrices is feasible using an iterative method, and have presented two simple preconditioning strategies based firstly on ILU decomposition of a sparse approximation to the matrix and secondly on approximating the inverse of a single Kronecker product. In experiments, the overall solution time using the two methods was similar, with the ILUT approach being slightly faster. However, the extra memory required to store the ILUT preconditioner means that, if limited storage is an issue, the IKP approach may be feasible when ILUT is not.

These are by no means the only possible preconditioning approaches that could be tried, and future work will include investigation into alternative methods. In particular, the inverse of the wavelet-approximated matrix E can be expected to be pseudo-sparse, suggesting that it might be fruitful to seek an approximate inverse of this matrix. This would provide a better approximation to

E^{-1} than the IKP preconditioner described above. Such 'direct' preconditioners are of particular interest when parallel processing is employed, as may well be the case for very large matrices, because the construction and application of approximate inverse preconditioners can be readily parallelized. Preliminary experiments using a Newton iteration (see e.g. [**?**, §2.1]) to form an approximate inverse for E produced preconditioners that gave good convergence, but were expensive to compute. We anticipate that an alternative method of computing the approximate inverse (such as the factorized approximate inverses of [**?**, **?**, **?**], the polynomial preconditioners of [**?**] or the minimization approaches of [**?**, **?**]) and/or the choice of a better starting matrix for the Newton iteration could be expected to yield improved results. In particular, we have implemented versions of the Benzi, Meyer and Tuma algorithms [**?**, **?**] for computing factorized sparse approximate inverse preconditioners using sets of E-biconjugate vectors in which the matrix E is defined as a sum of Kronecker products. So far our results have been promising, but more work is needed to establish reliable criteria for dropping entries in the P_i and Q_i to give E and subsequently for dropping entries to maintain the sparsity of factors of the approximate inverse. we have also carried out some preliminary work using the Grote and Huckle algorithm [**?**], but so far have not found a way of combining good convergence with a reasonable cost for computing the preconditioner.

It may be possible to reduce further the cost of the IKP preconditioner by using the alternative 'DWTPerMod' transform (presented in [**?**]), in which the large entries are confined to an 'arrow' structure (a diagonal band and blocks at the bottom and right-hand edges) by permutation of the rows and columns of the transformed matrix. This substantially reduces fill-in under LU factorization.

At present we have only used wavelets from the Daubechies family. These have the virtue of being extremely easy to implement and their orthogonality ensures that a close approximation in the wavelet basis implies a close approximation in the standard basis. However, it is likely that improved sparse approximations (i.e. more sparse or giving closer approximation or both) could be obtained by using other wavelet transforms to approximate the Kronecker factors. In particular, using the 'Lifting' scheme of Sweldens [**?**] it may be possible to design biorthogonal wavelets with specific matrix approximations in mind. Another avenue that we intend to explore is the use of NS-forms [**?**, **?**], which offer an alternative way of solving linear systems in a wavelet basis.

We believe that we have laid the foundations for some exciting new developments in approximation and numerical solution of large dense matrix problems.

Acknowledgements

The authors would like to thank the anonymous referees and our colleagues Dr. David Silvester and Prof. Neville Ford for their careful reading of the draft of this paper and for their helpful suggestions.