WAVELET-BASED PRECONDITIONERS FOR DENSE MATRICES WITH NON-SMOOTH LOCAL FEATURES

JUDITH FORD[1*] and KE CHEN[2]

[1] *Department of Mathematical Sciences, University of Liverpool, Liverpool, L69 7ZL, UK. email: judyford@liv.ac.uk*

[2] *Department of Mathematical Sciences, University of Liverpool, Liverpool, L69 7ZL, UK. email: k.chen@liv.ac.uk*
Web: http//www.liv.ac.uk/maths/applied

Abstract.

We present algorithms for the detection of local non-smooth features within a dense matrix and show how, by isolating such features, we are able to use wavelet compression to design preconditioners for the corresponding dense linear system. We illustrate our approach with examples from the solution of Elastohydrodynamic Lubrication problems and Boundary Integral Equations.

AMS subject classification: 65F10, 65F50, 76D08, 45E05.

*Key words:*Preconditioning, GMRES, wavelet transform, dense matrices, elastohydrodynamic lubrication, boundary integral equations.

1 Introduction

We are concerned with the solution of large linear systems

$$Ax = b, \tag{1.1}$$

where A is a dense, non-symmetric matrix, smooth apart from a diagonal band and localized, off-diagonal, non-smooth features. We are motivated to seek fast solution methods for such systems by a desire to accelerate the solution of the large non-linear systems that arise when finite difference methods are used to solve Elastohydrodynamic Lubrication (EHL) problems ([16, 21, 26]) using a Newton-based search method. The methods that we present here are more widely applicable and in §6.3 we illustrate how they can be used to accelerate the solution of a wider class of problems by means of an example Boundary Integral Equation.

The Generalized Minimum Residual (GMRES) iterative method ([28]) may be used to solve (1.1), but to obtain sufficiently fast convergence of the method, the matrix A must be preconditioned:

$$AM^{-1}y = b, \qquad x = M^{-1}y, \tag{1.2}$$

with the preconditioner, M, being chosen so that

1. AM^{-1} gives fast convergence of GMRES, *and*
2. $M^{-1}v$ can be easily calculated for any vector v.

The first of these criteria will be satisfied if M is chosen to be a close approximation to A, thus preconditioning often involves a search for a matrix $M \approx A$ with a convenient structure that allows $M^{-1}v$ to be computed at low cost. Preconditioning may be either *implicit* or *explicit* (see [7]). A large number of different preconditioning strategies have been developed. (See [30] for a brief survey of some of the more popular methods.)

*Supported by the Engineering and Physical Sciences Research Council, UK

When comparing alternative preconditioners, an indication of the speed of convergence of GMRES may be found by consideration of the eigen-spectrum of the preconditioned matrix, AM^{-1}. Fast convergence is expected when the eigenvalues are clustered near to 1. (See [30] for a brief description of the relationship between the eigenvalues of A and the convergence of GMRES or [24] for a more detailed account.) However, such an eigen-spectrum is only useful for normal matrices (see [19]). For a general non-symmetric A, the ϵ-pseudo-spectrum is often a better guide to the convergence of GMRES ([20, 24]). Here we concentrate on constructing a sparse M such that $M \approx A$ and assume that the preconditioned eigen-spectrum is not sensitive to small perturbations. The effect of our methods on the pseudo-spectra of preconditioned A is illustrated by an example in §6.

Our algorithms are based on a Discrete Wavelet Transform (DWT). Such methods have been used in the design of preconditioners for smooth, dense matrices ([8, 12, 13]) and for matrices whose inverse is known to be smooth ([10]), but methods are not well-established for matrices with off-diagonal non-smooth features. In this paper we show how local non-smooth areas may be detected and isolated so that a preconditioner may be produced based on using DWT to compress the remaining smooth part. In §4.1 we present algorithms for the isolation of local non-smooth features by permuting the rows and columns of the matrix. In §4.2 and §4.3 we give two alternative algorithms for preconditioner design. In §4.4 we present algorithms for the detection of non-smooth features. We give a theoretical justification for our choice of preconditioner structure in §5 and illustrate the effectiveness of our methods with two examples in §6.

2 Wavelet Transforms

2.1 *Standard wavelet transforms*

Here we are concerned with orthogonal wavelet transforms from the Daubechies family. A DWT is defined by its filter coefficients, $c_0, c_1, \ldots, c_{m-1}$. The number, m, of coefficients is known as the *order* of the wavelet transform. The coefficients $d_0, d_1, \ldots, d_{m-1}$ are derived from the c_i by the following relation:

$$d_i = (-1)^i c_{m-1-i}.$$

Let $v = (s_0^{(0)}, s_1^{(0)}, \ldots, s_{n-1}^{(0)})^T$ be a vector of length n. The standard DWT of v may be computed using the recurrence formulae

$$s_j^{(i+1)} = \sum_{l=0}^{m-1} c_l s^{(i)}_{\langle l+2j\rangle_{n/2^i}}, \quad d_j^{(i+1)} = \sum_{l=0}^{m-1} d_l s^{(i)}_{\langle l+2j\rangle_{n/2^i}}. \tag{2.1}$$

The level k wavelet transform of a $2^n p \times 2^n p$ matrix A is obtained by k applications of (2.1 on both the rows and the columns. This is equivalent to forming the product

$$\tilde{A} = WAW^T = W_k W_{k-1} \ldots W_1 A W_1^T \ldots W_{k-1}^T W_k^T$$

where the W_i are orthogonal matrices defined by the wavelet coefficients (see [25, §3.3.2]).

The initial (level 1) wavelet transform of a column (or row) of a smooth matrix has the effect of concentrating large elements at one end of the column (or row) with smaller entries in the other half. A second (level 2) transform of the half-column containing larger elements produces a further reduction in the number of large elements. This process may be repeated until the number of elements to be transformed is no longer divisible by 2. When we transform a matrix A into the matrix $\tilde{A} = WAW^T$, we are, in effect, expressing the linear transformation defined by A with respect to a new basis given by the columns of W^T. A more detailed description of the DWT may be found in, for example, [14, 15, 29].

An approximation to the transformed matrix may be obtained by 'thresholding', that is setting to zero all elements whose magnitude falls below a given threshold. The smaller the threshold the more accurate the approximation, but at the cost of an increased number of non-zero entries. This technique is frequently used in image compression (see, for example, [14] for a detailed account).

If it is known that the *inverse* of A is smooth then DWT may be used ([10]) to take advantage of this smoothness, which corresponds to small entries in the wavelet transform of A^{-1}. Sparse approximate inverse techniques ([5, 6]) are then used to obtain a sparse matrix M^{-1} to be used for explicit preconditioning. If the matrix A is itself smooth then by applying a DWT and imposing a threshold we obtain a sparse approximation to the transformed matrix that may be used for implicit preconditioning of the transformed equation.

For example, the Calderon-Zygmund type matrix $A = (a_{ij})$, where

$$a_{ij} = \begin{cases} 3/2 & : \quad i = j \\ \frac{1}{|i-j|} & : \quad i \neq j \end{cases} \tag{2.2}$$

is smooth, with entries decreasing in magnitude away from the diagonal. When a DWT is applied to A the resultant $\tilde{A}$ has a weak 'finger' pattern, with most of the largest entries confined to a narrow diagonal band. Figure 2.1 shows mesh plots of A and $\tilde{A}$. A sparse approximation, $\bar{A}$, to $\tilde{A}$ can be formed by setting to zero all elements outside a diagonal band, since this will include all the elements of significant size. The resultant matrix is convenient for implicit preconditioning since little fill-in occurs when a band matrix is decomposed into LU factors.

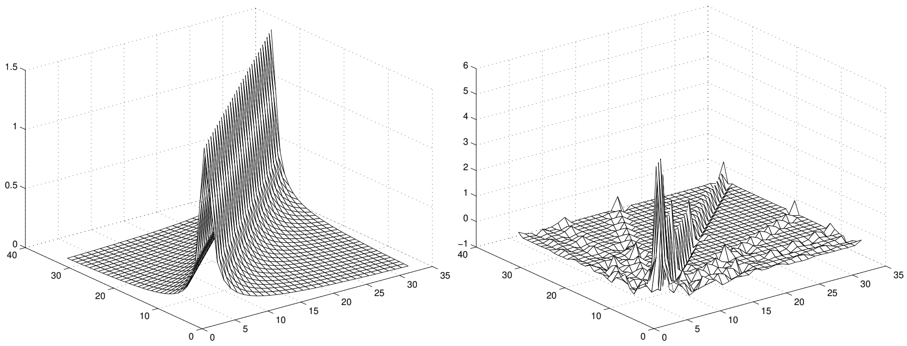

Figure 2.1: Mesh plots of Calderon-Zygmund type matrix and its wavelet transform.

2.2 The wavelet transform with permutations (DWTPer)

A new wavelet transform based on Daubechies wavelet transforms with additional permutations is defined in [13]. This transform has the effect of preserving the general structure (by which we mean the 'shape' of the areas of singularity) of a matrix. To apply the new transform the matrices W_i of the standard transform are replaced by

$$\hat{W}_i = \begin{pmatrix} c_0 & \phi & c_1 & \phi & c_2 & \phi & \cdots & c_{m-1} & \phi & \phi & \phi \\ \phi & I & \phi & \phi & \phi & \phi & \cdots & \phi & \phi & \phi & \phi \\ d_0 & \phi & d_1 & \phi & d_2 & \phi & \cdots & d_{m-1} & \phi & \phi & \phi \\ \phi & \phi & \phi & I & \phi & \phi & \cdots & \phi & \phi & \phi & \phi \\ & & & c_0 & \phi & \ddots & \ddots & & & & \\ & & & \phi & I & \ddots & \ddots & & & & \\ \vdots & \vdots & \vdots & \vdots & \cdots & \cdots & \ddots & \ddots & \ddots & \ddots & \\ c_2 & \phi & \cdots & c_{m-1} & & & & c_0 & \phi & c_1 & \phi \\ \phi & \phi & \phi & \phi & \phi & \phi & \cdots & \phi & I & \phi & \phi \\ d_2 & \phi & \cdots & d_{m-1} & & & & d_0 & \phi & d_1 & \phi \\ \phi & \phi & \cdots & \phi & & & & \phi & \phi & \phi & I \end{pmatrix} \tag{2.3}$$

Here I is an identity matrix of dimension $2^{i-1} - 1$ and the ϕ 's are block zero matrices of the appropriate sizes. For $i = 1$, both I and ϕ are of dimension 0. Figure 2.2 shows the result of performing this new transform on a diagonal matrix. The bandwidth is increased and the new matrix includes wrap-around boundaries. The precise effect of applying DWTPer to a diagonal band matrix is established in [13], where the following theorem is proved.

THEOREM 2.1. *A diagonal band matrix with wrap-around boundaries, lower band width α and upper band width β is transformed by a level k Daubechies DWTPer of order m into a band matrix with wrap-round boundaries, lower block band width $\leq \alpha + m(2^k - 1)$and upper block band width $\leq \beta + m(2^k - 1)$.*

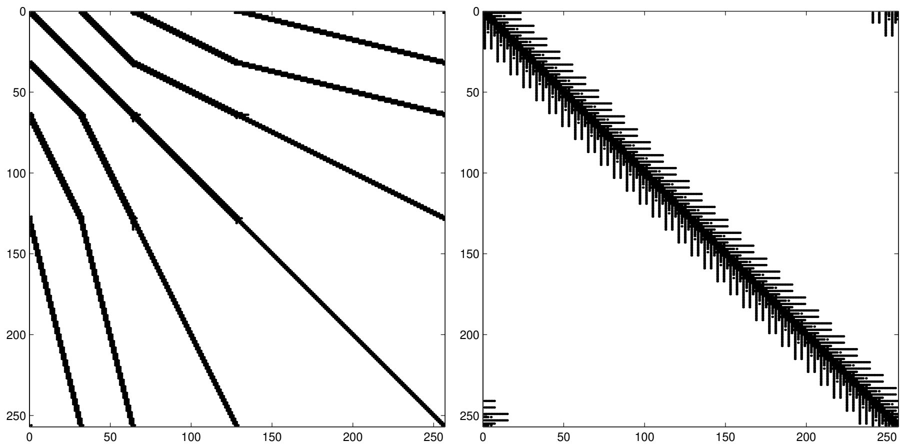

Figure 2.2: Level 3 Daubechies 4 transforms of a diagonal matrix, left : standard DWT, right: DWTPer

The effect of DWTPer on a horizontal or vertical band within a smooth matrix is to produce a corresponding band of large entries in the transformed matrix. Theorem 2.2 enables us to predict the effect of DWTPer on a horizontal non-smooth band situated at the bottom of a smooth matrix. An exactly equivalent result holds for a vertical band in the final (right-hand) columns.

THEOREM 2.2. *A bottom band matrix with bandwidth r (that is a matrix zero everywhere apart from a band of width r at the bottom) is transformed by an order m DWTPer of level k into a bottom band matrix of bandwidth s where $r + (m-2)(2^k - 1) \leq s \leq r + (m-1)(2^k - 1)$.*

PROOF. The proof is straightforward by considering the effect of the DWTPer transform matrices $\hat{W}_i$, $i = 1, \dots, k$ on the band of non-zero entries. We omit the details. ☐

Notice that the bandwidths of the transformed matrix do not depend on the dimension of the matrix, but only on the original bandwidths and the order and level of the wavelet transform. This means that, for a large matrix, the increase in non-zero entries may be a small *proportion* of the whole matrix.

REMARK 2.1. If this band is located in the middle, the increase in bandwidth is greater since non-zero rows are introduced both above and below the original band. This fact is used in §4.2.

3 DWT-based Preconditioners

Equation (1.1) may be transformed by DWT or DWTPer to give

$$WAW^T\tilde{x} = Wb, \quad \tilde{x} = Wx. \tag{3.1}$$

Since the wavelet transform matrix, W, is orthogonal, the eigenvalues of $\tilde{A} = WAW^T$ are the same as those of A. This means that convergence of GMRES will be unaffected by the DWT.

However, if A is smooth it may be possible to find a sparse approximation, M, to $\tilde{A}$ that will be a good preconditioner for (3.1) (see [8, 12, 13]). For example, our experience (using MATLAB) is that when the 256×256 Calderon-Zygmund matrix (2.2) (see Figure 2.1) is transformed with a Daubechies 4 DWT, a band preconditioner may be obtained that has the effect of reducing the number of GMRES iterations required for convergence to a relative residual tolerance of 10^{-6} from 60 to 14 (for an example randomly chosen right-hand-side vector).

In many applications matrices arise that are smooth apart from a diagonal band. Such a matrix A, may be partitioned in the following way:

$$A = S + D \tag{3.2}$$

where S is the smooth part of A and D is a band matrix. DWTPer can be used to make the smooth part, S, sparse while retaining the overall banded structure. A sparse, banded preconditioner can then be chosen to include all the information from the diagonal band D. This process (originally presented in [13]) is set out in Algorithm 3.1 below:

ALGORITHM 3.1. *Banded wavelet preconditioner*

1. *Decide on an operator splitting $A = S + D$ with D a band matrix.*
2. *From D determine a band-width for $\tilde{D}$ from Theorem 2.1.*
3. *Apply DWTPer to $Ax = b$ to obtain $\tilde{A}\tilde{x} = \tilde{b}$.*
4. *Select the preconditioner M as a band matrix of $\tilde{A}$ with the chosen bandwidth.*
5. *Apply a threshold to give a new band matrix, $\bar{M}$, with fewer non-zero entries, and use it to solve $\tilde{A}\tilde{x} = \tilde{b}$ iteratively.*

If the matrix A has off-diagonal non-smooth areas Algorithm 3.1 is of only limited value, since these non-smooth features will produce large entries in the transformed matrix, $\tilde{A}$, that are not included in the preconditioner, M, when $\tilde{A}$ is cut to a band. We now extend this method to the design of wavelet-based preconditioners for a wider class of matrices.

4 Preconditioners for matrices with local off-diagonal non-smooth features

The general idea behind Algorithm 3.1 is to use knowledge of the positions of any non-smooth parts of A to predict where the most significant entries of $\tilde{A}$ will lie and then to construct a sparse M from a portion of $\tilde{A}$ chosen to include all these entries. In the case where the non-smoothness is restricted to a diagonal band we can do this using a banded M, but if A has off-diagonal areas of non-smoothness it will no longer be possible to find a suitable bandwidth to include all of the required entries of $\tilde{A}$.

We first consider a special case: a bordered block matrix. If the off-diagonal features of A take the form of bands at the bottom and/or right-hand edge of the matrix then we can use Theorem 2.2 to predict the increase in bandwidth when DWTPer is applied. Figure 4.1 illustrates the effect of wavelet transforms on a matrix consisting of a single diagonal band, bottom row and final column. Notice that although the bands have increased in width in line with Theorem 2.2, the resulting bands are not dense; they include a large number of zero entries. Such matrices will be used in our preconditioning approach here.

4.1 Isolating local non-smooth features

In realistic problems, the non-smooth features are not generally so conveniently situated. Consider the matrix, A, shown in Figure 4.2 with its level 3 standard DWT and DWTPer transforms. The off-diagonal non-smooth feature is a 8×2 rectangle. This small non-smooth area results in a large number of non-zero entries in the transformed matrices. If we were to follow the philosophy of Algorithm 3.1 in constructing M from the DWTPer transform, $\tilde{A}$, we would have to include the large rectangular area shown in the diagram bottom-left in Figure 4.2. The fourth diagram in

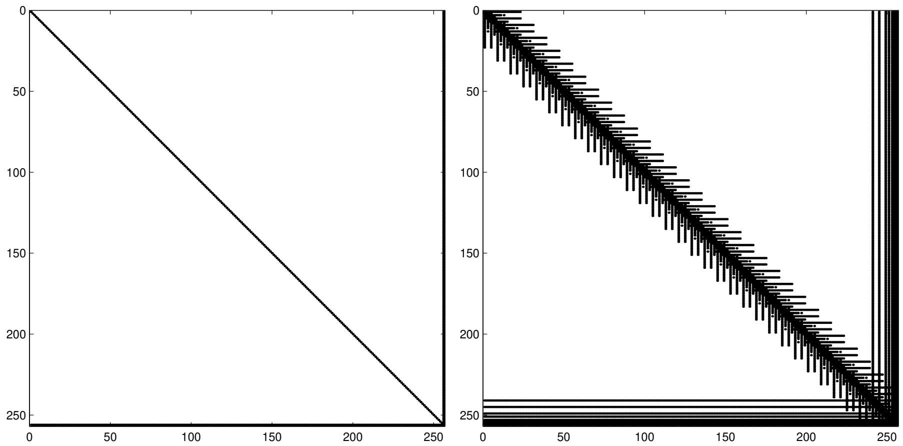

Figure 4.1: The effect of a level 3, Daubechies 4 DWTPer on a bordered block matrix.

Figure 4.2 shows the DWTPer transform of a matrix with a non-smooth 8×2 rectangle positioned at the right-hand edge of the matrix. Notice that the transform contains fewer non-zero entries and that they are contained within a band at the right-hand edge of the matrix. We can move the rectangle to the right-hand edge of the matrix by permuting the columns. Non-smooth features located at the top or left-hand edge of the matrix present similar problems, so they also need to be moved.

If the matrix has both diagonal and off-diagonal non-smooth features (see Figure 4.3) the diagonal band will be disrupted by permuting the columns. This can be remedied by permuting the rows in an identical way. Applying DWTPer to this permuted matrix gives a matrix with sparse diagonal, horizontal and vertical bands (Figure 4.3, bottom right).

More generally, suppose we have a matrix

$$A = \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix},$$

where each A_{ij} is an $m_i \times n_j$ matrix and A_{22} encloses an off-diagonal area of non-smoothness. To permute this to bordered block form we first find $\min(m_2, n_2)$. Suppose, without loss of generality, that $m_2 > n_2$. Then we permute the columns of A to bring A_{22} to the right-hand edge of A and permute the rows in an identical manner.

$$\hat{A} = PAP^T, \tag{4.1}$$

where $P = (\mathbf{e}_1, \ldots, \mathbf{e}_{n_1}, \mathbf{e}_{n_1+n_2+1}, \ldots, \mathbf{e}_n, \ldots, \mathbf{e}_{n_1+1}, \ldots, \mathbf{e}_{n_1+n_2})^T$.

Section 4.4 will discuss ways to detect such features. We now concentrate on preconditioner designs based on a knowledge of these features.

4.2 DWTPer bordered block matrix preconditioning: Approach 1.

Provided that the local non-smooth features can each be enclosed in a rectangle whose smaller dimension is not too large, we can permute the matrix to form a bordered block matrix whose sparsity pattern under DWTPer can be predicted using Theorems 2.1 and 2.2. If more than one non-smooth area is present, each is moved in turn to the bottom or right-hand edge. Permutation to move the non-smooth features to the edges has two advantages:

1. By isolating the non-smooth areas at the bottom and right-hand edges of the matrix, the number of large entries in $\tilde{A}$ is reduced (see remark 2.1).

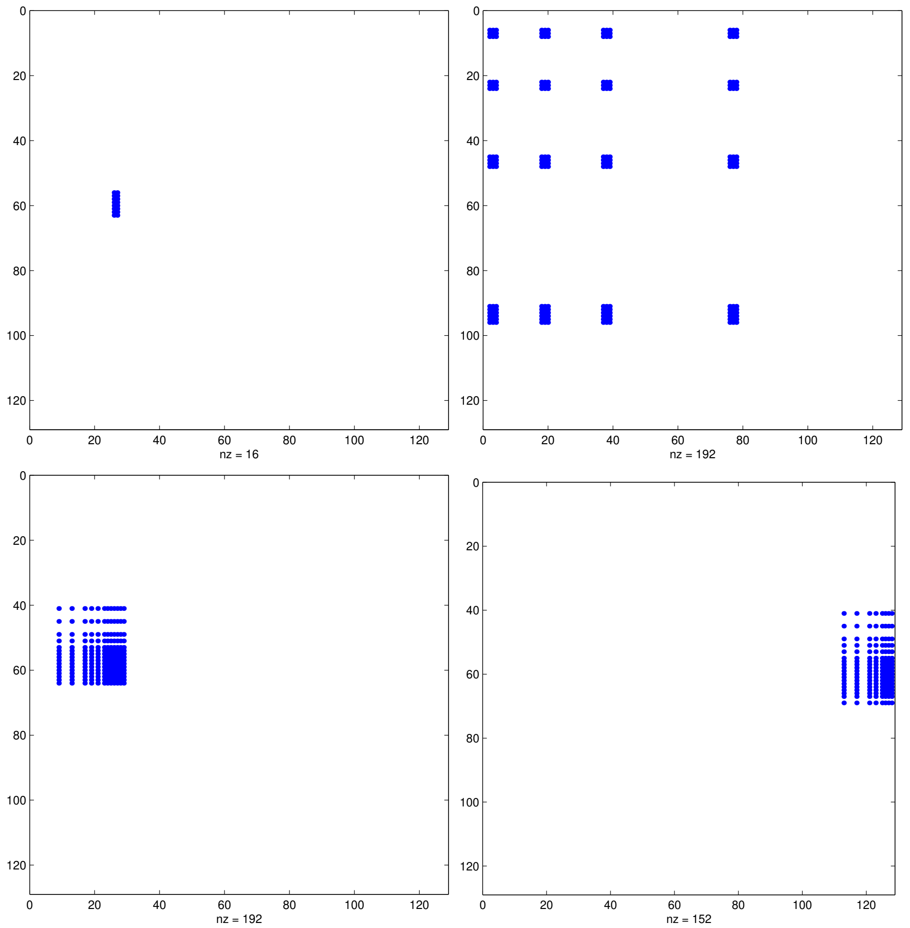

Figure 4.2: A matrix with a localized area of non-smoothness and its level 3 Daubechies 4 transforms. Top left : original matrix, top right: standard DWT, bottom left: DWTPer, bottom right: DWTPer after moving non-smooth feature to the right-hand edge

2. The large entries in $\tilde{A}$ are confined to lie within a bordered block structure, which allows us to use a bordered block matrix as our preconditioner. This ensures that the preconditioning step can be implemented at low cost (see §5).

This preconditioning strategy, which we shall refer to as Approach 1, is summarized in our new Algorithm 4.1.

ALGORITHM 4.1. *DWTPer bordered block matrix wavelet preconditioner with thresholding*

1. *Identify any non-smooth areas of* A.
2. *Apply a permutation matrix,* P*, to* $Ax = b$ *to move any off-diagonal features to the bottom or right-hand edge, producing a bordered block matrix,* $\hat{A} = PAP^T$*, giving the new equation* $\hat{A}\hat{x} = \hat{b}$*, where* $\hat{x} = Px$*,* $\hat{b} = Pb$.
3. *Determine a suitable bordered block structure from Theorems 2.1 and 2.2.*
4. *Apply DWTPer to* $\hat{A}\hat{x} = \hat{b}$ *to obtain* $\tilde{A}\tilde{x} = \tilde{b}$.
5. *Select the preconditioner* M *as a bordered block matrix of the chosen structure derived from* $\tilde{A}$.

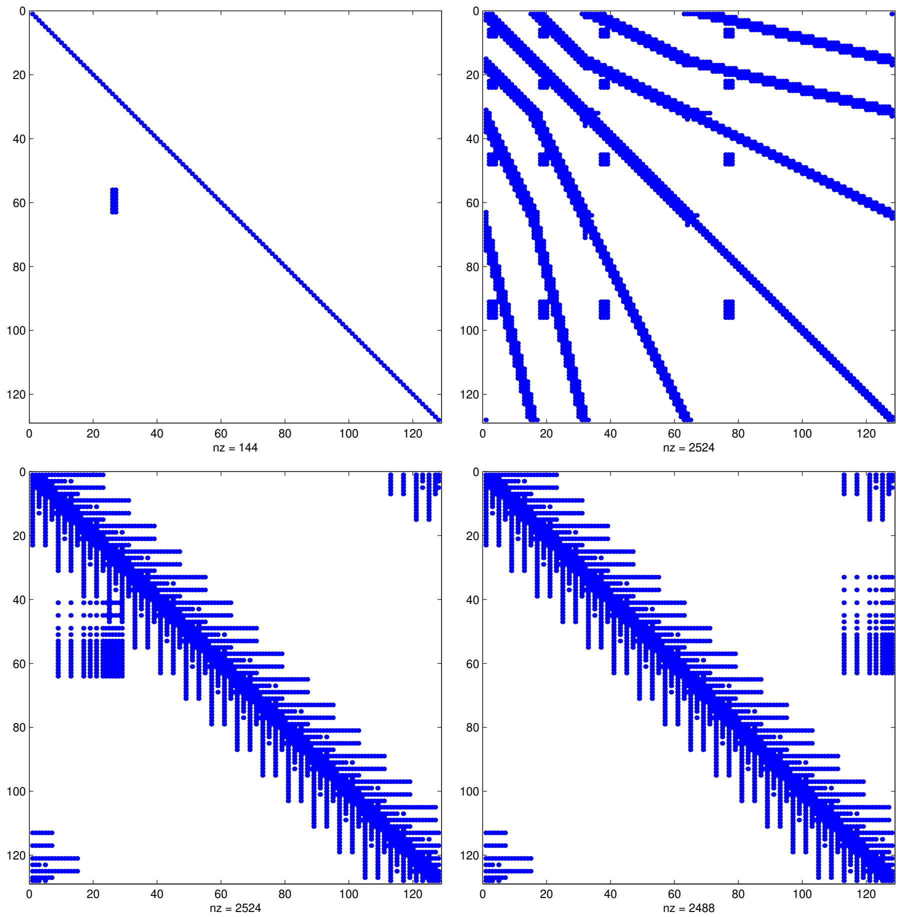

Figure 4.3: A matrix with a diagonal band and a localized area of non-smoothness. Top left : original matrix, top right: standard DWT, bottom left: DWTPer, bottom right: DWTPer after moving the local feature to the right-hand edge

6. *Set to zero all elements of M which fall below a certain threshold to give a new bordered block matrix $\bar{M}$ and use it to solve $\tilde{A}\tilde{x} = \tilde{b}$ iteratively.*

4.3 DWTPer bordered block matrix preconditioning: Approach 2.

Approach 1 reduces the number of large entries introduced as a result of the non-smooth features and allows them to be confined to the bottom and right-hand borders of the transformed matrix. It is possible to reduce the adverse effect of non-smooth features still further by using a block DWTPer. The wavelet transform matrix, W, is replaced by $\hat{W} = \begin{pmatrix} W & \Phi \\ \Phi & I \end{pmatrix}$, where the dimension of W is chosen to match as closely as possible that of the bordered block matrix *excluding* the bottom and right-hand borders. An exact match will not in general be possible since, for a level k DWTPer, the dimension of W must be a multiple of 2^k. This alternative method of preconditioning, which we shall call Approach 2, is summarized in our second new Algorithm 4.2.

ALGORITHM 4.2. *Block DWTPer bordered block wavelet preconditioner with thresholding*

1. *Identify any non-smooth areas of A.*

2. *Apply a permutation matrix, P, to $Ax = b$ to move any off-diagonal features to the bottom*

or right-hand edge, producing a bordered block matrix, $\hat{A} = PAP^T$*, giving the new equation* $\hat{A}\hat{x} = \hat{b}$*, where* $\hat{x} = Px$*,* $\hat{b} = Pb$*.*

3. *Decide on a suitable level,* k*, for DWTPer.*
4. *Let the width of the borders be* r *and the dimension of the whole matrix be* n*. Calculate* m *such that* $m = 2^k p \leq n - r$*.*
5. *Determine the width of the diagonal band from Theorem 2.1.*
6. *Set the width of the borders to* $n - m$*.*
7. *Apply a block DWTPer, block dimension* m*, to* $\hat{A}\hat{x} = \hat{b}$ *to obtain* $\tilde{A}\tilde{x} = \tilde{b}$*.*
8. *Select the preconditioner* M *as a bordered block matrix of the chosen structure derived from* $\tilde{A}$*.*
9. *Set to zero all elements of* M *which fall below a certain threshold to give a new bordered block matrix* $\bar{M}$ *and use it to solve* $\tilde{A}\tilde{x} = \tilde{b}$ *iteratively.*

An advantage of Approach 2 over Approach 1 is that the width of the preconditioner borders is less, thus making the preconditioner more sparse and hence cheaper to apply. However, using Approach 2 results in some of the smooth part of the matrix being excluded from the transform, which means that compression will be less good. We would expect that the wider border preconditioner produced using Approach 1 will give faster convergence of GMRES, but the additional cost of applying the preconditioner at each iteration may, in some cases, outweigh this advantage. We have not yet been able to find a general rule as to which of the alternative methods will perform best in any given case.

4.4 Detection of non-smooth features

Algorithms 4.1 and 4.2 require that the positions of any non-smooth features are known. In some cases the underlying problem may dictate the positions of such features sufficiently accurately, but more generally it will be necessary to identify such areas by inspection of the $n \times n$ matrix A. To do this we need to establish ways of measuring the smoothness of a matrix in a way that will give an indication of the effect on the sizes of the corresponding entries in $\tilde{A}$. The sizes of the wavelet coefficients of a function depend on its derivatives (see [29] Theorem 7.6), so a simple measure of smoothness for a matrix would be to consider finite differences along the rows and columns. For each element, $a_{i,j}$, of A $(i, j, \neq n)$ we define the *difference vector*, $d^{i,j}$ whose two components are the differences $a_{i+1,j} - a_{i,j}$ and $a_{i,j+1} - a_{i,j}$. The norm of this vector gives a measure of the non-smoothness of A at position (i, j). We chose to use the infinity norm, $\|d^{i,j}\|_\infty = \max(|d_1^{i,j}|, |d_2^{i,j}|)$, defining the gradient matrix $G = (g_{i,j})\,, \quad g_{i,j} = \|d^{i,j}\|_\infty$. This norm is cheap to implement and ensures that non-smoothness along either the row or the column containing a_{ij} will be detected. $a_{i,j}$ is considered to be part of a non-smooth feature of A if $g_{i,j} > d_{thresh}$, where d_{thresh} is a chosen threshold value. (See §6.3 for a description of how this value was chosen in the case of the Helmholtz equation.) The rows (or columns) of A that should be permuted in order to isolate the non-smooth features are then given by those rows (or columns) of G that contain a 'large' number of entries greater than d_{thresh}. This detection method is summarized in Algorithm 4.3 below

ALGORITHM 4.3. *Detection of local non-smooth features in an* $n \times n$ *matrix,* A*.*

1. *Choose a threshold value,* d_{thresh}*, and a minimum feature-size,* m*.*
2. *Calculate the gradient matrix,* G*.*
3. **For** $i = 1 \ldots n - 1$
   If more than m *entries in row* i *of* G *are* $> d_{thresh}$
   mark row i *of* A *as part of a local non-smooth feature.*
4. **End For**

5. **For** $j = 1 \dots n-1$
   If more than m *entries in column* j *of* G *are* $> d_{thresh}$
   mark column j *of* A *as part of a local non-smooth feature.*

6. **End For**

In practice it is often possible to predict that the non-smooth features will tend to lie either along the rows or columns of the matrix, so that it may be necessary only to check for non-smoothness in one direction. For example, in the case of the Helmholtz equation (see §6.3) the non-smooth features lie along the rows of the matrix.

In the case of the EHL Jacobian matrices (see §6.2) an even simpler detection method was found to be satisfactory. The non-smooth regions in this case are characterized by entries of A much larger in magnitude than the surrounding entries. This means that it is not necessary to calculate finite differences in order to locate the singularities. Instead non-smoothness can be detected by looking for entries, $a_{i,j}$, for which $|a_{i,j}| > \bar{a}$, where $\bar{a}$ is the mean absolute value of the $a_{i,j}$ excluding the dominant diagonal band.

5 Cost analysis

Although it is not uncommon to consider the reduction in iteration counts as a result of using a preconditioner as a test of its efficacy, this is not necessarily a good measure of preconditioner performance. What is really important is the overall cost of the iterations including the preconditioning steps. When a matrix, M, is used as a preconditioner for GMRES the calculation $v \longleftarrow M^{-1}v$ must be done at each iteration. In order to do this, the LU decomposition of M is calculated at the start of the GMRES process and then, at each iteration, forward and backward substitution are used to calculate $M^{-1}v$. Thus, the additional expense of using M may be thought of as a one-off cost of factorization plus the cost of substitution at each iteration. Both of these costs are affected by the structure of the matrix. There will also be a cost associated with the initial calculation of M itself. In the cases that we are considering here, this will comprise:

1. permutation of the rows and columns to move off-diagonal features;
2. application of block DWTPer;
3. thresholding and cutting to bordered block form.

Of these we expect only the wavelet transform to involve significant expense, and this cost will depend only on the block size, which will typically be $\approx n$.

If we are using an effective preconditioner then we expect these additional costs to be far outweighed by the reduction in the number of GMRES iterations required for convergence to the required tolerance. To make a fair comparison of alternative preconditioners we have chosen to consider the overall cost of using each preconditioner to perform the number of iterations needed for convergence.

The cost in terms of flops of performing a standard DWT may be calculated in a straightforward manner ([25]). Since DWTPer simply involves a permutation of standard DWT, the flop count for DWTPer is identical. The cost of applying a block size N, order m, level l DWTPer to an $n \times n$ rectangular matrix is

$$8mnN\left(1 - \frac{1}{2^l}\right). \tag{5.1}$$

The total cost of solving the linear system by GMRES iteration can be shown to be

$$8mnN\left(1 - \frac{1}{2^l}\right) + 2n(p+r)^2 + 2n^2k + \mathcal{O}(n), \tag{5.2}$$

where p, r are the widths of the diagonal and horizontal bands in P respectively and k is the number of iterations required for convergence to the required tolerance. In order to assess the effectiveness of the iterative method we will compare this cost with the flop count for direct solution of the system by Gaussian elimination:

$$\frac{2n^3}{3} + 2n^2. \tag{5.3}$$

Of course, an estimation of the flop count alone may not be a good indicator of the performance of an algorithm since this does not take into account other costs, such as subscripting and memory access [18, p.19]. In particular, the cost of solution of dense linear systems by Gaussian elimination may depend to large extent on the efficient use of cache or an effective blocking strategy. However, we believe that these factors would be likely to play a similar rôle in determining the cost of solving the transformed systems. In the following section we include results that demonstrate substantial improvements in CPU time as well as in flop count when solving an example problem.

6 Numerical Examples

We illustrate the effectiveness of our methods by three examples: a Calderon-Zygmund matrix, an elastohydrodynamic lubrication problem and a boundary integral equation. We compare the performance of GMRES using the following preconditioning methods:

1. Unpreconditioned (NONE).
2. Tridiagonal band preconditioner (tridiagonal).
3. Band preconditioner based on standard DWT (Standard DWT).
4. Band preconditioner based on DWTPer using Algorithm 3.1 (DWTPer band).
5. Approach 1: bordered block preconditioner based on DWTPer of permuted matrix (A1).
6. Approach 2: bordered block preconditioner based on block DWTPer of permuted matrix (A2).

The expressions in brackets indicate the label used for each method in the tables of results. The wavelet transform used was Daubechies order 4. For each of the two example types of matrix a suitable wavelet transform level for each size of matrix was established experimentally for Algorithm 3.1 and this same level used for the other tests. We also considered the use of a standard DWT-based preconditioner formed by setting to zero all entries whose magnitude is less than a chosen threshold value (as done in [22]), and found that such preconditioners were unsatisfactory for the following reasons:

- It was difficult to choose a threshold that made the preconditioner sparse without the risk that it would become singular.
- Although the preconditioner itself was sparse, the positioning of the non-zero entries resulted in a large amount of fill-in during LU decomposition. As a result, the preconditioner was expensive to apply.

The pseudo-spectra of the preconditioned matrices were also examined (see §6.2 and Figure 6.2) and these confirmed that our preconditioners would be expected to give fast convergence of GMRES.

6.1 Calderon-Zygmund matrix

We tested our methods on the Calderon-Zygmund matrix given by (2.2), which is known to respond well to DWT-based preconditioning. Although the Calderon-Zygmund matrix does not have non-smooth local features, Algorithm 4.3 detects non-smoothness in the bottom row, because the entries here are not similar in value to the corresponding entries at the top of the matrix. No rows are moved (since the non-smoothness is already at the bottom of the matrix), but the bandwidths for the bordered block preconditioners A1 and A2 are adjusted to take into account the effect of this 'wrap-round' non-smoothness on the DWTPer transformed matrix. We found that DWT-based preconditioners substantially improved the speed of convergence of GMRES, with DWTPer-based preconditioners performing slightly better than those based on standard DWT. Table 6.1 shows the number of iterations required to obtain a relative residual norm of below 10^{-6} using different preconditioners.

Table 6.1: Iteration count when using different preconditioners for the Calderon-Zygmund matrix given by (2.2).$*$ denotes stagnation of GMRES(25).

n Preconditioner	64	128	256	512	1024
NONE	40	51	60	66	70
tridiagonal	51	103	243	494	$*$
Standard DWT	14	14	14	14	14
DWTPer band	11	12	12	13	13
A1	10	10	11	11	12
A2	11	12	12	13	13

To find out whether our methods are useful in improving overall performance as well as in reducing the number of GMRES iterations required for convergence, we also compared the number of flops used to solve the system using each of the preconditioners. We also compared with the flop count using Gaussian elimination. Table 6.2 shows the number of flops used to reduce the relative residual to below 10^{-6}. From this we can see that, for matrices of dimension 256 or more, useful savings can be made by using GMRES preconditioned with a DWT-based preconditioner rather than Gaussian elimination. Of the alternative preconditioners considered, A1 is slightly superior but for this example similar performance is exhibited by all the wavelet-based preconditioners. In the following two examples we will see how the presence of local areas of non-smoothness makes the choice of preconditioning method more important.

Table 6.2: Flop count when using different preconditioners for the Calderon-Zygmund matrix given by (2.2). $*$ denotes stagnation of GMRES(25)

n Preconditioner	64	128	256	512	1024
NONE	9.0×10^5	4.1×10^6	1.8×10^7	7.4×10^7	3.1×10^8
Tridiagonal	1.2×10^6	8.3×10^6	6.4×10^7	5.6×10^8	$*$
Standard DWT	3.9×10^5	1.4×10^6	5.3×10^6	2.1×10^7	8.1×10^7
DWTPer band	3.2×10^5	1.2×10^6	4.7×10^6	1.9×10^7	7.7×10^7
A1	3.1×10^5	1.1×10^6	4.6×10^6	1.8×10^7	7.5×10^7
A2	3.3×10^5	1.3×10^6	4.8×10^6	2.0×10^7	7.9×10^7
G. E.	2.1×10^5	1.5×10^6	1.2×10^7	9.2×10^7	7.2×10^8

6.2 Isothermal EHL line contact problem

The characteristic problem in EHL line contact modelling is the entrainment of oil between two elastically deforming, rotating cylinders under an applied load w per unit axial length of the cylinders. The theory of isothermal EHL line contacts is covered in several standard texts, for example [16]. For a steady state contact, the pressure P is determined by the *Reynolds equation*. In non-dimensional form, this is

$$\frac{d}{dX}\left(\frac{\bar{\rho}H^3}{\bar{\eta}}\frac{\partial P}{\partial X}\right)-\lambda\frac{d}{dX}\left(\bar{\rho}H\right)=0, \tag{6.1}$$

where

$$H(X)=H_0+\frac{X^2}{2}-\frac{1}{\pi}\int_{-\infty}^{\infty}\ln\left(|X-X'|\right)P(X')dX'. \tag{6.2}$$

H_0 is determined indirectly by the load balance equation, given in non-dimensional form by

$$\int_{-\infty}^{\infty}P(X)dX=\frac{\pi}{2}. \tag{6.3}$$

These equations are discretised using second order finite differences giving a system of non-linear equations that are solved by a Newton method. Although there are now a number of alternative numerical methods for the solution of equations (6.1-6.3), the combination of a finite difference discretisation and Newton-Raphson iteration is still popular in the context of EHL problems (see, for example, [21, 26]) due in part to its simplicity and the availability of software. There has been a growing interest in the use of multi-grid methods as an alternative to the Newton-Raphson iteration particularly for point-contact problems. However, for line-contacts there is evidence that the Newton-Raphson method is to be preferred ([23]).

At each step of the Newton iteration a linear system (corresponding to the Jacobian matrix of the nonlinear system at the current iteration point) must be solved. This can be done by Gaussian elimination, but, because the Jacobian is full this is expensive, so a GMRES iterative method is used. Unpreconditioned GMRES does not converge quickly enough to be useful, so it is necessary to find an effective preconditioner. Previous work (see [17]) has shown that DWTPer-based preconditioners are effective in reducing the cost. For example, the total cost of solving a dimension 1025 non-linear system is typically reduced by a factor of 10 using preconditioned GMRES compared to using Gaussian elimination. In this paper we show that, by using bordered block matrix preconditioners, we can further reduce the time required to solve each linear system.

The Jacobian matrices are dense with a non-smooth diagonal band arising from the derivative discretisation. The logarithmic kernel in (6.2) implies that the Jacobian will be smooth away from the diagonal band, apart from a single row and column corresponding to (6.3). This smoothness motivates us to consider wavelet compression as a possible preconditioning strategy. Horizontal bands of large elements also appear in the Jacobian matrices. These bands are most significant in the early stages of the Newton iteration, becoming weaker as the solution is approached. If the horizontal band features are moved to the bottom the Jacobian matrix and its DWTPer transform become more suited to bordered block preconditioning. An example of the type of matrix and it DWTPer transform before and after moving the horizontal bands are shown in Figure 6.1. In each case entries of magnitude $\geq 10^{-6}$ are shown.

Algorithms 4.1 and 4.2 have been implemented as part of a program, written in C, to solve EHL problems. The Jacobian matrices are formed numerically column-by-column using finite differences. By changing the parameters of the underlying problem we were able to test the algorithms for different matrices all having the same basic structure: relatively small, smoothly varying elements outside of the band structures; a narrow, dominant, diagonal band; two horizontal bands, one being close to the bottom of the matrix; a single row and column at the bottom and right-hand edges containing larger elements. The last row and column were excluded from the wavelet transform, since they were known to be non-smooth. Because the lower band lies close to the bottom of the matrix we thought that it might be unnecessary to move this band, so two versions of each of Approaches 1 and 2 were tested: in the first version only the upper band is moved, while in the second version both bands are moved to the bottom.

When Approach 1 is used, the number of iterations required to obtain a relative residual norm of below 10^{-6} is reduced when the upper horizontal band is moved. Moving the lower band as well often gave further improvements, but this is not always the case. Indeed, for matrices where

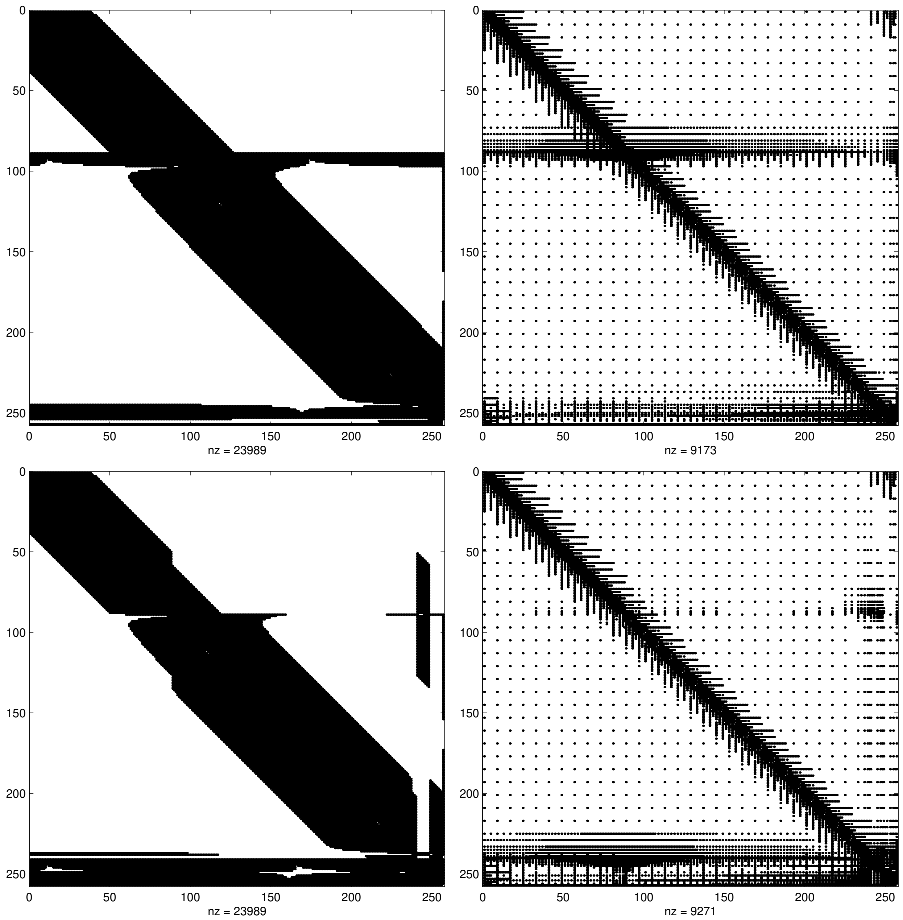

Figure 6.1: Example EHL matrix. Top left : original matrix; top right: DWTPer; bottom left: matrix with non-smooth bands permuted to the bottom; bottom right: DWTPer of permuted matrix.

moving the upper band gave particularly good iteration reductions, moving both bands often gave a slight *increase* in iteration count compared with moving only the upper band. However, in such cases, there is still a substantial reduction when compared with no band moving.

When Approach 2 is used, however, the number of iterations is often reduced only marginally (if at all) by moving the bands. This confirms our prediction in §4 that Approach 1 would give faster convergence than Approach 2, and makes it likely that, for these particular matrices, Approach 1 is the better of the two. A comparison of CPU times for each approach confirms that Approach 1 is to be preferred.

Table 6.3 shows the results for one example choice of parameters to the EHL problem. The different matrix dimensions represent different mesh sizes in the discretisation of the problem. Typical dimensions used in the practical solution of such problems are 513 (for moderate accuracy) or 1025 (if greater accuracy is required). This choice is partly influenced by the length of time taken to solve the linear systems using Gaussian elimination. With a GMRES iterative approach using DWTPer-based preconditioners it is practicable to use a larger number of mesh points. In the case of the EHL Jacobian matrices, a tridiagonal band preconditioner was found to be not possible, since such matrices were singular; so a modified tridiagonal matrix was used whereby the bottom row and final (right hand) column were also included in the preconditioner. Notice that the iteration count does not increase much as the dimension of the matrix increases.

Table 6.3: Iteration count (for convergence to a tolerance of 10^{-6})when using different preconditioners for the EHL Jacobian matrices. $*$ denotes stagnation of GMRES(25).

n Preconditioner	129	257	513	1025	2049
NONE	$*$	$*$	$*$	$*$	$*$
tridiagonal	$*$	$*$	$*$	$*$	$*$
Standard DWT	$*$	$*$	$*$	$*$	$*$
DWTPer band	9	19	19	14	21
A1 upper band	7	16	5	10	6
A1 both bands	5	9	7	4	4
A2 upper band	9	17	16	13	12
A2 both bands	8	10	10	9	11

Table 6.4: Bandwidths and number of GMRES iterations for convergence to a tolerance of 10^{-6} using example 1025×1025 preconditioners.

	$p =$ diagonal half band width	$r =$ bottom band width	$k =$ number of iterations
DWTPer band	13	1	14
A1 upper band	13	17	10
A1 both bands	13	21	4
A2 upper band	13	5	13
A2 both bands	13	9	9

To establish whether or not the iteration reductions associated with moving the horizontal bands are likely to produce useful reductions in CPU time we need to consider the increased work involved in carrying out the GMRES process using the different preconditioners. Let us look in detail at the 1025×1025 case. Table 6.4 shows the values of p, r and k for each preconditioner. In practice, the lower bandwidth, q of the factorized preconditioner is observed to be $\approx p$. Using these values, we can estimate the number of flops required for the GMRES iteration using each preconditioner, giving the values shown in Table 6.5. For the purposes of comparison the cost of solution by Gaussian elimination (G.E.) is also shown. The final column gives the total cost of preconditioned GMRES as a percentage of the cost using Gaussian elimination. The additional work involved in using either version of Approach 1 is less than the savings as a result of the reduction in the number of iterations required. Hence, assuming that the amount of time taken to move the bands is small, we expect to obtain savings in the CPU time when the bands are moved. The corresponding CPU times (shown in Table 6.6) confirm this. These are average timings over 10 runs of the program on a Silicon Graphics IRIX64. With Approach 2 moving only the upper band there is only a small saving in flop count, so it is not surprising that the CPU times indicate that this method is less efficient than using a band DWTPer.

Remark 6.1. In practice greater savings compared with Gaussian elimination can be achieved by using preconditioned GMRES, because it has been found that a tolerance of 10^{-3} (rather than 10^{-6} as in Table 6.6) is sufficient to ensure fast convergence of the Newton iteration.

To give an indication why our proposed preconditioners would perform better, let us consider the preconditioned eigenvalues and pseudo-spectra. As is known [20, 30], for non-symmetric

Table 6.5: Estimated flop counts for convergence to a tolerance of 10^{-6} using example 1025×1025 preconditioners.

	DWTPer	Factor	Iteration	Total
DWTPer band	2.6×10^7	4.0×10^5	3.1×10^7	5.6×10^7
A1 upper band	2.6×10^7	1.8×10^6	2.3×10^7	5.0×10^7
A1 both bands	2.6×10^7	2.4×10^6	9.1×10^6	3.7×10^7
A2 upper band	2.5×10^7	6.6×10^5	2.9×10^7	5.4×10^7
A2 both bands	2.5×10^7	9.9×10^5	2.0×10^7	4.6×10^7
Gaussian elimination	-	-	-	7.2×10^8

Table 6.6: CPU time (sec) for components of GMRES using bordered block matrix preconditioners for a 1025×1025 matrix. (GMRES is stopped when the relative residual falls below 10^{-6}.)

	Preparation of preconditioner	Factorization	Iteration	Total
DWTPer band	3.63	0.02	1.78	5.43
A1 upper band	3.71	0.11	1.31	5.13
A1 both bands	3.71	0.15	0.54	4.39
A2 upper band	3.94	0.03	1.66	5.63
A2 both bands	3.80	0.05	1.16	5.01
Gaussian Elimination	-	-	-	40.49

matrices, the pseudo-spectra provide a guide to the likely convergence behavior of GMRES while the eigenvalues (with clustering) only become useful indicators if they are not sensitive to small perturbations. For an example of highly sensitive eigenvalues, refer to [19].

Figure 6.2 shows the pseudo-spectra of an example EHL Jacobian preconditioned using 3 methods: (a) tridiagonal, (b) DWTPer band, and (c) A1 DWTPer bordered block after permuting to move both horizontal non-smooth bands. The dots represent the eigenvalues, and the boundaries of the ϵ-pseudo-spectra for $\epsilon = 10^{-8}, \ldots, 10^{-1}$ are shown as solid lines. Clearly one can observe that case (c), our proposed preconditioner, has the best pseudo-spectra and thus the least sensitive eigenvalues (clustered at 1). In fact, the pseudo-spectrum for $\epsilon = 10^{-4}$ is considerably reduced in size when method (b) is used rather than method (a), and it becomes indistinguishable in Figure 6.2 from that for $\epsilon = 0$ (i.e. the eigenvalues) when method (c) is employed. This shows that the eigenvalues of the preconditioned matrix are least sensitive to small perturbations when method (c) is used and explains why our proposed methods are to be preferred.

We also note in passing that measuring numerically the size of a bounded set in the complex plane that contains the pseudo-spectra for a specific ϵ (say 10^{-4}), using the convergence results of [20], can lead to an estimate of the convergence rate i.e. a upper bound for $\|r_n\|/\|b\|$. For case (c), this set can be taken as a small ellipse and for cases (a) and (b) larger ellipses indicating that the estimated rate is the smallest for case (c).

6.3 Exterior Helmholtz equation

The application of wavelet compression techniques to BIEs is a comparatively recent innovation ([4]). There are several distinct approaches. Wavelet transforms may be used (either in standard or non-standard form) to make possible a sparse approximation to the transformed matrix that is sufficiently close to allow it to be used *in the place of* the original matrix either in a direct or an iterative solver (as in [9, 8]). Alternatively, a sparse approximation to the wavelet-transformed matrix may be used as a preconditioner for an iterative method (as in [12]).

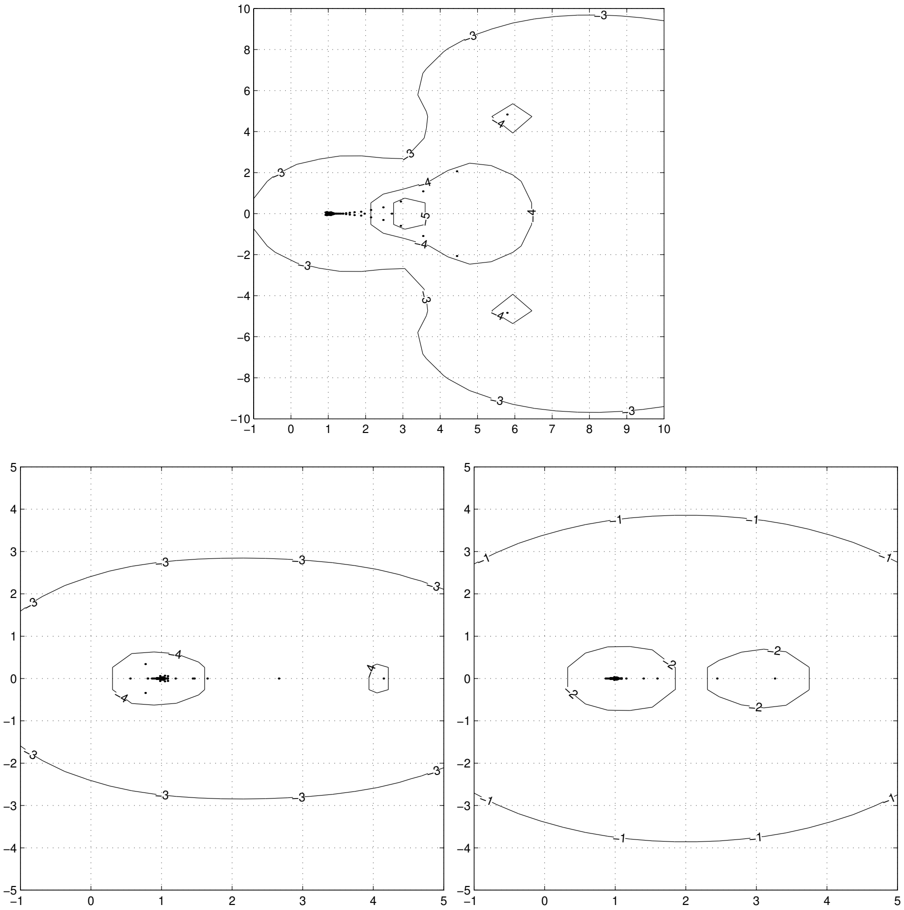

Figure 6.2: Pseudo-spectra of 257×257 preconditioned EHL Jacobian matrices: top - tridiagonal; bottom left - DWTPer band; bottom right - DWTPer bordered block

The exterior Helmholtz equation (see [1])

$$(\nabla^2 + \mathtt{k}^2)\phi(p) = 0 \tag{6.4}$$

is of importance in acoustic scattering problems. For Neumann's boundary conditions, a unique BIE formulation due to Burton and Miller is the following

$$(-\frac{1}{2}\mathcal{I} + \mathcal{M}_{\mathtt{k}} + i\eta\mathcal{N}_{\mathtt{k}})\phi = [\mathcal{L}_{\mathtt{k}} + i\eta(\frac{1}{2}\mathcal{I} + \mathcal{M}_{\mathtt{k}}^T)]\frac{\partial\phi}{\partial n}.$$

Here $\mathcal{L}_{\mathtt{k}}$ and $\mathcal{M}_{\mathtt{k}}$ are the usual single and double layer potential operators respectively,

$$(\mathcal{L}_{\mathtt{k}}\phi)(p) = \int_\Gamma G_{\mathtt{k}}(p,q)\phi(q)dS, \qquad (\mathcal{M}_{\mathtt{k}}\phi)(p) = \int_\Gamma \frac{\partial G_{\mathtt{k}}}{\partial n_q}\phi(q)dS.$$

$\mathcal{M}_{\mathtt{k}}^T$ is the adjoint of $\mathcal{M}_{\mathtt{k}}$ and $\mathcal{N}_{\mathtt{k}}$ is the hypersingular operator. The 2D Green function is defined to be $G_{\mathtt{k}}(p,q) = \frac{i}{4}H_0^{(1)}(\mathtt{k}|p-q|)$ with $H_0^{(1)}$ the Hankel function. (See [1, 2, 3].)

For the case where the interior boundary Γ is a convex curve such as an ellipse the matrices that arise when a collocation method is used to solve (6.3) are smooth apart from an area of non-smoothness along the diagonal corresponding to points on the curve that are very close to one another. In this case banded preconditioners based on DWTPer have been demonstrated to be effective (see [13]). However, when the boundary is such that points on the curve may be close to one another *spatially*, while being distant in terms of *distance along the boundary curve* this gives rise to additional off-diagonal non-smooth features. For example, the 'peanut' curve $x = \sqrt{\left(\cos 2t + \sqrt{a - \sin^2 2t}\right)} \cos t,\ y = \sqrt{\left(\cos 2t + \sqrt{a - \sin^2 2t}\right)} \sin t$ (shown in Figure 6.3) gives a matrix with the areas of non-smoothness shown in Figure 6.4.

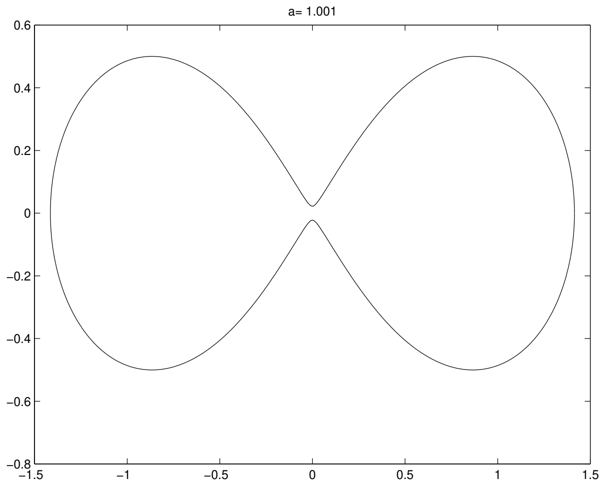

Figure 6.3: $x = \sqrt{\left(\cos 2t + \sqrt{a - \sin^2 2t}\right)} \cos t$, $y = \sqrt{\left(\cos 2t + \sqrt{a - \sin^2 2t}\right)} \sin t$.

We have tested Algorithms 4.1 and 4.2 for the cases

$$\mathtt{k} = 10, \quad \eta = 1, \quad a = 1.001, 1.0001, 1.00001, 1.0000001$$

using MATLAB. A suitable $d_{thresh} = d^{(0)}_{thresh}$ (see §4.4) for detecting non-smoothness was found empirically for $n = 128$, using wavelet transform level $l = 3$, by trial and error based on viewing the gradient matrix ,G, using MATLAB. For other transform levels the formula $d^{(l)}_{thresh} = \frac{d^{(0)}_{thresh}}{2^l}$ (see [29] Theorem 7.6) was used.

Table 6.7: Iteration count when using different preconditioners for the complex BIE matrices with $a = 1.00001$, tolerance $= 10^{-6}$.

n Preconditioner	64	128	256	512	1024	2048
NONE	*	*	*	*	*	*
Tridiagonal	147	166	*	*	*	*
Standard DWT	50	394	400	434	549	*
DWTPer band	24	24	47	44	42	40
A1	12	12	18	19	19	20
A2	18	18	24	24	26	31

For each value of a the preconditioners designed using our new algorithms performed better than the alternatives that were considered. Table 6.7 shows the number of iterations required for convergence of GMRES(25) to a tolerance of 10^{-6} for $a = 1.00001$. Table 6.8 shows the corresponding

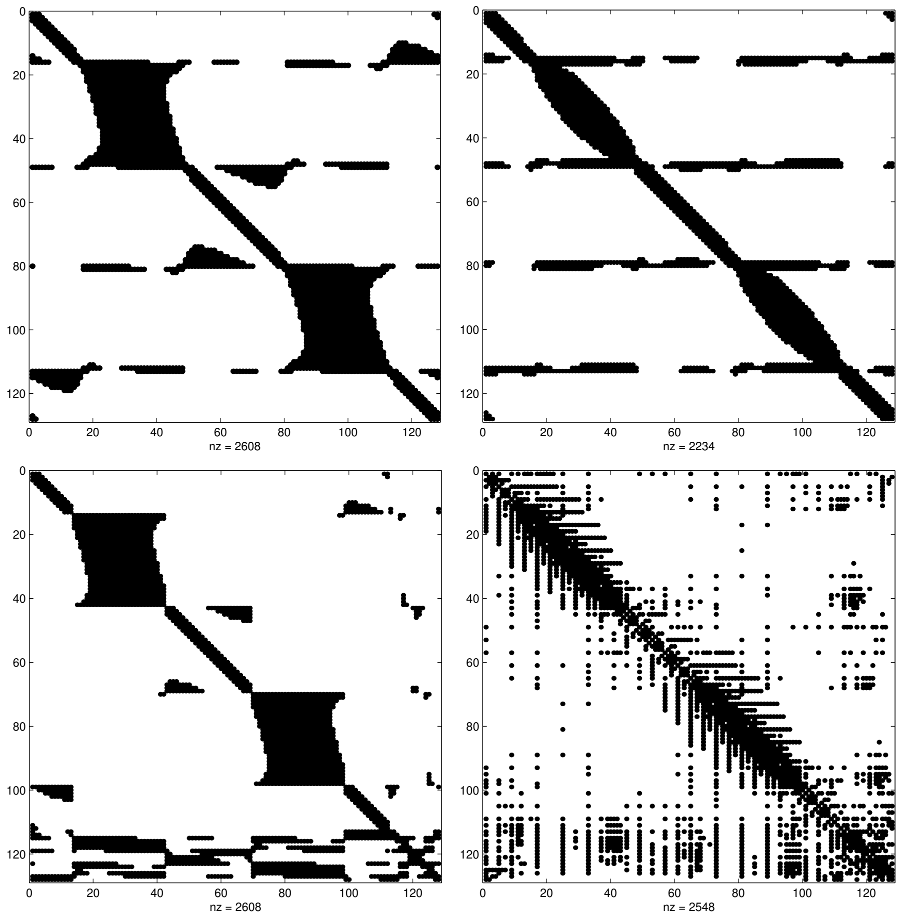

Figure 6.4: Example matrix for Helmholtz equation with 'peanut' boundary. Top left : original matrix; top right: gradient matrix, G; bottom left: matrix after permuting the local non-smooth features to the bottom ; bottom right: DWTPer of permuted matrix

Table 6.8: Flop count when using different preconditioners for the complex BIE matrices with $a = 1.00001$, tolerance $= 10^{-6}$. * denotes stagnation of GMRES(25)

n / Preconditioner	64	128	256	512	1024	2048
NONE	*	*	*	*	*	*
Tridiagonal	1.4×10^7	5.3×10^7	*	*	*	*
Standard DWT	6.6×10^6	7.1×10^7	4.9×10^8	2.0×10^9	1.8×10^{10}	*
DWTPer band	3.1×10^6	1.2×10^7	6.9×10^7	2.6×10^8	1.0×10^9	4.0×10^9
A1	2.4×10^6	1.0×10^7	3.8×10^7	1.6×10^8	6.7×10^8	2.9×10^9
A2	2.8×10^6	1.1×10^7	4.3×10^7	1.8×10^8	7.8×10^8	3.6×10^9
G. E.	8.3×10^5	6.1×10^6	4.7×10^7	3.7×10^8	2.9×10^9	2.3×10^{10}

approximate flop counts, with an additional row giving the flop counts using Gaussian elimination (G.E.). For matrices of dimension less than 256, the direct solution method is cheaper than preconditioned GMRES. For larger matrices the iterative method offers savings that increase with the matrix dimension. Approaches 1 and 2 both give improved performance when compared with the alternative preconditioning methods considered here. In both cases the solution cost increases with n^2, whereas the direct solution cost increases with n^3. Approach 1 gives faster convergence but is more expensive per iteration than Approach 2. The experimental results indicate that, for the matrices tested, Approach 1 gives the best overall performance whenever $n \geq 256$.

REMARK 6.2. In practice it is appropriate to use a tolerance for the iteration that is related to the discretisation error (see [11, 27]). This means that a larger tolerance than 10^{-6} could be used for the smaller systems, which would result in reduced iteration counts and improved performance compared with G.E.

7 Conclusion

In this paper we have developed effective DWT-based preconditioners for dense matrices with both non-smooth diagonal bands and localized off-diagonal areas of non-smoothness. For such matrices standard application of DWT does not produce useful preconditioners. Our main idea is to isolate the non-smooth features and then to make use of a discrete wavelet transform to compress the smooth part of the matrix using one of two alternative approaches. We have proposed a detection algorithm based on gradients to identify such features. We have demonstrated the effectiveness of such preconditioners by means of examples from Elastohydrodynamic Lubrication and Boundary Integral Equations.

For both of the examples that we have tested, Approach 1 has given better results (in terms of computation time) than Approach 2, but this may not always be the case. In future work we hope to find ways of predicting which of our two alternative approaches will perform best for a given linear system.

We are currently investigating preconditioning strategies for matrices arising from the solution of thermal EHL problems by a homotopy method. These matrices have a block structure including dense blocks with non-smooth diagonal bands and local features. We anticipate that we will be able to design preconditioners for such matrices based on isolation of non-smooth areas within blocks and use of a block DWTPer to compress smooth areas.

Acknowledgement

The authors wish to thank Dr. L. E. Scales from Shell Global Solutions for helping us to understand the generic EHL problem and its solution methods. We are also grateful to the anonymous referees for their helpful comments.

REFERENCES

1. S. AMINI, P. J. HARRIS, AND D. T. WILTON, *Coupled boundary and finite element methods for the solution of the dynamic fluid-structure interaction problem*, Springer-Verlag, Berlin, 1992.
2. S. AMINI AND N. MAINES, *Iterative solutions of boundary integral equations*, in Boundary Element Methods XVI, C. A. Brebbia, ed., Springer-Verlag, 1994, pp. 193–200.
3. ———, *Qualitative properties of boundary integral operators and their discretizations*, Report MCS-95-12, University of Salford, UK, 1995.
4. K. E. ATKINSON, *The numerical solution of boundary integral equations*, in The state of the art in Numerical Analysis, I. S. Duff and G. A. Watson, eds., Oxford University Press, New York, 1997, pp. 223–259.
5. O. AXELSSON, *Iterative solution methods*, Cambridge University Press, Cambridge, UK, 1996.
6. M. W. BENSON AND P. O. FREDERICKSON, *Iterative solution of large sparse linear systems arising in certain multidimensional approximation problems*, Utilitas Math., 22 (1982), pp. 127–140.
7. M. BENZI AND M. TUMA, *A sparse approximate inverse preconditioner for nonsymmetric linear systems*, SIAM J. Sci. Comput., 19 (1998), pp. 968–994.
8. G. BEYLKIN, R. R. COIFMAN, AND V. ROKHLIN, *Fast wavelet transforms and numerical algorithms I*, Commun. Pure Appl. Math., XLIV (1991), pp. 141–183.
9. D. M. BOND AND S. A. VAVASIS, *Fast wavelet transforms for matrices arising from boundary element methods*, Computer Science Research Report TR-174, Cornell University, USA, 1994.

10. T. F. Chan, W. P. Tang, and W. L. Wan, *Wavelet sparse approximate inverse preconditioners*, BIT, 37 (1997), pp. 644–660.
11. K. Chen, *Efficient iterative solution of linear systems from discretizing singular integral equations*, Elec. Trans. Numer. Anal., 2 (1994), pp. 76–91.
12. ———, *On a class of preconditioning methods for dense linear systems from boundary elements*, SIAM J. Sci. Comput., 20 (1998), pp. 684–698.
13. ———, *Discrete wavelet transforms accelerated sparse preconditioners for dense boundary element systems*, Elec. Trans. Numer. Anal., 8 (1999), pp. 138–153.
14. C. K. Chui, *Wavelets: A Mathematical Tool for Signal Analysis*, SIAM, Philadelphia, 1997.
15. I. Daubechies, *Ten Lectures on Wavelets*, SIAM, Philadelphia, 1992.
16. D. Dowson and G. R. Higginson, *Elastohydrodynamic lubrication*, Pergamon Press, Oxford, 1977.
17. J. Ford, K. Chen, and L. Scales, *A new wavelet transform preconditioner for iterative solution of elastohydrodynamic lubrication problems.*, Int. J. Comput. Maths. (to appear).
18. G. H. Golub and C. F. V. Loan, *Matrix Computations, third edition*, Johns Hopkins, Baltimore and London, 1996.
19. A. Greenbaum, V. Pták, and Z. Strakoš, *Any nonincreasing convergence curve is possible for GMRES*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 465–469.
20. M. Hochbruck and C. Lubich, *Error analysis of Krylov methods in a nutshell*, SIAM J. Sci. Comput., 19 (1998), pp. 695–701.
21. T. Hughes, C. D. Elcoate, and H. P. Evans, *A novel method for integrating first- and second-order differential equations in elastohydrodynamic lubrication for the solution of smooth isothermal, line contact problems*, International Journal for Numerical methods in Engineering, 44 (1999), pp. 1099–1113.
22. D. Miller, *Wavelet transforms and linear algebra*, MSc dissertation, University of Liverpool, UK, 1995.
23. G. E. Morales-Espejel and A. Felix-Quinonez, *Kinematics of two-sided surface features in elastohydrodynamic lubrication*, Proc. Instn. Mech. Engrs., 213 (1999), pp. 95–108.
24. N. M. Nachtigal, S. C. Reddy, and L. N. Trefethen, *How fast are nonsymmetric matrix iterations?*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 778–795.
25. O. M. Nielsen, *Wavelets in Scientific Computing*, PhD thesis, Technical University of Denmark, Lyngby, 1998.
26. T.-J. Park and K.-W. Kim, *Elastohydrodynamic lubrication of a finite line contact*, Wear, 223 (1998), pp. 102–109.
27. L. Reichel, *Parallel iterative methods for the solution of Fredholm integral equations of the second kind*, in Hypercube Multiprocessors, M. T. Heath, ed., SIAM, Philadelphia, 1987, pp. 520–529.
28. Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS, Boston, 1996.
29. G. Strang and T. Nguyen, *Wavelets and Filter Banks*, Wellesey-Cambridge Press, USA, 1996.
30. L. N. Trefethen and D. Bau, *Numerical Linear Algebra*, SIAM, Philadelphia, 1997.