APPLIED AND COMPUTATIONAL HARMONIC ANALYSIS **5,** 156–201 (1998)
ARTICLE NO. HA970227

LU Factorization of Non-standard Forms and Direct Multiresolution Solvers

D. Gines[1]

Department of Electrical Engineering, University of Colorado at Boulder, Boulder, Colorado 80309-0425

G. Beylkin[2]

Department of Applied Mathematics, University of Colorado at Boulder, Boulder, Colorado 80309-0526

and

J. Dunn[1]

Department of Electrical Engineering, University of Colorado at Boulder, Boulder, Colorado 80309-0425

Communicated by Charles K. Chui

Received May 23, 1996; revised May 12, 1997

In this paper we introduce the multiresolution LU factorization of non-standard forms (NS-forms) and develop fast *direct multiresolution* methods for solving systems of linear algebraic equations arising in elliptic problems.

The NS-form has been shown to provide a sparse representation for a wide class of operators, including those arising in strictly elliptic problems. For example, Green's functions of such operators (which are ordinarily represented by dense matrices, e.g., of size N by N) may be represented by $-\log \epsilon \cdot N$ coefficients, where ϵ is the desired accuracy.

The NS-form is not an ordinary matrix representation and the usual operations such as multiplication of a vector by the NS-form are different from the standard matrix–vector multiplication. We show that (up to a fixed but arbitrary accuracy) the sparsity of the LU factorization is maintained on any finite number of scales for self-adjoint strictly elliptic operators and their inverses. Moreover, the condition number of matrices for which we compute the usual LU factorization at different scales is $O(1)$. The direct multiresolution solver presents, therefore, an alternative to a multigrid approach and may be interpreted as a multigrid method with a single V-cycle.

[1] This research was partially supported by DARPA Grant FEE615-95-C-1756.

[2] This research was partially supported by DARPA Grant F49620-93-1-0474 and ONR Grant N00014-91-J4037.

1063-5203/98 $25.00

For self-adjoint strictly elliptic operators the multiresolution LU factorization requires only $O((-\log \epsilon)^2 \cdot N)$ operations. Combined with $O(N)$ procedures of multiresolution forward and back substitutions, it yields a fast direct multiresolution solver. We also describe direct methods for solving matrix equations and demonstrate how to construct the inverse in $O(N)$ operations (up to a fixed but arbitrary accuracy). We present several numerical examples which illustrate the algorithms developed in the paper. Finally, we outline several directions for generalization of our algorithms. In particular, we note that the multidimensional versions of the multiresolution LU factorization maintain sparsity, unlike the usual LU factorization.

Contents.

1. INTRODUCTION

In [1], a new algebraic multiresolution structure, the non-standard form (NS-form), has been introduced to represent operators in wavelet bases. As is shown in [1], for a large class of operators (which are ordinarily represented by dense matrices), the NS-form is sparse for a finite, arbitrary precision. This observation allows one to accelerate iterative methods for solving systems of linear algebraic equations which involve such operators since the cost of application of the matrix to a vector is reduced from $O(N^2)$ to $O(N)$ operations.

Since operators from a wide class admit a sparse representation in wavelet bases, it is possible to consider numerical calculus of operators [1–3], i.e., consider functions of operators computed via *fast* algorithms. The product of two operators in the NS-form requires $N \cdot (-\log \epsilon)^2$, where ϵ is the desired accuracy [4]. A fast multiplication algorithm may then be used in an iterative algorithm for constructing the generalized inverse [5–7]. For a wide class of operators it takes only $O(N \cdot (-\log \epsilon) \cdot (\log \kappa))$ operations, where κ is the condition number of the matrix, to compute the inverse operator with accuracy ϵ. Various numerical examples and applications may be found in [2, 3] (although the standard form was used in the multiplication algorithm in these papers).

In this paper we introduce a *direct* method for solving systems of linear algebraic equations and constructing the inverse using the NS-form for a class of matrices outlined below. Although direct methods have been used extensively on sparse linear systems, especially for differential operators, such algorithms suffer a loss of efficiency

if the factorized matrix does not admit the same sparse structure as the original matrix. Techniques such as graph theory have been developed to minimize the generation of so-called fill-ins [8]. Here we demonstrate that the specialized structure of NS-forms may be preserved during factorization, leading to efficient factorization algorithms using sparse data structures.

We begin by describing a factorization procedure for the NS-form. The factorization of the NS-form is superficially similar to the standard LU factorization but, in fact, is distinct in several significant ways. First, it is an approximate factorization where the accuracy, ϵ, is finite but arbitrary. Second, the factorization of NS-forms requires $O(N\cdot(-\log\epsilon)^2)$ operations for operators arising from strictly elliptic problems. Combined with $O(N\cdot(-\log\epsilon))$ procedures of ''multiscale'' forward and back substitutions, it yields a direct multiresolution solver. Third, the actual LU factorization is performed on well-conditioned matrices even if the original matrix (arising from a strictly elliptic problem) has a large condition number. Using the multiresolution solver, we also construct the inverse in $O(N(-\log\epsilon)^2))$ operations. We note that in problems where the choice of the size of the matrix and of accuracy are connected, typically $\epsilon \propto N^{-\alpha}$, $\alpha > 0$.

Our direct multiresolution solver presents an alternative to an iterative multigrid approach. In fact, the algorithms of this paper may be viewed as a ''direct multigrid,'' without V and W cycles (or, more precisely within this terminology, with a single V cycle). The absence of cycles of the usual full multigrid methods (for references see [9]) is easy to explain since we generate (within computational accuracy) a linear system for the exact projection of the solution on coarse scales. Once such a system is solved, there is no need to revisit that scale again. Thus, the algorithms of this paper provide a connection between multigrid methods and classical techniques of Gaussian elimination and LU decomposition. In this role, our approach provides a systematic algebraic structure to multiresolution computations (what we call multiresolution linear algebra), and we go to some length to provide details of such algebraic operations.

We recall that the non-standard form is not an ordinary matrix. The NS-form has a multiresolution structure, and the usual operations such as multiplication of a vector by the NS-form or multiplication of NS-forms are different from the standard matrix–vector and matrix–matrix multiplications. The remarkable feature of the non-standard form is the decoupling it achieves among the scales.

The outline of the paper is as follows: in Sections 2 and 3 we introduce multiresolution analysis and the notion of the non-standard form which serve as a foundation for the algorithms presented in the paper. We do so without specific reference to the properties of wavelets (e.g., number of vanishing moments, size of the support, etc). This allows us to describe the algebraic structure of the algorithms without considering specific bases. On the other hand, the sparsity of the non-standard form for a given accuracy, and thus the operation count of the algorithms, does depend on the choice of the basis. We discuss the issues of sparsity separately in Section 5.

In Section 4 we describe multiresolution LU factorization. In particular, we describe in Section 4.2 the procedure for computing lower and upper NS-forms where the NS-form of an operator has been precomputed. In Section 4.3 we describe how to construct the lower and upper NS-forms directly from the original operator, without precomput-

ing the NS-form; such a procedure is computationally more efficient. Finally, in Section 4.4, we present an alternative version of the factorization procedure which may be useful during partial pivoting (although pivoting is needed for matrices outside the class for which we prove that the multiresolution LU factorization is sparse. We discuss partial pivoting in the Appendix).

In Section 6 we show how the factorization procedure may be incorporated into a fast direct solver for linear systems of equations. Toward that end, we describe the algorithms of multiresolution forward and backward substitution. We then describe in Section 7 how the direct methods may be applied to solving matrix equations. The forward and backward substitution algorithms for matrices may be used to construct the inverse operator, which we describe separately in Section 7.5.

In Section 8 we describe the relationship between the non-standard and standard forms, and demonstrate that factorization using S-forms leads to sparse matrices if NS-forms are sparse.

In Section 9 we present various numerical examples which illustrate the algorithms developed in the paper. The purpose of these tests is to show the behavior of such algorithms as the size of matrices becomes large.

Finally, in Section 10 we make a number of observations concerning generalization of our approach. In particular, we note that the multidimensional versions of the multiresolution LU factorization maintain sparsity, unlike the usual LU decomposition. Thus, fast (adaptive) direct methods may be developed for elliptic problems in multiple dimensions, but we leave this subject matter for another paper.

2. PRELIMINARY CONSIDERATIONS

We start by reviewing notions of multiresolution analysis (see [10, 11]) and the non-standard form [1] and describe algorithms for applying the NS-form to vectors. We introduce the NS-form using projection operators without invoking wavelet bases explicitly. This approach allows us to explain the algebraic structure of algorithms separately from considerations of sparsity. The sparsity of structures that we develop in this paper is critical in obtaining fast algorithms. We defer the discussion of sparsity and its dependence on the choice of the wavelet basis to Section 5.

2.1. Multiresolution Analysis for Operators

Let us consider a multiresolution analysis of $\mathbf{L}^2(\mathbf{R}^d)$,

$$\mathbf{V}_n \subset \cdots \subset \mathbf{V}_2 \subset \mathbf{V}_1 \subset \mathbf{V}_0 \subset \mathbf{V}_{-1} \subset \mathbf{V}_{-2} \subset \cdots, \tag{2.1}$$

and define subspaces $\mathbf{W}_j$ as orthogonal complements of $\mathbf{V}_j$ in $\mathbf{V}_{j-1}$, $\mathbf{V}_{j-1} = \mathbf{V}_j \oplus \mathbf{W}_j$, so that $\mathbf{L}^2(\mathbf{R}^d) = \mathbf{V}_n \oplus_{j \le n} \mathbf{W}_j$. If the number of scales is finite, then we set $j = 0$ to be the finest scale and consider

$$\mathbf{V}_n \subset \cdots \subset \mathbf{V}_2 \subset \mathbf{V}_1 \subset \mathbf{V}_0, \qquad \mathbf{V}_0 \subset \mathbf{L}^2(\mathbf{R}^d) \tag{2.2}$$

instead of (2.1). In numerical realizations the subspace $\mathbf{V}_0$ is always of a finite dimension.

Let T be an operator

$$T: \mathbf{L}^2(\mathbf{R}^d) \to \mathbf{L}^2(\mathbf{R}^d). \tag{2.3}$$

By defining projection operators on the subspace $\mathbf{V}_j$, $j \in \mathbf{Z}$,

$$P_j: \mathbf{L}^2(\mathbf{R}^d) \to \mathbf{V}_j, \tag{2.4}$$

and expanding T in a ''telescopic'' series, we obtain

$$T = \sum_{j=-\infty}^{n} (Q_j T Q_j + Q_j T P_j + P_j T Q_j) + P_n T P_n, \tag{2.5}$$

where $Q_j = P_{j-1} - P_j$ is the projection operator on the subspace $\mathbf{W}_j$,

$$Q_j: \mathbf{L}^2(\mathbf{R}^d) \to \mathbf{W}_j. \tag{2.6}$$

If the scale $j = 0$ is the finest scale, then

$$T_0 = \sum_{j=1}^{n} (Q_j T Q_j + Q_j T P_j + P_j T Q_j) + P_n T P_n, \tag{2.7}$$

where $T \sim T_0 = P_0 T P_0$ is a discretization of T on the finest scale. Expansions (2.5) and (2.7) decompose T into a sum of contributions from different scales.

2.1.1. The Non-standard Form

The NS-form introduced in [1] is a representation of an operator T as a chain of triplets $T = \{\{A_j, B_j, C_j\}_{-\infty \le j \le n}, T_n\}$, where operators A_j, B_j, C_j (as well as T_j) are defined as

$$A_j = Q_j T Q_j, \tag{2.8a}$$

$$B_j = Q_j T P_j, \tag{2.8b}$$

$$C_j = P_j T Q_j, \tag{2.8c}$$

$$T_j = P_j T P_j, \tag{2.8d}$$

and admit the recursive definition

$$T_j = A_{j+1} + B_{j+1} + C_{j+1} + T_{j+1}. \tag{2.9}$$

The operators A_j, B_j, C_j make up the blocks of the NS-form and operate on the subspaces $\mathbf{V}_j$ and $\mathbf{W}_j$,

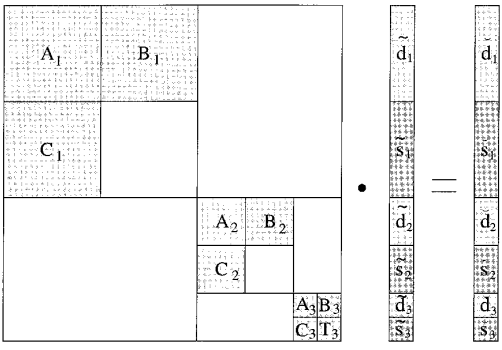

FIG. 1. Organization of the non-standard form of a matrix. The submatrices A_j, B_j, and C_j, $j = 1, 2, 3$, and T_3 are the only non-zero submatrices.

$$A_j: \mathbf{W}_j \to \mathbf{W}_j, \tag{2.10a}$$

$$B_j: \mathbf{V}_j \to \mathbf{W}_j, \tag{2.10b}$$

$$C_j: \mathbf{W}_j \to \mathbf{V}_j, \tag{2.10c}$$

whereas operators T_j operate on subspaces $\mathbf{V}_j$,

$$T_j: \mathbf{V}_j \to \mathbf{V}_j. \tag{2.11}$$

The wavelet transform recursively represents operators T_j as

$$\begin{pmatrix} A_{j+1} & B_{j+1} \\ C_{j+1} & T_{j+1} \end{pmatrix}, \tag{2.12}$$

which is a mapping

$$\begin{pmatrix} A_{j+1} & B_{j+1} \\ C_{j+1} & T_{j+1} \end{pmatrix}: \mathbf{W}_{j+1} \oplus \mathbf{V}_{j+1} \to \mathbf{W}_{j+1} \oplus \mathbf{V}_{j+1}, \tag{2.13}$$

where $\mathbf{V}_j = \mathbf{W}_{j+1} \oplus \mathbf{V}_{j+1}$. If the number of scales is finite, then we obtain $T_0 = \{\{A_j, B_j, C_j\}_{1 \le j \le n}, T_n\}$, and the blocks of the NS-form are organized as blocks of a matrix shown in Fig. 1.

We note that for $d \ge 2$ the blocks of the NS-form have additional structure. From now on we will assume that $d = 1$, although many considerations are essentially the same for $d \ge 2$. We will defer additional remarks about dimensions $d \ge 2$ to Section 10.

Remark 2.1. Since projection operators involve subsampling, equalities like that in (2.9) may appear inconsistent if we compare sizes of blocks in Fig. 1. For example, the size of blocks A_3, B_3, C_3, and T_3 is not the same as that of T_2. The transition from operator notation as in (2.9) to a matrix notation involves combining the blocks A_3, B_3, C_3, and T_3 as in (2.12), and performing one step of the two-dimensional

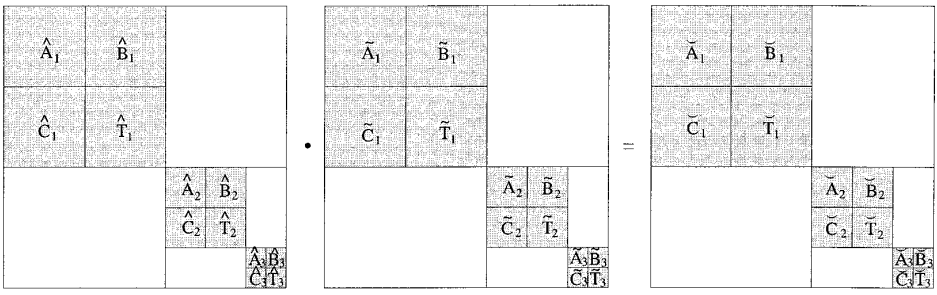

FIG. 2. Organization of the multiresolution product of $\hat{T}_0$ and $\tilde{T}_0$, which is based on extended NS-forms.

discrete wavelet transform (an orthogonal transformation). To obtain the blocks A_j, B_j, C_j, and T_j, for example, we compute the wavelet *decomposition* of T_{j-1},

$$\begin{pmatrix} A_j & B_j \\ C_j & T_j \end{pmatrix} = WT_{j-1}W^*, \tag{2.14}$$

where W is an orthogonal matrix representing one level of the discrete wavelet transform. Similarly, the block T_{j-1} may be *reconstructed* from A_j, B_j, C_j, and T_j by computing

$$T_{j-1} = W^*\begin{pmatrix} A_j & B_j \\ C_j & T_j \end{pmatrix}W. \tag{2.15}$$

With a slight abuse of notation, we will use the same letters to denote operators and matrices, and we will rely on the specific context for separating the two.

Remark 2.2 (Alternate Matrix Representations). Since in this paper we work with several matrix structures, we provide a brief summary:

An extended non-standard form is the NS-form which includes the blocks T_j at all scales and is denoted $T_0 = \{A_j, B_j, C_j, T_j\}_{1\le j\le n}$, as shown in Fig. 2. This representation is used for multiplying matrices or computing the matrix inverse, and its full description is found in Section 3.3.

An intermediate non-standard form is used to hold intermediate values and is denoted $T_0 = \{\breve{A}_j, \breve{B}_j, \breve{C}_j, \breve{T}_j\}_{1\le j\le n}$ (see Fig. 2). The symbol ($\breve{}$) (pronounced ''brehv'') is used throughout the paper to indicate that values are intermediate and that additional projections are required. For a full description, see Section 3.3.

The matrix representation on $\mathbf{V}_0$ is simply the ordinary matrix at the finest scale, T_0: $\mathbf{V}_0 \to \mathbf{V}_0$.

The standard form is the representation of a discretized operator in the tensor-product wavelet basis. See Section 8 for details.

2.2. Multiresolution Analysis for Functions

Using MRA with finite number of scales (2.2), we decompose a vector $f_0 \in \mathbf{V}_0$ onto subspaces $\mathbf{V}_j$ and $\mathbf{W}_j$ using projection operators P_j and Q_j,

$$f_0 = \sum_{j=1}^{n} Q_j f + P_n f, \tag{2.16}$$

and obtain the wavelet expansion $f_0 = \{\{d_j\}_{1 \le j \le n}, s_n\}$, where

$$d_j = Q_j f \in \mathbf{W}_j, \tag{2.17a}$$

$$s_j = P_j f \in \mathbf{V}_j. \tag{2.17b}$$

The terms in (2.17) admit the recursive definition

$$s_{j-1} = d_j + s_j. \tag{2.18}$$

Remark 2.3. We note again that the transition from operator notation as in (2.18) to a vector notation involves combining the vectors d_j and s_j, and performing one step of the one-dimensional discrete wavelet transform. The vectors d_j and s_j may be obtained by computing the decomposition of s_{j-1},

$$\{d_j, s_j\}^T = W s_{j-1}, \tag{2.19}$$

where W is an orthogonal matrix representing one level of the discrete wavelet transform. Similarly, s_{j-1} may be reconstructed from d_j and s_j by computing

$$s_{j-1} = W^* \{d_j, s_j\}^T. \tag{2.20}$$

Remark 2.4. If $\{f_1, f_2, \ldots, f_N\}$ represent N coefficients of a function $f \in \mathbf{V}_j$, then the coefficients $\{d_j, s_j\}$ may be computed using (2.19) at a cost of $O(N)$ via a pyramid scheme as described in, e.g., [1].

Remark 2.5 (Alternate Vector Representations). There are several representations of vectors which we use throughout the paper:

An extended representation includes vectors s_j at all scales, $f_0 = \{d_j, s_j\}_{1 \le j \le n}$. This representation is first described in Section 3.2.

The intermediate representation contains intermediate results and is denoted $f_0 = \{\check{d}_j, \check{s}_j\}_{1 \le j \le n}$. See Section 3.2 for details.

3. MULTIRESOLUTION LINEAR ALGEBRA

The approach we develop in this paper for solving systems of linear algebraic equations allows us to introduce the notion of multiresolution linear algebra. By this term we mean linear algebra applied to NS-forms, rather than standard matrix representations. It turns out that in order to develop multiresolution linear algebraic algorithms, all we need to do is interlace the operations of standard linear algebra with projections. The linear algebraic operations (e.g. matrix–vector, matrix–matrix multiplication) are confined to a given scale, whereas projections convey information *between* scales. Before describing this approach in greater detail within specific algorithms, let us describe matrix operations and projection procedures common to all algorithms of this paper.

3.1. Matrix Operations and Projections

The matrix operations at each scale involve blocks of the original NS-form *and* blocks obtained via projections. As an example, in the matrix–matrix multiplication of two NS-forms, $\hat{T}_0$ and $\tilde{T}_0$, we compute $A_j = \check{A}_j + \bar{A}_j$, where $\check{A}_j = \hat{A}_j\tilde{A}_j + \hat{B}_j\tilde{C}_j$ is computed from the original NS-forms, whereas $\bar{A}_j$ is a projection onto scale j, from scale $j - 1$. Relations of this type show how projections and matrix operations are combined at each scale. Such relations define the necessary algebraic operations for a given algorithm, and we will refer to them as the governing equations for that algorithm.

On each scale, the multiresolution algorithms produce intermediate matrices or vectors, some of which are projected to other scales. To describe the projection procedure, let us consider some intermediate vectors $\{\check{s}_j\}_{1\le j\le n}$. We need to project all vectors $\check{s}_j$ on subspaces $\mathbf{W}_k$, $k = j, \ldots, n$. Instead of projecting each part of the vector separately, we combine contributions on a given scale before applying the projection operator. To illustrate, let us assume that projections have been completed up to scale $k - 1$ and describe the procedure for scale k. We proceed in a recursive manner,

$$\check{s}_{k-1} + \bar{s}_{k-1} = \bar{d}_k + \bar{s}_k, \tag{3.1}$$

where $\bar{s}_{k-1}$ is a projection from the previous scale, and

$$\bar{d}_k = Q_k(\check{s}_{k-1} + \bar{s}_{k-1}), \tag{3.2a}$$

$$\bar{s}_k = P_k(s_{k-1} + \bar{s}_{k-1}), \tag{3.2b}$$

are computed via the wavelet transform described in Remark 2.3. Similarly, for operators, we compute projections of the intermediate variables $\{\check{T}_j\}_{1\le j\le n}$ for $k = j, \ldots, n$. We have

$$\check{T}_{k-1} + \bar{T}_{k-1} = \bar{A}_k + \bar{B}_k + \bar{C}_k + \bar{T}_k, \tag{3.3}$$

where $\bar{T}_{k-1}$ is a projection from the previous scale, and

$$\bar{A}_k = Q_k(\check{T}_{k-1} + \bar{T}_{k-1})Q_k, \tag{3.4a}$$

$$\bar{B}_k = Q_k(\check{T}_{k-1} + \bar{T}_{k-1})P_k, \tag{3.4b}$$

$$\bar{C}_k = P_k(\check{T}_{k-1} + \bar{T}_{k-1})Q_k, \tag{3.4c}$$

$$\bar{T}_k = P_k(\check{T}_{k-1} + \bar{T}_{k-1})P_k, \tag{3.4d}$$

are computed via the wavelet transform described in Remark 2.1. Relations such as (3.1) and (3.3) will be refered to as the projection equations for a particular algorithm.

3.2. Application of NS-Forms to Vectors

Let us first describe multiresolution matrix–vector multiplication. Let us denote by $T_0 = \{\{A_j, B_j, C_j\}_{1\le j\le n}, T_n\}$ the NS-form of an operator T_0, and let $f_0 = \{\tilde{d}_j, \tilde{s}_j\}_{1\le j\le n}$

be an extended wavelet representation for f_0. In what follows we define the *multiresolution product* of T_0 and f_0 as

$$g_0 = T_0 \cdot f_0, \tag{3.5}$$

where g_0 is the same vector obtained using matrix–vector multiplication with the usual representations of T_0 and f_0.

In order to derive the necessary matrix operations for (3.5), we write a telescopic series,

$$T_0 f_0 - T_n f_n = \sum_{j=1}^{n} (P_{j-1} T_0 P_{j-1})(P_{j-1} f_0) - (P_j T_0 P_j)(P_j f_0). \tag{3.6}$$

Since $P_{j-1} = P_j + Q_j$ we obtain

$$T_0 f_0 - T_n f_n = \sum_{j=1}^{n} (Q_j T_0 Q_j)(Q_j f_0) + (Q_j T_0 P_j)(P_j f_0) + (P_j T_0 Q_j)(Q_j f_0), \tag{3.7}$$

or, using (2.8) and (2.17),

$$T_0 f_0 = \sum_{j=1}^{n} (A_j \tilde{d}_j + B_j \tilde{s}_j + C_j \tilde{d}_j) + T_n \tilde{s}_n. \tag{3.8}$$

For the terms in (3.8) we have

$$A_j \tilde{d}_j + B_j \tilde{s}_j \in \mathbf{W}_j, \tag{3.9a}$$

$$C_j \tilde{d}_j \in \mathbf{V}_j, \tag{3.9b}$$

for $j = 1, 2, \ldots, n$, and

$$T_n \tilde{s}_n \in \mathbf{V}_n. \tag{3.10}$$

Let us denote

$$\check{d}_j = A_j \tilde{d}_j + B_j \tilde{s}_j, \tag{3.11a}$$

$$\check{s}_j = C_j \tilde{d}_j, \tag{3.11b}$$

for $j = 1, 2, \ldots, n$, and on the last scale,

$$\check{s}_n = C_n \tilde{d}_n + T_n \tilde{s}_n. \tag{3.12}$$

The computations in (3.11) and (3.12) are performed using the usual matrix operations at each scale and may be organized as shown in Fig. 1.

In order to convert $\{\check{d}_j, \check{s}_j\}_{1 \le j \le n}$ to a wavelet representation $\{\{d_j\}_{1 \le j \le n}, s_n\}$, we expand vectors $\check{s}_j$ by setting $\bar{s}_1 = \bar{d}_1 = 0$, and, for $j = 1, 2, \ldots, n$, computing the projections

$$\check{s}_{j-1} + \bar{s}_{j-1} = \bar{d}_j + \bar{s}_j, \tag{3.13}$$

via (3.2), and by forming the sum

$$d_j = \check{d}_j + \bar{d}_j. \tag{3.14}$$

At the last scale we compute

$$s_n = \check{s}_n + \bar{s}_n. \tag{3.15}$$

Equations (3.14) and (3.15) are the governing equations for multiresolution matrix vector multiplication, while (3.13) is the corresponding projection equation. The following algorithm summarizes the process:

ALGORITHM 3.1 (Application of NS-Forms to Vectors). *Given the* NS-*form* $T_0 = \{\{A_j, B_j, C_j\}_{1 \le j \le n}, T_n\}$, *and the extended wavelet representation* $f_0 = \{\check{d}_j, \check{s}_j\}_{1 \le j \le n}$, *we compute the multiresolution product* $T_0 \cdot f_0 = b_0$ *where* $b_0 = \{\{d_j\}_{1 \le j \le n}, s_n\}$ *using the following steps:*

1. *Initialization: set* $\bar{d}_1 = \bar{s}_1 = 0$.
2. *For* $j = 1, 2, \ldots, n$, *compute*
   (a) $\check{d}_j$ *and* $\check{s}_j$ *via* (3.11),
   (b) $\bar{d}_j$ *and* $\bar{s}_j$ $(j \neq 1)$ *via* (3.13),
   (c) d_j *via* (3.14).
3. *At the last scale compute* s_n *via* (3.15).

Computational cost. For operators with sparse NS-forms (see [1] and Section 5) Algorithm 3.1 requires $O((-\log \epsilon)N)$ operations, where ϵ is the desired accuracy, and $N \times N$ is the dimension of T_0 in the ordinary matrix representation. The cost of Step 2a is proportional to number of non-zero elements, N_s, in the blocks of the NS-form, which is $N_s = C(-\log \epsilon)N$, where C is a constant. Step 2b is computed in $O((-\log \epsilon)N)$ operations using the pyramid scheme. Step 2c clearly requires $O((-\log \epsilon)N)$ operations. The number of vanishing moments of the wavelet transform (and, thus, its cost) is usually chosen to be proportional to the number of accurate digits $-\log \epsilon$, see [1].

3.3. Multiplication of NS-Forms

In this section we describe an algorithm for the multiplication of NS-forms [4], in order to derive the governing equations for multiresolution LU factorization. Let us denote by $\hat{T}_0 = \{\hat{A}_j, \hat{B}_j, \hat{C}_j, \hat{T}_j\}$ and $\tilde{T}_0 = \{\tilde{A}_j, \tilde{B}_j, \tilde{C}_j, \tilde{T}_j\}$ the extended NS-forms of operators $\hat{T}_0$ and $\tilde{T}_0$. In what follows we define the *multiresolution product* of $\hat{T}_0$ and $\tilde{T}_0$ as

$$\hat{T}_0 \cdot \tilde{T}_0 = T_0, \tag{3.16}$$

where T_0 is the same NS-form obtained from the usual product of $\hat{T}_0$ and $\tilde{T}_0$ in $\mathbf{V}_0$.

In the following algorithm we use operators $\hat{T}_j$ and $\tilde{T}_j$. We note that the algorithm requires only a band around the diagonal of their matrices, even though these operators are generally not sparse. We will consider sparsity of the blocks of NS-forms separately and here present a formal derivation of the algorithm for multiplication of NS-forms following [4].

In order to derive the necessary matrix operations, we again write a telescopic series,

$$\hat{T}_0\tilde{T}_0 - \hat{T}_n\tilde{T}_n = \sum_{j=1}^{j=n} [(P_{j-1}\hat{T}_0P_{j-1})(P_{j-1}\tilde{T}_0P_{j-1}) - (P_j\hat{T}_0P_j)(P_j\tilde{T}_0P_j)]. \quad (3.17)$$

Using the relation $P_{j-1} = P_j + Q_j$ and Eq. (2.8) we obtain

$$\hat{T}_0\tilde{T}_0 - \hat{T}_n\tilde{T}_n = \sum_{j=1}^{j=n} [(\hat{A}_j\tilde{A}_j + \hat{B}_j\tilde{C}_j) + (\hat{A}_j\tilde{B}_j + \hat{B}_j\tilde{T}_j) + (\hat{C}_j\tilde{A}_j + \hat{T}_j\tilde{C}_j) + \hat{C}_j\tilde{B}_j]. \quad (3.18)$$

The operators in (3.18) are acting on following subspaces,

$$\hat{A}_j\tilde{A}_j + \hat{B}_j\tilde{C}_j\colon \mathbf{W}_j \to \mathbf{W}_j, \quad (3.19a)$$

$$\hat{A}_j\tilde{B}_j + \hat{B}_j\tilde{T}_j\colon \mathbf{V}_j \to \mathbf{W}_j, \quad (3.19b)$$

$$\hat{C}_j\tilde{A}_j + \hat{T}_j\tilde{C}_j\colon \mathbf{W}_j \to \mathbf{V}_j, \quad (3.19c)$$

$$\hat{C}_j\tilde{B}_j\colon \mathbf{V}_j \to \mathbf{V}_j, \quad (3.19d)$$

for $j = 1, 2, \ldots, n$, and

$$\hat{T}_n\tilde{T}_n\colon \mathbf{V}_n \to \mathbf{V}_n. \quad (3.20)$$

Let us denote

$$\check{A}_j = \hat{A}_j\tilde{A}_j + \hat{B}_j\tilde{C}_j, \quad (3.21a)$$

$$\check{B}_j = \hat{A}_j\tilde{B}_j + \hat{B}_j\tilde{T}_j, \quad (3.21b)$$

$$\check{C}_j = \hat{C}_j\tilde{A}_j + \hat{T}_j\tilde{C}_j, \quad (3.21c)$$

$$\check{T}_j = \hat{C}_j\tilde{B}_j, \quad (3.21d)$$

for $j = 1, 2, \ldots, n$, and on the last scale

$$\check{T}_n = \hat{C}_n\tilde{B}_n + \hat{T}_n\tilde{T}_n. \quad (3.22)$$

The computations in (3.21) and (3.22) require the usual matrix operations at each scale and may be organized as shown in Fig. 2.

In order to construct a non-standard form $\{\{A_j, B_j, C_j\}_{1\le j\le n}, T_n\}$ from the blocks $\{\check{A}_j, \check{B}_j, \check{C}_j, \check{T}_j\}_{1\le j\le n}$, we expand the operators $\check{T}_j$ by setting $\bar{A}_1 = \bar{B}_1 = \bar{C}_1 = \bar{T}_1 = 0$, and, for $j = 1, 2, \ldots, n$, computing the projections

$$\check{T}_{j-1} + \bar{T}_{j-1} = \bar{A}_j + \bar{B}_j + \bar{C}_j + \bar{T}_j, \tag{3.23}$$

via (3.4), and by forming the sums

$$A_j = \check{A}_j + \bar{A}_j, \tag{3.24a}$$

$$B_j = \check{B}_j + \bar{B}_j, \tag{3.24b}$$

$$C_j = \check{C}_j + \bar{C}_j. \tag{3.24c}$$

At the last scale we compute

$$T_n = \check{T}_n + \bar{T}_n. \tag{3.25}$$

Equations (3.24) and (3.25) are the governing equations for multiresolution matrix multiplication, and (3.23) is the corresponding projection equation. We note that the terms T_j may also be obtained, if desired, by combining all operators which act on the subspace $\mathbf{V}_j \to \mathbf{V}_j$. From Eqs. (3.19d), (3.20), and (3.21d) we have

$$T_j = \hat{T}_j\tilde{T}_j + \check{T}_j + \bar{T}_j. \tag{3.26}$$

ALGORITHM 3.2 (Product of NS-Forms). *Given the extended* NS-*forms* $\hat{T}_0 = \{\hat{A}_j, \hat{B}_j, \hat{C}_j, \hat{T}_j\}_{1\le j\le n}$, *and* $\tilde{T}_0 = \{\tilde{A}_j, \tilde{B}_j, \tilde{C}_j, \tilde{T}_j\}_{1\le j\le n}$, *we compute the multiresolution product* $\hat{T}_0 \cdot \tilde{T}_0 = T_0$, *where* $T_0 = \{\{A_j, B_j, C_j\}_{1\le j\le n}, T_n\}$ *is an* NS-*form, using the following steps:*

1. *Initialization: set* $\bar{A}_1 = \bar{B}_1 = \bar{C}_1 = \bar{T}_1 = 0$.
2. *For* $j = 1, 2, \ldots, n$ *compute*
   (a) $\check{A}_j, \check{B}_j, \check{C}_j$, *and* $\check{T}_j$ *via* (3.21),
   (b) $\bar{A}_j, \bar{B}_j, \bar{C}_j$, *and* $\bar{T}_j$ $(j \neq 1)$ *via* (3.23),
   (c) A_j, B_j, C_j *via* (3.24).
3. *At the last scale compute* T_n *via* (3.25).

Computational cost. For operators whose NS-form admits a sparse structure, the above algorithm may be shown to be of $O((-\log \epsilon)^2 N)$, where $N \times N$ is the dimension of T_0 in the usual representation. It is shown in [4] that, for a wide class of operators, Step 2a involves multiplication of banded matrices, and only a band around the diagonal of $\hat{T}_j$ and $\tilde{T}_j$ is needed. The number of operations is shown to be proportional to $(-\log \epsilon)^2 N$, since the bandwidth is proportional to $-\log \epsilon$. Step 2b is done via the pyramid algorithm and, since $\hat{C}_j\tilde{B}_j$, $j = 1, \ldots, n$ are banded, the total number of operations at this step is also proportional to $(-\log \epsilon)N$. Finally, Step 2c requires $O(-\log \epsilon)N)$ operations.

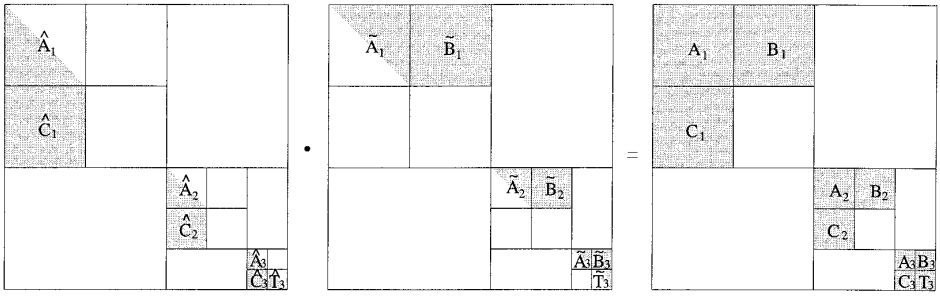

FIG. 3. Organization of the multiresolution LU factorization.

4. MULTIRESOLUTION LU FACTORIZATION

We now describe the main tool in our approach, LU factorization with respect to the multiresolution product $(\cdot)$ introduced in Section 3.3. Given an NS-form $T_0 = \{\{A_j, B_j, C_j\}_{1\le j\le n}, T_n\}$, our goal is to compute the NS-forms $\hat{T}_0$ and $\tilde{T}_0$, such that

$$T_0 = \hat{T}_0 \cdot \tilde{T}_0, \tag{4.1}$$

where $\hat{T}_0$ and $\tilde{T}_0$ are analogous to lower and upper triangular matrices. In defining lower and upper triangular NS-forms, we require that $\hat{B}_j = \tilde{C}_j = 0$ for $j = 1, 2, \ldots, n$; that blocks $\{\{\hat{A}_j\}_{1\le j\le n}, \hat{T}_n\}$ are lower triangular; and that blocks $\{\{\tilde{A}_j\}_{1\le j\le n}, \tilde{T}_n\}$ are upper triangular, as shown in Fig. 3. We call $\hat{T}_0 = \{\{\hat{A}_j, \hat{C}_j\}_{1\le j\le n}, \hat{T}_n\}$ and $\tilde{T}_0 = \{\{\tilde{A}_j, \tilde{B}_j\}_{1\le j\le n}, \tilde{T}_n\}$ lower and upper NS-forms, respectively. We note that if we convert $\hat{T}_0$ and $\tilde{T}_0$ to their corresponding S-forms (see Section 8), then these representations are the usual lower and upper triangular matrices in the wavelet system of coordinates.

The purpose of factorizing NS-forms into lower and upper NS-forms is completely analogous to that of ordinary LU factorization; namely, the goal is to obtain a direct solver. Unlike with ordinary LU factorization, the NS-form blocks being factorized are well conditioned for a class of operators associated with elliptic problems. We illustrate this point in examples of Section 9.

We first consider the multiplication of lower and upper NS-forms $\hat{T}_0$ and $\tilde{T}_0$ to obtain the governing equations, and then develop an algorithm for the reverse process of obtaining $\hat{T}_0$ and $\tilde{T}_0$ as factors of T_0.

4.1. Multiplication of Lower and Upper NS-Forms

For lower and upper NS-forms, the matrix operations described in Section 3.3 simplify, since we require that $\hat{B}_j = 0$ and $\tilde{C}_j = 0$ for $j = 1, 2, \ldots, n$. We combine formulas (3.21) and (3.24), and eliminate the intermediate blocks $\breve{A}$, $\breve{B}$, $\breve{C}$, $\breve{T}$, to obtain

$$A_j = \hat{A}_j\tilde{A}_j + \bar{A}_j, \tag{4.2a}$$

$$B_j = \hat{A}_j\tilde{B}_j + \bar{B}_j, \tag{4.2b}$$

$$C_j = \hat{C}_j\tilde{A}_j + \bar{C}_j, \tag{4.2c}$$

for $j = 1, 2, \ldots, n$, and from (3.22) and (3.25),

$$T_n = \hat{C}_n \tilde{B}_n + \hat{T}_n \tilde{T}_n + \bar{T}_n. \tag{4.3}$$

The operators $\bar{A}_j$, $\bar{B}_j$, $\bar{C}_j$ are computed by first setting $\bar{A}_1 = \bar{B}_1 = \bar{C}_1 = \bar{T}_1 = 0$, and then, for $j = 1, 2, \ldots, n$, computing the projections as in (3.4),

$$\hat{C}_{j-1} \tilde{B}_{j-1} + \bar{T}_{j-1} = \bar{A}_j + \bar{B}_j + \bar{C}_j + \bar{T}_j. \tag{4.4}$$

4.2. Factorization

Let us now assume that T_0 is given and obtain a recurrence relation that permits us to compute the lower and upper NS-forms $\hat{T}_0$ and $\tilde{T}_0$. According to (4.2) and (4.3), the blocks of the NS-forms $\hat{T}_0$ and $\tilde{T}_0$ satisfy the relations

$$\hat{A}_j \tilde{A}_j = A_j - \bar{A}_j, \tag{4.5a}$$

$$\hat{A}_j \tilde{B}_j = B_j - \bar{B}_j, \tag{4.5b}$$

$$\hat{C}_j \tilde{A}_j = C_j - \bar{C}_j, \tag{4.5c}$$

on all scales $j = 1, 2, \ldots, n$, and on the last scale,

$$\hat{T}_n \tilde{T}_n = T_n - \bar{T}_n - \hat{C}_n \tilde{B}_n. \tag{4.6}$$

Equations (4.5) and (4.6) are the governing equations for multiresolution LU factorization, and (4.4) is the corresponding projection equation. Comparing (4.5) and (4.6) to (4.2) and (4.3) we note that for the purposes of finding $\hat{A}_j$, $\tilde{A}_j$, $\hat{C}_j$, and $\tilde{B}_j$, the order of operations is reversed; namely, *first* projections are *subtracted*, and *then* matrix operations are performed. To make the procedure clear, let us assume that factorization has been completed up to scale $k - 1$, and describe the procedure for scale k. Thus, let us assume Eqs. (4.5) are satisfied for $1 \le j \le k - 1$.

The first step is to compute projections of the matrix $\hat{C}_{k-1} \tilde{B}_{k-1} + \bar{T}_{k-1}$ from scale $k - 1$ to obtain $\bar{A}_k$, $\bar{B}_k$, $\bar{C}_k$, $\bar{T}_k$, as in (4.4). Next, we compute the usual LU factorization,

$$A_k - \bar{A}_k = L_k U_k, \tag{4.7}$$

and set $\hat{A}_k = L_k$ and $\tilde{A}_k = U_k$. Given $\hat{A}_k$ and $\tilde{A}_k$, we obtain $\tilde{B}_k$ and $\hat{C}_k$ by solving

$$\hat{A}_k \tilde{B}_k = B_k - \bar{B}_k, \tag{4.8a}$$

$$\hat{C}_k \tilde{A}_k = C_k - \bar{C}_k, \tag{4.8b}$$

using the usual forward and backward substitution. We have now satisfied (4.5) for $j = k$. On the final scale, $j = n$, we use usual LU factorization to compute

$$T_n - \bar{T}_n - \hat{C}_n \tilde{B}_n = L_n U_n, \tag{4.9}$$

and set $\hat{T}_n = L_n$ and $\tilde{T}_n = U_n$. We summarize our results as

ALGORITHM 4.1 (Multiresolution LU Factorization). *Given the* NS-*form* $T_0 = \{\{A_j, B_j, C_j\}_{1 \le j \le n}, T_n\}$, *we perform the multiresolution* LU *factorization* $T_0 = \hat{T}_0 \cdot \tilde{T}_0$, *where* $\hat{T}_0 = \{\{\hat{A}_j, \hat{C}_j\}_{1 \le j \le n}, \hat{T}_n\}$, *and* $\tilde{T}_0 = \{\{\tilde{A}_j, \tilde{B}_j,\}_{1 \le j \le n}, \tilde{T}_n\}$ *are lower and upper* NS-*forms, using the following steps:*

1. *Initialization: set* $\bar{A}_1 = \bar{B}_1 = \bar{C}_1 = \bar{T}_1 = 0$.
2. *For* $j = 1, 2, \ldots, n$ *compute*
   (a) $\bar{A}_j, \bar{B}_j, \bar{C}_j$, *and* $\bar{T}_j$ $(j \neq 1)$ *via* (4.4),
   (b) $\hat{A}_j$ *and* $\tilde{A}_j$ *via* (4.5a),
   (c) $\tilde{B}_j$ *via* (4.5b),
   (d) $\hat{C}_j$ *via* (4.5c).
3. *At the last scale compute* $\hat{T}_n$ *and* $\tilde{T}_n$ *via* (4.6).

We consider the computational cost of Algorithm 4.1 in Section 5.5.

Remark 4.1. Although we have just described LU factorization, we may replace it with Choleski factorization provided that the matrix is symmetric positive definite. In fact, in some of the examples of Section 9 we used multiresolution Choleski factorization of NS-forms.

4.3. Direct Construction of Factored NS-Forms

A variant of the algorithm described in Section 4.2 may be used to construct the multiresolution LU factorization of the usual matrix representation in $\mathbf{V}_0$. Instead of constructing the NS-form of T_0 first and then computing its LU factorization, it is possible to combine decomposition and factorization.

First we recall that blocks of the NS-form are computed for $j = 1, \ldots, n$ via (2.9),

$$T_{j-1} = A_j + B_j + C_j + T_j. \tag{4.10}$$

This recursive procedure may easily be incorporated into multiresolution LU factorization by combining projection operations at each scale. To illustrate, let us again assume that computations are complete for scales $1 \le j \le k - 1$ and show the necessary steps for scale k.

We note that the projections of matrix T_{k-1} from scale $k - 1$, as in (4.10), and projections prescribed by the multiresolution LU factorization algorithm (Eq. (4.4)), may be combined as

$$\begin{aligned} T_{k-1} - (\hat{C}_{k-1}\tilde{B}_{k-1} + \bar{T}_{k-1}) \\ = (A_k - \bar{A}_k) + (B_k - \bar{B}_k) + (C_k - \bar{C}_k) + (T_k - \bar{T}_k), \end{aligned} \tag{4.11}$$

which is the projection equation for the direct construction of factored NS-forms.

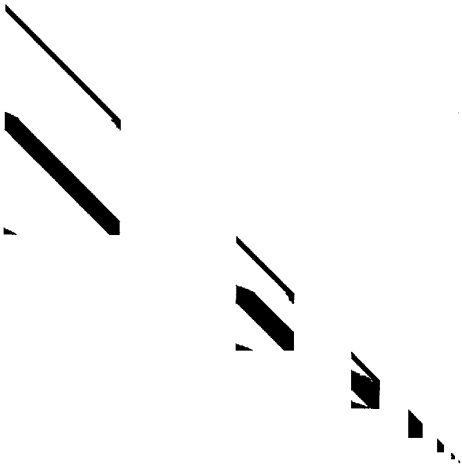

FIG. 4. The lower NS-form for Example 1 of Section 9. All entries whose absolute values are larger than 10^{-7} are shown in black.

To proceed, we compute the LU factorization at scale k. We note that the governing equations for this procedure are the same as those in Section (4.2), and thus, factorization proceeds as in (4.7) and (4.8). We have:

ALGORITHM 4.2 (Direct Construction of Factored NS-Forms). *Given an operator* $T_0: \mathbf{V}_0 \to \mathbf{V}_0$, *we perform the simultaneous wavelet decomposition and multiresolution* LU *factorization* $T_0 = \hat{T}_0 \cdot \tilde{T}_0$, *where* $\hat{T}_0 = \{\{\hat{A}_j, \hat{C}_j\}_{1 \le j \le n}, \hat{T}_n\}$, *and* $\tilde{T}_0 = \{\{\tilde{A}_j, \tilde{B}_j,\}_{1 \le j \le n}, \tilde{T}_n\}$ *are lower and upper* NS-*forms, using the following steps:*

1. *Initialization: set* $\bar{A}_1 = \bar{B}_1 = \bar{C}_1 = \bar{T}_1 = \hat{C}_0 = \tilde{B}_0 = 0$.
2. *For* $j = 1, 2, \ldots, n$ *compute*
   (a) $(A_j - \bar{A}_j)$, $(B_j - \bar{B}_j)$, $(C_j - \bar{C}_j)$, *and* $(T_j - \bar{T}_j)$ *using* (4.11),
   (b) $\hat{A}_j$ *and* $\tilde{A}_j$ *via* (4.5a),
   (c) $\tilde{B}_j$ *via* (4.5b),
   (d) $\hat{C}_j$ *via* (4.5c).
3. *At the last scale compute* $\hat{T}_n$ *and* $\tilde{T}_n$ *via* (4.6).

Remark 4.2. In Section 5 we demonstrate that the NS-forms of $\hat{T}_0$ and $\tilde{T}_0$ are sparse for a wide class of operators. Under such conditions, the fast NS-form decomposition algorithm described in [1] may be combined with sparse LU factorization. The result is a fast algorithm for computing $\hat{T}_0$ and $\tilde{T}_0$ (see Algorithm 5.3). To illustrate that such operators may be sparse, we display in Fig. 4 the lower NS-form of Example 1 in Section 9. The original matrix in this example is dense.

4.4. An Alternative Algorithm and the Reduced Operator

In Sections 4.2 and 4.3 we computed the terms $\hat{A}_j$, and $\tilde{A}_j$ using LU factorization, whereas we computed $\tilde{B}_j$ and $\hat{C}_j$ using forward and backward substitution. All of these blocks may be computed in one step, using LU factorization. The approach also allows us to introduce the notion of the reduced operator.

Let us organize the blocks $(A_j - \bar{A}_j)$, $(B_j - \bar{B}_j)$, $(C_j - \bar{C}_j)$, and $(T_j - \bar{T}_j)$, as in (2.12)), and compute a partial LU factorization,

$$\begin{bmatrix} \hat{A}_j & 0 \\ \hat{C}_j & I \end{bmatrix} \begin{bmatrix} \tilde{A}_j & \tilde{B}_j \\ 0 & R_j \end{bmatrix} = \begin{bmatrix} (A_j - \bar{A}_j) & (B_j - \bar{B}_j) \\ (C_j - \bar{C}_j) & (T_j - \bar{T}_j) \end{bmatrix}. \tag{4.12}$$

We halt factorization after eliminating the first $N_j/2$ columns, where $N_j/2$ is the dimension of A_j, and observe that (4.12) is equivalent to (4.5) (by direct examination). We note that an additional term, R_j, satisfies

$$R_j = T_j - (\bar{T}_j + \hat{C}_j \tilde{B}_j) \tag{4.13}$$

and already contains the matrices in (4.11) to be projected to the next scale. Thus, equation (4.11) is satisfied by computing projections of R_j. The operator R_j is called the reduced operator at scale j [12]. Using (4.5), R_j may also be written as the Schur's complement

$$R_j = (T_j - \bar{T}_j) - C_j A_j^{-1} B_j. \tag{4.14}$$

In this context, multiresolution LU factorization may be viewed as a recursive algorithm where the Schur's complement is computed at each scale j and then projected to scale $j + 1$.

5. COMPRESSION OF OPERATORS AND FAST ALGORITHMS

In this section we demonstrate that the algorithms described in this paper require $O(N)$ operations for a wide class of operators. We show that for strictly elliptic operators all steps of the algorithms employ sparse (banded) matrices. However, the actual class of operators for which these algorithms are fast is somewhat wider.

As was shown in [1], the NS-forms of a wide class of operators are sparse in wavelet bases. In Section 5.2 we demonstrate that for a class of operators associated with strictly elliptic problems, the lower and upper NS-forms introduced in Section 4 are compressible. Thus, the sparsity is maintained during factorization. Although the usual LU factorization is an $O(N^3)$ procedure, we show in Section 5.5 that multiresolution LU factorization requires only $O(N)$ operations if operators satisfy the conditions in Sections 5.1 and 5.2. We also show in Section 6 that multiresolution forward and backward substitutions require $O(N)$ operations. Thus, we obtain an $O(N)$ procedure for solving linear systems of equations.

In Section 7 we demonstrate that when matrices satisfy the conditions in Sections 5.1 and 5.2, matrix equations may also be solved in $O(N)$ steps. In particular, the inverse of an operator may be obtained in $O(N)$ steps.

The complexity estimates $O(N)$ imply that the operation count is $c \cdot N$, where c is a constant. Typically, c is proportional to the bandwidth w or w^2 of the matrices involved. The bandwidth, in turn, is typically proportional to the desired number of accurate digits, $-\log \epsilon$. In some applications (notably in numerical PDEs) the size

of the matrix and the selection of accuracy are connected and, thus, c cannot be considered a constant. We prefer not to mix the choices of size and accuracy in our estimates so as to encompass a wider range of applications, and this remark should be sufficient to avoid any confusion.

Finally, we note that in this paper we provide only an outline of the proofs and refer to a companion paper [13] for additional details.

5.1. Compression of Operators

The compression of operators, or, in other words, the construction of their sparse representations in orthonormal bases, has been proposed in [1]. The standard and non-standard forms of operators described in [1] may be viewed as compression schemes for a wide class of operators frequently encountered in analysis and applications, namely, Calderón–Zygmund and pseudo-differential operators.

Let us briefly state the results of [1]. Although in what follows we consider matrix representations of these operators, we note that any appropriate discretization procedure may be used, such as the Nyström method or the method of moments. In such cases the wavelet transform is simply a linear algebra tool.

Given an NS-form T_0, the operators A_j, B_j, C_j, T_j are represented by the matrices α^j, β^j, γ^j, s^j, where

$$\alpha^j_{k,k'} = \int\int K(x, y)\psi_{j,k}(x)\psi_{j,k'}(y)\,dxdy, \tag{5.1a}$$

$$\beta^j_{k,k'} = \int\int K(x, y)\psi_{j,k}(x)\phi_{j,k'}(y)\,dxdy, \tag{5.1b}$$

$$\gamma^j_{k,k'} = \int\int K(x, y)\phi_{j,k}(x)\psi_{j,k'}(y)\,dxdy, \tag{5.1c}$$

$$s^j_{k,k'} = \int\int K(x, y)\phi_{j,k}(x)\phi_{j,k'}(y)\,dxdy, \tag{5.1d}$$

for $j = 1, 2, \ldots, n$. The function $\phi(x)$ is the scaling function and its translates and dilates $\{\phi_{j,k}(x) = 2^{-j/2}\phi(2^{-j}x - k)\}_{k\in\mathbf{Z}}$ form an orthonormal basis of $\mathbf{V}_j$. The function $\psi(x)$ is the wavelet and $\{\psi_{j,k}(x) = 2^{-j/2}\psi(2^{-j}x - k)\}_{k\in\mathbf{Z}}$ forms an orthonormal basis of $\mathbf{W}_j$. We will require that the function $\psi(x)$ have M vanishing moments,

$$\int_{-\infty}^{\infty} \psi(x)x^m dx = 0, \tag{5.2}$$

for $m = 0, \ldots, M - 1$.

We label the coefficients $\alpha^j_{k,k'}$, $\beta^j_{k,k'}$, and $\gamma^j_{k,k'}$ in (5.1) by the intervals $I = I^j_k$ and $I' = I^j_{k'}$ denoting the supports of the basis functions. If the kernel $K = K(x, y)$ is smooth on the square $I \times I'$, then we have the estimate

$$|\alpha_{II'}| + |\beta_{II'}| + |\gamma_{II'}| \le C|I|^{M+1} \sup_{(x,y)\in I\times I'} (|\partial_x^M K| + |\partial_y^M K|). \tag{5.3}$$

The right-hand side of (5.3) is small whenever either $|I|$ or the derivatives involved are small. The estimate in (5.3) allows us to obtain sparse representations of integral operators by discarding the coefficients that are smaller than a chosen threshold.

Let us assume that the kernel K satisfies the conditions

$$|K(x, y)| \leq C_0 |x - y|^{-1}, \tag{5.4}$$

$$|\partial_x^M K(x, y)| + |\partial_y^M K(x, y)| \leq C_1 |x - y|^{-M-1}, \tag{5.5}$$

and, in addition, assume that the kernel K defines a bounded operator on L^2 or satisfies a substantially weaker condition (the so-called ''weak cancellation condition''),

$$\left| \int_{I \times I} K(x, y) dx dy \right| \leq C|I|, \tag{5.6}$$

for all dyadic intervals I. Under these assumptions we have

THEOREM 5.1. *If the wavelet basis has M vanishing moments, then for any kernel K satisfying the conditions* (5.4), (5.5), *and* (5.6) *the matrices α^j, β^j, γ^j satisfy the estimate*

$$|\alpha_{k,l}^j| + |\beta_{k,l}^j| + |\gamma_{k,l}^j| \leq C_M (1 + |k - l|)^{-M-1}, \tag{5.7}$$

for all integer k, l.

In particular these considerations apply to pseudo-differential operators. Let T be a pseudo-differential operator with symbol $\sigma(x, \xi)$ defined by the formula

$$T(f)(x) = \int e^{ix\xi} \sigma(x, \xi) \hat{f}(\xi) d\xi = \int K(x, y) f(y) dy, \tag{5.8}$$

where K is the distributional kernel of T.

THEOREM 5.2. *If the wavelet basis has M vanishing moments, then for any pseudo-differential operator with symbol σ of T and σ^* of T^* satisfying the standard conditions*

$$|\partial_\xi^\alpha \partial_x^\beta \sigma(x, \xi)| \leq C_{\alpha,\beta} (1 + |\xi|)^{\lambda - \alpha + \beta} \tag{5.9}$$

$$|\partial_\xi^\alpha \partial_x^\beta \sigma^*(x, \xi)| \leq C_{\alpha,\beta} (1 + |\xi|)^{\lambda - \alpha + \beta}, \tag{5.10}$$

the matrices α^j, β^j, γ^j of the non-standard form satisfy the estimate

$$|\alpha_{i,l}^j| + |\beta_{i,l}^j| + |\gamma_{i,l}^j| \leq 2^{\lambda j} C_M (1 + |i - l|)^{-M-1}, \tag{5.11}$$

for all integers i, l.

If we approximate the operator T_0^N by the operator $T_0^{N,B}$ obtained from T_0^N by setting to zero all coefficients of matrices $\alpha_{i,l}^j$, $\beta_{i,l}^j$, and $\gamma_{i,l}^j$ outside of bands of width $B \geq 2M$ around their diagonals, then it is easy to see that

FIG. 5. A matrix representing the NS-form of the matrix of Example 1. All entries whose absolute values are larger than 10^{-7} are shown in black.

$$\|T_0^{N,B} - T_0^N\| \leq \frac{C}{B^M} \log_2 N, \tag{5.12}$$

where C is a constant determined by the kernel K. In most numerical applications, the accuracy ϵ of calculations is fixed, and the parameters of the algorithm (in our case, the bandwidth B and order M) have to be chosen in such a manner that the desired precision of calculations is achieved. If M is fixed, then B has to be such that $\|T_0^{N,B} - T_0^N\| \leq C/B^M \log_2 N \leq \epsilon$, or, equivalently, $B \geq (C/\epsilon \log_2 N)^{1/M}$.

The estimate (5.12) is sufficient for practical purposes. It is possible, however, to obtain

$$\|T_0^{N,B} - T_0^N\| \leq \frac{C}{B^M} \tag{5.13}$$

instead of (5.12) (see [1]).

Finally we note that strictly elliptic operators and their Green's functions are compressible in the wavelet bases and the decay of the elements of the blocks of the nonstandard forms away from the diagonal may be controlled by choosing appropriate number of vanishing moments of the wavelet.

As an illustration, we display in Fig. 5 the NS-form of the matrix in Example 1 of Section 9. Using periodized wavelets with 6 vanishing moments and setting to zero all entries whose absolute values are smaller than 10^{-7}, we display the remaining non-zero elements in black in Fig. 5. This matrix is dense in the usual representation.

Remark 5.1. Truncating all elements of the matrix below a certain threshold (related to the desired accuracy) involves the matrix norm. We have

$$\frac{\|\delta x\|}{\|x + \delta x\|} \leq \|T^{-1}\| \, \|T\| \, \frac{\|\delta T\|}{\|T\|}. \tag{5.14}$$

The condition number $k = \|T^{-1}\| \, \|T\|$ may be viewed as an amplification factor for the relative error. Thus, (5.14) implies that if thresholding has been performed using $\|\delta T\| \leq \epsilon \|T\|$, then the relative error of the solution will not exceed $k\epsilon$.

Let us denote by $\hat{T}_\epsilon$ and $\tilde{T}_\epsilon$ approximations to $\hat{T}$ and $\tilde{T}$ obtained by setting all entries that are less than ϵ to zero, and assuming (without a loss of generality) $\|\hat{T}\| = \|\tilde{T}\| = 1$. We obtain $\|\hat{T} - \hat{T}_\epsilon\| \leq \epsilon$, $\|\tilde{T} - \tilde{T}_\epsilon\| \leq \epsilon$, and therefore, $\|\hat{T}\tilde{T} - (\hat{T}_\epsilon \tilde{T}_\epsilon)_\epsilon\| \leq \epsilon + \epsilon(1 + \epsilon) + \epsilon(1 + \epsilon)^2$. The right side is dominated by 3ϵ. Thus, for truncating the factors $\hat{T}$ and $\tilde{T}$, we use a threshold which is one-third of the threshold used for T.

5.2. Compression of Lower and Upper NS-Forms

In Section 4.4 we introduced the recursively defined reduced operator

$$R_j = T_j - (\bar{T}_j + \hat{C}_j \tilde{B}_j). \tag{5.15}$$

Let us denote by A_{R_j}, B_{R_j}, and C_{R_j} the blocks obtained from the projections of R_j. These blocks have the same rate of decay as the blocks A_j, B_j, and C_j of the NS-form as shown in the following theorem from [13].

THEOREM 5.3 (Preservation of Structure over Finitely Many Scales). *Let us assume that the operator T and the wavelet basis satisfy conditions of Theorem* 5.1. *In addition, we assume that T is a self-adjoint, strictly elliptic operator.*

Let R_j be the reduced operator on some scale j, where reduction started on subspace $\mathbf{V}_0$ and $1 \leq j \leq n$, and let A_{R_j}, B_{R_j}, and C_{R_j} be its blocks. Then the bi-infinite matrices $\alpha^{r,j}$, $\beta^{r,j}$, and $\gamma^{r,j}$ representing these blocks satisfy

$$|\alpha^{r,j}_{k,l}| + |\beta^{r,j}_{k,l}| + |\gamma^{r,j}_{k,l}| \leq C_M(1 + |k - l|)^{-M-1}, \tag{5.16}$$

for all integers k, l.

In order to prove Theorem 5.3 [13], it is necessary to consider bi-infinite matrices $\{m_{k,l}\}_{k,l \in \mathbf{Z}}$ such that

$$|m_{k,l}| < C(1 + |k - l|)^{-r}, \tag{5.17}$$

where $r > 1$ is a parameter. We note that matrices α^j, β^j, γ^j of the NS-form satisfy this estimate (see Theorems 5.1 and 5.2) where $r = M + 1$. Considering the algebra of invertible matrices $\{m_{k,l}\}_{k,l \in \mathbf{Z}}$, the following theorem (an enhancement of the result presented in [14] following [15]) is used to prove Theorem 5.3.

THEOREM 5.4. *If the matrix $\{m_{k,l}\}_{k,l \in \mathbf{Z}}$ is invertible on l^2, then*

$$|m^{-1}_{k,l}| < C'(1 + |k - l|)^{-r}. \tag{5.18}$$

The proof uses relations between commutators of unbounded operator X on l^2 defined by $X(y_k) = \{k y_k\}$ and operators $M = \{m_{k,l}\}_{k,l \in \mathbf{Z}}$ and $M^{-1} = \{m^{-1}_{k,l}\}_{k,l \in \mathbf{Z}}$. The proof is quite elaborate and we refer to [14] for the details.

From Theorem 5.3 it follows that entries of the blocks A_{R_j}, B_{R_j}, and C_{R_j} have the same rate of decay as blocks of the original NS-form. Similar to Theorem 5.1, Theorem 5.3 does not give sharp estimates for the constants. We provide numerical examples in Section 9 to show that the constant in the decay estimate in Theorem 5.3 is not significantly different from that in Theorem 5.1 since the sparsity (after applying accuracy cutoff) of the multiresolution LU factors is almost the same as that of the original NS-form.

In order to prove that all multiresolution LU factors are sparse, we note that both A_{R_j} and $A_{R_j}^{-1}$ have a fast rate of decay away from the diagonal as stated in theorems of this section. Let us now select an ϵ, the desired accuracy, and truncate both A_{R_j} and $A_{R_j}^{-1}$ independently so that the error (in the operator norm) is ϵ. Using results in [16] (i.e., Proposition 2.1 therein), we observe that the banded bi-infinite matrix A^{-1} has banded Choleski factors (or LU factors). Since computing $\tilde{B}_{R_j}$ and $\hat{C}_{R_j}$ involves the product of two banded matrices, we conclude that all blocks of NS-forms in the multiresolution LU factorization are banded for a given accuracy ϵ.

We note that in actual computations truncation is performed by restricting computations to a band (which is selected to accommodate all the entries up to a certain size) and that matrices are finite rather than bi-infinite.

Combining previous results, we obtain

THEOREM 5.5. *Let us assume that the operator T and the wavelet basis satisfy conditions of Theorems* 5.1 *and* 5.3. *Let T_0 be the projection of T on the subspace $\mathbf{V}_0$. For such operators the* NS*-form of T_0, $\hat{T}_0$, the lower, and $\tilde{T}_0$, the upper* NS*-forms ($T_0 = \hat{T}_0 \cdot \tilde{T}_0$) have banded blocks for any accuracy ϵ.*

Although we demonstrate the sparsity of the lower and upper NS-forms only for representations with bi-infinite blocks, we observe an excellent confirmation of such behavior in finite-dimensional numerical experiments in Section 9.

5.3. Fast Decomposition of NS-Forms

As was observed in [1], an approximation (to arbitrary precision) of the NS-form of a dense matrix may be constructed in $O(N)$ operations provided that the location of discontinuities of the kernel is known beforehand. A typical case is where the blocks of NS-forms are banded and have $O(N)$ non-zero entries. Following the derivation presented in [1], we present a fast algorithm for construction of a banded approximation to the NS-form.

We recall that operators $\{A_j, B_j, C_j, T_j\}$ are constructed by recursively computing

$$T_{j-1} = A_j + B_j + C_j + T_j, \tag{5.19}$$

using one step of the discrete wavelet transform (see Remark 2.1). To construct the fast algorithm, we limit use of the wavelet transform to computing entries inside the bands.

We note that blocks T_j do not have rapid decay away from the diagonal and may not be truncated accurately. However, in regions outside these bands the kernel is smooth (well represented locally by polynomials of degree less or equal to M, where

M is the number of vanishing moments of the wavelet basis). Thus, coefficients of the scaling function at coarser scales may be computed without the wavelet transform using a quadrature formula. The simplest (one-point) quadrature formula is obtained if we require $\phi(x)$ to have $M-1$ (shifted) vanishing moments. Using such wavelets, the coefficients $s_{i,l}$ at any scale may be approximated by

$$s^j_{i,l} \approx 2^j s^0_{2^j(i-1+\tau)-\tau+1,2^j(l-1+\tau)-\tau+1} \tag{5.20}$$

where s^0 is the original matrix and τ is the shift parameter,

$$\tau = \int_{-\infty}^{\infty} \phi(x)x dx. \tag{5.21}$$

The shift parameter τ may be chosen to be an integer. We note, however, that for any wavelet basis a quadrature formula (with M terms) may be constructed and used instead of (5.20) (see [1]).

We describe now how to use quadrature formulas in constructing the banded NS-form. Let us begin by filling the matrix for T_0 within a band of width $2w$. We then compute the banded approximate operators A_1, B_1, C_1, T_1 using the wavelet transform. We note that A_1, B_1, C_1, T_1 will have a bandwidth w, since the procedure of wavelet decomposition down-samples the result by a factor of 2.

To proceed to the next scale, we use the quadrature formula (5.20) to extend the bandwidth of T_1 to $2w$. We then compute the banded approximate operators $\{A_2, B_2, C_2, T_2\}$ from T_1 using the wavelet transform. These operators will have a bandwidth of w. The process is repeated at each scale, $j = 1, 2, \ldots, n$.

Remark 5.2. Let us provide an additional explanation on the use of the quadrature formula in constructing approximations to T_j, $0 \le j \le n$ within a band of width w. On the first scale, the matrix for T_0 is constructed explicitly within a band of $2w$. Consequently, T_1 can be constructed explicitly only within a band of w (due to subsampling) and in general, the operator T_j can be constructed explicitly only within the band of width $w/2^{j-1}$. Obviously, we have to fill bands to the width w on all scales.

In the algorithm, the bandwidth of operators T_j is extended to $2w$ on each scale using the approximate quadrature formula. In doing so, we effectively interpolate the matrix for T_0 as follows: for T_1, we compute interpolated entries of T_0 within the band $[2w, 4w]$. For T_2, we compute interpolated entries of T_0 within the band $[4w, 8w]$, and so forth on each scale. However, we reduce the sampling rate by a factor of two on each scale (which is consistent with the assumptions regarding the smoothness of T_0). Thus, the amount of work to fill in a band of width $2w$ remains constant on each scale.

Therefore, although blocks T_j are ''truncated'' to a bandwidth of $2w$, the operator T_0 is sampled well beyond this bandwidth.

ALGORITHM 5.1. (Fast Construction of NS-Form). *Given an operator* T_0*:* $\mathbf{V}_0 \to \mathbf{V}_0$*, a banded approximation to its* NS-*form* (*to arbitrary precision*) *may be computed in a fast manner using the following steps:*

1. *Compute entries of T_0 within a band of width $2w$.*
2. *For $j = 1, 2, \ldots, n$,*
   (a) *compute A_j, B_j, C_j, and T_j via* (5.19),
   (b) *extend the band of T_j to $2w$ via* (5.20).

Computational cost. Step 1 is computed in $O(N)$ steps. The cost of Step 2a is $O(wN_j)$, where N_j is the size of T_{j-1}. The cost of Step 2b is $O(wN_j)$. The total computational cost is $O((-\log \epsilon)N)$ since the bandwidth w is proportional to $-\log \epsilon$.

5.4. Fast Reconstruction of NS-Forms

Let us now show how to *reconstruct* a banded version of the operators T_j from an operator in the NS-form. That is, given the operators $\{\{A_j, B_j, C_j\}_{1\le j\le n}, T_n\}$, we wish to compute the operators $\{T_j\}_{1\le j\le n-1}$. We note that if T_0 satisfies the conditions of Section 5.1, then the blocks $\{\{A_j, B_j, C_j\}_{1\le j\le n}, T_n\}$ are banded, with bandwidth w. The operators $\{T_j\}_{1\le j\le n-1}$, however, are represented using a bandwidth $2w$ (see above).

At each scale j, the operator T_{j-1} may be reconstructed from the blocks A_j, B_j, C_j, and T_j via the inverse wavelet transform (see Remark 2.1). Let us demonstrate the process of reconstructing operators T_j beginning at the final scale, $j = n$.

We compute T_{n-1} from operators A_n, B_n, C_n, T_n using the inverse transform. We note that the bandwidth of T_{n-1} will be $2w$. To proceed, we first truncate T_{n-1} to a bandwidth of w. Next, we reconstruct T_{n-2} from the operators A_{n-1}, B_{n-1}, C_{n-1}, T_{n-1} using the inverse transform. We note that T_{n-2} will have a bandwidth of $2w$. The process is repeated at each scale, where in general T_{j-1} is computed by first truncating T_j to a bandwidth of w, and then computing the inverse transform.

ALGORITHM 5.2. (Fast Reconstruction of NS-Forms). *Given the banded* NS-*form $T_0 = \{\{A_j, B_j, C_j\}_{1\le j\le n} T_n\}$ with width w, the banded operators $\{T_j\}_{1\le j\le n-1}$ may be computed using the following steps:*

1. *For $j = n, n-1, \ldots, 1$,*
   (a) *truncate T_j to a bandwidth w,*
   (b) *compute T_{j-1} via* (2.15).

Computational cost. The cost of Step 1b is $O(wN_{j-1})$, where N_{j-1} is the size of T_{j-1}. The total computational cost is $O((-\log \epsilon)N)$ since the bandwidth w is proportional to $-\log \epsilon$.

5.5. Sparse Multiresolution LU Factorization

Let us consider the number of operations necessary for computing multiresolution LU factorization for sparse NS-forms. We assume that T_0 and its factors $\hat{T}_0$, $\tilde{T}_0$ are compressible. Under these conditions, we show that multiresolution LU factorization may be performed in $O(N)$ steps. We note that sparse data structures are required to achieve this result. To represent an operator in a sparse format, we truncate matrix entries below a threshold, related to the desired accuracy.

5.5.1. Sparse Factorization

We begin by computing operators $\hat{A}_j$, and $\tilde{A}_j$, as described in Section 4. Using standard LU factorization, we compute

$$A_j - \bar{A}_j = L_j U_j, \tag{5.22}$$

and set $\hat{A}_j = L_j$ and $\tilde{A}_j = U_j$. If the matrix representing $A_j - \bar{A}_j$ is sparse, then the factors in (5.22) are computed using sparse LU factorization, where computations are restricted to non-zero entries. For banded matrices, the computational cost is reduced from $O(N_j^3)$ to $O(w^2 N_j)$ where N_j is the dimension of $A_j - \bar{A}_j$ and w is its half-bandwidth.

According to Theorem 5.5, operators $\tilde{B}_j$ and $\hat{C}_j$ are compressible and may be represented by banded matrices. Thus, these operators may be obtained efficiently using sparse forward and backward substitutions. We note that fill-in may naturally occur during this process, but the banded structure is maintained by truncating values below the threshold. For this reason, the entire procedure may be viewed as an incomplete factorization scheme. We note, however, that unlike standard incomplete factorization schemes, which are generally used within an iterative method, multiresolution LU factorization may be used to solve the problem directly (see Section 6).

The computational cost of this procedure is $O(ww' N_j)$, where N_j is the dimension of matrices, w is the bandwidth of $B_j - \bar{B}_j$ and $C_j - \bar{C}_j$, and w' is the bandwidth of $\tilde{B}_j$ and $\hat{C}_j$.

5.5.2. Sparse Projections

In this section we consider the projections used during multiresolution LU factorization. First, we compute the product of $\hat{C}_{j-1}$ and $\tilde{B}_{j-1}$, each with bandwidth w. The product $\hat{C}_{j-1}\tilde{B}_{j-1}$ is obtained using sparse matrix multiplication and has bandwidth $2w$. The cost is $O(w^2 N_j)$. Next, we compute the projections $\bar{A}_j$, $\bar{B}_j$, $\bar{C}_j$, $\bar{T}_j$, which are also banded, with width w. These are computed using the fast methods developed in Section 5.3.

We now describe an algorithm, which combines the fast projection methods of Section 5.3, with sparse LU factorization, to obtain an $O(N)$ algorithm for the direct construction of LU factors.

ALGORITHM 5.3. (Fast Construction of Factored NS-Forms). *Given an operator* T_0*:* $\mathbf{V}_0 \to \mathbf{V}_0$*, we perform the fast wavelet decomposition and multiresolution* LU *factorization* $T_0 = \hat{T}_0 \cdot \tilde{T}_0$*, where* $\hat{T}_0 = \{\{\hat{A}_j, \hat{C}_j\}_{1 \le j \le n}, \hat{T}_n\}$ *and* $\tilde{T}_0 = \{\{\tilde{A}_j, \tilde{B}_j,\}_{1 \le j \le n}, \tilde{T}_n\}$ *are banded lower and upper* NS-*forms, using the following steps:*

1. *Initialization: set* $\bar{A}_1 = \bar{B}_1 = \bar{C}_1 = \bar{T}_1 = \hat{C}_0 = \tilde{B}_0 = 0$.
2. *Compute entries of* T_0 *within a band of width* $2w$.
3. *For* $j = 1, 2, \ldots, n$,

   (a) *compute* $(A_j - \bar{A}_j)$, $(B_j - \bar{B}_j)$, $(C_j - \bar{C}_j)$, *and* $(T_j - \bar{T}_j)$, *as described in Section* 5.3,

   (b) *extend the band of* $(T_j - \bar{T}_j)$ *to* $2w$ *via* (5.20),

   (c) *compute* $\hat{A}_j$ *and* $\tilde{A}_j$ *via* (4.5a) *using sparse factorization,*

   (d) *compute* $\tilde{B}_j$, *and* $\hat{C}_j$ *via* (4.5b) *and* (4.5c) *using sparse forward and backward substitution,*

   (e) *compute* $\hat{C}_j \tilde{B}_j$ *using sparse multiplication.*
4. *At the last scale compute* $\hat{T}_n$ *and* $\tilde{T}_n$ *via* (4.6) *using* LU *factorization.*

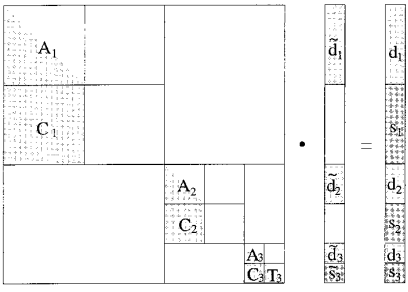

FIG. 6. Organization of multiresolution forward substitution.

Computational cost. Step 2 is computed in $O(wN)$ steps. The cost of Step 3a is $O(wN)$. Step 3b is $O(wN)$. Steps 3c and 3d are $O(w^2N)$. Step 3e is $O(w^2N)$. The total cost of this procedure is $O((-\log \epsilon)^2 N)$ since the bandwidth is proportional to $-\log \epsilon$.

6. SOLUTIONS OF LINEAR ALGEBRAIC EQUATIONS

In this section we combine the results of previous sections and describe a *direct multiresolution solver* for operators described in Section 5. We proceed along the usual lines of using LU factorization for this purpose.

Given an NS-form $T_0 = \{\{A_j, B_j, C_j\}_{1 \le j \le n}, T_n\}$, and the vector $b_0 = \{\{d_j,\}_{1 \le j \le n}, s_n\}$ in a wavelet basis, we seek to find a vector $x_0 = \{\{\tilde{d}_j,\}_{1 \le j \le n}, \tilde{s}_n\}$ which satisfies

$$T_0 \cdot x_0 = b_0, \tag{6.1}$$

where $(\cdot)$ is the multiresolution product defined in Section 3.2. We note that solutions of (6.1) are easily obtained when T_0 is a lower or upper NS-form. We refer to the algorithms in Sections 6.1 and 6.2 as *multiresolution forward and backward substitutions*, since they share common characteristics with the usual forward and backward substitutions. Combining these algorithms in Section 6.3, we obtain a direct multiresolution solver which may be used to compute solutions to (6.1).

6.1. Multiresolution Forward Substitution

First, we consider (6.1) where T_0 is the lower NS-form $T_0 = \{\{A_j, C_j\}_{1 \le j \le n}, T_n\}$. The structure of the resulting linear system is shown in Fig. 6. The governing equations are obtained by combining Eqs. (3.11a) and (3.14). Noting that $B_j = 0$ for $j = 1, 2, \ldots, n$, we have

$$A_j \tilde{d}_j = d_j - \bar{d}_j \tag{6.2}$$

on all scales $j = 1, 2, \ldots, n$, and on the last scale

$$T_n \tilde{s}_n = s_n - \bar{s}_n - C_n \tilde{d}_n. \tag{6.3}$$

The terms $\bar{d}_j$, $\bar{s}_j$ are computed via the projection equation obtained from (3.13)

$$C_{j-1} \tilde{d}_{j-1} + \bar{s}_{j-1} = \bar{d}_j + \bar{s}_j. \tag{6.4}$$

At a given scale k, Eq. (6.2) is satisfied by first computing the projection of vector $C_{k-1}\tilde{d}_{k-1} + \bar{s}_{k-1}$ on scale k to obtain $\bar{d}_k$ and $\bar{s}_k$ as in (6.4) and then solving (6.2) for d_k using standard forward substitution. On the final scale we compute $\tilde{s}_n$ by solving (6.3).

ALGORITHM 6.1 (Multiresolution Forward Substitution). *Given the lower* NS-*form* $T_0 = \{\{A_j, C_j\}_{1 \le j \le n}, T_n\}$, *and the vector* $b_0 = \{\{d_j\}_{1 \le j \le n}, s_n\}$ *in a wavelet basis, we solve the system* $T_0 \cdot x_0 = b_0$, *where* $x_0 = \{\{\tilde{d}_j\}_{1 \le j \le n}, \tilde{s}_n\}$ *is a vector in the wavelet basis, by performing multiresolution forward substitution as follows:*

1. *Initialization: set* $\bar{d}_1 = \bar{s}_1 = 0$.
2. *For* $j = 1, 2, \ldots, n$ *compute*
   (a) $\bar{d}_j$, $(j \neq 1)$, *via* (6.4),
   (b) $\tilde{d}_j$ via (6.2).
3. *At the last scale compute* $\tilde{s}_n$ *via* (6.3).

Computational cost. For operators with banded lower NS-forms, the above algorithm is $O(N)$. The projection in Step 2a requires $O(N)$ steps using the pyramid scheme, while the application of C_j to $\tilde{d}_j$ requires $O(wN)$ steps, where w is the matrix bandwidth. The cost for Steps 2b and 3 is $O(wN)$. The total cost is $O((-\log \epsilon)N)$ since w is usually proportional to $-\log \epsilon$.

Remark 6.1. It is possible to combine the wavelet decomposition of b_0 with multiresolution forward substitution, analogous to the method used in Section 4.3 for matrix factorization.

6.2. Multiresolution Backward Substitution

We now consider (6.1) where T_0 is the upper NS-form $T_0 = \{\{A_j, B_j\}_{1 \le j \le n}, T_n\}$. The structure of the resulting linear system is shown in Fig. 7. We obtain the solution using a multiresolution backward substitution algorithm.

The governing equations are obtained by combining Eqs. (3.11a) and (3.14). Noting that $C_j = 0$ for $j = 1, 2, \ldots, n$, we have

$$A_j \tilde{d}_j = d_j - B_j \tilde{s}_j \tag{6.5}$$

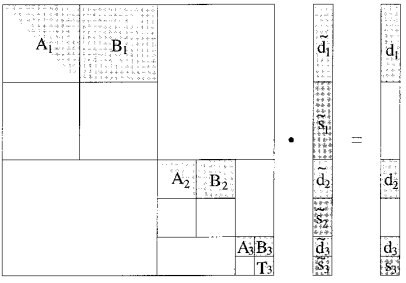

FIG. 7. Organization of multiresolution backward substitution.

on all scales $j = 1, 2, \ldots, n$, and on the last scale,

$$T_n \tilde{s}_n = s_n. \tag{6.6}$$

We note that no projections enter into Eqs. (6.5) and (6.6), since $C_j = 0$ in (3.11b) implies that $\check{s}_j = \bar{s}_j = 0$. We show, however, that projections from coarser to finer scales (i.e., reconstructions) will be required throughout the algorithm.

To demonstrate this, let us begin on the coarsest scale, $j = n$, and solve (6.6) for $\tilde{s}_n$ using the usual backward substitution. This completes the procedure for $j = n$. We now assume that Eq. (6.5) has been satisfied for $j = n, \ldots, k + 1$, and *reconstruct* $\tilde{s}_k$,

$$\tilde{s}_k = \tilde{d}_{k+1} + \tilde{s}_{k+1}, \tag{6.7}$$

using the inverse wavelet transform (see Remark 2.3). Next, we compute $\tilde{d}_{n-1}$ by solving (6.5) using the usual backward substitution. We have

ALGORITHM 6.2. (Multiresolution Backward Substitution). *Given the upper* NS-*form* $T_0 = \{\{A_j, B_j\}_{1 \le j \le n}, T_n\}$, *and the vector* $b_0 = \{\{d_j\}_{1 \le j \le n}, s_n\}$ *in a wavelet basis, we solve the system* $T_0 \cdot x_0 = b_0$, *where* $x_0 = \{\{\tilde{d}_j\}_{1 \le j \le n}, \tilde{s}_n\}$ *is a vector in the wavelet basis, by performing multiresolution backward substitution as follows:*

1. *At the last scale compute* $\tilde{s}_n$ *using* (6.6).
2. *For* $j = n, n - 1, \ldots, 1$ *compute*
   (a) $\tilde{s}_j$, $(j \neq n)$, *via* (6.7),
   (b) $\tilde{d}_j$ *via* (6.5).

Computational cost. The computational cost for this procedure is the same as for forward substitution, namely, $O((-\log \epsilon)N)$.

6.3. Multiresolution Direct Solver

We have now developed all the tools necessary to solve the linear system of algebraic equations

$$T_0 \cdot x_0 = b_0, \tag{6.8}$$

for general NS-forms $T_0 = \{\{A_j, B_j, C_j\}_{1\le j\le n}\} T_n\}$. The procedure is analogous to direct methods based on the usual LU factorization. The only difference is that the product in (6.8) refers to the multiresolution product defined in Section (3.2).

We begin by computing the multiresolution LU factorization $\hat{T}_0 \cdot \tilde{T}_0 = T_0$ as outlined in Sections 4.2, 4.3, and 4.4. We proceed to solve the system $\hat{T}_0 \cdot y_0 = b_0$ for y_0 using multiresolution forward substitution, as described in Section 6.1. Given y_0, we may obtain x_0 by solving $\tilde{T}_0 \cdot x_0 = y_0$ using multiresolution backward substitution, as described in Section 6.2.

We emphasize that for operators which satisfy the conditions of Section (5.2), the multiresolution LU decomposition requires $O((-\log \epsilon)^2 N)$ operations and multiresolution forward and backward substitutions $O((-\log \epsilon) N)$ operations.

7. SOLUTIONS OF MATRIX EQUATIONS

In this section we present a direct method for solving matrix equations. Specifically, given the NS-forms $\hat{T}_0 = \{\{\hat{A}_j, \hat{B}_j, \hat{C}_j\}_{1\le j\le n}, \hat{T}_n\}$ and $B_0 = \{\{A_j, B_j, C_j\}_{1\le j\le n}, T_n\}$, we seek to find a NS-form $\tilde{X}_0 = \{\{\tilde{A}_j, \tilde{B}_j, \tilde{C}_j\}_{1\le j\le n}, \tilde{T}_n\}$ which satisfies

$$\hat{T}_0 \cdot \tilde{X}_0 = B_0, \tag{7.1}$$

where $(\cdot)$ is the multiresolution product defined in Section 3.3.

Similar to the usual LU decomposition, we may consider the column vectors of B_0 in a wavelet basis and solve for columns of $\tilde{X}_0$. Unlike the usual LU decomposition, however, there is a significant difference between considering columns of B_0 separately and considering B_0 as a matrix; namely, the matrix representation of B_0 is sparser. For example, under the wavelet transform the identity matrix does not change, whereas unit columns of the identity matrix develop bands on all scales.

The algorithms of this section may therefore be viewed as generalizations of multiresolution forward and backward substitution for matrix equations. Such a solver may be used to compute the inverse operator T_0^{-1}.

7.1. Factorization of Blocks T_j

In what follows, we use the lower and upper triangular factors $\hat{T}_j$ and $\tilde{T}_j$, which belong to the extended lower and upper NS-forms $\hat{T}_0$ and $\tilde{T}_0$. Since these factors are not normally computed during the multiresolution LU factorization, we begin by describing a procedure for their construction. These operators may not be sparse, but algorithms of this section require only a band around the diagonal.

Combining formulas (3.21d) and (3.26), and eliminating the intermediate variable $\breve{T}_j$, we obtain

$$T_j = \hat{T}_j \tilde{T}_j + \hat{C}_j \tilde{B}_j + \bar{T}_j \tag{7.2}$$

for $j = 1, 2, \ldots, n$. We compute the factors $\hat{T}_j$ and $\tilde{T}_j$ on each scale using the usual LU factorization,

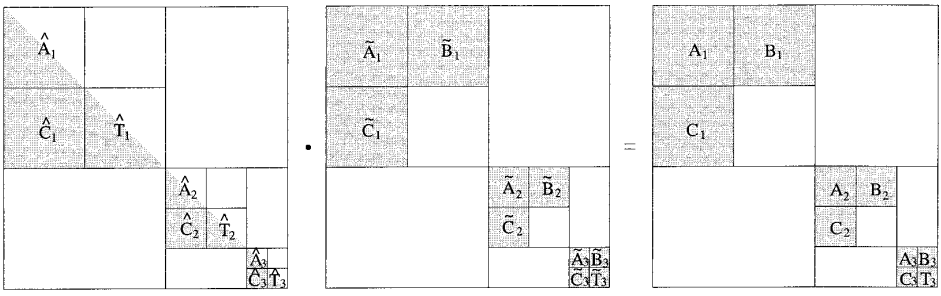

FIG. 8. Organization of multiresolution forward substitution for matrices.

$$L_j U_j = T_j - \hat{C}_j \tilde{B}_j - \overline{T}_j, \tag{7.3}$$

where $L_j = \hat{T}_j$ and $U_j = \tilde{T}_j$.

Remark 7.1. We note that this procedure requires the multiplication of $\hat{C}_j$ and $\tilde{B}_j$, and the computation of $\overline{T}_j$ via (4.4). These computations are a part of the LU factorization of T_0 (see, e.g., Step 2a of Algorithm 4.1). The results may be stored as $T_j - \hat{C}_j \tilde{B}_j - \overline{T}_j$ at each scale j during LU factorization.

ALGORITHM 7.1. (LU Factorization of Blocks T_j). *Given the extended* NS-*form* $T_0 = \{A_j, B_j, C_j, T_j\}_{1 \le j \le n}$, and its LU *factors* $\hat{T}_0 = \{\{\hat{A}_j, \hat{C}_j\}_{1 \le j \le n}, \hat{T}_n\}$ *and* $\tilde{T}_0 = \{\{\tilde{A}_j, \tilde{B}_j,\}_{1 \le j \le n}, \tilde{T}_n\}$, *the lower and upper triangular factors* $\hat{T}_j$ *and* $\tilde{T}_j$ *may be computed using the following steps:*

1. *Initialization: set* $\overline{T}_1 = 0$.
2. *For* $j = 1, 2, \ldots, n$ *compute*
   (a) $\overline{T}_j$ $(j \neq 1)$ *via* (4.4),
   (b) $\hat{T}_j$ *and* $\tilde{T}_j$ *via* (7.3).

Computational cost. We note that operations in this algorithm are performed on dense matrices. However, since we require only a banded version of the operators, we may perform computations in a fast manner. Step 2a requires the projection of a banded matrix on a coarser scale and may be computed in $O(wN)$ steps using the fast algorithm described in Section 5.3. We also require the matrix multiplication of C_j and B_j which requires $O(w^2N)$ operations, where w is the matrix bandwidth. In Step 2b we compute the LU factorization of a banded operator, which requires $O(w^2N)$ operations. The total computational cost is $O((-\log \epsilon)^2 N)$ since w is proportional to $-\log \epsilon$.

7.2. Multiresolution Forward Substitution

We now consider (7.1), where $\hat{T}_0$ is the lower NS-form $\hat{T}_0 = \{\{\hat{A}_j, \hat{C}_j\}_{1 \le j \le n}, \hat{T}_n\}$. The structure of the resulting matrix equation is illustrated in Fig. 8.

The governing equations are obtained by combining Eqs. (3.21) and (3.24), and noting that $\hat{B}_j = 0$ for $j = 1, 2, \ldots, n$. We have

$$\hat{A}_j\tilde{A}_j = A_j - \bar{A}_j, \tag{7.4a}$$

$$\hat{A}_j\tilde{B}_j = B_j - \bar{B}_j, \tag{7.4b}$$

$$\hat{T}_j\tilde{C}_j = C_j - \bar{C}_j - \hat{C}_j\tilde{A}_j, \tag{7.4c}$$

for $j = 1, 2, \ldots, n$, and at the last scale

$$\hat{T}_n\tilde{T}_n = T_n - \bar{T}_n - \hat{C}_n\tilde{B}_n. \tag{7.5}$$

The terms $\bar{A}_j$, $\bar{B}_j$, $\bar{C}_j$, and $\bar{T}_j$ are computed via the projection equation in (4.4)

$$\hat{C}_{j-1}\tilde{B}_{j-1} + \bar{T}_{j-1} = \bar{A}_j + \bar{B}_j + \bar{C}_j + \bar{T}_j. \tag{7.6}$$

At a given scale k, Eqs. (7.4) are satisfied by first computing projections of the matrix $\hat{C}_{k-1}\tilde{B}_{k-1} + \bar{T}_{k-1}$ to obtain $\bar{A}_k$, $\bar{B}_k$, $\bar{C}_k$, and $\bar{T}_k$, as in (7.6). Next we use the usual matrix operations to solve (7.4a) and (7.4b) for $\tilde{A}_k$ and $\tilde{B}_k$. Given $\tilde{A}_k$, we obtain $\tilde{C}_k$ by solving (7.4c) using the usual forward substitution. We have now satisfied Eqs. (7.4) for $j = k$. On the last scale, we compute the term $\tilde{T}_n$ by solving (7.5) using the usual forward substitution.

ALGORITHM 7.2. (Multiresolution Forward Substitution (Matrix Version)). *Given the extended lower* NS-*form* $\hat{T}_0 = \{\hat{A}_j, \hat{C}_j, \hat{T}_j\}_{1\le j\le n}$, *and the* NS-*form* $B_0 = \{\{A_j, B_j, C_j\}_{1\le j\le n}, T_n\}$, *we solve the system* $\hat{T}_0 \cdot \tilde{X}_0 = B_0$, *where* $\tilde{X}_0 = \{\{\tilde{A}_j, \tilde{B}_j, \tilde{C}_j\}_{1\le j\le n}, \tilde{T}_n\}$ *is a* NS-*form, using multiresolution forward substitution as follows:*

1. *Initialization: set* $\bar{A}_1 = \bar{B}_1 = \bar{C}_1 = \bar{T}_1 = 0$.
2. *For* $j = 1, 2, \cdots n$ *compute*
   (a) $\bar{A}_j, \bar{B}_j, \bar{C}_j$ $(j \neq 1)$ *via* (7.6),
   (b) $\tilde{A}_j$ *via* (7.4a),
   (c) $\tilde{B}_j$ *via* (7.4b),
   (d) $\tilde{C}_j$ *via* (7.4c).
3. *At the last scale compute* $\tilde{T}_n$ *via* (7.5).

Computational cost. The projection in Step 2a requires $O(wN)$ operations using the fast algorithm described in Section 5.3. The matrix multiplication of C_j with B_j may be computed in $O(w^2N)$ steps, where w is the matrix bandwidth. Steps 2b, 2c, and 2d require $O(w^2N)$ operations. The total computational cost is $O((-\log \epsilon)^2 N)$, since w is usually proportional to $-\log \epsilon$.

7.3. Multiresolution Backward Substitution

We now consider (7.1), where $\hat{T}_0$ is the upper NS-form $\hat{T}_0 = \{\{\hat{A}_j, \hat{B}_j\}_{1\le j\le n}, \hat{T}_n\}$. The structure of the resulting matrix equation is illustrated in Fig. 9. We obtain the governing equations by combining Eqs. (3.21) and (3.24) and noting that $\hat{C}_j = 0$ for $j = 1, 2, \ldots, n$. We have

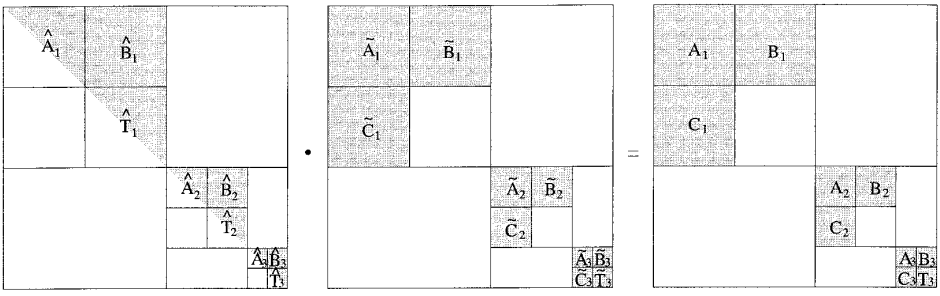

FIG. 9. Organization of multiresolution backward substitution for matrices.

$$\hat{A}_j\tilde{A}_j = A_j - \hat{B}_j\tilde{C}_j, \tag{7.7a}$$

$$\hat{A}_j\tilde{B}_j = B_j - \hat{B}_j\tilde{T}_j, \tag{7.7b}$$

$$\hat{T}_j\tilde{C}_j = C_j, \tag{7.7c}$$

for $j = 1, 2, \ldots, n$, and at the last scale

$$\hat{T}_n\tilde{T}_n = T_n. \tag{7.8}$$

We note that no projections enter into Eqs. (7.7) and (7.8), since $C_j = 0$ in (3.21c) implies that $\bar{A}_j = \bar{B}_j = \bar{C}_j = \bar{T}_j = 0$.

We start at the scale $j = n$ and compute $\tilde{T}_n$ by solving (7.8) using the usual backward substitution. Let us now assume that (7.7) has been satisfied for $j = n, \ldots, k + 1$. We first reconstruct $\tilde{T}_k$ via (2.9),

$$\tilde{T}_k = \tilde{A}_{k+1} + \tilde{B}_{k+1} + \tilde{C}_{k+1} + \tilde{T}_{k+1}, \tag{7.9}$$

using the inverse wavelet transform (see Remark 2.1). We next compute $\tilde{C}_k$ by solving (7.7c) using the usual backward substitution. Given $\tilde{T}_k$ and $\tilde{C}_k$, we may obtain $\tilde{A}_k$ and $\tilde{B}_k$ by solving (7.7a) and (7.7b) using the usual backward substitution.

ALGORITHM 7.3. (Multiresolution Backward Substitution (Matrix Version)). *Given the extended upper* NS-*form* $\hat{T}_0 = \{\hat{A}_j, \hat{B}_j, \hat{T}_j\}_{1\le j\le n}$, *and the* NS-*form* $B_0 = \{\{A_j, B_j, C_j\}_{1\le j\le n}, T_n\}$, *we solve the system* $\hat{T}_0 \cdot \tilde{X}_0 = B_0$, *where* $\tilde{X}_0 = \{\{\tilde{A}_j, \tilde{B}_j, \tilde{C}_j\}_{1\le j\le n}, \tilde{T}_n\}$ *is a* NS-*form, using multiresolution backward substitution as follows:*

1. *At the last scale compute* $\tilde{T}_n$ *using* (7.8).
2. *For* $j = n, n - 1, \ldots, 1$, *compute*
   (a) $\tilde{T}_j$ $(j \neq n)$ *via* (7.9),
   (b) $\tilde{C}_j$ *via* (7.7c),
   (c) $\tilde{B}_j$ *via* (7.7b),
   (d) $\tilde{A}_j$ *via* (7.7a).

Computational cost. The computational cost for this algorithm is the same as for forward substitution, namely, $O((-\log \epsilon)^2 N)$.

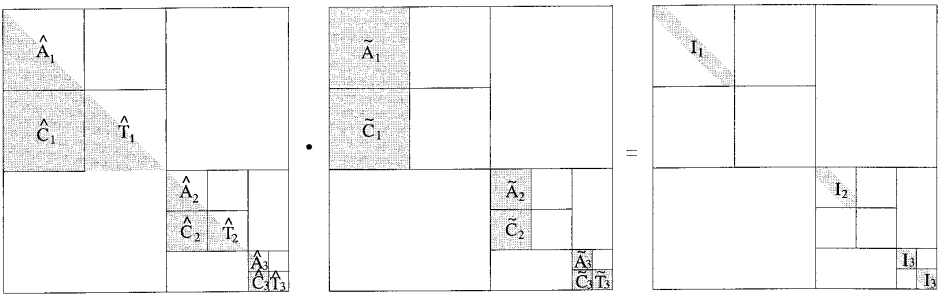

FIG. 10. Organization of multiresolution forward substitution for finding the inverse.

7.4. Multiresolution Direct Solver for Matrix Equations

We now consider the solution to matrix equations of the type in (7.1) for general NS-forms $T_0 = \{\{A_j, B_j, C_j\}_{1 \le j \le n} T_n\}$. This procedure is completely analogous to the multiresolution direct methods developed for linear systems in Section 6.3.

We begin by computing the multiresolution LU factorization $\hat{T}_0 \cdot \tilde{T}_0 = T_0$ as outlined in Sections 4.2, 4.3, and 4.4. In addition, we store the banded versions of the blocks $\hat{T}_j$ and $\tilde{T}_j$ as described in Section 7.1. We then solve the system $\hat{T}_0 \cdot Y_0 = B_0$ for Y_0 using multiresolution forward substitution for matrices, as described in Section 7.2. Given Y_0, we may obtain X_0 by solving $\tilde{T}_0 \cdot X_0 = Y_0$ using multiresolution backward substitution for matrices, as described in Section 7.3.

Each of these algorithms is $O((-\log \epsilon)^2 N)$, where operators satisfy the conditions in Section 5.2. We thus have a direct, $O((-\log \epsilon)^2 N)$ method for computing solutions to matrix equations.

7.5. Computing the Inverse Operator

Let us describe the algorithm to compute the inverse operator in greater detail. Given the NS-form $\hat{T}_0 = \{\{\hat{A}_j, \hat{B}_j, \hat{C}_j\}_{1 \le j \le n} \hat{T}_n\}$, we seek to find the inverse NS-form $T_0^{-1} = \{\{\tilde{A}_j, \tilde{B}_j, \tilde{C}_j\}_{1 \le j \le n} \tilde{T}_n\}$ which satisfies

$$T_0 \cdot T_0^{-1} = I_0, \tag{7.10}$$

where $I_0 = \{\{I_j\}_{1 \le j \le n}, I_n\}$ is the NS-form of an identity matrix. We note that the identity blocks I_j occupy the space of A_j and that the block I_n occupies the space of T_n, as may be seen in Fig. 10. Although the NS-form T_0^{-1} may be obtained using the multiresolution direct solver as described in Section 7.4, the algorithm is simpler for this special case.

7.5.1. Forward Substitution

We first consider the multiresolution forward substitution $\hat{T}_0 \cdot \tilde{Y}_0 = I_0$ to obtain $\tilde{Y}_0$. We note that $\tilde{Y}_0$ represents the inverse of $\hat{T}_0$ and that $\tilde{Y}_0$ is lower triangular. The latter is true since the inverse of a lower triangular matrix is also lower triangular. Since $\tilde{Y}_0$ is lower triangular, $\tilde{B}_j = 0$ on all scales, and $\bar{A}_j = \bar{B}_j = \bar{C}_j = \bar{T}_j = 0$ for $j = 1, 2,$

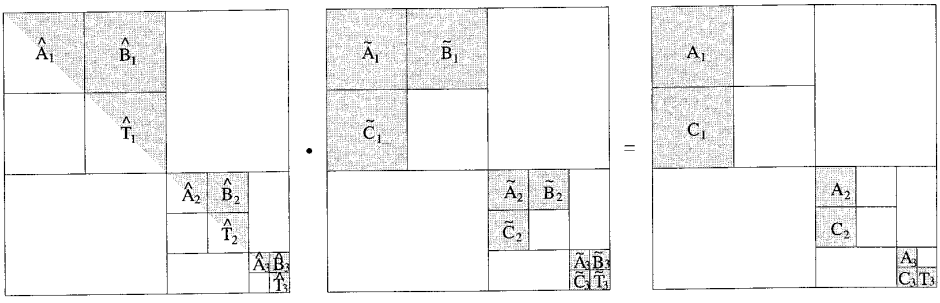

FIG. 11. Organization of multiresolution backward substitution for finding the inverse.

. . . , n, and no projections need be computed. Thus, operations on each scale may be performed independently. From (7.4) and (7.5) we obtain the simplified equations

$$\hat{A}_j\tilde{A}_j = I_j, \tag{7.11a}$$

$$\hat{T}_j\tilde{C}_j = -\hat{C}_j\tilde{A}_j, \tag{7.11b}$$

for $j = 1, 2, \ldots, n$, and at the last scale

$$\hat{T}_n\tilde{T}_n = I_n. \tag{7.12}$$

7.5.2. Backward Substitution

We next consider the multiresolution backward substitution, $\hat{T}_0 \cdot \tilde{X}_0 = Y_0$, to obtain $\tilde{X}_0$, as illustrated in Fig. 11. Using (7.7) and (7.8), we have

$$\hat{A}_j\tilde{A}_j = A_j - \hat{B}_j\tilde{C}_j, \tag{7.13a}$$

$$\hat{A}_j\tilde{B}_j = -\hat{B}_j\tilde{T}_j, \tag{7.13b}$$

$$\hat{T}_j\tilde{C}_j = C_j, \tag{7.13c}$$

for $j = 1, 2, \ldots, n$, and at the last scale

$$\hat{T}_n\tilde{T}_n = T_n. \tag{7.14}$$

8. LU DECOMPOSITION OF STANDARD FORMS

The LU factorization of matrices represented in the standard form (S-form) is the usual LU factorization in a wavelet system of coordinates. Although such representation is less efficient than the multiresolution LU factorization of NS-forms, we show that if lower and upper NS-forms $\hat{T}_0$ and $\tilde{T}_0$ are compressible in a wavelet basis, then the corresponding S-forms are also compressible.

The S-form may be obtained by applying the wavelet transform to each row and column of the matrix representing T_0. Alternatively, in [1] it was demonstrated that

S-forms may be obtained by first constructing the NS-form and then converting it to an S-form using the following algorithm:

ALGORITHM 8.1. (Computing the S-Form from the NS-Form). *Given the* NS-*form* $T_0 = \{\{A_j, B_j, C_j\}_{1 \le j \le n}, T_n\}$, *the* S-*form may be obtained at each scale* j *using the following steps:*

1. *Recursively apply the wavelet transform to each row in* B_j *for* $k = j, j + 1, \ldots, n$.
2. *Recursively apply the wavelet transform to each column in* C_j *for* $k = j, j + 1, \ldots, n$.
3. *Place the blocks* $\{A_j, B_j, C_j\}$ *in the space occupied by* T_{j-1}. (*This step returns the system to its original dimension.*)

We now use Algorithm 8.1 to construct the S-form of lower and upper NS-forms. We have

THEOREM 8.1. *Let* $T_0 = \hat{T}_0 \cdot \tilde{T}_0$ *be the multiresolution LU factorization of the NS-form for* T_0, *and let* $T_0 = LU$ *be the standard LU factorization of the S-form for* T_0. *Then* $\hat{T}_0$ *is the NS-form of* L, *and* $\tilde{T}_0$ *is the NS-form of* U. *Therefore, sparsity of lower and upper NS-forms implies sparsity of lower and upper factors of the standard form.*

Remark 8.1. We note that since S-form representations also admit a sparse structure, standard LU factorization may be used to compute the factors L and U. However, such approach is less efficient than that of using the NS-forms. The loss of efficiency may be clearly seen from Theorem 8.1. Single bands of $\hat{C}_j$ and $\tilde{B}_j$ are expanded into several bands to account for interaction between scales, thus reducing the sparsity of corresponding matrices by a significant factor.

9. NUMERICAL EXAMPLES

In this section we present numerical examples to demonstrate the performance of the algorithms of this paper. The programs were written in C and all calculations were performed on an HP 735/125 computer. As far as the raw speed is concerned, the code was not optimized. The goal of these examples is to illustrate the behavior of the algorithms as the size of matrices increases.

In Examples 1 through 4 we compute solutions to linear systems of the type $Ax = b$, where A is an $N \times N$ matrix, using the fast direct solver described in Section 6.3. In Example 5 we compute the S-form of a matrix and then perform standard LU decomposition using sparse data structures to illustrate Theorem 8.1 in Section 8. In Example 6 we compute the inverse of a matrix using a sparse version of the algorithm described in Section 7.5. In all cases, the experiments were performed for $N = 128$, 256, 512, 1024, and 2048, and in all figures the matrices are depicted for $N = 256$.

Accuracy estimates are obtained by first computing the r.h.s. b as $b = Ax$, where x is a random vector with $\|x\|_2 = 1$. We then compute the solution x' using the specified method. Finally, the L_2 and L_∞ norms of the error vector $e = x - x'$ are evaluated. In the examples the number of vanishing moments of wavelet coefficients was chosen to achieve roughly the single precision accuracy.

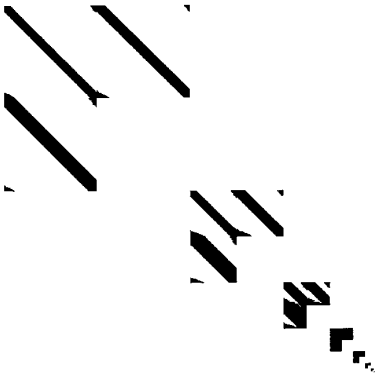

FIG. 12. The NS-form of LU factors for Example 1. The factors are combined together as $\hat{T}_0$ & $\tilde{T}_0$ and entries above the threshold 10^{-7} are shown in black.

Let us describe organization of the tables. Column 1 indicates the size of the matrix, N. Column 2 contains CPU time t_{fact} required to compute the factored NS-forms $\hat{T}_0$ and $\tilde{T}_0$ using Algorithm 4.2 of Section 4.3. The time t_{fact} includes the matrix fill, wavelet decomposition, and multiresolution LU factorization. Column 3 contains the time t_{sub} necessary to compute the solution of the linear system using sparse multiresolution forward and backward substitution of Section 6. Columns 4 and 5 contain the L_∞ and L_2 errors of the computations. Column 6 contains the compression ratio for T_0, the NS-form of the operator. The compression ratio is defined as the ratio of N^2 to N_s, where N_s is the number of significant entries in the matrix after truncation. Finally, column 7 contains the compression ratio for the lower and upper NS-forms $\hat{T}_0$ and $\tilde{T}_0$. This compression ratio is computed for the matrix obtained by placing both the lower and upper factors into the same matrix (this is how we store the factors). We denote this combination $\hat{T}_0$ & $\tilde{T}_0$ and note that this matrix contains all significant entries of the multiresolution LU factorization. We display the combination $\hat{T}_0$ & $\tilde{T}_0$ for various examples to illustrate the sparsity of the LU factors.

EXAMPLE 1. We consider the matrix

$$A_{ij} = \begin{cases} C/\tan(\pi(i-j)/N), & i \neq j \\ 1, & i = j, \end{cases} \tag{9.1}$$

where $i, j = 1, \ldots, N$ and $C = 1/N$. The constant C is chosen so that $A_{ij} \approx 1/(\pi(i-j))$ for small $|i-j|$. Multiresolution LU factorization was performed using periodized wavelets with six vanishing moments. Operations were restricted to a half-bandwidth of 20, and elements of absolute value less than 10^{-7} were truncated. In Fig. 12 we show the truncated matrix $\hat{T}_0$ & $\tilde{T}_0$. Timing and error results are given in Table 1.

We include in Table 1 a comparison of CPU times with usual direct methods. Column 4 contains the time necessary to compute the usual LU factorization of the original matrix, which requires $O(N^3)$ operations since the original matrix is dense. Column 5 contains the time used for dense forward and backward substitution, which

TABLE 1
Numerical Results for Example 1

	MultiR. LU		Dense LU		Errors		Comp. ratios	
N	t_{fact}	t_{sub}	t_{fact}	t_{sub}	L_∞	L_2	T_0	$\hat{T}_0$ & $\tilde{T}_0$
128	0.46	0.01	0.05	0.01	2.75×10^{-7}	1.31×10^{-7}	2.53	2.22
256	1.15	0.02	0.47	0.01	3.50×10^{-7}	1.35×10^{-7}	4.76	4.09
512	2.57	0.05	9.93	0.02	2.46×10^{-6}	4.43×10^{-7}	9.25	7.85
1024	5.66	0.12	97.05	0.11	3.54×10^{-6}	7.33×10^{-7}	18.22	15.41
2048	13.27	0.28	776.4[a]	0.4[a]	3.67×10^{-6}	7.45×10^{-7}	36.19	30.55

[a] Estimated values.

is $O(N^2)$. Columns 6 and 7 contain the L_∞ and L_2 errors of the sparse multiresolution computations.

EXAMPLE 2. We consider the matrix

$$A_{ij} = \begin{cases} 1, & |i-j| = 1, N-1 \\ -2, & i = j \\ 0, & \text{elsewhere} \end{cases} \tag{9.2}$$

which is a periodized version of the second derivative operator. We note that this matrix has a one-dimensional nullspace which contains a constant. This nullspace may easily be removed in the NS-form by computing the decomposition to the last scale and then eliminating the equations involving T_n. Multiresolution Choleski factorization was performed using periodized wavelets with eight vanishing moments. Operations were restricted to a half-bandwidth of 22, and elements of absolute value less than 10^{-10} were truncated. In Fig. 13 we show the truncated matrix $\hat{T}_0$ & $\tilde{T}_0$ which

FIG. 13. The NS-form of LU factors for Example 2. The factors are combined together as $\hat{T}_0$ & $\tilde{T}_0$ and entries above the threshold 10^{-10} are shown in black.

TABLE 2
Numerical Results for Example 2

	Run times		Errors		Comp. ratios	
N	t_{fact}	t_{sub}	L_∞	L_2	T_0	$\hat{T}_0$ & $\tilde{T}_0$
128	0.61	0.01	3.50×10^{-7}	3.17×10^{-7}	2.01	1.60
256	1.53	0.03	9.13×10^{-7}	9.46×10^{-7}	3.71	2.83
512	3.46	0.06	2.49×10^{-6}	2.37×10^{-6}	7.17	5.38
1024	7.30	0.14	6.52×10^{-6}	5.94×10^{-6}	14.11	10.50
2048	17.41	0.33	1.34×10^{-5}	1.11×10^{-5}	28.03	20.78

contains the lower and upper NS-forms. Results for this example are summarized in Table 2.

EXAMPLE 3. We consider the operator $\partial/\partial n \ln(1/r)$ (i.e., the normal derivative of the two-dimensional static Green's function). We discretize the operator on the boundary of an ellipse and obtain the matrix $I + A$ where

$$A_{ij} = \frac{1}{N} \frac{\cosh(u)\sinh(u)}{\cosh^2(u)\sin^2(\theta_{ij}) + \sinh^2(u)\cos^2(\theta_{ij})}, \tag{9.3}$$

and $\theta_{ij} = \pi(i + j)/N$, and u is a parameter related to the eccentricity of the ellipse. The eccentricity of an ellipse is defined as the ratio of the distance between foci to the length of the major axis. For this example, we use $u = 1.0$, which corresponds to an eccentricity of 0.65.

Multiresolution Choleski factorization was performed using periodized wavelets with six vanishing moments. Operations were restricted to a half-bandwidth of 10, and elements of absolute value less than 10^{-7} were truncated. The results for this example are shown in Table 3. Remarkably, in this case the compressiton ratios are the same for both the NS-form and the LU factors.

TABLE 3
Numerical Results for Example 3

	Run times		Errors		Comp. ratios	
N	t_{fact}	t_{sub}	L_∞	L_2	T_0	$\hat{T}_0$ & $\tilde{T}_0$
128	0.31	0.01	1.08×10^{-7}	7.14×10^{-8}	17.73	17.73
256	0.70	0.01	1.43×10^{-7}	9.21×10^{-8}	64.38	64.38
512	1.52	0.03	5.69×10^{-8}	3.36×10^{-8}	198.29	198.29
1024	3.61	0.06	4.37×10^{-8}	2.71×10^{-8}	576.14	576.14
2048	8.16	0.12	3.88×10^{-8}	2.50×10^{-8}	1474.79	1474.79

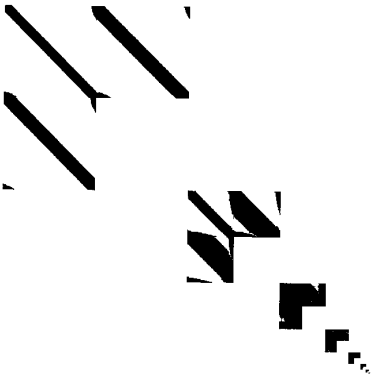

FIG. 14. The NS-form of LU factors for Example 4. The factors are combined together as $\hat{T}_0$ & $\tilde{T}_0$ and entries above the threshold 10^{-7} are shown in black.

EXAMPLE 4. We consider the matrix

$$A_{ij} = \begin{cases} 1, & |i-j| = 1 \\ -1.5, & i = j \\ 0, & \text{else} \end{cases} \tag{9.4}$$

which is similar to the second derivative operator. This matrix, however, is not positive-definite and does not satisfy the requirements of Section 5.2. We include this example to demonstrate that some improvement in speed may still be obtained. In Fig. 14 we show the lower and upper NS-forms $\hat{T}_0$ & $\tilde{T}_0$ constructed using periodized wavelets with 6 vanishing moments. All entries above a threshold of 10^{-7} are shown in black. The first two scales correspond to a positive definite submatrix, and the blocks of the LU factors are banded. Beyond scale $j = 3$, the matrix is no longer banded, but some sparsity remains. The algorithm is modified by allowing fill-ins to be generated outside the bands (the initial half-bandwidth was 15). The effect is that the bandwidth is allowed to grow and, at $j = 3$, the bandwidth equals the size of the sub-block. Since the matrix is not positive-definite, we use partial pivoting as described in the Appendix. The results are summarized in Table 4.

TABLE 4
Numerical Results for Example 4

	Run times		Errors		Comp. ratios	
N	t_{fact}	t_{sub}	L_∞	L_2	T_0	$\hat{T}_0$ & $\tilde{T}_0$
128	0.37	0.01	1.72×10^{-6}	1.49×10^{-6}	2.63	1.97
256	0.94	0.04	5.05×10^{-6}	4.91×10^{-6}	5.04	3.31
512	2.60	0.11	7.01×10^{-6}	6.33×10^{-6}	9.88	6.09
1024	8.32	0.38	7.68×10^{-6}	5.91×10^{-6}	19.62	11.35
2048	30.05	1.40	3.13×10^{-5}	2.37×10^{-5}	39.11	20.57

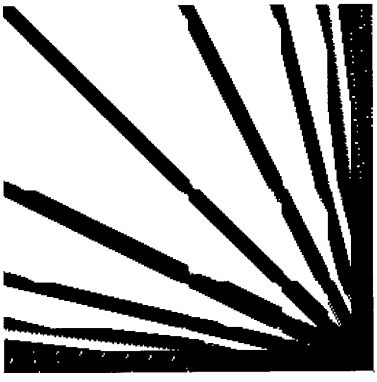

FIG. 15. The S-form of LU factors for Example 5. The factors are combined together as L & U and entries above the threshold 10^{-7} are shown in black.

EXAMPLE 5. We compute the S-form of the operator used in Example 1 and then perform standard LU factorization using sparse data structures. All elements of absolute value less than 10^{-7} were truncated. We compute the S-form directly (without computing the NS-form first) and it requires $O(N^2)$ operations. We use this example to demonstrate that if NS-forms remain sparse during multiresolution LU factorization (as in Example 1), then the corresponding S-forms will also be sparse (see Section 8 for details). In Fig. 15 we show the truncated matrix L & U which contains the lower and upper triangular matrices produced during LU factorization. From Table 5 we observe that the compression ratio for the S-form is worse than that for the NS-form in Table 1.

EXAMPLE 6. We compute the inverse of the operator used in Example 3 and leave the result in the NS-form. All operations were performed using periodized wavelets with six vanishing moments. Operations were restricted to a half-bandwidth of 10, and elements of absolute value less than 10^{-7} were truncated.

Error analysis for this example was performed by computing the solution vector x' as $x' = T_0^{-1}b$, *and then comparing* x' to x as previously described. In Fig. 16 we show the resulting matrix T_0^{-1}. All entries whose absolute value is greater than 10^{-7} are shown in black.

TABLE 5
Numerical Results for Example 5

	Run times		Errors		Comp. ratios	
N	t_{fact}	t_{sub}	L_∞	L_2	A	L & U
128	0.52	0.00	1.86×10^{-7}	9.52×10^{-8}	1.84	1.75
256	2.22	0.02	2.49×10^{-7}	1.18×10^{-7}	2.87	2.73
512	8.55	0.05	2.70×10^{-7}	1.30×10^{-7}	4.82	4.60
1024	31.68	0.18	2.50×10^{-7}	1.32×10^{-7}	8.72	8.37
2048	124.53	0.59	2.95×10^{-7}	1.42×10^{-7}	16.60	15.95

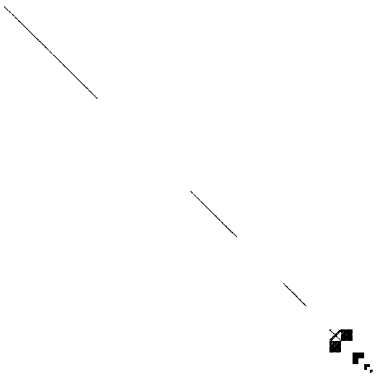

FIG. 16. The NS-form of T_0^{-1}, the inverse for operator in Example 6. Entries above the threshold 10^{-7} are shown in black. We observe that blocks $\hat{C}$ and $\tilde{B}$ are zero on several scales.

The results of this test are shown in Table 6. Column 2 contains the total time required to fill the matrix, compute the multiresolution LU factors, and compute the inverse. Columns 3 and 4 contain the error in the computed solution, and column 5 contains the compression ratio for the inverse operator. We observe that for this example, the time required to compute the inverse is roughly a factor of 2 greater than for computing the LU factorization.

Condition numbers. In Table 7 we present the condition numbers of matrices in four examples and the condition numbers of blocks which are actually factorized during the multiresolution *LU* factorization. The top row shows the condition number of the original matrix of size $N = 256$. In rows 2 through 7 we present the condition number of blocks A_j of the NS-form at different scales $j = 1, \ldots, 7$, which are factorized during multiresolution LU factorization.

The second column of Table 7 (Example 2) is most interesting since it shows nearly perfect condition numbers on all scales, whereas the original operator has condition number of $O(N^2)$, where the size of the matrix is $N \times N$.

10. GENERALIZATIONS AND CONCLUSIONS

The sparsity of multiresolution LU factorization algorithms does not depend on dimension. This is in a sharp contrast with the usual practice, where LU factorization

TABLE 6
Numerical Results for Example 6

	Run time	Errors		Comp. ratio
N	t_{inv}	L_∞	L_2	T_0^{-1}
128	0.55	2.14×10^{-7}	1.88×10^{-7}	21.90
256	1.44	2.18×10^{-7}	2.29×10^{-7}	74.73
512	3.05	2.60×10^{-7}	2.04×10^{-7}	222.34
1024	6.83	1.57×10^{-7}	1.55×10^{-7}	615.36
2048	15.79	1.53×10^{-7}	1.48×10^{-7}	1572.08

TABLE 7
Condition Numbers for Blocks A_j (N = 256)

	Examples			
Scale (j)	1	2	3	4
0[a]	1.41	6641[b]	2.31	1751
1	1.05	2.00	1.00	2.35
2	1.25	3.41	1.00	17.21
3	1.56	3.85	1.00	40.93
4	1.76	3.96	1.00	2.51
5	1.87	3.99	1.00	1.16
6	1.93	4.00	1.01	1.15
7	1.96	4.00	1.14	1.09

[a] Condition number of original matrix.
[b] Null space was removed from matrix.

is not recommended as an efficient approach in problems of dimension two or higher. For example, if we consider the Poisson equation, then LU decomposition is not considered as a practical option since the fill-ins will yield dense LU factors. We emphasize that the off-diagonal decay described in Theorems 5.1 and 5.3 is not specific to dimension one. Thus, multiresolution LU factorization in the wavelet system of coordinates becomes an option in solving elliptic problems in higher dimensions and we plan to demonstrate the multidimensional algorithm in a separate paper.

In multidimensional generalizations it is important to satisfy boundary conditions, and this implies using non-periodized wavelets. We note that the wavelet transform appears only as an orthogonal transformation in our approach and thus multiwavelets [3] or other orthogonal transformations may be used (provided the sparsity is maintained). We forsee future work in this direction, where one would try to optimize the choice of the basis (or coordinate transformation) in an adaptive manner for a given operator.

An additional feature of multiresolution forward and backward substitution algorithms that we did not address in this paper is adaptivity. Namely, if the right-hand side of the equation is compressible in wavelet bases, then the solution may be obtained with the number of operations proportional to the number of significant entries of the right-hand side.

The multiresolution LU factorization algorithms are intimately related to the idea of multiresolution homogenization [12]. The development in this direction may be found in [13], where multiresolution LU factorization algorithms are used as tools for fast computation. It is shown that these algorithms may be used for computing small eigenvalues (we note that the approach in [13] is not the power method for the inverse operator). It is clear, however, that the power method with inverse iteration may be used in conjunction with the fast multiresolution LU decomposition to find small eigenvalues of the original operator.

We note that it appears possible to generalize our approach to perform fast multiresolution QR (or LQ) factorization of NS-forms rather than multiresolution LU factorization. The operators for which QR factorization should work have sparse NS-forms, and thus Householder transformations may be described by sparse vectors. We plan to develop this algorithm at a later date. Such an algorithm will have a number of important applications.

Finally, we note that an easy access to inverse operators is very useful in a variety of situations, e.g., preconditioning, low rank updates of inverse operators, etc. In signal processing variants of LU algorithms are used in various linear estimation schemes and we hope that the algorithms of this paper will have an impact on this area as well.

APPENDIX: PIVOTING

In this Appendix we consider linear systems which are not positive-definite. Strictly speaking, our approach is not guaranteed to work for such systems, as may be seen in Example 4. However, some improvement has been observed and we describe the use of partial pivoting with the multiresolution direct methods which have been developed.

1. **Factorization.** We consider the effects of partial pivoting on multiresolution LU factorization described in Section 4. When a row exchange is encountered, we modify the governing equations in (4.5) using a permutation matrix R which contains the pivoting information

$$(R\hat{A}_j R^*)(R\tilde{A}_j) = R(A_j - \bar{A}_j), \tag{11.1a}$$

$$(R\hat{A}_j R^*)(R\tilde{B}_j) = R(B_j - \bar{B}_j), \tag{11.1b}$$

$$(\hat{C}_j R^*)(R\tilde{A}_j) = C_j - \bar{C}_j, \tag{11.1c}$$

and solve for $R\hat{A}_j R^*$, $R\tilde{A}_j$, $R\tilde{B}_j$, and $\hat{C}_j R^*$. To proceed to the next scale, $j + 1$, we require the decomposition of the term $\hat{C}_j \tilde{B}_j$. We note that this product may be obtained from the relation $(\hat{C}_j R^*)(R\tilde{B}_j) = \hat{C}_j \tilde{B}_j$, which implies that scale $j + 1$ is unaffected by row exchanges at scale j.

2. **Forward and backward substitution.** We note that multiresolution forward substitution with pivoting is completely analogous to standard forward substitution with pivoting. The governing equation (6.2) becomes

$$(RA_j R^*)R\tilde{d}_j = R(d_j - \bar{d}_j), \tag{11.2}$$

which we solve for $R\tilde{d}_j$. The projections of $C_j\tilde{d}_j$ on the next scale are computed by noting that $(C_j R^*)R\tilde{d}_j = C_j\tilde{d}_j$, which again implies that scale $j + 1$ is unaffected by row exchanges. The governing equations for multiresolution back substitution are unaffected by pivoting.

3. **Factorization of T_j blocks.** To solve matrix equations of the type in Eq. (7.1), we require the factorization of the blocks T_j, as described in Section (7). We

denote R_t as the permutation matrix describing the row exchanges in T_j, and let R describe the row exchanges in A_j. We obtain

$$(R\hat{A}_jR^*)(R\tilde{A}_j) = R(A_j - \bar{A}_j), \tag{11.3a}$$

$$(R\hat{A}_jR^*)(R\tilde{B}_j) = R(B_j - \bar{B}_j), \tag{11.3b}$$

$$(R_t\hat{C}_jR^*)(R\tilde{A}_j) = R_t(C_j - \bar{C}_j), \tag{11.3c}$$

$$(R_t\hat{T}_jR_t^*)(R_t\tilde{T}_j) = R_tT_j - (R_t\hat{C}_jR^*)(R\tilde{B}_j). \tag{11.3d}$$

We note that row exchanges in T_j will affect terms A_{j+1}, B_{j+1}, C_{j+1}, and T_{j+1} at the next scale, since these terms are computed as projections of $R_t(T_j - \bar{T}_j - \hat{C}_j\tilde{B}_j)$, instead of those in (4.11). To avoid modifying these terms, we use the direct factorization method in Section 4.3, where A_{j+1}, B_{j+1}, C_{j+1}, and T_{j+1} are computed *after* the factorization of T_j.

4. **Forward and backward substitution of matrices.** The governing equations in (7.4) for multiresolution forward substitution for matrices become

$$(R\hat{A}_jR^*)(R\tilde{A}_j) = R(A_j - \bar{A}_j), \tag{11.4a}$$

$$(R\hat{A}_jR^*)(R\tilde{B}_j) = R(B_j - \bar{B}_j), \tag{11.4b}$$

$$(R_t\hat{T}_jR_t^*)(R_t\tilde{C}_j) = R_t(C_j - \bar{C}_j) - (R_t\hat{C}_jR^*)(R\tilde{A}_j). \tag{11.4c}$$

Again, terms at the next scale, $j + 1$, are affected by the row exchanges in T_j, since we require the projection of $(R_t\hat{C}R^*)(R\tilde{B}_j) = R_t\hat{C}\tilde{B}_j$ instead of $\hat{C}_j\tilde{B}_j$. Multiresolution backward substitution for matrices is unaffected by pivoting.

5. **Factorization using alternate algorithm.** We now consider the effects of partial pivoting when using the alternate algorithm described in Section 4.4. We note that the factorization in (4.12) allows us to interchange rows between A_j and C_j (and, hence, B_j and T_j). This provides greater flexibility when choosing the pivots and may lead to a more accurate solution. We note, however, that since T_j is modified during the procedure, the terms A_{j+1}, B_{j+1}, C_{j+1}, T_{j+1} will be affected. Again, we use the direct factorization method described in Section 4.3 so that A_{j+1}, B_{j+1}, C_{j+1}, T_{j+1} are computed *after* T_j.

REFERENCES

1. G. Beylkin, R. R. Coifman, and V. Rokhlin, Fast wavelet transforms and numerical algorithms, I, *Comm. Pure Appl. Math.* **44** (1991), 141–183.
2. G. Beylkin, R. R. Coifman, and V. Rokhlin, Wavelets in numerical analysis, *in* ''Wavelets and Their Applications,'' pp. 181–210. Jones & Bartlett, Boston, 1992.
3. B. Alpert, G. Beylkin, R. R. Coifman, and V. Rokhlin, Wavelet-like bases for the fast solution of second-kind integral equations, *SIAM J. Sci. Statist. Comput.* **14** (1993), 159–174.
4. G. Beylkin, Wavelets, multiresolution analysis and fast numerical algorithms, unpublished manuscript, INRIA Lectures, *available at* ftp://amath.colorado.edu/pub/wavelets/papers/INRIA.ps.z.
5. G. Schulz, Iterative Berechnung der reziproken Matrix, *Z. Angew. Math. Mech.* **13** (1993), 57–59.

6. A. Ben-Israel and D. Cohen, On iterative computation of generalized inverses and associate projections, *SIAM J. Numer. Anal.* **3** (1966), 410–419.
7. T. Söderström and G. W. Stewart, On the numerical properties of an iterative method for computing the Moore–Penrose generalized inverse, *SIAM J. Numer. Anal.* **11** (1974), 61–74.
8. I. S. Duff, A. M. Erisman, and J. K. Reid, ''Direct Methods for Sparse Matrices,'' Clarendon Press, Oxford, 1986.
9. Multigrid repository, available at http://na.cs.yale.edu/mgnet/www/mgnet.html.
10. Y. Meyer, Wavelets and operators, *in* ''Analysis at Urbana,'' (N. T. Peck, E. Berkson, and J. Uhl, Eds.), Vol. 1, London Math. Society, Lecture Notes Series 137, Cambridge Univ. Press, Cambridge, UK, 1989.
11. S. Mallat, Multiresolution approximations and wavelet orthonormal bases in $\mathbf{L}^2(\mathbf{R})$, *Trans. Amer. Math. Soc.* **315** (1989), 69–87.
12. M. E. Brewster and G. Beylkin, A multiresolution strategy for numerical homogenization, *Appl. Comput. Harmon. Anal.* **2** (1995), 327–349; PAM Report 187, 1994.
13. G. Beylkin and N. Coult, A multiresolution strategy for homogenization of elliptic PDE's and associated eigenvalue problems, *Appl. Comput. Harmon. Anal.* **5** (1998), 129–155; PAM Report 270, 1996.
14. P. Tchamitchian, ''Wavelets: Theory and Application'' (G. Erlebacher, M. Y. Hussaini, and L. Jameson, Eds.) pp. 83–181, ICASE/LaRC Series in Computational Science and Engineering, Oxford Univ. Press, New York, 1996.
15. S. Jaffard, Propriétés des matrices bien localisees près de leur diagonale et quelques applications, *Ann. Inst. H. Poincaré Anal. Non Linéaire* **7** (1990), 461–476.
16. C. K. Chui, J. D. Ward, and P. W. Smith, Cholesky factorization of positive definite bi-infinite matrices, *Numer. Funct. Anal. Optim.* **5** (1982), 1–20.