UCLA
COMPUTATIONAL AND APPLIED MATHEMATICS

# On Two Variants of an Algebraic Wavelet Preconditioner

Tony F. Chan
Ke Chen

January 2002

CAM Report 02-02

Department of Mathematics
University of California, Los Angeles
Los Angeles, CA. 90095-1555

http://www.math.ucla.edu/applied/cam/index.html

# On two variants of an algebraic wavelet preconditioner

Tony F. Chan* and Ke Chen†

**Abstract**

A recursive method of constructing preconditioning matrices for the nonsymmetric stiffness matrix in a wavelet basis is proposed for solving a class of integral and differential equations. It is based on a level-by-level application of the wavelet scales de-coupling the different wavelet levels in a matrix form just as in the well-known NS form. The result is a powerful iterative method with built-in preconditioning leading to two specific algebraic multi-level iteration algorithms, one with an exact Schur preconditioning and the other with an approximate Schur preconditioning. Numerical examples are presented to illustrate the efficiency of the new algorithms.

**Keywords:** Multi-level preconditioner, Multi-resolution, Level-by-level transforms, Sparse approximate inverse, Schur complements, Wavelets.

**AMS subject class:** 65Y05, 65F35, 65Y20.

## Contents



*Department of Mathematics, University of California, Los Angeles, CA 90095-1555, USA. Email: chan@math.ucla.edu. Web: http://www.math.ucla.edu/~chan

†Department of Mathematical Sciences, University of Liverpool, Peach Street, Liverpool L69 7ZL, UK. Email: k.chen@liverpool.ac.uk. Web: http://www.liv.ac.uk/~cmchenke

# 1 Introduction

The discovery of wavelets is usually described as one of the most important advances in mathematics in the twentieth century as a result of joint efforts of pure and applied mathematicians. Through the powerful compression property, wavelets have satisfactorily solved many important problems in applied mathematics e.g. signal and image processing; see [23, 20, 34, 38] for a summary.

There remain many mathematical problems to be tackled before wavelets can be used for solution of differential and integral equations in a general setting. The traditional wavelets were designed mainly for regular domains and uniform meshes. This was one of the reasons why wavelets may not be immediately applicable to arbitrary problems. The introduction of the lifting idea, interpolatory wavelets [35, 25, 1] and adaptivity [17] provides a useful way of constructing wavelets functions in non-regular domains and in high dimensions.

However, the algebraic (sparse) structure of the matrix generated by a wavelet method is usually a finger-like one that is a difficult sparse pattern to deal with; refer to [10, 13, 15, 18]. Firstly direct solution of a linear system with such a matrix is either not feasible or inefficient. Secondly iterative solution requires a suitable preconditioner and this choice of preconditioner is usually dependent of the smoothness of the underlying operator (in addition to the assumptions for wavelets compression). Often a diagonal preconditioner is not sufficient. For some particular problems, several preconditioning techniques have been suggested. For instance, the finger matrix from wavelets representation of a Calderon-Zygmund operator plus a non-constant diagonal matrix cannot be preconditioned effectively by a diagonal matrix. In this case, one can use the idea of two-stage preconditioning as proposed in [13] or to use other modified wavelets methods such as a centring algorithm [15]; see also [38] for another modified algorithm and [19] for using approximate inverses. Overall, there exists a gap in realizing the full efficiency offered by wavelet bases for model problems. That is to say, a generally applicable iterative algorithm is still lacking. For a recent and general survey of iterative methods, refer to [31].

This paper proposes two related and efficient iterative algorithms based on the wavelet formulation for solving an operator equation with conventional arithmetic. Both algorithms use the Schur complements recursively but differ in how to use coarse levels to solve Schur complements equations. In the first algorithm, we precondition a Schur complement by using coarse levels while in the second we use approximate Schur complements to construct a preconditioner. We believe that our algorithms can be adapted to higher dimensional problems more easily than previous work in the subject.

The motivation of this work follows from the observation that any 1-scale compressed results (matrices) can be conveniently processed before applying the next scale. In this way, regular patterns created by past wavelet scales are not destroyed by the new scales like in the non-standard (NS) form [10] and unlike in the standard wavelet bases; we define the notation and give further details in Section 2. This radical but simple idea will be combined in Section 3 with the Schur complement method and Richardson iterations in a multi-level iterative algorithm. Moreover the Richardson iterations can be replaced by a recursive generalized minimal residuals (GMRES) method [30]. The essential assumption for this new algorithm to work is the invertibility of an approximate band matrix; in the Appendix we show that for a class of Calderon-Zygmund and pseudo-differential operators such an invertibility is ensured. In practice we found that our method works equally well for certain operators outside the type for which we can provide proofs. In Section 4, we present an alternative way of constructing the preconditioner by using approximate Schur complements. Section 5 discusses the complexity issues while Section 6 presents several numerical experiments to illustrate the effectiveness of the two new algorithms.

We remark that our first algorithm is similar to the framework of a NS form reformulation of the standard wavelets bases (based on the pyramid algorithm) but does not make use of the NS form itself, although our algorithm avoids a finger matrix (just like a NS form method) that could

arise from overlapping different wavelet scales. The NS form work was by Beylkin, Coifman and Rokhlin [10] often known as the BCR paper. As a by-product, the NS form reduces the flops from $O(n \log n)$ to $O(n)$. However, the NS form does not work with conventional arithmetic although operations with the underlying matrix (that has a regular sparse pattern) can be specially designed; in fact the NS form matrix itself is simply singular in conventional arithmetic. The recent work in [22] has attempted to develop a direct solution method based on the NS form that requires a careful choice of a threshold; here our method is iterative. In the context of designing recursive sparse preconditioners, it is similar to the ILUM type preconditioner to a certain extent [33]. Our second algorithm is similar to the algebraic multi-level iteration methods (AMLI) that were developed for finite elements [4, 2, 3, 36]; here our method uses wavelets and does not require estimating eigenvalues.

## 2 Wavelets splitting of an operator

This section will set up the notation to be used later and motivate the methods in the next sections. We first introduce the standard wavelet method. For simplicity, we shall concentrate on the Daubechies' order $m$ orthogonal wavelets with low pass filters $c_0, c_1, \ldots, c_{m-1}$ and high pass filters $d_0, d_1, \cdots, d_{m-1}$ (such that $d_j = (-1)^j c_{m-1-j}$). In fact, the ideas and expositions in this paper apply immediately to the more general bi-orthogonal wavelets [20].

Following the usual setting of [10, 20, 22, 11, 34], the filter coefficients $c_j$'s and $d_j$'s define the scaling function $\phi(x)$ and the wavelet function $\psi(x)$. Further, dilations and translations of $\phi(x)$ and $\psi(x)$ define a multi-resolution analysis for $\mathbf{L}^2$ in $\mathbf{d}$-dimensions, in particular,

$$\mathbf{L}^2(\mathbb{R}^{\mathbf{d}}) = \mathbf{V}_\ell \bigoplus \mathbf{W}_\ell \bigoplus \mathbf{W}_{\ell-1} \bigoplus \cdots \bigoplus \mathbf{W}_{-1} \bigoplus \mathbf{W}_{-2} \bigoplus \cdots = \mathbf{V}_\ell \bigoplus_{j=-\infty}^{\ell} \mathbf{W}_j, \tag{1}$$

where the subspaces satisfy the relations

$$\begin{cases} \mathbf{V}_\ell \subset \mathbf{V}_{\ell-1} \subset \cdots \subset V_1 \subset V_0 \subset V_{-1} \subset \cdots, \\ \mathbf{V}_{j-1} = \mathbf{V}_j \bigoplus \mathbf{W}_j. \end{cases}$$

In numerical realisations, we select a finite dimension space $\mathbf{V}_0$ (in the finest scale) as our approximation space to the infinite decomposition of $\mathbf{L}^2$ in (1) i.e. effectively use

$$\mathbf{V}_\ell \bigoplus_{j=1}^{\ell} \mathbf{W}_j \tag{2}$$

to approximate $\mathbf{L}^2(\mathbb{R}^{\mathbf{d}})$. Consequently for a given operator $\mathcal{T} : \mathbf{L}^2 \to \mathbf{L}^2$, its infinite and exact operator representation in wavelet bases

$$\begin{aligned} \mathcal{T} &= \mathcal{P}_\ell \mathcal{T} \mathcal{P}_\ell + \sum_{j=-\infty}^{\ell} (\mathcal{Q}_{j-1} \mathcal{T} \mathcal{P}_{j-1} - \mathcal{P}_j \mathcal{T} \mathcal{P}_j) \\ &= \mathcal{P}_\ell \mathcal{T} \mathcal{P}_\ell + \sum_{j=-\infty}^{\ell} (\mathcal{Q}_j \mathcal{T} \mathcal{Q}_j + \mathcal{Q}_j \mathcal{T} \mathcal{P}_j + \mathcal{P}_j \mathcal{T} \mathcal{Q}_j) \end{aligned}$$

is *approximated* in space $\mathbf{V}_0$ by

$$\mathcal{T}_0 = \mathcal{P}_0 \mathcal{T} \mathcal{P}_0 = \mathcal{P}_\ell \mathcal{T} \mathcal{P}_\ell + \sum_{j=1}^{\ell} (\mathcal{Q}_j \mathcal{T} \mathcal{Q}_j + \mathcal{Q}_j \mathcal{T} \mathcal{P}_j + \mathcal{P}_j \mathcal{T} \mathcal{Q}_j),$$

where $\mathcal{P}_j : \mathbf{L}^2 \to \mathbf{V}_j$ and $\mathcal{Q}_j = \mathcal{P}_{j-1} - \mathcal{P}_j : \mathbf{L}^2 \to \mathbf{V}_{j-1} - \mathbf{V}_j \equiv \mathbf{W}_j$ are both projection operators. For brevity, define operators

$$\begin{array}{llll} \mathcal{A}_j = \mathcal{Q}_j \mathcal{T} \mathcal{Q}_j : & \mathbf{W}_j \to \mathbf{W}_j, & \mathcal{B}_j = \mathcal{Q}_j \mathcal{T} \mathcal{P}_j : & \mathbf{V}_j \to \mathbf{W}_j, \\ \mathcal{C}_j = \mathcal{P}_j \mathcal{T} \mathcal{Q}_j : & \mathbf{W}_j \to \mathbf{V}_j, & \mathcal{T}_j = \mathcal{P}_j \mathcal{T} \mathcal{P}_j : & \mathbf{V}_j \to \mathbf{V}_j. \end{array}$$

Then one can observe that

$$\mathcal{T}_{j-1} = \mathcal{A}_j + \mathcal{B}_j + \mathcal{C}_j + \mathcal{T}_j. \tag{3}$$

A further observation based on $\mathcal{T}_{j-1} : \mathbf{V}_{j-1} \to \mathbf{V}_{j-1}$ and $\mathbf{V}_{j-1} = \mathbf{V}_j \bigoplus \mathbf{W}_j$ is that the wavelet coefficients of $\mathcal{T}_{j-1}$ will be equivalently generated by the block operator

$$\begin{bmatrix} \mathcal{A}_j & \mathcal{B}_j \\ \mathcal{C}_j & \mathcal{T}_j \end{bmatrix} \quad : \quad \begin{pmatrix} \mathbf{W}_j \\ \mathbf{V}_j \end{pmatrix} \to \begin{pmatrix} \mathbf{W}_j \\ \mathbf{V}_j \end{pmatrix}.$$

Now we change the notation and consider the discretization of all continuous operators. Define matrix $T_0 = A$ as the representation of operator $\mathcal{T}_0$ in space $\mathbf{V}_0$. Assume that $A$ on the finest level (scale) $j = 0$ is of dimension $\tau_0 = n$. Then the dimension of matrices on a coarse level $j$ is $\tau_j = \tau_0/2^j$ for $j = 1, 2, \ldots, \ell$. The operator splitting in (3) for the case of $\mathbf{d} = 1$ (higher dimensions can be discussed similarly [10, 22]) corresponds to the two-dimensional wavelet transform

$$\widetilde{T}_{j-1} = W_{j-1} T_{j-1} W_{j-1}^{\top} = \begin{bmatrix} A_j & B_j \\ C_j & T_j \end{bmatrix}_{\tau_{j-1} \times \tau_{j-1}} \tag{4}$$

where the one level transform from $j-1$ to $j$ (for any $j = 1, 2, \cdots, \ell$) is

$$W_{j-1} = (W_{j-1})_{\tau_{j-1} \times \tau_{j-1}} = \begin{bmatrix} P_j \\ Q_j \end{bmatrix},$$

$$\begin{array}{ll} A_j = (A_j)_{\tau_j \times \tau_j} = Q_j T_{j-1} Q_j^{\top}, & B_j = (B_j)_{\tau_j \times \tau_j} = Q_j T_{j-1} P_j^{\top}, \\ C_j = (C_j)_{\tau_j \times \tau_j} = P_j T_{j-1} Q_j^{\top}, & T_j = (T_j)_{\tau_j \times \tau_j} = P_j T_{j-1} P_j^{\top}, \end{array}$$

with rectangular matrices $P_j$ and $Q_j$ (corresponding to operators $\mathcal{P}_j$ and $\mathcal{Q}_j$) defined respectively as

$$P_j = \begin{bmatrix} c_0 & c_1 & \cdots & \cdots & c_{m-1} & & \\ & & c_0 & c_1 & \cdots & c_{m-1} & \\ & & & & \ddots & \ddots & \ddots \\ c_2 & c_3 & \cdots & c_{m-1} & & c_0 & c_1 \end{bmatrix}_{\tau_j \times \tau_{j-1}},$$

$$Q_j = \begin{bmatrix} d_0 & d_1 & \cdots & \cdots & d_{m-1} & & \\ & & d_0 & d_1 & \cdots & d_{m-1} & \\ & & & & \ddots & \ddots & \ddots \\ d_2 & d_3 & \cdots & d_{m-1} & & d_0 & d_1 \end{bmatrix}_{\tau_j \times \tau_{j-1}}.$$

For a class of useful and strongly elliptic operators i.e. Calderon-Zygmund and pseudo-differential operators, it was shown in BCR [10] that matrices $A_j = (\alpha_{k,i}^j)$, $B_j = (\beta_{k,i}^j)$, $C_j = (\gamma_{k,i}^j)$ are indeed 'sparse' satisfying the decaying property

$$\left|\alpha_{k,i}^j\right| + \left|\beta_{k,i}^j\right| + \left|\gamma_{k,i}^j\right| \leq \frac{c_{m,j}}{(1 + |k-i|)^{m+1}}, \tag{5}$$

where $|k-i| \geq 2m$ and $c_{m,j}$ is a generic constant depending on $m$ and $j$ only.

To observe a relationship between the above level-by-level form and the standard wavelet representation, define a square matrix of size $\tau_0 \times \tau_0 = n \times n$ for any $j = 1, 2, \ldots, \ell$

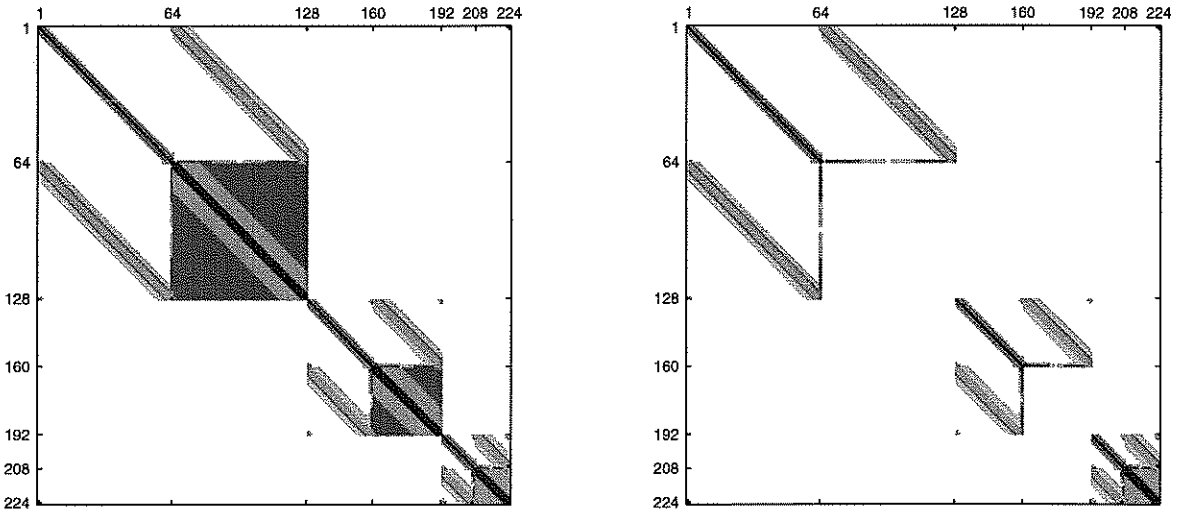
$$\overline{W}_j = \begin{bmatrix} I_{\nu_j} & \\ & W_{j-1} \end{bmatrix}, \tag{6}$$

where $\nu_j = n - \tau_{j-1}$; clearly $\nu_1 = 0$ and $\overline{W}_1 = W_0$. Then the standard wavelet transform can be written as

$$W = \overline{W}_\ell \cdots \overline{W}_2 \overline{W}_1, \tag{7}$$

that transforms matrix $A$ into $\widetilde{A} = WAW^\top$.

Figure 1: The level-by-level form (left) versus the non-standard wavelet form [10] (right)




Thus the diagonal blocks of $\widetilde{A}$ are the same as $A_j$'s of a level-by-level form. However the off-diagonal blocks of the former are different from $B_j$ and $C_j$ of the latter. To gain some insight into the structure of the off-diagonal blocks of matrix $\widetilde{A}$ with the standard wavelet transform, we consider the following case of $\ell = 3$ (3-levels) and $m = 4$ (order 4) wavelets. Firstly after level 1 transform, we obtain

$$\widetilde{A}_1 = \overline{W}_1 A \overline{W}_1^\top = W_0 T_0 W_0^\top = \begin{bmatrix} A_1 & B_1 \\ C_1 & T_1 \end{bmatrix}_{n \times n}.$$

Secondly after level 2 transform, we get

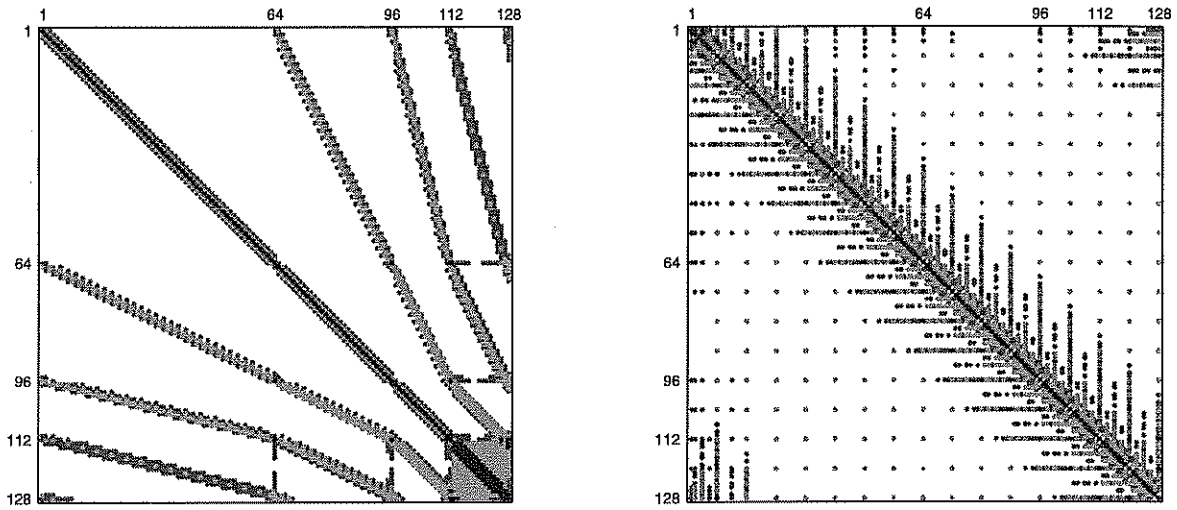
$$\widetilde{A}_2 = \overline{W}_2 \widetilde{A}_1 \overline{W}_2^\top = \begin{bmatrix} A_1 & B_1 W_1^\top \\ W_1 C_1 & \begin{bmatrix} A_2 & B_2 \\ C_2 & T_2 \end{bmatrix} \end{bmatrix}_{n \times n}.$$

Finally after level 3 transform, we arrive at

$$\widetilde{A}_3 = \overline{W}_3 \widetilde{A}_2 \overline{W}_3^\top = \begin{bmatrix} A_1 & B_1 W_1^\top \begin{bmatrix} I_{3n/4} & \\ & W_2^\top \end{bmatrix} \\ \begin{bmatrix} I_{3n/4} & \\ & W_2 \end{bmatrix} W_1 C_1 & \begin{bmatrix} A_2 & B_2 W_2^\top \\ W_2 C_2 & \begin{bmatrix} A_3 & B_3 \\ C_3 & T_3 \end{bmatrix} \end{bmatrix} \end{bmatrix}_{n \times n}. \tag{8}$$

Clearly the off-diagonal blocks of $\widetilde{A}_3$ are perturbations of that of the level-by-level form off-diagonal blocks $B_j$ and $C_j$; in fact the one-sided transforms for the off-diagonal blocks are responsible for the resulting (complicated) sparsity structure. This can be observed more clearly for a typical example with $n = 128$ (3 levels and $m = 4$) in Fig.1 where the left plot shows the level-by-level representation set-up that will be used in this paper and in Fig.2 where the left plot shows the standard wavelet representation as in (8).

Figure 2: The standard wavelet form representation (left) versus an alternative centering form [15] (right) for the example in Figure 1




Motivated by the exposition in (8) of the standard form, we shall propose a preconditioning and iterative scheme that operates on recursive 1-level transforms. Thus it will have the advantage of making full use of the NS form idea and its theory while avoiding the problem of a non-operational NS form matrix.

**Remark 1** *Starting from the level-by-level set-up, taking $T_\ell$ and the collection of all triplets $\{(A_j, B_j, C_j\}_{1\le j\le \ell}$ as a sparse approximation for $T_0$ is the idea of the NS form [10, 22]. By way of comparison, in Fig.1, the NS form representation versus the level-by-level form are shown. It turns out that this work uses the identical set-up to the NS form without using the NS form formulation itself because we shall not use the proposed sparse approximation. Note that the centering algorithm [15] (see the right plot in Fig.2) is designed as a permutation of the standard wavelet form (see the left plot of Fig.2) and is only applicable to a special class of problems where its performance is better.*

## 3 An exact Schur preconditioner with level-by-level wavelets

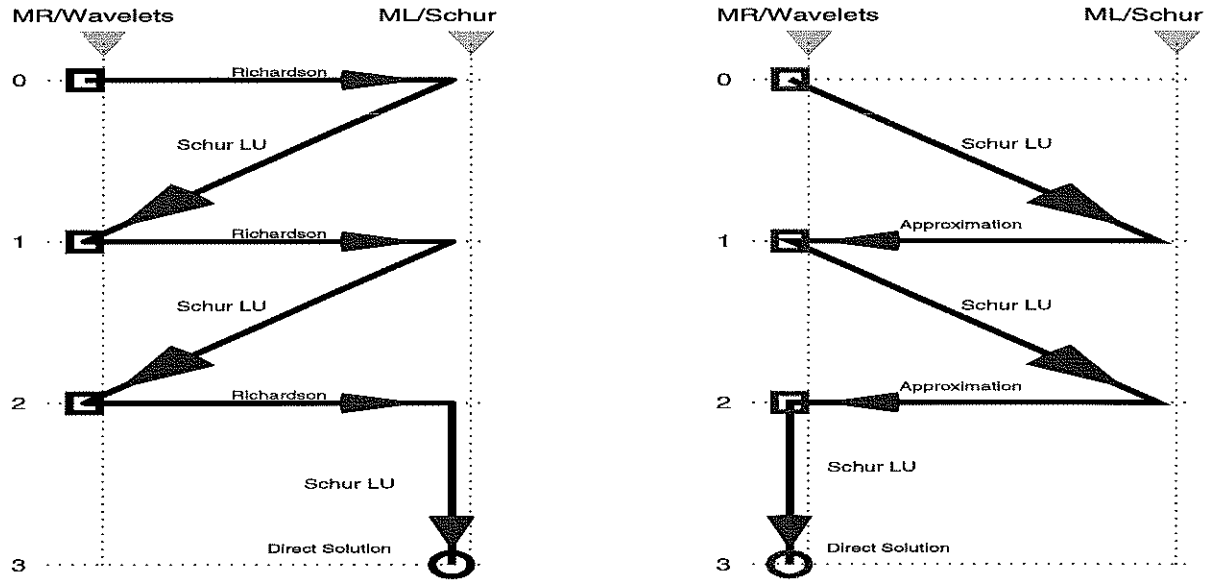
We now present our first and new recursive method for solving the linear system $Ax = b$ defined on the finest scale $\mathbf{V}_0$ i.e.

$$T_0 x_0 = b_0, \tag{9}$$

where $T_0 = A_0 = A$ is of size $\tau_0 \times \tau_0 = n \times n$ as discussed in the previous section, and $x_0,\ b_0 \in \mathbb{R}^n$. Instead of considering a representation of $T_0$ in the decomposition space (2) and then the resulting linear system, we propose to follow the space decomposition and the intermediate linear system in a

level-by-level manner. A sketch of this method is given in Fig.3 (the left plot) where we try to show a relationship between the multi-resolution (MR for wavelet representation) and the multi-level (ML for preconditioning via Schur) ideas from the finest level (top) to the coarsest level (bottom).

Figure 3: Illustration of Algorithms 1 (left) and 2 (right). Here we take $\ell = 3$ levels (0 indicates the finest level and 3 the coarsest level), use '□' to indicate a DWT step (one level of wavelets) and '◯' to denote the direct solution process on the coarsest level. The arrows denote the sequence of operations (written on the arrowed lines) with each algorithm interacting the two states (two columns on the plots) of multi-resolution wavelets and multi-level Schur decomposition. The left plot shows that for Algorithm 1, a Richardson step (or GMRES) takes the results of a DWT step to the next level via the Schur decomposition while the right plot shows that the Schur decomposition takes the results of a DWT step to the next level via a Schur approximation.




Firstly at level 0, we consider $\mathbf{V}_0 = \mathbf{V}_1 \bigoplus \mathbf{W}_1$ and the wavelet transform (4) yields

$$\widetilde{T}_0 \widetilde{x}_0 = \widetilde{b}_0 \tag{10}$$

where $\widetilde{x}_0 = W_0 x_0$ and $\widetilde{b}_0 = W_0 b_0$. Since

$$\widetilde{T}_0 = \begin{bmatrix} A_1 & B_1 \\ C_1 & T_1 \end{bmatrix}_{n \times n}, \tag{11}$$

following the general result in (5), it is appropriate to consider the approximation of $A_1, B_1, C_1$ by band matrices. To be more precise, let $\mathbf{B}_\mu(D)$ denote a banded matrix of $D$ with semi-bandwidth $\mu$

$$(\mathbf{B}_\mu(D))_{ik} = \begin{cases} D_{ik}, & \text{if } |i-k| \le \mu, \\ 0, & \text{otherwise} \end{cases}$$

where integer $\mu \ge 0$. Define $\overline{A}_1 = \mathbf{B}_\mu(A_1)$, $\overline{B}_1 = \mathbf{B}_\mu(B_1)$, $\overline{C}_1 = \mathbf{B}_\mu(C_1)$ for some suitable $\mu$ (to be specified later). Then matrix $\overline{T}_0 = \overline{T}_0 - \overline{R}_0$ can be approximated by

$$\overline{T}_0 = \begin{bmatrix} \overline{A}_1 & \overline{B}_1 \\ \overline{C}_1 & T_1 \end{bmatrix}_{n \times n}. \tag{12}$$

Or equivalently matrix

$$\overline{R}_0 = \begin{bmatrix} \overline{A}_1 - A_1 & \overline{B}_1 - B_1 \\ \overline{C}_1 - C_1 & 0 \end{bmatrix}_{n \times n} \tag{13}$$

is expected to be small in some norm (refer to the Appendix). Write equation (10) as

$$\left(\overline{T}_0 - \overline{R}_0\right) \widetilde{x}_0 = \widetilde{b}_0. \tag{14}$$

Consequently we propose to use $M_0 = \overline{T}_0$ as our preconditioner to equation (14). This preconditioner can be used to accelerate iterative solution; we shall consider two such methods: the Richardson method and the GMRES method [30].

The most important step in an iterative method is to solve the preconditioning equation:

$$\overline{T}_0 y_0 = r_0 \tag{15}$$

or in a decomposed form

$$\begin{bmatrix} \overline{A}_1 & \overline{B}_1 \\ \overline{C}_1 & T_1 \end{bmatrix} \begin{pmatrix} y_0^{(1)} \\ y_0^{(2)} \end{pmatrix} = \begin{pmatrix} r_0^{(1)} \\ r_0^{(2)} \end{pmatrix}. \tag{16}$$

Using the Schur complement method we obtain

$$\begin{cases} \overline{A}_1 z_1 &= r_0^{(1)} \\ z_2 &= r_0^{(2)} - \overline{C}_1 z_1 \\ \left(T_1 - \overline{C}_1 \overline{A}^{-1} \overline{B}_1\right) y_0^{(2)} &= z_2 \\ y_0^{(1)} &= z_1 - \overline{A}^{-1} \overline{B}_1 y_0^{(2)}. \end{cases} \tag{17}$$

Here the third equation of (17), unless its dimension is small (i.e. $\mathbf{V}_1$ is the coarsest scale), has to be solved by an iterative method with the preconditioner $T_1$; we shall denote the preconditioning step by

$$T_1 x_1 = b_1, \tag{18}$$

where $T_1$ is of size $\tau_1 \times \tau_1$. This sets up the sequence of a multilevel method where the main characteristic is that each Schur complements equation in its exact form is solved iteratively with a preconditioner that involves coarse level solutions.

At any level $j$ $(1 \leq j < \ell)$, the solution of the following linear system

$$T_j x_j = b_j, \tag{19}$$

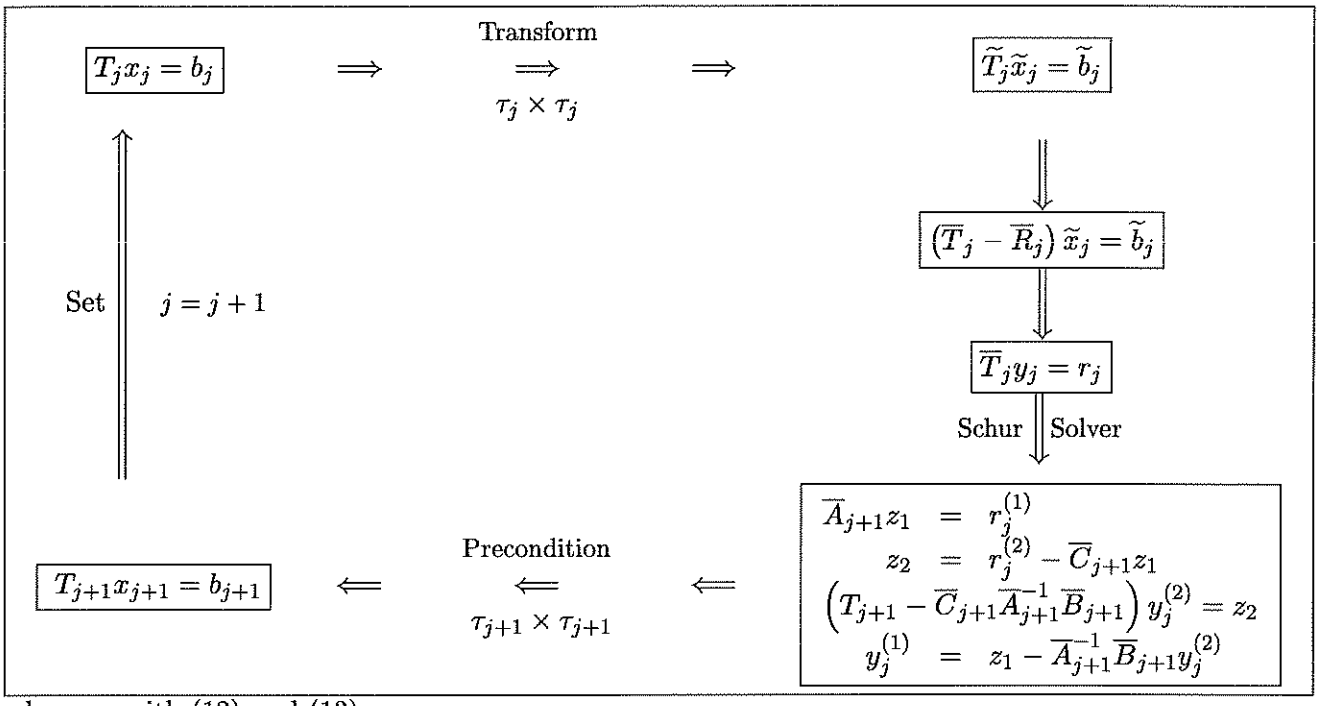
with $T_j$ of size $\tau_j \times \tau_j$ and through solving

$$\widetilde{T}_j \widetilde{x}_j = \widetilde{b}_j,$$

can be similarly reduced to that of

$$T_{j+1} x_{j+1} = b_{j+1}, \tag{20}$$

with $T_{j+1}$ of size $\tau_{j+1} \times \tau_{j+1}$. The solution procedure from the finest level to the coarsest level can be illustrated by the following diagram (for $j = 1, 2, \ldots, \ell - 1$):




where as with (12) and (13)

$$\overline{T}_j = \begin{bmatrix} \overline{A}_{j+1} & \overline{B}_{j+1} \\ \overline{C}_{j+1} & T_{j+1} \end{bmatrix}_{n\times n} \quad \text{and} \quad \overline{R}_j = \begin{bmatrix} \overline{A}_{j+1} - A_{j+1} & \overline{B}_{j+1} - B_{j+1} \\ \overline{C}_{j+1} - C_{j+1} & 0 \end{bmatrix}_{n\times n}. \tag{21}$$

The coarsest level is $j = \ell$, as set up in the previous section, where a system like (20) is solved by a direct elimination method. As with conventional multi-level methods, each fine level iteration leads to many coarse level iteration cycles. This can be illustrated in Fig. 4 where $\ell = 3$ and $\mu = 2$ (top plot), 3 (bottom plot) are assumed and at the coarsest level ($\bigotimes$) a direct solution is used. In practice, a variable $\mu = \mu_j$ on level $j$ may be used to achieve certain accuracy for the preconditioning step i.e. convergence up to a tolerance is pursued whilst by way of comparison smoothing rather convergence is desired in an usual multilevel method. Our experiments have shown that $\mu = 1,\ 2$ are often sufficient to ensure the overall convergence.

We now summarise the formulation as an algorithm. The iterative solver for (19) at level $i$ can be the Richardson method

$$\overline{T}_j \widetilde{x}_j^{(k)} = \overline{R}_j \widetilde{x}_j^{(k-1)} + \widetilde{b}_j \qquad \text{for } k = 1, 2, \cdots, \mu_j$$

or the GMRES method [30] for solving $\widetilde{T}_j \widetilde{x}_j = \widetilde{b}_j$ (or actually a combination of the two). For simplicity and generality, we shall use the word "SOLVE" to denote such an iterative solver (either Richardson or GMRES).

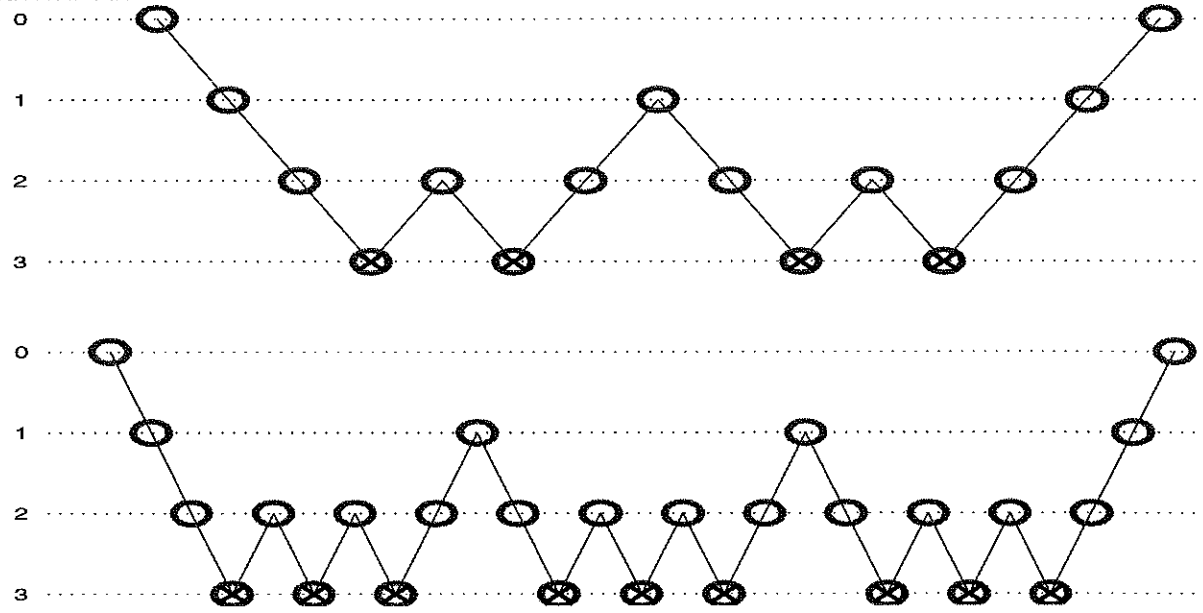
**Algorithm 1 (Recursive I)**

1. *Set $j = 0$ and start on the finest level.*
2. *Apply one level DWT to $T_j x_j = b_j$ to obtain* $\tilde{T}_j \tilde{x}_j = \tilde{b}_j$.
3. **Use** $\mu_j$ *steps of* SOLVE *for* $\tilde{T}_j \tilde{x}_j = \tilde{b}_j$.
4. *In each step, implement the preconditioner $\overline{T}_j$:* *i.e. solve $\overline{T}_j y_j = r_j$ or*

$$\begin{cases} \overline{A}_{j+1} z_1 = r_j^{(1)} \\ z_2 = r_j^{(2)} - \overline{C}_{j+1} z_1 \\ \left(T_{j+1} - \overline{C}_{j+1}\overline{A}_{j+1}^{-1}\overline{B}_{j+1}\right) y_j^{(2)} = z_2 \\ y_j^{(1)} = z_1 - \overline{A}_{j+1}^{-1}\overline{B}_{j+1} y_j^{(2)} \end{cases}$$

**Restrict** *to the coarse level:*

Figure 4: Iteration cycling patterns of Algorithm 1 with $\ell = 3$ levels: top for $\mu = 2$ and bottom for $\mu = 3$. In each case, one solves a fine level equation (starting from the finest level 0) by iteratively solving coarser level equations $\mu$ times; on the coarsest level $\otimes$ (here level 3) a direct solution is carried out.




5. **Use** SOLVE *for the above third equation with the preconditioner* $T_{j+1}$ *i.e. solve* $T_{j+1}x_{j+1} = b_{j+1}$.

6. *Set* $j := j + 1$.

7. *If* $j = \ell$ *(on the coarsest level), apply a* **direct solver** *to* $T_j x_j = b_j$ *and proceed with Step 8; otherwise return to Step 2.*

8. *Set* $j := j - 1$.

9. **Interpolate** *the coarse level* $j + 1$ *solution to the fine level* $j$*:*

$$x_j^{(2)} = x_{j+1}, \qquad x_j^{(1)} = z_1 - \bar{A}_j^{-1}\bar{B}_j x_{j+1} \qquad \textit{i.e.}$$

$$x_j = \begin{bmatrix} x_j^{(1)} \\ x_j^{(2)} \end{bmatrix} = \begin{bmatrix} x_j^{(1)} \\ x_{j+1} \end{bmatrix}.$$

10. *Apply one level inverse DWT to* $\widetilde{y}_j$ *to obtain* $y_j$.

11. *If* $j = 0$ *(on the finest level), check the residual error — if small enough accept the solution* $x_0$ *and stop the algorithm. If* $j > 0$*, check if* $\mu_j$ *steps (cycles) have been carried out; if not, return to Step 2 otherwise continue with Step 8 on level* $j$.

The rate of convergence of this algorithm depends on how well the matrix $\overline{T}_j$ approximates $T_j$ and this approximation is known to be accurate for a suitable $\mu$ and for a class of Calderon-Zygmund and pseudo-differential operators [10]. For this class of problems, it remains to discuss the invertibility of matrix $\overline{A_j}$ which is done in the Appendix; a detailed analysis on $\overline{T}_j \approx T_j$ may be done along the same lines as Lemma 1. For other problem classes, the algorithm may not work at all for the simple reason that $\overline{A_j}$ may be singular e.g. the diagonal of matrix $A = T_0$ in (9) may have zero entries. Some extensions based on the idea of [13] may be applied as discussed in Section 6.

**Remark 2** *We remark that for a class of general sparse linear systems, Saad, Zhang, Botta, Wubs et al [28, 12, 32, 33] have proposed a recursive multi-level preconditioner (named as ILUM) similar to this Algorithm 1. The first difference is that we need to apply one level of wavelets to achieve a nearly sparse matrix while these works start from a sparse matrix and permute it to obtain a desirable pattern suitable for Schur decomposition. The second difference is that we propose an iterative step before calling for the Schur decomposition while these works try to compute the exact Schur decomposition approximately. Therefore it is feasible to refine our Algorithm 1 to adopt the ILUM idea (using independent sets) for other problem types. However one needs to be careful in selecting the dimensions of the leading Schur block if a DWT is required for compression purpose.*

# 4 An approximate Schur preconditioner with level-by-level wavelets

In the previous algorithm, we use coarse level equations to precondition the fine level Schur complement equation. We now propose an alternative way of constructing a preconditioner for a fine level equation. Namely we approximate and compute the fine level Schur complement before employing coarse levels to solve the approximated Schur complement equation. A sketch of this method is shown in Fig.3 showing the natural coupling of wavelet representation (level-by-level form) and Schur complement. symbol '□'. To differentiate from Algorithm 1, we change the notation for all matrices.

At any level $k$ for $k = 0, 1, 2, \ldots, \ell - 1$, consider the solution (compare to (19))

$$A^{(k)} x_k = b_k. \tag{22}$$

Applying 1-level of DWT, we obtain

$$\widetilde{A^{(k)}}\widetilde{x_k} = \widetilde{b_k} \quad \text{with} \quad \widetilde{A^{(k)}} = W_k A^{(k)} W_k^\top = \begin{bmatrix} A_{11}^{(k)} & A_{12}^{(k)} \\ A_{21}^{(k)} & A_{22}^{(k)} \end{bmatrix}. \tag{23}$$

Note that we have the block LU decomposition

$$A^{(k)} = \begin{bmatrix} A_{11}^{(k)} & 0 \\ A_{21}^{(k)} & I \end{bmatrix} \begin{bmatrix} I & A_{11}^{(k)^{-1}} A_{12}^{(k)} \\ 0 & \mathbf{S}^{(k)} \end{bmatrix}$$

where $\mathbf{S}^{(k)} = A_{22}^{(k)} - A_{21}^{(k)} A_{11}^{(k)^{-1}} A_{12}^{(k)}$ is the true Schur complement. To approximate this Schur complement, we must consider approximating the second term in the above $\mathbf{S}^{(k)}$. We propose to form band matrix approximations

$$\begin{aligned} B_{11} = \mathbf{B}_\mu(A_{11}^{(k)}) &\approx A_{11}^{(k)^{-1}}, \\ \overline{A_{12}} = \mathbf{B}_\mu(A_{12}^{(k)}) &\approx A_{12}^{(k)}, \\ \overline{A_{21}} = \mathbf{B}_\mu(A_{21}^{(k)}) &\approx A_{21}^{(k)}. \end{aligned}$$

For level $k = 0$, these approximations are possible for a small bandwidth $\mu$; see Appendix. Seeking a band approximation to the inverse of $A_{11}^{(k)}$ makes sense because $A_{11}^{(k)} = A_k$ is expected to have a decaying property (refer to (5)). Let $\mathcal{S}$ denote the set of all matrices that have the sparsity pattern of a band $\mu$ matrix $\mathbf{B}_\mu(A_{11}^{(k)})$. The formation of a sparse approximate inverse (SPAI) is to find a band matrix $B_{11} \in \mathcal{S}$ such that

$$\min_{B \in \mathcal{S}} \|A_{11}^{(k)} B - I\|_F = \|A_{11}^{(k)} B_{11} - I\|_F.$$

Refer to [8, 24, 16]. Briefly as with most SPAI methods, the use of $F$-norm decouples the minimisation into least squares (LS) problems for individual columns $c_j$ of $B_{11}$. More precisely, owning to

$$\|A_{11}^{(k)}B_{11} - I\|_F^2 = \sum_{j=1}^{\tau_k} \|A_{11}^{(k)}c_j - e_j\|_2^2,$$

the $j$-th LS problem is to solve $A_{11}^{(k)}c_j = e_j$ which is not expensive since $c_j$ is sparse. Once $B_{11}$ is found, define an approximation to the true Schur complement $\mathbf{S}^{(k)}$ as

$$S^{(k)} = A_{22}^{(k)} - \overline{A_{21}^{(k)}}B_{11}\overline{A_{12}^{(k)}},$$

and set $A^{(k+1)} = S^{(k)}$. This generates a sequence of matrices $A^{(k)}$.

Now comes the most important step about the new preconditioner. Setting $M^{(\ell)} = A^{(\ell)} = S^{(\ell-1)}$, the fine level preconditioner $M^{(0)}$ is defined recursively by

$$M^{(k)} = \begin{bmatrix} B_{11}^{(k)^{-1}} & 0 \\ A_{21}^{(k)} & I \end{bmatrix} \begin{bmatrix} I & B_{11}^{(k)}A_{12}^{(k)} \\ 0 & S^{(k)} \end{bmatrix}, \tag{24}$$

where $S^{(k)} \approx M^{(k+1)}$ is an approximation to the true Schur complement $\mathbf{S}^{(k)}$ of $A^{(k)}$. Here $k = \ell-1, \ell-2, \ldots, 1, 0$. Observe that this preconditioner is defined through the V-cycling pattern recursively using the coarse levels.

To go beyond the V-cycling, we propose a simple residual correction idea. We view the solution $y_k^{[j]}$ of the preconditioning equation (compare to (15) and (9))

$$M^{(k)}y_k = r_k \tag{25}$$

as an approximate solution to the equation

$$T_k y_k = r_k.$$

Then the residual vector is $\bar{r}_k = r_k - T_k y_k^{[j]}$. This calls for a repeated solution $M^{(k)}\delta_j = \bar{r}_k$ and gives the correction and a new approximate solution to (25):

$$y_k^{[j+1]} = y_k^{[j]} + \delta_j,$$

for $j = 1, 2, \ldots, \nu_k$. In practice we take $\nu_k = \nu$ for a small $\nu$ (say $\nu = 2$ for a W-cycling; see the top plot in Fig.4) as our experiments suggest that $\nu \leq 2$ is sufficient to ensure the overall convergence.

Thus an essential feature of this method different from Algorithm 1 is that every approximated Schur complement matrix needs to be transformed to the next wavelet level in order to admit the matrix splitting (23) while inverse transforms are needed to pass coarse level information back to a fine level as illustrated in Fig.3. Ideally we may wish to use $A^{(k)} = A_{22}^{(k)} - A_{21}^{(k)}B_{11}^{(k)}A_{12}^{(k)}$ generate the approximate Schur complement but taking $A_{21}^{(k)}$ and $A_{12}^{(k)}$ as full matrices would jeopardize the efficiency of the overall iterative method. We proposed to use band matrices (or thresholding) to approximate these two quantities just as in (13) with Algorithm 1.

To summarise, the solution of the preconditioning equation from the finest level to the coarsest level and back up is illustrated in Fig.5 for $k = 0, 1, 2, \ldots, \ell-1$, where '$\star$' means the same entry and exit point. The general algorithm for solving $M^{(0)}y_0 = r_0$ can be stated as follows:

**Algorithm 2 (Recursive II)**

**Set-up Stage**

*for* $k = 0, 1, 2, \cdots, \ell - 1$

1. *Apply one level DWT to* $T_k$ *to obtain* $A_{11}^{(k)}, A_{12}^{(k)}, A_{21}^{(k)}, A_{22}^{(k)}$.
2. *Find the approximate inverse* $B_{11}^{(k)} \approx A_{11}^{(k)^{-1}}$.
3. *Generate the matrix* $T_{k+1} = A^{(k)} = A_{22}^{(k)} - A_{21}^{(k)} B_{11}^{(k)} A_{12}^{(k)}$.

*end*

**Solution Stage**

1. *Set* $k = 0$ *and start on the finest level.*
2. *Apply one level DWT to* $r_k$ *and consider* $\tilde{T}_k \tilde{y}_k = \tilde{r}_k$.
3. **Solve** *the preconditioning equation* $M^{(k)} \tilde{y}_k = \tilde{r}_k$ *by*

   **Restricting** *to the coarse level:*
   $$\begin{cases} z_1 &= B_{11}^{(k)} \tilde{r}_k^{(1)} \\ z_2 &= \tilde{r}_k^{(2)} - A_{21}^{(k)} z_1 \\ T_{k+1} \tilde{y}_k^{(2)} &= z_2 \end{cases}$$
4. **Solve** *for the above third equation at the next level* $T_{k+1} y_{k+1} = r_{k+1}$.
5. *Set* $k := k + 1$.
6. *If* $k = \ell$ *(on the coarsest level), apply a* **direct solver** *to* $T_k y_k = r_k$ *and proceed with Step 8; otherwise return to Step 2.*
7. *Set* $k := k - 1$.
8. **Interpolate** *the coarse level* $k + 1$ *solution to the fine level* $k$*:*
   $$\tilde{y}_k^{(2)} = y_{k+1}, \qquad \tilde{y}_k^{(1)} = z_1 - B_{11}^{(k)} A_{12}^{(k)} \tilde{y}_k^{(2)} \; i.e.$$
   $$\tilde{y}_k = \begin{bmatrix} \tilde{y}_k^{(1)} \\ \tilde{y}_k^{(2)} \end{bmatrix} = \begin{bmatrix} \tilde{y}_k^{(1)} \\ y_{k+1} \end{bmatrix}.$$
9. *Apply one level inverse DWT to* $\tilde{y}_k$ *to obtain* $y_k$.
10. *When* $k = 0$ *(on the finest level), check the residual error — if small enough accept the solution* $y_0$ *and stop the algorithm.*
    *When* $k > 1$*, check if* $\nu$ *cycles have been carried out; if not, find the residual vector and return to Step 2 otherwise continue with Step 7 on level* $k$.

**Remark 3** *It turns out that this algorithm is similar to the algebraic multi-level iteration methods (AMLI) that was developed for a class of symmetric positive definite finite element equations in a hierarchical basis [4, 2, 3, 36]. In fact, let* $\nu_k = \nu$ *and then* $S^{(k)}$ *is implicitly defined by the following*

$$S^{(k)} = A^{(k+1)} \left[ I - P_\nu \left( M^{(k+1)^{-1}} A^{(k+1)} \right) \right]^{-1},$$

*where* $P_\nu$ *is a degree* $\nu$ *polynomial satisfying*

$$0 \le P_\nu(t) < 1, \quad , 0 < t \le 1, \quad P_\nu(0) = 1.$$

*As with AMLI, for* $\nu = 1$*, the valid choice* $P_1(t) = 1 - t$ *implies* $S^{(k)} = M^{(k+1)}$ *and gives rise to the V-cycling pattern and For* $\nu > 1$*, the polynomial* $P_\nu(t)$ *is chosen to improve the preconditioner;*

*ideally* $\kappa\left(M^{(k)^{-1}}A_{11}^{(k)}\right) \approx O(1)$ *asymptotically. Refer to these original papers about how to work out the coefficients of* $P_\nu(t)$ *based on eigenvalue estimates. However we do not use any eigenvalue estimates to construct* $P_\nu$.

*We also remark that an alternative definition of a recursive preconditioner different from AMLI is the ILUM method as mentioned in Remark 2, where in a purely algebraic way (using independent sets of the underlying matrix graph)* $A_{11}^{(k)}$ *is defined as a block diagonal form after a suitable permutation. This would give rise to another way of approximating the true Schur complement* $\mathbf{S}^{(k)} = A_{22} - A_{21}^{(k)}A_{11}^{(k)^{-1}}A_{12}^{(k)}$. *However the sparsity structures of blocks* $A_{21}^{(k)}$ *and* $A_{12}^{(k)}$ *will affect the density of nonzeros in matrix* $\mathbf{S}^{(k)}$ *and an incomplete LU decomposition has to be pursued as in [28, 12, 33].*

# 5 Complexity analysis

Here we mainly compare the complexity of Algorithms 1-2 in a full cycle. Note that both algorithms can be used by a main driving iterative solver where each step of iteration will require $n^2$ flops (one flop refers to 1 multiplication and 1 addition) unless some fast matrix vector multiplication methods are used. One way to reduce this flop count is to use a small threshold so that only a sparse form of $\widetilde{A}$ is stored. Also common to both algorithms are the DWT steps which are not counted here.

To work out flop counts for differing steps of the two algorithms, we list the main steps as follows

| | Algorithm 1 | Algorithm 2 |
|---|---|---|
| Set-up | | Approximate inverses: $\mu^2 n_i^2$ |
| Main step | 3 band solves: $3n_i\mu^2$<br>4 band-vector multiplications: $8n_i\mu$<br>3 off-band-vector multiplications ($R_i$): $3(n_i - 2\mu)n_i$ | 2 band-band multiplications: $8n_i\mu^2$<br>4 band-vector multiplications: $8n_i\mu$ |

Therefore an $\nu$-cycling (iteration) across all levels would require these flops (assuming $\nu \leq 3$)

$$F_I = \sum_{i=1}^{\ell-1} \frac{6n^2\nu^i}{2^{2i}} + \frac{3n\mu^2}{2^i} + \frac{2n\mu}{2^i} \approx \frac{6\nu}{4-\nu}n^2,$$

and

$$F_{II} = \sum_{i=1}^{\ell-1} \frac{n^2}{2^{2i}}\mu^2 + 8\sum_{i=1}^{\ell-1} \frac{n\mu^2\nu^2}{2^i} + \frac{n\mu}{2^i} \approx \frac{\mu^2}{3}n^2,$$

after ignoring the low order terms. Therefore we obtain $F_{II}/F_I = \frac{(4-\nu)\mu^2}{9\nu}$; for a typical situation with $\mu = 10$ and $\nu = 2$, $F_{II}/F_I \approx 10$. Thus we expect Algorithm I to be cheaper than II if the same number of iteration steps are recorded. Here by Algorithm I we meant the use of a Richardson iteration (in SOLVE of Algorithm 1); however if a GMRES iteration is used for preconditioning then the flop count will increase. Of course as is well known, flop count is not always a reliable indicator for execution speed; especially if parallel computing is desired a lot of other factors have to be considered.

**Remark 4** *For sparse matrices, all flop counts will be much less as DWT matrices are also sparse. The above complexity analysis is done for a dense matrix case. Even in this case, setting up preconditioners only adds a few equivalent iteration steps to a conventional iteration solver. One can usually observe overall speed-up. For integral operators, the proper implementation is to use the biorthorgonal wavelets as trial functions to yield sparse matrices* $A_i$, $B_i$, $C_i$ *directly (for a suitable threshold); in this case a different complexity analysis is needed as all matrices are sparse and experiments have shown that although this approach is optimal a much larger complexity constant*

*(independent of n) is involved. For structured dense matrices where FFT is effective, wavelets may have to be applied implicitly to preserve FFT representation. Further work is in progress.*

# 6 Numerical experiments

Here we shall compare the new algorithms with two previous methods: the WSPAI method by Chan-Tang-Wan [14] (CTW) and the two stage method by Chan-Chen [13] (CC). Further comparisons with other methods such as SPAI and ILU type can be found in [14] and [39]. Note that WSPAI, where applicable, is faster that SPAI and ILU.

We now present numerical experiments from *two* sets of problems solved by these 4 methods:

- M1 — Algorithm 1 with the SOLVE step replaced by a Richardson iteration method for $\nu$ steps on each level.
- M2 — Algorithm 1 with the SOLVE step replaced by a GMRES iteration method on each level; in particular GMRES(25) for the main finest level iteration and GMRES($\nu$) for coarser levels.
- M3 — Algorithm 1 with the SOLVE step replaced by a GMRES(25) iteration method on the finest level (outer iteration method) and a Richardson iteration method for $\nu$ steps on coarser levels.
- M4 — Algorithm 2 with the main iteration method being a GMRES(25)iteration method on the finest level and $\nu$ steps of residual correction on all levels for preconditioning.

The test problems are the following:

- Set 1:
  - Example 1. Symmetric case [10]: $$A_{ij} = \begin{cases} \frac{1}{|i-j|} & i \neq j, \\ 2 & i = j, \end{cases}$$
  - Example 2. Symmetric case ($L = 2^{n-1}$): $$A_{ij} = \begin{cases} \frac{\log|i-L| - \log|j-L|}{i-j} & i \neq j; i \neq L; j \neq L \\ 6 & \text{otherwise} \end{cases}$$
- Set 2:
  - Example 3. Unsymmetric case: $$A_{ij} = \begin{cases} \frac{1}{i-j} & i \neq j, \\ 2 & i = j, \end{cases}$$
  - Example 4. An anisotropic PDE problem in both $x$ and $y$ directions:

$$a(x,y)u_{xx} + b(x,y)u_{yy} = 1,$$

where the coefficients are defined as ([39, 13, Ch.5])

$$a(x,y) = \begin{cases} 100 & (x,y) \in [0,0.5] \times [0,0.5] \text{ or } [0.5,1] \times [0.5,1] \\ 1 & \text{otherwise;} \end{cases}$$

$$b(x,y) = \begin{cases} 100 & (x,y) \in [0,0.5] \times [0.5,1] \text{ or } [0.5,1] \times [0,0.5] \\ 1 & \text{otherwise.} \end{cases}$$

Table 1: Number of iteration steps to reach $TOL = 10^{-6}$ for Example 1 (set 1). Here $\nu$ is for cycling pattern, $n$ is the problem size and $\ell$ denotes the number of levels used.

| Method used | Size $n$ | Levels $\ell$ | Case of $\nu = 1$ Steps | Case of $\nu = 2$ Steps |
|---|---|---|---|---|
| M1 | 128 | 3 | 6 | 6 |
| | 256 | 4 | 6 | 6 |
| | 512 | 5 | 6 | 5 |
| | 1024 | 6 | 6 | 5 |
| M2 | 128 | 3 | 15 | 9 |
| | 256 | 4 | 13 | 6 |
| | 512 | 5 | 11 | 6 |
| | 1024 | 6 | 10 | 5 |
| M3 | 128 | 3 | 5 | 5 |
| | 256 | 4 | 5 | 5 |
| | 512 | 5 | 5 | 5 |
| | 1024 | 6 | 5 | 5 |
| M4 | 128 | 3 | 8 | 4 |
| | 256 | 4 | 10 | 4 |
| | 512 | 5 | 11 | 4 |
| | 1024 | 6 | 12 | 4 |

– Example 5. A discontinuous coefficient PDE problem as tested in [14, 39, 13]:

$$(a(x,y)u_x)_x + (b(x,y)u_y)_y + u_x + u_y = \sin(\pi xy),$$

where the coefficients are defined as ([39, Ch.3])

$$a(x,y) = b(x,y) = \begin{cases} 10^{-3} & (x,y) \in [0,0.5] \times [0.5,1] \\ 10^{3} & (x,y) \in [0.5,1] \times [0,0.5] \\ 1 & \text{otherwise.} \end{cases}$$

Here set 1 problems fall into the class for which our algorithms are expected to work; we mainly test the dependence of the variants of Algorithms 1-2 on the parameter choices. This is because for symmetric positive definite (SPD) matrices, their principal submatrices are also SPD and invertible so the assumptions of Lemma 1 (Appendix) are satisfied. Set 2 problems, although outside the class of problems that we have provided an analysis, are solvable by previously studied wavelets preconditioners so we include these for comparison purpose. For all cases, we stop an iterative method when the residual in the 2-norm is reduced by $TOL = 10^{-6}$. The coarsest level is fixed at $n_\ell = 16$.

Tables 1-2 display the numerical results for different problem sizes $n = n_0$ from solving set 1 problems, where 'Steps' means the number of iteration steps required. Clearly one can observe that 'Steps' is approximately independent of the problem size which is usually expected of a good preconditioner.

For set 2 problems, we do not expect M1 – M4 to work. To extend these methods, we propose to combine one step of Stage-1 preconditioning as proposed in [13] to smooth out the given matrix (w.r.t. level 0 to 1). Specifically we select a diagonal matrix $D = diag(d_j)$ such that the sum of diagonal entries of $B_1$ and $C_1$ in

$$W_0 D^{-1} A W_0^\top = \begin{bmatrix} A_1 & B_1 \\ C_1 & T_1 \end{bmatrix},$$

is minimised. Then our algorithms can be applied to the new linear system $D^{-1}Ax = D^{-1}b$. This points to one way of possible further work. Other approaches similar to stage 1 preconditioning

Table 2: Number of iteration steps to reach $TOL = 10^{-6}$ for Example 2 (set 1). Here $\nu$ is for cycling pattern, $n$ is the problem size and $\ell$ denotes the number of levels used.

| Method used | Size $n$ | Levels $\ell$ | Case of $\nu = 1$ Steps | Case of $\nu = 2$ Steps |
|---|---|---|---|---|
| M1 | 128 | 3 | 7 | 6 |
| | 256 | 4 | 10 | 7 |
| | 512 | 5 | 21 | 10 |
| | 1024 | 6 | 49 | 43 |
| M2 | 128 | 3 | 11 | 11 |
| | 256 | 4 | 16 | 16 |
| | 512 | 5 | 23 | 23 |
| | 1024 | 6 | 32 | 32 |
| M3 | 128 | 3 | 3 | 3 |
| | 256 | 4 | 4 | 4 |
| | 512 | 5 | 5 | 5 |
| | 1024 | 6 | 6 | 6 |
| M4 | 128 | 3 | 3 | 3 |
| | 256 | 4 | 4 | 4 |
| | 512 | 5 | 5 | 5 |
| | 1024 | 6 | 6 | 6 |

may be considered e.g. for some indefinite problems we have found that the permutation idea by Duff [21] can be combined with our algorithms here; the idea was recently used in [9] to extend the applicability of sparse approximate inverse preconditioners.

In Table 3, we use 'Stage-1' to indicate if such a step has been carried out; whenever such an one-off stage 1 diagonal preconditioning is used, we put "Yes" in this column. The symbol '*' denotes a case where no convergence has been achieved after 100 steps. Clearly the simple idea of Stage-1 preconditioning does improve the performance of M1–M4 except for M1 (although M2, M4 appear to be more robust than M3). Therefore we shall only compare M2–M4 with the work of CTW [14] and CC [13] next.

Finally in Table 4, we show results from solving Examples 4 and 5 where the CPU seconds are obtained from a Sun Ultra-2 workstation (using Matlab 5) and the other notation is the same as in Table 3. Here 'diag' refers to the diagonal preconditioner which does not work for these two examples and data with a little 'f' in column 5 are used in constructing Figs.6 and 7. Table 4 demonstrates that when combined with stage-1 preconditioning, the new algorithms can out perform the previous methods. In particular, it appears that M3 is the fastest method in the table. To compare the residuals and CPU time of M3 (the best case), CTW [14] and CC [13] in solving Examples 4 and 5, we plot respectively in Fig.6 and Fig.7 the convergence history and CPU time (all data are taken from Table 4), where one can observe that M3 (New I) out performs the others. This again confirms that the new algorithm M3 converges the fastest. As remarked already, comparisons with other non-wavelets preconditioners (e.g. SPAI and ILU) can be found in [14] and [39] where it was concluded that WSPAI is faster.

As also remarked earlier, the proposed multi-level algorithms can be potentially developed much further by incorporating the ideas from [6, 13, 9, 21]. More importantly as they are closely based on wavelets compression theory, generalizations to multi-dimensions appear to be more straightforward than similar and known wavelet preconditioners; these aspects are currently being investigated.

Table 3: Number of iteration steps to reach $TOL = 10^{-6}$ for Example 3 (set 2). Note that this example is outside the scope of $M1 - M4$. Here $\nu$ is for cycling pattern, $n$ is the problem size and $\ell$ denotes the number of levels used.

| Method used | With Stage-1 | Size $n$ | Levels $\ell$ | Case of $\nu = 1$ Steps | Case of $\nu = 2$ Steps |
|---|---|---|---|---|---|
| M1 | No | 512 | 5 | * | * |
| | | 1024 | 6 | * | * |
| | Yes | 512 | 5 | * | * |
| | | 1024 | 6 | * | * |
| M2 | No | 512 | 5 | 82 | 12 |
| | | 1024 | 6 | 96 | 12 |
| | Yes | 512 | 5 | 82 | 12 |
| | | 1024 | 6 | 96 | 12 |
| M3 | No | 512 | 5 | * | * |
| | | 1024 | 6 | * | * |
| | Yes | 512 | 5 | 18 | 13 |
| | | 1024 | 6 | 20 | 14 |
| M4 | No | 512 | 5 | 19 | 8 |
| | | 1024 | 6 | 22 | 10 |
| | Yes | 512 | 5 | 13 | 5 |
| | | 1024 | 6 | 14 | 5 |

Table 4: Comparison of the new algorithms with previous work for Examples $4 - 5$ (set 2). Data indicated by 'f' in column 5 are used in Figs.6-7.

| Problem | Stage-1 | Method | $\nu$ | Convergence Steps | | CPU |
|---|---|---|---|---|---|---|
| Example 4 | No | M2 | 2 | 35 | | 411 |
| | | M3 | 2 | 59 | | 195 |
| | | M4 | 2 | * | | * |
| | | CTW [14] | | 63 | f | 297 |
| | | Diag | | * | | * |
| | Yes | M2 | 2 | 42 | | 505 |
| | | M3 | 1 | 31 | f | 54 |
| | | | 2 | 17 | | 69 |
| | | M4 | 1 | 81 | | 381 |
| | | | 2 | 81 | | 383 |
| | | CC [13] | | 64 | f | 347 |
| Example 5 | No | M2 | 2 | * | | * |
| | | M3 | 2 | * | | * |
| | | M4 | 2 | * | | * |
| | | CTW [14] | | * | f | * |
| | | Diag | | * | | * |
| | Yes | M2 | 2 | 29 | | 348 |
| | | M3 | 1 | 20 | f | 40 |
| | | | 2 | 11 | | 49 |
| | | M4 | 1 | 21 | | 115 |
| | | | 2 | 21 | | 116 |
| | | CC [13] | | 54 | f | 302 |

# 7 Conclusions

This paper has presented two related algorithms implementing an algebraic wavelet preconditioner. The first one is similar to the set up of a NS form representation of a wavelet basis while the second resembles the AMLI preconditioner designed for finite elements. Both algorithms are observed to give excellent performance for a class of symmetric positive definite Calderon-Zygmund and pseudo-differential operators. Combined with a minor stage 1 preconditioning step, they are immediately applicable to problems outside this class of problems. We note that there are several methods that are designed to deal with anisotropic and highly indefinite elliptic problems [6, 9, 21] and alternative methods of constructing a sparse preconditioner [33, 12]; these should be investigated in the near future to further extend the multi-level preconditioner to an even wider class of problems.

# 8 Acknowledgements

This work is partially supported by grants NSF ACR 97-20257, NASA Ames NAG2-1238, Sandia Lab LG-4440 and UK EPSRC GR/R22315. The second author wishes to thank the Department of Mathematics, UCLA for its hospitality during his visits conducting part of this work.

# References

[1] K. Amaratunga (2000), *A wavelet-based approach for the compression of kernel data in large scale simulations of 3D integral problems*, IEEE Comput. Sci. Engng., 2, pp.34-45.

[2] O. Axelsson and P. S. Vassilevski (1989), *Algebraic multilevel preconditioning methods I*, Numer. Math., 56, pp.157-177.

[3] O. Axelsson and P. S. Vassilevski (1990), *Algebraic multilevel preconditioning methods II*, SIAM J. Numer. Anal., 27 (6), pp.1569-1590.

[4] O. Axelsson and M. Neytcheva (1994), *Algebraic multilevel iteration method for Stieljes matrices*, Numer. Lin. Alg. Appl., 1 (1), pp.213-236.

[5] O. Axelsson and M. Neytcheva (1994), *The algebraic multilevel iteration methods — theory and applications*, preprint.

[6] O. Axelsson and A. Padiy (1999), *On the additive version of the algebraic multilevel iteration method for anisotropic elliptic problems*, SIAM J. Sci. Comput., 20 (5), pp.1807-1830.

[7] R. E. Bank (1996), *Hierarchical bases and the finite element method*, Acta Numerica, pp.1-43.

[8] M. W. Benson and P. O. Frederickson P. O. (1982), *Iterative solution of large sparse linear systems arising in certain multidimensional approximation problems*, Utilitas Math., 22, pp.127-140.

[9] M. Benzi, J. C. Haws and M. Tuma (2000), *Preconditioning highly indefinite and nonsymmetric matrices*, SIAM J. Sci. Comput., 22, pp.1333-1353.

[10] G. Beylkin, R. Coifman and V. Rokhlin (1991), *Fast wavelet transforms and numerical algorithms I*, Commu. Pure Appl. Math., 44, pp.141-183.

[11] D. Bond and S. Vavasis (1994), *Fast wavelet transforms for matrices arising from boundary element methods*, Computer Science Research Report TR-174, Cornell University, USA.

[12] E. F. F. Botta and F. W. Wubs (1999), *Matrix renumbering ILU: An effective algebraic multilevel ILU preconditioner for sparse matrices*, SIAM J. Matrix Anal. Appl., 20 (4), pp.1007-1026.

[13] T. F. Chan ad K. Chen (2000), *Two-stage preconditioners using wavelet band splitting and sparse approximation*, CAM report 00-26, Dept of Maths, UCLA, USA.

[14] T. F. Chan, W. P. Tang and W. L. Wan (1997), *Wavelet sparse approximate inverse preconditioners*, BIT, 37, pp.644-660.

[15] K. Chen (1999), *Discrete wavelet transforms accelerated sparse preconditioners for dense boundary element systems*, Elec. Trans. Numer. Anal., 8, pp.138-153.

[16] K. Chen (2001), *An analysis of sparse approximate inverse preconditioners for boundary integral equations*, SIAM J. Matr. Anal. Appl., 22(3), pp.1058-1078.

[17] A. Cohen, W. Dahmen and R. DeVore (2001), *Adaptive wavelet methods for elliptic operator equations – convergence rates*, Math. Comp., 70 (233), pp.27-75.

[18] A. Cohen and R. Masson (1999), *Wavelet methods for second-order elliptic problems, preconditioning, and adaptivity*, SIAM. J. Sci. Comput., 21, pp.1006-1026.

[19] A. Cohen and R. Masson (2000), *Wavelet adaptive method for second order elliptic problems: boundary conditions and domain decomposition*, Numer. Math., 86, pp.193-238.

[20] W. Dahmen (1997), *Wavelet and multiscale methods for operator equations*, Acta Numerica, 6, 55-228 (Cambridge University Press, UK).

[21] I. S. Duff and J. Koster (1999), *The design and use of algorithms for permuting large entries to the diagonal of sparse matrices*, SIAM J. Matr. Anal. Appl., 20, pp.889-901. (also RAL-TR-1999-030, http://www.numerical.rl.ac.uk/reports/)

[22] D. Gines, G. Beylkin, and J. Dunn (1998), *LU factorization of non-standard forms and direct multiresolution solvers*, Applied and Computational Harmonic Analysis, 5, pp.156-201.

[23] A. Harten (1994), *Multiresolution representation and numerical algorithms: a brief review*, CAM report 94-12, Dept of Maths, UCLA, USA.

[24] L. Y. Kolotilina and A. Y. Yeremin (1986), *On a family of two-level preconditionings of the incomplete block factorization type*, Soviet J. Numer. Anal. Math. Model., 1, pp.293-320.

[25] J. Kovačević and W. Sweldens (2000), *Wavelet Families of Increasing Order in Arbitrary Dimensions*, IEEE Trans. Image Proc., 9, (3), pp.480-496.

[26] M. Neytcheva (1998), *Algebraic multilevel iteration preconditioning technique — a Matlab implementation*, http://www.netlib.org

[27] M. Neytcheva and P. S. Vassilevski (1998), *Preconditioning of indefinite and almost singular finite element elliptic equations*, SIAM J. Sci. Comput., 19 (5), pp.1471-1485.

[28] Y. Saad (1996), *ILUM: a multi-elimination ILU preconditioner for general sparse matrices*, SIAM J. Sci. Comput., 17 (4), pp.830-847.

[29] Y. Saad (1996), *Iterative Methods for Sparse Linear Systems*, PWS Publishing, Boston.

[30] Y. Saad and M. H. Schultz (1986), *GMRES: a generalized minimal residual algorithm for solving unsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (3), pp.856-869.

[31] Y. Saad and H. A. van der Vorst (2000), *Iterative solution of linear systems in the 20th century*, J. Comput. Appl. Math., 123 (1-2), pp.1-33.

[32] Y. Saad and J. Zhang (1999), *BILUM: Block versions of multielimination and multilevel ILU preconditioner for general sparse linear systems*, SIAM J. Sci. Comput., 20 (6), pp.2103-2121.

[33] Y. Saad and J. Zhang (2001), *Enhanced multi-level block ILU preconditioning strategies for general sparse linear systems*, J. Comput. Appl. Math., 130 (1-2), pp.99-118. See also http://www.cs.engr.uky.edu/ jzhang/pub/PREPRINT/enhanced.ps.gz (1998).

[34] G. Strang and T. Nguyen (1996), *Wavelets and Filter Banks*, Wellesley-Cambridge Press, Wellesley, MA, USA.

[35] W. Sweldens (1998), *The lifting scheme: a construction of second generation of wavelets*, SIAM J. Math. Anal., 29 (2), pp.511-546.

[36] P. S. Vassilevski (1997), *On two ways of stabilizing the hierarchical basis multilevel methods*, SIAM Rev., 39 (1), pp.18-53.

[37] P. S. Vassilevski (1989), *Nearly optimal iterative methods for solving finite element elliptic equations based on the multilevel splitting of the matrix*, Report #1989-09, ISC, University of Wyoming, WY, USA (see also Math. Comp., 58 (1992), pp.489-512).

[38] H. M. Zhou (2000), *Wavelet Transforms and PDE Techniques in Image Compression*, Ph.D. Thesis, CAM report 00-21, Dept of Mathematics, UCLA, USA.

[39] W. L. Wan (1998), *Scalable and multilevel iterative methods*, Ph.D thesis, CAM report 98-29, Dept of Mathematics, UCLA, USA.

[40] H. Yserentant (1986), *On the multi-level splitting of finite element spaces*, Numer. Math., 49, pp.379-412.

# Appendix: An analysis of preconditioner I

We now show that Algorithm 1 does not break down for a class of Calderon-Zygmund and pseudo-differential operators [10], or precisely that $\overline{A}_i = \mathbf{B}_\mu(A_i)$ is always invertible. More importantly the bandwidth $\mu$ can be very small, independent of the problem size ($n$).

For our analysis purpose, we assume that all principal submatrices $A_i$ (resulting from matrix $A_0 = A$) are non-singular and diagonalizable. Of course this is a much weaker assumption than requiring matrix $A$ being symmetric positive definite. For nonnegative integers of $\mu$ and $m$, define the function

$$b(n, m, \mu) = \sum_{k=1}^{n-\mu-1} \frac{2}{(\mu + k)^{m+1}},$$

which is understood in the usual summation convention i.e. if $\mu \geq n-1$, $b(n, m, \mu) = 0$. We can verify the simple properties: $b(n, m, \mu) \leq (\mu+1)^{1-m} b(n, 1, \mu)$ if $m \geq 1$, $b(n, m, \mu) < b(\infty, m, \mu)$, and

$$b(n, 1, \mu) < b(\infty, 1, \mu) \leq b(\infty, 1, 0) = \sum_{k=1}^{\infty} 2/k^2 = \pi/3.$$

Therefore if $m > 1$, $\lim_{\mu\to\infty} b(\infty, m, \mu) = 0$ as $\lim_{\mu\to\infty}(\mu+1)^{1-m} = 0$. Then the following result holds.

**Lemma 1** *Let matrix $O_i = A_i - \overline{A}_i$ with $\overline{A}_i$ from Algorithm 1 and $c_{m,i} \leq c$ in (5) for a generic constant $c$ that does not depend on $m$, $n, \mu$. Then*

$$\|O_i\|_\infty \leq cb(\tau_i, m, \mu) < cb(\infty, m, \mu).$$

*Proof.* It suffices to consider the case of $i = 1$ i.e. $\overline{A}_1$. Write $A_1 = (a_{kj})$ for simplicity. Since $\overline{A}_1 = \mathbf{B}_\mu(A_1)$ is banded and of dimension $\tau_1$, to estimate the $\infty$-norm of $O_1$, take any row index $k \in [1,\ \tau_1]$ with $\mu \geq 2m$ (note that for the first and last $\mu$ rows there is only one nonzero sum at the right hand side of the following line 1 as $O_1$ is an off band matrix)

$$\begin{aligned}
&\sum_{j=1}^{\tau_1} |(O_1)_{kj}| = \sum_{j=1}^{k-\mu-1} |A_{kj}| + \sum_{j=k+\mu+1}^{\tau_1} |A_{kj}| \\
\leq\ & c_{m,1}\left[\sum_{j=1}^{k-\mu-1} \frac{1}{1+|k-j|^{m+1}} + \sum_{j=k+\mu+1}^{\tau_1} \frac{1}{1+|k-j|^{m+1}}\right] = c_{m,1}\left[\sum_{j=\mu+1}^{k-1} \frac{1}{1+j^{m+1}} + \sum_{j=\mu+1}^{\tau_1-k} \frac{1}{1+j^{m+1}}\right] \\
\leq\ & c_{m,1}\left[\sum_{j=1}^{k-\mu-1} \frac{1}{(\mu+j)^{m+1}} + \sum_{j=1}^{\tau_1-\mu-k} \frac{1}{(\mu+j)^{m+1}}\right] \leq c_{m,1}\left[\sum_{j=1}^{\tau_1-\mu-1} \frac{1}{(\mu+j)^{m+1}} + \sum_{j=1}^{\tau_1-\mu-1} \frac{1}{(\mu+j)^{m+1}}\right] \\
\leq\ & cb(\tau_1, m, \mu).
\end{aligned}$$

Therefore

$$\|O_1\|_\infty = \max_{1\leq k\leq \tau_1} \sum_{j=1}^{\tau_1} |(O_1)_{kj}| \leq cb(\tau_1, m, \mu).$$

Using the properties of function $b$ completes the proof. ■

Note that this Lemma suggests $\|O_i\|_\infty$ can be arbitrarily small if $\mu$ is large. However it is more useful to quantify how large $\mu$ needs to be in order for $\overline{A}_i$ be invertible. This is now stated as follows.

**Theorem 1** *Under the above assumption of $A_i$, the band matrix $\overline{A}_i$ (used in the preconditioning) is invertible if $m \geq 2$ (i.e. there are 2 or more vanishing moments) and the semi-bandwidth $\mu$ of $\overline{A}_i$ satisfies $\mu \geq \mu_{\min}$ where*

$$\mu_{\min} = \left(\frac{c\pi}{3|\lambda_i^s|}\right)^{\frac{1}{m-1}} - 1$$

*and $\lambda_i^s$ denotes the smallest eigenvalue of $A_i$ (in modulus).*

**Remark 5** *The theorem implies that Algorithm 1 does not break down. The confirmation in Lemma 1 of a fast decay in elements of $O_i$ or fast deduction of $\|O_i\|_\infty$ (as $\mu$ is large) ensures that a small bandwidth $\mu$ is practically sufficient; this justifies the efficient approximation of $\widetilde{T}_i$ by $\overline{T}_i$ in Algorithm 1. For instance, when $c = 20$ and $\lambda_i^s = 0.1$, $\mu_{\min} = 2.8$ if $m = 5$ and $\mu_{\min} = 13.5$ if $m = 3$. Here the requirement of $m \geq 2$ (number of vanishing moments) is not restrictive. Also as remarked before, the assumption of non-singularity and diagonalization is assured if $A_0 = A$ is SPD [4, 36].*

*Proof.* Following Lemma 1, let $\lambda(A_i)$ and $\lambda(\overline{A}_i)$ denote an eigenvalue of the matrix $A_i$ and $\overline{A}_i$ respectively. From the eigenvalue perturbation theorem (for a diagonalizable matrix),

$$\left|\lambda(A_i) - \lambda(\overline{A}_i)\right| \leq \|O_i\|_\infty.$$

Using the above inequality with Lemma 1 yields the sufficient condition

$$(\mu+1)^{m-1}\frac{\pi}{3}c \leq |\lambda_i^s|$$

and further the required result for $\mu_{\min}$. With such a $\mu \geq \mu_{\min}$, the preconditioning matrix $\overline{A}_i$ has only non-zeros eigenvalues and hence is invertible. ■

Figure 5: Flow chart of Algorithm II (Implementation of the algebraic multi-level iteration preconditioner). Here on a typical fine level $k$ (starting from the finest level 0), we illustrate on the top half of the diagram the process of restricting to the coarse level $k+1$ (up to the coarsest level $\ell$) whilst on a typical coarse level $k$ (starting from $\ell$) we show on the bottom half of the diagram the process of interpolating to the fine level $k-1$ (up to the finest level 0). Consult Fig.4 for the overall cycling pattern.

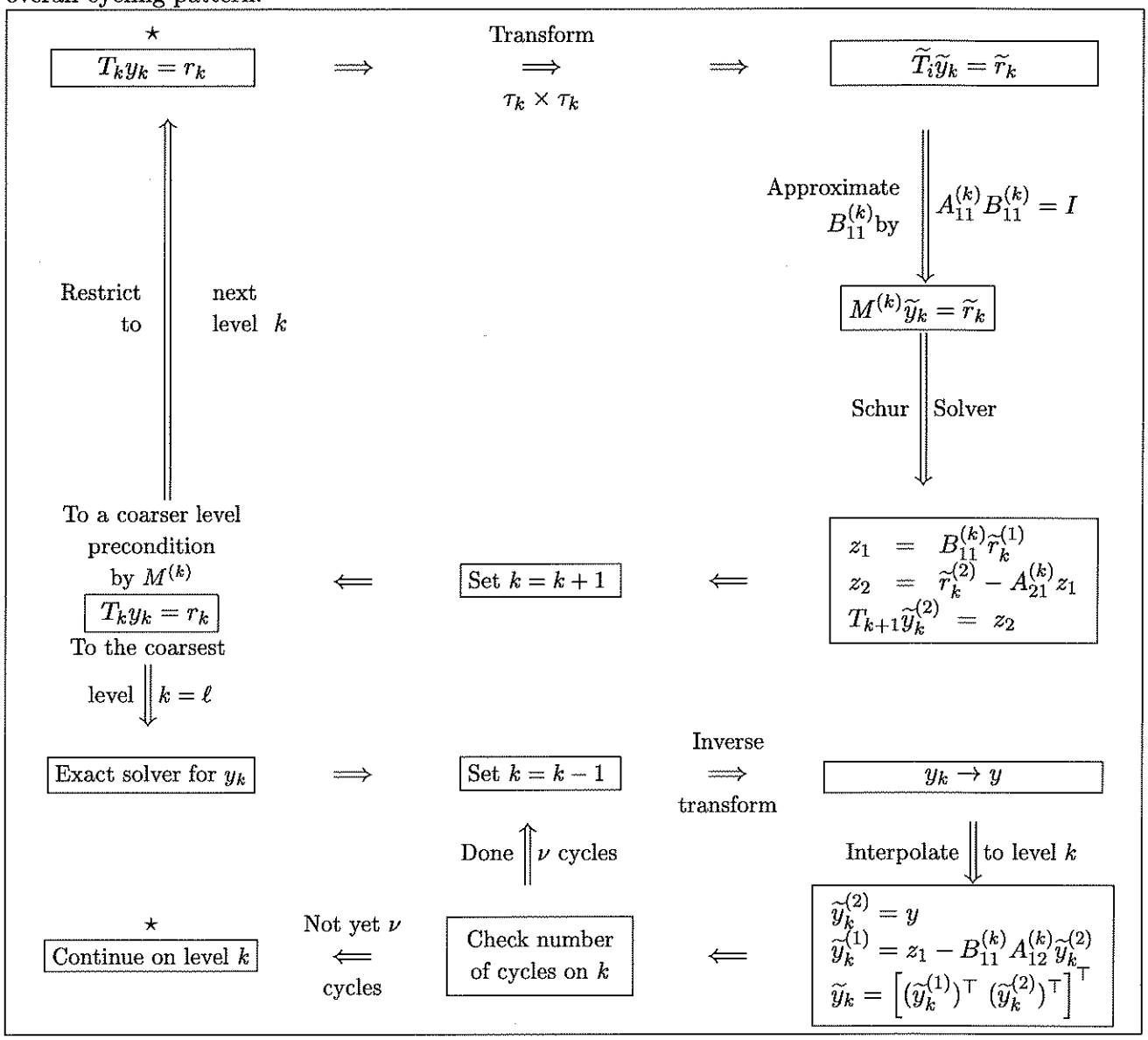


Figure 6: Comparison of convergence behaviour of three methods for Example 4 to a relative residual error below $TOL = 10^{-6}$: New I (Algorithm 1, M3), CTW [14] (WSPAI) and CC [13] (the two-stage preconditioner). The left plot compares the iteration steps while the right the CPU time; clearly New I requires less CPU as well as less iteration steps.

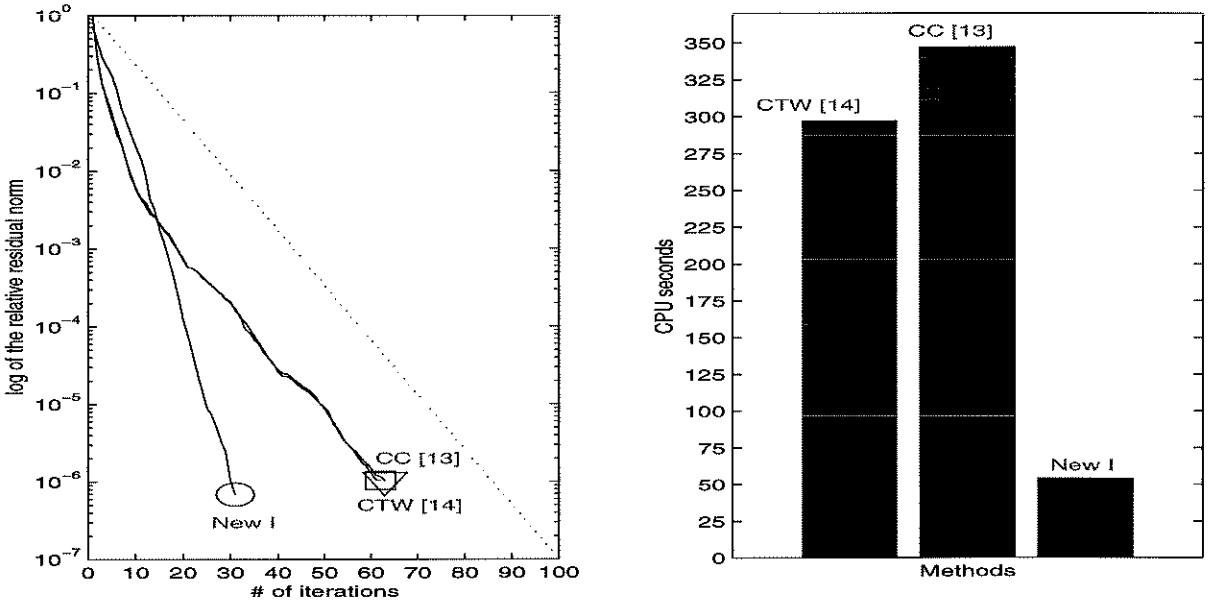



Figure 7: Comparison of convergence behaviour of three methods for Example 5 to a relative residual error below $TOL = 10^{-6}$: New I (Algorithm 1, M3), CTW [14] (WSPAI) and CC [13] (the two-stage preconditioner). The left plot compares the iteration steps while the right the CPU time; again New I requires less CPU as well as less iteration steps. Note the CTW [14] (WSPAI) method does not lead to convergence and is thus given a CPU count of $\infty$.