Adaptive Solution of Multidimensional PDEs via Tensor Product Wavelet Decomposition

A. Averbuch[1], G. Beylkin[2], R. Coifman[3], P. Fischer[4*], M. Israeli[5]
[1]School of Mathematical Sciences, Tel Aviv University, Tel Aviv 69978 Israel
[2]Program in Applied Mathematics, University of Colorado at Boulder, Boulder CO 80309-0526, USA
[3] Department of Mathematics, Yale University, P.O.Box 2155 Yale Station New Haven, CT 06520, USA
[4] Mathematiques Appliquees de Bordeaux, Universite de Bordeaux 1, France
[5] Faculty of Computer Science, Technion- Israel Institute of Technology Haifa 32000, Israel

Abstract

In this paper we describe efficient adaptive discretization and solution of elliptic PDEs which are forced by right hand side (r.h.s) with regions of smooth (non-oscillatory) behavior and possibly localized regions with non-smooth structures. Classical discretization methods lead to dense representations for most operators. The method described in this paper is based on the wavelet transform which provides sparse representations of operator kernels. In addition, the wavelet basis allows for automatic adaptation (using thresholding) in the sense that only a few coefficients are needed to describe smooth sections of the r.h.s. while more coefficients are needed to describe sharp transitions and singular points.

In this work we develop adaptive algorithms, i.e. algorithms such that the number of operations performed is proportional to the number of significant coefficients in the wavelet expansion of the "inputs" of a given differential equation problem. We adapt an iterative approach thus we can succeed if we must do only a fixed number of iterations where each iteration requires a fixed number of operations, independent of the resolution (but dependent on the chosen accuracy).

The basic tool in our approach is the preconditioned conjugate gradient (CG) iteration in a "constrained" form. In the wavelet basis diagonal preconditioners are available which render the condition number of elliptic operators to $O(1)$. This implies that the number of iterations of the CG method does not depend on the size of the

*Receipint of the 1995-1996 Israeli Academy of Sciences Post Doctoral Fellowship for research at the Computer Science Department, Tel Aviv University, Israel.

problem. Each iteration consists of applying the non-standard-form of an operator to the wavelet expansion of a function, this translates to a multiplication of a sparse vector by a sparse matrix. The "constraint" is applied in a form of a mask such that only elements of the vector in the mask are used in the computation while other elements are ignored. We present implementations in one, two and three dimensions using sparse data structures to take advantage of the algorithm.

1 Introduction

Efficient discrete representation of continuous operators is a basic problem in the numerical solution of differential and integral equations. Classical discretization methods lead to dense representations (full matrices) for most operators. Sparse representations have the advantage of minimizing storage requirements, while decreasing the computational times. A "good sparse" representation means "fewer coefficients for a given accuracy". Thus, an important step in numerical computations consists of building a sparse representations of common operators and algorithms for using them efficiently.

We seek efficient discretization and solution of partial differential operators which are forced by right hand sides (r.h.s) which consist mostly of smooth (non-oscillatory) regions with possibly localized regions of non-smooth behaviour.

The method described in the present paper is based on the wavelet transform which often provides sparse representations of operator kernels. It consists of expanding functions or operators over a set of basis functions obtained by dilation and translations of an elementary wavelet function localized in both direct and Fourier spaces. Although it is widely known that the wavelet transform leads to more compact representations than the Fourier transform, its current applications for solving PDEs are mostly limited to 1-dimensional problem [1, 2, 4, 5, 7, 17, 18, 19, 20]. Higher dimensional algorithms and software are rare. Some existing applications are concerned with particular 2-dimensional and 3-dimensional cases [11, 22].

An important advantage of wavelet basis representations is that they allow for automatic adaptation (using thresholding) in the sense that only a few coefficients are needed to describe smooth sections of the solution (or right hand side) while many more coefficients are needed for sharp transitions and for singular points. This natural adaptivity or localization, is due to the vanishing moments property of high order wavelets. Spectral and other high order methods can not handle in an efficient way cases with both smooth and oscillatory regions in the computational domain.

The BCR algorithm[7] that was developed for the 1-dimensional case is generalized here to the solution of two and three dimensional partial differential equations whose operators can be written as sums (and products) of 1-dimensional operators. Usually, 1-dimensional kernels in the non-standard form have a matrix (two dimensional) representation, a two dimensional kernel in the non-standard form has a four dimensional representation and a three dimensional kernel has a six dimensional representation. The methods proposed in the present paper can efficiently solve 2 and 3-dimensional equations whose operators can be written as the sum (or the product) of 1-dimensional operators i.e. equations whose operators are separable. Therefore, the 2 and 3-dimensional cases can be implemented via a tensor product of one dimensional operators. As a rule, the present algorithms allow the

treatment of 2-dimensional problems in the form of a 2-dimensional matrix representation and 3-dimensional problems using 3-dimensional arrays. One of the most frequently appearing operators in physics, the Laplacian, $\Delta = \sum_{k=1}^{d} \frac{\partial^2}{\partial x_k^2}, x \in I\!R^d$, will be treated first using the above mentioned method.

Our main goal is to develop adaptive algorithms, i.e. algorithms such that the number of operations performed is proportional to the number of significant coefficients in the wavelet expansion of the "inputs" of the given (partial differential equation) problem. We adapt an iterative approach thus, we can succeed only if we perform only a fixed number of iterations were each iteration requires a fixed number of operations, independent of the resolution (but dependent on the chosen accuracy).

The important feature of the algorithm is that with a minimal computation we find the subspace (i.e. the labels of the significant coefficients) for the wavelet expansion of the solution. Once such subspace is found, we use the diagonal preconditioning within the conjugate gradient (CG) method to find the solution to within specified accuracy. Therefore, the basic tool in our approach is the preconditioned conjugate gradient (PCG) iteration in a "constrained" form. The diagonal preconditioning in the wavelet basis renders the condition number for the class of elliptic operators considered in this paper to $O(1)$. This means in the PCG context that a constant number of iterations is required for a solution to a prescribed accuracy. Each iteration consists of applying the non-standard-form of an operator to the wavelet expansion of a function, this translates to a multiplication of a sparse vector by a sparse matrix. In the case of the non-standard-form the operator is a convolution and is represented by short filters, thus the number of operations is a constant multiple of the number of non vanishing elements in the vector. In addition, we have sparse inner products and sparse multiplication by constants or functions. The "constraint" is applied in a form of a mask such that only elements of the vector in the mask are used in the computation, other elements are not used or even generated (in other words, they are ignored). The mask on the high-pass coefficients is determined by the size of the wavelet coefficients of the right hand side and in the case of non constant coefficients also by coefficients of a solution of certain "related" problems (to be explained below). The high pass filters also determine the mask for the low-pass coefficients. The determination of the mask structure is done once. The implementations use sparse data structures (described in the appendix) in all steps of the algorithm. The mask keeps the scales that contribute to the computation in a fixed size during all the iterations and therefore during the wavelet multiresolution decomposition and reconstruction the scales are not "filled up" beyond the borders of the predetermined masks.

Two physical applications, represented by the elliptic Poisson equation and by the parabolic heat equation, are used as model problems. The solutions are achieved by means of the PCG iteration [21] for the Poisson equation and a first order explicit scheme in time, for the heat equation.

In the wavelet system of coordinates differential operators may be preconditioned by a diagonal matrix (see e.g [5, 23, 13]. The book [28] surveys the methods for multilevel finite element approximation with detailed description on iterative solvers for discretization of elliptic problems (section 4) where the emphasis is on special cases of computationally relevant splittings of Sobolev spaces into multilevel finite element spaces. Sharp estimates are given for multilevel preconditioners. For a specific example where finite elements are

being used see [10].

The algorithm which is proposed in this paper may be efficiently modified to solve non-linear PDEs. We want to track the evolution of high frequency coefficients by following the singularties from paraproduct calculus [26] (chapter 16). Once the singularities are tracked then we can put the masks on these moving singularities. And the solver will operates between the mask boundaries as the proposed algorithm in this paper. We will use the methodology of Coifman and Meyer [26] that later was extended to analysis of non-linear PDEs by Bony [8] to track (follow) the movements of high frequency coefficients. In other words, this enable us to predict where the signularities will move next. Then we know where to place the masks at each iteration. This is a work in progress.

The paper is organized as follows. After the description of the problem that is given in section 2, there is a short description in section 3 of the wavelet background that is needed. The applications of the masks on different subspaces is described in section 4. Section 5 outlines the algorithm. Section 6 describes the 1-D - 3-D preconditioners which are very easy to apply on the standard form. The validity of the application of the mask on the wavelet decomposition is given in section 7. Section 8 estimates the number of operations that the Poisson solver necessitated. Solutions for 1,2 and 3-dimensional problems are described and illustrated by detailed numerical results in section 9. A general purpose software package based on sparse data structures was developed for the implementation of the solver. The data structures that were used for the sparse implementation are described in the appendix.

2 The problem

In this paper we describe adaptive discretization of elliptic PDEs and a method for their solution using wavelet basis. This is a fast adaptive method for solving certain elliptic equations with periodic boundary conditions. It also describes a framework for solving problems with general boundary conditions. Let us consider the partial differential equation

$$\mathcal{L}u = f \quad x \in \mathbf{D} \subset \mathbf{R}^{\mathrm{d}}, \tag{2.1}$$

with the boundary condition

$$\mathcal{B}u|_{\partial \mathbf{D}} = g,$$

where $\mathcal{L}$ is an elliptic operator,

$$\mathcal{L}u = -\sum_{i,j=1,\ldots,d} \left(a_{ij}(x)\ u_{x_i}\right)_{x_j} + b(x)\ u,$$

and $\mathcal{B}$ is the boundary operator,

$$\mathcal{B}u = \alpha u + \beta \frac{\partial u}{\partial N}.$$

Initial 1-D results for the adaptive solution for the case $d = 1$, which utilized the proposed method, were reported in [1, 2]. In this paper we generalize the 1-D algorithm of [1, 2] to the dimensions d = 2, 3 via tensor product, though our considerations are valid for higher dimensions as well.

We generate a function f_{ext}, a smooth extension of f outside the domain $\mathbf{D}$, such that f_{ext} is compactly supported in a rectangular box $\mathbf{B}$, $\mathbf{D} \subset \mathbf{B} \subset \mathbf{R}^{\mathrm{d}}$, and $f = f_{ext}$ for $x \in \mathbf{D}$.

We want to devise adaptive efficient method to solve

$$\mathcal{L}u = f_{ext} \quad x \in \mathbf{B} \tag{2.2}$$

with periodic boundary conditions.

Our goal in solving (2.2) is to develop an adaptive algorithm where the number of operations will be proportional to the number of significant coefficients in the representation of f_{ext}. The usual procedure by current numerical methods to derive the solution requires discretization of the r.h.s. and of the solution in terms of a grid or a basis such that the representations will resolve all features of interest. This might require a large number of grid points or elements not only near the singularities of the functions involved but also in the regions of smooth behavior thus requiring proportionally large number of operations. Current adaptive procedures (for example, adaptive grids or irregular elements) are cumbersome especially in higher dimensions and imply a considerable overhead both for the algorithmic and programming levels.

Our approach is based on using properties for the representations of functions in wavelet bases and allows us to obtain a simple adaptive algorithm.

Let us illustrate it by considering Poisson's equation

$$\Delta u = f \quad x \in \mathbf{B} \tag{2.3}$$

with periodic boundary conditions where (with a slight abuse of notation) we used f instead of f_{ext} to denote the source term. The source term f may have discontinuities in the domain $\mathbf{B}$.

3 Preliminaries on Wavelet Analysis - Theoretical background

In this section we review the relevant material associated with wavelet basis expansions of functions and operators. The wavelet decomposition is based on the notion of multiresolution analysis [24] and the basis functions are compactly supported orthogonal wavelets constructed in [14]. These wavelets lead to band matrices with only few nonzero values around the main diagonal.

Multiresolution analysis (MRA) is a decomposition of a Hilbert space, e.g. $\mathbf{L}^2(\mathbb{R})$, into a chain of closed subspaces

$$\cdots \subset \mathbf{V}_2 \subset \mathbf{V}_1 \subset \mathbf{V}_0 \subset \mathbf{V}_{-1} \subset \mathbf{V}_{-2} \subset \cdots \tag{3.1}$$

that satisfy well established properties [14]. We define an associated sequence of subspaces $\mathbf{W}_j$ as the orthogonal complements of $\mathbf{V}_j$ in $\mathbf{V}_{j-1}$ such that $\mathbf{V}_{j-1} = \mathbf{V}_j \bigoplus \mathbf{W}_j$, and the subspace $\mathbf{V}_j$ can be written as the direct sum of subspaces $\mathbf{W}_{j'}$, i.e. $\mathbf{V}_j = \bigoplus_{j'>j} \mathbf{W}_{j'}$.

The set of dilation and translations of the scaling function $\varphi(\cdot)$, $\{\varphi_{j,k}(x) = 2^{-j/2}\varphi(2^{-j}x - k)\}_{k\in\mathbb{Z}}$, forms an orthonormal axis of $\mathbf{V}_j$ and the set of dilations and translations of the

wavelet $\varphi(\cdot)$, $\psi_{jh,k}(x) = 2^{-j/2}\psi(2^{-j}x - k)\}_{k\in\mathbb{Z}}$, forms an orthonormal basis of $\mathbf{W}_j$. The scaling function $\varphi(x)$ satisfies the two-scale difference equation

$$\varphi(x) = \sqrt{2}\sum_{k=0}^{L-1} h_k\varphi(2x-k) \tag{3.2}$$

and the wavelet $\psi(x)$ is defined by

$$\psi(x) = \sqrt{2}\sum_{k=0}^{L-1} g_k\varphi(2x-k)\ . \tag{3.3}$$

where the sets of coefficients $H = \{h_k\}$ and $G = \{g_k\}$ are called Quadrature Mirror Filters (QMF's) that, once chosen, define a particular wavelet basis.

3.1 Representation of Functions in Wavelet Bases

The projection of a function $f(x)$ onto subspace $\mathbf{V}_j$ is given by

$$(P_j f)(x) = \sum_{k\in\mathbb{Z}} s_k^j \varphi_{j,k}(x) \tag{3.4}$$

where P_j denotes the projection operator onto subspace $\mathbf{V}_j$. The set of coefficients $\{s_k^j\}_{k\in\mathbb{Z}}$, which we refer to as "averages", are computed via the inner product $s_k^j = \int_{-\infty}^{+\infty} f(x)\varphi_{j,k}(x)dx$. It follows that we can also write $(P_j f)(x)$ as a sum of projections of $f(x)$ onto subspaces $\mathbf{W}_{j'}$, $j' > j$

$$(P_j f)(x) = \sum_{j'>j}\sum_{k\in\mathbb{Z}} d_k^{j'}\psi_{j',k}(x) \tag{3.5}$$

where the set of coefficients $\{d_k^j\}_{k\in\mathbb{Z}}$, which we refer to as "difference", are computed via the inner product $d_k^j = \int_{-\infty}^{+\infty} f(x)\psi_{j,k}(x)dx$. The projection of a function on subspace $\mathbf{W}_j$ is denoted $(Q_j f)(x)$, where $Q_j = P_{j-1} - P_j$.

3.2 The Standard and Non-Standard Form of Operators

In order to represent an operator $T : \mathbf{L}^2(\mathbb{R}) \to \mathbf{L}^2(\mathbb{R})$ in the wavelet system of coordinates, we consider two ways to define two-dimensional wavelet bases. First, we consider a two-dimensional wavelet basis which is arrived at by computing the tensor product of two one-dimensional wavelet basis functions, e.g.

$$\psi_{j,j',k,k'}(x,y) = \psi_{j,k}(x)\psi_{j',k'}(y) \tag{3.6}$$

where $j, j', k, k' \in \mathbb{Z}$. This choice of basis leads to the standard form (S-form) of an operator [5, 7]. The projection of the operator T into the multiresolution analysis is represented in the S-form by the set of operators

$$T = \{A_j, \{B_j^{j'}\}_{j'\geq j+1}, \{\Gamma_j^{j'}\}_{j'\geq j+1}\}_{j\in\mathbb{Z}}\ , \tag{3.7}$$

where the operators A_j, $B_j^{j'}$, and $\Gamma_j^{j'}$ are projections of the operator T into the multiresolution analysis as follows

$$\left.\begin{array}{rcl} A_j &=& Q_j T Q_j \quad : \quad \mathbf{W}_j \to \mathbf{W}_j \\ B_j^{j'} &=& Q_j T Q_{j'} \quad : \quad \mathbf{W}_{j'} \to \mathbf{W}_j \\ \Gamma_j^{j'} &=& Q_{j'} T Q_j \quad : \quad \mathbf{W}_j \to \mathbf{W}_{j'} \end{array}\right\} \tag{3.8}$$

for $j = 1, 2, \ldots, n$ and $j' = j+1, \ldots, n$.

If n is the finite number of scales, then (3.7) is restricted to the set of operators

$$T_0 = \{A_j, \{B_j^{j'}\}_{j'=j+1}^{j'=n}, \{\Gamma_j^{j'}\}_{j'=j+1}^{j'=n}, B_j^{n+1}, \Gamma_j^{n+1}, T_n\}_{j=1,\ldots,n} \ , \tag{3.9}$$

where T_0 is the projection of T on $\mathbf{V}_0$. Here the operator T_n is the coarse scale projection of the operator T on $\mathbf{V}_n$,

$$T_n = P_n T P_n : \mathbf{V}_n \to \mathbf{V}_n \ . \tag{3.10}$$

The subspaces $\mathbf{V}_j$ and $\mathbf{W}_j$ appearing in (3.8) and (3.10) can be periodized.

The operators $A_j, B_j^{j'}, \Gamma_j^{j'}$, and T_j appearing in (3.7) and (3.9) are represented by matrices $\alpha^j, \beta^{j,j'}, \gamma^{j,j'}$ and s^j with entries defined by

$$\left.\begin{array}{rcl} \alpha_{k,k'}^j &=& \int\int \psi_{j,k}(x) K(x,y) \psi_{j,k'}(y) dx dy \\ \beta_{k,k'}^{j,j'} &=& \int\int \psi_{j,k}(x) K(x,y) \varphi_{j',k'}(y) dx dy \\ \gamma_{k,k'}^{j,j'} &=& \int\int \varphi_{j,k}(x) K(x,y) \psi_{j',k'}(y) dx dy \\ s_{j,k'}^j &=& \int\int \varphi_{j,k}(x) K(x,y) \varphi_{j,k'}(y) dx dy \end{array}\right\} \tag{3.11}$$

where $K(x,y)$ is the kernel of the operator T. The operators in (3.9) are organized as blocks of a matrix as shown in Figure 1.

In [5] it is observed that if the operator T is a Calderón-Zygmund or pseudo-differential operator then, for a fixed accuracy, all the operators in (3.7) are banded. As a result the S-form has several "finger" bands, illustrated in Figure 1.

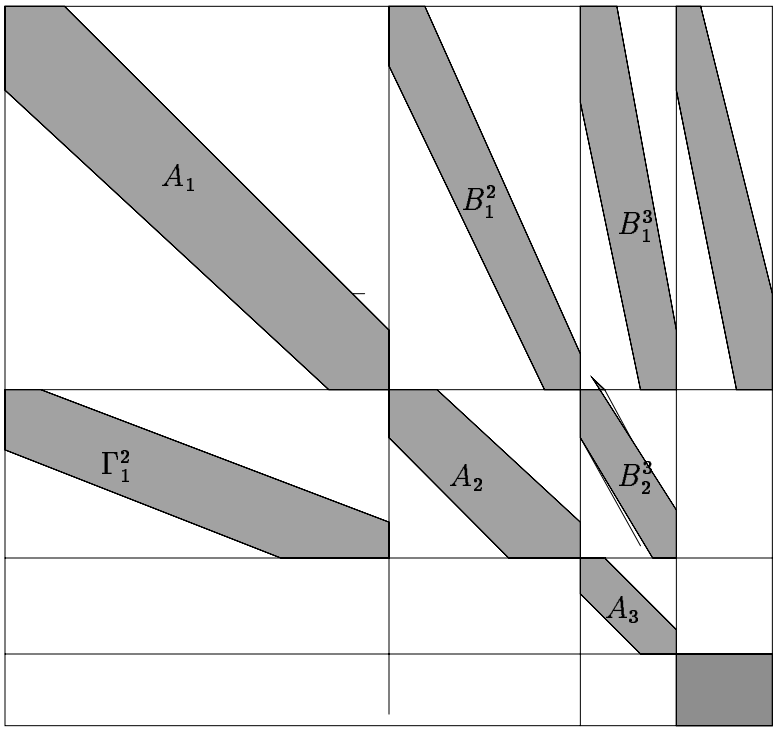

Figure 1: The band structure of the standard form

These "finger" bands correspond to interactions between different scales. For a large class of operators, e.g. pseudo-differential, the interaction between different scales (characterized by the size of the coefficients in the bands) decays as the distance $|j - j'|$ between the scales increases. Therefore, if the scales j and j' are well separated, then for a given accuracy, the operators $B_j^{j'}$ and $\Gamma_j^{j'}$ can be neglected. For compactly supported wavelets, the distance $|j - j'|$ is quite significant. The control of the interaction between scales is better in the non-standard representation of operators.

An alternative to forming two-dimensional wavelet basis functions using the tensor product (which led us to the S-form representation of operators) is to consider functions which are combinations of the wavelet, $\psi(\cdot)$, and the scaling function, $\varphi(\cdot)$. The wavelet representation of an operator in the non-standard form (NS-form) is arrived at using bases formed by combinations of wavelet and scaling functions, for example, in $\mathbf{L}^2(\mathbb{R}^2)$

$$\begin{aligned} &\psi_{j,k}(x)\psi_{j,k'}(y) \\ &\psi_{j,k}(x)\varphi_{j,k'}(y) \\ &\varphi_{j,k}(x)\psi_{j,k'}(y) \end{aligned} \tag{3.12}$$

where $j, k, k' \in \mathbb{Z}$. The NS-form of an operator T is obtained by expanding T in the "telescopic" series

$$T = \sum_{j \in \mathbb{Z}} (Q_j T Q_j + Q_j T P_j + P_j T Q_j) \ , \tag{3.13}$$

where P_j and Q_j are projectors on subspaces $\mathbf{V}_j$ and $\mathbf{W}_j$, respectively. We observe that in (3.13) the scales are decoupled. The expansion of T into the NS-form is thus represented by the set of operators $T = \{A_j, B_j, \Gamma_j\}_{j \in \mathbb{Z}}$, where the operators A_j, B_j, and Γ_j act on subspaces $\mathbf{V}_j$ and $\mathbf{W}_j$ as follows

$$\left.\begin{array}{rcl} A_j &=& Q_j T Q_j \quad : \quad \mathbf{W}_j \to \mathbf{W}_j \\ B_j &=& Q_j T P_j \quad : \quad \mathbf{V}_j \to \mathbf{W}_j \\ \Gamma_j &=& P_j T Q_j \quad : \quad \mathbf{W}_j \to \mathbf{V}_j \end{array}\right\} \tag{3.14}$$

see e.g. [5].

If $J \leq n$ is the finite number of scales, then (3.13) is truncated to

$$T_0 = \sum_{j=1}^{J} (Q_j T Q_j + Q_j T P_j + P_j T Q_j) + P_J T P_J \ , \tag{3.15}$$

and the set of operators is restricted to $T_0 = \{\{A_j, B_j, \Gamma_j\}_{j=1}^{j=J}, T_J\}$, where T_0 is the projection of the operator on $\mathbf{V}_0$ and T_J is a coarse scale projection of the operator T $T_J = P_J T P_J$: $\mathbf{V}_J \to \mathbf{V}_J$.

The operators A_j, B_j, Γ_j and T_J appearing in the NS-form are represented by matrices $\alpha^j, \beta^j, \gamma^j$, and s^j with entries defined by

$$\left.\begin{array}{rcl} \alpha^j_{k,k'} &=& \int\int K(x,y) \psi_{j,k}(x) \psi_{j,k'}(y) dx dy \\ \beta^j_{k,k'} &=& \int\int K(x,y) \psi_{j,k}(x) \varphi_{j,k'}(y) dx dy \\ \gamma^j_{k,k'} &=& \int\int K(x,y) \varphi_{j,k}(x) \psi_{j,k'}(y) dx dy \\ s^j_{k,k'} &=& \int\int K(x,y) \varphi_{j,k}(x) \varphi_{j,k'}(y) dx dy \end{array}\right\} \tag{3.16}$$

in $\mathbf{L}^2(\mathbb{R}^2)$. The operators are organized as blocks of a matrix as shown in Fig. 2.

The price of uncoupling the scale interactions in (3.13) is the need for an additional projection into the wavelet basis of the product of the NS-form and a function.

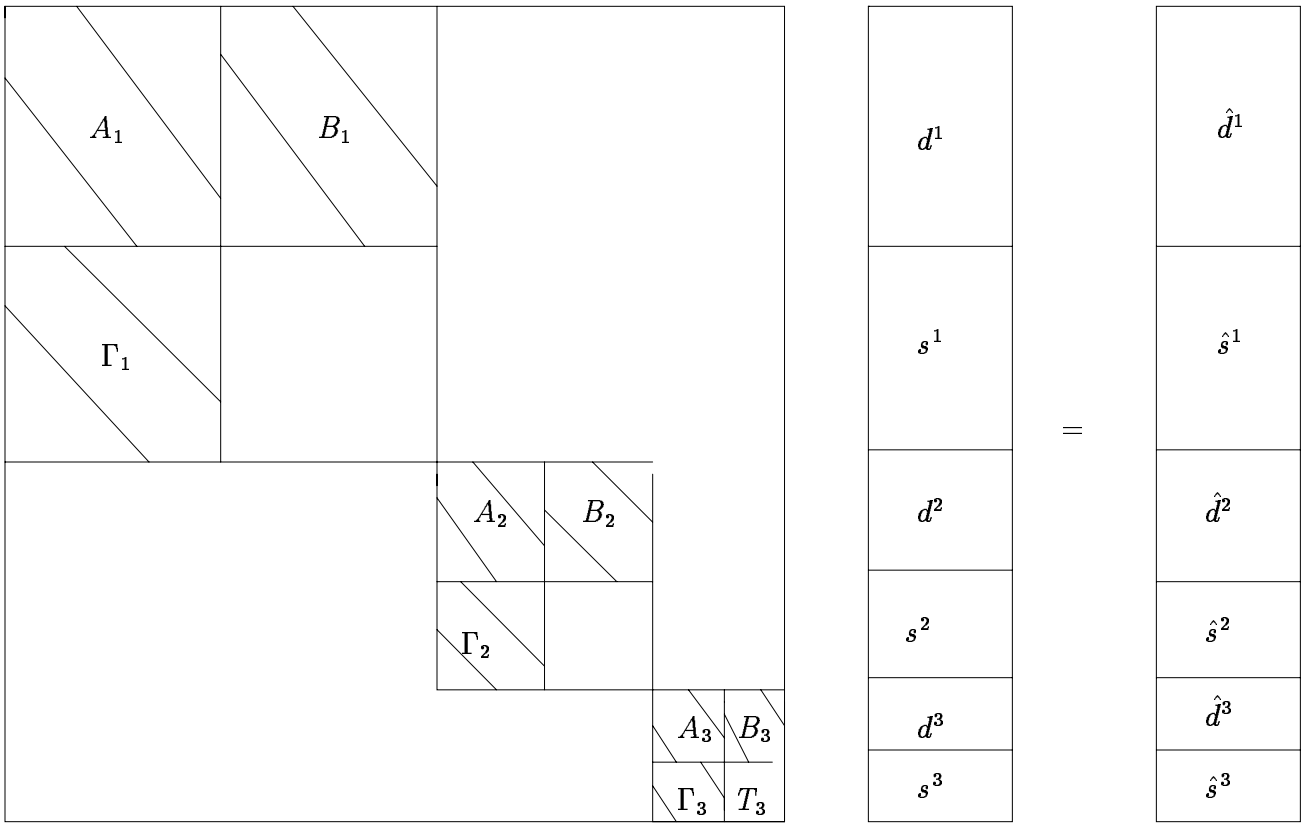

Figure 2: Application of non-standard form to a vector

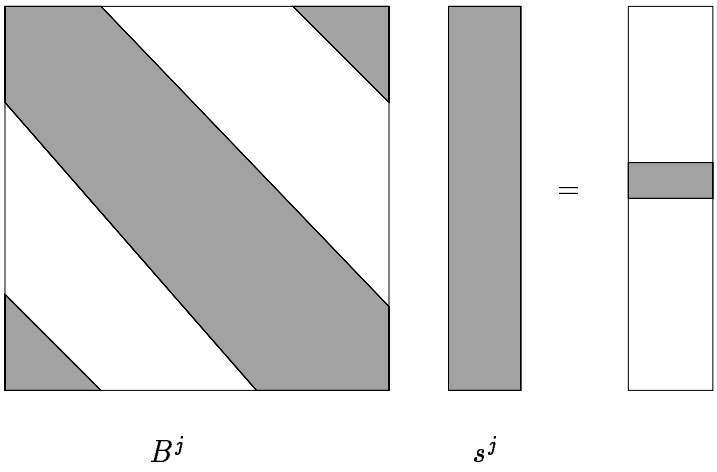

Figure 3: Application of non-standard form to a vector

Referring to Fig. 2 we see that the NS-form is applied to both averages and differences of the wavelet expansion of a function.

It follows from (3.13) that after applying the NS-form to a vector we arrive at the representation

$$(T_0 f_0)(x) = \sum_{j=1}^{J} \sum_{k \in \mathbb{Z}_{2^{n-j}}} \hat{d}_k^j \psi_{j,k}(x) + \sum_{j=1}^{J} \sum_{k \in \mathbb{Z}_{2^{n-j}}} \hat{s}_k^j \varphi_{j,k}(x) \ . \tag{3.17}$$

The representation (3.17) consists of both averages and differences on all scales which can either be projected into the wavelet basis or reconstructed to space $\mathbf{V}_0$. In order to project (3.17) into the wavelet basis we form the representation

$$(T_0 f_0)(x) = \sum_{j=1}^{J} \sum_{k \in \mathbb{Z}_{2^{n-j}}} d_k^j \psi_{j,k}(x) + \sum_{k \in \mathbb{Z}_{2^{n-J}}} s_k^J \varphi_{J,k}(x) \ . \tag{3.18}$$

using the decomposition algorithm described before.

In this work we are interested in developing adaptive algorithms, i.e. algorithms such that the number of operations performed is proportional to the number of significant coefficients in the wavelet expansion of solutions of partial differential equations. The S-form has the adaptivity property, i.e. applying the S-form of an operator to the wavelet expansion of a function is a matter of multiplying a sparse vector by a sparse matrix. All the implementation in this paper is performed using sparse data structures. On the other hand, as we have mentioned before, the S-form is not a very efficient representation of, for example, convolution operators.

In section 5 we address the issue of adaptively multiplying the NS-form and a vector. Since the NS-form of a convolution operator remains a convolution, the A^j, B^j, and Γ^j blocks may be thought of as being represented by short filters. We can exploit the efficient representation afforded us by the NS-form and use the vanishing-moment property of the B^j and Γ^j blocks of the NS-form of differential operators to develop an adaptive algorithm.

4 Application of masks on the V and W subspaces

Let us represent the source term f and the solution u in (2.3) in the wavelet basis,

$$f(x) = \sum_{j \le n} \sum_{\mathbf{k}} \sum_{\sigma} f_{j,\mathbf{k}}^{\sigma} \psi_{j,\mathbf{k}}^{\sigma}(x) + \sum_{\mathbf{k}} s_{n,\mathbf{k}}^{f} \varphi_{n,\mathbf{k}}(x), \tag{4.1}$$

$$u(x) = \sum_{j \le n} \sum_{\mathbf{k}} \sum_{\sigma} u_{j,\mathbf{k}}^{\sigma} \psi_{j,\mathbf{k}}^{\sigma}(x) + \sum_{\mathbf{k}} s_{n,\mathbf{k}}^{u} \varphi_{n,\mathbf{k}}(x), \tag{4.2}$$

where

$$f_{j,\mathbf{k}}^{\sigma} = \langle f, \psi_{j,\mathbf{k}}^{\sigma} \rangle, \ \ u_{j,\mathbf{k}}^{\sigma} = \langle u, \psi_{j,\mathbf{k}}^{\sigma} \rangle, \ \ s_{n,\mathbf{k}}^{f} = \langle f, \varphi_{n,\mathbf{k}} \rangle \ \text{ and } \ s_{n,\mathbf{k}}^{u} = \langle u, \varphi_{n,\mathbf{k}} \rangle. \tag{4.3}$$

We now define the ϵ-accuracy subspace for f to be the subspace on which f may be represented with accuracy ϵ, namely,

$$M_{r.h.s}^{\epsilon} = \mathbf{V}_n \bigcup \{ \text{span}\{\psi_{j,\mathbf{k}}^{\sigma}\} \,|\, (j,\mathbf{k},\sigma) \ : \ |f_{j,\mathbf{k}}^{\sigma}| > \epsilon \}, \tag{4.4}$$

and observe that the ϵ-accuracy subspace for the solution

$$M^{\epsilon}_{sol} = \mathbf{V}_n \bigcup \{\text{span}\{\psi^{\sigma}_{j,\mathbf{k}}\} \,|\, (j,\mathbf{k},\sigma) \;:\; |u^{\sigma}_{j,\mathbf{k}}| > \epsilon\} \tag{4.5}$$

may be estimated given $M^{\epsilon}_{r.h.s}$.

Proposition 4.1 *Let*

$$u(x) = \sum_{j} \sum_{\mathbf{k}} \sum_{\sigma} u^{\sigma}_{j,\mathbf{k}} \psi^{\sigma}_{j,\mathbf{k}}(x) + \text{constant} \tag{4.6}$$

be the solution of

$$\Delta u = \psi^{\sigma'}_{j',\mathbf{k}'} \quad x \in \mathbf{B} \tag{4.7}$$

with periodic boundary conditions. For any $\epsilon > 0$ there exist $\lambda > 0$ and $\mu > 0$ such that all indices $(j,\mathbf{k},\sigma)$ corresponding to the significant coefficients of the solution, $|u^{\sigma}_{j,\mathbf{k}}| \geq \epsilon$, satisfy $|\mathbf{k} - \mathbf{k}'| \leq \lambda$ and $|j - j'| \leq \mu$.

The size of $\mu > 0$ and $\lambda > 0$ depends on the particular choice of basis and, of course, on ϵ. Given $M^{\epsilon}_{r.h.s}$, we may construct the set $M_{\lambda,\mu}$ as a (λ,μ)-neighborhood of $M^{\epsilon}_{r.h.s}$. According to Proposition 4.1, $M^{\epsilon}_{sol} \subset M_{\lambda,\mu}$. We note that estimating the subspace amounts to constructing a mask which contains indices of significant coefficients.

Instead of estimating $M_{\lambda,\mu}$ directly, we may use an iterative approach. For example, solving directly on $M^{\epsilon}_{r.h.s}$ produces a solution $\tilde{u}$ with accuracy $\tilde{\epsilon} > \epsilon$. Applying the Laplacian to $\tilde{u}$, we generate $\tilde{f}$. Estimating the ϵ-accuracy subspace for $\tilde{f}$, we may use it to continue the iteration to improve the accuracy of the solution. In other words, the mask for M^{ϵ}_{sol} may be generated iteratively.

There are three main features in our approach to solve (2.3) :

1. **Estimation of the ϵ-accuracy subspace for the solution.** Our first step is to explicitly estimate the subspace M^{ϵ}_{sol} given $M^{\epsilon}_{r.h.s}$. For elliptic operators the dimension of M^{ϵ}_{sol} is proportional to that of $M^{\epsilon}_{r.h.s}$.

2. **Preconditioning of the operator.** A simple diagonal preconditioner is available for periodized differential operators in the wavelet bases [5, 6] which yields a condition number of $O(1)$. We will show in section 6 how to construct simple preconditioners in wavelet bases for more general operators.

3. **Constrained Iterative Solver.** We use preconditioned Conjugate Gradient (CG) method which we constrain to the subspace estimated at Step 1, e.g. $M_{\lambda,\mu}$. The CG method requires only a constant number of iterations due to preconditioning at Step 2, whereas the cost of each iteration is proportional to the dimension of M^{ϵ}_{sol} provided we succeed to limit the number of operations required for the application of the operator (matrix) in the CG method (see below).

Steps 1-3 constitute an adaptive algorithm for solving Poisson's equation.

Convergence. In order to demonstrate convergence of our method, we may use results in [29], where it is shown that if P_ϵ is a projector on M_f^ϵ, the ϵ-accuracy subspace for f, then for any L^p function, $1 < p < \infty$,

$$\lim_{\epsilon \to 0} P_\epsilon f(x) \to f(x) \tag{4.8}$$

almost everywhere. Since our method recovers the ϵ-accuracy subspace for the solution, M_{sol}^ϵ, in the limit as $\epsilon \to 0$ we obtain the pointwise convergence almost everywhere.

The justification for the proposition is given in section 7 after the introduction of the preconditioners for the 1-D - 3-D cases because it contains the use of masks with the application of the preconditioner.

4.1 The relation between the masks on V and W subspaces

The procedure in section 4 means that after having multiscale decomposition of the r.h.s we create masks on the "d" coefficients. The masks on the d coefficients are determined according to a predefined threshold. But during the application of the multiscale wavelet decomposition/reconstruction the S part of the multiscale is filled up (it becomes "dense") and we may loose the sparsity advantages of the whole process. It is proved in section 7, that although the location of the s_k^j coefficients is generally dense it suffices to consider in the non-standard form only those labels (j, k) near the corresponding labels of d_k^j that are used to define the mask. As it is demonstrated in Fig. 4, each mask on the s is determined by corresponding mask on the d.

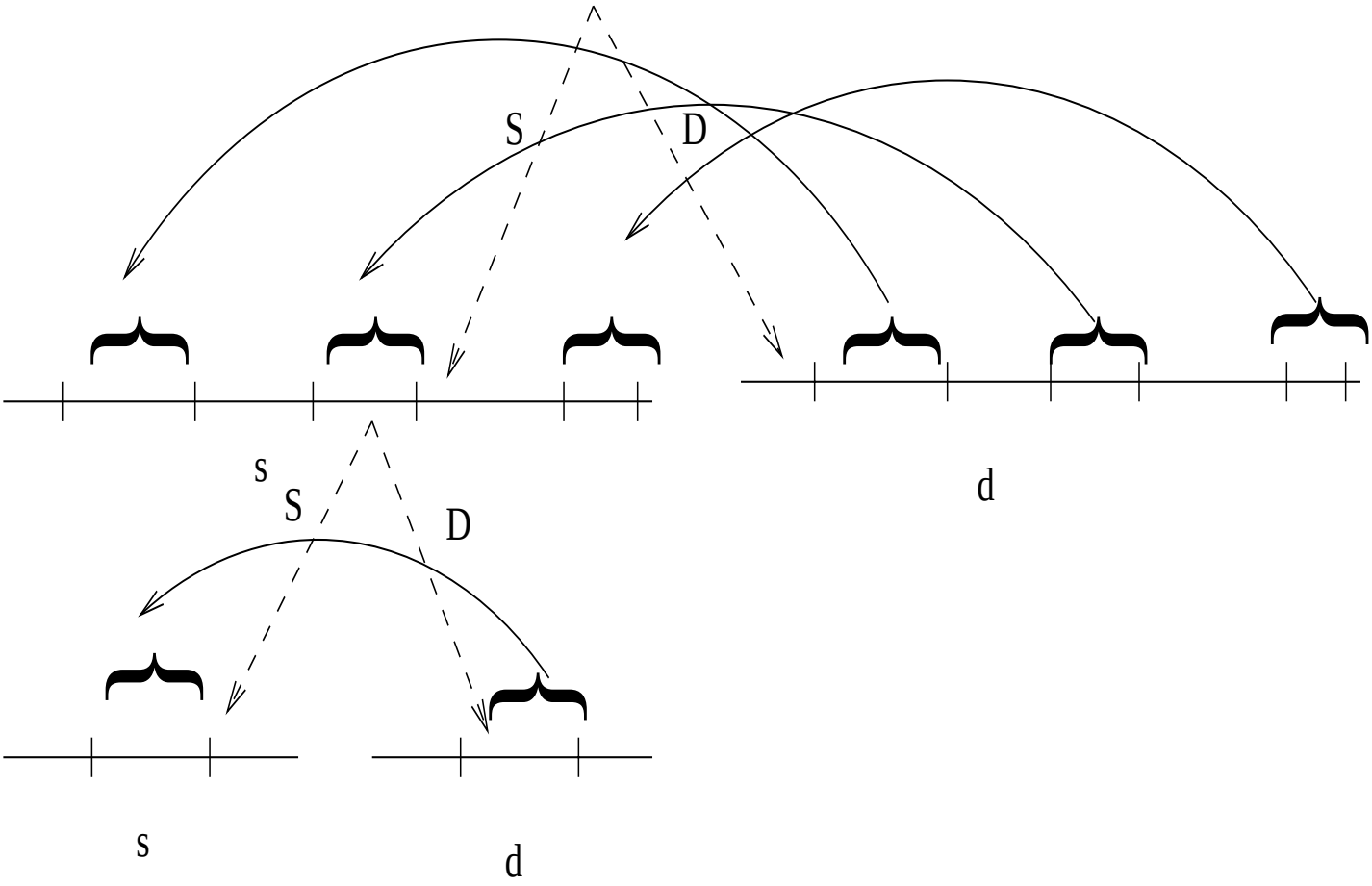

Figure 4: Illustration how the "d" masks for two scales 1-D wavelet multiscale decomposition, determine exactly the masks on the s coefficients

5 Outline of the algorithm

5.1 1-D algorithm

Let us consider the projection L_0 of the periodized operator Δ on $\mathbf{V}_0$, the finest scale under consideration,

$$L_0 = P_0 \Delta P_0, \tag{5.1}$$

and L_s and L_{ns} are its standard (s-form) and non-standard forms (ns-form) [7], respectively.

One of the difficulties in solving (2.2) stems from the inherently large condition number of the linear system resulting from the discretization of (2.2). As it was shown in [5, 6], using a diagonal preconditioner in the wavelet system of coordinates yields a linear system with the condition number typically less than 10, independently of its size. Let $\mathcal{P}$ denote such a diagonal preconditioner.

In [6] the s-form is used to solve the two-point boundary value problem. Alternatively, we may use the ns-form. Some care is required at this point since the preconditioned ns-form is dense unlike the s-form which remains sparse. Thus, in the process of solving the linear system, it is necessary to apply the preconditioner and the ns-form sequentially in order to maintain sparsity. The ns-form is preferable in multiple dimensions since, for example, differential operators require $O(1)$ elements for representation on all scales (see e.g. [5]).

We develop a constrained preconditioned CG algorithm for solving (2.2) in an adaptive manner. Both the s-form and the ns-form may be used for this purpose but it appears that using the ns-form is more efficient especially if compactly supported wavelets are used and high accuracy is required.

Let us consider (2.2) in the wavelet system of coordinates

$$L_{ns} u_w = f_w, \tag{5.2}$$

where f_w and u_w are representations of f and u in the wavelet system of coordinates. This equation should be understood to include the rules for applying the ns-form (see [7]).

Let us rewrite (5.2) using the preconditioner $\mathcal{P}$ as

$$\mathcal{P}\, L_{ns}\, \mathcal{P} v = \mathcal{P} f_w, \tag{5.3}$$

where $\mathcal{P} v = u$. For example, for the second derivative the preconditioner $\mathcal{P}$ is as follows:

$$\mathcal{P}_{il} = \delta_{il} 2^j \tag{5.4}$$

where $1 \le j \le n$ is chosen depending on i, l so that $n - n/2^{j-1} + 1 \le i, l \le n - n/2^j$, and $\mathcal{P}_{nn} = 2^n$.

The periodized operator Δ has the null space of dimension one which contains constants. If we use the full decomposition (over all n scales) in the construction of the ns-form then the null space coincides with the subspace $\mathbf{V}_n$ which in this case has dimension one (see [6]). This allows us to solve (5.3) on the range of the operator,

$$\bigoplus_{1 \le j \le n} \mathbf{W}_j \tag{5.5}$$

where the linear system (5.3) is well conditioned.

Constrained Iterative Solver. In order to solve (5.3) we apply the Conjugate Gradient method constrained to the subspace $M_{\lambda,\mu}$. Without such constrain the conjugate directions become "dense" at early stages of the iteration only to become small outside the subspace $M_{\lambda,\mu}$ later. Thus, constraining the solution to a subspace is critical for an adaptive algorithm.

In applying the conjugate gradient method in the wavelet coordinates, we generate only those entries of conjugate directions which are in the set of significant indices which define the subspace $M_{\lambda,\mu}$ (called the masks). This yields an algorithm where the number of operations at each iteration is proportional to the number of elements of $M_{\lambda,\mu}$. The number of iterations is $O(1)$ and, thus, the overall number of operations is proportional to the number of significant coefficients of f, i.e., the dimension of $M^{\epsilon}_{r.h.s}$.

Remark: operators with variable coefficients. As in the case of the Laplacian, the ϵ-accuracy subspace for the solution may be estimated using corresponding subspaces for the r.h.s and the coefficients. Essentially, we consider the union of such subspaces as a starting point for constructing $M_{\lambda,\mu}$. These estimates may be revised in the process of iteration. The proof that the above algorithm is applicable to operators with variable coefficients is given in section 7.

5.2 The 2-D and 3-D algorithms

The 1-D algorithm was described in [1, 2]. The wavelet decomposition of 2-D and 3-D functions can be obtained by applying the quadrature formula to each variable. The various steps which have to be performed in order to apply the Laplacian operator to a given function can be summarized as follows (the description of the steps is related to Fig. 2):

1. Computation of matrices A_j, B_j, and Γ_j.
2. Computation of vectors d^j and s^j.
3. Multiplication of A_j, B_j and Γ_j by d^j and s^j in order to obtain d'^j and s'^j. This is done within the masks only.
4. Reconstruction of the result from d'^j and s'^j. This is done within the masks only.

where the above 4 steps are performed for $j = 1, \ldots, J$ and J is the number of decomposed scales. The computation is done only with coefficients in the corresponding masks.

The use of the tensor product wavelet enables us to apply the 1-D algorithm [1] for the 2-D and 3-D Laplacian operator. In the above algorithm step 1 is performed only once in the beginning to determine the representation of the 1-D second derivative operator. The other three steps are performed on each row of the matrix which corresponds to the 2-D function. This provides the partial second derivative in the x direction. The same process is applied to each column which produces the partial second derivative for the y direction. The summation of the two resulting matrices yields the Laplacian operator. In fact, the computations on the rows and on the columns are performed in the same time. For an $N \times N$ matrix, the computations are then described by the following algorithm:

- Computation of matrices A_j, B_j, and Γ_j.
- Loop: for $i = 1$ to N
    1. Computation of row vectors d^j and s^j which correspond to row i.
    2. Computation of column vectors d^j and s^j which correspond to column i.
    3. Multiplication of A_j, B_j and Γ_j by rows d^j and s^j in order to obtain the rows d'^j and s'^j. This is done within the masks only.
    4. Multiplication of A_j, B_j and Γ_j by columns d^j and s^j in order to obtain columns d'^j and s'^j. This is done within the masks only.

5. Reconstruction of the result from rows d'^j and s'^j. This is done within the masks only.
6. Reconstruction of the result from columns d'^j and s'^j. This is done within the masks only.
7. Partial construction of the final matrix from the two vectors obtained in steps 5 and 6.

where the above steps are performed for $j = 1, \ldots, J$ and J is the number of decomposed scales. The computation is done only with coefficients in the corresponding masks.

The 3-D is treated similarly while taking into consideration the third direction (height).

6 Preconditioners

In this section we describe how to construct preconditioner to 1-D - 3-D problems. It is based on the s-form of the decomposition (see Fig. 1).

6.1 1-D Preconditioner for the operator $-\Delta + Const$

Let us demonstrate how to construct a diagonal preconditioner for the sum of operators $-\Delta + Const$ in wavelet bases. We observe that if A and B are diagonal operators with diagonal entries a_i and b_i, then the diagonal operator with entries $1/(a_i + b_i)$ (provided $a_i + b_i \neq 0$) is an ideal preconditioner.

In our case, the operator $-\Delta$ is not diagonal but we know a good diagonal preconditioner for it in wavelet bases (5.4). Let us use this preconditioner instead of $-\Delta$ for the purpose of constructing a preconditioner for $-\Delta + Const$, where $Const > 0$. We note that in wavelet bases the identity operator remains unchanged. We restrict $Const \cdot I$, where I is the identity operator, to the subspace

$$\bigoplus_{1 \leq j \leq n} \mathbf{W}_j \tag{6.1}$$

and construct a preconditioner on this subspace.

We obtain

$$\mathcal{P}_{il} = \frac{\delta_{il}}{\sqrt{2^{-2j} + Const}} \tag{6.2}$$

where $1 \leq j \leq n$ is chosen depending on i, l so that $n - n/2^{j-1} + 1 \leq i, l \leq n - n/2^j$, and $\mathcal{P}_{nn} = 1/\sqrt{2^{-2n} + Const}$. The square root appears in (6.2) in order to symmetrize the application of the preconditioner as shown before. In Table 1 we illustrate the effect of preconditioning of the operator $-\frac{d^2}{dx^2} + Const$ by the diagonal matrix (6.2).

Const	κ	κ_p
$7.1{\cdot}10^{-0}$	$2.4{\cdot}10^{0}$	2.1
$7.1{\cdot}10^{-1}$	$1.5{\cdot}10^{1}$	6.3
$7.1{\cdot}10^{-2}$	$1.4{\cdot}10^{2}$	9.4
$7.1{\cdot}10^{-3}$	$1.3{\cdot}10^{3}$	9.5
$7.1{\cdot}10^{-4}$	$6.7{\cdot}10^{3}$	7.5
$7.1{\cdot}10^{-5}$	$1.5{\cdot}10^{4}$	5.0

Table 1: Condition numbers κ before and κ_p after preconditioning of the operator $-\frac{d^2}{dx^2} + Const$ in the basis of Daubechies' wavelets with six vanishing moments. There are 8 scales and the matrix size is 256×256.

Remark. If we consider an operator $-\Delta + V$, where V is an operator of multiplication by a function $V(x)$, a similar construction may be obtained on fine scales. On fine scales where the function $V(x)$ does not change significantly over the support of wavelets, we may consider the diagonal operator V^{diag},

$$\left(V^{diag}\psi^{\sigma}_{j,\mathbf{k}}\right) = V(x_{j,\mathbf{k}})\psi^{\sigma}_{j,\mathbf{k}}, \tag{6.3}$$

where $x_{j,\mathbf{k}}$ is a point within the support of the wavelet $\psi^{\sigma}_{j,\mathbf{k}}$. Using V^{diag} instead of V, we obtain the preconditioner in a manner outlined above. We will address the problem of constructing preconditioners for operators of the form $-\Delta + V$ elsewhere.

For many iterative methods, the algebraic error of iterative schemes decays exponentially and the rate of decay is controlled by the condition number κ of the matrix corresponding to the operator. For the Conjugate Gradient method, for instance, the error between the exact solution U and the approximated solution U^m after m iterations is given by:

$$\|U - U^m\|_2 \leq \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^m \|U\|_2 . \tag{6.4}$$

For symmetric matrices, the condition number κ is the absolute value of the ratio between the highest and the lowest eigenvalue. In order to improve the convergence, the condition number κ can be reduced by solving an equivalent linear system,

$$PAP^T x = Pb \tag{6.5}$$

where P has to satisfy two requirements:

- it must lower the system's condition number, $\kappa(PAP^T) \ll \kappa(A)$,

- The application of P on an arbitrary vector must be simple from the computational point of view.

For instance, diagonal preconditioners are fast to apply. In the wavelet basis, the second derivative being almost diagonal, a good diagonal preconditioner for this operator can be easily defined by:

$$P_{il} = \delta_{il} 2^j, \quad 1 \leq j \leq n, \quad P_{NN} = 2^n, \tag{6.6}$$

such that $N - N/2^{j-1} + 1 \leq i, l \leq N - N/2^j$ where $N = 2^n$ is the number of discretization points. It can be noticed that the computation of the matrix-matrix multiplication to apply the CG process is not necessary, and all the computations can be performed by using only matrix-vector multiplications. In other words, the CG is a 1-D process independent of the dimensionality of the problem. In the same way that the explicit matrix representation of the operator itself was not required in dimension greater than one, the matrix corresponding to the preconditioner is also not necessary. It is sufficient to specify the effect of its application to a given vector.

6.2 2-D and 3-D preconditioners

Let's proceed by analogy with the 1-D case to define the corresponding preconditioner for 2-D and 3-D problems. For this, we have to consider the tensor product wavelet transform (the standard decomposition-Fig. 1) of a function (Figs. 5 and 6).

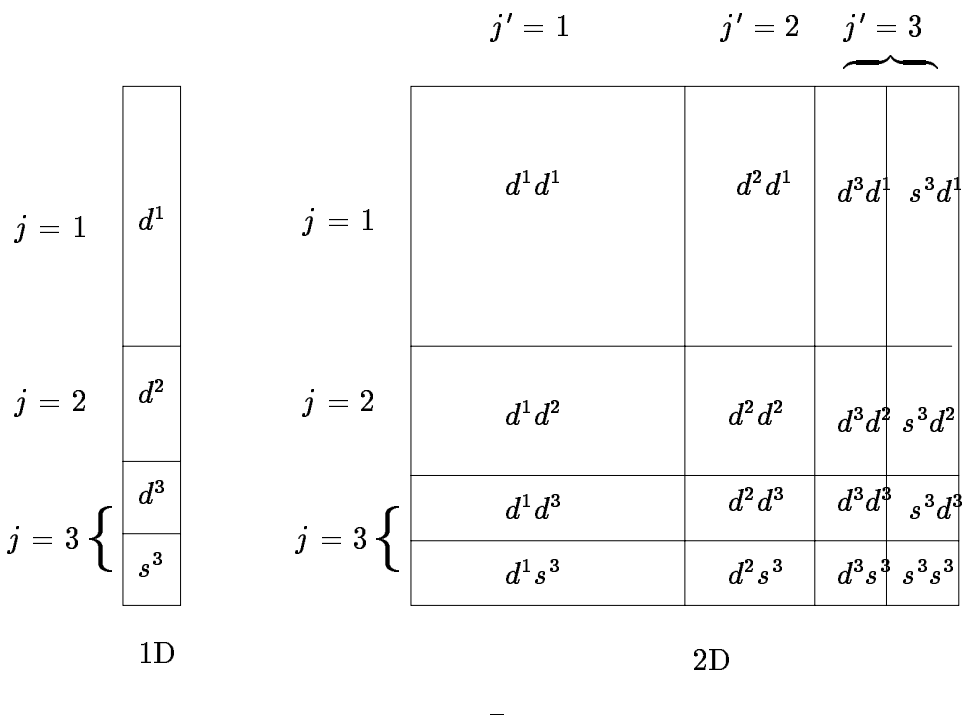

Figure 5: 1-D and 2-D Wavelet tensor product (standard decomposition)

In the 1-D case (Fig. 5), the previously defined diagonal preconditioner is operated on the matrix by multiplying all the wavelet coefficients from a given scale j by the same number 2^j. In the same way, the 2-D preconditioner will act by multiplying the wavelet coefficients located in the same rectangle by a constant which has to be defined. The matrix P has to be built in order to reduce the condition number of the Laplacian in a wavelet basis. As we know the Laplacian can be decomposed using $P\Delta P^T = PD_xP^T + PD_yP^T$, ($D_x$ and D_y are partial second derivatives in x and y, respectively) and P is defined as

$$P(j,j') = 2^{min(j,j')}, \quad 1 \leq j,j' \leq n. \tag{6.7}$$

In the same way, the 3-D preconditioner is defined by:

$$P(j,j',j'') = 2^{min(j,j',j'')}, \quad 1 \leq j,j',j'' \leq n. \tag{6.8}$$

This means that all the wavelet coefficients which belong to the same box has to be multiplied by this factor.

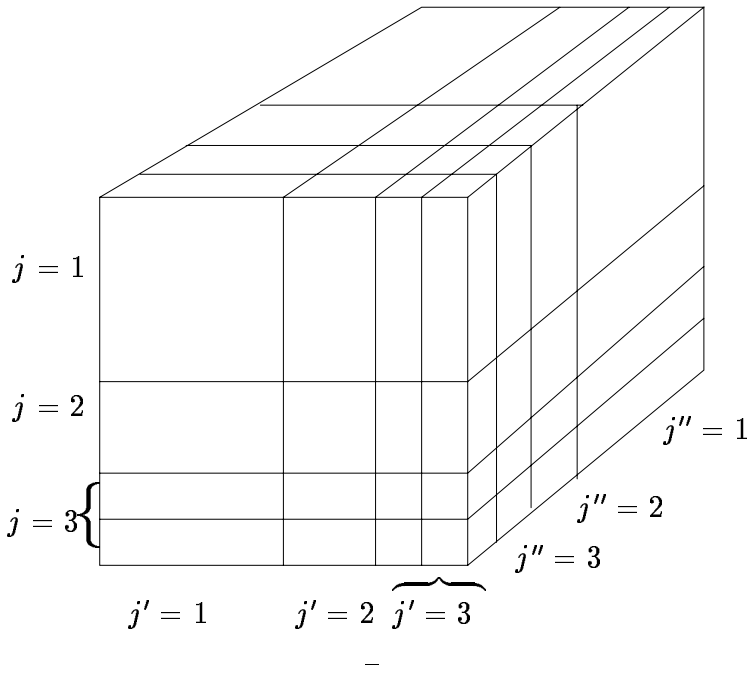

Figure 6: 3-D tensor wavelet

7 Justification of the masking procedure for second order elliptic equations

Consider the operator

$$\mathcal{L}(u) = \sum \partial_i(a_{ij}(x)\partial_j u) + q(x)u = f \tag{7.1}$$

where $a_{ij}(x)$ are smooth coefficients satisfying

$$\sum_{i,j} a_{ij}(x)\xi_i\xi_j \geq |\xi|^2 \quad \text{and} \quad q(x) \leq 2^{2j_0} \tag{7.2}$$

We claim that the solution of (7.1) can be reduced to solving (7.1) within a low dimensional space obtained as the union of V_{j_0} and a high frequency space defined as the $\epsilon-$"mask" of f. (Moreover, if $a_{ij}(x)$ are less regular an appropriate $\epsilon-$mask of $a_{ij}(x)$ can be introduced to account for the high frequencies of $a_{ij}(x)$).

We begin by observing that the solution of (7.1) can be reduced to the solution of a sparse system on V_{j_0}.

In fact, let $u = u_0 + u_1,\ \ u_0 \in V_{j_0},\quad u_1 \in V_{j_0}^{\perp}$ and similarly, $f = f_0 + f_1$.

Let us rewrite (7.1) as:

$$\begin{pmatrix} \mathcal{L}_0 & A \\ B & \mathcal{L}_1 \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \end{pmatrix} = \begin{pmatrix} f_0 \\ f_1 \end{pmatrix}$$

We claim that $\mathcal{L}_1$ is invertible on $V_{j_0}^{\perp}$ since

$$\langle \mathcal{L}_1 h, h\rangle = \langle \mathcal{L}h, h\rangle \text{ for } h \in V_{j_0}^{\perp}$$

we have that the preconditionned operator $\mathcal{L}_1^0 = \mathcal{P}\mathcal{L}\mathcal{P}$, where $\mathcal{P}$ is the preconditionner, satisfies

$$\|h\|^2 - 2^{2j_0}\|\mathcal{P}h\|^2 \leq \langle \mathcal{L}_1^0 h, h\rangle = \langle \mathcal{L}^0 h, h\rangle \leq c_2\|h\|^2$$

where the preconditionner is the diagonal operator defined by

$$\mathcal{P}(2^{\frac{i}{2}}\psi(2^j x - k)) = 2^{-j}(2^{\frac{i}{2}}\psi(2^j x - k))$$

has norm $\leq 2^{-j_0-1}$ on $V_{j_0}^{\perp}$ from which it follows that

$$2^{2j_0}\|\mathcal{P}h\|_2^2 \leq \frac{1}{4}\|h\|_2^2 \quad \text{if} \quad h \in V_{j_0}^{\perp}$$

and that $\mathcal{L}_1^0$ is invertible.

The solution of (7.1) is reduced to showing

$$\left(\mathcal{L}_0 - A\mathcal{L}_1^{-1}B\right)u_0 = f_0 - A\mathcal{L}_1^{-1}f_1$$

on the low frequency space V_{j_0}.

We now discuss the masking procedure for inverting $\mathcal{L}_1^0$. We can view $\mathcal{L}_1^0$ as the restriction to $V_{j_0}^{\perp}$ of the operator $L^0 = P_{j_0} + \mathcal{L}_1^0$ which is an invertible Calderon Zygmund operator (see ...) given by

$$Tf = \int k(x,y)f(y)dy \tag{7.3}$$

where $k(x,y)$ is a Calderon Zygmund kernel (see B.C.R). We start by assuming that $T(1) = 0$ in order to justify our masking procedure, we will prove later that this condition is "effectively" satisfied in our case.

As shown in [26] p. 52 the operator T is a self adjoint Calderon Zygmund operator whose matrix in standard form is sparse around the diagonal with exponentially decaying fingers (representing cross scale interactions). Consequently, the approximate localization of the wavelet coefficients of Tf around the coefficients of f is assured, i.e. there is a projection M^ϵ (called mask projection) such that $\|f - M^\epsilon(f)\| \leq \epsilon$ and $\|Tf - M^\epsilon(Tf)\| \leq \epsilon$ from which it follows that $\|M^\epsilon L^0 M^\epsilon T f - f\| \leq 2\epsilon$ and that it is enough to solve the equation $M^\epsilon L^0 M^\epsilon u_1 = f_1$ in the range of M^ϵ in order to get an ϵ approximation of the solution $\mathcal{L}_1^{-1} f_1$.

This equation does not account explicitly for the nature of the mask around the large wavelet coefficients of f since each coefficient interacts with coefficients at larger scales. In order to account for this multiscale interaction effect it is easier to work in the nonstandard form in which the scales are independent and the mixing occurs only through the s_k coefficients and the β, γ matrices.

Our goal now is to prove that although the location of the s_k^j coefficients is generally dense it suffices to consider in the nonstandard form only those labels (j,k) near the corresponding labels of d_k^j used to define the mask M^ϵ.

We start by observing that $\sum_{k'} \beta_{k,k'}^j = 0$. In fact, since $\sum_{k'} \varphi_{j,k'}(y) = const$

$$\sum_{k'} \beta_{k,k'}^j = \int \psi_{j,k}(x) \left(\int k(x,y) \sum_{k'} \varphi_{j,k'}(y) dy \right) dx$$

$$= const \int \psi_{j,k}(x) \left(\int k(x,y) dy \right) dx = \int \psi_{j,k}(x) T(1)(x) dx = 0.$$

Therefore,

$$\sum \beta_{k,k'}^j s_{k'}^j = \sum B_{k,k'}^j (s_{k'+1}^j - s_{k'}^j)$$

where $B_{k,k'}^j = \sum_{l>k'} \beta_{k,l}^j$.

Since $\sum_l \beta_{k,l}^j = 0$ and $|\beta_{k,l}^j| \leq \frac{c_m}{(1+|k-l|)^m}$ we conclude that

$$|B_{k,k'}^j| \leq \frac{C}{1 + |k-k'|^{m-1}}$$

Repeating the summation by parts argument we get

$$\sum \beta_{k,k'}^j s_{k'}^j = \sum b_{k,k'}^j (s_{k+1}^j - 2s_{k'}^j + s_{k'-1}^j)$$

since $\sum \varphi_{j,k'}(y) k' = y$ it would be enough to have $T(y) = 0$ in order to conclude that $|b_{k,k'}^j| \leq \frac{C}{1+|k-k'|^{m-2}}$.

The coefficients

$$\delta_{k'}^j = s_{k'+1}^j - 2s_{k'}^j + s_{k'-1}^j = \langle \varphi_{j,k'+1} - 2\varphi_{j,k'} + \varphi_{j,k'-1}, f \rangle$$

are correlations with a wavelet having two vanishing moments and are therefore located near the singularities of f and are around the d_k^j coefficients. Their size can be used together with the d_k^j to define the mask M^ϵ.

The matrix $b_{k,k'}^j$ being of rapid decay away from the diagonal we find that the (d,s) coefficients of Tf are close to those of f.

We now have to show that the assumptions on $T(1)$ and $T(y)$ are automatically satisfied in our case. We recall that

$$T = (P_{j_0} + \mathcal{L}_1^0)^{-1} = P_{j_0} + P_{j_0}^{\perp} \left(\mathcal{L}_1^0\right)^{-1} P_{j_0}^{\perp}$$

Therefore, for a function $\alpha(y) \in V_{j_0}$ we have $T(\alpha) = \alpha$ and $T_0 = T - P_{i_0}$ satisfies $T_0\alpha = 0$ for $\alpha \in V_{j_0}$ satisfies the required conditions.

8 Complexity analysis

In order to perform the complexity analysis we present first the version of the conjugate gradient algorithm [21] which is being used and the 1-D pseudo code that describes the Poisson solver.

8.1 Conjugate Gradient (CG)

Given $b \in I\!R^N$, a symmetric positive definite matrix $A \in I\!R^{N \times N}$, and a tolerance ϵ, the following algorithm computes a vector x such that $\|Ax - b\|_2 \approx \epsilon \|b\|_2$:

$x := 0, \quad r := b, \quad \rho_0 := \|r\|_2^2, \quad k := 1$

Do while $(\rho_{k-1} > \epsilon^2 \|b\|_2)$
{
If $k = 1$
then
$p := r$
else
$\beta_k := \rho_{k-1}/\rho_{k-2}, \quad p := r + \beta_k p$

$w := Ap$
$\alpha_k := \rho_{k-1}/p^T w$
$x := x + \alpha_k p$
$r := r - \alpha_k w$
$\rho_k := \|r\|_2^2$
$k := k + 1$
}

This algorithm requires only one application of the operator at each iteration step.

8.2 Poisson solver: 1-D pseudo code

Assume that the r.h.s has size $N = 2^J$, so we can decompose it into J levels. The basic 1-D structure that is being used in the algorithm consists of an array of size $2J + 1$. Each odd entry in this array, $2j - 1, \;\; j = 1, \ldots, J$, is a pointer to a sparse vector of size $N/2^j$ and there

we store the "s" (averaged) wavelet coefficients of scale j. Each even entry in this array, $2j$, $j = 1, \ldots, J$, is a pointer to a sparse vector of size $N/2^j$ and there we store the "d" (derivative) wavelet coefficients of scale j. The 1-D algorithm uses only this basic structure. $w_t, w_r, w_p, w_{p_1}, w_s, w_x$, which are used in the pseudo code, have this structure.

Preprocessing -

Second derivative - computes the non-standard form of the second derivative according to the filter type and filter size, number of scales in the multiresolution decomposition, and the required accuracy.

Setup of the r.h.s.- transform the r.h.s to a sparse data structure by having multiscale wavelet decomposition of the r.h.s, thresholding the wavelet coefficients, and reconstruction of the r.h.s into a sparse data structure. Then, the r.h.s transformed to be in ns-form by applying the operator (in other words, multiplication of the vector by ns-form).

Wavelet decomposition of the r.h.s - store it in w_t

Create the mask on the "d" coefficients - this mask is determined by the "d" wavelet coefficients w_t of the decomposed r.h.s.

Create the mask on the "s" coefficients - this mask is determined by the mask on the "d" coefficients.

Application of the preconditioner - on the wavelet coefficients w_t of the r.h.s.

Copy - $w_t \rightarrow w_r$

L_2 **norm of** w_r **-** denoted by ρ

The main loop on the number of iterations of the CG -

If first iteration -

$$w_r \rightarrow w_p$$
$$w_r \rightarrow w_{p_1}$$

If it is not the first iteration - $\beta = \frac{\rho}{\rho_1}$
sparse linear combination: $w_r + \beta w_p \rightarrow w_{p_1}$.
$w_{p_1} \rightarrow w_p$

Apply preconditioner - on w_p

Sparse wavelet reconstruction - from w_p using the masks on the "s" and "d".

Apply the non-standard form - on w_p using the masks on "s" and "d" to get w_s.

Apply preconditioner - on w_s

Sparse inner product - of w_s and w_{p_1} to produce r.

Compute α **-** $\alpha = \frac{\rho}{r}$

Sparse linear combination - $w_x + \alpha w_{p_1} \rightarrow w_q$.

Copy - $w_q \rightarrow w_x$.

Sparse linear combination - $w_r - \alpha w_s \to w_q$.

Copy - $w_q \to w_r$

Save - $\rho_1 = \rho$

Copy - $w_{p_1} \to w_p$

Go to beginning of the loop -

8.3 Estimation of the number of operations

The Poisson equation is solved using the Conjugate Gradient method[21] where the computation of the inverse operator is not necessary (just the application of the operator itself is required). This property enables us to use these methods for solving 2-D and 3-D problems where the operators kernels are separable.

All the computations of the solver are performed within the masks as was explained in section 4. The wavelet transform of the solution consists of N_s significant coefficients in the masks which are concentrated near the singularities (see Fig. 7 and table 2 which describe the masks around the singularities).

Although N is the total number of discretization points in each direction, we will use for our estimations the size of N_s.

Usually, the convolution with filter of size l takes $2l$ operations. By utilizing factorization properties of the wavelet filters, as was described in [3, 27], we can reduce the number of operations of the 1-D convolution by a factor of at least 2. With a 2-D convolution, which is performed as a tensor product, we can reduce the number of operations by a factor of at least 4.

During each iteration of the CG the kernel is decomopsed into the standard form (for the the application of the preconditioner and recontruction back) and then the non-standard form is applied once. We assume that $2lN_s$ operations are needed for 1-D sparse wavelet convolution with filter of length l. The same is true for the application of the non-standard form. Sparse inner product needs $2N_s$ operations. And the same is true for sparse linear combination. Therefore, one iteration of the Poisson solver necessitated $2 \cdot 2lN_s + 3 \cdot 2N_s \approx 5lN_s$ operations. Therefore, the total number of operations for N_{iter} is $N_{iter} \cdot 5lN_s$. To get ϵ-accuracy we need $N_{iter} \cdot 5lN_s \log \epsilon$ operations.

When we process $d > 1$ dimensional problem the size of the mask becomes N_s^d and the total number of operations to reach ϵ-accuracy is $N_{iter} \cdot 5lN_s^d \log \epsilon$.

The sizes of the masks are independent of the sampling rate, as can be seen in table 2. There, even with "super" sampling rate we still get the same sizes for the masks.

The algorithm is very efficient compared to the classical methods that require $O(N^{3d})$ operations to invert a matrix, for example. The efficiency of the process relies on the fact that a full description of the operator is not necessary to solve the equations.

9 Numerical results

The algorithm has been tested on 1-D, 2-D and 3-D Poisson equations. The results obtained for the 2-D and 3-D cases enforce the idea that application of the 1-D decomposition in each

direction is valid for solving problems in higher dimensions. The preconditioner has been tested on the 1-D, 2-D and 3-D cases. As it will be shown in the sequel, the preconditioner gives very good results after only few iterations. In order to illustrate the validity of the method with another type of problem, some results concerning the 2-D heat equation and equations with potential terms are also presented.

When we analyze the results in the rest of the section we assume that the total error in our case has four components:

1. The iteration error,

2. The truncation (or discretization error) which is due to the resolution properties and depends on the smoothness of the function (i.e. the number of basis functions or mesh steps per "wave length" or characteristic scale for changes or gradients) and the number of moments in the analyzing wavelet. Once you have resolution above some number of points per "wavelength" (the width of the Gaussian in our case), the error will drop in relation to a power of $1/N$ which depends on the number of vanishing moments.

3. The roundoff error which becomes significant when the other errors approach small values.

4. This is the error due to periodization, i.e. if the Gaussian does not decay sufficiently fast in the interval specified, the function is not periodic, and the periodization introduces discontinuities in function and derivatives which change the rate of convergence. This error may be small and can be avoided by modifying the test function.

   It becomes important when the scaling is large.

The *iteration error* is the most important in order to check the method as the truncation error depends on the input function and basis functions and number of points.

The iteration error is only weakly dependent on the input function and the rate of convergence is the most important issue in our method. As we can check it is also weakly dependent on the basis functions and on the scaling parameter.

Therefore isolating the convergence rate with and without preconditioning is the basic issue. The accuracy of representation of operators and derivatives by wavelets is not the issue we deal with.

As we can see the error should drop down to truncation error levels once the iteration converges. This only happens if we compute the right hand side as we did . This procedure takes care both of the periodization and of the resolution error.

9.1 Model problems

The algorithm was tested on three model problems:

Elliptic problem - the Poisson equation

$$\sum_{k=1}^{d} \frac{\partial^2 u(x)}{\partial x_k^2} = f(x), \quad x \in I\!R^d, \quad d = 1, \ldots, 3 \tag{9.1}$$

Poisson equation with potential

$$\Delta u + V(x)u + V(y)u = f(x,y) \tag{9.2}$$

Parabolic problem - the heat equation

$$\frac{\partial u(x,t)}{\partial t} - \alpha \sum_{k=1}^{d} \frac{\partial^2 u(x,t)}{\partial x_k^2} = f(x,t), \quad x \in I\!R^d, \quad d = 1,\ldots,3, \quad t \in I\!R^+. \tag{9.3}$$

The following results have been obtained for a function defined on a fixed interval (1-D), or a fixed square (2-D) or a fixed cube (3-D).

9.2 1-D examples

9.2.1 Example 1

Assume we have

$$\frac{\partial^2 u(x)}{\partial x^2} = f(x) \tag{9.4}$$

where the r.h.s $f(x)$ is numerically computed such that the exact solution is

$$u(x) = xe^{-256x^2} + 20(x+0.2)e^{-32000(x+0.2)^2} \tag{9.5}$$

Assume that N is the number of points and that $0 \leq i < N$. Then, the values of $u(x)$ for $x = (i - N/2)/N$ are determined in the following way:

$$u(i) = \begin{cases} 0.0 & \text{if } x < -0.2 \\ u(i) + x & \text{if } x \geq -0.2 \text{ and } x < -0.1 \\ u(i) - x & \text{if } x \geq -0.1 \text{ and } x < 0.1 \\ u(i) + x - 0.2 & \text{if } x \geq 0.1 \text{ and } x < 0.2 \\ 0.0 & \text{if } x \geq 0.2 \end{cases} \tag{9.6}$$

The graph with 16384 points, which describes Eqs. (9.5) and (9.6), is given in Fig. 7.

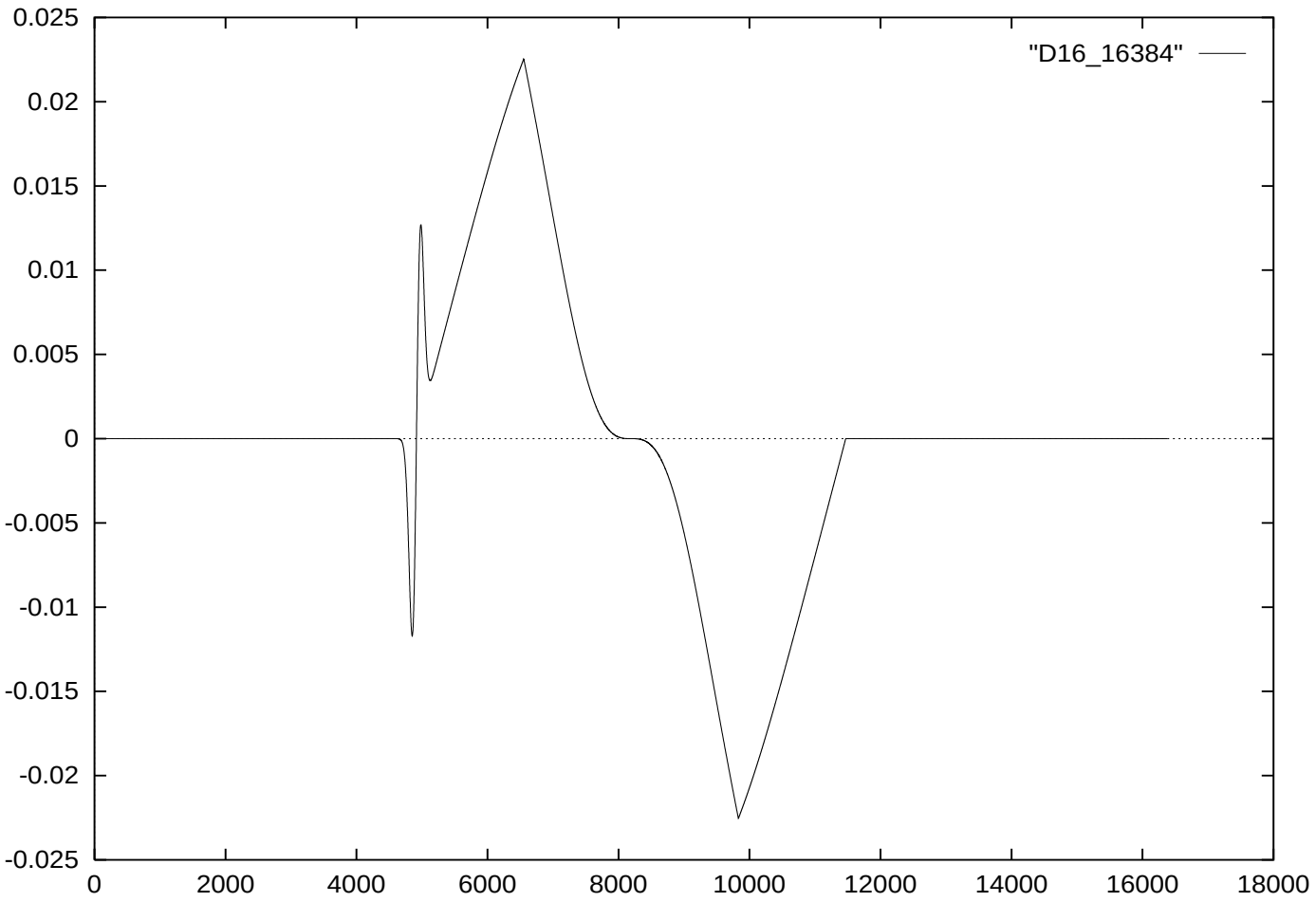

Figure 7: Graph of the 1-D function given by Eqs. (9.5) and (9.6) on 16384 points

We apply the algorithm on this function which was sampled with 1024 and 16384 points, respectively. We use Daubechies's filter of length 16 (eight vanishing moments). In the multiscale decomposition, the transformed function has four different masks in the first level, two and four respectively in the second level, etc, as it described in table 2. This table describes the beginning, end and length of each mask in each level of the multiresolution.

	Daubechies 16			
	1024 points		16384 points	
Level	from-to	length	from-to	length
1	120-178	59	2433-2474	42
1	180-221	42	3252-3293	42
1	282-323	42	4890-4932	43
1	333-374	42	5709-5751	43
2	49-117	69	1203-1243	41
2	128-193	66	1613-1653	42
2			2432-2472	42
2			2842-2881	40
3	12-102	91	558-661	105
3			794-832	40
3			1203-1242	40
3			1408-1447	40
4	0-59	60	262-342	81
4			384-422	39
4			588-627	40
4			691-729	39
5	full		119-178	60
5			179-217	39
5			281-320	39
5			333-371	39
6	full		49-115	67
6			128-192	65
7	full		12-102	91
8	full		0-58	59

Table 2: The beginning, end and length of each mask in each level of the multiresolution applied on the 1-D functions given by Eqs. (9.5) and (9.6) using Daubechies filter of length 16. The masks' accuracy is 10^{-14}.

The masks are located around the singular points. Table 2 describes the structure of the masks obtained on the decomposition of Eqs. (9.5) and (9.6). These are the masks that were determined by the application of the thresholding on the "d" coefficients. The length and the number of masks in the first 3 and 4 finer levels of the decomposition determine the performance of the algorithm. From this table we can see that the number of points to process are almost the same when we compare between 1024 and 16384 sample points. Therefore, we can conclude that the mask enables us to have comparable performance when we applied the solver on 1024 and 16384 points, respectively. Of course, the function with 16384 points has 14 levels of multiscale decomposition while in the 1024 points case there are only 10 levels in the decomposition. But the coarse levels are relatively small and a difference among them do not degrade the performance substantially. As we said, the upper (fine) levels hold the major influence and contributes the most to the performance of the algorithm and

in this case the 1024 and 16384 points have almost identical mask sizes. Therefore, their performance is almost the same.

Table 3 describes the L_2 error (convergence error) when we apply Daubechies filters of length 8 and 16, respectively. The table shows how the L_2 error depends upon the number of iterations, whether the mask is used or not on the "s" (smooth) and "d" (derivative) coefficients and on the accuracy of the mask itself. We can see here and in all the other tables in the rest of the section that without the preconditioner we get accuracy of $10^{-1} - 10^{-2}$. We can see from table 3 that the application of the mask on "s" does not degrade the accuracy of the result or reduce the rate of convergence.

No. of iterations	Mask on s	Mask on d	Preconditioner	Mask accuracy	L_2 error D 8	L_2 error D 16
20	yes	yes	yes	14	$1.00 \cdot 10^{-07}$	$5.00 \cdot 10^{-10}$
20	no	yes	yes	14	$1.00 \cdot 10^{-07}$	$5.03 \cdot 10^{-10}$
20	no	no	yes	14	$8.41 \cdot 10^{-08}$	$3.77 \cdot 10^{-10}$
20	no	no	no	14	$9.78 \cdot 10^{-01}$	$9.77 \cdot 10^{-01}$
20	yes	yes	yes	10	$1.12 \cdot 10^{-07}$	$6.84 \cdot 10^{-10}$
20	no	yes	yes	10	$1.12 \cdot 10^{-07}$	$5.98 \cdot 10^{-10}$
20	yes	yes	yes	8	$3.72 \cdot 10^{-07}$	$8.16 \cdot 10^{-08}$
25	yes	yes	yes	14	$1.47 \cdot 10^{-09}$	$1.79 \cdot 10^{-12}$
30	yes	yes	yes	14	$2.18 \cdot 10^{-11}$	$3.96 \cdot 10^{-13}$
30	yes	yes	yes	10	$4.07 \cdot 10^{-08}$	$2.01 \cdot 10^{-10}$
35	yes	yes	yes	14	$3.25 \cdot 10^{-13}$	$4.02 \cdot 10^{-13}$

Table 3: 1-D example given by Eqs. (9.5) and (9.6): The L_2 error with respect to different parameters of the mask. Here we use Daubechies filters of length 8 and 16

9.2.2 Example 2

If the solver is applied to

$$\frac{\partial^2 u(x)}{\partial x^2} = K\, 2\, x\, (2\, x^2 \; - \; 3)\, e^{-x^2}, \tag{9.7}$$

whose the exact solution is :

$$u(x) = K\, x\, e^{-x^2} \tag{9.8}$$

then the L_2 error with respect to the number of iterations is given in table 4.

No. of iterations	with preconditioner	L_2 error
1	yes	$5.31 \cdot 10^{-1}$
5	yes	$2.03 \cdot 10^{-3}$
10	yes	$1.32 \cdot 10^{-6}$
10	no	$3.73 \cdot 10^{-1}$
15	yes	$3.58 \cdot 10^{-9}$
20	yes	$1.89 \cdot 10^{-11}$
30	yes	$1.47 \cdot 10^{-11}$
30	no	$2.95 \cdot 10^{-1}$
50	yes	$1.47 \cdot 10^{-11}$

Table 4: 1-D example: the L_2 error after the application of the algorithm on Eq. (9.7). It uses Daubechies filter of length 16 on 512 points, with masks on "s" and "d". The masks accuracy is 10^{-14}

9.3 2-D examples

9.3.1 Example 1

Assume

$$\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} = f(x,y), \tag{9.9}$$

where the r.h.s $f(x,y)$ is numerically computed such that the exact solution is:

$$u(x,y) = 10.0\, x\, e^{-250.0\,(x^2+y^2)}. \tag{9.10}$$

Table 5 describes the L_2 error when we apply Daubechies filters of length 8 on 64 points. The table show how the L_2 error depends upon the number of iterations, whether the mask is used or not on the "s" (smooth) and "d" (derivative) coefficients and on the accuracy of the mask itself. We can see from the table that the application of the mask on "s" does not degrade the accuracy of the result.

No. of iterations	Mask on s	Mask on d	Preconditioner	Mask accuracy	L_2 error D 8	L_2 error D 16
25	yes	yes	yes	10	$1.17 \cdot 10^{-5}$	$3.72 \cdot 10^{-6}$
30	yes	yes	yes	10	$8.43 \cdot 10^{-6}$	$2.64 \cdot 10^{-6}$
30	yes	yes	yes	14	$1.30 \cdot 10^{-9}$	$1.30 \cdot 10^{-9}$
40	yes	yes	yes	10	$5.22 \cdot 10^{-6}$	$2.37 \cdot 10^{-7}$
50	yes	yes	yes	10	$3.54 \cdot 10^{-6}$	$1.89 \cdot 10^{-7}$
50	no	yes	yes	10	$1.37 \cdot 10^{-6}$	$1.89 \cdot 10^{-7}$
50	yes	yes	yes	14	$3.02 \cdot 10^{-8}$	$1.30 \cdot 10^{-9}$
50	no	no	yes	14	$4.66 \cdot 10^{-12}$	$4.11 \cdot 10^{-11}$

Table 5: 2-D example given by Eqs. (9.9) and (9.10): The L_2 error with respect to different parameters of the mask. It uses Daubechies filter of length 8 (D 8) and length 16 (D 16) on 64 points

9.3.2 Example 2

$$u(x,y) = 10.0\, x\, e^{-250.0\,(x^2+y^2)} + 7.0\, x\, e^{-50.0\,(x^2+y^2)}. \tag{9.11}$$

No. of iterations	Mask on s	Mask on d	Preconditioner	Mask accuracy	L_2 error
25	yes	yes	yes	10	$1.24 \cdot 10^{-7}$
25	no	yes	yes	14	$4.95 \cdot 10^{-9}$
25	yes	yes	yes	14	$4.95 \cdot 10^{-9}$
40	yes	yes	yes	14	$3.81 \cdot 10^{-12}$
40	yes	yes	yes	10	$4.53 \cdot 10^{-8}$
40	no	yes	yes	10	$4.53 \cdot 10^{-8}$
50	yes	yes	yes	10	$2.94 \cdot 10^{-8}$
50	no	yes	yes	10	$2.94 \cdot 10^{-8}$
50	yes	yes	yes	14	$1.40 \cdot 10^{-12}$
50	yes	no	yes	14	$8.55 \cdot 10^{-13}$

Table 6: 2-D example given by Eq. (9.11): The L_2 error with respect to different parameters of the mask. It uses Daubechies filter of length 8 on 64 points.

9.3.3 Example 3

$$\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} + (\frac{1}{x^2} + y^2)\, u(x,y) = f(x,y), \tag{9.12}$$

where the r.h.s $f(x,y)$ is numerically computed such that the exact solution is:

$$u(x,y) = x\, e^{-(x^2+y^2)}. \tag{9.13}$$

Table 7 describes the results by using Eqs. (9.12) and (9.13).

No. of iterations	with preconditioner	L_2 error
1	yes	$6.59 \cdot 10^{-1}$
5	yes	$1.30 \cdot 10^{-1}$
10	yes	$1.07 \cdot 10^{-2}$
10	no	$6.70 \cdot 10^{-1}$
15	yes	$2.97 \cdot 10^{-4}$
20	yes	$6.93 \cdot 10^{-6}$
30	yes	$2.17 \cdot 10^{-8}$
30	no	$2.64 \cdot 10^{-1}$
50	yes	$2.37 \cdot 10^{-9}$

Table 7: 2-D example given by Eqs. (9.12) and (9.13): The L_2 error with respect to different number of iterations. It uses Daubechies filter of length 8 on 64 points, with masks on "s" and "d" where the mask accuracy is 10^{-14}

9.3.4 Example 4

$$\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} + e^y\, u(x,y) = f(x,y), \tag{9.14}$$

where the r.h.s $f(x,y)$ is numerically computed such that the exact solution is:

$$u(x,y) = x\, e^{-(x^2+y^2)}. \tag{9.15}$$

Table 8 describes the results by using Eqs. (9.14) and (9.15).

No. of iterations	with preconditioner	L_2 error
1	yes	$5.39 \cdot 10^{-1}$
5	yes	$3.92 \cdot 10^{-2}$
10	yes	$1.10 \cdot 10^{-3}$
10	no	$2.32 \cdot 10^{-1}$
15	yes	$1.39 \cdot 10^{-5}$
20	yes	$2.27 \cdot 10^{-7}$
30	yes	$8.03 \cdot 10^{-11}$
30	no	$8.36 \cdot 10^{-2}$
50	yes	$7.31 \cdot 10^{-10}$

Table 8: 2-D example given by Eqs. (9.14) and (9.15): The L_2 error with respect to different number of iterations. It uses Daubechies filter of length 8 on 64 points, with masks on "s" and "d" where the mask accuracy is 10^{-14}

9.3.5 2-D heat equation

The first order explicit scheme in time using the separability property has been applied to the following problem:

$$\begin{cases} \frac{\partial u(x,y,t)}{\partial t} - \left(\frac{\partial^2 u(x,y,t)}{\partial x^2} + \frac{\partial^2 u(x,y,t)}{\partial y^2}\right) = sin(2\pi x)\, cos(2\pi y) \\ u(x,y,0) = 0 \\ u(0,y,t) = u(1,y,t) \\ u(x,0,t) = u(x,1,t) \end{cases} \tag{9.16}$$

whose exact solution is:

$$u(x,y,t) = \frac{1}{8\pi^2}\left(1 - e^{-8\pi^2 t}\right) sin(2\pi x)\, cos(2\pi y). \tag{9.17}$$

Explicit schemes in time Let us consider the following time dependent problem

$$\frac{\partial u}{\partial t} + Au = f$$

with the following time discretization

$$u^n(x) = u(x, n\delta t)\ .$$

A family of explicit schemes in time can be defined using a parameter θ:

$$\theta\left(\frac{u^{n+1}-u^n}{\delta t}\right)+(1-\theta)\left(\frac{u^n-u^{n-1}}{\delta t}\right)+Au^n=f$$

or as

$$u^{n+1}=(2-\frac{1}{\theta}+\frac{\delta tA}{\theta})u^n+(\frac{1}{\theta}-1)u^{n-1}+\frac{\delta t}{\theta}f. \tag{9.18}$$

Depending on the value of the parameter θ one can obtain different methods. With θ equals to 1 we obtain the usual first order explicit scheme. It is a stable scheme which leads to results with low accuracy. When θ equals 0.5 one obtains a second order scheme that generates good results which are very unstable (the discretization step in time has to be very small in order to have a decay of the error). Between these two values of θ various schemes can be defined which lead to a good compromise between accuracy and stability. If θ is less than 0.5 then the method does not converge at all.

When the time step equals to 10^{-5} the L_2 error is $7.94\cdot10^{-4}$, when the time step equals to 10^{-4} the L_2 error is $7.94\cdot10^{-3}$.

This result is the same as the one obtained by Charton[11] with an Euler like scheme (implicit).

Better results can be obtained with a smaller time step, but this explicit scheme will remain a scheme of order 1 in time.

We are using an only first order explicit scheme in time. Better results can be achieved when using implicit scheme, but in this case we should also need to invert a matrix at each iteration in order to get a better accuracy.

9.4 3-D example

$$\frac{\partial^2u(x,y,z)}{\partial x^2}+\frac{\partial^2u(x,y,z)}{\partial y^2}+\frac{\partial^2u(x,y,z)}{\partial z^2}=f(x,y,z) \tag{9.19}$$

where the r.h.s $f(x,y,z)$ is numerically computed such that the exact solution is:

$$u(x,y,z)=10.0\,x\,e^{-250.0\,(x^2+y^2+z^2)}. \tag{9.20}$$

Table 9 describes the results by using Eqs. (9.19) and (9.20).

No. of iterations	with preconditioner	L_2 error on 32 points	L_2 error on 64 points
15	yes	$3.43\cdot10^{-5}$	$3.74\cdot10^{-5}$
20	yes	$4.34\cdot10^{-7}$	$7.15\cdot10^{-7}$
30	yes	$6.62\cdot10^{-11}$	$1.34\cdot10^{-10}$
30	no	$9.19\cdot10^{-3}$	$2.84\cdot10^{-2}$
50	yes	$5.05\cdot10^{-13}$	$2.89\cdot10^{-12}$

Table 9: 3-D example given by Eqs. (9.19) and (9.20): The L_2 error with respect to different number of iterations. It uses Daubechies filter of length 8 on 32 and 64 points, respectively, with masks on "s" and "d" where the mask accuracy is 14

10 Conclusion

The algorithm described in this paper is using 1-D computations enables to solve some problems in higher dimensions. The operators that can be studied have only to be separable.

Applications and extensions are already in progress. The separability of the computations (due to the separability of the operator) allows to consider each variable in a different way according to the problem under study. That means that irregular grids of discretization can be used.

The extension for solving problems connected to non separable operators is also in progress. A first possibility consists in considering operators whose kernel can be approximated by linear combinations of separable kernels.

The proof in section 7 demonstrates that the algorithm, which is described in this paper, can work also for solving PDEs with non constant coefficients. We believe that the same idea will be useful and applicable for solving non linear PDEs. Both topics are currently being explored.

Appendix

The 1-D - 3-D sparse formats which are used for manipulation and storage are based on the "Yale Sparse Matrix Package, I. The Symmetric Codes", Eisenstat et al., TR #112. The aim is to utilize a compact sparse data structure in the 1-D - 3-D computations in order to reduce the number of operations and data storage. The format proposed here keeps in memory only data that is above a given threshold. The description of the multidimensional sparse formats uses the "C" programming language conventions.

1-D storage

The following is a description of the data structures which are needed to manipulate sparse vector.

- int *size* : the size of the dense vector.
- int *non_zero* : the number of non-zero coefficients in the vector which are above a given threshold.
- int **array* : contains the values of the non-zero (above threshold) coefficients.
- int **row_p* : contains the locations (the row number) of the non-zero coefficients.

Example: The following vector,

$$\begin{bmatrix} 23 \\ 0 \\ 5 \\ 2 \\ 0 \\ 0 \\ 12 \\ 8 \end{bmatrix}$$

is described by :

$$\begin{aligned} size &\longrightarrow 8 \\ non\_zero &\longrightarrow 5 \\ *array &\longrightarrow 23, 5, 2, 12, 8 \\ *row\_p &\longrightarrow 0, 2, 3, 6, 7 \end{aligned}$$

2-D storage

The following is a description of the data structures which are needed to manipulate sparse matrices.

- int *n_row* : the number of rows.
- int *n_col* : the number of columns.
- int **mat* : contains the values of the non-zero (above threshold) coefficients.
- int **row_p* : the number of non-zero coefficients in each row (*row_p[i+1]-row_p[i]* is the number of non-zero (above threshold) coefficients in row *i*).
- int **col_p* : contains the locations of the non-zero (above threshold) coefficients in each row (column numbers).

Example: The following matrix,

$$\begin{bmatrix} 0 & 6 & 0 & 8 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 9 & 10 & 11 & 0 & 12 & 0 \\ 0 & 0 & 13 & 0 & 14 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 8 & 0 & 1 & 2 & 3 \end{bmatrix}$$

is described by :

$$\begin{aligned} n\_row &\longrightarrow 6 \\ n\_col &\longrightarrow 6 \\ *mat &\longrightarrow 6, 8, 9, 10, 11, 12, 13, 14, 8, 1, 2, 3 \\ *row\_p &\longrightarrow 0, 2, 2, 6, 8, 8, 12 \\ *col\_p &\longrightarrow 1, 3, 0, 1, 2, 4, 1, 3, 4, 5 \end{aligned}$$

Remark: the last coefficient in **row_p* is the number of non-zero (above threshold) values in the matrix.

3-D storage

The following is a description of the data structures which are needed to manipulate sparse three dimensional matrices.

- int *n_row* : the number of rows.
- int *n_col* : the number of columns.
- int *n_hei* : the number of heights.
- int **mat* : the values of the non-zero (above threshold) coefficients.
- int **hei_p* : gives some information about the number of non zero coefficients in each surface (*hei_p[i+1]-hei_p[i]* is the number of non-zero (above threshold) coefficients in surface i).
- int **row_p* : the number of non-zero (above threshold) coefficients in each row (the rows are numbered surface by surface).
- int **col_p* : contains the locations of the non-zero coefficients in each row (column numbers).

Example: The following cube,

$$\begin{bmatrix} 0 & 6 & 0 & 8 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 9 & 10 & 11 & 0 \end{bmatrix} \begin{bmatrix} 12 & 0 & 0 & 0 \\ 13 & 0 & 14 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 8 \end{bmatrix} \begin{bmatrix} 0 & 1 & 2 & 3 \\ 0 & 0 & 65 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 12 & 34 & 0 \end{bmatrix} \begin{bmatrix} 23 & 0 & 0 & 0 \\ 5 & 0 & 0 & 18 \\ 0 & 0 & 0 & 0 \\ 22 & 0 & 63 & 64 \end{bmatrix}$$

is described by :

$$\begin{aligned} n\_row &\longrightarrow 4 \\ n\_col &\longrightarrow 4 \\ n\_hei &\longrightarrow 4 \\ *mat &\longrightarrow 6,8,9,10,11,12,13,14,8,1,2,3,65,12,34,23,5,18,22,63,64 \\ *hei\_p &\longrightarrow 0,5,9,15,21 \\ *row\_p &\longrightarrow 0,2,2,2,5,6,8,8,9,12,13,13,15,16,18,18,21 \\ *col\_p &\longrightarrow 1,3,0,1,2,0,0,2,3,1,2,3,2,1,2,0,0,3,0,2,3 \end{aligned}$$

Remark: the last coefficient in **hei_p* is the number of non-zero (above threshold) values in the cube. The pointer **hei_p* could be removed since its coefficients are also in **row_p*.

Acknowledgments. The research of A. Averbuch, R. Coifman and M. Israeli was supported by U.S-Israel Binational Science Foundation grant # 92-00269/1.
The research of G. Beylkin was partially supported by ARPA grant F49620-93-1-04 74 and ONR grant N00014-91-J4037.
The research of P. Fischer was supported by Post-Doc Fellow, awarded by the Israeli Academy of Sciences, at the department of computer science, Tel Aviv University, Israel, in 1995-1996.

References

[1] Averbuch, A., Beylkin, G., Coifman, R., Israeli, M., *Multiresolution Solution of Elliptic and Parabolic PDEs*, The Samuel Neaman Workshop on Signal and Image Representation in Combined Space, Technion, Haifa, Israel, May 8-11, 1994: edited by Y. Zeevi, R. Coifman, pp. 341-360, Academic Press, 1998.

[2] Averbuch, A., Beylkin, G., Coifman, R., Fischer P., Israeli, M., *A wavelets based constrained Preconditioned Conjugate Gradient for elliptic problems*, Conference on Preconditioned Iterative Solution Methods for Large Scale Problems in Scientific Computations, May 27-29, 1997 University of Nijmegen, The Netherlands.

[3] A. Averbuch, M. Israeli, F. Meyer, *Speed vs. Quality in Low Bit-Rate Still Image Compression*, Image Communication, 1999.

[4] Bertoluzza, S., Maday, Y., Ravel, J.C., *A dynamically adaptive wavelet method for solving partial differential equations*, Comput. Methods Appl. Mech. and Eng., Vol. 116, pp. 293-299, 1994.

[5] Beylkin, G., *On the Representation of Operators in Bases of Compactly Supported Wavelet Bases*, SIAM J. Num. Anal., Vol. 6, pp. 1716-1740, 1992.

[6] Beylkin, G., *On wavelet-based algorithms for solving differential equations*, In John J. Benedetto and Michael W. Frazier, editors, *Wavelets: Mathematics and Applications*, pages 449–466. CRC Press, 1994.

[7] Beylkin, G., Coifman, R., Rokhlin, V., *Fast Wavelet Transforms and Numerical Algorithms I*, Comm. on Pure and Applied Mathematics, Vol. XLIV, pp. 141-183, 1991.

[8] Bony, J.M., *Clcul symbolique et propagation des singularities pour les equations aux derives partielles non-linaries*, Ann. Scient. E.N.S., 14, pp. 209-246, 1981.

[9] A. Brandt, A.A. Lubrecht, *Multilevel matrix multiplication and fast solution of integral equations*, J. Comp. Phys. 90, pp. 348-370, 1990.

[10] J.H. Bramble, J.E. Pasciak, J. Xu, *Parallel multilevel preconditioners*, Math. Comp. 55, pp. 1-22, 1990.

[11] Charton, P., Perrier, V., *Factorisation sur base d'ondelettes du noyau de la chaleur et algorithmes matriciels rapides associés*, C. R. Acad. Sci. Paris Sér. I, vol. 320, pp. 1013-1018, 1995.

[12] Chui, C.K., *An Introduction to Wavelets*, Academic Press, 1992.

[13] W. Dahmen, A. Kunoth, *Multilevel Preconditioning*, Numer.Math. 63, pp. 315-344, 1992.

[14] Daubechies, I., *Ten Lectures on Wavelets*, SIAM, 1992.

[15] Daubechies, I., *The Wavelet Tranform, Time-Frequency Localization and Signal Analysis*, IEEE Trans. on Information Theory, Vol. 36, No. 5, pp. 961-1005, September 1990.

[16] Daubechies, I., *Orthonormal Bases of Compactly Supported Wavelets*, Comm. on Pure and Applied Mathematics, Vol. XLI, pp. 909-996, 1988.

[17] Dorobantu, M., *Wavelet based algorithms for one-dimensional parabolic equations*, Technical report No. 9214, NADA, KTH, Stockholm, 1992.

[18] Engquist, B.,Osher, S. Zhong, S., *Fast wavelet based algorithms for linear evolution equation*, ICASE Report 92-14, April 1992.

[19] Fischer, P., Defranceschi, M., *Representation of the atomic Hartree-Fock equations in a wavelet basis by means of the BCR algorithm*, in: *Wavelets: Theory, Algorithms, and Applications*, (Chui, C., Montefusco, L., Puccio, L., Eds.; Academic Press Inc., San Diego), pp. 495-506, 1994.

[20] Fischer, P., Defranceschi, M., *Numerical solution of the Schrödinger equation in a wavelet basis for hydrogenlike atoms*, to appear in SIAM J. Num. Anal.

[21] Golub, G.H., Van Loan, C.H., *Matrix Computations*, (J. Hopkins University Press, Baltimore), 1983.

[22] S.J. Gortler, P. Schroder, M.F. Cohen, P. Hanrahan, *Wavelet Radiosity*, COMPUTER GRAPHICS Proceedings, Annual Conf., pp. 221-230, 1993.

[23] S. Jaffard, *Wavelet methods for fast resolution of elliptic problems*, SIAM J. Numer. Anal. 29, pp. 965-986, 1992.

[24] Mallat, S.G., *Multiresolution Approximations and Wavelet Orthonormal Bases for* $L^2(\mathbb{R})$, Trans. of AMS, Sept. 1989.

[25] Y. Meyer, *Wavelets and Operators I*, Cambridge University Press, 1989.

[26] Y. Meyer, R. Coifman, *Wavelets: Calderon-Zygmund and multilinear operators*, Cambridge University Press, 1997.

[27] F. Meyer, A. Averbuch, J-O Strömberg, *Fast Adaptive Wavelet Packet Image Compression*, submitted to IEEE Image Processing.

[28] P. Oswald, *Multilevel Finite Element Approximation: Theory and Applications*, B.G. Teubner Stuttgart, 1994.

[29] T. Tao. *On the almost everywhere convergence of wavelet summation methods*, preprint, 1995.

[30] J. Wang, *Convergence analysis of Schwarz algorithm and multilevel decomposition interative methods II: non-selfadjoint and indefinite elliptic problems*, SIAM J. Numer. Anal. 30, pp. 953-970, 1993.

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

[32] X. Zhang, *Multilevel Schwarz methods*, Numer. Math. 63, pp. 521-539, 1992.