International Journal of Pure and Applied Mathematics

Volume 44 No. 1 2008, 75-115

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]School of Mathematics and Computer Science
University Bordeaux 1
351 Cours de la Libération, Talence Cedex, 33405, FRANCE

[5]Faculty of Computer Science
Technion – Israel Institute of Technology
Haifa, 32000, ISRAEL

Abstract: An adaptive multidimensional algorithm for solving strictly elliptic PDEs is described. The main points of the present approach are: 1) For a given finite (but arbitrary) accuracy ϵ we find scales and locations of ϵ-significant wavelet coefficients by examining the wavelet decompositions of the right-hand side as well as the coefficients of the equation. In doing so we effectively construct subspaces for the solution (or "masks") for different accuracy thresholds. 2) We use a diagonally preconditioned "constrained" conjugate gradient (CG) method where the preconditioner is that of an extended problem with periodic bondary conditions to compute the actual coefficients of the approximate solution. We use sparse data structures in our implementations in order to take advantage of $O(N_s)$ complexity of the algorithm, where N_s is the number of significant coefficients required for a given accuracy. We present

Received: January 26, 2008

§Correspondence author

numerical examples in one, two and three spatial dimensions.

Professor Moshe Israeli passed away on February 18, 2007.
This paper is dedicated to his memory.

AMS Subject Classification: 35D05
Key Words: multidimensional wavelet decomposition, adaptive discretization, preconditioned constrained conjugate gradient, sparse implementation

1. Introduction

It is well known that linear elliptic operators do not create or move singularities but only modify them (see e.g. [20, 27, 32, 34]). The practical connection of such theoretical results with numerical computing has been uncovered in [6, 8], where it was shown that the Green's function for the elliptic operator is effectively sparse, i.e., in appropriate system of coordinates (e.g. wavelet bases) differs from a sparse operator by an operator of the norm less than $\epsilon > 0$, where ϵ is a finite but arbitrary accuracy. Since the solution is obtained by applying the Green's function to the right-hand side of the equation, the structure of singularities of the solution is fully encoded in the representation of the Green's function and that of the right-hand side. In addition, it was recognized (see e.g., [11, 6, 26] in slightly different contexts) that elliptic operators have diagonal preconditioners in wavelet bases.

These results opened up a possibility of constructing adaptive solvers for elliptic equations, where the selection of appropriate variables is done by examining the right-hand side and the coefficients of the operator by the wavelet transform (see [1, 2] and discussion below).

It was recognized in [1, 2] that the adaptive solution of elliptic PDE's presents a number of difficulties that general theorems used in some theoretical investigations do not capture. Specifically, we note that

1. Theoretical results on preconditioning are based on functional analysis (see e.g., [26]) and, although yield estimate $O(1)$ for the resulting condition number, give no indication of the size of the constant.

2. The boundary conditions may introduce rapid changes of the solution in the vicinity of the boundary whereas the coefficients of the equation do not display any indication of such behavior. Also, the condition number is affected

by the boundary conditions if the boundary conditions are incorporated into the disctretization of the operator.

3. Large condition number of the "multiplicative" component of the elliptic operator, may result in a special behavior of the solution near locations where this large condition number is attained. To illustrate this point consider

$$-\frac{d}{dx}(1+\sin(x)+\delta)\frac{d}{dx}u = f,$$

on $I = [-\pi, \pi]$ with periodic boundary conditions and smooth, non-oscillatory right-hand side f. The ratio $\max_I(1+\sin(x)+\delta)/\min_I(1+\sin(x)+\delta) = O(\delta^{-1})$ can be made as large as we wish by selecting δ, so that in the vicinity of, for example, $x = 0$ the solution will contain high frequency components. Yet, such behavior cannot be detected by the wavelet transform of the function $(1+\sin(x)+\delta)$ or f.

In one spatial dimension, there is an easy way to deal with such problems by considering instead the equation

$$-\frac{1}{1+\sin(x)+\delta}\frac{d}{dx}(1+\sin(x)+\delta)\frac{d}{dx}u = \frac{f}{1+\sin(x)+\delta}\,.$$

The function $f/(1+\sin(x)+\delta)$ will change rapidly near $x = 0$ and a wavelet transform will easily "identifies" this location. Unfortunately, in multiple dimensions such simple approach is not available but this type of problem may create a serious difficulty for an adaptive solver that does not explicitly take it into account.

4. It is important to understand the growth of *constants* with dimension in the complexity estimates of any algorithm

The goal of the approach in [1, 2] is to avoid the *search* of the locations of singularities in an adaptive algorithm. In order to avoid the search and in view of (1) and (2), we have selected in [1, 2] to split the problem into two, one inside the volume, another on the boundary. The advantage of such splitting addresses difficulties (1) and (2), since the volumetric problem can be made periodic in a larger box, thus avoiding (2) and making sure that the preconditioning is effective.

Such splitting is known classically but was considered impractical for computing due to apparently large number of operations. Following the results of [8], new complexity estimates make this approach practical. The only difficulty associated with such splitting is that in the case of variable coefficients one needs to compute the corresponding Green's function (again, for a periodic problem). We will address this issue in a separate paper and note here that this task is manageable.

In this paper we carry out the development to include dimensions two and three. We describe a method for efficient adaptive discretization and solution of elliptic PDEs forced by the right hand side (r.h.s) with regions of smooth (non-oscillatory) behavior and, possibly, localized regions with non-smooth (oscillatory) structures. We use the wavelet transform as a tool to provide sparse representations both of operator kernels and of functions with singularities or with rapid growth at particular locations. The wavelet basis allows us to detect the necessary coefficients to describe smooth sections of the r.h.s. and sharp transitions and singular points. As long as we avoid boundary effects (by splitting the problem) and do not have effects described in (3) (which is an intertesting problem on its own, associated with the so-called "loss of ellipticity" in problems of the theory of elasticity), one can argue that we have captured all locations where the solution will have a rapid change.

This paper should be viewed as a feasibility study of adaptive algorithms which focus the computational effort in regions where the r.h.s f has sharp transitions and singular points. Such algorithms have the desirable property that the number of operations performed is proportional to the number of significant coefficients in the wavelet expansion of the "inputs" of a given problem.

The essense of the adaptive algorithm is that it allows us to allocate the appropriate variables for the solution without actual solving of the equation. In other words, we identify a subspace for an approximate solution by representing the r.h.s. With a good preconditioner available for the periodic problem, we then use an iterative solver where we have to do only a fixed number of iterations.

The main steps in our approach are outlined below.

1. We reduce the computational cost upfront by imposing a mask derived from wavelet thresholding. Our algorithm, thus, reduces attention to a subspace associated with "large" coefficients within a mask pattern derived from thresholding the r.h.s. While the solution we seek does not have coefficients exactly zero outside such a mask, we can show that such a mask allows us to capture an accurate approximation. This mask can then be easily modified (expanded) to allow for greater precision, if necessary.

2. We use a preconditioned conjugate gradient (CG) iteration constrained to a subspace. Preconditioned CG iterations are widely known and used in solving elliptic problems. The important distinction of our approach is that the the unknowns are restricted to the subspace identified by the mask. This is an important feature since numerical experiments show that if not constrained to a subspace, intermediate approximations attained via CG tend to be dense

thus negating the advantage of such solver.

The subspace is identified in a very simple manner, namely, only elements of the vector present in the mask are used in the computation while other elements are ignored and never created.

We note again that in the wavelet basis diagonal preconditioners are available which render the condition number of elliptic operators to $O(1)$. More to the point, since we limit ourselves to solving periodic problem, the condition number is actually very small.

3. Fundamental to the success of the whole strategy is rapid computation of the preconditioner and of the constrained operator used in Step 2. This is possible because of the representation which allows us to exploit alternatively the standard and the non-standard form of operators. Each iteration of the PCG consists of applying the non-standard-form of an operator to the wavelet expansion of a function and, thus, translates to a multiplication of a sparse vector by a sparse matrix. The difficulty that we overcome here is that the non-standard form of the preconditioned operator is not sparse.

4. We fully utilize tensor product of operators that allows us to effectively control the growth of the number of operations with dimension. Usually, a one-dimensional kernel in the non-standard form has a matrix 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 two and three-dimensional problems with operators written as the sum (or the product) of one-dimensional operators, i.e., equations with separable operators. The simplest example is the Laplacian, $\Delta = \sum_{k=1}^{d} \frac{\partial^2}{\partial x_k^2}, x \in I\!R^d$. We present implementations in one, two and three dimensions using sparse data structures to take advantage of the algorithm.

For the interested reader we note that there is a vast literature on iterative solution of elliptic problems. For a specific example where finite elements are being used see [11]. Many more recent works relevant to the subject of our paper although with somewhat different goals and approaches were presented in [4, 14].

The paper is organized as follows. After the description of the problem given in Section 2, the solution strategy is introduced in Section 2.2 which include a preliminary heuristic justification of mask strategy and statement of the main result. Section 3 describes the 1-D - 3-D preconditioners. Section 4 outlines the algorithm. In Section 5 we estimate the number of operations for

the Poisson solver. Solutions for 1, 2 and 3-dimensional problems are described and illustrated by detailed numerical results in Section 6. Some theoretical background is given in Appendix I. Further extensions of our approach are briefly discussed in Section 7.

2. Theoretical Considerations

Initial results using the proposed method for the adaptive solution of elliptic problems in dimension $d = 1$ were reported in [1, 2]. In this paper we extend these results, namely, (i) we generalize the 1-D algorithm of [1, 2] to dimensions $d = 2, 3$ (and higher) via tensor product and, (ii) provide a justification for the use of masks in the solution process.

2.1. Problem Formulation

Let us start by briefly describing a general framework for solving adaptively linear elliptic problems with general boundary conditions (see also [1, 2]). 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,

$$\mathcal{L}u = -\sum_{i,j=1,\ldots,d} \left(a_{ij}(x)\ u_{x_i}\right)_{x_j} + b(x)\ u, \tag{2.2}$$

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

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

We generate a function f_{ext}, a smooth extension of f outside the domain $\mathbf{D}$, such that f_{ext} is periodic in a rectangular box $\mathbf{B}$, $\mathbf{D} \subset \mathbf{B} \subset \mathbf{R}^{\mathrm{d}}$, and $f = f_{ext}$ for $x \in \mathbf{D}$. Our goal is to construct an efficient adaptive method to solve

$$\mathcal{L}u = f_{ext} \quad x \in \mathbf{B} \tag{2.3}$$

with periodic boundary conditions. The resulting solution does not have the correct boundary values and the complete solution requires solving also the problem on the boundary using integral equations (see [1, 2]). We concentrate here on problem (2.3).

Our goal in solving (2.3) 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}.

2.2. Solution Strategy

Let us illustrate our approach by considering the Poisson's equation

$$\Delta u = f \quad x \in \mathbf{B} \tag{2.4}$$

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 $\mathbf{B}$.

Let us represent the source term f and the solution u in (2.4) in the wavelet basis,

$$f(x) = \sum_{j\le n}\sum_{\mathbf{k}}\sum_{\sigma} f^{\sigma}_{j,\mathbf{k}}\psi^{\sigma}_{j,\mathbf{k}}(x) + \sum_{\mathbf{k}} s^{f}_{n,\mathbf{k}}\varphi_{n,\mathbf{k}}(x), \tag{2.5}$$

$$u(x) = \sum_{j\le n}\sum_{\mathbf{k}}\sum_{\sigma} u^{\sigma}_{j,\mathbf{k}}\psi^{\sigma}_{j,\mathbf{k}}(x) + \sum_{\mathbf{k}} s^{u}_{n,\mathbf{k}}\varphi_{n,\mathbf{k}}(x), \tag{2.6}$$

where

$$f^{\sigma}_{j,\mathbf{k}} = \langle f, \psi^{\sigma}_{j,\mathbf{k}}\rangle,\ \ u^{\sigma}_{j,\mathbf{k}} = \langle u, \psi^{\sigma}_{j,\mathbf{k}}\rangle,\ \ s^{f}_{n,\mathbf{k}} = \langle f, \varphi_{n,\mathbf{k}}\rangle \text{ and } s^{u}_{n,\mathbf{k}} = \langle u, \varphi_{n,\mathbf{k}}\rangle. \tag{2.7}$$

We define the ϵ-accuracy subspace for f as

$$M^{\epsilon}_{r.h.s} = \mathbf{V}_n \bigcup \{\text{span}\{\psi^{\sigma}_{j,\mathbf{k}}\} \,|\, (j,\mathbf{k},\sigma) \ :\ |f^{\sigma}_{j,\mathbf{k}}| > \epsilon\}, \tag{2.8}$$

and observe that the ϵ-accuracy subspace for the solution

$$M^{\epsilon}_{sol} = \mathbf{V}_n \bigcup \{\text{span}\{\psi^{\sigma}_{j,\mathbf{k}}\} \,|\, (j,\mathbf{k},\sigma) \ :\ |u^{\sigma}_{j,\mathbf{k}}| > \epsilon\} \tag{2.9}$$

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

It follows from

Proposition 2.1. *Let*

$$u(x) = \sum_{j}\sum_{\mathbf{k}}\sum_{\sigma} u^{\sigma}_{j,\mathbf{k}}\psi^{\sigma}_{j,\mathbf{k}}(x) + \text{constant} \tag{2.10}$$

be the solution of

$$\Delta u = \psi^{\sigma'}_{j',\mathbf{k}'} \quad x \in \mathbf{B} \tag{2.11}$$

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, $|u^{\sigma}_{j,\mathbf{k}}| \ge \epsilon$, satisfy $|\mathbf{k}-\mathbf{k}'| \le \lambda$ and $|j-j'| \le \mu$.

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 $M^{\epsilon}_{r.h.s}$.

Since the solution u can be written as

$$u(x) = \int_{\mathbf{B}} G(x-y)\psi^{\sigma'}_{j',\mathbf{k}'}(y)dy, \tag{2.12}$$

where G is the Green's function for the periodic problem, the proof of the proposition follows from the results of [8]. It has been shown in [8] that in wavelet bases the Green's functions for elliptic problems (like the one in (2.12)) are well approximated in the operator norm by banded either standard or nonstandard forms for any $\epsilon > 0$. By replacing the Green's function in (2.12) by such banded approximation, we obtain the result where the size of the constants λ and μ is determined by the size of the bands of the Green's function.

According to Proposition 2.1, $M^{\epsilon}_{sol} \subset M_{\lambda,\mu}$. We note that estimating the subspace amounts to constructing a mask which contains indices of significant coefficients.

2.3. A Justification of Masking Strategy

There are three main features in our approach to solve (2.4):

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 wavelet bases [1, 7] which yields a condition number that is of $O(1)$. We will show in Section 3 how to construct in wavelet bases simple preconditioners for more general operators.

3. Constrained Iterative Solver. We use preconditioned Conjugate Gradient (CG) method restricted to the subspace estimated at Step 1. The CG method requires only a constant number of iterations due to the preconditioning at Step 2 and the cost of each iteration is proportional to the dimension of M^{ϵ}_{sol} provided that we use an appropriate sparse data structure to apply the matrix in the CG method (see below).

Instead of estimating $M_{\lambda,\mu}$ directly, we can also 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 can also 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.

Remark on Convergence. In order to demonstrate convergence of our method, we use results in [33], where it is shown that if P_ϵ is a projector on M^{ϵ}_{f}, the ϵ-accuracy subspace for f, then for any L^p function, $1 < p < \infty$,

$$\lim_{\epsilon \to 0} P_\epsilon f(x) \to f(x) \tag{2.13}$$

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

Our next step is to extend this approach to more general elliptic operators. We start by considering

Proposition 2.2. *Let*

$$u(x) = \sum_j \sum_{\mathbf{k}} \sum_{\sigma} u^{\sigma}_{j,\mathbf{k}} \psi^{\sigma}_{j,\mathbf{k}}(x) \tag{2.14}$$

be the solution of

$$\mathcal{L}u = \psi^{\sigma'}_{j',\mathbf{k}'} \quad x \in \mathbf{B} \tag{2.15}$$

with periodic boundary conditions, where $\mathcal{L}u = -\sum_{i,j=1,\ldots,d} \left(a_{ij}(x)\ u_{x_i}\right)_{x_j}$. *We assume that the coefficients are infinitely differentiable functions and*

$$|\xi|^2 \leq \sum_{i,j=1,\ldots,d} a_{ij}\xi_i\xi_j \leq C|\xi|^2,$$

where $C \geq 1$. *Then, for any* $\epsilon > 0$, *there exist constants* $\lambda > 0$ *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 proof is completely analogous to that of Proposition 2.1, where $G(x-y)$ is replaced by the Green's function $G(x, y)$ for the elliptic problem (2.15) (see [8]). 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 $M^{\epsilon}_{r.h.s}$.

Since the solution u can be written as

$$u(x) = \int_{\mathbf{B}} G(x, y)\psi^{\sigma'}_{j',\mathbf{k}'}(y)dy, \tag{2.16}$$

where G is the Green's function for the periodic problem, the proof of the proposition follows from the results of [8]. It has been shown in [8] that in wavelet bases the Green's functions for elliptic problems (like the one in (2.16)) are well approximated in the operator norm by banded either standard or nonstandard

forms for any $\epsilon > 0$. By replacing the Green's function in (2.16) by such banded approximation, we obtain the result where the size of the constants λ and μ is determined by the size of the bands of the Green's function. According to Proposition 2.1, $M^{\epsilon}_{sol} \subset M_{\lambda,\mu}$. We note that estimating the subspace amounts to constructing a mask which contains indices of significant coefficients.

Next we would like to consider operators $\mathcal{L}$ in (2.2). If $b \geq 0$ then Proposition 2.2 can be simply extended to include this case. However, if b is allowed to be negative, then the operator may be ill-conditioned since it may no longer be positive definite and its spectrum well separated from zero. Let us show that in this case the difficulty (if any) in inverting the operator and, thus, solving the system is limited to a particular coarse subspace. Moreover, we can still use our masking strategy on the fine subspaces.

Proposition 2.3. *Let*

$$u(x) = \sum_{j} \sum_{\mathbf{k}} \sum_{\sigma} u^{\sigma}_{j,\mathbf{k}} \psi^{\sigma}_{j,\mathbf{k}}(x) \tag{2.17}$$

be the solution of

$$\mathcal{L}u = \psi^{\sigma'}_{j',\mathbf{k}'} \quad x \in \mathbf{B} \tag{2.18}$$

with periodic boundary conditions, where $\mathcal{L} = -\sum_{i,j=1,\ldots,d} (a_{ij}(x)\ u_{x_i})_{x_j} + b(x)\ u$ *. We assume that the coefficients are infinitely differentiable functions,*

$$|\xi|^2 \leq \sum_{i,j=1,\ldots,d} a_{ij}\xi_i\xi_j \leq C|\xi|^2, \tag{2.19}$$

where $C \geq 1$ *and*

$$-2^{2j_0} < b\,. \tag{2.20}$$

Then, for any $\epsilon > 0$*, there exist constants* $\lambda > 0$ *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* $|j - j'| \leq \mu$*.*

Proof. If $b(x) = 0$ then the operator $\mathcal{L}$ in the wavelet representation can be preconditioned by the diagonal operator (see e.g. [26]) defined by

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

Let us show that if $b \neq 0$, then solving (2.18) can be reduced to finding a solution on a union of a "coarse" subspace V_{j_0} and a high frequency space defined as the $\epsilon-$"mask" of the right-hand side f.

Let us consider the subspace V_{j_0} (where j_0 is the same as in (2.20)) and consider $V^{\perp}_{j_0}$. Let $u = u_0 + u_1,\ u_0 \in V_{j_0},\quad u_1 \in V^{\perp}_{j_0}$ and, similarly, $f = f_0 + f_1$,

so that we rewrite (2.18) as:

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

Here f_0 or f_1 may be zero.

We claim that the operator $\mathcal{L}_1$, the projection of $\mathcal{L}$ onto $V_{j_0}^{\perp}$, is invertible on that subspace. First we observe that the norm of the preconditioner $\mathcal{P}$ on $V_{j_0}^{\perp}$ is less than 2^{-j_0-1}. Then it follows that

$$2^{2j_0}\|\mathcal{P}h\|_2^2 \leq \frac{1}{4}\|h\|_2^2 \quad \text{if} \quad h \in V_{j_0}^{\perp}. \tag{2.21}$$

On the other hand, we have from (2.19)

$$\|h\|^2 - 2^{2j_0}\|\mathcal{P}h\|^2 \leq \quad \langle \mathcal{P}\mathcal{L}\mathcal{P}h, h\rangle \,,$$

where we used that $\mathcal{P}$ is the perfect preconditioner for $\mathcal{L}$. It follows now from (2.21) that $\mathcal{L}_1^p = \mathcal{P}\mathcal{L}_1\mathcal{P}$ is invertible, since $\mathcal{L}_1 = \mathcal{L}$ for $h \in V_{j_0}^{\perp}$.

The solvability of (2.18) is, thus, reduced to showing the solvability of

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

on the low frequency subspace V_{j_0}.

We now discuss the masking procedure for inverting $\mathcal{L}_1^p$. We can view $\mathcal{L}_1^p$ as the restriction to $V_{j_0}^{\perp}$ of the operator $L^0 = P_{j_0} + \mathcal{L}_1^p$, where P_{j_0} is the projector on V_{j_0}. We essentially set the operator to be the identity on V_{j_0}. The operator L^0 is an invertible Calderon-Zygmund operator (see [29]) and we write the inverse as

$$Tf = \int k(x,y) f(y) dy \,, \tag{2.22}$$

where $k(x,y)$ is a Calderon-Zygmund kernel (see [8, 29, 30]). We start by assuming that $T(1) = 0$ in order to justify our masking procedure, and then show that this condition is "effectively" satisfied in our case.

As shown in [30] the operator T is a self-adjoint Calderon-Zygmund operator whose matrix in the standard form is sparse, and has the "finger " structure with exponential decay of 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} such that $\|f - M^{\epsilon}(f)\| \leq \epsilon$ and $\|Tf - M^{\epsilon}(Tf)\| \leq \epsilon$. Therefore it follows that $\|M^{\epsilon}L^0M^{\epsilon}Tf - f\| \leq 2\epsilon$ and that it is enough to solve the equation $M^{\epsilon}L^0M^{\epsilon}u_1 = f_1$ 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 the effect of multiscale interactions, it is easier to work in the nonstandard form. Let α, β, and γ denote the blocks of the nonstandard form (see [8] for definitions). Then all scales are independent and the mixing occurs only through the s_k coefficients (of the projections on V_j) and the β and γ matrices.

Our goal now is to prove that although the coefficients s^j_k are not small and, thus the corresponding matrices are dense, it suffices to consider in the nonstandard form only those labels (j,k) near the corresponding labels of d^j_k (projections on wavelet subspaces) used to define the mask M^ϵ.

We start by observing that $\sum_{k'} \beta^j_{k,k'} = 0$, since $T(1) = 0$. In fact, since $\sum_{k'} \varphi_{j,k'}(y) =$const. and based on equation (8.6)

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

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

Therefore,

$$\sum \beta^j_{k,k'} s^j_{k'} = \sum B^j_{k,k'} (s^j_{k'+1} - s^j_{k'})$$

where $B^j_{k,k'} = \sum_{l>k'} \beta^j_{k,l}$.

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

$$|B^j_{k,k'}| \le \frac{C}{1+|k-k'|^{m-1}}\,.$$

Repeating the summation by parts argument, we get

$$\sum \beta^j_{k,k'} s^j_{k'} = \sum b^j_{k,k'} (s^j_{k+1} - 2s^j_{k'} + s^j_{k'-1}).$$

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

The coefficients

$$\delta^j_k = s^j_{k'+1} - 2s^j_{k'} + s^j_{k'-1} = \langle \varphi_{j,k'+1} - 2\varphi_{j,k'} + \varphi_{j,k'-1}, f \rangle$$

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

The matrix $b^j_{k,k'}$ 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 $T(1)$ are automatically sat-

isfied in our case. We recall that

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

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.

3. Preconditioners

It is well understood that a strictly elliptic operator $\mathcal{L}$ preconditioned by $\mathcal{P}$, $\mathcal{P}\mathcal{L}\mathcal{P}$, where

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

attains the condition number $O(1)$. However, the resulting condition number can still be large since its size is not accounted for by the theory. The theory only states that the refinement of the problem does not change the condition number of the resulting linear system once such system is sufficiently large [26].

It has been noticed, however, that for periodic problems the condition number of the preconditioned operator is actually small [7]. Let us first consider a simple 1D example and demonstrate how to construct a diagonal preconditioner for the sum of operators $-\Delta$+Const in wavelet bases for the periodic problem.

We observe that if A and B are diagonal operators with diagonal entries a_i and b_i, where $a_i + b_i \neq 0$, then the diagonal operator with entries $1/(a_i + b_i)$ is an ideal preconditioner. In our case, the operator $-\Delta$ is not diagonal, but it has a good diagonal preconditioner, (4.4), in wavelet bases (4.4). Let us use this idea also for the purpose of constructing a preconditioner for $-\Delta + C$, where C is a positive constant. We note that in wavelet bases the identity operator remains unchanged. We restrict $I(C)$, where I is the identity operator, to the subspace

$$\bigoplus_{1 \le j \le n} \mathbf{W}_j \tag{3.1}$$

and construct a preconditioner on this subspace.

We obtain

$$\mathcal{P}_{il} = \frac{\delta_{il}}{\sqrt{2^{-2j} + C}}\,, \tag{3.2}$$

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 $\mathcal{P}_{nn} = 1/\sqrt{2^{-2n} + C}$. The square root appears in (3.2) in order to symmetrize the application of the preconditioner (as shown above). In Table 1 we illustrate the effectiveness of preconditioning of the operator $-\frac{d^2}{dx^2} + C$ by

C	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 $-\frac{d^2}{dx^2} + C$ in the basis of Daubechies' wavelets with six vanishing moments. There are 8 scales and the matrix size is 256×256.

the diagonal matrix (3.2).

If we consider instead 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 such scales the function $V(x)$ does not change significantly over the support of the wavelets, we may consider the diagonal operator V^{diag},

$$\left(V^{diag}\psi^{\sigma}_{j,\mathbf{k}}\right) = V(x_{j,\mathbf{k}})\psi^{\sigma}_{j,\mathbf{k}}, \tag{3.3}$$

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.

It appears that the following procedure provides us with very good preconditioners for periodic problems on wavelet subspaces. As the first step we compute the wavelet decomposition of the operator and consider the diagonal of the standard form, D_s. For positive definite elliptic operators all elements of D_s are positive. We then set $\mathcal{P} = D_s^{-1/2}$ as the preconditioner.

For many iterative methods, the 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 iterative error in the approximate solution U^m after m iterations is given by:

$$\|U - U^m\|_2 \leq \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^m \|U\|_2 \,. \tag{3.4}$$

Here U is the converged solution. In order to improve the convergence, a standard practice is to consider solving an equivalent linear system,

$$PAP^T x = Pb \,, \tag{3.5}$$

where P satisfies two requirements:

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

— The application of P to an arbitrary vector must be rapidily computable.

Since we do not want in general to compute the condition number of the preconditioned operator, the number of iterations required by the iterative methods to converge is a good indication of the efficiency of the preconditioning.

In the wavelet basis, the second derivative being almost diagonal, a good diagonal preconditioner for this operator can be easily defined by:

$$P_{il} = \delta_{il} 2^j, \quad 1 \leq j \leq n, \quad P_{NN} = 2^n, \tag{3.6}$$

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.

3.1. 2-D and 3-D Preconditioners

Let us 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) of a function (Figures 1 and 2).

In the 1-D case (see Figure 1), the previously defined diagonal preconditioner is applied 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 certain constant, which we now derive. 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

$$P(j, j') = 2^{\min(j,j')}, \quad 1 \leq j, j' \leq n. \tag{3.7}$$

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

$$P(j, j', j'') = 2^{\min(j,j',j'')}, \quad 1 \leq j, j', j'' \leq n. \tag{3.8}$$

In short, to precondition, all the wavelet coefficients which belong to the same

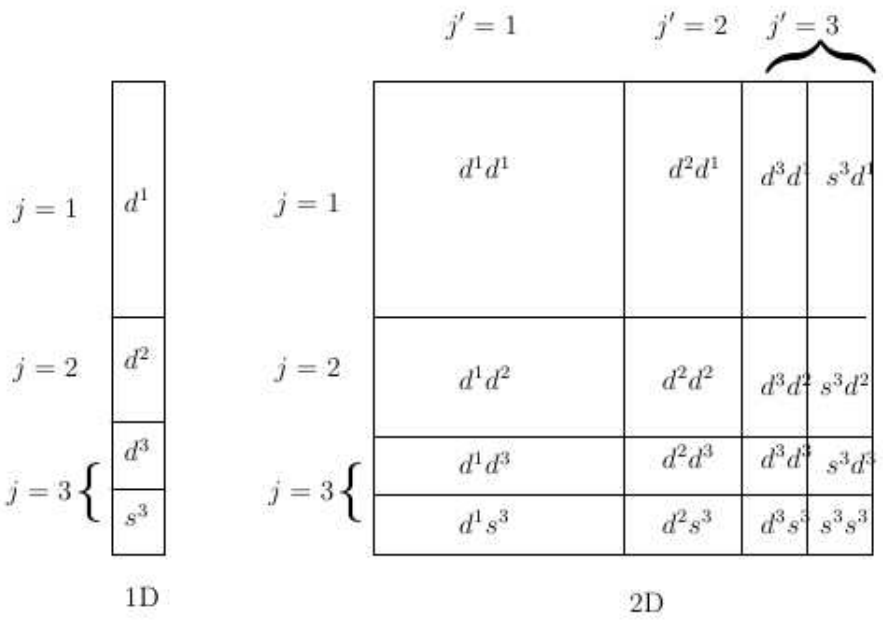

Figure 1: 1-D and 2-D wavelet tensor product (standard decomposition)

box are multiplied by the factors given in (3.7)-(3.8), depends on dimensions.

4. Outline of the Algorithm

4.1. 1-D Algorithm

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

$$L_0 = P_0 \, \Delta \, P_0, \tag{4.1}$$

and L_s and L_{ns} are its standard (s-form) and non-standard forms (ns-form) [8], respectively.

One of the difficulties in solving (2.3) stems from the inherently large condition number of the linear system resulting from the discretization of (2.3). As it was shown in [6, 7], 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.

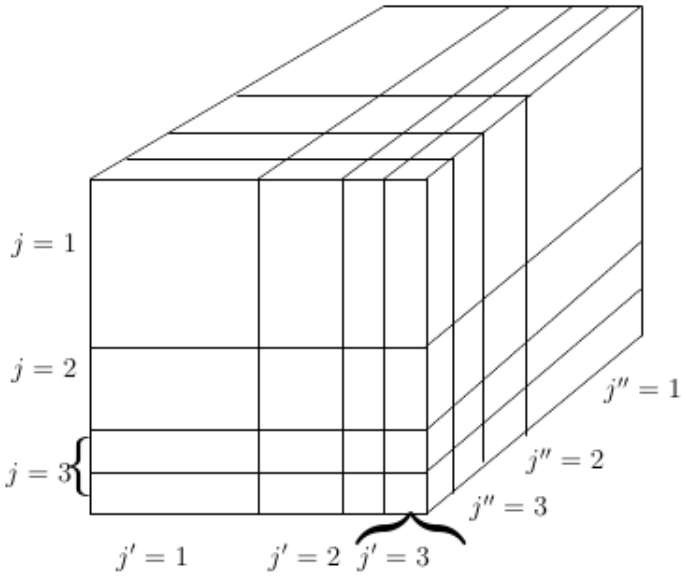

Figure 2: 3-D tensor wavelet organization

In [7] 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. [6]).

We develop a constrained preconditioned CG algorithm for solving (2.3) 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 (2.3) in the wavelet system of coordinates

$$L_{ns} u_w = f_w, \tag{4.2}$$

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 [8]).

Let us rewrite (4.2) using the preconditioner $\mathcal{P}$ as

$$\mathcal{P}\, L_{ns}\, \mathcal{P} v = \mathcal{P} f_w, \tag{4.3}$$

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

$$\mathcal{P}_{il} = \delta_{il} 2^j \,, \tag{4.4}$$

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 [7]). This allows us to solve (4.3) on the range of the operator,

$$\bigoplus_{1 \le j \le n} \mathbf{W}_j \,, \tag{4.5}$$

where the linear system (4.3) is well conditioned.

Constrained Iterative Solver. In order to solve (4.3) we apply the Conjugate Gradient method constrained to the subspace $M_{\lambda,\mu}$. Without such constraint 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 2.3.

4.2. 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 Figure 4):

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.

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 tensor product wavelets enables us to exploit the 1-D algorithm [1] in the setting of 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.

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 case is treated similarly while taking into consideration the third direction (height).

5. Complexity Analysis

5.1. Poisson Solver: 1-D Pseudo Code

Assume that the r.h.s is of size $N = 2^J$, so we can decompose it into J levels. The basic 1-D structure in the algorithm consists of an array of pointers of size $2J + 1$. Each odd entry in this array, $2j - 1, \quad j = 1, \ldots, J$, is a pointer to a sparse vector of size $N/2^j$, where we store the "s" (averaged) wavelet coefficients of scale j. Each even entry in this array, $2j, \quad j = 1, \ldots, J$, is a pointer to a sparse vector of size $N/2^j$, where we store the "d" (derivative) wavelet coefficients of scale j. The 1-D algorithm uses only this basic structure for $w_t, w_r, w_p, w_{p_1}, w_s, w_x$, which are used in the pseudo code.

1. Preprocessing: Second derivative in the NSF form - Compute 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.-
1. Find the wavelet coefficients of the exact solution u by multiresolution decomposition of u.
2. Transform the coefficients of the exact solution to a sparse data structure by thresholding the wavelet coefficients.
3. Reconstruct of the r.h.s into a sparse data structure.
4. Apply the second derivative operator which is multiplication of the vector by ns-form.

Wavelet decomposition of the r.h.s - gives the "d" coefficients. They are stored it in w_t

Generate the mask for the "d" coefficients - this mask is determined by thresholding w_t.

Generate the mask for the "s" coefficients - this mask is also determined by the mask of the "d" coefficients.

Apply the preconditioner to the wavelet coefficients w_t.

Copy the w_t into w_r and compute the L_2 norm of w_ri, denoted by ρ

2. The main iteration loop of the CG: First iteration - copy w_r into w_p, and w_r into w_{p_1}

If other iterations compute $\beta = \frac{\rho}{\rho_1}$. Sparse linear combination: $w_r + \beta w_p$ into w_{p_1}. Copy w_{p_1} into w_p.

Apply the 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 the preconditioner on w_s

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

Compute $\alpha = \frac{\rho}{r}$

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

Copy w_q into w_x.

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

Copy w_q into w_r

Save $\rho_1 = \rho$

Copy w_{p_1} into w_p

Repeat until convergence

5.2. Estimation of the Number of Operations

The Poisson equation is solved using the Conjugate Gradient method [24] 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 2.2. The wavelet transform of the solution consists of N_s significant coefficients in the masks which are concentrated near the singularities (see Figure 3 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 [31], 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 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 requires $4lN_s+3\cdot 2N_s = (4l+6)N_s$ operations. Therefore, the total number of operations for N_{iter} is $N_{iter} \cdot (4l+6)N_s$. To get ϵ-accuracy we need $N_{iter} \cdot (4l+6)N_s \log \epsilon$ 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 $N_{iter} \cdot (4l + 6)N_s^d \log \epsilon$.

The sizes of the masks are independent of the sampling rate, as can be seen in Table 2. There, even with "super" spatial sampling rate we still get a similar size for the masks.

The speed of the algorithm compares well with the usual methods for elliptic problems such as multigrid.

6. Numerical Results

The algorithm was tested on the 1-D, 2-D and 3-D Poisson equations. All the computations were performed on predetermined masks according to the required accuracy. The results obtained for the 2-D and 3-D cases support the idea that application of the 1-D decomposition in each direction is useful for solving problems in higher dimensions. The preconditioner was also tested on the 1-D, 2-D and 3-D cases. In Section 3 extensive experiments were reported concerning the efficiency of the preconditioner for the modified Helmholtz equation. As it will be shown in the sequel, the preconditioner gives very good results after only few iterations.

For the interpretation of the numerical results below, we note that the er-

ror of the solution has several components. First of all, the truncation error is introduced by using masks selected according to the accuracy requirements. It is the behaviour of the error of the solution as a function the accuracy requirements in constructing masks that we want to illustrate in our examples. However, we note that as we approach the roundoff error (double precision in our case), the roundoff error dominates as one would expect. Also, the functions in our examples are only approximately periodic, thus introducing an error if the Gaussian does not decay sufficiently fast in the interval specified. This error can be avoided by periodizing the test functions but our selection of the test functions is sufficient to illustrate the point.

We observe that the error of the solution is only weakly dependent on the input function. We also observe that it is weakly dependent on the basis functions and on the scaling parameter. Thus, the rate of convergence (number of iterations) is the most important issue in our method.

For this reason we illustrate the convergence rate with and without preconditioning. As we will see the error drops down to truncation error levels once the iteration converges. This happens if we compute the right-hand side by substitution of the solution in the left hand side thus removing the periodization error.

6.1. Model Problems

The algorithm was tested on elliptic problems such as the constant-coefficient Poisson equation.

$$\sum_{k=1}^{d} \frac{\partial^2 u(x)}{\partial x_k^2} = f(x), \quad x \in I\!R^d, \quad d = 1, \ldots, 3\,. \tag{6.1}$$

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).

6.2. 1-D Examples

6.2.1. Example 1

Assume we have

$$\frac{\partial^2 u(x)}{\partial x^2} = f(x)\,, \tag{6.2}$$

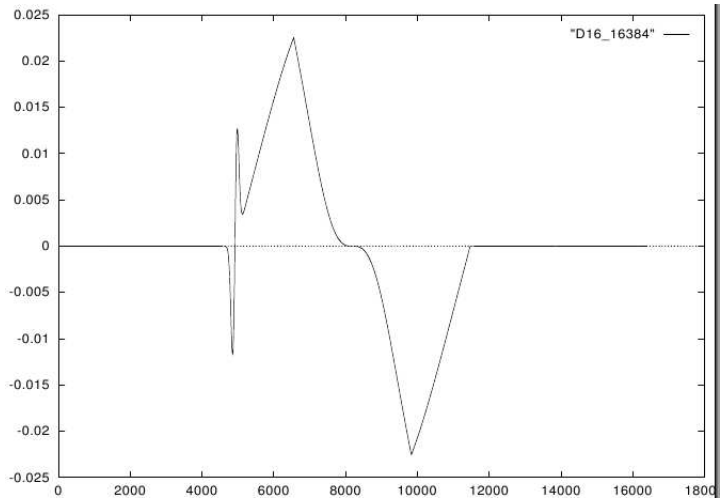

Figure 3: Graph of $u(x)$, the 1-D function, given by equation (6.3) on 16384 points

where the r.h.s $f(x)$ is numerically computed with the wavelet derivative operator such that the exact analytic solution is

$$u(x) = xe^{-256x^2} + 20(x + 0.2)e^{-32000(x+0.2)^2}\,. \tag{6.3}$$

This function, which has many length scales, is plotted Figure 3 using 16384 points.

We apply the masking algorithm on the r.h.s $f(x)$ which was sampled with 1024 and 16384 points, respectively. We use Daubechies's filter of length 16 (eight vanishing moments). After the multiscale decomposition using 1024 and 16384 points, the transformed function has four distinct masks in the first level, two and four respectively in the second level, etc, as it described in Table 2. This table exhibits all the related information about the masks in each sampling rate (1024 vs. 16384 points). This information contains: the number of masks in each scale, the starting position (the left most) and the right most point of each mask, the length of each mask in each level of the multiresolution.

The masks are located around points where the r.h.s has sharp variations. Table 2 describes the structure of the masks obtained for the solution $u(x)$ (equation (6.3)). These are the masks that were determined by the application of thresholding on the "d" coefficients. The same masks were used for the "s" coefficients. The length and the number of masks in the first 3 to 4 finest levels of the decomposition control the performance of the algorithm. This

Level	Daubechies 16			
	1024 points		16384 points	
	start and end locations of the mask	length of the mask	start and end locations of the mask	length of the mask
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 equation (6.3) using Daubechies filter of length 16. 'full' means that the whole scale was taken. The masks' accuracy is 10^{-14}.

table shows that the number of coefficients to be processed in the first 4 scales is almost the same for the case of 1024 and 16384 sample points. We may conclude that the masks enable the solver to have comparable performance for 1024 and 16384 points. 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 small in size and not degrade the performance substantially. In conclusion the performance is almost the same in the two cases.

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 this section

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

Table 3: 1-D example given by equation (6.3): The L_2 error with respect to different parameters of the mask. Here we use Daubechies filters of lengths 8 and 16. The last entry in the table shows the accuracy without the application of the preconditioner.

that without the preconditioner we get low 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.

In all tables of this section, "yes" and "no" indicate whether or not the mask and/or preconditioner were used.

6.2.2. Example 2

When the solver is applied to

$$\frac{\partial^2 u(x)}{\partial x^2} = K\, 2\, x\, (2\, x^2\, -\, 3)\, e^{-x^2}, \tag{6.4}$$

where the exact solution is:

$$u(x) = K\, x\, e^{-x^2}, \tag{6.5}$$

then the L_2 error dependence on the number of iterations is given in Table 4.

No. of iterations	with preconditioner?	L_2 error
10	no	$3.73 \cdot 10^{-1}$
30	no	$2.95 \cdot 10^{-1}$
1	yes	$5.31 \cdot 10^{-1}$
5	yes	$2.03 \cdot 10^{-3}$
10	yes	$1.32 \cdot 10^{-6}$
15	yes	$3.58 \cdot 10^{-9}$
20	yes	$1.89 \cdot 10^{-11}$
30	yes	$1.47 \cdot 10^{-11}$
50	yes	$1.28 \cdot 10^{-12}$

Table 4: 1-D example: the L_2 error after the application of the algorithm on equation (6.4). It uses Daubechies filter of length 16 on 512 points, with masks on "s" and "d". The masks accuracy is 10^{-14}.

6.3. 2-D Examples

6.3.1. Example 1

Assume

$$\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} = f(x,y), \tag{6.6}$$

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

$$u(x,y) = 10.0\, x\, e^{-50.0\,(x^2+y^2)}. \tag{6.7}$$

Table 5 describes the L_2 error when we apply Daubechies filters of length 8 and 16 on 64 and 128 points. The table shows how the L_2 error depends on 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.

Assume that the r.h.s $f(x,y)$ of (6.6) is numerically computed such that the exact solution is:

$$u(x,y) = 10.0\, x\, e^{-250.0\,(x^2+y^2)}. \tag{6.8}$$

6.3.2. Example 2

$$u(x,y) = 10.0\, x\, e^{-250.0\,(x^2+y^2)} + 7.0\, x\, e^{-50.0\,(x^2+y^2)}. \tag{6.9}$$

No. of iterat-ions	Mask on s?	Mask on d?	Precond-itioner?	Mask accuracy (threshold)	64 points L_2 error D 8	64 points L_2 error D 16	128 points L_2 error D 8	128 points L_2 error D 16
25	yes	yes	yes	10	$2.25 \cdot 10^{-6}$	$7.84 \cdot 10^{-8}$	$1.43 \cdot 10^{-6}$	$1.47 \cdot 10^{-7}$
30	yes	yes	yes	10	$1.49 \cdot 10^{-6}$	$5.55 \cdot 10^{-8}$	$9.60 \cdot 10^{-7}$	$1.38 \cdot 10^{-7}$
40	yes	yes	yes	10	$9.47 \cdot 10^{-7}$	$7.58 \cdot 10^{-8}$	$5.19 \cdot 10^{-7}$	$2.88 \cdot 10^{-7}$
50	yes	yes	yes	10	$6.24 \cdot 10^{-7}$	$1.14 \cdot 10^{-7}$	$4.07 \cdot 10^{-7}$	$2.85 \cdot 10^{-7}$
50	no	yes	yes	10	$6.25 \cdot 10^{-7}$	$1.02 \cdot 10^{-7}$	$4.10 \cdot 10^{-6}$	$2.18 \cdot 10^{-7}$
20	yes	yes	yes	14	$5.65 \cdot 10^{-8}$	$3.40 \cdot 10^{-8}$	$2.89 \cdot 10^{-7}$	$6.52 \cdot 10^{-8}$
30	yes	yes	yes	14	$4.78 \cdot 10^{-8}$	$1.23 \cdot 10^{-8}$	$1.47 \cdot 10^{-7}$	$5.12 \cdot 10^{-8}$
40	yes	yes	yes	14	$4.38 \cdot 10^{-8}$	$5.25 \cdot 10^{-9}$	$7.11 \cdot 10^{-8}$	$4.56 \cdot 10^{-9}$
50	yes	yes	yes	14	$4.07 \cdot 10^{-8}$	$4.57 \cdot 10^{-9}$	$5.51 \cdot 10^{-8}$	$2.15 \cdot 10^{-9}$
50	no	no	yes	14	$1.91 \cdot 10^{-12}$	$9.50 \cdot 10^{-13}$	$6.20 \cdot 10^{-12}$	$1.05 \cdot 10^{-11}$

Table 5: 2-D example given by equations (6.6) and (6.7): 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 and 128 points.

No. of iterat-ions	Mask on s?	Mask on d?	Precond-itioner?	Mask accuracy (threshold)	64 points L_2 error D 8	64 points L_2 error D 16	128 points L_2 error D 8	128 points L_2 error D 16
25	yes	yes	yes	10	$7.19 \cdot 10^{-6}$	$1.84 \cdot 10^{-6}$	$5.48 \cdot 10^{-6}$	$3.78 \cdot 10^{-7}$
30	yes	yes	yes	10	$5.07 \cdot 10^{-6}$	$1.33 \cdot 10^{-6}$	$3.77 \cdot 10^{-6}$	$2.96 \cdot 10^{-7}$
40	yes	yes	yes	10	$3.49 \cdot 10^{-6}$	$8.00 \cdot 10^{-7}$	$2.25 \cdot 10^{-6}$	$2.34 \cdot 10^{-7}$
50	yes	yes	yes	10	$2.45 \cdot 10^{-6}$	$5.50 \cdot 10^{-7}$	$1.30 \cdot 10^{-6}$	$2.10 \cdot 10^{-7}$
50	no	yes	yes	10	$2.43 \cdot 10^{-6}$	$5.73 \cdot 10^{-7}$	$1.30 \cdot 10^{-6}$	$2.06 \cdot 10^{-7}$
20	yes	yes	yes	14	$8.01 \cdot 10^{-8}$	$7.20 \cdot 10^{-7}$	$5.95 \cdot 10^{-7}$	$5.25 \cdot 10^{-8}$
30	yes	yes	yes	14	$7.12 \cdot 10^{-8}$	$6.49 \cdot 10^{-7}$	$5.46 \cdot 10^{-7}$	$3.98 \cdot 10^{-8}$
40	yes	yes	yes	14	$5.12 \cdot 10^{-8}$	$4.32 \cdot 10^{-7}$	$3.86 \cdot 10^{-7}$	$2.48 \cdot 10^{-8}$
50	yes	yes	yes	14	$3.02 \cdot 10^{-8}$	$2.64 \cdot 10^{-8}$	$1.79 \cdot 10^{-7}$	$1.52 \cdot 10^{-9}$
50	no	no	yes	14	$4.66 \cdot 10^{-12}$	$4.11 \cdot 10^{-11}$	$9.12 \cdot 10^{-12}$	$3.34 \cdot 10^{-11}$

Table 6: 2-D example given by equations (6.6) and (6.8): 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 and 128 points.

The results are given in Table 7.

6.4. 3-D Example

$$\frac{\partial^2 u(x,y,z)}{\partial x^2} + \frac{\partial^2 u(x,y,z)}{\partial y^2} + \frac{\partial^2 u(x,y,z)}{\partial z^2} = f(x,y,z)\,, \tag{6.10}$$

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

No. of iterations	Mask on s?	Mask on d?	Precond-itioner?	Mask accuracy (threshold)	64 points L_2 error D 8	64 points L_2 error D 16	128 points L_2 error D 8	128 points L_2 error D 16
25	yes	yes	yes	10	$4.55 \cdot 10^{-6}$	$6.60 \cdot 10^{-7}$	$4.51 \cdot 10^{-6}$	$6.47 \cdot 10^{-7}$
30	yes	yes	yes	10	$3.43 \cdot 10^{-6}$	$5.20 \cdot 10^{-7}$	$3.34 \cdot 10^{-6}$	$4.56 \cdot 10^{-7}$
40	yes	yes	yes	10	$1.68 \cdot 10^{-6}$	$3.29 \cdot 10^{-7}$	$1.59 \cdot 10^{-6}$	$3.08 \cdot 10^{-7}$
50	yes	yes	yes	10	$7.95 \cdot 10^{-7}$	$2.73 \cdot 10^{-7}$	$6.35 \cdot 10^{-7}$	$2.13 \cdot 10^{-7}$
50	no	yes	yes	10	$7.91 \cdot 10^{-7}$	$1.56 \cdot 10^{-7}$	$6.31 \cdot 10^{-7}$	$2.20 \cdot 10^{-7}$
30	yes	yes	yes	14	$3.00 \cdot 10^{-7}$	$6.88 \cdot 10^{-8}$	$2.06 \cdot 10^{-7}$	$8.21 \cdot 10^{-8}$
50	yes	yes	yes	14	$4.50 \cdot 10^{-8}$	$2.64 \cdot 10^{-9}$	$6.30 \cdot 10^{-8}$	$1.82 \cdot 10^{-9}$
50	no	no	yes	14	$2.56 \cdot 10^{-12}$	$1.26 \cdot 10^{-12}$	$2.09 \cdot 10^{-12}$	$8.50 \cdot 10^{-12}$

Table 7: 2-D example given by equations (6.6) and (6.9): 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 and 128 points

No. of iterations	Mask on s?	Mask on d?	Precond-itioner?	Mask accuracy (threshold)	32 points L_2 error D 8	32 points L_2 error D 16	64 points L_2 error D 8	64 points L_2 error D 16
15	yes	yes	yes	10	$9.88 \cdot 10^{-6}$	$1.70 \cdot 10^{-5}$	$6.56 \cdot 10^{-5}$	$1.30 \cdot 10^{-5}$
20	yes	yes	yes	10	$7.26 \cdot 10^{-6}$	$6.29 \cdot 10^{-6}$	$3.26 \cdot 10^{-5}$	$3.81 \cdot 10^{-6}$
30	yes	yes	yes	10	$3.11 \cdot 10^{-6}$	$2.44 \cdot 10^{-6}$	$1.44 \cdot 10^{-5}$	$1.54 \cdot 10^{-6}$
40	yes	yes	yes	10	$1.38 \cdot 10^{-6}$	$1.80 \cdot 10^{-6}$	$9.81 \cdot 10^{-6}$	$1.05 \cdot 10^{-6}$
50	yes	yes	yes	10	$7.81 \cdot 10^{-7}$	$1.57 \cdot 10^{-6}$	$8.61 \cdot 10^{-6}$	$1.10 \cdot 10^{-6}$
30	yes	yes	yes	14	$2.82 \cdot 10^{-6}$	$2.30 \cdot 10^{-9}$	$2.14 \cdot 10^{-6}$	$9.88 \cdot 10^{-10}$
50	yes	yes	yes	14	$4.66 \cdot 10^{-7}$	$4.07 \cdot 10^{-11}$	$6.02 \cdot 10^{-7}$	$3.58 \cdot 10^{-9}$
60	yes	yes	yes	14	$1.26 \cdot 10^{-7}$	$2.82 \cdot 10^{-12}$	$4.17 \cdot 10^{-7}$	$1.40 \cdot 10^{-12}$
50	no	no	yes	14	$5.05 \cdot 10^{-13}$	$1.93 \cdot 10^{-12}$	$2.89 \cdot 10^{-12}$	$3.04 \cdot 10^{-13}$

Table 8: 3-D example given by equations (6.10) and (6.11): 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 and 128 points

is:

$$u(x, y, z) = 10.0\, x\, e^{-250.0\,(x^2+y^2+z^2)}. \tag{6.11}$$

Table 8 describes the results by using equations (6.10) and (6.11).

6.5. Experiments with Line Singularities

Following our numerical experiments with isolated singularities we consider here (following the advice of a referee) the case of line singularities. What we would like to investigate is the behaviour of the masking procedure so as to find out if there is any difference between the present case and the former one.

The theoretical considerations do not imply a special behaviour depending on the dimensionality of the singular regions, however such a behaviour can not be ruled out in advance. Furthermore it is interesting to find out if the preconditioners remain efficient in the line singularity case.

We choose a "singularity" which is simple to implement, based on a Gaussian function having a fast decay in the direction normal to a chosen curve and a possible slow variation along the curve. For simplicity we chose curves belonging to a family of concentric circles with the center at the origin of coordinates.

More specifically the form of the functions is:

$$Z = F(x,y)\exp\left(-A\cdot(\sqrt{x^2+y^2}-K)^2\right). \tag{6.12}$$

Here K is the radius, A takes the value of 50 for $K = 0.1,\ 0.2,\ 0.3,\ 0.4,\ 0.5$, then we repeat the runs $K = 0.2,\ 0.5$, with $A = 250$. The function $F(x,y)$ is chosen to be equal to x which molifies a singular behaviour at the origin

$$xe^{-50\left((\sqrt{x^2+y^2}-K)^2\right)}, \tag{6.13}$$

$$xe^{-250\left((\sqrt{x^2+y^2}-K)^2\right)}. \tag{6.14}$$

For the choice $A = 50$, we observe that the most accurate results are obtained with the intermediate values of the radiiae, K =0.2 and 0.3. The full accuracy is not obtained with $K = 0.1$ and it deteriorates further for $K = 0.4$ and especially for the largest radius 0.5. There are two reasons for the limited accuracy, the first is related to the above mentioned (weak) singularity at the origin (the center of the circles). The second is due to the mismatch between the periodicity condition imposed by the nature of the chosen wavelets which implies periodicity of the computational box. Our forcing however is not periodic and consequently the solution itself is not periodic but should have a discontinuity in the first derivative. The strength of the discontinuity however depend on the magnitude of the derivative at the boundary which in turn depends on the distance of the center of the gaussian from the boundary as well as on the rate of decay.

Indeed for $A = 250$, we recover a very high accuracy even for the larger values of K. The higher regularity allows us to observe the effect of the higher number of vanishing moments (Daubechies D16) as compared to the D8 wavelets. For the case of $A = 50$ the low regularity of the solution at the boundary and origin contaminates the solution and affects negatively the higher accuracy of the wider wavelets.

No. of iterat-ions	Mask on s?	Mask on d?	Precond-itioner?	64 points L_2 error D 8	64 points L_2 error D 16	128 points L_2 error D 8	128 points L_2 error D 16
				$K = 0.1$			
20	no	no	yes	$1.85 \cdot 10^{-7}$	$1.84 \cdot 10^{-9}$	$2.25 \cdot 10^{-7}$	$1.55 \cdot 10^{-9}$
20	yes	yes	yes	$2.60 \cdot 10^{-7}$	$2.97 \cdot 10^{-9}$	$3.64 \cdot 10^{-7}$	$2.22 \cdot 10^{-8}$
30	no	no	yes	$2.08 \cdot 10^{-11}$	$3.25 \cdot 10^{-12}$	$3.41 \cdot 10^{-11}$	$2.68 \cdot 10^{-11}$
30	yes	yes	yes	$4.66 \cdot 10^{-8}$	$1.73 \cdot 10^{-10}$	$5.65 \cdot 10^{-8}$	$5.24 \cdot 10^{-9}$
40	no	no	yes	$3.77 \cdot 10^{-13}$	$3.65 \cdot 10^{-12}$	$6.88 \cdot 10^{-12}$	$5.65 \cdot 10^{-11}$
40	yes	yes	yes	$2.09 \cdot 10^{-10}$	$4.24 \cdot 10^{-11}$	$2.91 \cdot 10^{-8}$	$3.02 \cdot 10^{-9}$
40	no	no	no			$2.74 \cdot 10^{-1}$	
				$K = 0.2$			
20	no	no	yes	$1.63 \cdot 10^{-7}$	$1.26 \cdot 10^{-9}$	$1.95 \cdot 10^{-7}$	$9.45 \cdot 10^{-10}$
20	yes	yes	yes	$1.62 \cdot 10^{-7}$	$1.26 \cdot 10^{-9}$	$1.85 \cdot 10^{-7}$	$1.33 \cdot 10^{-9}$
30	no	no	yes	$1.65 \cdot 10^{-11}$	$4.71 \cdot 10^{-13}$	$2.87 \cdot 10^{-11}$	$6.83 \cdot 10^{-12}$
30	yes	yes	yes	$5.21 \cdot 10^{-9}$	$4.78 \cdot 10^{-13}$	$8.78 \cdot 10^{-9}$	$4.20 \cdot 10^{-10}$
40	no	no	yes	$1.35 \cdot 10^{-12}$	$5.10 \cdot 10^{-13}$	$3.20 \cdot 10^{-12}$	$6.80 \cdot 10^{-10}$
40	yes	yes	yes	$2.01 \cdot 10^{-9}$	$4.73 \cdot 10^{-13}$	$3.07 \cdot 10^{-9}$	$2.35 \cdot 10^{-10}$
				$K = 0.3$			
20	no	no	yes	$1.43 \cdot 10^{-7}$	$1.2 \cdot 10^{-9}$	$1.7 \cdot 10^{-7}$	$3.98 \cdot 10^{-10}$
20	yes	yes	yes	$1.42 \cdot 10^{-7}$	$1.2 \cdot 10^{-9}$	$1.68 \cdot 10^{-7}$	$9.48 \cdot 10^{-10}$
30	no	no	yes	$1.58 \cdot 10^{-11}$	$6.18 \cdot 10^{-13}$	$2.65 \cdot 10^{-11}$	$3.22 \cdot 10^{-12}$
30	yes	yes	yes	$1.58 \cdot 10^{-11}$	$6.51 \cdot 10^{-13}$	$1.34 \cdot 10^{-9}$	$2.27 \cdot 10^{-11}$
40	no	no	yes	$1.75 \cdot 10^{-12}$	$5.75 \cdot 10^{-13}$	$6.89 \cdot 10^{-13}$	$3.73 \cdot 10^{-12}$
40	yes	yes	yes	$1.68 \cdot 10^{-12}$	$8.75 \cdot 10^{-13}$	$2.99 \cdot 10^{-10}$	$1.09 \cdot 10^{-11}$
				$K = 0.4$			
20	no	no	yes	$1.47 \cdot 10^{-7}$	$1.39 \cdot 10^{-9}$	$1.56 \cdot 10^{-7}$	$1.01 \cdot 10^{-9}$
20	yes	yes	yes	$1.47 \cdot 10^{-7}$	$1.39 \cdot 10^{-9}$	$1.56 \cdot 10^{-7}$	$1.01 \cdot 10^{-9}$
30	no	no	yes	$3.89 \cdot 10^{-10}$	$3.89 \cdot 10^{-10}$	$1.95 \cdot 10^{-10}$	$1.92 \cdot 10^{-10}$
30	yes	yes	yes	$3.89 \cdot 10^{-10}$	$3.89 \cdot 10^{-10}$	$2.87 \cdot 10^{-10}$	$1.92 \cdot 10^{-10}$
40	no	no	yes	$3.89 \cdot 10^{-10}$	$3.89 \cdot 10^{-10}$	$1.92 \cdot 10^{-10}$	$1.92 \cdot 10^{-10}$
40	yes	yes	yes	$3.89 \cdot 10^{-10}$	$3.89 \cdot 10^{-10}$	$1.92 \cdot 10^{-10}$	$1.92 \cdot 10^{-10}$
				$K = 0.5$			
20	no	no	yes	$1.46 \cdot 10^{-7}$	$7.55 \cdot 10^{-8}$	$1.39 \cdot 10^{-7}$	$3.77 \cdot 10^{-8}$
20	yes	yes	yes	$1.46 \cdot 10^{-7}$	$7.55 \cdot 10^{-8}$	$1.39 \cdot 10^{-7}$	$3.77 \cdot 10^{-8}$
30	no	no	yes	$7.55 \cdot 10^{-8}$	$7.55 \cdot 10^{-8}$	$3.77 \cdot 10^{-8}$	$3.77 \cdot 10^{-8}$
30	yes	yes	yes	$7.55 \cdot 10^{-8}$	$7.55 \cdot 10^{-8}$	$3.77 \cdot 10^{-8}$	$3.77 \cdot 10^{-8}$
40	no	no	yes	$7.55 \cdot 10^{-8}$	$7.55 \cdot 10^{-8}$	$3.77 \cdot 10^{-8}$	$3.77 \cdot 10^{-8}$
40	yes	yes	yes	$7.55 \cdot 10^{-8}$	$7.55 \cdot 10^{-8}$	$3.77 \cdot 10^{-8}$	$3.77 \cdot 10^{-8}$

Table 9: 2-D example given by equation (8.1): 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 and 128 points in each direction. The mask accuracy (threshold) is 10^{-14}.

7. Conclusion and Further Work

The algorithm that was described in this paper uses 1-D computations that enables to solve certain problems in higher dimensions. The data structures

No. of iterations	Mask on s?	Mask on d?	Preconditioner?	64 points L_2 error D 8	64 points L_2 error D 16
			$K = 0.2$		
20	no	no	yes	$2.13 \cdot 10^{-7}$	$2.95 \cdot 10^{-9}$
20	yes	yes	yes	$6.46 \cdot 10^{-6}$	$3.97 \cdot 10^{-7}$
30	no	no	yes	$3.15 \cdot 10^{-11}$	$1.66 \cdot 10^{-12}$
30	yes	yes	yes	$2.93 \cdot 10^{-8}$	$1.86 \cdot 10^{-8}$
40	no	no	yes	$4.54 \cdot 10^{-12}$	$2.55 \cdot 10^{-12}$
40	yes	yes	yes	$1.78 \cdot 10^{-8}$	$1.11 \cdot 10^{-8}$
			$K = 0.5$		
20	no	no	yes	$2.11 \cdot 10^{-7}$	$2.85 \cdot 10^{-9}$
20	yes	yes	yes	$1.26 \cdot 10^{-6}$	$2.85 \cdot 10^{-9}$
30	no	no	yes	$3.04 \cdot 10^{-11}$	$2.83 \cdot 10^{-13}$
30	yes	yes	yes	$3.26 \cdot 10^{-8}$	$3.62 \cdot 10^{-13}$
40	no	no	yes	$8.92 \cdot 10^{-13}$	$4.21 \cdot 10^{-13}$
40	yes	yes	yes	$1.88 \cdot 10^{-9}$	$9.73 \cdot 10^{-13}$

Table 10: 2-D example given by equation (8.2): 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 in each direction. The mask accuracy (threshold) is 10^{-14}.

that were used for the sparse implementation in Section 6 are described in http://www.math.tau.ac.il/∼amir

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 at solving problems connected with 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 2.3 demonstrates that the algorithm described in this paper can work also for solving PDEs with non constant coefficients ([3]). The same algorithm 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 [30]. Once the singularities are tracked then we can

put the masks on these moving singularities. And the solver will operate between the mask boundaries as the proposed algorithm in this paper. We will use the methodology of Coifman and Meyer [30] that later was extended to analysis of non-linear PDEs by Bony [9] to track the movements of high frequency coefficients. In other words, this enable us to predict where the singularities will move next. Then we know where to place the masks at each iteration. Both topics, non-constant coefficients and non-linear PDEs, are currently being explored. This approach was also applied to strictly elliptic linear homogeneous PDE's [3] with non-constant coefficients of the form $\Delta U - b(\vec{x})U = f(\vec{x})$. There asparse multiplication algorithm with an existing diagonally preconditioned conjugate gradient (CG) method was adaptively combined.

Acknowledgments

The research of A. Averbuch, R. Coifman and M. Israeli was supported by U.S-Israel Binational Science Foundation grant # 92-00269/1.

M. Israeli was partially supported by the Technion VPR Fund.

The research of G. Beylkin was partially supported by DARPA 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.

References

[1] A. Averbuch, G. Beylkin, R. Coifman, M. Israeli, Multiresolution solution of elliptic and parabolic PDEs, In: *The Samuel Neaman Workshop on Signal and Image Representation in Combined Space* (Ed-s: Y. Zeevi, R. Coifman), Technion, Haifa, Israel, May 8-11, 1994, Academic Press (1998), 341-360.

[2] A. Averbuch, G. Beylkin, R. Coifman, P. Fischer, M. Israeli, A wavelets based constrained Preconditioned Conjugate Gradient for elliptic problems, In: *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, L. Beliak, M. Israeli, Adaptive calculation of variable coffi-cients elliptic differential equations via wavelets, *Report*, Technion (2003).

[4] A. Barinka, T. Barsch, P. Charton, A. Cohen, S. Dahlke, W. Dahmen, K. Urban, Adaptive wavelet schemes for elliptic problems - implementation and numerical experiments, *Reprint*.

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

[6] G. Beylkin, On the representation of operators in bases of compactly supported wavelet bases, *SIAM J. Num. Anal.*, **6** (1992), 1716-1740.

[7] G. Beylkin, On wavelet-based algorithms for solving differential equations, In: *Wavelets: Mathematics and Applications* (Ed-s: John J. Benedetto, Michael W. Frazier), CRC Press (1994), 449-466.

[8] G. Beylkin, R. Coifman, V. Rokhlin, Fast wavelet transforms and numerical algorithms I, *Comm. on Pure and Applied Mathematics*, **XLIV** (1991), 141-183.

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

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

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

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

[13] C.K. Chui, *An Introduction to Wavelets*, Academic Press (1992).

[14] A. Cohen, W. Dahmen, R. DeVore, Adaptive wavelet methods for elliptic operator equations - convergence rates, *Math. Comput.*, To Appear.

[15] W. Dahmen, A. Kunoth, Multilevel preconditioning, *Numer. Math.*, **63** (1992), 315-344.

[16] I. Daubechies, *Ten Lectures on Wavelets*, SIAM (1992).

[17] I. Daubechies, The wavelet tranform, time-frequency localization and signal analysis, *IEEE Trans. on Information Theory*, **36**, No. 5 (1990), 961-1005.

[18] I. Daubechies, Orthonormal bases of compactly supported wavelets, *Comm. on Pure and Applied Mathematics*, **XLI** (1988), 909-996.

[19] M. Dorobantu, Wavelet based algorithms for one-dimensional parabolic equations, *Technical Report*, No. 9214, NADA, KTH, Stockholm (1992).

[20] J.J. Duistermaat, L. Hörmander, Fourier integral operators. II, *Acta Math.*, **128**, No-s: 3-4 (1972), 183-269.

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

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

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

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

[25] S.J. Gortler, P. Schroder, M.F. Cohen, P. Hanrahan, Wavelet radiosity, In: *Computer Graphics Proceedings*, Annual Conf. (1993), 221-230.

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

[27] Lars Hörmander, *Fourier integral operators, I*, *Acta Math.*, **127**, No. 1-2 (1971), 79-183.

[28] S.G. Mallat, Multiresolution approximations and wavelet orthonormal bases for $L^2(\mathbb{R})$, *Trans. of AMS* (1989).

[29] Y. Meyer, *Wavelets and Operators I*, Cambridge University Press (1989).

[30] Y. Meyer, R. Coifman, *Wavelets: Calderon-Zygmund and Multilinear Operators*, Cambridge University Press (1997).

[31] F. Meyer, A. Averbuch, J-O Strömberg, Fast adaptive wavelet packet image compression, *IEEE Trans. on Image Processing*, **9**, No. 5 (2000), 792-800.

[32] L. Nirenberg, Lectures on linear partial differential equations, *Uspehi Mat. Nauk*, **30**, No. 4 (1975), 147-204.

[33] T. Tao, On the almost everywhere convergence of wavelet summation methods, *Preprint* (1995).

[34] Trèves, François, *Introduction to Pseudodifferential and Fourier Integral Operators*, Volumes 1-2, Plenum Press, New York (1980).

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

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

[37] X. Zhang, Multilevel Schwarz methods, *Numer. Math.*, **63** (1992), 521-539.

Appendix I: Theoretical Background

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.

$$\psi_{j,j',k,k'}(x,y) = \psi_{j,k}(x)\psi_{j',k'}(y)\,, \tag{8.1}$$

where $j, j', k, k' \in \mathbb{Z}$. This choice of basis leads to the standard form (S-form) of an operator [6, 8]. The projection of the operator T into the multiresolution analysis is represented in the S-form by the set of operators

$$T = \left\{A_j, \{B_j^{j'}\}_{j'\geq j+1}, \{\Gamma_j^{j'}\}_{j'\geq j+1}\right\}_{j\in\mathbb{Z}}, \tag{8.2}$$

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

$$\left.\begin{array}{rcl} A_j & = & Q_j T Q_j \quad : \quad \mathbf{W}_j \to \mathbf{W}_j \\ B_j^{j'} & = & Q_j T Q_{j'} \quad : \quad \mathbf{W}_{j'} \to \mathbf{W}_j \\ \Gamma_j^{j'} & = & Q_{j'} T Q_j \quad : \quad \mathbf{W}_j \to \mathbf{W}_{j'} \end{array}\right\} \tag{8.3}$$

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

If n is the finite number of scales, then (8.2) is restricted to the set of operators

$$T_0 = \left\{A_j, \{B_j^{j'}\}_{j'=j+1}^{j'=n}, \{\Gamma_j^{j'}\}_{j'=j+1}^{j'=n}, B_j^{n+1}, \Gamma_j^{n+1}, T_n\right\}_{j=1,\ldots,n} , \tag{8.4}$$

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 $\mathbf{V}_n$,

$$T_n = P_n T P_n : \mathbf{V}_n \to \mathbf{V}_n \ . \tag{8.5}$$

The subspaces $\mathbf{V}_j$ and $\mathbf{W}_j$ appearing in (8.3) and (8.5) can be periodized.

The operators $A_j, B_j^{j'}, \Gamma_j^{j'}$, and T_j appearing in (8.2) and (8.4) are represented by matrices $\alpha^j, \beta^{j,j'}, \gamma^{j,j'}$ and s^j with entries defined by

$$\left.\begin{array}{rcl} \alpha_{k,k'}^j & = & \int\int \psi_{j,k}(x) K(x,y) \psi_{j,k'}(y) dx dy \\ \beta_{k,k'}^{j,j'} & = & \int\int \psi_{j,k}(x) K(x,y) \varphi_{j',k'}(y) dx dy \\ \gamma_{k,k'}^{j,j'} & = & \int\int \varphi_{j,k}(x) K(x,y) \psi_{j',k'}(y) dx dy \\ s_{j,k'}^j & = & \int\int \varphi_{j,k}(x) K(x,y) \varphi_{j,k'}(y) dx dy \end{array}\right\} , \tag{8.6}$$

where $K(x,y)$ is the kernel of the operator T. The operators in (8.4) are organized as blocks of a matrix.

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 $\mathbf{L}^2(\mathbb{R}^2)$

$$\begin{array}{l} \psi_{j,k}(x)\psi_{j,k'}(y) , \\ \psi_{j,k}(x)\varphi_{j,k'}(y) , \\ \varphi_{j,k}(x)\psi_{j,k'}(y) , \end{array} \tag{8.7}$$

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

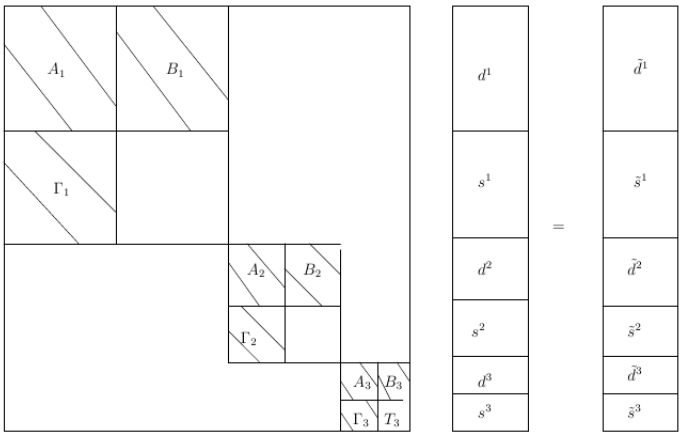

Figure 4: Application of non-standard form to a vector

in the "telescopic" series

$$T = \sum_{j \in \mathbb{Z}} (Q_j T Q_j + Q_j T P_j + P_j T Q_j) \ , \tag{8.8}$$

where P_j and Q_j are projectors on subspaces $\mathbf{V}_j$ and $\mathbf{W}_j$, respectively. We observe that in (8.8) 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

$$\left.\begin{array}{rcccl} A_j & = & Q_j T Q_j & : & \mathbf{W}_j \to \mathbf{W}_j \\ B_j & = & Q_j T P_j & : & \mathbf{V}_j \to \mathbf{W}_j \\ \Gamma_j & = & P_j T Q_j & : & \mathbf{W}_j \to \mathbf{V}_j \end{array}\right\} , \tag{8.9}$$

see e.g. [6].

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 equation (8.6) in $\mathbf{L}^2(\mathbb{R}^2)$. The operators are organized as blocks of a matrix.

The price of uncoupling the scale interactions in (8.8) is the need for an additional projection into the wavelet basis of the product of the NS-form and a function.

Referring to Figure 4 we see that the NS-form is applied to both averages and differences of the wavelet expansion of a function.

Appendix II: Description of Sparse Data Structures

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*.