Adaptive Wavelet Methods for Hyperbolic PDEs

Mats Holmström* Johan Waldén*†

Abstract

We analyze how to solve hyperbolic PDEs with compactly supported orthonormal wavelets adaptively. We use thresholded wavelet expansions of signals and operators. A tree structure is used to represent the signal, and a multi-dimensional analogue of the fast wavelet transform is used to expand the operators. We solve the advection equation and Burgers' equation on a periodic domain.

Key words. wavelets, adaptive PDE methods, hyperbolic equations, numerical algorithms

AMS subject classifications. 65D25, 65M50, 65M60

1 Introduction

When using wavelets for solving partial differential equations (PDEs), there are two main approaches. With the first approach, the PDE is solved with a standard method, and wavelets are used to analyze the "local frequency contents" of the signal (see, e.g. B. L. Bihari and A. Harten [8]). With the second, "pure" approach, wavelets are used to actually solve the PDE, i.e., the operators of the equation are approximated in wavelet bases. This can for example be done with a collocation method as proposed by S. Bertoluzza and G. Naldi [4], or with a Galerkin method see; e.g. P. Fisher and M. Defranceschi [11]. For elliptic PDEs there are diagonal preconditioners that give slowly increasing or bounded condition numbers for the system of linear equations that arise when the PDE is discretized. This system can then be solved by an iterative method.

The literature on pure wavelet methods for hyperbolic PDEs is sparse. One idea for how wavelet methods might be efficient for hyperbolic PDEs is that a function that is smooth, except for some isolated shocks, can be approximated with few coefficients

*Uppsala University, Department of Scientific Computing, P.O. Box 120, S–751 04 Uppsala, Sweden.

†Research supported by the Swedish Natural Science Research Council, NFR, under grant F–FU 04370–304.

in the wavelet domain. This could be used to construct an adaptive method with a non-uniform grid.

Here, two different approaches are applicable. One can choose an approach similar to what is done when using pseudo-spectral methods, i.e., to use both the physical representation of the signal, and the wavelet representation, e.g., doing multiplication in the physical space, and differentiation in the wavelet space; see for example J. Keiser [14]. With this approach one transforms back and forth between the physical domain and the wavelet domain in each time step, which introduces some problems as a representation, sparse in one of the domains, may be non-sparse in the other.

We choose the second approach, i.e., to do all the computations in the wavelet domain, and only transform back and forth once between the domains. This has been done by E. Bacry et al., in [3] with focus on how to do time-stepping. We use a Galerkin method for discretization. Analogously to the signal, the operators can be approximated by sparse operators in the wavelet domain, which will give an overall fast way of solving the problem. We use an explicit time-marching scheme, and solve the advection equation and Burgers' equation, on a periodic domain.

In Section 2 we introduce some notation, and review wavelet theory. In Section 3 we describe how to obtain quadrature formulae for inner products. We introduce sparse signals and operators, and describe how to periodize these. Furthermore, we describe the data structures involved in the PDE solver. Finally, in Section 4 we solve the advection equation with different wavelets, and Burgers' equation. We also discuss generalizations to PDEs in more than one dimension.

2 Preliminaries

We use the Hilbert spaces $L^2(\mathbb{R})$ and $L^2(0,1)$ with the inner products

$$\langle f, g\rangle_{\mathbb{R}} = \int_{-\infty}^{\infty} f\bar{g}dx, \tag{1}$$

$$\langle f, g\rangle_{(0,1)} = \int_{0}^{1} f\bar{g}dx. \tag{2}$$

The following notation will be used: Thin letters correspond to scalars (a, A, α). Boldface upper case letters correspond to matrices $(\mathbf{A}, \mathbf{B})$ and boldface lower case letters to vectors $(\mathbf{a}, \mathbf{b})$. The scalar on the ith row and in the jth column of a matrix, $\mathbf{A}$, will be denoted $(\mathbf{A})_{ij}$, and the sub-matrix consisting of elements from row r_1 to r_2 and column c_1 to c_2, we denote $(\mathbf{A})_{r_1:r_2,c_1:c_2}$. The same notation will be used for vectors. Scalar operations on vectors are to be thought of as element-wise, e.g., $2^{\mathbf{j}} = (2^{(\mathbf{j})_1}, 2^{(\mathbf{j})_2}, \ldots)$, $\max(\mathbf{j}, \mathbf{k}) = (\max((\mathbf{j})_1, (\mathbf{k})_1), \max((\mathbf{j})_2, (\mathbf{k})_2), \ldots)$. We will also use element-wise multiplication for vectors, $\mathbf{a} .\times \mathbf{b} = ((\mathbf{a})_1(\mathbf{b})_1, (\mathbf{a})_2(\mathbf{b})_2, \ldots)$. Special vectors are $\mathbf{1} = (1, 1, 1, \ldots)$, $\mathbf{0} = (0, 0, 0, \ldots)$ and δ_i, $(\delta_i)_i = 1, (\delta_i)_{j\neq i} = 0$. The transpose of a matrix (vector) is written $\mathbf{A}^T$. Arbitrary multi-dimensional indexed sets will be denoted by upper case Greek letters, (Γ, Ω). We will skip ranges in sums and integrals when obvious. Throughout the paper we will assume that we are working with real-valued wavelets.

A useful way to approach wavelet analysis is through a so called multiresolution analysis (MRA) which was introduced by S. Mallat, [15]. An MRA consists of a sequence of closed subspaces, V_j, such that

$$\cdots \subset V_{-2} \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \subset \cdots, \tag{3}$$

$$\overline{\bigcup_{j\in\mathbb{Z}} V_j} = L^2(\mathbb{R}), \tag{4}$$

$$\bigcap_{j\in\mathbb{Z}} V_j = \{0\}, \tag{5}$$

$$f \in V_j \Leftrightarrow f(2^{-j}\cdot) \in V_0, \tag{6}$$

$$f \in V_0 \Leftrightarrow f(\cdot - k) \in V_0, \tag{7}$$

$$\exists \varphi : \{\varphi(\cdot - k)\}_k \text{ is an orthonormal basis of } V_0. \tag{8}$$

We denote the orthogonal complement of V_j in V_{j+1} by W_j,

$$V_{j+1} = V_j \bigoplus W_j. \tag{9}$$

With this construction we get

$$\varphi(x) = \sqrt{2}\sum_k h_k \varphi(2x - k), \tag{10}$$

where (8) ensures that

$$\sum_k |h_k|^2 = 1. \tag{11}$$

By defining

$$\psi(x) = \sqrt{2}\sum_k g_k \varphi(2x - k), \tag{12}$$

where $g_k = (-1)^k h_{1-k}$, we get an orthonormal basis of $L^2(\mathbb{R})$, i.e., if we put

$$\psi_{jk}(x) = 2^{j/2}\psi(2^j x - k), \tag{13}$$

$$\varphi_{jk}(x) = 2^{j/2}\varphi(2^j x - k), \tag{14}$$

we have

$$\langle \psi_{jk}, \psi_{lm} \rangle = \delta_{jl}\delta_{km}, \tag{15}$$

$$\overline{\text{span}(\{\psi_{jk}\}_{jk})} = L^2(\mathbb{R}). \tag{16}$$

To get good approximation properties, we furthermore require the wavelets to have a number of vanishing moments, i.e.,

$$\int x^n \psi(x) dx = 0, \qquad n = 0, \ldots, N-1. \tag{17}$$

The scaling functions which give wavelets with the shortest support compared to their number of vanishing moments are the Daubechies scaling functions, D_{2N}, with N vanishing moments and a support of $[0, 2N-1]$.

We denote the orthogonal projection onto V_n by E_n, and by Q_n the projection onto W_n. It is a well-known fact, carrying back to [12], that for compactly supported wavelets (17) implies that

$$\|E_j f - f\|_2 = \mathcal{O}(2^{-jN}). \tag{18}$$

Eq. (17) also is a necessary condition if we require some regularity of ψ; $\psi \in C^N \Rightarrow \int x^n \psi(x) dx = 0,\ n = 0, \ldots, N$.

One way to periodize wavelets is to use the following definitions

$$\varphi_{jk}^{\mathrm{per}}(x) = \sum_l \varphi_{jk}(x + l), \tag{19}$$

$$\psi_{jk}^{\mathrm{per}}(x) = \sum_l \psi_{jk}(x + l), \tag{20}$$

generating the spaces

$$V_j^{\mathrm{per}} = \mathrm{span}(\{\varphi_{jk}^{\mathrm{per}}(x)\}_{k=0,\ldots,2^j-1}), \tag{21}$$

$$W_j^{\mathrm{per}} = \mathrm{span}(\{\psi_{jk}^{\mathrm{per}}(x)\}_{k=0,\ldots,2^j-1}). \tag{22}$$

This construction gives an MRA of $L^2(0,1)$ with the coarsest level V_0 as $V_j = \mathrm{span}(\{1\})$ for $j \leq 0$. In analogy with the non-periodic case we define the projection operators E_n^{per}, Q_n^{per}. Throughout the paper, we will use the non-periodic wavelets for analysis, whereas for numerical results, we use the periodized wavelets. The modifications needed are treated in Section 3.6.

The fast wavelet transform (FWT) is the corner stone for wavelets being so useful in numerical analysis. If we, for a function $f \in L^2(\mathbb{R})$, make the following definitions

$$s_{j,k} = \langle f, \varphi_{j,k} \rangle, \tag{23}$$

$$d_{j,k} = \langle f, \psi_{j,k} \rangle, \tag{24}$$

the projection operators can be written

$$E_j f = \sum_k s_{j,k} \varphi_{j,k} = (\mathbf{s}^j)^T (\mathbf{\Phi}^j), \tag{25}$$

$$Q_j f = \sum_k d_{j,k} \psi_{j,k} = (\mathbf{d}^j)^T (\mathbf{\Psi}^j). \tag{26}$$

The FWT provides an $\mathcal{O}(N)$ algorithm (where $N = 2^n$, is the number of elements in $\mathbf{s}^n$) to go between $s_{j,k}$s and $d_{j,k}$s. From (10), (12) we get

$$s_{j-1,k} = \sum_m h_{m-2k} s_{j,m}, \tag{27}$$

$$d_{j-1,k} = \sum_m g_{m-2k} s_{j,m}, \tag{28}$$

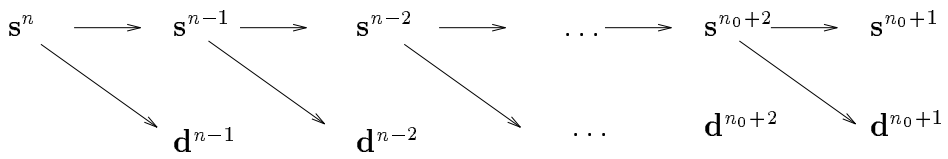

Figure 1: *The pyramid fast wavelet transform.*

and repeating this will give the fast "pyramid" scheme for computing the wavelet representation of E_n,

$$E_n = Q_{n-1} + Q_{n-2} + Q_{n-3} + \ldots + Q_{n_0+1} + E_{n_0+1}. \tag{29}$$

The idea is shown if Figure 1.

With the notation

$$\begin{aligned} t_{j,k} &= d_{j,k}, \qquad j = n_0+1, \ldots, n-1, \\ t_{n_0,k} &= s_{n_0+1,k}, \end{aligned} \tag{30}$$

$$\begin{aligned} \lambda_{j,k} &= \psi_{j,k}, \qquad j = n_0+1, \ldots, n-1, \\ \lambda_{n_0,k} &= \varphi_{n_0+1,k}, \end{aligned} \tag{31}$$

the decomposition, (29), can simply be written

$$E_n f = \sum_{j=n_0}^{n-1} \sum_{k \in \mathbb{Z}} t_{j,k} \lambda_{j,k}. \tag{32}$$

In the same manner we can define $\lambda_{j,k}^{\mathrm{per}}$ and $t_{j,k}^{\mathrm{per}}$.

3 Theory

3.1 Quadrature

To project a function, f, onto a scaling function space, V_j, we need a way to compute the scaling coefficients, $s_{j,k}$. The wavelet coefficients can then be computed with the FWT. Since functions often are given as equi-spaced point values, we would like a way to compute scaling coefficients from point values, and also to compute the reverse, point values from scaling coefficients. This section presents quadrature formulae to accomplish this.

As before, we express the projection of a function f, onto the space V_j as,

$$E_j f(x) = \sum_k s_{j,k} \varphi_{j,k}(x) = \sum_k \langle f, \varphi_{j,k} \rangle \varphi_{j,k}(x). \tag{33}$$

Given equi-spaced samples of f, $f(ih), i \in \mathbb{Z}, h = 2^{-j}$, on the real line, we approximate the scaling coefficients by finding weights, w_k, such that

$$s_{j,k} = \sum_k w_k f(2^{-j}(j+k)) + \mathcal{O}(h^p), \tag{34}$$

where p is the order of the approximation. Since $\varphi_{j,k}$ is a scaled and translated version of φ, we can, without loss of generality, examine the case

$$s_{0,0} = \int_{-\infty}^{\infty} f(x)\varphi(x)dx. \tag{35}$$

If we use the Trapezoidal rule to approximate this integral we get

$$s_{0,0} \approx \sum_k \varphi(k)f(k), \tag{36}$$

i.e., the weights $w_k = \varphi(k)$. In general the Trapezoidal rule is a second order approximation ($p = 2$ in (34)), but as shown by Sweldens and Piessens [16], for a wavelet with N vanishing moments, defined by (17), the Trapezoidal rule is an approximation of order N. We then have a quadrature formula that is of the same order as the function approximation itself, since by (18) the approximation error for a wavelet with N vanishing moments is of order N.

If we restrict our attention to compactly supported scaling functions, with $\text{supp}\varphi(x) = [0, 2m-1]$ and N vanishing moments, the Trapezoidal rule leads to the approximation

$$s_{j,k} = 2^{-j/2} \sum_{l=1}^{2m-2} \varphi(l)f(2^{-j}(l+k)) + \mathcal{O}(h^N), \tag{37}$$

as a solution to the original problem (34), where we have used the dilation equation (14). This is not optimal in the sense that we must use $2m-2$ function values to compute one scaling coefficient for a wavelet with N vanishing moments. In most cases $2m - 2 > N$ (e.g, for Daubechies wavelets $m = N$) and in [16] it is shown that there always are weights such that we can get an approximation of order N, using at most $N+1$ function values. Still we choose to use the Trapezoidal rule due to its simplicity.

We solve the inverse problem of recovering the sampled function values, $f(2^{-j}l)$, from the scaling coefficients $s_{j,k}$ by sampling the projection of f, i.e., we compute

$$f(2^{-j}l) \approx E_j f(2^{-j}l) = 2^{j/2} \sum_{k=l-2m}^{l-1} s_{j,k}\varphi(l-k). \tag{38}$$

The remaining problem is to find the values of the scaling functions at the integers. Using the scaling equation (10) for φ, we get the $2m-2$ by $2m-2$ system of linear equations

$$\varphi(k) = \sum_{l=l_1}^{l_2} h_l \varphi(2k-l), \qquad 1 \le k \le 2m-2, \tag{39}$$

where $l_1 = \max(0, 2k-2m+2)$ and $l_2 = \min(2m-1, 2k-1)$. Since this is a homogeneous system of equations, we exchange one of the above equations with the normalization equation

$$\sum_{k=1}^{2m-2} \varphi(k) = 1, \tag{40}$$

so that the system of equations has a unique solution.

Numerical experiments suggest that this system of equations, at least for Daubechies wavelets, can be solved in a few fixed point iterations

$$\begin{cases} \varphi^{n+1}(k) = \sum_{l=l_1}^{l_2} h_l \varphi^n(2k-l), & 1 \le k < 2m-2, \\ \varphi^{n+1}(2m-2) = 1 - \sum_{l=1}^{2m-1} \varphi^n(l), & n = 0, 1, 2, \ldots, \\ \varphi^0(l) = \dfrac{1}{2m-2}, & 1 \le l \le 2m-2. \end{cases} \tag{41}$$

Here $\varphi^n(l)$ is the approximation of $\varphi(l)$ after n fixed point iterations.

3.2 Sparse representation of functions in a wavelet basis

Functions which are smooth, except for in localized regions, can be efficiently represented in a wavelet basis up to any given accuracy ϵ. Such functions appear for example in acoustic problems, where we can have a low-frequency wave, with a localized high-frequency burst. Another example is in fluid dynamics, where the solution can be smooth except for in regions with shocks or discontinuities.

Assume that we project a function, $f(x)$, onto the space V_n, and perform the wavelet transform. We then have an expansion

$$E_n f(x) = \sum_k s_{n_0+1,k} \varphi_{n_0+1,k}(x) + \sum_{n_0+1 \le j < n,k} d_{jk} \psi_{jk}(x) = \sum_{n_0 \le j < n,k} t_{jk} \lambda_{jk}(x). \tag{42}$$

To get a sparse representation we remove all wavelet coefficients with absolute value less than some threshold value ϵ. Then we have the thresholded expansion

$$E_n^\epsilon f(x) = f^\epsilon(x) = \sum_{(j,k) \in I_n^\epsilon} t_{jk} \lambda_{jk}(x). \tag{43}$$

The set I_n^ϵ of retained coefficients is defined by

$$I_n^\epsilon = \{(j,k) : n_0 \le j < n, |t_{jk}| > \epsilon\}. \tag{44}$$

We will denote the number of retained coefficients by N_s. The total number of coefficients is N. To illustrate this thresholding we choose a 1-periodic function

$$f(x) = \sin(2\pi x) + g_\mathrm{p}(x), \tag{45}$$

that is smooth in most of the domain except near $x = 0.5$ where we have a "spike." Here $g_\mathrm{p}(x)$ is a periodized Gaussian, centered at $x = 0.5$, of height 1, defined by

$$g_\mathrm{p}(x) = \sum_{k \in \mathbb{Z}} g(x-k), \qquad g(x) = e^{-\beta(x-1/2)^2}, \tag{46}$$

and β is a parameter that controls the width of the Gaussian. A plot of $E_n^\epsilon f(x)$ is shown in Figure 2, where we have chosen the threshold value as large as $\epsilon = 0.1$ in order to get a visually perceptible error.

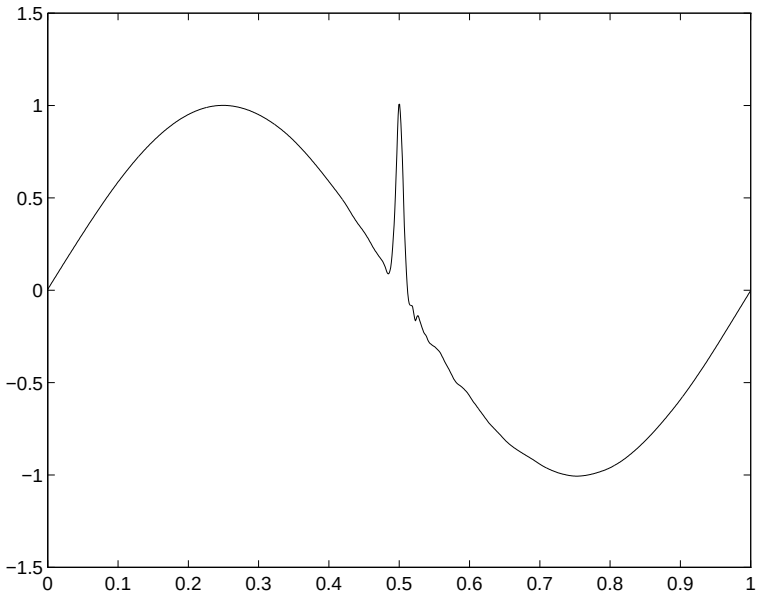

Figure 2: *A plot of the truncated wavelet expansion $E_{10}^{\epsilon} f(x)$, for $f(x)$ defined in (45), with $\beta = 20000$. The wavelet basis is that of Daubechies with six vanishing moments, D_{12}. The threshold is $\epsilon = 0.1$, which leads to 12 retained coefficients, out of the original 2^{11}, in I_{10}^{ϵ}.*

3.3 Sparse operators

There are several ways of making a wavelet approximation of an operator, $T : (L^2)^N \to L^2$, e.g., by a collocation method [4], or a Galerkin method [11, 14]. We will use a Galerkin method to define the approximating operators by

$$T_n : V_n^N \to V_n, \tag{47}$$

where

$$T_n : (f_1, \ldots, f_N) \longmapsto E_n T(f_1, \ldots, f_N), \qquad f_i \in V_n,\ i = 1, \ldots, N. \tag{48}$$

If T is an N-linear operator, i.e., if

$$\begin{aligned} &T(f_1, \ldots, a\alpha(x) + b\beta(x), \ldots, f_N) \\ &\qquad = aT(f_1, \ldots, \alpha(x), \ldots, f_N) + bT(f_1, \ldots, \beta(x), \ldots, f_N), \end{aligned} \tag{49}$$

then we define

$$\Phi_{\mathbf{j},\mathbf{k}} = \langle T(\varphi_{(\mathbf{j})_1,(\mathbf{k})_1}, \ldots, \varphi_{(\mathbf{j})_N,(\mathbf{k})_N}), \varphi_{(\mathbf{j})_{N+1},(\mathbf{k})_{N+1}} \rangle, \tag{50}$$

$$\Psi_{\mathbf{j},\mathbf{k}} = \langle T(\psi_{(\mathbf{j})_1,(\mathbf{k})_1}, \ldots, \psi_{(\mathbf{j})_N,(\mathbf{k})_N}), \psi_{(\mathbf{j})_{N+1},(\mathbf{k})_{N+1}} \rangle, \tag{51}$$

and, more generally

$$\Theta_{\mathbf{j},\mathbf{k},\mathbf{N}} = \langle T(\gamma^1_{(\mathbf{j})_1,(\mathbf{k})_1}, \ldots, \gamma^N_{(\mathbf{j})_N,(\mathbf{k})_N}), \gamma^{N+1}_{(\mathbf{j})_{N+1},(\mathbf{k})_{N+1}} \rangle, \tag{52}$$

where $\mathbf{j} \in \mathbb{Z}^{N+1}$, $\mathbf{k} \in \mathbb{Z}^{N+1}$, $\mathbf{N} \in \{0,1\}^{N+1}$, $\gamma^i_{j,k} = \varphi_{j,k}$ if $(\mathbf{N})_i = 1$ and $\gamma^i_{j,k} = \psi_{j,k}$ if $(\mathbf{N})_i = 0$.

We want a representation of the operator that takes advantage of the sparse representation we have of the functions. Therefore, we should not use the FWT as the function has a non-sparse representation in the space of scaling functions. Thus, operations that are simple in the physical domain (such as multiplication) will require some work in the sparse wavelet domain. This means that we should view the operator in accordance with the decomposition (29). We define the function, M,

$$\begin{aligned} M(j) &= 1, \qquad j = n_0, \\ M(j) &= 0, \qquad j = n_0+1, \ldots, n-1, \end{aligned} \tag{53}$$

and the vector $\mathbf{M}(\mathbf{j})$,

$$\mathbf{M}(\mathbf{j}) = (M((\mathbf{j})_1), M((\mathbf{j})_2), \ldots, M((\mathbf{j})_{N+1})). \tag{54}$$

This gives us the following formula for an N-linear operator using the expansion (32), $f_i = \sum_{j,k} t^i_{j,k} \lambda_{j,k}$,

$$\begin{aligned} T_n(f_1, \ldots, f_N) = \sum_{\mathbf{j} \in J,\, \mathbf{k} \in K} (\prod_{i=1}^{N} t^i_{(\mathbf{j})_i,(\mathbf{k})_i}) \Theta_{\mathbf{j}+\mathbf{M}(\mathbf{j}),\mathbf{k},\mathbf{M}(\mathbf{j})} \lambda_{(\mathbf{j})_{N+1},(\mathbf{k})_{N+1}}, \\ J = \{n_0, \ldots, n-1\}^{N+1},\ K = \mathbb{Z}^{N+1}. \end{aligned} \tag{55}$$

Equation (55) includes a lot of cross terms, which seems to make it numerically costly, even if we have sparse representations of our functions. To make it numerically efficient we will have to exclude some of the terms. One way of doing this, which we will not consider, is to combine the pyramid scheme with the $\Theta_{\mathbf{j},\mathbf{k},\mathbf{N}}$s to get the so called *non-standard form* of the operator, which will be banded [7]. Another way is to ignore the "small" coefficients in the operator, (which hopefully are many) and make an approximation of the (already approximated) operator. As we will be using explicit time-marching schemes, with operators which are typically differentiation and multiplication, the operator coefficients will vanish for wavelets that are far apart, i.e., with non-overlapping support. The decay between different scales will also be fast. We refer to [17] for results on this, where orthonormal wavelets with better decay between different scales are constructed. If one leaves the setting of orthonormal compactly supported wavelets, there are results that imply that spline wavelets are a good choice [6]. We will use orthonormal, compactly supported wavelets.

With this in mind we, in analogy with the truncation of the function, define a truncated operator, T^ϵ_j, which performs the same as (55), but only for Θs with absolute values larger than ϵ. In analogy with the function representation, we define the set I^ϵ_T by

$$I^\epsilon_T = \{(\mathbf{j},\mathbf{k}) : |\Theta_{\mathbf{j}+\mathbf{M}(\mathbf{j}),\mathbf{k},\mathbf{M}(\mathbf{j})}| > \epsilon,\, \mathbf{j} \in J,\, \mathbf{k} \in K\}. \tag{56}$$

The truncated operator, T^ϵ_n, is now defined as

$$T^\epsilon_n(f_1, \ldots, f_N) = \sum_{(\mathbf{j},\mathbf{k}) \in I^\epsilon_T} (\prod_{i=1}^{N} t^i_{(\mathbf{j})_i,(\mathbf{k})_i}) \Theta_{\mathbf{j}+\mathbf{M}(\mathbf{j}),\mathbf{k},\mathbf{M}(\mathbf{j})} \lambda_{(\mathbf{j})_{N+1},(\mathbf{k})_{N+1}}. \tag{57}$$

Finally, we would like to apply this operator to truncated functions, and with the notation

$$\begin{aligned} I_f^\epsilon = \{(\mathbf{j},\mathbf{k}) : ((\mathbf{j})_i,(\mathbf{k})_i) \in I_n^\epsilon(f_i),\ i=1,\dots,N, \\ (\mathbf{j})_{N+1} \in \{n_0,\dots,n-1\},\ (\mathbf{k})_{N+1} \in \mathbb{Z}\}, \end{aligned} \tag{58}$$

we get

$$T_n^\epsilon(f_1^\epsilon,\dots,f_N^\epsilon) = \sum_{(\mathbf{j},\mathbf{k})\in I_T^\epsilon \cap I_f^\epsilon} (\prod_{i=1}^N t^i_{(\mathbf{j})_i,(\mathbf{k})_i}) \Theta_{\mathbf{j}+\mathbf{M}(\mathbf{j}),\mathbf{k},\mathbf{M}(\mathbf{j})} \lambda_{(\mathbf{j})_{N+1},(\mathbf{k})_{N+1}}. \tag{59}$$

To get the Θ coefficients one can of course do numerical approximations of the integrals, but following [5] we use (10), (12) to get a recursive relationship between the coefficients. We shall study operators that, apart from being linear, satisfy dilation and translation invariance, i.e.,

$$\begin{aligned} (T(f_1(\alpha\cdot),\dots,f_N(\alpha\cdot)))(x) &= \alpha^\nu (T(f_1,\dots,f_N))(\alpha x), \\ (T(f_1(\cdot-\alpha),\dots,f_N(\cdot-\alpha)))(x) &= (T(f_1,\dots,f_N))(x-\alpha). \end{aligned} \tag{60}$$

We call such operators LDT (Linear Dilation Translation invariant) operators. We define the generalized N-dimensional autocorrelation and convolution by

$$(h \circ^j_{s,t} \Gamma)_{\mathbf{k}} = \sum_r h_r \Gamma_{\mathbf{k}+((t-1)(\mathbf{k})_s + jr)\delta_s}, \tag{61}$$

$$(h *^{\mathbf{j}}_{\mathbf{t}} \Gamma)_{\mathbf{k}} = \sum_r h_r \Gamma_{\mathbf{t}\cdot\times\mathbf{k}-r\mathbf{j}}, \tag{62}$$

where Γ is any N-dimensional indexed set. We define

$$\begin{aligned} F(s,t) &= 1, \qquad s \geq t, \\ F(s,t) &= 2, \qquad s < t, \\ \mathbf{F}(\mathbf{s},\mathbf{t}) &= (F((\mathbf{s})_1,(\mathbf{t})_1),\dots,F((\mathbf{s})_N,(\mathbf{t})_N)). \end{aligned} \tag{63}$$

We then have the recursive formulae

$$\left\{ \begin{aligned} (\Omega^{\mathbf{0},\mathbf{1}})_{((\mathbf{k})_1,\dots,(\mathbf{k})_N)} &= \Theta_{\mathbf{0},((\mathbf{k})_1,\dots,(\mathbf{k})_N,0),\mathbf{1}}, = \Phi_{\mathbf{0},((\mathbf{k})_1,\dots,(\mathbf{k})_N,0)}, \\ \Omega^{\mathbf{j}+\mathbf{1},\mathbf{N}} &= \nu\Omega^{\mathbf{j},\mathbf{N}}, \\ \Omega^{\mathbf{j}-\delta_s,\mathbf{N}} &= h \circ^{2^{\max((\mathbf{j})_{N+1}-(\mathbf{j})_s,0)}}_{s,F((\mathbf{j})_{N+1},(\mathbf{j})_s)} \Omega^{\mathbf{j},\mathbf{N}}, \qquad s=1,\dots,N, \\ \Omega^{\mathbf{j}-\delta_s,\mathbf{N}-\delta_s} &= g \circ^{2^{\max((\mathbf{j})_{N+1}-(\mathbf{j})_s,0)}}_{s,F((\mathbf{j})_{N+1},(\mathbf{j})_s)} \Omega^{\mathbf{j},\mathbf{N}}, \qquad s=1,\dots,N, \\ \Omega^{\mathbf{j}-\delta_{N+1},\mathbf{N}} &= h *^{2^{\max((\mathbf{j})_{1:N}-(\mathbf{j})_{N+1}\mathbf{1},\mathbf{0})}}_{\mathbf{F}((\mathbf{j})_{1:N},(\mathbf{j})_{N+1}\mathbf{1})} \Omega^{\mathbf{j},\mathbf{N}}, \\ \Omega^{\mathbf{j}-\delta_{N+1},\mathbf{N}-\delta_{N+1}} &= g *^{2^{\max((\mathbf{j})_{1:N}-(\mathbf{j})_{N+1}\mathbf{1},\mathbf{0})}}_{\mathbf{F}((\mathbf{j})_{1:N},(\mathbf{j})_{N+1}\mathbf{1})} \Omega^{\mathbf{j},\mathbf{N}}, \end{aligned} \right. \tag{64}$$

and we have

$$\Theta_{\mathbf{j},\mathbf{k},\mathbf{N}} = \Omega^{\mathbf{j},\mathbf{N}}_{2^{\max((\mathbf{j})_{N+1}\mathbf{1}-(\mathbf{j})_{1:N},\mathbf{0})} \times (\mathbf{k})_{1:N} - 2^{\max((\mathbf{j})_{1:N}-(\mathbf{j})_{N+1}\mathbf{1},\mathbf{0})}(\mathbf{k})_{N+1}\mathbf{1}}. \tag{65}$$

Here, each Ω is an N-dimensional indexed set, and $\mathbf{j} \in \mathbb{Z}^{N+1}$, $\mathbf{N} \in \{0,1\}^{N+1}$. This gives us a way to construct every Θ from the coefficients $\Theta_{\mathbf{0},\mathbf{k},\mathbf{0}}$. The transformations become an N-dimensional version of the pyramid algorithm. This is shown in Figure 3. As $\circ^j_{s,t}$, $*^{\mathbf{j}}_{\mathbf{t}}$, are commutative, it is irrelevant in which order these operations are applied and this reduces the number of arrows in the figure. If the operator satisfies some symmetry property one can further reduce the number of coefficients required.

Example 1 The differential operator, $\frac{d^r}{dx^r}$, is an LDT-operator with dilation factor, $\nu = 2^r$, represented in the following way for $j_1 \geq j_2$

$$\begin{aligned}
\Theta^{(j_1,j_2),(1,1)}_{(k_1,k_2)} &= \langle \varphi^{(r)}_{j_1,k_1}, \varphi_{j_2,k_2} \rangle = \nu^{(j_1-j_2)} (\Omega^{(j_2-j_1,0),(1,1)})_{2^{j_1-j_2}k_1-k_2}, \\
\Theta^{(j_1,j_2),(1,0)}_{(k_1,k_2)} &= \langle \varphi^{(r)}_{j_1,k_1}, \psi_{j_2,k_2} \rangle = \nu^{(j_1-j_2)} (\Omega^{(j_2-j_1,0),(1,0)})_{2^{j_1-j_2}k_1-k_2}, \\
\Theta^{(j_1,j_2),(0,1)}_{(k_1,k_2)} &= \langle \psi^{(r)}_{j_1,k_1}, \varphi_{j_2,k_2} \rangle = \nu^{(j_1-j_2)} (\Omega^{(j_2-j_1,0),(0,1)})_{2^{j_1-j_2}k_1-k_2}, \\
\Theta^{(j_1,j_2),(0,0)}_{(k_1,k_2)} &= \langle \psi^{(r)}_{j_1,k_1}, \psi_{j_2,k_2} \rangle = \nu^{(j_1-j_2)} (\Omega^{(j_2-j_1,0),(0,0)})_{2^{j_1-j_2}k_1-k_2}.
\end{aligned} \tag{66}$$

For the cases where $j_1 < j_2$ we can use the symmetry property

$$\Omega^{(j_1,j_2),(N_1,N_2)}_{(k_1,k_2)} = (-1)^r \Omega^{(j_2,j_1),(N_2,N_1)}_{(k_2,k_1)},$$

which allows us to only compute half of the coefficients. The truncated differential operator for a function, $f^\epsilon = \sum_{(j,k)\in I^\epsilon_n} t_{j,k}\lambda_{j,k}$, is defined by

$$D^{\epsilon,d^r/dx^r}_n(f^\epsilon) = \sum_{(j_1,j_2,k_1,k_2)\in I^\epsilon_D \cap I^\epsilon_f} t_{j_1,k_1} \Theta_{(j_1,j_2)+(M(j_1),M(j_2)),(k_1,k_2),(M(j_1),M(j_2))} \lambda_{j_2,k_2}. \tag{67}$$

The reduced fast operator transform for the differential operator is shown in Figure 4.

Example 2 The multiplication operator is an LDT-operator with two-dimensional Ωs and $\nu = \sqrt{2}$. There are eight equations corresponding to (66). The operator has the symmetry property

$$\Omega^{(j_1,j_2,j_3),(N_1,N_2,N_3)}_{(k_1,k_2,k_3)} = \Omega^{P((j_1,j_2,j_3)),P((N_1,N_2,N_3))}_{P((k_1,k_2,j_3))},$$

for any permutation of the indices, P, allowing us to compute only one sixth of the coefficients. As we only need the coarsest level for the scaling functions the only combinations of Ωs we need are $\Omega^{\mathbf{j},\mathbf{M}(\mathbf{j})}_{\mathbf{k}}$, $\mathbf{j} \in \{n_0,\ldots,n-1\}^3$. The reduced operator transform is shown in Figure 5.

The only task left is finding the coefficients $\Omega^{\mathbf{0},\mathbf{1}}_{\mathbf{k}}$. Here we can again use (10) which will give a system of linear equations. This approach was introduced in [5], where it was used for finding the coefficients for the differential operator. It can be generalized to LDT-operators, although the uniqueness of the solution to the resulting system of equations yet has to be studied.

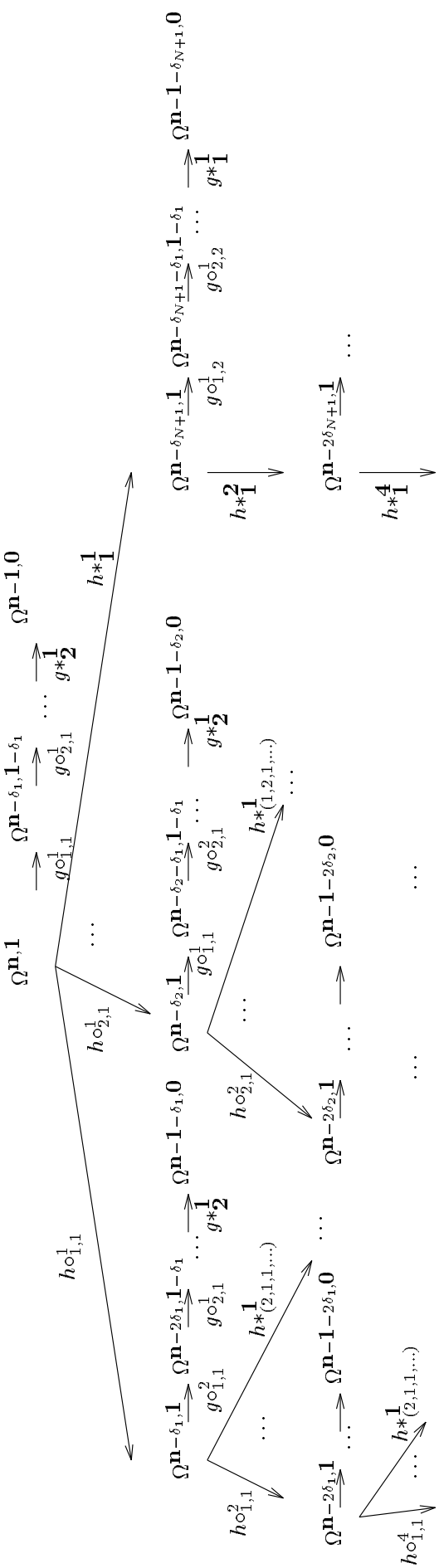

Figure 3: *The pyramid fast operator transform (where* $\mathbf{t}$ *stands for* $t\mathbf{1}$*).*

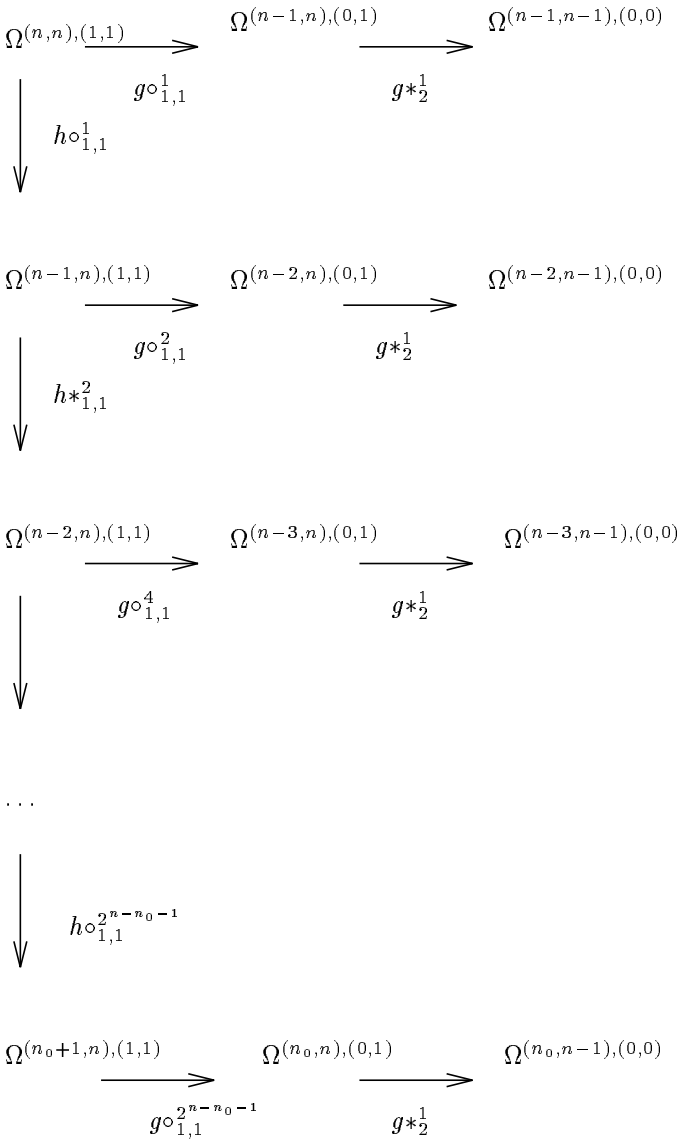

Figure 4: *The reduced fast operator transform for the differential operator.*

Figure 5: *The reduced fast operator transform for the multiplication operator.*

3.4 Tree representation

Since we want to use wavelets to obtain sparse representations of functions, we need a data structure that takes advantage of this sparsity. In this section we will describe such a sparse data structure.

When implementing algorithms that involve thresholded wavelet expansions, such as $E_n^\epsilon f(x)$, defined by Equation (43), we need an efficient representation of this sparse data structure. There are several criteria that such a data structure has to meet. It must be memory efficient and it must also permit fast evaluation of operations on the thresholded wavelet expansion, such as addition, multiplication and differentiation.

We propose a tree structure for the representation of thresholded wavelet expansions. The tree is a binary tree with added pointers to parents and nearest neighbors. Each node is a block of consecutive wavelet coefficients. The number of coefficients in each block is denoted n_b and must be an even power of two. An example of such a tree is presented in Figure 6.

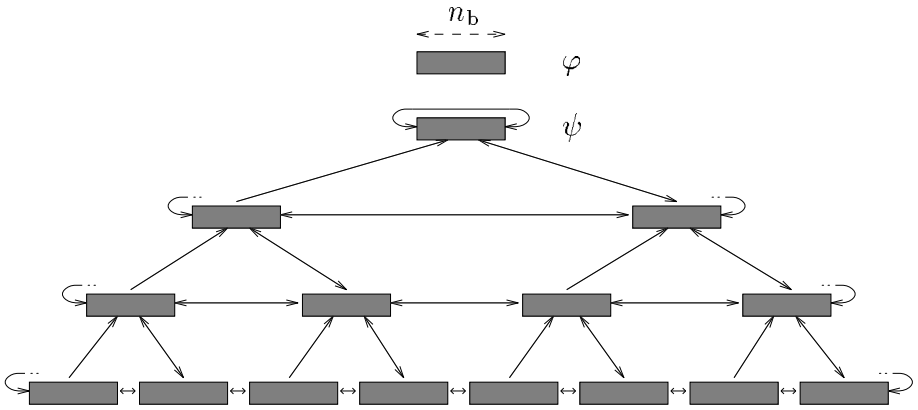

Figure 6: *Schematic picture of the block tree.*

We now define the mapping of the thresholded wavelet expansion on the tree more precisely. We have the same pyramid structure as for the ordinary wavelet expansion of a function (42), except that we now have blocks of coefficients instead of single coefficients. The scaling coefficients are stored in a block $(s_0 s_1 \cdots s_{n_\mathrm{b}-1})$ and the wavelet coefficients are stored in blocks $(d_{j,kn_\mathrm{b}} d_{j,kn_\mathrm{b}+1} \cdots d_{j,kn_\mathrm{b}+n_\mathrm{b}-1})$, where the index j is the level as before but $k, 0 \leq k \leq 2^j - 1$, is now the block index. Since we now are dealing with blocks of coefficients, we also have to redefine the set I_n^ϵ. Instead of retaining single coefficients after thresholding, we retain whole blocks. There are several norms that we can use for this block thresholding. One is that we retain all blocks that have at least one coefficient with magnitude greater than the threshold value, ϵ. Another strategy would be to calculate the l^2 norm, $\sum d_{jk}^2$, of each block. An example of a tree after thresholding is shown in Figure 7.

The binary tree representation is natural for wavelets with a dilation factor of two since the number of wavelet coefficients doubles at each level. Also, blocks on the same level that are adjacent in space are neighbors in the tree. Between levels, the wavelet coefficients in a parent block have approximately the same support as its

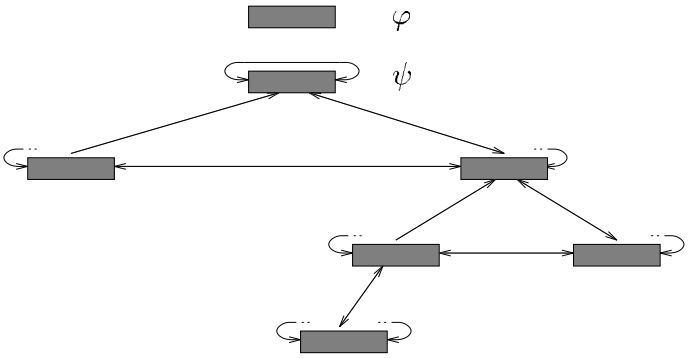

Figure 7: *Schematic picture of a thresholded block tree.*

children.

The reason behind choosing a block representation is to spread the the cost of memory management and pointer references over n_b wavelet coefficients. The number of wavelet coefficients in each block, n_b, has to be chosen with two competing aims in mind. If on one hand n_b is small we can get a sparser tree (the total number of coefficients after thresholding is small), but on the other hand if n_b is large we get less overhead.

3.5 Operations on trees

In this section we will briefly describe the algorithms for performing operations such as addition, differentiation and multiplication of the sparse trees described in the previous section.

First of all, some generalities that hold true for all tree operations. The result tree is built from the root and downwards computing one block at a time. When a block norm is smaller than the threshold value we do not compute any blocks in the subtree of that block. This can be justified by the fact that for functions with no worse singularities than discontinuities, the wavelet coefficients decrease asymptotically when the level, j, increases.

3.5.1 Addition

Since the addition of two wavelet expansions simply amounts to addition of wavelet coefficients with equal index, we build the result tree, including all blocks that are non-zero in either one of the operand trees. This is an $\mathcal{O}(N_\mathrm{s})$ operation.

3.5.2 Differentiation

To compute the coefficients in a block in the result tree, we iterate over the block's neighbors in the original tree. This is an $\mathcal{O}(c_\mathrm{d} N_\mathrm{s})$ operation, where c_d depends on the wavelet we use and the precision we want.

3.5.3 Multiplication

We proceed as in the case of differentiation, except that we now have two trees, which amounts to an extra inner loop in the calculations. This is an $\mathcal{O}(c_{\mathrm{m}} N_{\mathrm{s}}^2)$ operation, where c_{m} again depends on the wavelet we use and the precision we want.

3.6 Periodization

There are several approaches for constructing wavelets on the interval, see e.g. L. Andersson et al. [1], or P. Auscher [2]. We choose the simplest, to work with periodized wavelets. Some slight modifications have to be made when working on a periodic domain. For the signal, this means that

$$t_{j,k}^{\mathrm{per}} = \langle f, \lambda_{j,k}^{\mathrm{per}} \rangle_{(0,1)} = \sum_l t_{j,k+2^{j+M(j)}l}. \tag{68}$$

We know that $|\mathrm{supp}\varphi_{j,k}| = 2^{-j}(m-1)$, where m is the number of non-zero h coefficients. This means that for a choice of, n_0 such that $2^{n_0+1}+1 \geq m$, the pyramid algorithm can be modified, simply by treating the filters, $\mathbf{s}^{n_0+1}$, $\mathbf{d}^j$, as circulant. In the sparse representation of the signal this carries over to the "circular arrows" at the edges of each level in Figure 6.

The situation is the same for the operator. The new coefficients will be defined as

$$\begin{aligned} \Theta_{\mathbf{j},\mathbf{k},\mathbf{N}}^{\mathrm{per}} = \sum_{\mathbf{r}\in\mathbb{Z}^N} \Theta_{\mathbf{j},\mathbf{k}+(2^{(\mathbf{j})_1}(\mathbf{r})_1, 2^{(\mathbf{j})_2}(\mathbf{r})_2, \ldots, 2^{(\mathbf{j})_N}(\mathbf{r})_N, (\mathbf{k})_{N+1}), \mathbf{N}}, \\ \mathbf{j} \in \mathbb{N}^{N+1}, \qquad \mathbf{N} \in \{0,1\}^{N+1}, \\ (\mathbf{k})_i \in \{0, \ldots, 2^{(\mathbf{j})_i} - 1\}, \qquad i = 1, \ldots, N+1. \end{aligned} \tag{69}$$

For *support dependent* operators, i.e, operators with $\Theta_{\mathbf{j},\mathbf{k},\mathbf{N}} = 0$ if $|\mathrm{supp}\gamma^l_{(\mathbf{j})_l,(\mathbf{k})_l} \cap \mathrm{supp}\gamma^s_{(\mathbf{j})_s,(\mathbf{k})_s}| = 0$ for some l, s, we can use circulant versions of the N-dimensional autocorrelation and convolution, $\circ_{s,t}^{j,\mathrm{per}}$, $*_{\mathbf{t}}^{\mathbf{j},\mathrm{per}}$, as long as $2^{(\mathbf{j})_i - 1}+1 \geq m$, $i = 1, \ldots, N+1$. The recursive formulae, (64) are exactly the same but with the operations, $\circ_{s,t}^{j,\mathrm{per}}$, $*_{\mathbf{t}}^{\mathbf{j},\mathrm{per}}$.

4 Numerical results

In this section we solve two one-dimensional PDEs with periodic boundary conditions; the linear advection equation and the non-linear Burgers' equation. For time-stepping we use an explicit 4th order Runge–Kutta scheme. We also discuss the number of arithmetic operations (a.o.) needed for differentiation and multiplication in a wavelet basis.

4.1 Adaptive time-steps

Since the size of the time-step, Δt, is restricted by the number of levels in the wavelet expansion, i.e., the mesh size on the finest level, we choose a time-step that is dependent of the number of levels in the wavelet expansion at each time. This is done by doubling Δt when the solution tree shrinks by one level, and by halving Δt when the solution tree grows by one level.

4.2 The advection equation

We want to solve

$$\begin{cases} u_t = u_x, & 0 \le x \le 1, \quad t \ge 0, \\ u(x,0) = f(x), & u = u(x,t), \quad u(0,t) = u(1,t). \end{cases} \tag{70}$$

The solution is a translation of the initial condition, $u(x,t) = f((x+t) \bmod 1)$. We choose an initial condition $f(x)$, that is smooth in most of the domain except near $x = 0.5$ where we have a "spike." This initial function is a sum of a sine wave and a Gaussian as described in Section 3.2. The thresholded wavelet expansion of the solution $u(x,t)$, will have few coefficients, except for in the neighborhood of the "spike" at $x = 0.5 - t$.

In our first example we chose a spike with an effective width of 3×10^{-2}. The initial function is shown in Figure 8. We stepped forward to $t = 0.3$, where the solution looks like in Figure 9. Too *see* the effects of the thresholding we chose large threshold values. The problem was solved with the Daubechies wavelet, D_8, with $|\mathrm{supp}\,\varphi| = 7$, and four vanishing moments. The time-step was chosen as $\Delta t = 1 \times 10^{-3}$ to ensure stability. The threshold for the blocks was chosen as $\epsilon_\varphi = 5 \times 10^{-3}$ and the operator was truncated with $\epsilon_{d/dx} = 5 \times 10^{-2}$. The block-size was $n_\mathrm{b} = 16$ and six different scales were used in the tree. Furthermore, the number of different scales, $d = j_{\max} - j_{\min}$, used in $\Theta^{d/dx}$ was $d = 3$. The number of blocks in the signal varied between 4 and 7. The approximated solution at $t = 0.3$ is shown in Figure 10. We see here that the error is large around the spike, and that we have some overshoot for the spike. The L^2 norm of the exact solution is 0.717168 and for the approximated solution it is 0.716673. For larger thresholds, the norm of the approximated solution decreases, which is natural as energy is lost when doing the thresholding.

We next considered a "nastier" problem, with an effective spike width of 5×10^{-3}. The initial function is shown in Figure 11. We stepped forward to $t = 0.3$, where the solution looks like in Figure 12. Once again, we used the Daubechies wavelet, D_8. The time-step was chosen as $\Delta t = 1 \times 10^{-4}$. The threshold for the blocks was chosen as $\epsilon_\varphi = 1 \times 10^{-8}$ and the operator was truncated with $\epsilon_{d/dx} = 1 \times 10^{-7}$. The block-size was chosen as $n_\mathrm{b} = 16$. To resolve the spike, nine scales were needed, and this number was chosen automatically by the thresholding procedure described earlier. The error of the approximated solution at $t = 0.3$ is shown in Figure 13. We see here that with these values for the thresholding, the error is small. It is also worth noticing that the thresholding makes the error stay localized near the spike. This nice property was observed for all the experiments.

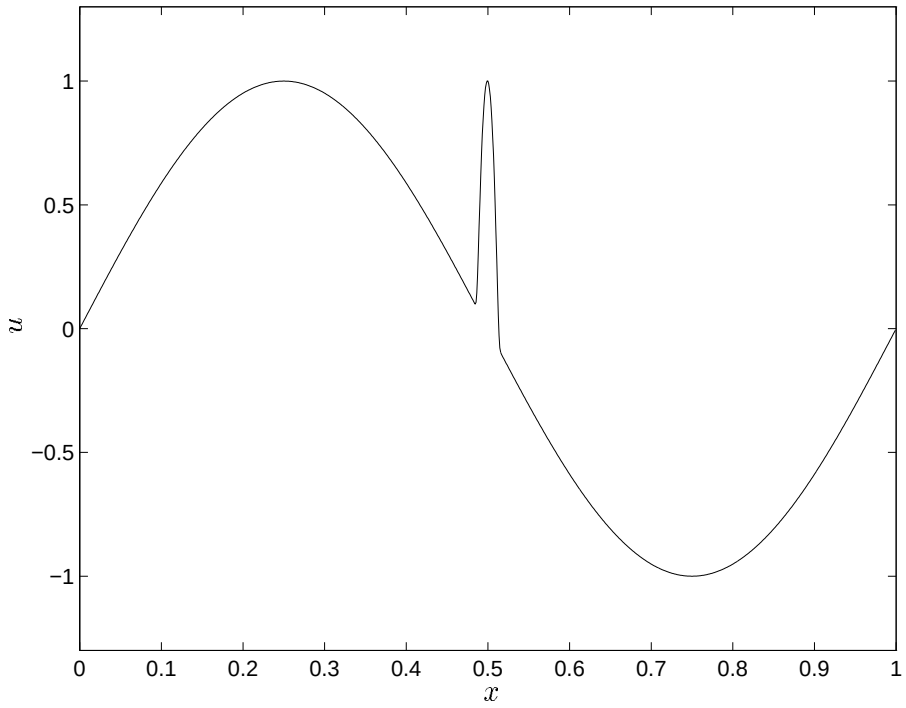

Figure 8: *Initial function for advection equation,* $u_t = u_x$*, spike width* 3×10^{-2}.

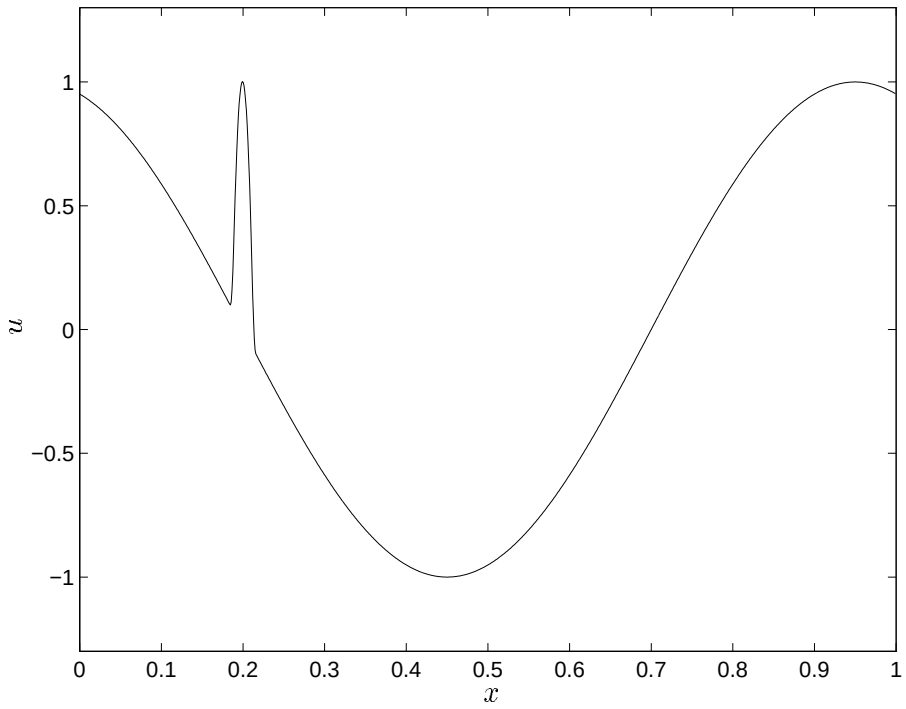

Figure 9: *Solution for advection equation,* $t = 0.3$*, spike width* 3×10^{-2}.

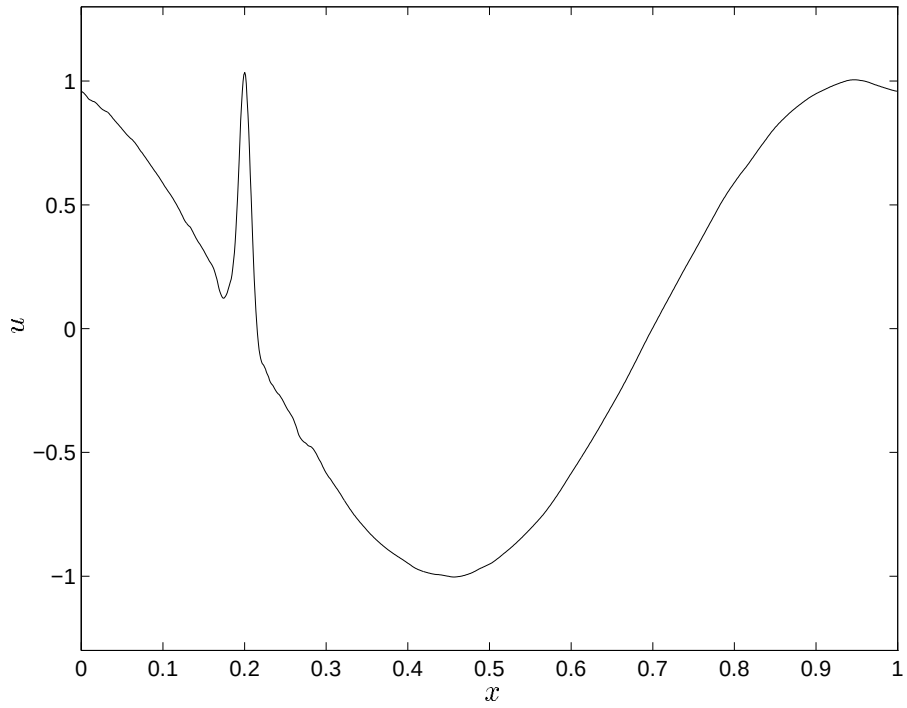

Figure 10: *Approximated solution for advection equation,* $t = 0.3$.

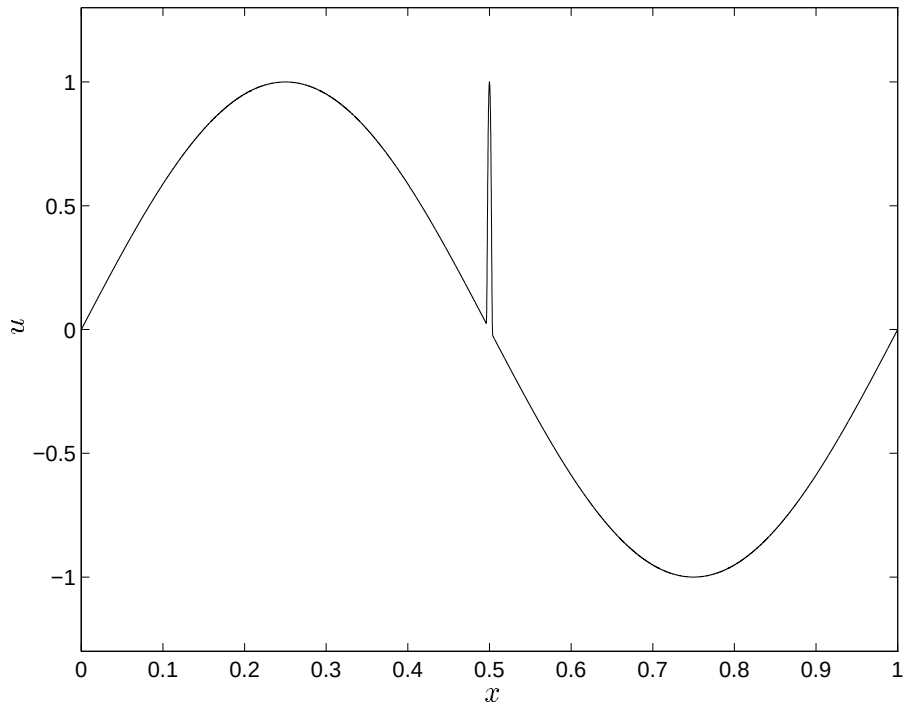

Figure 11: *Initial function for advection equation,* $u_t = u_x$*, spike width* 5×10^{-3}.

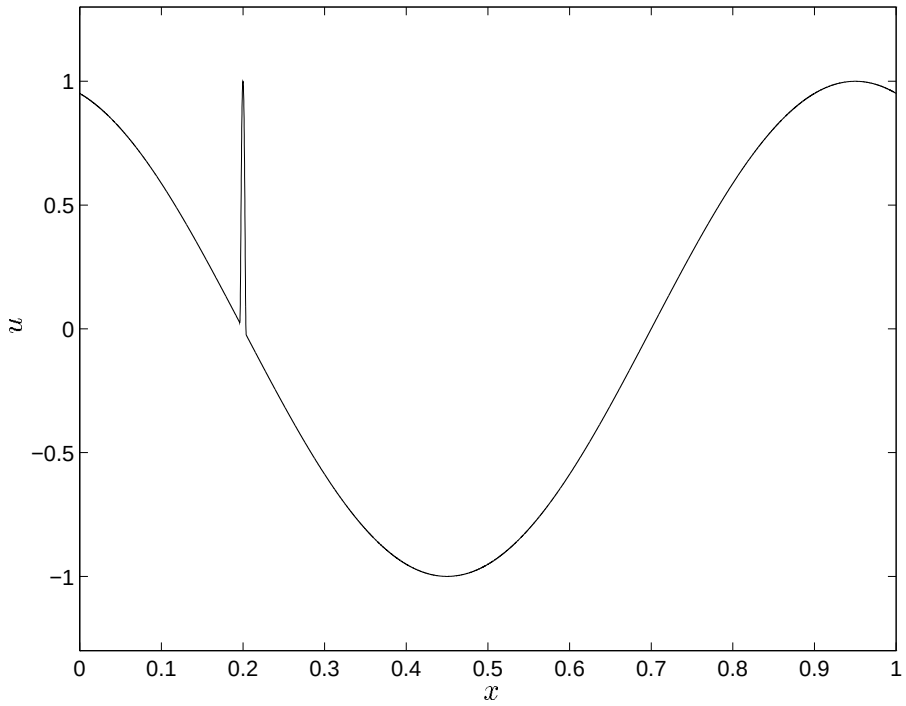

Figure 12: *Solution for advection equation,* $t = 0.3$*, spike width* 5×10^{-3}.

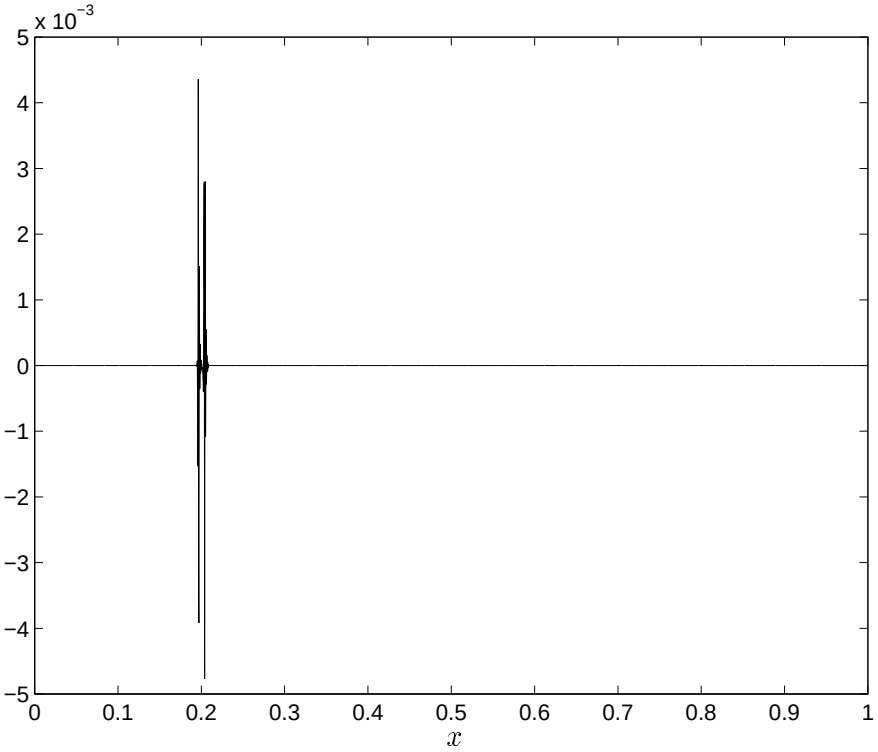

Figure 13: *Error in approximated solution with* D_8 *for advection equation,* $t = 0.3$.

The number of blocks in the signal varied between 19 and 27. The bottle-neck when using this wavelet was d. In the previous example eight levels were needed for the method to be numerically stable. An example where the number was reduced to six is shown in Figure 14, after only five time-steps. The same happened with seven levels, but not as quickly. By increasing the length of the support of the wavelets d

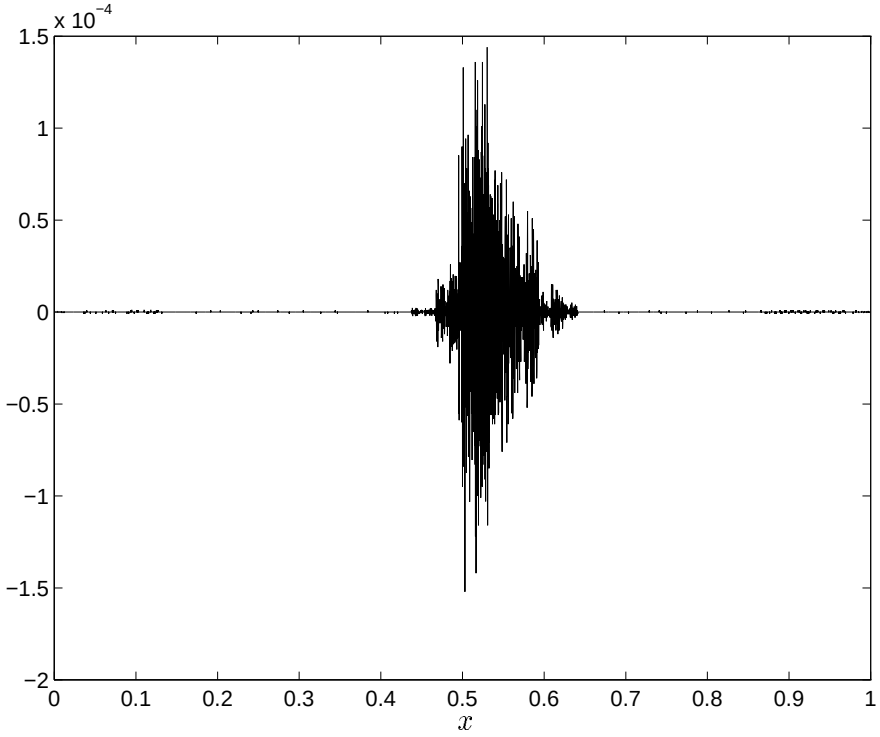

Figure 14: *Error in approximated solution after five time steps with* $j_{\max} - j_{\min} = 6$.

can be decreased. If we choose the Daubechies scaling function, D_{20}, it is enough to choose $d = 5$. The error in the approximated solution at $t = 0.3$ with $\epsilon_\varphi = 1 \times 10^{-8}$, $\epsilon_{d/dx} = 1 \times 10^{-7}$ and $\Delta t = 1 \times 10^{-4}$ is shown in Figure 15. The number of blocks used to represent the signal decreases to 10–14.

The number can further be decreased by using the wavelets proposed for hyperbolic PDEs in [17]. We use the scaling function $W_{20}^{(2,2)}$ (corresponding to a wavelet with the same filter length as D_{20}, but with four less vanishing moments). For the problem with the same data, $\epsilon_\varphi = 1 \times 10^{-8}$, $\epsilon_{d/dx} = 1 \times 10^{-7}$ and $\Delta t = 1 \times 10^{-4}$, it is enough to have $d = 3$ to represent the differential operator. The error in this case is shown in Figure 16. The number of blocks varies between 10 and 15. The error in L^2 and L^∞ norm for the three wavelets is shown in Table 1. We see here that the errors for the different wavelets are approximately the same, in both norms.

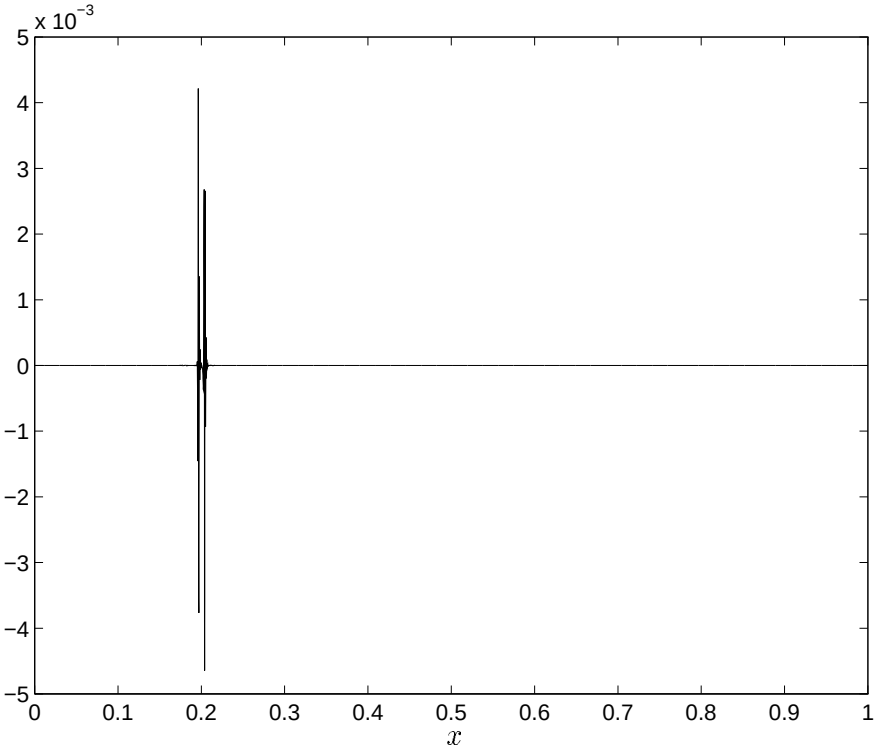

Figure 15: *Error in approximated solution for* D_{20} *at* $t = 0.3$.

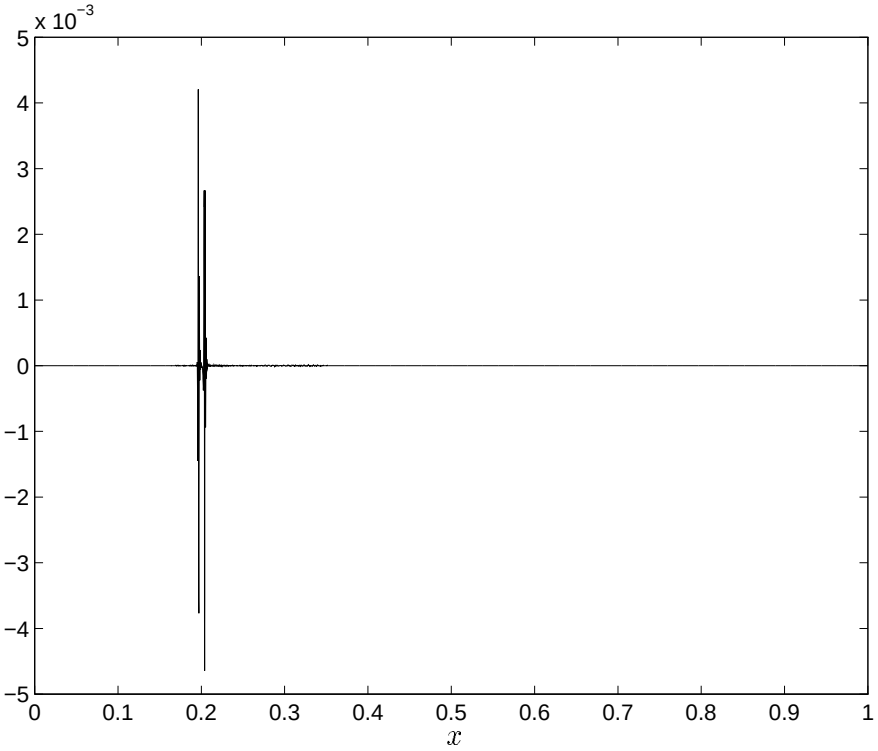

Figure 16: *Error in approximated solution for* $W_{20}^{(2,2)}$ *at* $t = 0.3$.

Scaling function	$\|e\|_{L^2} \times 10^{-4}$	$\|e\|_{L^\infty} \times 10^{-3}$
D_8	1.6017	4.7733
D_{20}	1.5206	4.6442
$W_{20}^{(2,2)}$	1.5423	4.6441

Table 1: *Error with different wavelets for advection equation, $u_t = u_x$, with effective spike width 5×10^{-3}, at $t = 0.3$.*

4.3 Burgers' equation

We want to solve the non-linear equation

$$\begin{cases} u_t + uu_x = \mu u_{xx}, & 0 \le x \le 1, \quad t \ge 0, \mu > 0, \\ u(x,0) = f(x), & u = u(x,t), \quad u(0,t) = u(1,t). \end{cases} \tag{71}$$

The exact solution, given by the Cole–Hopf transformation [9, 13], can for the case $f(x) = \sin(2\pi x)$ be stated as

$$u(x,t) = \frac{\int_{-\infty}^{\infty} \sin(2\pi(x-y)) \exp\left[\frac{1}{4\mu}\left(\frac{\cos(2\pi(x-y))}{\pi} - \frac{y^2}{t}\right)\right] dy}{\int_{-\infty}^{\infty} \exp\left[\frac{1}{4\mu}\left(\frac{\cos(2\pi(x-y))}{\pi} - \frac{y^2}{t}\right)\right] dy}. \tag{72}$$

This solution, for $\mu = 10^{-3}$, is plotted at different times in Figure 17. We can note that the initially smooth function has evolved into a sawtooth function at $t = 0.3$, with a sharp gradient at $x = 0.5$. This is another example of a function that is efficiently represented in a wavelet basis since we will have few wavelet coefficients, except in the vicinity of $x = 0.5$.

We now turn to numerical experiments. In Figure 18 we can see what happens when the resolution is not good enough. The block size $n_{\mathrm{b}} = 16$, the wavelet is D_8, $\Delta t = 5 \times 10^{-3}$ and $\mu = 10^{-4}$. The truncations used are $\epsilon_\varphi = 10^{-6}$ and $\epsilon_{d/dx} = \epsilon_{\mathrm{mul}} = 10^{-12}$, and the corresponding norm is a block l^2-norm. The dashed line is the approximated solution at $t = 0.14$ computed using only one scale (corresponding to 32 points). This is not enough and leads to spurious oscillations around the sharp gradient. The solid line shows that when we allow the number of scales to increase to three (128 points), the resolution is sufficient. Note that this increase in scales is handled automatically by the solver, since we add new levels to the tree when the coefficients in a block is larger than the threshold.

In Figure 19 we can see a thresholded block representation of the exact solution at time $t = 0.3$. At finer levels we only have two blocks around the sharp gradient at $x = 0.5$.

In Figure 20 we examine the error after one time-step at $t = 0.3$. Here $\Delta t = 10^{-4}$, the wavelet is D_8, $\mu = 10^{-3}$, $n_{\mathrm{b}} = 16$, $\epsilon_\varphi = 10^{-6}$ and $\epsilon_{d/dx} = \epsilon_{\mathrm{mul}} = 10^{-6}$. We limit the interaction between scales to four scales for the multiplication.

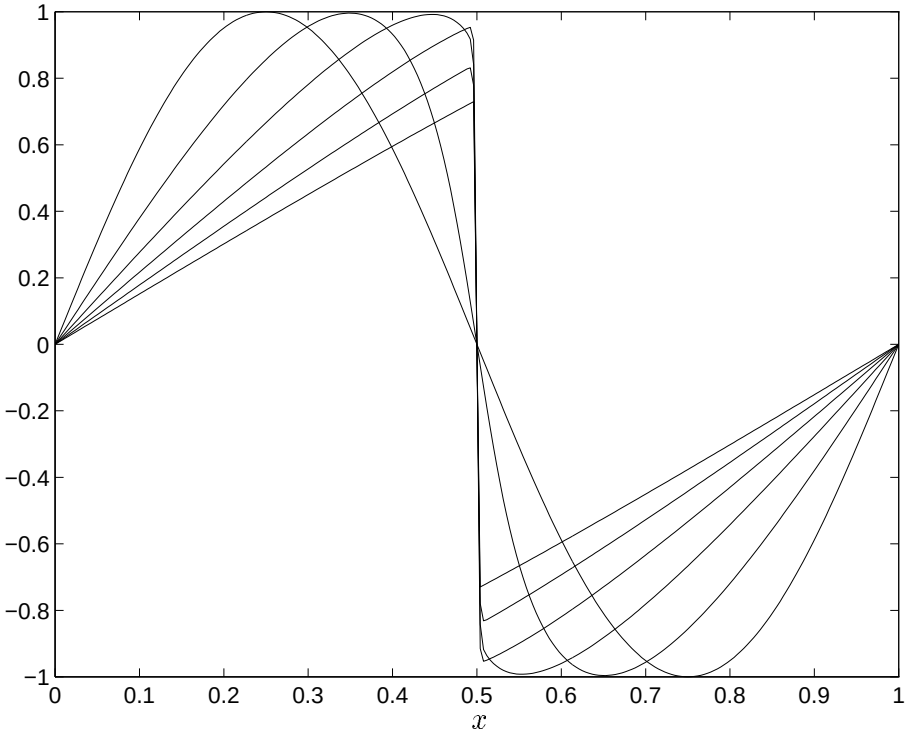

Figure 17: *The exact solution of Burgers' equation when the initial function* $f(x) = \sin(2\pi x)$, *at times* $t = 0, 0.1, 0.2, 0.3, 0.4$ *and* 0.5. *The viscosity* $\mu = 10^{-3}$.

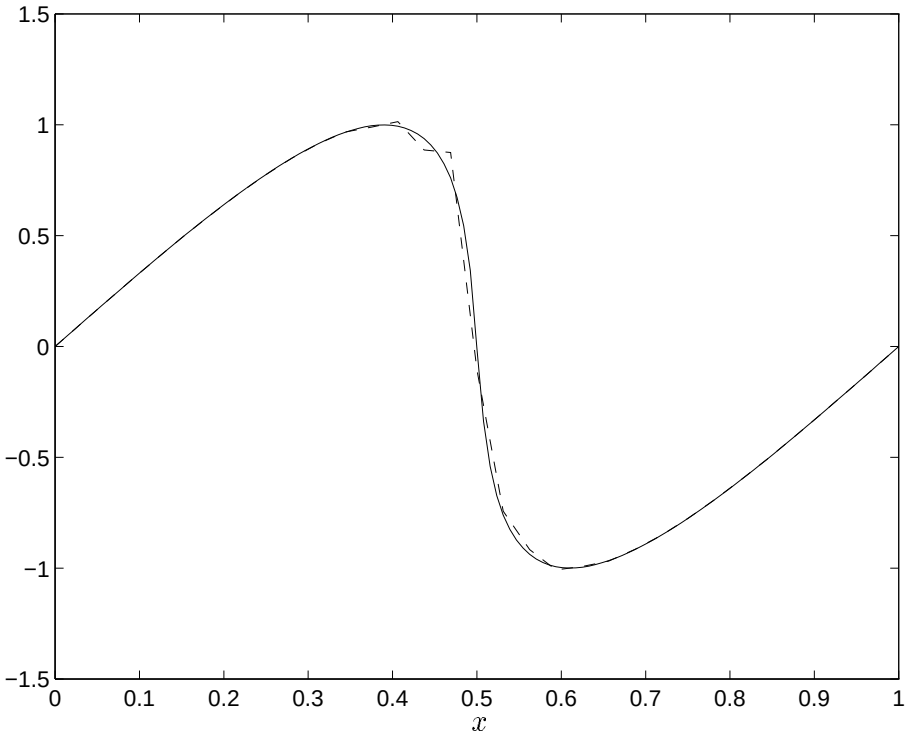

Figure 18: *Solutions at* $t = 0.14$. *The dashed line is the one level approximated solution and the solid line the three level approximated solution.*

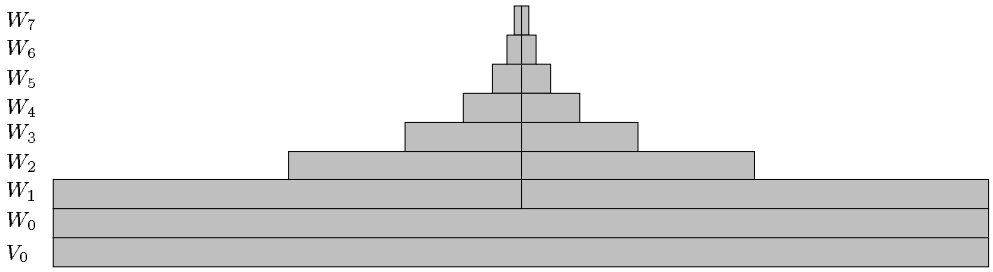

Figure 19: *Block representation of the solution at* $t = 0.3$ *for* $\mu = 10^{-3}$ *with* $\epsilon_\varphi = 10^{-6}$. *The wavelet is* D_8. *The block size* $n_\mathrm{b} = 16$, *and we have eight levels in the wavelet tree. This corresponds to 4096 points on the finest level* V_8.

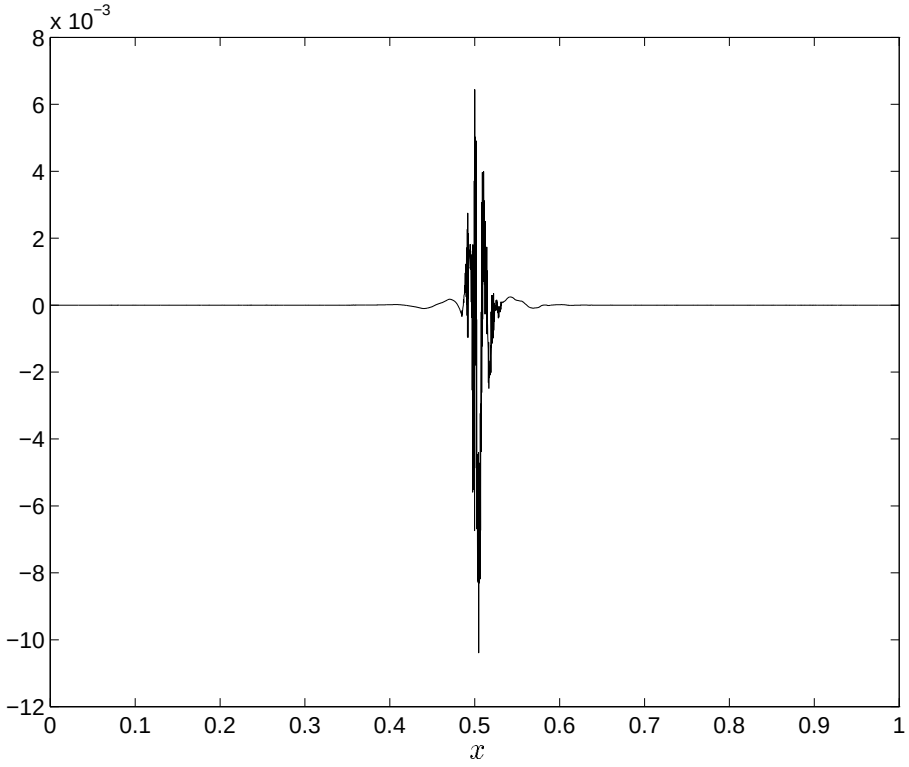

Figure 20: *The error of the numerical solution at* $t = 0.3001$ *after one time-step of length* $\Delta t = 10^{-4}$. *The wavelet is* D_8 *and* $\mu = 10^{-3}$.

4.4 Work estimates

As stated earlier, in Section 3.5, differentiation is an $\mathcal{O}(c_{\mathrm{d}}N_{\mathrm{s}})$ operation in a wavelet basis, and the multiplication of two functions an $\mathcal{O}(c_{\mathrm{m}}N_{\mathrm{s}}^2)$ operation. We now try to estimate the size of the constants in these work estimates. The function v, that we choose to study is the exact solution to Burgers' equation at $t = 0.3$, and is plotted in Figure 17. For the derivative we examine v_x and for the multiplication $v_x v$. We then study the number of arithmetic operations (a.o.) needed as a function of the number of levels j, or equivalently the number of points on the finest scale N. The wavelet used is D_8 and we limit the interaction between scales to three scales.

In Figure 21 we compare the thresholded wavelet derivative with a five-point centered finite difference derivative that uses $7N$ a.o. As can be seen, the break-even point, where it is advantageous to use wavelets, is around $N = 2^{11} = 2048$ points on the finest scale.

In Figure 22 we compare the thresholded wavelet product with point-wise multiplication on the finest scale (N a.o.). We make an approximate extrapolation of the number of a.o. for the wavelet product and find the break-even around $N = 2^{23} \approx 8.4 \times 10^6$ points.

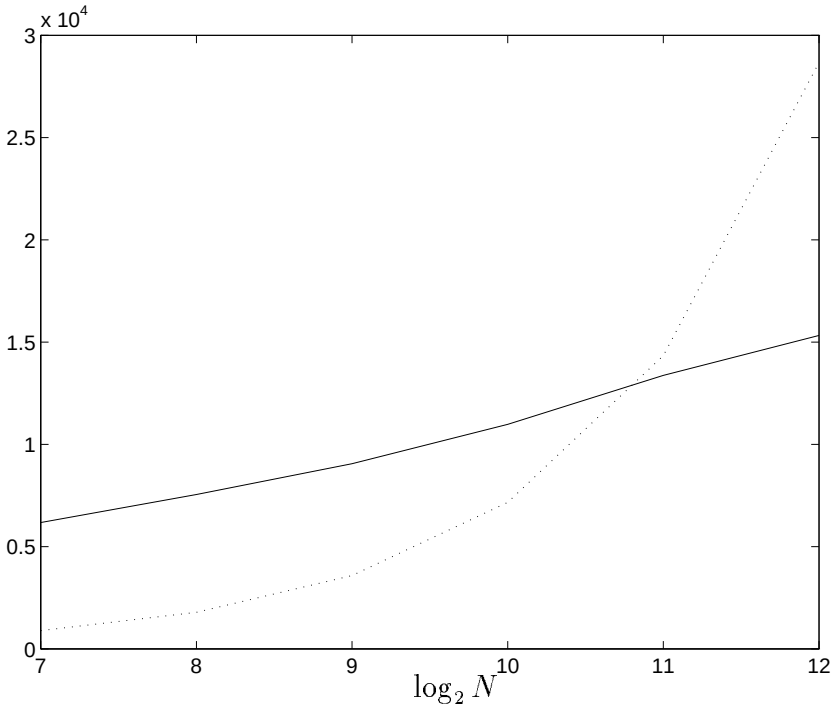

Figure 21: *The solid line shows the number of* a.o. *needed to compute the derivative in a wavelet basis. The dotted line is the number of* a.o. *using five point finite differences on the finest scale.*

A natural way to construct a multiresolution analysis in two dimensions is by using the tensor product (see [10]),

$$\mathcal{V}_{j+1} = \mathcal{V}_j \oplus \mathcal{W}_j, \tag{73}$$

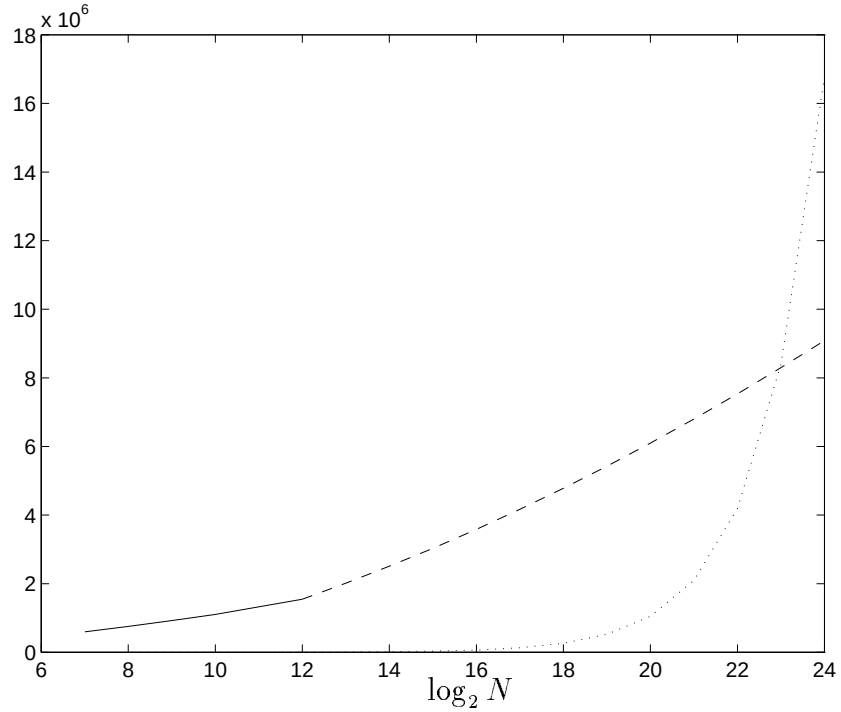

Figure 22: *Number of a.o. to multiply u with u_x.*

where

$$\begin{aligned} \mathcal{V}_j &= V_j \otimes V_j, \\ \mathcal{W}_j &= (V_j \otimes W_j) \oplus (W_j \otimes W_j) \oplus (W_j \otimes V_j). \end{aligned} \tag{74}$$

This approach leads to wavelets with the same support width in both dimensions. For these wavelets, differentiation will be an $\mathcal{O}(N_{\mathrm{s}})$ operation and multiplication an $\mathcal{O}(N_{\mathrm{s}}^2)$ operation, although the constants involved will be larger in higher dimensions. This makes the wavelet method well-suited for problems with isolated large gradients (for example problems with point sources), as $N_{\mathrm{s}} = \mathcal{O}(\log N)$ for such problems. Thus, in higher dimensions the wavelet method will be relatively more advantageous compared with the finite difference scheme.

5 Conclusions

In this paper we have presented a methodology for solving hyperbolic PDEs in a wavelet basis with computations done to a prescribed accuracy. The number of wavelet coefficients that represent the solution is increased or decreased adaptively by thresholding, during the computations to keep this accuracy. The operators are also thresholded.

The method can be viewed as an adaptive mesh method, where the mesh is automatically refined around sharp variations of the solution. An advantage of the

method is that we never have to care about where and how to refine the mesh. All this is handled by thresholding the wavelet coefficients.

The method is well suited for large problems with a solution that is well compressed in a wavelet basis. This is the case for functions that are mostly smooth with well localized sharp variations. Then the number of significant wavelet coefficients N_s, is proportional to $\log N$, where N is the number of points on the finest grid. The work required to differentiate a function is then proportional to N_s, and to multiply two similar functions the work is proportional to N_s^2. These work estimates also hold for problems in higher dimensions.

The constants in the estimates are large. They can be reduced by using other wavelets than Daubechies wavelets but the problem size has to be large before the wavelet method outperforms classical methods applied to the finest grid. This is especially true for the case of multiplication due to the quadratic dependency on N_s. Another consideration for the multiplication is the large amount of memory needed to store the coefficients of the operator. To make this method efficient also for problems of smaller size, a new approach to the multiplication might be needed.

References

[1] L. Andersson, N. Hall, B. Jawerth, and G. Peters. Wavelets on closed subsets of the real line. In *Wavelet Analysis and its Applications-Recent Advances in Wavelet Analysis*, volume 3, pages 1–61. Academic Press, 1994.

[2] P. Auscher. Ondelettes à support compact et conditions aux limites. *Journal of Functional Analysis*, 111(1):29–43, 1993.

[3] E. Bacry, S. Mallat, and G. Papanicolaou. A wavelet based space-time adaptive numerical method for partial differential equations. *Math. Mod. Num. Anal.*, 26(793), 1992.

[4] S. Bertoluzza and G. Naldi. A wavelet collocation method for the numerical solution of partial differential equations. Technical Report 887, Istituto di Analisi Numerica del Consiglio Nazionale Delle Riceche, Italy, 1993.

[5] G. Beylkin. On the representation of operators in bases of compactly supported wavelets. *SIAM J. Numer. Anal.*, 6(6):1716–1740, 1992.

[6] G. Beylkin. On the fast Fourier transform of functions with singularities. University of Colorado at Boulder, 1994.

[7] G. Beylkin, R. Coifman, and V. Rokhlin. Fast wavelet transforms and numerical algorithms I. *Comm. Pure and Appl. Math*, 44:141–183, 1991.

[8] B. Bihari and A. Harten. Application of generalized wavelets: an adaptive multiresolution series. Presented at the International conference on Wavelets, Taormina, Italy, October 14-20, 1993.

[9] J. D. Cole. On a quasilinear parabolic equation occurring in aerodynamics. *Quarterly of applied mathematics*, 9:225–236, 1951.

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

[11] P. Fisher and M. Defranceschi. Representation of the atomic hartree-fock equations in the wavelet basis by means of the BCR algorithm. In *Wavelet Analysis and its Applications*, volume 5, pages 495–508. Academic Press, 1994.

[12] G. Fix and G. Strang. A Fourier analysis of the finite element variational method. In *Constructive aspects of Functional Analysis*, Rome, 1973.

[13] E. Hopf. The partial differential equation $u_t + uu_x = \mu u_{xx}$. *Comm. Pure Appl. Math.*, 3:201–230, 1950.

[14] J. Keiser. *Wavelet Based Approach to Numerical Solution of Nonlinear Partial Differential Equations*. PhD thesis, University of Colorado, 1995.

[15] S. Mallat. Multiresolution approximations and wavelet orthonormal bases of $L^2(\mathbb{R})$. *Trans. Am. Math. Soc.*, 315(1):69–87, 1989.

[16] W. Sweldens and R. Piessens. Quadrature formulae and asymptotic error expansions for wavelet approximations of smooth functions. *SIAM J. of Numer. Anal.*, 31(4):1240–1264, August 1994.

[17] J. Waldén. Orthonormal compactly supported wavelets for solving hyperbolic PDEs. Technical Report 170, Uppsala University, Dept. of Scientific Computing, Box 120, Uppsala, Sweden, 1995.