Filter Bank Methods for Hyperbolic PDEs*

Johan Waldén†

October 2, 1996

*This work was supported by the Swedish Natural Science Research Council, NFR, under grant F–FU 04370–304.
†Uppsala University, Department of Scientific Computing, P.O. Box 120, S–751 04 Uppsala, Sweden.

Abstract

We use biorthogonal filter banks to solve hyperbolic PDEs adaptively with a sparse multilevel representation of the signal. The methods described are of finite difference type, and the filter banks are used to give a sparse representation of signals, and to transform between grids on different scales. We derive bounds for the error and number of coefficients in the sparse representation. These bounds also apply for filter banks that are not associated with any wavelets. We develop algorithms for fast differentiation and multiplication in detail. The strength of the method is shown in various test problems.

Contents

1 Introduction

We wish to solve time dependent initial-value problems of hyperbolic type,

$$\begin{cases} u_t = Pu, \\ u(x,0) = u_0(x), \\ x \in \mathbb{R}^n,\ t \geq 0, \end{cases} \tag{1}$$

where the solution is smooth in large parts of the domain, but has small regions of sharp gradients. For such problems wavelets offer a way of solving the equation adaptively. This was used in [17] to develop a solver working in the wavelet domain, based on a Galerkin method. The method used orthonormal Daubechies wavelets, and worked satisfactorily for PDEs with constant coefficients. Unfortunately it was not efficient for multiplication with non-constant functions, a problem that seems hard to avoid when working purely in the wavelet domain. This has also been noticed elsewhere, e.g., in [16, 19, 30].

On the contrary, multiplication is easy in the domain of scaling coefficients. Therefore, we need a method that works both in the wavelet and scaling coefficient domain. Multiplication will be especially simple when the scaling coefficients can be viewed as point values. This was used in [21] with coiflets, and in [16, 27, 30, 31], with collocation methods. When a collocation method is used, a problem is that a convolution with a filter of infinite length (but with coefficients that decrease fast) is needed, even if the wavelets have compact support. In [19, 20] finite difference methods were developed, using only point value representations of function, but using Daubechies wavelets to refine grids adaptively.

In this paper we develop a new method working both in the wavelet domain and the physical domain. We use biorthogonal filter banks that do not necessarily correspond to wavelets in L^2. The filter banks used are spline filters with duals of shortest possible support, and biorthogonal versions of coiflets, where the dual is the shortest possible, the δ-filter. The wavelet transform is seen purely as a transform of point values, which is shown to be more general and work better than the wavelet Galerkin approach. We will borrow notation from signal analysis where perfect reconstruction filter banks were originally introduced [2, 26]. The use of such filter banks will be justified theoretically. We will point out the connection to continuous theory whenever possible. Differential operators will be approximated with finite difference operators, which leads to shorter filters than when using Galerkin or collocation methods. The analysis in this paper of which filter banks should be used shows that neither the filter banks used in [21] (coiflets), nor the ones in [19, 20] (Daubechies filters) are optimal. The method of transforming point values has been used without theoretical analysis, in [13] to compress the solution for the heat equation and in [12] to get sparse representations of integral kernels when solving the Laplace equation with a boundary element method. A difference between the method presented in this paper and the methods in [19, 20] is that we only perform operators on uniform grids, which means that we avoid using non-uniform finite difference formulae. This makes the method easy to generalize to higher dimensions.

With our approach, the wavelet transform is used to perform what is its strength:

- It detects local high frequencies.
- It gives compact representations of functions with small regions of large gradients.
- It provides a 'rule' of how to go from a coarse scale representation to a finer scale.

The paper consists of the following parts. In Section 2 we introduce some preliminary notation on wavelets and filter banks. In Section 3 we introduce the truncated wavelet representation, and compare the filter bank method with the wavelet Galerkin method. We also present fast algorithms for differentiating a function and multiplying two functions in the wavelet domain. These algorithms are presented in detail and we call them Λ-cycles as they, in contrast to the multigrid V-cycle go from coarse scales to fine. Finally, in Section 4 and 5 we study a number of linear and non-linear test problems. The proposed method is shown to perform well, with compression factors of up to 110. All problems studied are with periodic boundary conditions (b.c.s), but the combination of working (partly) in the physical space, and with filters of short support implies that adding b.c.s ought not to be a major difficulty.

2 Preliminaries

Although we mainly will be concerned with the wavelet transform on $\mathbb{Z}$ we start with the continuous theory. We will use the Hilbert space $L^2(\mathbb{R})$, with the inner product

$$\langle f, g\rangle_{L^2} = \int_{-\infty}^{\infty} f\bar{g}\, dx, \tag{2}$$

and the Banach spaces $L^p(\mathbb{R})$, $L^\infty(\mathbb{R})$, with norms

$$\|f\|_{L^p} = \left(\int_{-\infty}^{\infty} |f|^p\, dx\right)^{1/p}, \qquad p \in [1, \infty), \tag{3}$$

$$\|f\|_{L^\infty} = \operatorname*{essup}_x |f(x)|. \tag{4}$$

The Hölder spaces, C^γ, are defined by

$$\begin{aligned} f \in C^\gamma \quad &\Leftrightarrow \quad \sup_{x,y} \frac{|f(x) - f(y)|}{|x-y|^\gamma} < \infty, \qquad \gamma \in (0, 1], \\ f' \in C^\gamma \quad &\Leftrightarrow \quad f \in C^{\gamma+1}, \end{aligned} \tag{5}$$

(This is a slight abuse of notation as, for $\gamma \in \mathbb{N}$, C^γ actually is the Lipschitz space, Lip_γ). We have the semi-norms on C^γ,

$$\|f\|_{\dot{C}^\gamma} = \sup_{x,y} \frac{|f^{(\lfloor\gamma\rfloor)}(x) - f^{(\lfloor\gamma\rfloor)}(y)|}{|x-y|^{\gamma-\lfloor\gamma\rfloor}}, \tag{6}$$

where, $\lfloor\gamma\rfloor$, is defined as the largest integer not larger than γ.

We define the Fourier transform of a function, $f \in L^1(\mathbb{R})$, by

$$\hat{f}(\xi) = \int f(x) e^{-i\xi x}\, dx, \tag{7}$$

and we will use the standard generalizations to other spaces. The Sobolev spaces, $H^s(\mathbb{R})$, $s \in \mathbb{R}$, are Hilbert spaces with inner products

$$\langle f, g\rangle_{H^s} = \frac{1}{2\pi} \int \hat{f}\bar{\hat{g}}(1+\omega^2)^s\, d\omega. \tag{8}$$

Downsampling of a function, $\downarrow$, is defined as $(\downarrow f)(x) = f(2x)$, and upsampling as $(\uparrow f)(x) = f(x/2)$. By $\mathcal{P}^k$, we mean the set of polynomials, of degree at most k. We define $\mathbb{T}$ as $\mathbb{R}/\mathbb{Z}$ (and *not* as $\mathbb{R}/2\pi\mathbb{Z}$ which is more common).

For the discrete theory we will borrow some notation from signal analysis. A filter (discrete function, grid function) is a function, $u : \mathbb{Z} \to \mathbb{C}$. We will sometimes use the notation u_k for $u(k)$. Some subspaces of the space of all filters, $\mathbb{C}^{\mathbb{Z}}$, are the Banach spaces, l^p, with norms

$$\|u\|_{l^p} = \left(\sum_k |u(k)|^p \right)^{1/p}, \qquad p \in [1, \infty), \tag{9}$$

and the Hilbert space, l^2, with inner product,

$$\langle u, v \rangle_{l^2} = \sum_k u(k)\overline{v(k)}. \tag{10}$$

We will use the mirror operator $\check{u}(k) = u(-k)$. The convolution of two filters is $(f * g)(k) = \sum_n f(n)g(k-n)$ (when the right hand side is defined). Downsampling of a filter is defined as $(\downarrow f)(k) = f(2k)$ and upsampling as $(\uparrow f)(k) = f(k/2)$ when k is even, and $(\uparrow f)(k) = 0$ when k is odd. In signal analysis these operations are often denoted $\downarrow 2$ and $\uparrow 2$, but as we will only work with a sub-sampling factor of 2, we choose the shorter notation.

We define the discrete Fourier transform, acting on grid functions (in l^1), and with an image consisting of 2π-periodic functions by

$$\hat{u}(\omega) = \sum_k u_k e^{-i\omega k}. \tag{11}$$

The inverse transform will then be

$$u(k) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \hat{u}(\omega) e^{i\omega k}\, d\omega, \tag{12}$$

and we have

$$\begin{aligned} \widehat{u * v} &= \hat{u}\hat{v}, \\ (\widehat{\uparrow u})(\omega) &= \hat{u}(2\omega), \\ (\widehat{\downarrow u})(\omega) &= \frac{1}{2}(\hat{u}(\omega/2) + \hat{u}(\omega/2 + \pi)). \end{aligned}$$

The convolution of u and v can be seen as applying a linear operator on v. General linear operators on filters, A, can be represented by infinite matrices, $\mathbf{A} \in \mathbb{C}^{\mathbb{Z}\times\mathbb{Z}}$ and the case of convolution arises when A is a Tœplitz operator, i.e., when $\mathbf{A}_{i,j} = \mathbf{A}_{i-j,0}\ \forall i,\ \forall j$. For linear operators we define the operator norms

$$\|A\|_p = \sup_u \frac{\|Au\|_p}{\|u\|_p}. \tag{13}$$

When combining different operators on filters we use a right-to-left rule, e.g.,
$\downarrow u * v = \downarrow (u * v)$. We will sometimes write hg instead of $h * g$ (interpreting h as the Tœplitz operator acting on g.) The adjoint, A^* of a linear operator on l^2, is the operator for which $\langle Au, v \rangle = \langle u, A^* v \rangle$, $\forall u, \forall v$. We have the following relations

$$\begin{aligned} (AB)^* &= B^* A^*, \\ \downarrow^* &= \uparrow, \\ h^* &= \check{h}, \end{aligned}$$

leading to $(\downarrow \check{h})^* = h \uparrow$.

The support of a filter is naturally defined as $\operatorname{supp} u = \{k : u_k \neq 0\}$. We will use the lower bound $m(u) = \inf \operatorname{supp} u$, the upper bound $M(u) = \sup \operatorname{supp} u$, and the length of a filter, $\operatorname{length} u = M(u) - m(u) + 1$. An easy way to display a filter with finite support is on the form,

$$u = (u_{m(u)}, u_{m(u)+1}, \ldots, u_{-1}, \underline{u_0}, u_1, \ldots, u_{M(u)-1}, u_{M(u)}),$$

where the zeroth coordinate of u is underlined. The discrete ordered interval $[a, b]_{\mathbb{Z}}$ is defined as $\{a, a+1, \ldots, b-1, b\}$.

We will go from functions on $\mathbb{R}$ to $\mathbb{Z}$ with the sampling operator, $[\ ]_h : \mathbb{C}^{\mathbb{R}} \to \mathbb{C}^{\mathbb{Z}}$, which is defined through $[f]_h(k) = f(kh)$. We will also use the convolution of a filter and a function on $\mathbb{R}$, $(u * f)(x) = \sum_k u_k f(x-k)$. A nice way to show operations on a filter is with a block diagram. For example, in Figure 1 the operation $h \uparrow\downarrow \check{\tilde{h}} u + g \uparrow\downarrow \check{\tilde{g}} u$ is shown (The operation S stands for 'split').

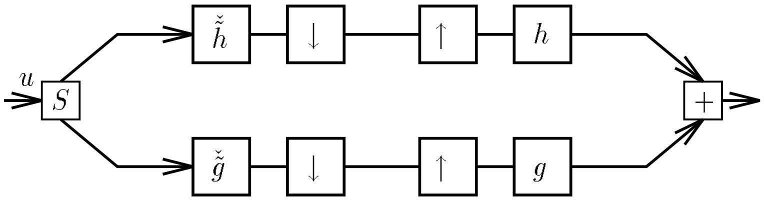


Figure 1: *Block diagram notation for operations on filters.*

We now turn to the continuous wavelet transform, by introducing the biorthogonal multiresolution analysis (MRA) [6, 24]. An MRA consists of two sequences of closed subspaces, V_j, $\tilde{V}_j$ of L^2, satisfying

$$\begin{gathered} \cdots \subset V_{-2} \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \subset \cdots, \\ \cdots \subset \tilde{V}_{-2} \subset \tilde{V}_{-1} \subset \tilde{V}_0 \subset \tilde{V}_1 \subset \tilde{V}_2 \subset \cdots, \end{gathered} \tag{14}$$

$$\overline{\bigcup_{j\in\mathbb{Z}} V_j} = \overline{\bigcup_{j\in\mathbb{Z}} \tilde{V}_j} = L^2(\mathbb{R}), \tag{15}$$

$$\bigcap_{j\in\mathbb{Z}} V_j = \bigcap_{j\in\mathbb{Z}} \tilde{V}_j = \{0\}, \tag{16}$$

$$\begin{gathered} f \in V_j \Leftrightarrow f(2^{-j}\cdot) \in V_0, \\ f \in \tilde{V}_j \Leftrightarrow f(2^{-j}\cdot) \in \tilde{V}_0, \end{gathered} \tag{17}$$

$$\begin{gathered} f \in V_0 \Leftrightarrow f(\cdot - k) \in V_0, \\ f \in \tilde{V}_0 \Leftrightarrow f(\cdot - k) \in \tilde{V}_0, \end{gathered} \tag{18}$$

$$\begin{aligned} \exists\varphi,\ \exists\tilde{\varphi} : \{\varphi(\cdot - k)\}_k &\text{ is a Riesz basis in } V_0, \\ \{\tilde{\varphi}(\cdot - k)\}_k &\text{ is a Riesz basis in } \tilde{V}_0, \\ \langle \varphi(\cdot - k), \tilde{\varphi} \rangle &= \delta_k. \end{aligned} \tag{19}$$

We call φ, $\tilde{\varphi}$ the *scaling functions*. We will throughout the paper assume that these are real. With this construction we have real filters, h, $\tilde{h}$, such that

$$\begin{aligned} \varphi(x) &= \sqrt{2} \sum_k h_k \varphi(2x - k), \\ \tilde{\varphi}(x) &= \sqrt{2} \sum_k \tilde{h}_k \tilde{\varphi}(2x - k), \end{aligned} \tag{20}$$

and we define

$$\begin{aligned}\varphi_{jk}(x) &= 2^{j/2}\varphi(2^j x - k),\\ \tilde{\varphi}_{jk}(x) &= 2^{j/2}\tilde{\varphi}(2^j x - k).\end{aligned} \tag{21}$$

By defining the *mother wavelets*

$$\begin{aligned}\psi(x) &= \sqrt{2}\sum_k g_k \varphi(2x-k),\\ \tilde{\psi}(x) &= \sqrt{2}\sum_k \tilde{g}_k \tilde{\varphi}(2x-k),\end{aligned} \tag{22}$$

where,

$$\begin{aligned}g_k &= (-1)^k \tilde{h}_{1-k},\\ \tilde{g}_k &= (-1)^k h_{1-k},\end{aligned} \tag{23}$$

the wavelets

$$\begin{aligned}\psi_{jk}(x) &= 2^{j/2}\psi(2^j x - k),\\ \tilde{\psi}_{jk}(x) &= 2^{j/2}\tilde{\psi}(2^j x - k),\end{aligned} \tag{24}$$

and the spaces $W_j = \overline{\operatorname{span}\{\psi_{jk}\}_k}$, $\tilde{W}_j = \overline{\operatorname{span}\{\tilde{\psi}_{jk}\}_k}$, we get biorthogonal bases of L^2, i.e., all $\psi_{j,k}$ and all $\tilde{\psi}_{j,k}$ are uniformly linearly independent, and

$$f = \sum_{j,k}\langle f, \psi_{jk}\rangle \tilde{\psi}_{jk} = \sum_{j,k}\langle f, \tilde{\psi}_{jk}\rangle \psi_{jk}. \tag{25}$$

We will always use the second of these expansions.

Our numerical results will be for problems with periodic boundary conditions. One way to define wavelets on the torus, $\mathbb{T}$, is to use the following definitions

$$\varphi_{jk}^{\text{per}}(x) = \sum_l \varphi_{jk}(x+l), \tag{26}$$

$$\psi_{jk}^{\text{per}}(x) = \sum_l \psi_{jk}(x+l), \tag{27}$$

generating the spaces

$$V_j^{\text{per}} = \operatorname{span}(\{\varphi_{jk}^{\text{per}}(x)\}_{k=0,\ldots,2^j-1}), \tag{28}$$

$$W_j^{\text{per}} = \operatorname{span}(\{\psi_{jk}^{\text{per}}(x)\}_{k=0,\ldots,2^j-1}), \tag{29}$$

and similarly for $\tilde{\varphi}$, $\tilde{\psi}$.

The biorthogonality relation, (19), implies that the filters h, $\tilde{h}$ satisfy the relation

$$\sum_n h_n \tilde{h}_{n+2k} = \delta_k, \tag{30}$$

and the dilation equation, (20), that we have the condition

$$\sum_k h_k = \sum_k \tilde{h}_k = \sqrt{2}. \tag{31}$$

For a function, $f \in V_j$, with $s_k^j = \langle f, \tilde{\varphi}_{jk} \rangle$, the coefficients $d_k^{j-1} = \langle f, \tilde{\psi}_{j-1,k} \rangle$, $s_k^{j-1} = \langle f, \tilde{\varphi}_{j-1,k} \rangle$ can be found from (22),

$$\begin{aligned} s_k^{j-1} &= \sum_n \tilde{h}_{n-2k} s_n^j, \\ d_k^{j-1} &= \sum_n \tilde{g}_{n-2k} s_n^j. \end{aligned} \tag{32}$$

We can also go backwards,

$$s_k^j = \sum_n \left(h_{k-2n} s_n^{j-1} + g_{k-2n} d_n^{j-1} \right). \tag{33}$$

Written in filter notation this becomes $s^j = h \upharpoonleft\downharpoonright \check{\tilde{h}} s^j + g \upharpoonleft\downharpoonright \check{\tilde{g}} s^j$, which is the operation in Figure 1. The same operation can of course be made recursively leading to the pyramid algorithm shown in Figure 2, which is an $\mathcal{O}(N)$ algorithm, where N is the number of non-zero coefficients in s^j. Of course, when doing numerical calculations, we cannot deal with

$$\begin{tikzcd} s^n \arrow[r] \arrow[dr] & s^{n-1} \arrow[r] \arrow[dr] & s^{n-2} \arrow[r] \arrow[dr] & \cdots \arrow[r] & s^{n_0+1} \arrow[r] \arrow[dr] & s^{n_0} \\ & d^{n-1} & d^{n-2} & \cdots & d^{n_0+1} & d^{n_0} \end{tikzcd}$$

Figure 2: *The pyramid fast wavelet transform.*

all scales, j; we have to choose a finest scale, j_0. The Strang–Fix condition [15] implies that vanishing moments (zero-moments) for the wavelets guarantee a high order of approximation, i.e., if we define

$$\Sigma_n(\tilde{\psi}) = \int x^n \tilde{\psi}(x)\, dx, \tag{34}$$

then

$$\Sigma_n(\tilde{\psi}) = 0, \qquad n = 0, \ldots, N-1, \tag{35}$$

implies that

$$\|E_j f - f\|_2 = \mathcal{O}(2^{-jN}), \tag{36}$$

where E_j is the orthogonal projection operator of L^2 onto V_j.

Not all choices of h, g, $\tilde{h}$, $\tilde{g}$ satisfying (23), (30), (31), lead to dual wavelet bases. Extra conditions must be satisfied. Whereas for orthogonal filter banks, the worst thing that can happen is that the ψ_{jk}s constitute a tight frame of L^2 instead of an orthonormal basis, when using non-orthogonal filter banks, blow-up of constants can appear, leading to φ or $\tilde{\varphi} \notin L^2$. For further discussions on this subtle topic, see [4, 5, 6, 8, 22, 23]. On the other hand, from a filter point of view we can always choose filters h, g, $\tilde{h}$, $\tilde{g}$ without worrying about if they come from an MRA or not. We will see that such filters also will work well.

Definition 2.1 A *perfect reconstruction filter bank* is a quadruple of filters, $(h, g, \tilde{h}, \tilde{g})$ satisfying

$$\sum_k h_k = \sum_k \tilde{h}_k = \sqrt{2}, \tag{37}$$

$$\sum_n h_n \tilde{h}_{n+2k} = \delta_k, \tag{38}$$

$$\begin{aligned} g_k &= (-1)^k \tilde{h}_{1-k}, \\ \tilde{g}_k &= (-1)^k h_{1-k}. \end{aligned} \tag{39}$$

We will only work with real filters with a finite number of non-zero coefficients. If we define the nth discrete moment as

$$\sigma_n(h) = \sum_k k^n h_k, \tag{40}$$

the zero-moment conditions (35) carry over to discrete zero-moments on $\tilde{g}$.

$$\sigma_k(\tilde{g}) = 0, \qquad k = 0, \ldots, N-1. \tag{41}$$

The transform of a filter, s^n, is now defined in the same manner as in the continuous case,

$$\begin{aligned} \mathcal{F}^{n-n_0} s^n &= (d^{n-1}, d^{n-2}, \ldots, d^{n_0}, s^{n_0}) = (t^{n-1}, t^{n-2}, \ldots, t^{n_0-1}) = \tau, \\ d^{n-j} &= \tilde{G}\tilde{H}^{j-1} s^n, \\ s^{n_0} &= \tilde{H}^{n-n_0} s^n, \end{aligned} \tag{42}$$

and with a perfect reconstruction filter bank we get the inversion formula

$$s^n = (\mathcal{F}^{n-n_0})^{-1}\tau = \mathcal{F}^{n_0-n}\tau = \sum_{j=1}^{n-n_0} (H^*)^{j-1} G^* d^{n-j} + (H^*)^{n-n_0} s^{n_0}. \tag{43}$$

Here we have defined

$$\begin{aligned} \tilde{H} &= \tfrac{1}{\sqrt{2}} \downarrow \check{\tilde{h}}, \qquad & \tilde{G} &= \sqrt{2} \downarrow \check{\tilde{g}}, \\ H &= \sqrt{2} \downarrow \check{h}, \qquad & G &= \tfrac{1}{\sqrt{2}} \downarrow \check{g}. \end{aligned} \tag{44}$$

The factors, $\sqrt{2}$, $\frac{1}{\sqrt{2}}$, appearing in the formulae will be justified when we introduce the ϵ-truncated wavelet representation. They also come in handy for the filters we will use as the coefficients will become rational numbers (with binary denominators). The superscript, n, in s^n naturally denotes the scale in the approximation, and we will sample $s^n = [f]_{2^{-n}}$ in the subsequent sections. This transform, acting on point values will be denoted the *filter bank transform* in this paper, whereas we will call the MRA interpretation the *wavelet transform*. Periodization of the filters is done by treating them as circulant, with length 2^j on scale j. We will always assume that the filters in the perfect reconstruction filter bank have length smaller then the coarsest scale, length h, length $\tilde{h} < 2^{n_0}$.

Some examples of perfect reconstruction filter banks are shown in Table 1–Table 4. Here N is the number of vanishing moments for g, and $\tilde{N}$ the number of vanishing moments for $\tilde{g}$. Table 1 contains the Daubechies filters that lead to orthonormal wavelets [8] (so $\tilde{h} = h$), whereas Table 2 contains spline filters, h, and the duals of short support that satisfy the perfect reconstruction identity. For the third order filter we have also listed a dual with longer support that has an associated function, $\tilde{\varphi} \in L^2$. The two different filters will be compared in the next section. In Table 3 we have listed some interesting filters that also satisfy zero-moment conditions for h, $\tilde{h}$ (except for the zeroth moment, σ_0). They are the biorthogonal versions of coiflets [10], and they have fewer non-zero coefficients than these. They are interesting as inner products can be approximated with point values for them, and sparse multiplication will be of high order as we will see. It will be shown that it is only the number of zero-moments in $\tilde{h}$ that is important. This means that the filters in Table 4 will be 'optimal' as they only have zero-moments for these. For these filters, the $\tilde{H}$ and G^* steps will be trivial. The δ-filters of even order were originally introduced as interpolation refinement schemes [11]. They are connected to Daubechies filters by the equation $h^{\delta_{2N-1}} = h^{D_{2N}} * \check{h}^{D_{2N}}$ [9]. This connection was used in [3] do develop a collocation method for elliptic problems.

Name	N, $\tilde{N}$	$h, \tilde{h}$
D_4	2	$(\underline{0.48296291314}, 0.836516303, 0.2241438680, -0.129409522)$
D_6	3	$(\underline{0.33267055295}, 0.8068915093, 0.4598775021, -0.13501102001, -0.085441273, 0.035226291)$
D_8	4	$(\underline{0.2303778133}, 0.7148465705, 0.63088076793, -0.0279837694, -0.18703481171, 0.0308413818, 0.032883011, -0.010597401)$

Table 1: *Daubechies orthogonal filter banks ($\tilde{h} = h$).*

Name	N	$\tilde{N}$	h	$\tilde{h}$
β_0	1	1	$\frac{\sqrt{2}}{2}(\underline{1}, 1)$	$\frac{\sqrt{2}}{2}(\underline{1}, 1)$
β_1	2	1	$\frac{\sqrt{2}}{4}(\underline{1}, 2, 1)$	$\frac{\sqrt{2}}{4}(-1, \underline{2}, 3)$
β_2	3	1	$\frac{\sqrt{2}}{8}(\underline{1}, 3, 3, 1)$	$\frac{\sqrt{2}}{4}(\underline{-1}, 3, 3, -1)$
β_3	4	1	$\frac{\sqrt{2}}{16}(\underline{1}, 4, 6, 4, 1)$	$\frac{\sqrt{2}}{16}(3, \underline{-12}, 10, 20, -5)$
β_4	5	1	$\frac{\sqrt{2}}{32}(\underline{1}, 5, 10, 10, 5, 1)$	$\frac{\sqrt{2}}{16}(\underline{3}, -15, 20, 20, -15, 3)$
β_5	6	1	$\frac{\sqrt{2}}{64}(\underline{1}, 6, 15, 20, 15, 6, 1)$	$\frac{\sqrt{2}}{32}(-5, \underline{30}, -63, 28, 77, -42, 7)$
	3	3	$\frac{\sqrt{2}}{8}(\underline{1}, 3, 3, 1)$	$\frac{\sqrt{2}}{64}(3, -9, \underline{-7}, 45, 45, -7, -9, 3)$

Table 2: *Spline filters (h) with duals with short support.*

Name	N	$\tilde{N}$	h	$\tilde{h}$
γ_1	2	2	$\frac{\sqrt{2}}{4}(1, \underline{2}, 1)$	$\frac{\sqrt{2}}{8}(-1, 2, \underline{6}, 2, -1)$
γ_2	3	3	$\frac{\sqrt{2}}{16}(3, \underline{8}, 6, 0, -1)$	$\frac{\sqrt{2}}{128}(3, 0, -12, 24, \underline{82}, 48, -12, -8, 3)$
γ_3	4	4	$\frac{\sqrt{2}}{32}(-1, 0, 9, \underline{16}, 9, 0, -1)$	$\frac{\sqrt{2}}{512}(-1, 0, 18, -16, -63, 144, \underline{348}, 144, -63, -16, 18, 0, -1)$
γ_4	5	5	$\frac{\sqrt{2}}{256}(-5, 0, 60, \underline{128}, 90, 0, -20, 0, 3)$	$\frac{\sqrt{2}}{32768}(15, 0, -280, 0, 1380, -640, -3240, 7680, \underline{20634}, 11520, -3240, -2560, 1380, 384, -280, 0, 15)$

Table 3: *Filter banks with vanishing moments for h, $\tilde{h}$, g, $\tilde{g}$.*

Name	N	$\tilde{N}$	h	$\tilde{h}$
δ_1	2	0	$\frac{\sqrt{2}}{4}(1,\underline{2},1)$	$(\underline{\sqrt{2}})$
δ_2	3	0	$\frac{\sqrt{2}}{16}(3,\underline{8},6,0,-1)$	$(\underline{\sqrt{2}})$
δ_3	4	0	$\frac{\sqrt{2}}{32}(-1,0,9,\underline{16},9,0,-1)$	$(\underline{\sqrt{2}})$
δ_4	5	0	$\frac{\sqrt{2}}{256}(-5,0,60,\underline{128},90,0,-20,0,3)$	$(\underline{\sqrt{2}})$
δ_5	6	0	$\frac{\sqrt{2}}{512}(3,0,-25,0,150,\underline{256},150,0,-25,0,3)$	$(\underline{\sqrt{2}})$

Table 4: *Filter banks with vanishing moments for $\tilde{g}$, $\tilde{h}$.*

3 The filter bank method

The most frequently used approach when solving PDEs with wavelet methods is to use the continuous theory and approximate the solution with a Galerkin method or a collocation method. To approximate inner products one can use quadrature formulae proposed in [29]. The simplest formula of the same order as (36) will be a convolution with $\tilde{\bar{r}}$, where $\tilde{r} = 2^{-j/2}[\tilde{\varphi}]_1$. On the other hand one could take the point values of the sampled function and transform these. Furthermore, in the reconstruction, one could approximate the value of the function in the dyadic points by the coefficients in the wavelet expansion, instead of evaluating the exact value which would correspond to convolution with $r = 2^{j/2}[\varphi]_1$. The approximations would be

$$\begin{array}{lcl} \langle f, \tilde{\varphi}_{jk} \rangle & \approx & 2^{-j/2} f(2^{-j}k), \\ \sum_k c_k \varphi_{jk}(2^{-j}s) & \approx & 2^{j/2} c_s. \end{array} \tag{45}$$

The approximations are of first order, but as the filters involved make up a perfect reconstruction filter bank, they give an exact reconstruction of the sampled values. The approach is of course the same as the filter bank method. There is one case when (45) actually gives a high order approximation even from the continuous point of view, and that is when using scaling functions with vanishing moments (i.e., coiflets, or the biorthogonal wavelets in Table 3). This was used in [21] to get a high order method. The question is what happens with the signal when we start manipulating it, e.g., truncating coefficients. It turns out, as we will see, that the filter bank approach performs better in this case too.

Another advantage with using filter banks instead of the underlying wavelets is that we can use filters that do not generate wavelets (for example $\tilde{h} = \frac{\sqrt{2}}{4}(\underline{-1},3,3,-1)$). This gives us the opportunity to use shorter filters. We now introduce the truncated transformed signal.

3.1 Truncation of signals

Suppose we transform a signal, s^n which can be thought of as a sampling of a function $s^n = [f]_{2^{-n}}$, $k = n - n_0 - 1$ steps, giving $\tau = \mathcal{F}^k s^n$. We define the set of retained coefficients,

$$I_n^\epsilon = \{(j,k) : n_0 \leq j < n, |t_k^j| > \epsilon\}, \tag{46}$$

and the number of significant coefficients,

$$N_{\mathrm{s}} = |I_n^\epsilon|. \tag{47}$$

The truncated representation is defined as

$$\tau_\epsilon = \mathcal{T}_\epsilon \tau = (t_\epsilon^{n-1}, \ldots, t_\epsilon^{n_0}), \tag{48}$$

where

$$\begin{aligned}(t^j_\epsilon)(l) &= t^j(l),\ (j,l)\in I^\epsilon_n,\\ (t^j_\epsilon)(l) &= 0,\ (j,l)\notin I^\epsilon_n.\end{aligned} \tag{49}$$

We compare the error in the filter bank approximation,

$$e_{\mathrm{f}} = \mathcal{F}^{-k}\mathcal{T}_\epsilon\mathcal{F}^k s^n - s^n, \tag{50}$$

with the error in the wavelet Galerkin approximation,

$$e_{\mathrm{w}} = r\mathcal{F}^{-k}\mathcal{T}_\epsilon\mathcal{F}^k\tilde{\tilde{r}}s^n - s^n. \tag{51}$$

We have a choice in what wavelets to use. Spline filters have shorter support than Daubechies wavelets with the same number of vanishing moments. Furthermore, it has been noted [28] that spline wavelets have better approximation properties than Daubechies wavelets with the same number of vanishing moments. This has also been the case for the examples in this paper. We therefore work with spline filter banks (and further on with δ filter banks). We choose the filters $h = \frac{\sqrt{2}}{8}(\underline{1},3,3,1)$, $\tilde{h} = \frac{\sqrt{2}}{64}(3,-9,\underline{-7},45,45,-7,-9,3)$, and a smooth function, with a 'bump,' $f = \sin(2\pi x) + e^{-20000(x-0.5)^2}$, shown in Figure 3. We truncate with $\epsilon = 10^{-3}$. We have the finest level, $n = 9$ with 512 points, and let $k = 5$. In Figure 4 we see the error when using the wavelet Galerkin method with quadratures to approximate inner products. We have also plotted where the retained coefficients are. In Figure 5 we see the error and coefficients when using the filter bank method. The error is about 30 times smaller (in maximum norm) with about the same number of coefficients retained. As we do not use any underlying scaling function in the filter bank method, we can also choose a dual filter that is not associated to a scaling function in L^2, $\tilde{h} = \frac{\sqrt{2}}{4}(\underline{-1},3,3,-1)$. The result when using this filter is shown in Figure 6. We see that the error is about the same as when using the longer dual filter. We also see that only a small portion of the total number of coefficients are needed. The behavior of the d^j_k coefficients for the wavelet method is for example studied in [25], where it is shown that they decay like $2^{-j(s+1/2)}\|f\|_{\dot{C}^s}$ for functions in C^s. As we are working with wavelets with compact support, this means that functions that are smooth nearly everywhere ought to have sparse representations in wavelet bases. We shall prove that this is also the case when using the filter bank method.

To be more precise, we introduce the function spaces

$$\begin{aligned}\mathcal{L}^{s,m}(\mathbb{T}) = \{f \in \mathbb{C}^{\mathbb{T}} : \exists N, \exists C, \forall y, \exists a_1,\ldots,a_N, \exists b_1,\ldots,b_N :&\\ \|f\|_{\dot{C}^s(\mathbb{T}\setminus\cup_{i=1}^N(a_i,b_i))} \le y, \sum_{i=1}^N |b_i - a_i| \le \frac{C}{y^m}\}.&\end{aligned} \tag{52}$$

The functions in these spaces are allowed to have a finite number of discontinuities, but the semi norms $\|f\|_{\dot{C}^s}$ are only allowed to be large on a finite number of intervals of decreasing length, so the spaces are slightly larger than the C^s spaces. For example the step function, $\Theta(x-0.5)$ belongs to $\mathcal{L}^{s,m}$ for all s and m, and so does $|x-0.5|^{-\alpha}$ for all $\alpha > 0$, $m \le 1/(s+\alpha)$ (when periodized). On the contrary, let $f(x) = |x-0.5|\log|x-0.5|$, $x \ne 0.5$, $f(0.5) = 0$. If we enumerate the rational numbers, $q \in \mathbb{Q}$ and define $g_s = \sum_{j=0}^\infty 2^{-j} f(\cdot - q_j)^s$, $s \in (0,1]$, then g_s is in $C^{s-\epsilon}$ for all $\epsilon > 0$, but not in $\mathcal{L}^{s,m}$ for any m.

We will now show that for functions in $\mathcal{L}^{s,m}$, the filter bank method will give a compact representation with small error. The theorems in this paper rely on the basic fact that for polynomials of degree less than the number of vanishing moments of $\tilde{g}$, $P \in \mathcal{P}^{N-1}$, we have $\tilde{G}\tilde{H}^{j-1}P = 0$, and therefore all information is kept in $\tilde{H}^jP$.

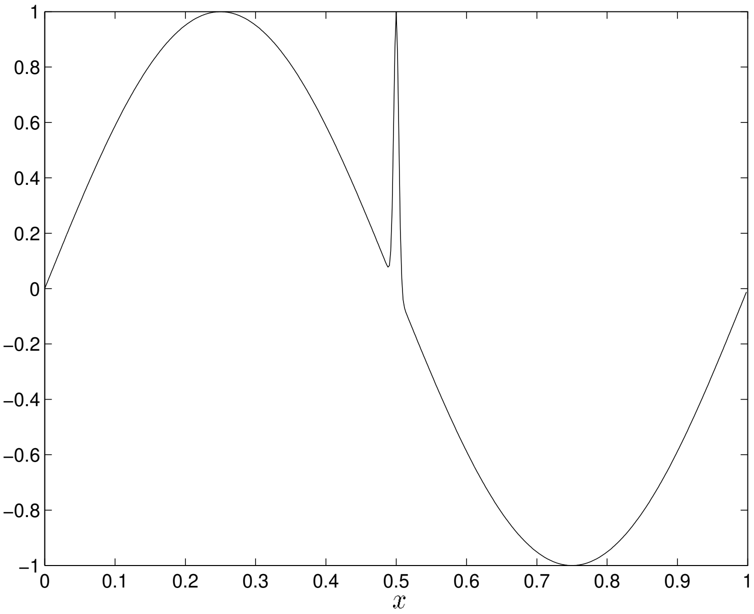


Figure 3: *Test function,* $f = \sin(2\pi x) + e^{-20000(x-0.5)^2}$.

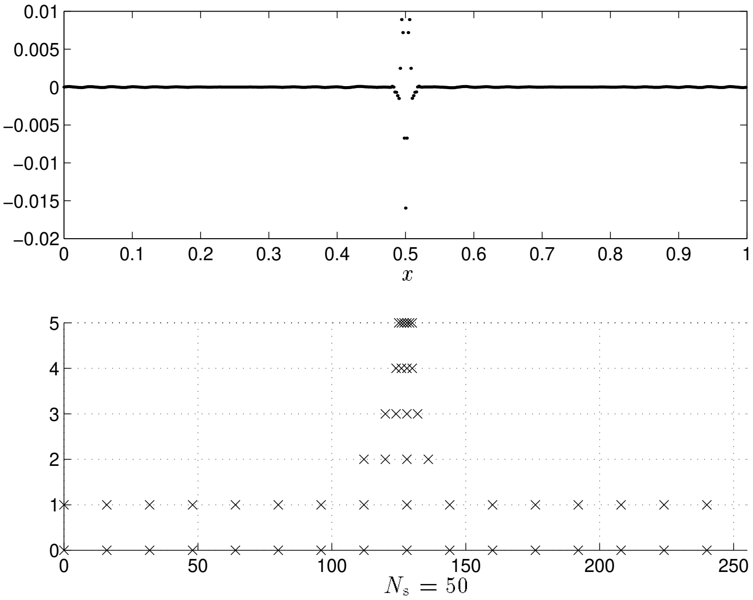


Figure 4: *Error,* e_w*, of truncated function with wavelet method,* $\epsilon = 10^{-3}$.

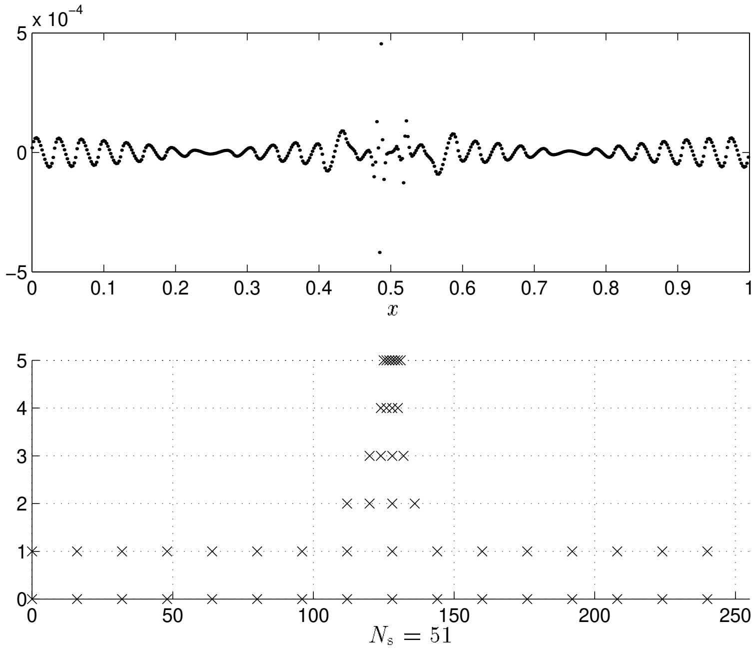


Figure 5: *Error, e_f, of truncated function with filter bank method, $\epsilon = 10^{-3}$.*

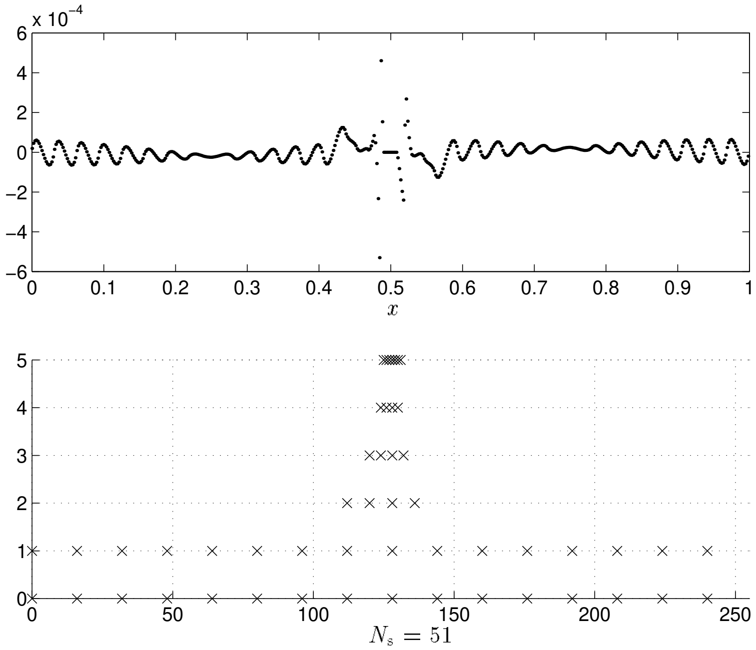


Figure 6: *Error, e_f, of truncated function with filter bank method, and dual filter with short support, $\epsilon = 10^{-3}$.*

Theorem 3.1 *Suppose $f \in \mathcal{L}^{s,m}(\mathbb{T})$, is sampled at scale n, wavelet transformed k steps with a perfect reconstruction filter bank, where $\tilde{g}$ has $N > s$ zero-moments, and f is truncated with $\mathcal{T}_\epsilon$. Furthermore, suppose that $\|\tilde{H}^p\|_\infty \le C2^{\zeta_1 p}$, $\|(H^*)^p\|_\infty \le C2^{\zeta_2 p}$. Then*

$$\|e_{\mathrm{f}}\|_\infty \le C_0(2^{\zeta_2 k} + k)\epsilon, \tag{53}$$

where

$$e_{\mathrm{f}} = \mathcal{F}^{-k}\mathcal{T}_\epsilon\mathcal{F}^k[f]_{2^{-n}} - [f]_{2^{-n}}, \tag{54}$$

and C_0 is independent of n, k. Furthermore, the number of non-zero coefficients in τ_ϵ, N_{s}, satisfies

$$N_{\mathrm{s}} \le C_1 k + 2^{n-k-1} + C_2 q, \tag{55}$$

where,

$$q = \begin{cases} 2^{\frac{\zeta_1}{s+\zeta_1}n}\epsilon^{-\frac{1}{s+\zeta_1}}, & m > \frac{1}{s+\zeta_1}, \\ k2^{\frac{\zeta_1}{s+\zeta_1}n}\epsilon^{-\frac{1}{s+\zeta_1}}, & m = \frac{1}{s+\zeta_1}, \\ 2^{(1-ms)n}\epsilon^{-m}, & m < \frac{1}{s+\zeta_1}, \end{cases} \tag{56}$$

and C_1, C_2 are independent of n, k.

Proof. For the error we have

$$\begin{aligned} \|e_{\mathrm{f}}\|_\infty &= \Big\|\sum_{j=0}^{k-1}(H^*)^j G^*(t^{n-j-1} - t_\epsilon^{n-j-1}) + (H^*)^k(t^{n-k-1} - t_\epsilon^{n-k-1})\Big\|_\infty \\ &\le (1 + \|G^*\|_\infty)\sum_{j=0}^{k}\|(H^*)^j\|_\infty\|(t^{n-j-1} - t_\epsilon^{n-j-1})\|_\infty \\ &\le \sum_{j=0}^{k} C2^{\zeta_2 j}\epsilon \le C_0(2^{\zeta_2 k} + k)\epsilon. \end{aligned} \tag{57}$$

For the estimate of N_{s}, we need to know what coefficients in s^n have influence on $(\tilde{H}^j s^n)(l)$. We call this the *influence interval*, $\mathrm{Infl}(\tilde{H}^j, l)$. The coefficients that have influence on $(\tilde{H}s^n)(l)$ are $[2l - M(\tilde{h}), 2l - m(\tilde{h})]_{\mathbb{Z}}$ so we get

$$\begin{aligned} \mathrm{Infl}(\tilde{G}\tilde{H}^j, l) &= [\underline{I}, \overline{I}]_{\mathbb{Z}} \\ &= [2^{j+1}l - (2^{j+1} - 2)M(\tilde{h}) - M(\tilde{g}), 2^{j+1}l - (2^{j+1} - 2)m(\tilde{h}) - m(\tilde{g})]_{\mathbb{Z}}. \end{aligned}$$

As we are working on $\mathbb{T}$ these coefficients should be modulo unity, but it is only the length of the interval that will be interesting so we identify the functions in $\mathcal{L}^{s,m}(\mathbb{T})$ with 1-periodic functions, remembering that the wavelet transform only will use coefficients on one of the periods. If we suppose that $2^{-n}[\underline{I}, \overline{I}]$ does not overlap the R points where f may have a discontinuity, we can make a generalized Taylor-expansion of f on $2^{-n}[\underline{I}, \overline{I}]$. We put $y = 2^{j+1-n}l$, and get

$$\begin{aligned} f(y+x) = f(y) + f^{(1)}(y)x &+ f^{(2)}(y)\frac{x^2}{2} + f^{(3)}(y)\frac{x^3}{6} + \cdots \\ &+ r_y(x)\frac{x^s}{s(s-1)\cdots(s-\lfloor s\rfloor+1)}, \end{aligned} \tag{58}$$

where $x \in 2^{-n}[\underline{I}, \overline{I}] - y$, and $\|r_y\|_\infty \le \|f\|_{\dot{C}^s(2^{-n}[\underline{I}, \overline{I}])}$. However, as the set of polynomials of degree $\lfloor s\rfloor$, $\mathcal{P}^{\lfloor s\rfloor}$ is closed under dilation and translation, we know that for a polynomial, P, of degree not more than $\lfloor s \rfloor$, $\tilde{H}^j[P(x)]_h = [Q(x)]_{2^h}$ where Q is also a polynomial of degree $\lfloor s \rfloor$. Therefore, the polynomial part of (58), P, will be mapped to zero, $(\tilde{G}\tilde{H}^j[P]_{2^{-n}})(l) = 0$, as the moment conditions on $\tilde{g}$ makes it map polynomials of degree $N-1$ or less to zero. For the last term in (58) we have

$$\begin{aligned}
\left|d_l^{n-j-1}\right| &= |(\tilde{G}\tilde{H}^j[f]_{2^{-n}})(l)| = \left|\left(\tilde{G}\tilde{H}^j\left[r_y(x)\frac{x^s}{s(s-1)\cdots(s-\lfloor s \rfloor+1)}\right]_{2^{-n}}\right)(l)\right| \\
&\le \frac{1}{s(s-1)\cdots(s-\lfloor s \rfloor+1)}\|\tilde{G}\|_\infty\|\tilde{H}^j\|_\infty\|r_y(x)\chi_{2^{-n}[\underline{I},\overline{I}]}\|_\infty\|x^s\chi_{2^{-n}[\underline{I},\overline{I}]}\|_\infty \\
&\le c_0 2^{(j-n)s}\|\tilde{H}^j\|_\infty\|f\|_{\dot{C}^s(2^{-n}[\underline{I},\overline{I}])} \le c_1 2^{(j-n)s+\zeta_1 j}\|f\|_{\dot{C}^s(2^{-n}[\underline{I},\overline{I}])},
\end{aligned} \tag{59}$$

where we have used $2^{-n}(\overline{I}-\underline{I}) \le c_2 2^{j-n}$, and $\|r_y\|_\infty \le \|f\|_{\dot{C}^s(2^{-n}[\underline{I},\overline{I}])}$.

We have two bounds on the number of coefficients on each level, N_s^{n-j}, that have $|d_k^{n-j}| \ge \epsilon$. The first is trivial,

$$N_s^{n-j} \le 2^{n-j}. \tag{60}$$

For the second bound, we use (59) to see that

$$|d_l^{n-j-1}| \ge \epsilon \Rightarrow \|f\|_{\dot{C}^s(2^{-n}[\underline{I},\overline{I}])} \ge c_1^{-1}\epsilon 2^{(n-j)s-\zeta_1 j}.$$

However, from the definition of $\mathcal{L}^{s,m}$, the regions where this can be satisfied are at most R intervals of total length $c_3\epsilon^{-m}2^{-m(n-j)s+\zeta_1 mj}$, so the total number of coefficients that have an influence set that overlaps these regions are at most

$$N_s^{n-j-1} \le \sum_{i=1}^{R}\frac{c_4 2^{1+j-n}+(b_i-a_i)}{2^{1+j-n}} = Rc_4 + c_3 2^{(n-j)(1-ms)+\zeta_1 mj-1}\epsilon^{-m}. \tag{61}$$

We first assume that $m > \frac{1}{s+\zeta_1}$ so that the exponent in (61) is less than zero when summing over j. The break-even between the two bounds lies near $j = \frac{\log_2\epsilon+ns}{s+\zeta_1}$, and we divide the sum into two parts, where only the largest terms are interesting as both sums form decreasing geometric series. The total number of coefficients therefore satisfies

$$\begin{aligned}
\sum_{j=1}^{k} N_s^{n-j} &\le Rc_4 k + \frac{2^n}{2}\sum_{j=0}^{k-1}2^{-j}\min\{1, c_3 2^{-ms(n-j)+\zeta_1 mj}\epsilon^{-m}\} \\
&\le Rc_4 k + c_5 2^n\epsilon^{-m}2^{-msn+(ms+m\zeta_1-1)(\log_2\epsilon+ns)/(s+\zeta_1)} \\
&\qquad + c_6 2^n 2^{-(\log_2\epsilon+ns)/(s+\zeta_1)} = C_1 k + C_2 2^{\frac{\zeta_1}{s+\zeta_1}n}\epsilon^{-\frac{1}{s+\zeta_1}}.
\end{aligned} \tag{62}$$

We now assume that $m \le \frac{1}{s+\zeta_1}$. In this case both (60) and (61) are increasing when j decreases. Therefore, there is no gain in summing over both series, and we estimate the sum with

$$\sum_{j=0}^{k-1} Rc_4 + c_3 2^{(n-j)(1-ms)+\zeta_1 mj-1}\epsilon^{-m},$$

which, for $m < \frac{1}{s+\zeta_1}$ is less than or equal to $C_1 k + C_2 2^{(1-ms)n}\epsilon^{-m}$ and, for $m = \frac{1}{s+\zeta_1}$, to $C_1 k + C_2 k 2^{(1-ms)n}\epsilon^{-m}$.

Finally, the number of coefficients in s^{n-j} are bounded by 2^{n-j}, leading to the total bound on the coefficients.

$\square$

We see here that it is not important for g to have vanishing moments. The only thing important for the dual filter, $\tilde{h}$ is to satisfy the perfect reconstruction identity, (38), and for ζ_1 to be small. The following theorem gives bounds on ζ_1.

Theorem 3.2 *Let $(h, g, \tilde{h}, \tilde{g})$ be a perfect reconstruction filter bank and $\tilde{L} = \text{length}\, \tilde{h}$. Define*

$$\tilde{M} = \frac{1}{2}\tilde{h} * \check{\tilde{h}} = (\tilde{m}_{-\tilde{L}+1}, \ldots, \underline{\tilde{m}_0}, \ldots, \tilde{m}_{\tilde{L}-1}), \tag{63}$$

and the $(2\tilde{L}-1) \times (2\tilde{L}-1)$ matrix

$$(\tilde{\mathbf{P}})_{j,k} = \begin{cases} 2\tilde{m}_{2j-k-\tilde{L}}, & |2j-k-\tilde{L}| \; < \; \tilde{L}, \\ 0, & |2j-k-\tilde{L}| \; \geq \; \tilde{L}. \end{cases}$$

Then,

$$\|\tilde{H}^p\|_\infty \leq C\rho(\tilde{\mathbf{P}})^{p/2}, \tag{64}$$

where $\rho(\tilde{\mathbf{P}})$ is the spectral radius of $\tilde{\mathbf{P}}$, and C is independent of p.

Proof. By $H|_{a,b}$ we mean the restriction of H to $[a,b]_{\mathbb{Z}}$,
$H|_{a,b}u \mapsto ((Hu)(a), \ldots, (Hu)(b))$. We have the following norm relations,

$$\|\tilde{H}^p\|_\infty = \|\tilde{H}^p|_{0,0}\|_\infty \leq C2^{p/2}\|\tilde{H}^p|_{0,0}\|_2 \leq C2^{p/2}\|\tilde{H}^p\|_2.$$

But

$$\widehat{\tilde{H}f} = \frac{1}{\sqrt{2}}\downarrow \widehat{\check{\tilde{h}}f} = \frac{1}{2\sqrt{2}}\left(\hat{\check{\tilde{h}}}(-\omega/2)\hat{f}(\omega/2) + \hat{\check{\tilde{h}}}(-\omega/2+\pi)\hat{f}(\omega/2+\pi)\right),$$

leading to

$$\widehat{\tilde{H}^p f} = \frac{1}{2^p}\sum_{k=0}^{2^p-1}\left(\prod_{j=1}^{p}\frac{1}{\sqrt{2}}\hat{\check{\tilde{h}}}(2^{-j}(-\omega+2\pi k))\right)\hat{f}(2^{-p}(\omega+2\pi k)).$$

Parseval's identity gives us

$$\|\tilde{H}^p\|_2 = \sup_{\hat{f}\in L^2(0,2\pi)} \frac{\left\|\frac{1}{2^p}\sum_{k=0}^{2^p-1}\left(\prod_{j=1}^{p}\frac{1}{\sqrt{2}}\hat{\check{\tilde{h}}}(2^{-j}(-\omega+2\pi k))\right)\hat{f}(2^{-p}(\omega+2\pi k))\right\|_2}{\|\hat{f}(\omega)\|_2},$$

and by using, $|\hat{\check{\tilde{h}}}(\omega)|^2/2 = \hat{\tilde{M}}(\omega)$, Cauchy–Schwarz' and Hölder's inequalities we get

$$\|\tilde{H}^p\|_2^2 \leq \frac{1}{2^p}\sup_{\omega\in[0,2\pi)}\sum_{k=0}^{2^p-1}\prod_{j=1}^{p}\hat{\tilde{M}}(2^{-j}(-\omega+2\pi k)). \tag{65}$$

If we define the operator

$$(P_{\hat{\tilde{M}}}f)(\xi) = \frac{1}{2}(\hat{\tilde{M}}(-\xi/2)f(\xi/2) + \hat{\tilde{M}}(-\xi/2+\pi)f(\xi/2+\pi)), \tag{66}$$

acting on 2π-periodic functions, (65) can be written

$$\|\tilde{H}^p\|_2^2 \leq \|P_{\hat{\tilde{M}}}^p 1\|_\infty.$$

For trigonometric polynomials of the form $Q(\omega) = \sum_{k=-\tilde{L}+1}^{\tilde{L}-1} a_k e^{ik\xi}$, we have $(P_{\hat{\tilde{M}}} Q)(\omega) = \sum_{k=-\tilde{L}+1}^{\tilde{L}-1} b_k e^{ik\xi}$, where $\mathbf{b} = \tilde{\mathbf{P}}\mathbf{a}/2$, and $(\mathbf{a})_j = a_{j-\tilde{L}}$, $(\mathbf{b})_j = b_{j-\tilde{L}}$, so

$$\|P_{\hat{\tilde{M}}}^p 1\|_\infty \leq c\|P_{\hat{\tilde{M}}}^p 1\|_2 \leq c_2 \frac{\rho(\tilde{\mathbf{P}})^p}{2^p},$$

altogether leading to

$$\|H^p\|_\infty \leq C\rho(\tilde{\mathbf{P}})^{p/2}.$$

□

We remark that a necessary condition on $\tilde{\mathbf{P}}$ for $\tilde{\varphi}$ to generate a Riesz basis of L^2 is that $\rho(\tilde{\mathbf{P}}) = 1$ (see e.g. [5]), so for such choices of $\tilde{h}$ we have $\zeta_2 = 0$. For bounds on $\|(H^*)^p\|_\infty$ we have the following result

Theorem 3.3 *Let $(h, g, \tilde{h}, \tilde{g})$ be a perfect reconstruction filter bank, such that there is a φ (with compact support) in an MRA associated with h (but not necessarily with $\tilde{h}$), and $\varphi \in L^\infty$. Then*

$$\|(H^*)^p\|_\infty \leq C, \tag{67}$$

where C is independent of p.

Proof. As φ generates a Riesz basis of L^2 we know from [25] that we, by defining the dual basis function,

$$\hat{\bar{\varphi}}(\omega) = \frac{\hat{\varphi}(\omega)}{\sum_k |\hat{\varphi}(\omega + 2\pi k)|^2},$$

get $\langle 2^{p/2}\varphi(2^p \cdot -k), 2^{p/2}\bar{\varphi}(2^p \cdot)\rangle = \delta_k$, and $\bar{\varphi} \in L^1$. For an arbitrary c, we wish to estimate $\|(H^*)^p c\|_\infty$, so we define $f = \sum_k c_k \varphi_{0,k}$. From Hölder's inequality we get

$$|\langle f, 2^{p/2}\bar{\varphi}(2^p \cdot -m)\rangle| \leq \|f\|_\infty \|2^{p/2}\bar{\varphi}(2^p \cdot -m)\|_1 \leq 2^{-p/2}\|\varphi\|_\infty \|\bar{\varphi}\|_1 \|c\|_\infty(\text{length } \tilde{h} - 1).$$

However, from (20), (44) we have $\langle f, 2^{p/2}\bar{\varphi}(2^p \cdot -m)\rangle = 2^{-p/2}((H^*)^p c)(m)$, so

$$\|(H^*)^p c\|_\infty \leq (\text{length } \tilde{h} - 1)\|\varphi\|_\infty \|\bar{\varphi}\|_1 \|c\|_\infty.$$

□

As we typically will require many vanishing moments for $\tilde{g}$ to get a high approximation order, we usually will have $\varphi \in L^2 \cap L^\infty$, so the conditions of Theorem 3.3 will be satisfied. The strength of the filter bank method lies in choosing $\tilde{h}$ not corresponding to any φ.

There are of course other bounds. From Young's inequality we get

$$\|\tilde{H}^p\|_\infty \leq \left(\frac{\|\tilde{h}\|_1}{\sqrt{2}}\right)^p,$$

$$\|(H^*)^p\|_\infty \leq (\sqrt{2}\|h\|_1)^p,$$

(where we, on the right-hand side, interpret h, $\tilde{h}$ as filters, *not* as operators), but the bound in Theorem 3.2 seems to be close to optimal. The filter banks we have listed in the tables have h corresponding to $\varphi \in L^2 \cap L^\infty$, and thus $\zeta_2 = 0$. The filter banks in Table 1 and Table 3 also have $\tilde{\varphi} \in L^2$, so for these $\zeta_1 = 0$. The filter banks in Table 2 and Table 4 do *not* have $\tilde{h}$ corresponding to $\tilde{\varphi} \in L^2$. In Table 5 we list the bounds found in Theorem 3.2 for these, and the exponents involved in Theorem 3.1 (where N denotes the number of vanishing moments for $\tilde{g}$, which will govern the decay for $\mathcal{L}^{s,m}$ when $s > N$). We choose the bound for $m > \frac{1}{s+\zeta_1}$ as this will be 'right' when f has no worse singularities than discontinuities.

Name	ζ_1	$\frac{\zeta_1}{\zeta_1+N}$	$\frac{1}{\zeta_1+N}$
β_0	0	0	1
β_1	0.16096	0.07448	0.46275
β_2	0.55923	0.15712	0.28095
β_3	1.28630	0.24332	0.18916
β_4	1.82486	0.26738	0.14652
β_5	2.59075	0.30157	0.11640
δ_1	$\frac{1}{2}$	$\frac{1}{5}$	$\frac{2}{5}$
δ_2	$\frac{1}{2}$	$\frac{1}{7}$	$\frac{2}{7}$
δ_3	$\frac{1}{2}$	$\frac{1}{9}$	$\frac{2}{9}$
δ_4	$\frac{1}{2}$	$\frac{1}{11}$	$\frac{2}{11}$
δ_5	$\frac{1}{2}$	$\frac{1}{13}$	$\frac{2}{13}$

Table 5: *Bounds for β (spline), and δ (zero-moment) filter banks.*

For the filter $\tilde{h} = \frac{\sqrt{2}}{4}(\underline{-1}, 3, 3, -1)$, we get $\zeta_1 \leq 0.559$ and, as $h = \frac{\sqrt{2}}{4}(\underline{1}, 3, 3, 1)$ gives a $\tilde{g}$ with 3 vanishing moments, the β_2 filter bank can be used without problems in numerical calculations. We also note that from [5, 14] it follows that for $\zeta_1 > 0$ we have

$$\tilde{\varphi} \in H^{-\zeta_1-\epsilon}, \qquad \epsilon > 0, \tag{68}$$

so ζ_1 gives a measure of how 'far' from L^2 the corresponding wavelet (now defined in a distributional sense) is. Another consequence of Theorem 3.1 is that we do not have to worry about Gibbs' phenomena, when using the filter bank method as in the wavelet Galerkin case. As an example we take the function in Figure 7, $f(x) = \sin(2\pi x) - \Theta(x-1/2) + 1/2$, transform it with the wavelet Galerkin method and compare this to the result when using the filter bank method. The result is shown in Figure 8. With the wavelet Galerkin method, the error will not decrease in max-norm as ϵ decreases, or n increases. In contrast, when using the filter bank method the error decreases when ϵ decreases in accordance with Theorem 3.1.

We also get good results, with a 'nasty' function like $f(x) = \sin(2\pi x) + \frac{1}{1000(x-0.5)^2}$, $x \neq 0.5$, $f(0.5) = 0$, the absolute value of which is plotted in $\log_{10}$ scale in Figure 9. We choose $n = 11$, $k = 7$, $\epsilon = 5 \times 10^{-4}$. The error is shown in Figure 10. We see that it is small, $\|e_f\|_\infty = 3.3 \times 10^{-3}$, with a compression factor of $2048/94 = 21.8$.

All the calculations in this and subsequent sections were performed in double precision. The error estimate in Theorem 3.1 gives us a hint that the maximum norm is the 'right' norm for the error. *We will therefore measure all errors in the maximum norm.*

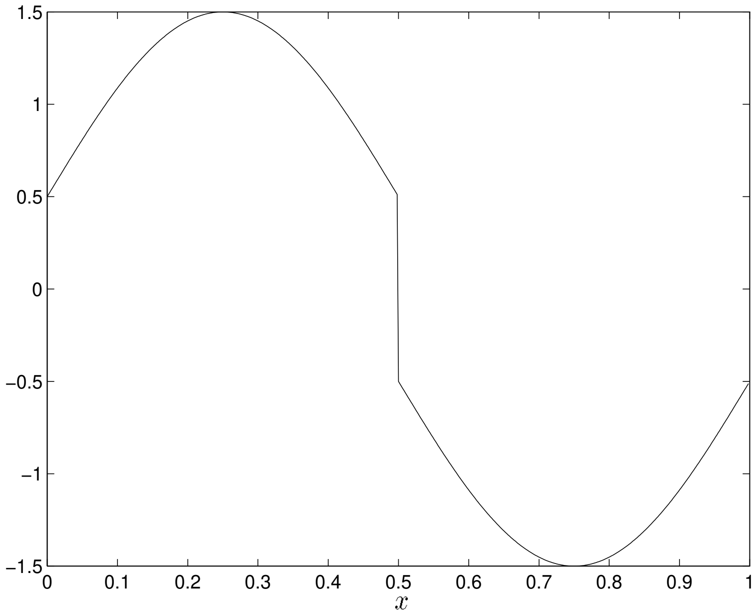


Figure 7: *Test function,* $f(x) = \sin(2\pi x) - \Theta(x - 1/2) + 1/2$.

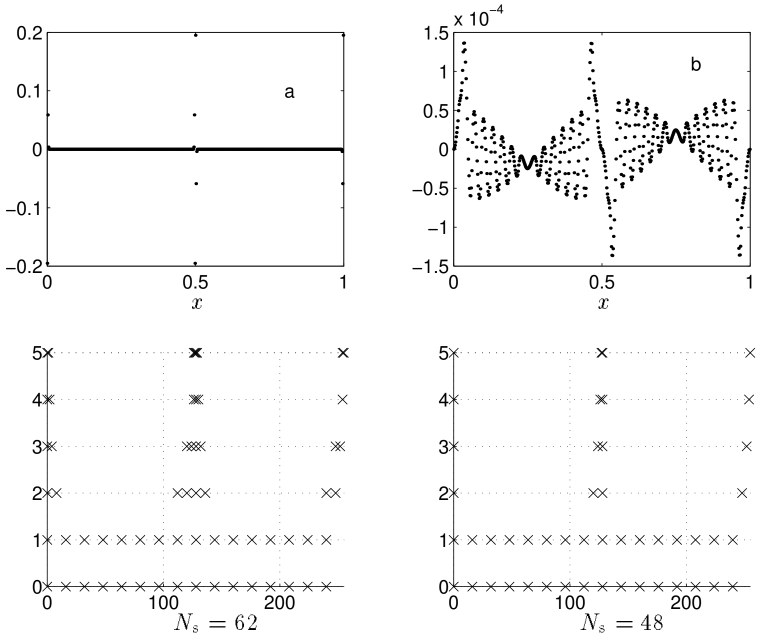


Figure 8: *Error of truncated function with wavelet method (a), and filter bank method (b),* $\epsilon = 10^{-3}$.

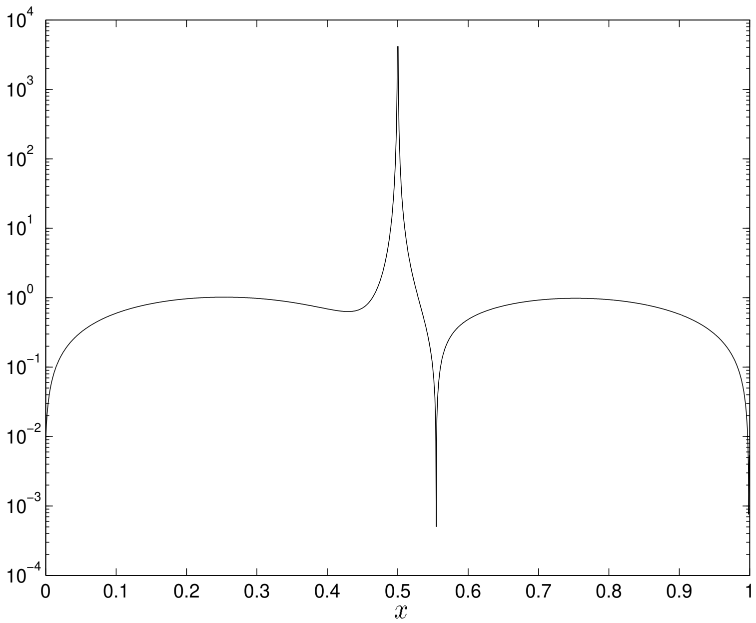


Figure 9: *Test function,* $f(x) = \sin(2\pi x) + \frac{1}{1000(x-0.5)^2}$, *absolute value plotted in* $\log_{10}$ *scale.*

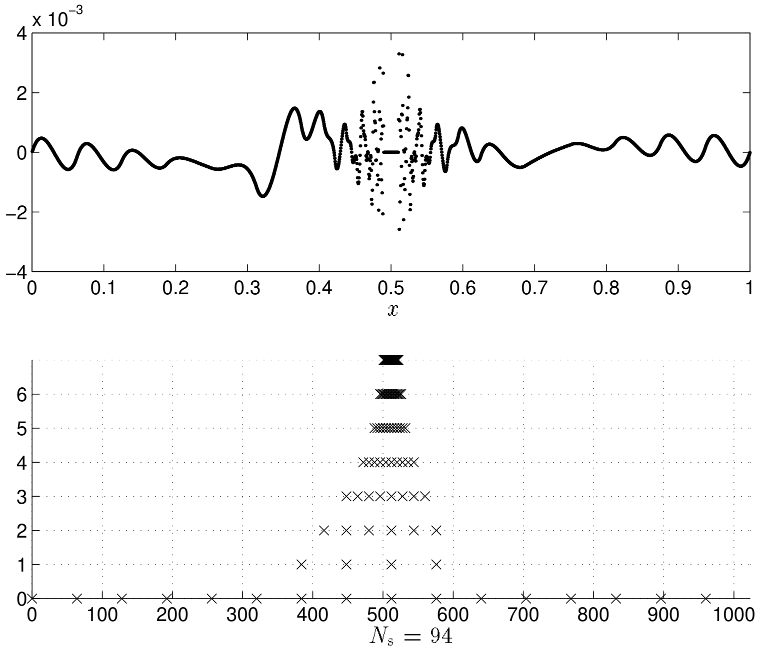


Figure 10: *Error of truncated function with filter bank method,* $\epsilon = 10^{-3}$.

3.2 Differentiation of truncated functions

We are now going to introduce a way to differentiate a truncated representation, τ_ϵ. As we have point values, it seems logical to use a finite difference approximation. In the language of filters, a finite difference operator is simply a convolution operator (where the coefficients depend on the grid size, h, used in the sampling), with a finite number of non-zero coefficients.

Definition 3.1 A convolution operator, represented by a filter, $p(h)$, is said to be an Nth order approximation of $\frac{d^s}{dx^s}$ if $p(h)[x^k]_h = [\frac{d^s(x^k)}{dx^k}]_h$, $k = 0, \ldots, N+s-1$.

An Nth order approximation of $\frac{d^s}{dx^s}$ will be of the form $p(h) = \frac{1}{h^s}(q_m, \ldots, q_M)$ and the order conditions that will be satisfied are

$$\frac{1}{s!}\sum_{j=m}^{M} q_{-j}j^k = \delta_{k-s}, \qquad k = 0, \ldots, N+s-1. \tag{69}$$

When differentiating we could, of course, transform the truncated representation back to the finest level and apply the finite difference operator to this representation, but this would not make any use of the sparse representation we have. Instead we would like to apply the operator on a coarse scale where this can be used. The following theorem justifies such an approach.

Theorem 3.4 *Suppose $d(h)$ is an Nth order finite difference approximation of $\frac{d^s}{dx^s}$. If $(h, g, \tilde{h}, \tilde{g})$ is a perfect reconstruction filter bank, and $\tilde{g}$ has N_2 zero-moments, then*

$$(H^*)^j d(2^j h)\tilde{H}^j[x^k]_h = \left[\frac{d^s(x^k)}{dx^s}\right]_h, \qquad k = 0, \ldots, \min\{N, N_2\} + s - 1. \tag{70}$$

Proof. We know that for any $P \in \mathcal{P}^k$, $\tilde{H}[P(x)]_h = [Q(x)]_{2h}$, where $Q \in \mathcal{P}^k$. The theorem now follows from

$$\tilde{H}\left[\frac{d^sP}{dx^s}\right]_h = \left[\frac{d^sQ}{dx^s}\right]_{2h}. \tag{71}$$

This is seen for $j = 1$, as $d(2h)$ differentiates every polynomial in $\mathcal{P}^{N+s-1}$ exactly, and every polynomial in $\mathcal{P}^{N_2-1}$, is mapped to 0 by $\tilde{G}$, so

$$\left[\frac{d^sP}{dx^s}\right]_h = (H^*\tilde{H} + G^*\tilde{G})\left[\frac{d^sP}{dx^s}\right]_h = H^*\tilde{H}\left[\frac{d^sP}{dx^s}\right]_h = H^*\left[\frac{d^sQ}{dx^s}\right]_{2h} = H^*d(2h)[Q]_{2h},$$
$$P \in \mathcal{P}^{\min\{N,N_2\}+s-1}.$$

By induction this is true for all j. It remains to show (71) which is a consequence of differentiation commuting with convolution,

$$\left[\frac{d^sQ}{dx^s}\right]_{2h} = \left[\frac{d^s}{dx^s}\frac{1}{\sqrt{2}}\tilde{h}P\right]_{2h} = \left[\frac{1}{\sqrt{2}}\downarrow\tilde{h}\frac{d^s}{dx^s}P\right]_h = \tilde{H}\left[\frac{d^sP}{dx^s}\right]_h,$$

where we in the intermediate steps used semi-continuous convolution, and continuous downsampling.

$\square$

This theorem leads us to a fast algorithm for differentiating an ϵ-truncated wavelet representation of a function. We simply apply the finite difference operator locally on each scale, j, where there are no t_ϵ coefficients on finer scales. As the inverse wavelet transform is local we can keep the number of operations at $\mathcal{O}(N_s)$. Convolution leads to an increase in support, so we need to add some extra coefficients at the edge points, so that the *final* coefficients are right. Furthermore, the differentiated function might have significant coefficients in new regions. We should therefore add a region where new coefficients might appear. The simplest way to do this is to study some more coefficients at the 'edges' of each t_ϵ^j. The total algorithm will then go from the coarsest scale, up to the finest scale, and down to the coarsest, as shown in Figure 11. As this cycle is opposite to the so called V-cycle in a multigrid algorithm (which goes from fine scales to coarse), we call this a Λ-cycle.

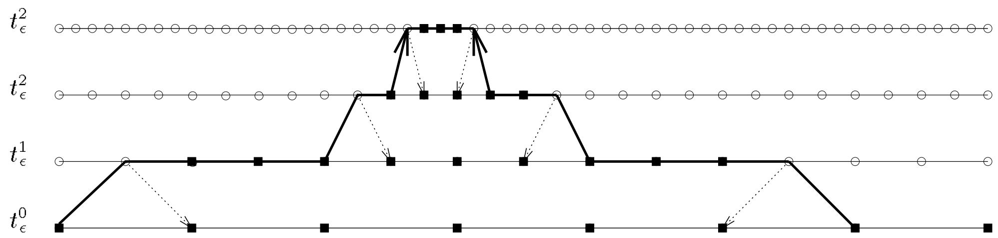


Figure 11: *Λ-cycle for fast differentiation of ϵ-truncated wavelet expansion, τ_ϵ.*

The addition of coefficients at 'edges' of t_ϵ^j is done through the operation **smear**. Until now we have viewed the truncated transformed signal as filters where many of the coefficients are zero. As we need to keep track of where the non-zero coefficients are to make the algorithm efficient and we do not want to waste memory on coefficients that are zero, we introduce a different, *linked list*, representation that is efficient for a filter with many zero coefficients.

Definition 3.2 A *sparse filter* is a (possibly bi-infinite) ordered list of pairs of numbers,

$$u = (\ldots, (u_{k-1}, n_{k-1}), (u_k, n_k), (u_{k+1}, n_{k+1}), \ldots), \qquad \forall j : u_j \in \mathbb{C},\ n_j \in \mathbb{Z},$$

such that $n_{j+1} > n_j$, $\forall j$.

We also define the *used set*, $S_u = \{k : \exists y : (y, k) \in u\}$. The interpretation of this definition is that $u(k) = y$ if $(y, k) \in u$, and $u(k) = 0$ if $k \notin S_u$. A filter is thus represented as in Figure 12. This sparse representation is used for the various numerical examples in the subsequent sections. The adding of t extra coefficients at 'edges' of a sparse filter can now formally be defined as

Definition 3.3 The sparse filter, $v = \mathbf{smear}_t(u)$, has its coefficients defined by $(x, s) \in v \Leftrightarrow (x, s) \in u$, or $(x = 0,\ k \notin S_u,\ \exists r,\ r \in S_u, |r - s| \leq t)$.

We also define an approximate left inverse of the smearing, **sharpen**, that takes away all of the added zero coefficients, except those that 'filled in holes' when smearing.

Definition 3.4 The sparse filter, $v = \mathbf{sharpen}_t(u)$, has its coefficients defined by $(x, s) \in v \Leftrightarrow (x, s) \in u$ and $\forall r : |s - r| \leq t, t \in S_u$.

The convolution previously defined is support changing, (and can of course be performed on sparse filters). We now define a convolution that keep the same support as *the second filter* (and thus it is not commutative).

Definition 3.5 *Sparse convolution* is defined by $(x,k) \in h *^{\text{sparse}} u \Leftrightarrow x = (h * u)(k),\ k \in S_u$.

We also define the operation $\mathbf{join}(u,v)$ of two filters that is the 'union' of u and v, where v has priority.

Definition 3.6 The operation, $w = \mathbf{join}(u,v)$, is defined by $(x,k) \in w \Leftrightarrow (x,k) \in v$ or $((x,k) \in u,\ k \notin S_v)$.

Finally, we define sparse addition, where the coefficients come from the intersection of the coefficients of the added signals.

Definition 3.7 *Sparse addition* is defined by $(z,k) \in w \Leftrightarrow z = x + y,\ k \in S_u,\ k \in S_v$.

With these definitions, one step in the Λ-cycle is defined as in Figure 13. Here, sparse convolution is used, $\mathbf{smear}_{t+r}$ uses t extra coefficients to avoid edge errors with sparse convolution, and r extra coefficients might be needed at the edges in the result. The convolution of 'box' and 'arrow' keeps the support of the 'arrow,' and the signal entering the $<$ part of **join** has priority.

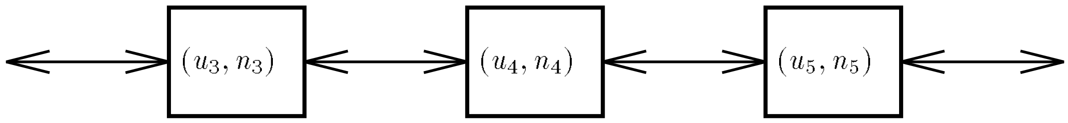


Figure 12: *Linked list representation of filters.*

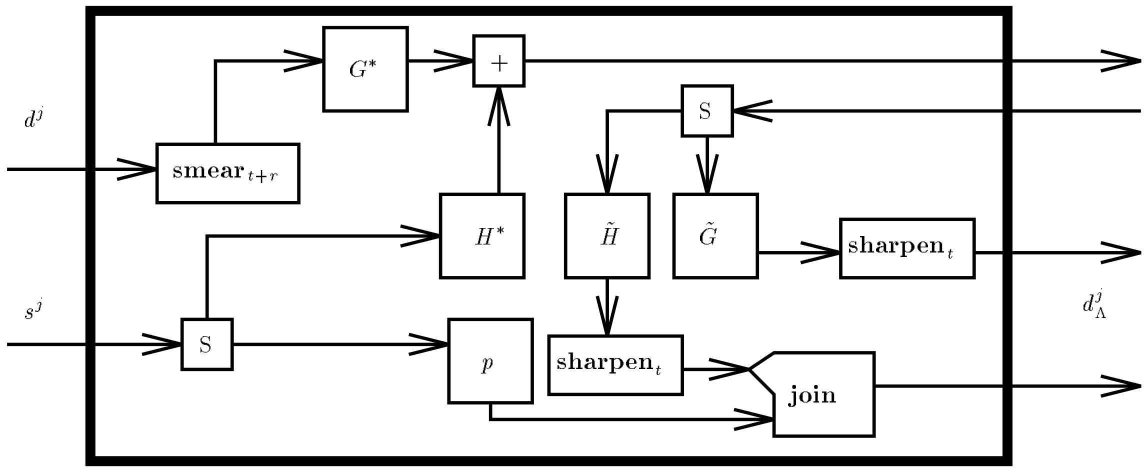


Figure 13: *One step in the Λ_p-cycle.*

The overall cycle can now be performed by doing the Λ-cycle as shown in Figure 14, and the total operation is called Λ_p, where p is the finite difference filter involved. We note that the Λ-cycle can be defined for any filter, p, not only when p is an approximation of a differential operator. The operational count for the two-level Λ-cycle, on (t^1_ϵ, t^0) when t^1_ϵ is on the form $(t_k, \ldots, t_{k+s})$ (sequential coefficients), t^0 is on the form $(t_0, \ldots, t_{N-1})$, and length $p(h) = a$, length $\tilde{h} = b$, length $h = c$ is

$$\mathrm{OP}_\Lambda = 2 + 2ac + 2bc + c^2 - ab + (6c + 2b)s + 2aN, \tag{72}$$

compared with applying p directly on the finer level,

$$\mathrm{OP}_{\text{full}} = 4aN. \tag{73}$$

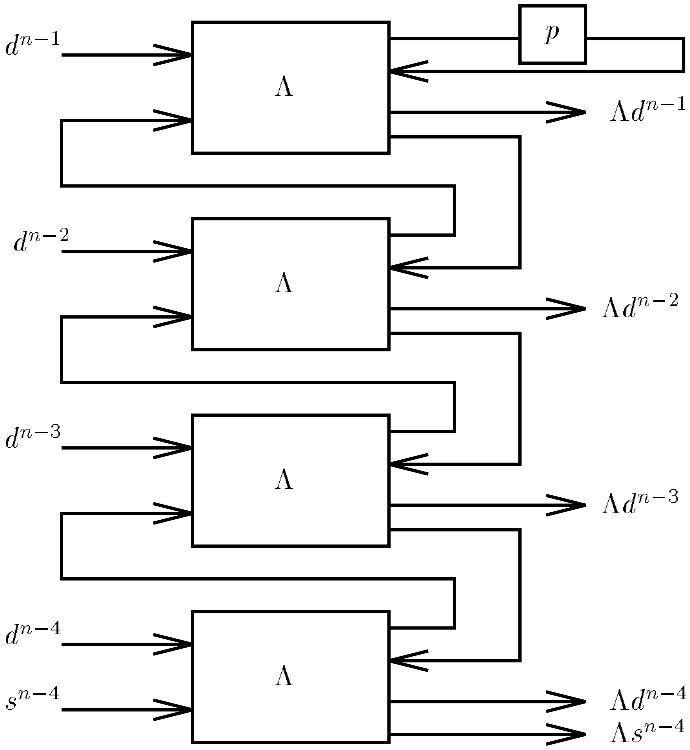


Figure 14: *Total sparse differentiation with* Λ*-cycle.*

We here assume that all coefficients within the support of the filters are non-zero. The β_2 filter bank has $b = c = 4$ and with the standard 4th order difference operator with $a = 5$ we get $\mathrm{OP}_\Lambda = 70 + 32s + 10N$, which should be compared with the work done by applying the operator directly on the finest grid, $4aN$, so the break-even is when $70 + 32s = 6N$. This gives us an idea of what the potential gains of the sparse method is. For the δ_3-filter, the operational count will be $-8 + 20s + 10N$.

We now study the same test function, $f = \sin(2\pi x) + e^{-20000(x-0.5)^2}$, with the same filter bank, β_2, the centered 4th order finite difference approximation of $\frac{d}{dx}$, $d(h) = \frac{1}{h}(-\frac{1}{12}, \frac{4}{3}, \underline{0}, -\frac{4}{3}, \frac{1}{12})$ and truncate with $\epsilon = 10^{-4}$. We choose the finest level, $n = 10$, and transform $k = 4$ levels forward. In Figure 15 the derivative of this function is shown, and in Figure 16 we see the error in the approximated derivative when approximating with three different methods. The first method is simply using the finite difference filter on the function,

$$e_{\mathrm{u}} = d(2^{-n})[f]_{2^{-n}} - \left[\frac{df}{dx}\right]_{2^{-n}}. \tag{74}$$

The second method is using the Λ-cycle, and truncating the result,

$$e_\Lambda = \mathcal{F}^{-k}\mathcal{T}_\epsilon\Lambda_d\mathcal{T}_\epsilon\mathcal{F}^k[f]_{2^{-n}} - \left[\frac{df}{dx}\right]_{2^{-n}}. \tag{75}$$

The last method is transforming the truncated signal all the way back to the finest level everywhere, performing $d(2^{-n})$ on this signal, transforming forward, and truncating,

$$e_{\mathrm{a}} = \mathcal{F}^{-k}\mathcal{T}_\epsilon\mathcal{F}^k d(2^{-n})\mathcal{F}^{-k}\mathcal{T}_\epsilon\mathcal{F}^k[f]_{2^{-n}} - \left[\frac{df}{dx}\right]_{2^{-n}}. \tag{76}$$

The error in Figure 16 is plotted in $\log_{10}$ scale, because of the large ranges. The lower part of the figure shows the non-zero coefficients of the representation. We see that the untruncated approximation performs best, but the Λ-cycle performs better than the approach of transforming back to the finest level everywhere which seems to introduce high frequency errors, that ruins the sparseness of the representation. In max-norm the Λ-cycle performs as well as the untruncated approximation as seen in Figure 17, where the error for the three methods is shown around the 'spike'. This is natural as the wavelet method detects local high frequencies and the Λ-cycle transforms all the way back in such regions.

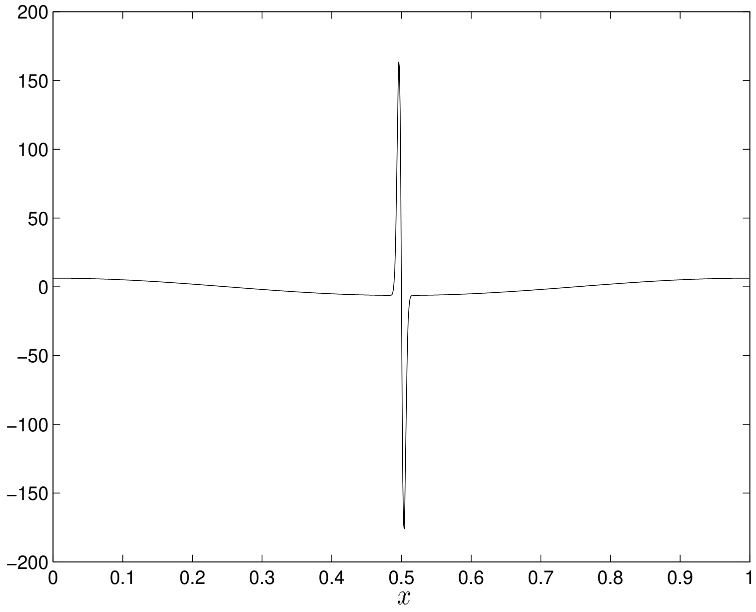


Figure 15: *Derivative of* $f = \sin(2\pi x) + e^{-20000(x-0.5)^2}$.

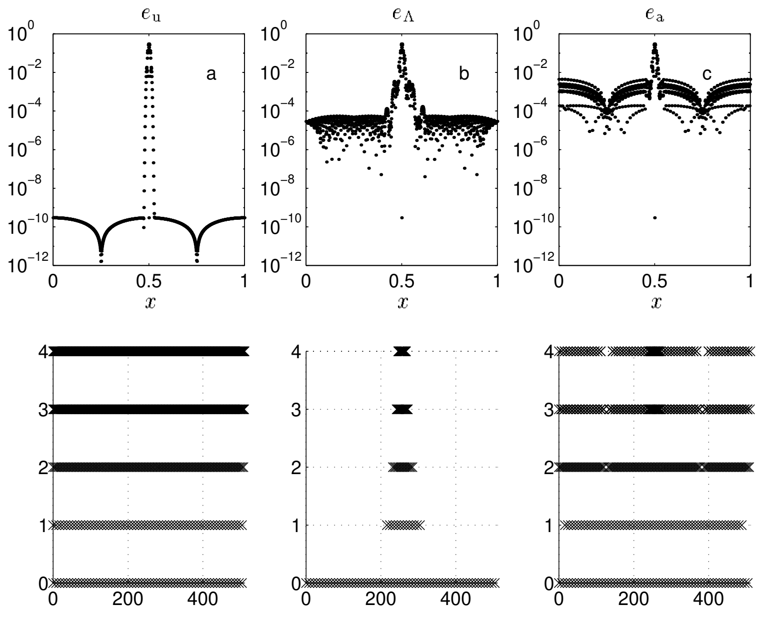


Figure 16: *Error, and sparse representation for three methods to differentiate, untruncated (a), Λ-cycle (b), full inverse transform (c).*

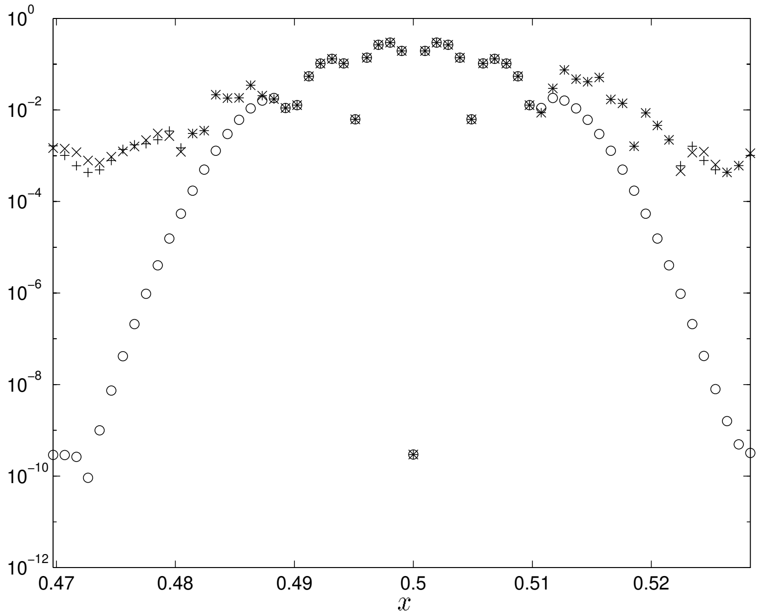


Figure 17: *Error around 'spike' for untruncated 'o', Λ-cycle '+', full inverse transform '×'.*

The index, r, in $\mathbf{smear}_{t+r}$ decides how many extra coefficients at the edges of the sparse filter that might be added. Unfortunately extra coefficients might be needed at a *higher level* in the representation. This will, e.g., be the case when solving Burgers' equation, where a shock forms as time increases. The Λ-cycle described will then be inappropriate, and we need to do some extra work to find a good approximation of the derivative. The trick is to transform back an extra level *everywhere*, to see if the frequencies increase in some region. This leads to the a modified cycle, the $\tilde{\Lambda}$-cycle, one step of which is shown in Figure 18. This cycle is more expensive in operational count than the Λ-cycle, and it needs at least $k = 2$ steps in the transform to be able to outperform the untruncated operation. The total cycle is shown in Figure 19. The addition operation in the figure is ordinary addition (*not* sparse addition).

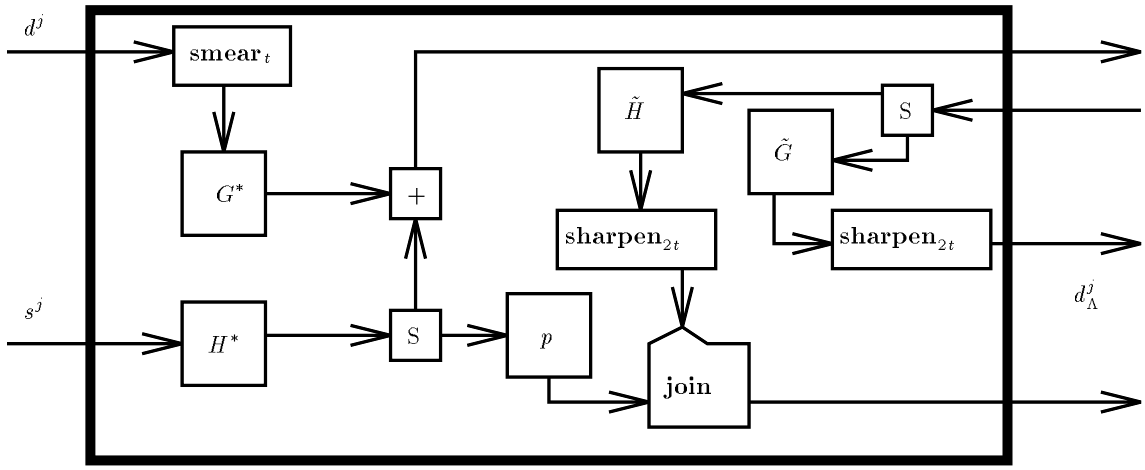


Figure 18: *One step in the $\tilde{\Lambda}$-cycle.*

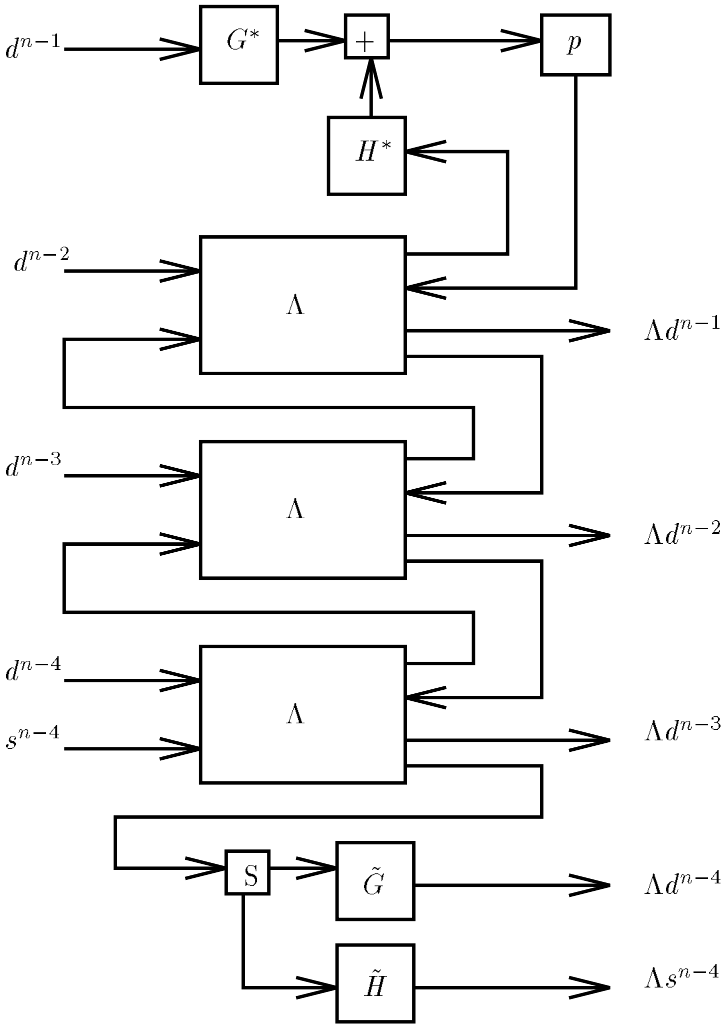


Figure 19: *Total sparse differentiation with $\tilde{\Lambda}$-cycle.*

We have used the Λ-cycle for the linear PDEs in subsequent sections, and the $\tilde{\Lambda}$-cycle for the non-linear PDEs.

3.3 Multiplication of truncated functions

The success of a multiplication Λ-cycle depends, just as in the case of differentiation, on whether polynomials can be treated on coarse scales. We need to determine when

$$(H^*)^j(\tilde{H}^j[x^k]_h \times \tilde{H}^j[x^s]_h) = [x^{k+s}]_h,$$

which is needed for multiplication of two wavelet representations, and when

$$(H^*)^j([x^k]_{2^j h} \times \tilde{H}^j[x^k]_h) = [x^{k+s}]_h,$$

which is needed for direct multiplication of a wavelet representation and a function. Here, $\times$, means coefficient-wise multiplication of filters, $(u \times v)_k = u_k v_k$. Unfortunately, we are not in the same luck as in the case of differentiation. We have the following result.

Theorem 3.5 *Suppose $(h, g, \tilde{h}, \tilde{g})$ is a perfect reconstruction filter bank, and $\tilde{g}$ has N zero-moments. Then a necessary and sufficient condition for*

$$(H^*)^j(\tilde{H}^j[x^k]_h \times \tilde{H}^j[x^s]_h) = [x^{k+s}]_h, \;\; \forall j, \; \forall k, \forall s : s + k \leq N_2 \leq N - 1, \tag{77}$$

is that

$$\sigma_p(\tilde{h}) = \sqrt{2}\left(\frac{\sigma_1(\tilde{h})}{\sqrt{2}}\right)^p, \qquad p = 1, \ldots, N_2 - 1. \tag{78}$$

A necessary and sufficient condition for

$$(H^*)^j([x^k]_{2^j h} \times \tilde{H}^j[x^s]_h) = [x^{k+s}]_h,\ \forall j,\ \forall k, \forall s : s + k \leq N_2 \leq N - 1, \tag{79}$$

is that

$$\sigma_p(\tilde{h}) = 0, \qquad p = 1, \ldots, N_2 - 1. \tag{80}$$

Proof. We start with $\tilde{H}^j[x^k]_h \times \tilde{H}^j[x^s]_h$ and prove ($\Rightarrow$). The same reasoning as in Theorem 3.4 shows that it is the same to say that

$$(\tilde{H}[x^k]_h \times \tilde{H}[x^s]_h) = \tilde{H}[x^{k+s}]_h,$$

i.e.,

$$\frac{1}{2}\sum_{n,n_2} \tilde{h}_{n-2p}\tilde{h}_{n_2-2p} n^s n_2^k = \frac{1}{\sqrt{2}}\sum_n \tilde{h}_{n-2p} n^{s+k},$$

or

$$\frac{1}{2}\sum_{n,n_2} \tilde{h}_n\tilde{h}_{n_2}(n+2p)^s(n_2+2p)^k = \frac{1}{\sqrt{2}}\sum_n \tilde{h}_n(n+2p)^{s+k}. \tag{81}$$

This is trivial for $s = 0$, $k < N$. For $s > 0$ we write the r.h.s of (81) as

$$\begin{aligned}
&\frac{1}{\sqrt{2}}\sum_n \tilde{h}_n(n+2p)^{s+k}\frac{1}{\sqrt{2}}\sum_{n_2}\tilde{h}_{n_2} \\
&\quad = \frac{1}{2}\sum_{n,n_2}\tilde{h}_n\tilde{h}_{n_2}(n_2+2p+n-n_2)^s(n+2p)^k \\
&\quad = \frac{1}{2}\sum_{n,n_2}\tilde{h}_n\tilde{h}_{n_2}(n_2+2p)^s(n+2p)^k \\
&+\frac{1}{2}\sum_{t=1}^{s}\binom{s}{t}\sum_{n,n_2}\tilde{h}_n\tilde{h}_{n_2}(n-n_2)^t(n_2+2p)^{s-t}(n+2p)^k,
\end{aligned}$$

so (81) is equivalent to

$$\sum_{t=1}^{s}\binom{s}{t}\sum_{n,n_2}\tilde{h}_n\tilde{h}_{n_2}(n-n_2)^t(n_2+2p)^{s-t}(n+2p)^k = 0,\ \forall p. \tag{82}$$

From the moment conditions of $\tilde{h}$ we can write this as

$$2\sum_{t=1}^{s}\binom{s}{t}\left(\frac{\sigma_1(\tilde{h})}{\sqrt{2}}\right)^t\left(\sum_{v=0}^{t}\binom{t}{v}(-1)^v\right)\left(\frac{\sigma_1(\tilde{h})}{\sqrt{2}}+2p\right)^{s-t}\left(\frac{\sigma_1(\tilde{h})}{\sqrt{2}}+2p\right)^k = 0. \tag{83}$$

This can be seen by Binomially expanding the powers, and replacing all sums over n, n_2, with the moment conditions. But (83) is trivially satisfied as the sum over v equals zero. So the ($\Rightarrow$) part of the theorem when multiplying two wavelet expansions is proved.

For the ($\Leftarrow$) part, we choose $s = 1$, $p = 0$, and $k > 0$ as the first number for which $\sigma_{k+1}(\tilde{h}) \neq \sqrt{2}(\sigma_1(\tilde{h})/\sqrt{2})^{k+1}$. We then have that (82) equals

$$\sum_{n,n_2} \tilde{h}_n \tilde{h}_{n_2}(n-n_2)n^k = \sqrt{2}\sigma_{k+1}(\tilde{h}) - \sigma_k(\tilde{h})\sigma_1(\tilde{h}) \neq 0,$$

so (82) is not satisfied, and as H^* is injective for polynomials, $(H^*)(\tilde{H}[x]_h \times \tilde{H}[x^M]_h) \neq [x^{M+1}]_h$.

We now turn to the second part of the theorem. Equation (79) is equivalent to

$$[x^k]_{2h} \times \tilde{H}[x^s]_h = \tilde{H}[x^{k+s}]_h, \qquad \forall k, \forall s : s + k \leq N_2 \leq N-1,$$

i.e.,

$$\frac{1}{\sqrt{2}} \sum_n \tilde{h}_n (n+2p)^s (2p)^k = \frac{1}{\sqrt{2}} \sum_n \tilde{h}_n (n+2p)^{s+k}, \qquad \forall p. \tag{84}$$

The l.h.s. of (84) is

$$\frac{1}{\sqrt{2}} \sum_n \tilde{h}_n \sum_{j=0}^{s} \begin{pmatrix} s \\ j \end{pmatrix} n^j (2p)^{s-j} (2p)^k = \sum_{j=0}^{s} \begin{pmatrix} s \\ j \end{pmatrix} \sigma_j(\tilde{h})(2p)^{s+k-j},$$

so $\sigma_j = 0$, $j = 1, \ldots, N_2 - 1$ is necessary and sufficient for (84) to be satisfied.

□

The filter bank β_2 will therefore fail to multiply linear functions correctly on lower levels. To get higher order results in the multiplication, we must have filter banks where $\tilde{h}$ satisfies zero-moment conditions. This was noted in [21] for the Galerkin method, where coiflets were used. The filter banks in Table 3 satisfy zero-moment conditions, but these 'waste' zero-moments on h, and the filter banks in Table 4 are preferably used. When $\tilde{h}$ corresponds to a scaling function, $\tilde{\varphi} \in L^2$, we know from [29] that

$$\Sigma_p(\varphi) = \frac{1}{2^p - 1} \sum_{i=1}^{p} \begin{pmatrix} p \\ i \end{pmatrix} \sigma_i(\tilde{h}) \Sigma_{p-i}(\tilde{\varphi}), \tag{85}$$

so (78) is equivalent to

$$\Sigma_p(\tilde{\varphi}) = (\Sigma_1(\tilde{\varphi}))^p, \qquad p = 1, \ldots, N_2 - 1, \tag{86}$$

and also to

$$\Sigma_p(\tilde{\varphi}(\cdot + \Sigma_1(\tilde{\varphi}))) = 0, \qquad p = 1, \ldots, N_2 - 1. \tag{87}$$

With this in mind we now turn to the algorithm for multiplying two functions.

The multiplication of a filter $u = [f]_{2^{-n}}$ and a sparse wavelet representation, τ_ϵ will lead to the same method as for differentiation, except that the convolution with $p(h)$ is replaced by coefficient-wise multiplication with $[f]_{2^{-n+j}}$. When multiplying *two* functions in wavelet representation, we define a step in the $\Lambda_\times$-, and $\tilde{\Lambda}_\times$-cycles in Figure 20 and Figure 21. The total Λ-cycle is shown in Figure 22. The total $\tilde{\Lambda}$-cycle can also be defined in the same manner as for differentiation.

For numerical tests we use the same test function as in the previous section, $f = \sin(2\pi x) + e^{-20000(x-0.5)^2}$, and square it giving the function in Figure 23. We compare the Λ-cycle, with transforming all the way back to the finest scale, giving

$$e_\Lambda = \mathcal{F}^{-k} \mathcal{T}_\epsilon \Lambda_{\times^2} \mathcal{T}_\epsilon \mathcal{F}^k [f]_{2^{-n}} - [f^2]_{2^{-n}}, \tag{88}$$

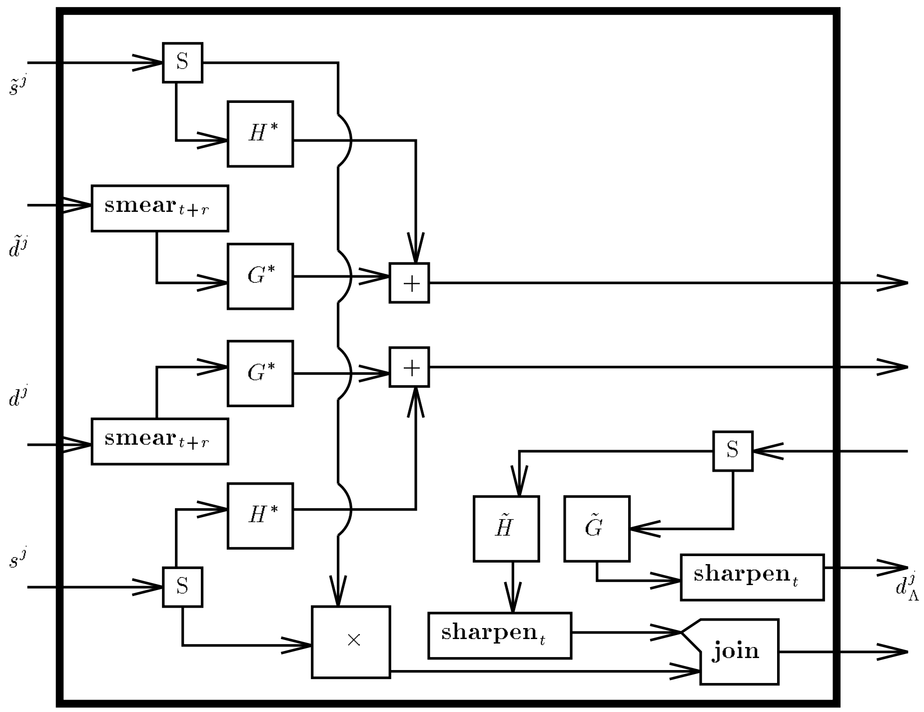


Figure 20: *One step in the $\Lambda_\times$-cycle.*

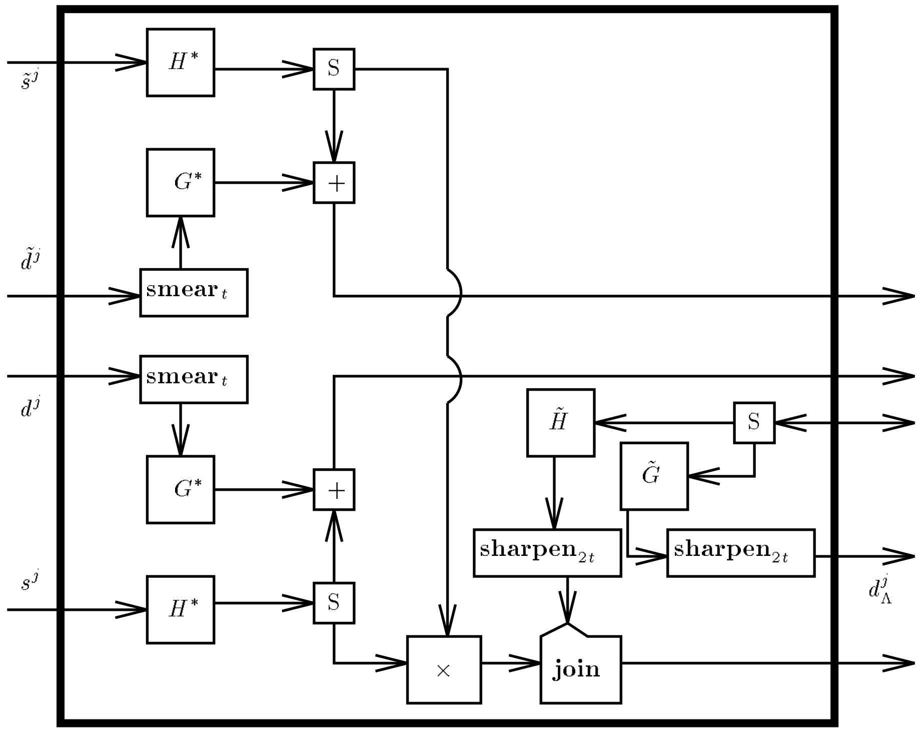


Figure 21: *One step in the $\tilde{\Lambda}_\times$-cycle.*

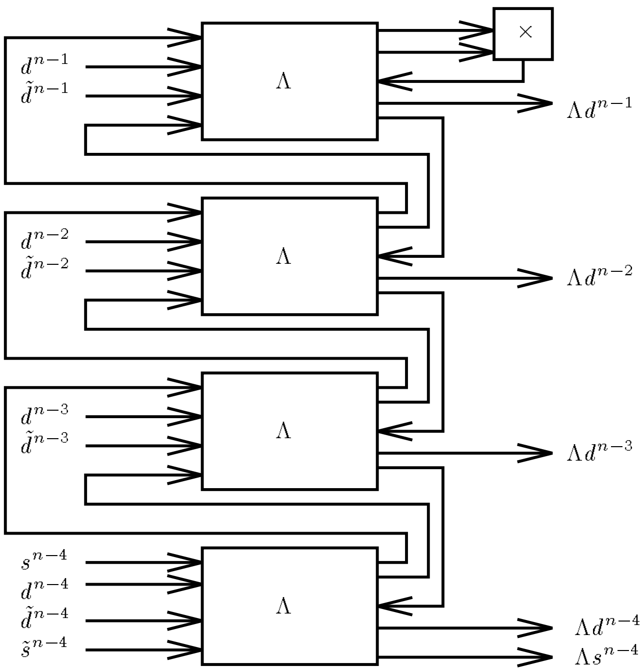


Figure 22: *Total sparse multiplication.*

$$e_{\mathrm{a}} = \mathcal{F}^{-k}\mathcal{T}_\epsilon\mathcal{F}^k(\times^2)\mathcal{F}^{-k}\mathcal{T}_\epsilon\mathcal{F}^k[f]_{2^{-n}} - [f^2]_{2^{-n}}. \tag{89}$$

In our first example we use the filter bank β_2, and choose $n = 10$, $k = 4$, $\epsilon = 10^{-4}$. The error is shown in Figure 24, plotted in $\log_{10}$ scale. We see here that, in contrast to differentiation, the method of transforming back everywhere gives a smaller error than the Λ-cycle, which seems to introduce a low-frequency error. This comes from the lack of zero-moments for $\tilde{h}$. As the total error is a combination of the error in the truncation step, $\mathcal{T}_\epsilon$, and in the Λ-cycle, this effect will only show for 'small' ϵ.

In our second example, we use the δ_3 filter bank, with the same parameters as in the previous example. The result is shown in Figure 25. We see here that we have improved the result, and there is no large low-frequency component in the error. The results with this filter bank when truncating, and differentiating with Λ-cycles are similar to the result when using the β_2 filter bank.

In the following two sections we shall use the filter bank method and the Λ-, $\tilde{\Lambda}$-cycles to solve PDEs in one and two dimensions.

4 Numerical experiments in one dimension

The methods described in the previous section are efficient for differentiating and multiplying functions in the functions spaces $\mathcal{L}^{s,m}$, i.e., functions that are smooth nearly everywhere, but with a few small regions with sharp gradients. Efficient here means that for F, an operator involving multiplication and differentiation:

- u can be represented by u_ϵ, with $N_{\mathrm{s}} \ll N$ coefficients, where $N = 2^n$ is the number of coefficients on the finest level.
- there is an approximation of F, $\tilde{F}$ such that $\tilde{F}u_\epsilon$ can be evaluated in $\mathcal{O}(N_{\mathrm{s}})$ arithmetic operations.
- the error, $e = Fu - \tilde{F}u_\epsilon$, is small, typically $\|e\| = \mathcal{O}(N^{-\alpha} + \epsilon^{-\gamma})$, with $\alpha > 1$, $\gamma < 1$.

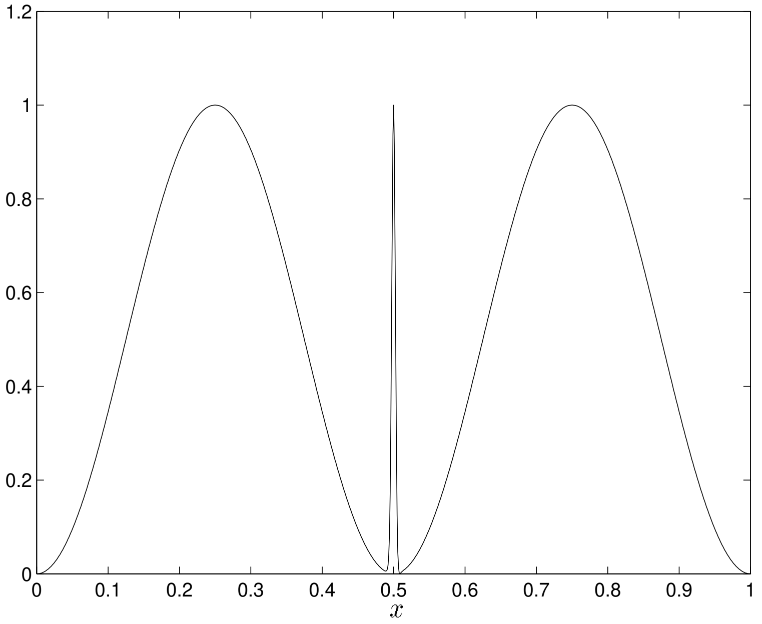


Figure 23: *Square of* $f = \sin(2\pi x) + e^{-20000(x-0.5)^2}$.

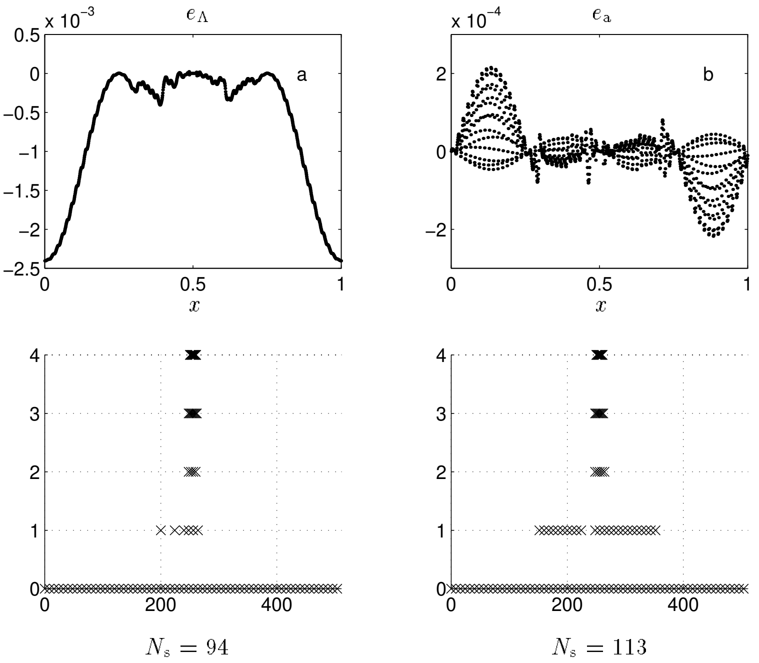


Figure 24: *Error, and sparse representation for two methods to perform multiplication,* Λ*-cycle (a), full inverse transform (b), with* β_2 *filter bank,* $\epsilon = 10^{-4}$.

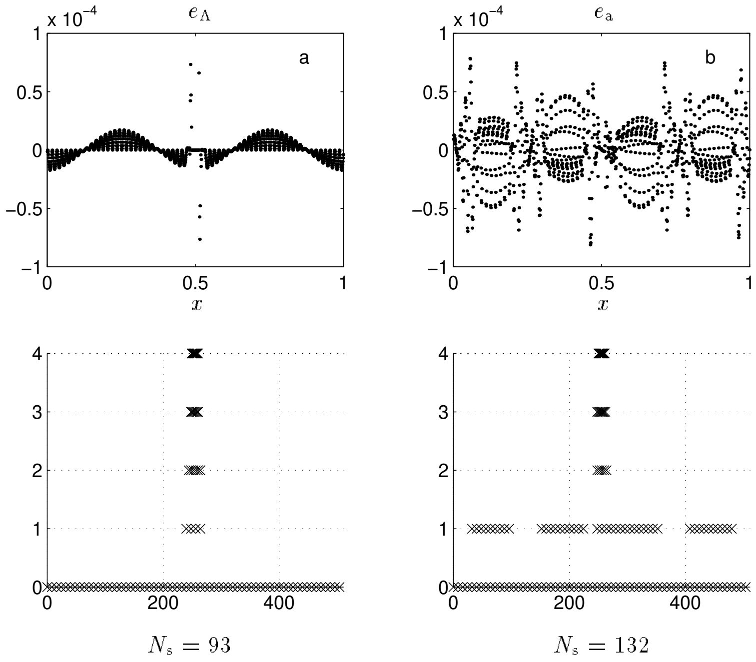


Figure 25: *Error, and sparse representation for two methods to perform multiplication, Λ-cycle (a), full inverse transform (b), with δ_3 filter bank, $\epsilon = 10^{-4}$.*

A method that satisfies these properties will be well-suited to use in an explicit solver for a hyperbolic PDE. We shall now see how well the Λ, $\tilde{\Lambda}$-cycles perform.

We consider a general initial value problem, with periodic b.c.s,

$$\begin{cases} u_t = P(d/dx, x, t, u), \\ u(0,t) = u(1,t), \\ u(x,0) = u_0(x), \\ 0 \leq x \leq 1,\ t \geq 0. \end{cases} \tag{90}$$

We suppose P can be Taylor expanded in each argument, and that the series can be truncated (without changing the solution 'too much'), giving a right-hand side of the equation that can be evaluated by repeated multiplication and differentiation. We discretize in space with a grid (filter), $\tilde{u}$, $\Delta x = 2^{-n}$ and use an explicit time-marching scheme, with time step Δt. We will use the 4th order classical Runge–Kutta method in all examples. This gives us an approximation

$$u(n\Delta t, k\Delta x) \approx \tilde{u}_k^n.$$

The time step has been chosen somewhat smaller than optimal in all examples, although the Λ-, $\tilde{\Lambda}$-cycles do not seem to decrease the optimal time step. This has earlier been noted for similar methods, e.g., in [19].

When implementing the Λ-cycles on a computer, linked lists were used to represent the sparse filters. As there is a considerable amount of overhead involved in finding coefficients, etc., blocks of $n_b = 2^l$ coefficients, (with l typically between 2 and 5), were grouped together, giving a representation as in Figure 26. By choosing a power of 2 for n_b the transformation between block- and coefficient-level is fast. The block versions of $\mathcal{T}_\epsilon$, $*^{\text{sparse}}$, **smear**, **sharpen**, **join**, are defined in the same manner as previously. We redefine N_s to mean the number of *blocks* in the thresholded expansion (instead of the number of *coefficients*).

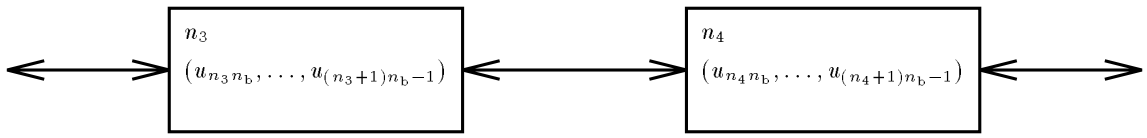


Figure 26: *Block linked list representation of filters.*

In [17] a tree representation was chosen for the transformed signal. The advantage of this representation is that it is easy to find coefficients that have overlapping support on different levels. On the other hand, extra work is needed to construct such a tree representation. In [17] all calculations were performed with the wavelet representation, and coefficients on different levels were needed at the same time. With the Λ-cycle, however, all calculations are performed with coefficients on the same level, so the links between different levels are not needed, and the linked list representation will perform better.

The one-dimensional test problems we study are the advection equation, and Burgers' equation.

4.1 The advection equation

We solve

$$\begin{cases} u_t = u_x, & 0 \le x \le 1, \quad t \ge 0, \\ u(x,0) = u_0(x), & u = u(x,t), \quad u(0,t) = u(1,t). \end{cases} \tag{91}$$

We choose the initial function, $u_0(x) = \sin(2\pi x) + e^{-5\times 10^5 (x-0.5)^2}$, which is the same as in the previous examples, but with a sharper 'spike.' The solution is a translation of the initial condition, $u(x,t) = u_0((x+t) \mathrm{mod} 1)$. We choose $\Delta t = 10^{-4}$, and step forward one period to $t = 1$. The parameters are $n = 13$, $k = 4$, $\epsilon = 6 \times 10^{-5}$, and we use the β_2 filter bank. Finally, we use the second order approximation of d/dx, $d(h) = D_0 = \frac{1}{2h}(1, \underline{0}, -1)$. The number of blocks in the approximated solution (each block containing $n_b = 8$ coefficients) was never more than 80, out of a total of 1024, so the compression factor was about 13. In Figure 27 we see the approximated solution. We see here that the 'spike' has been resolved, and it has not decreased in magnitude. In Figure 28 we show the solution and the approximation around the 'spike'. In Figure 29 we compare the error (plotted in $\log_{10}$ scale) of the Λ-cycle with solving without truncation. We see that the untruncated method performs somewhat better, even in max-norm. When solving with $n = 12$, the untruncated method did not manage to resolve the spike, and the error was large. The same pattern appeared in the other numerical examples. *When the untruncated method solved the problem properly, the Λ-cycle (and, for non-linear problems, $\tilde{\Lambda}$-cycle) usually did so too, with much fewer coefficients, but with a somewhat larger error.*

We next sharpen the 'spike' even more and choose, $u_0(x) = \sin(2\pi x) + e^{-4\times 10^6 (x-0.5)^2}$. For this problem it was difficult for β_2, and D_0 to resolve the 'spike' properly, and some high frequency errors appeared, ruining the sparse representation of the signal. By choosing β_4, and the 4th order scheme, $d(h) = \frac{1}{h}(-\frac{1}{12}, \frac{4}{3}, \underline{0}, -\frac{4}{3}, \frac{1}{12})$, this problem disappeared. The parameters were $n = 15$, $k = 6$, $\Delta t = 2.5 \times 10^{-5}$, $\epsilon = 10^{-6}$. The approximation at $t = 1$ is shown in Figure 30, and the error in Figure 31. We see that 85 blocks out of a total of 2048 are used, giving a compression rate of 24.1.

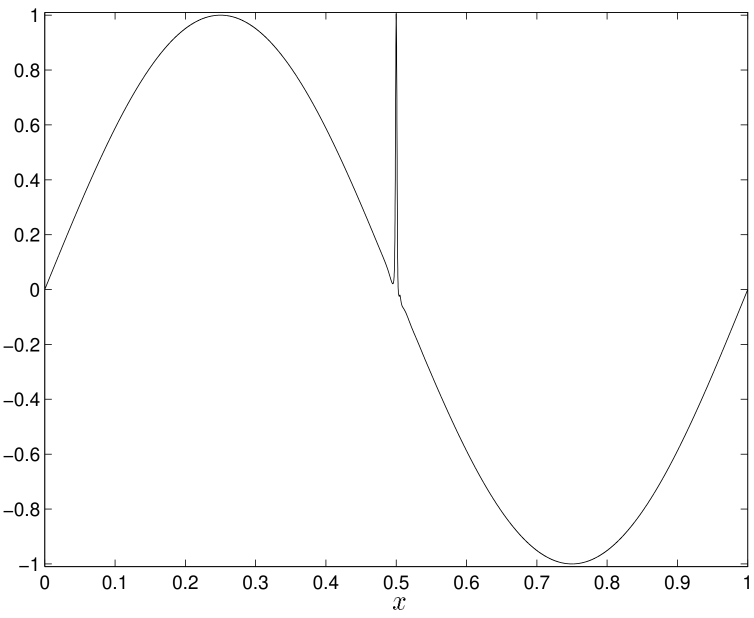


Figure 27: *Approximated solution at* $t = 1$ *when solving* $u_t = u_x$*, with* $\tilde{\Lambda}$*-cycle, and* β_2 *filter bank.*

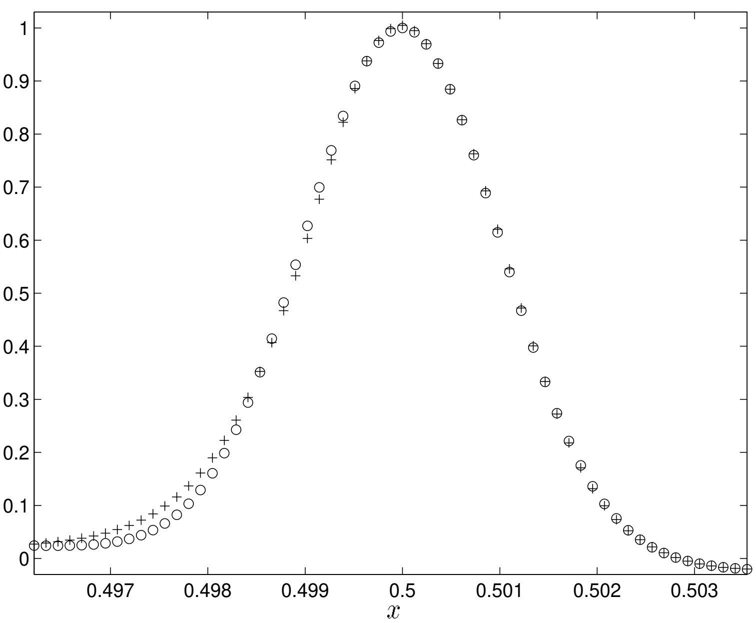


Figure 28: *Solution, 'o', and Approximated solution, '+', around 'spike' at* $t = 1$ *when solving with* Λ*-cycle, and* β_2*.*

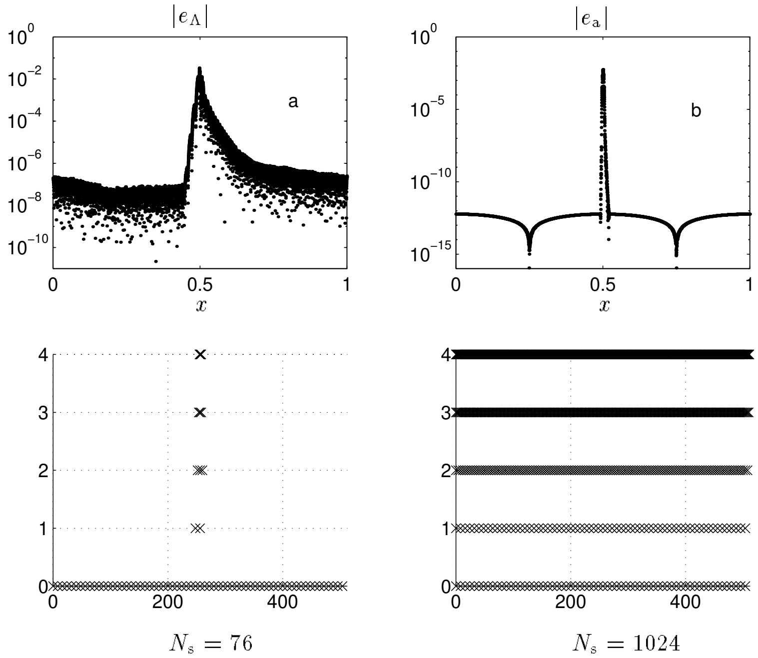


Figure 29: *Error, and sparse representation for two methods of solving $u_t = u_x$, Λ-cycle with filter bank β_2 (a), untruncated (b).*

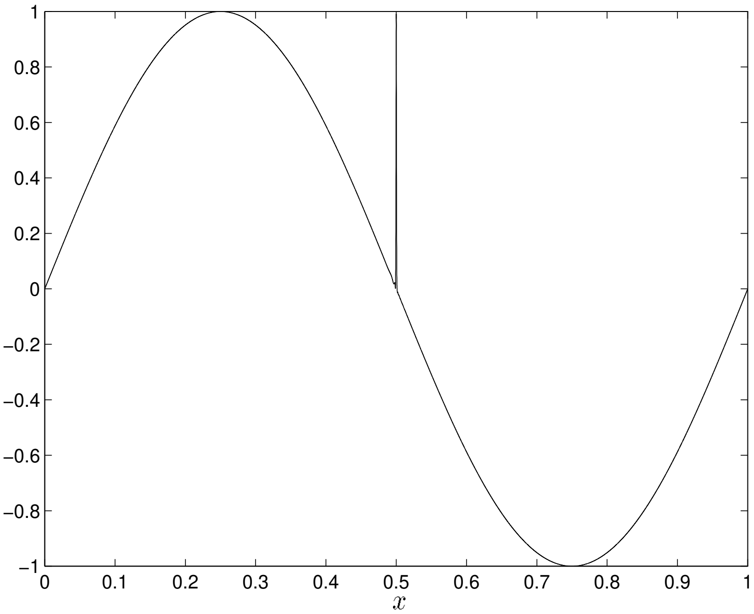


Figure 30: *Approximated solution at $t = 1$ when solving $u_t = u_x$, with Λ-cycle, and filter bank β_4.*

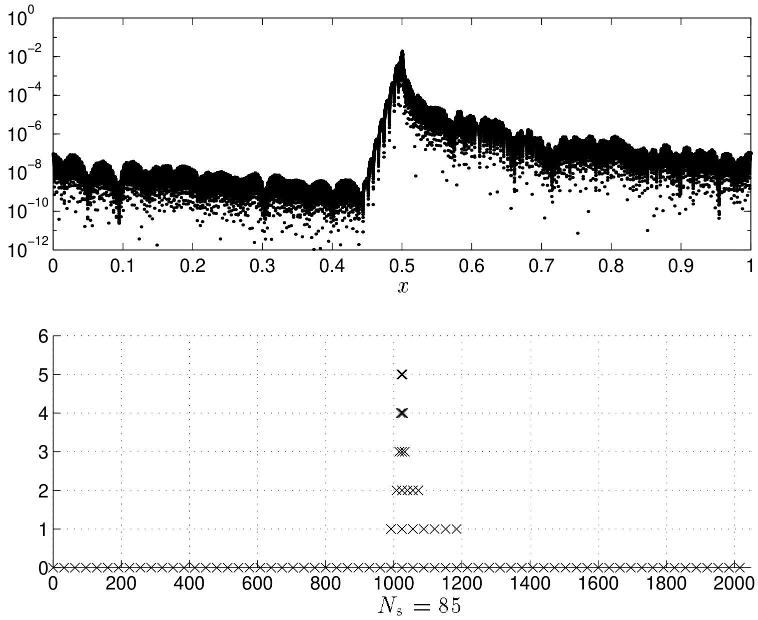


Figure 31: *Error, and representation when solving* $u_t = u_x$, *with* Λ*-cycle, and filter bank* β_4.

4.2 Burgers' equation

We want to solve the non-linear Burgers' equation,

$$\begin{cases} u_t + uu_x = \mu u_{xx}, & 0 \le x \le 1, \quad t \ge 0, \mu > 0, \\ u(x,0) = \sin(2\pi x), & u = u(x,t), \quad u(0,t) = u(1,t). \end{cases} \tag{92}$$

The exact solution is shown in Figure 32, for viscosity, $\nu = 10^{-4}$. The difficulty with this equation, apart from being non-linear, is the very sharp gradient that forms at $x = 0.5$, although the solution is smooth in other parts of the region.

One of the conclusions in [17] was that it is difficult to perform multiplication efficiently in the wavelet domain, and this was one reason for constructing the filter bank method and the Λ-cycle, that uses point value information. We therefore examine how well the filter bank method solves the problem. We chose the β_2 filter bank, $n = 13$, $k = 7$, $\epsilon = 10^{-4}$, and the 4th order finite difference filters, $d(h) = \frac{1}{h}(-\frac{1}{12}, \frac{4}{3}, \underline{0}, -\frac{4}{3}, \frac{1}{12})$, $d_{d^2/dx^2}(h) = \frac{1}{12h^2}(-1, 16, \underline{-30}, 16, -1)$. The time step is set to $\Delta t = 2.5 \times 10^{-5}$. The approximated solution at $t = 0.1$, $t = 0.2$, $t = 0.5$ is shown in Figure 33. The block size is $n_b = 8$. Here we see how the number of blocks go from 8 on scale $\frac{1}{64}$ at $t = 0.1$, to 22 with finest resolution $\frac{1}{8192}$ at $t = 0.2$. This increase in resolution is performed automatically by the $\tilde{\Lambda}$-cycle. In Figure 34 we show the solution near the sharp gradient. There are about 10 coefficients in the shock. The compression factor is $1024/22 = 46.5$. The exact solution can be found by the Cole-Hopf transformation [7, 18]. The error in max-norm at $t = 0.2$ is $\|e\|_\infty = 0.0628$. The filter bank method with $\tilde{\Lambda}$-cycle thus performs very well for Burgers' equation, even with the filter banks without zero-moments for $\tilde{\varphi}$; zero-moments that would increase the accuracy in the multiplication step.

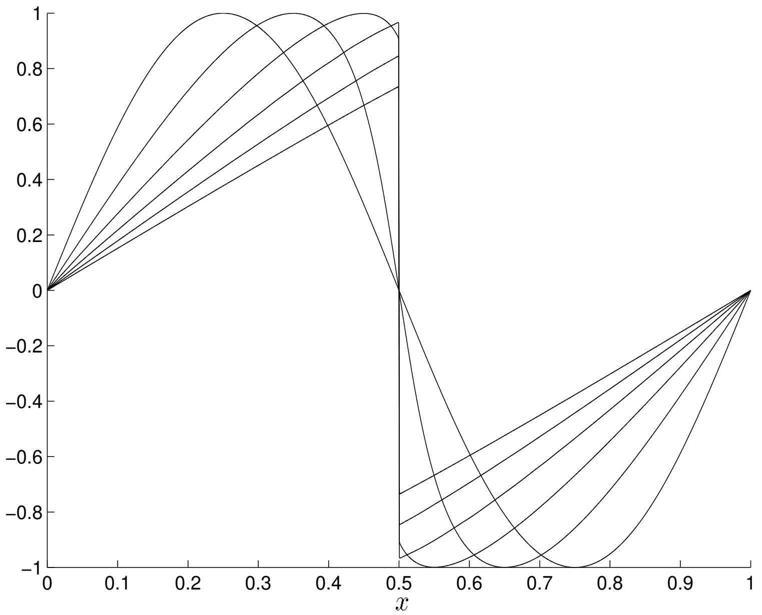


Figure 32: *Solution to Burgers' equation with viscosity,* $\mu = 10^{-4}$, *at times* $t = 0.0, 0.1, 0.2, 0.3, 0.4, 0.5$.

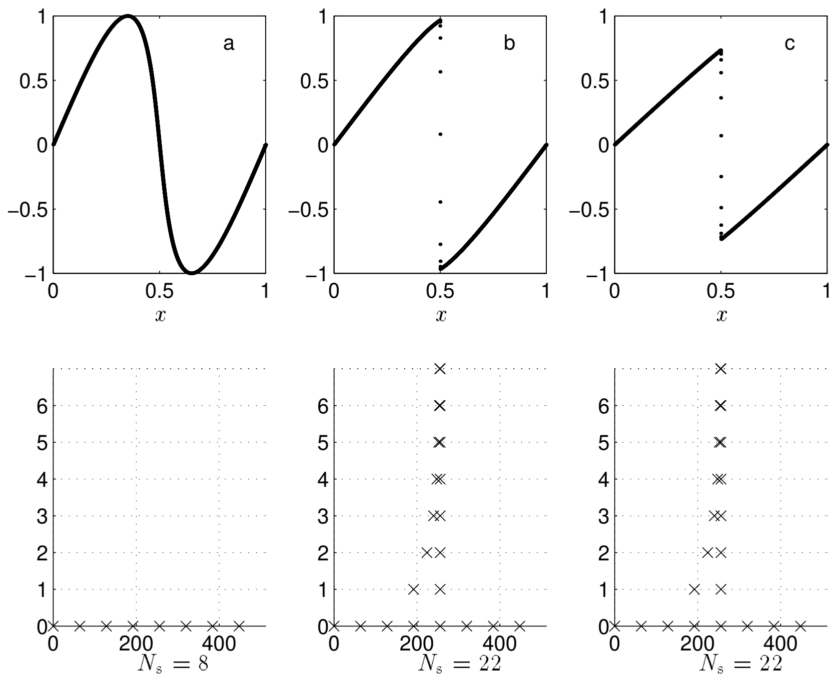


Figure 33: *Approximated solution at* $t = 0.1$ *(a),* $t = 0.2$ *(b),* $t = 0.5$ *(c), when solving Burgers' equation with viscosity* $\mu = 10^{-4}$ *using the* $\tilde{\Lambda}$*-cycle, filter bank* β_2*, and* $\epsilon = 5 \times 10^{-5}$.

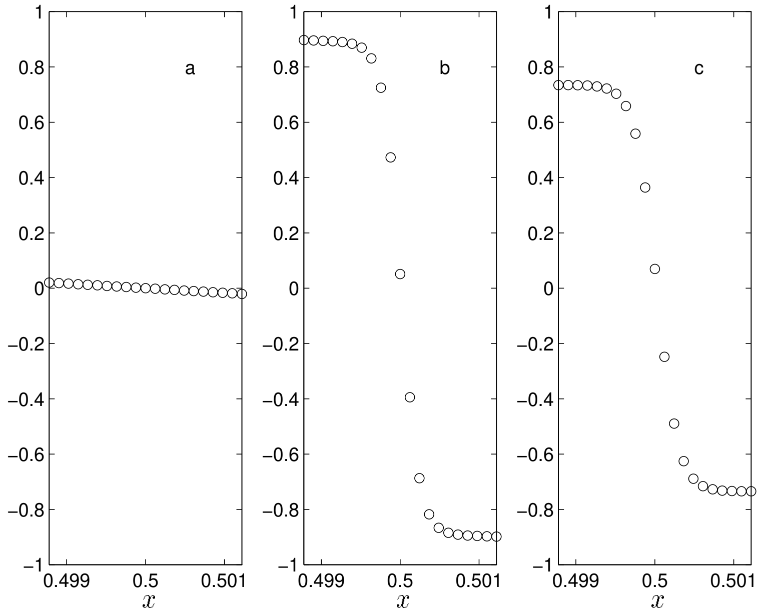


Figure 34: *Approximated solution near $x = 0.5$, at $t = 0.1$ (a), $t = 0.2$ (b), $t = 0.5$ (c), when solving Burgers' equation with viscosity $\mu = 10^{-4}$ using the $\tilde{\Lambda}$-cycle, filter bank β_2, and $\epsilon = 5 \times 10^{-5}$.*

5 Numerical experiments in two dimensions

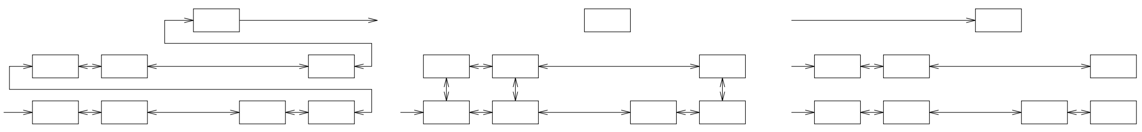
The extension of continuous wavelet theory to two dimensions, keeping the locality in both directions, is done by the tensor product spaces [9],

$$\begin{aligned} \mathcal{V}_{j+1} &= \mathcal{V}_j \oplus \mathcal{W}_j, \\ \mathcal{V}_j &= V_j \otimes V_j, \\ \mathcal{W}_j &= (V_j \otimes W_j) \oplus (W_j \otimes W_j) \oplus (W_j \otimes V_j). \end{aligned} \tag{93}$$

The filter bank transform will be on two-dimensional filters, $u : \mathbb{Z} \times \mathbb{Z} \to \mathbb{C}$, and convolution, up- and downsampling can now be made in either the x or y direction. The discrete two-dimensional filter bank transform relies on the identity

$$\begin{aligned} u = (h{*_x}{\uparrow_x}h{*_y}{\uparrow_y}{\downarrow_y}\check{\tilde{h}}{*_y}{\downarrow_x}\check{\tilde{h}}{*_x} \quad &+ \quad h{*_x}{\uparrow_x}g{*_y}{\uparrow_y}{\downarrow_y}\check{\tilde{g}}{*_y}{\downarrow_x}\check{\tilde{h}}\,{*_x} \\ +\, g{*_x}{\uparrow_x}h{*_y}{\uparrow_y}{\downarrow_y}\check{\tilde{h}}{*_y}{\downarrow_x}\check{\tilde{g}}{*_x} \quad &+ \quad g{*_x}{\uparrow_x}g{*_y}{\uparrow_y}{\downarrow_y}\check{\tilde{g}}{*_y}{\downarrow_x}\check{\tilde{g}}{*_x})u. \end{aligned} \tag{94}$$

The sparse linked list representation is not easily generalized to more than one dimension. A 'naive' approach, viewing a two-dimensional sparse filter as a long one-dimensional filter, $u = (u_{m_x(u),:}, u_{m_x(u)+1,:}, \ldots, u_{M_x(u),:})$ as in Figure 35a, will lead to unacceptable work to find neighbors in the y direction. On the contrary, a definition that has an ordering in both x and y directions, as in Figure 35b, leads to difficulties in how to treat filters where there are diagonal blocks involved. The way chosen in this paper is to see a two-dimensional filter as an array of one-dimensional filters, as in Figure 35c. This structure makes the Λ, $\tilde{\Lambda}$-cycles easy to generalize to two dimensions. For example, we have $\mathbf{smear}_{2\mathrm{d}}\, u = \mathbf{smear}_x\, \mathbf{smear}_y\, u$, $\mathbf{sharpen}\, u = \mathbf{sharpen}_x\, \mathbf{sharpen}_y\, u$. The extension to more than two dimensions is immediate.



a b c

Figure 35: *Linked list representations of two-dimensional filters.*

The general two-dimensional PDE we want to solve is

$$\begin{cases} u_t = P(d/dx, d/dy, x, y, t, u), \\ u(x,y,0) = u_0(x,y), \\ u(0,y,t) = u(1,y,t), \\ u(x,0,t) = u(x,1,t), \\ 0 \le x \le 1,\ 0 \le y \le 1,\ t \ge 0, \end{cases} \tag{95}$$

and we make the same assumption as in the one-dimensional case: We can Taylor-expand P and get an approximation that can be evaluated by repeated multiplication and differentiation. We also permit systems, i.e., $u = (u_1, \ldots, u_n)^T$.

5.1 The advection equation

We solve

$$\begin{cases} u_t = u_x + u_y, \\ u(x,0,t) = u(x,1,t), \\ u(0,y,t) = u(1,y,t), \\ u(x,y,0) = u_0(x,y), \\ 0 \le x \le 1,\ 0 \le y \le 1,\ t \ge 0, \end{cases} \tag{96}$$

with initial function $u_0(x,y) = \sin(2\pi x) + \sin(2\pi y) + 3e^{-2500((x-0.5)^2+(y-0.5)^2)}$. The solution to this equation is $u(x,y,t) = u_0((x+t) \bmod 1, (y+t) \bmod 1)$. We choose the β_2 filter bank, the parameters, $n = 9$, $k = 3$, $\Delta t = 10^{-3}$, $\epsilon = 10^{-5}$, and the 4th order centered finite difference scheme. The approximated solution at $t = 1$ is shown in Figure 36, and the error in Figure 37. We have $\|e\|_\infty = 6.1 \times 10^{-3}$. The total number of blocks used are 828 out of 16384 ($n_b = 4$), giving a compression factor of 19.8. These blocks are shown in Figure 38, where we have separated the four components of the transformed signal, e.g., $V \otimes W = \downarrow_y \check{\tilde{h}} *_y \downarrow_x \check{\tilde{g}} *_x (\downarrow_y \check{\tilde{h}} *_y \downarrow_x \check{\tilde{h}} *_x)^{j-1} u$.

We next choose the 'nastier' initial function $u_0(x,y) = \sin(2\pi x) + \sin(2\pi y) + 3e^{-10000((x-0.5)^2+(y-0.5)^2)}$, and solve the PDE with the β_4 filter bank and the following parameters, $n = 10$, $k = 4$, $\Delta t = 5 \times 10^{-4}$, $\epsilon = 10^{-5}$, and the 4th order centered finite difference scheme. The approximated solution at $t = 1$ is shown in Figure 39, and the error in Figure 40. We have $\|e\|_\infty = 0.02488$. The total number of blocks used are 148 out of 16384 ($n_b = 8$), giving a compression factor of 110.7. These blocks are shown in Figure 41.

The 'tail' behind the 'spike' appeared in several of the tests, and is due to the thresholding not killing enough high frequencies. *It usually did not grow as the time increased.* We see that the filter bank method automatically adapts to a grid that is locally refined in both the x and y direction; a problem that becomes 'messy' when dealing with locally refined grids and finite differences. Furthermore, changing order of the method is easily done by changing filter bank.

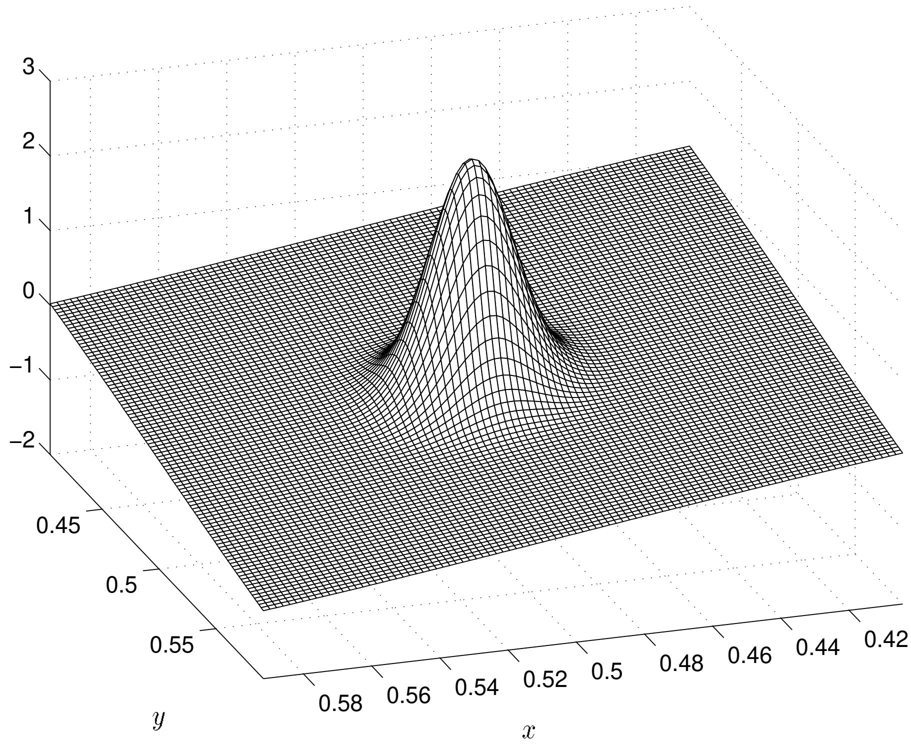


Figure 36: *Approximated solution at $t = 1$ when solving $u_t = u_x + u_y$, with β_2 filter bank.*

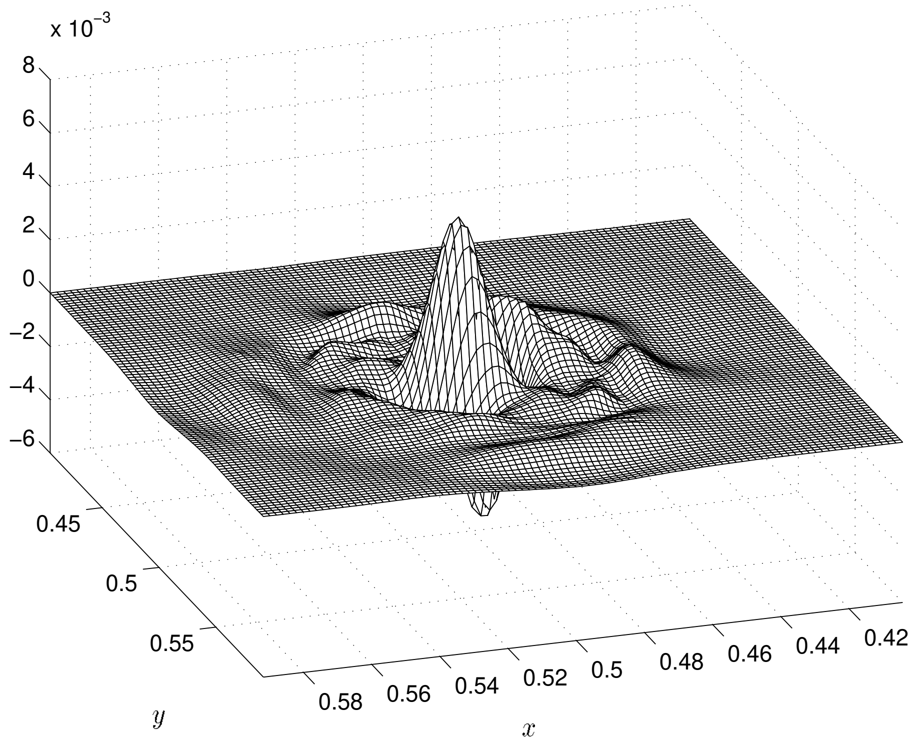


Figure 37: *Error at $t = 1$ when solving $u_t = u_x + u_y$, with β_2 filter bank.*

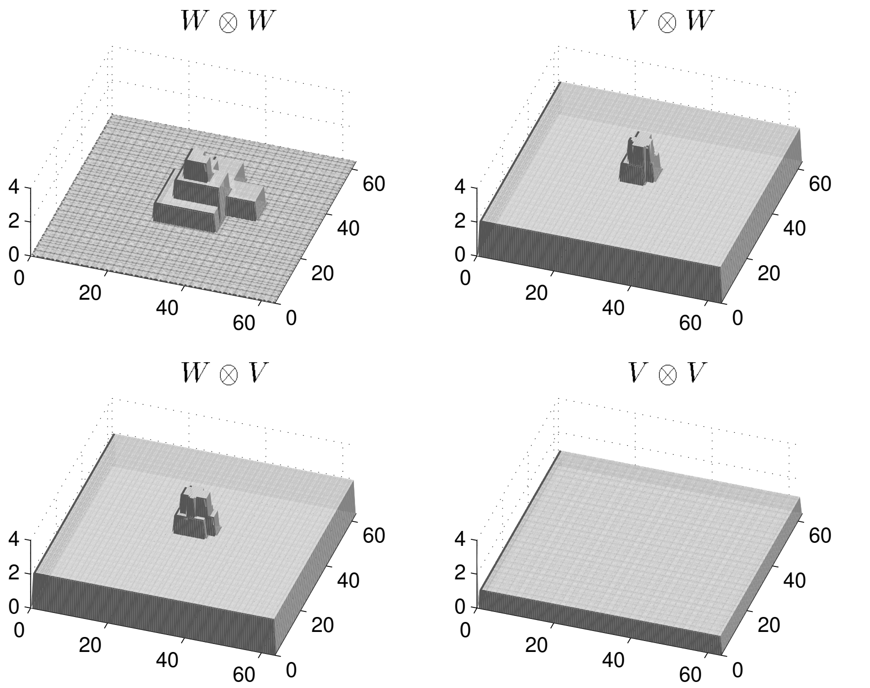


Figure 38: *Sparse representation at $t = 1$ when solving $u_t = u_x + u_y$, with β_2 filter bank.*

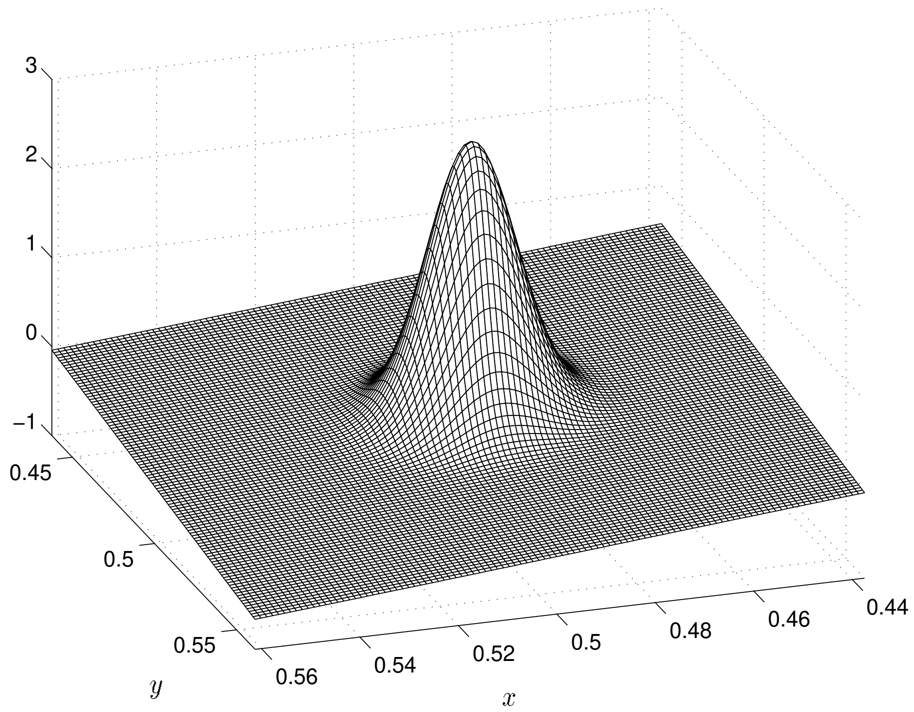


Figure 39: *Approximated solution at $t = 1$ when solving $u_t = u_x + u_y$, with β_4 filter bank.*

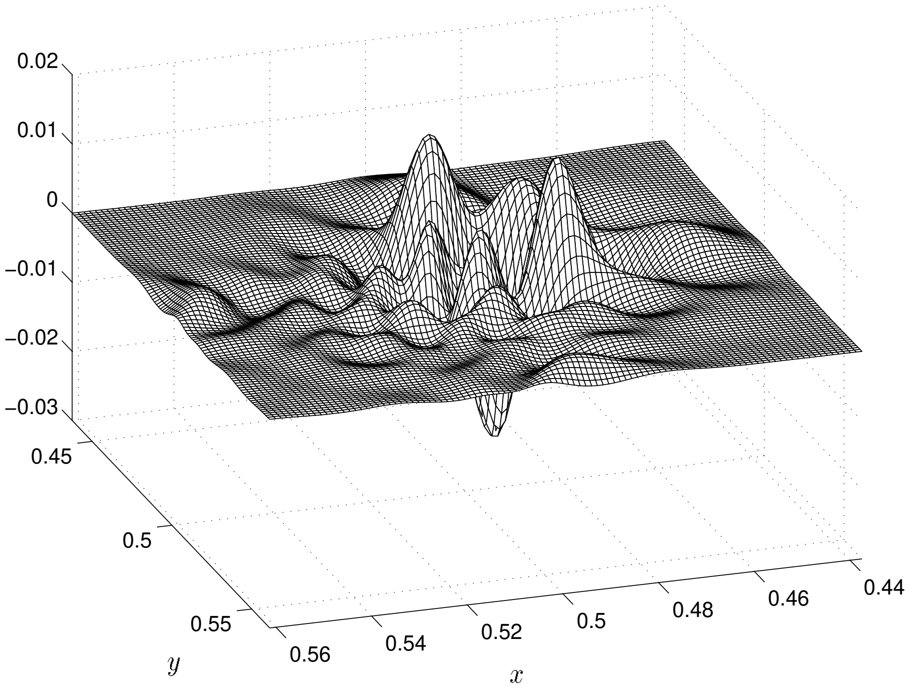


Figure 40: *Error at $t = 1$ when solving $u_t = u_x + u_y$, with β_4 filter bank.*

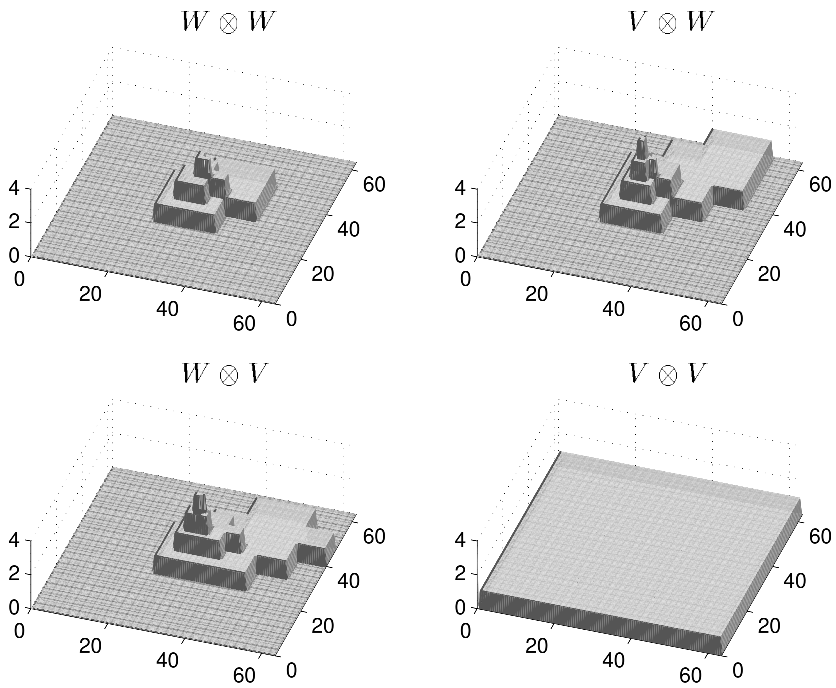


Figure 41: *Sparse representation at $t = 1$ when solving $u_t = u_x + u_y$, with β_4 filter bank.*

5.2 The equation $u_t = 2\pi(y-0.5)u_x - 2\pi(x-0.5)u_y$

The next equation is linear, but with variable coefficients.

$$\begin{cases} u_t = 2\pi(y-0.5)u_x - 2\pi(x-0.5)u_y, \\ u(x,y,0) = e^{-6400((x-0.25)^2+(y-0.5)^2)}, \\ (x,y) \in \mathbb{R}^2,\ t \geq 0. \end{cases} \tag{97}$$

The solution will rotate around (0.5,0.5) with period 1, so $u(x,y,1) = u(x,y,0)$. The β filter banks will be inadequate for this equation because of the low order of accuracy in the multiplication. Therefore, we choose the δ_3 filter bank. We choose the 4th order centered finite difference scheme, $n = 10$, $k = 3$, $\Delta t = 8 \times 10^{-4}$, $\epsilon = 3 \times 10^{-5}$. The approximated solution around the 'spike' at $t = 1$ is shown in Figure 42, and the error in Figure 43. The max-norm of the error is $\|e\|_\infty = 0.00973$. In Figure 44 we see the representation at $t = 1$. We have 57 blocks out of a total of 4096, giving a compression factor of 71.9. As there is no low frequency signal around the 'spike' there are regions where there is no grid at all (shown in the $V \otimes V$ part of Figure 44).

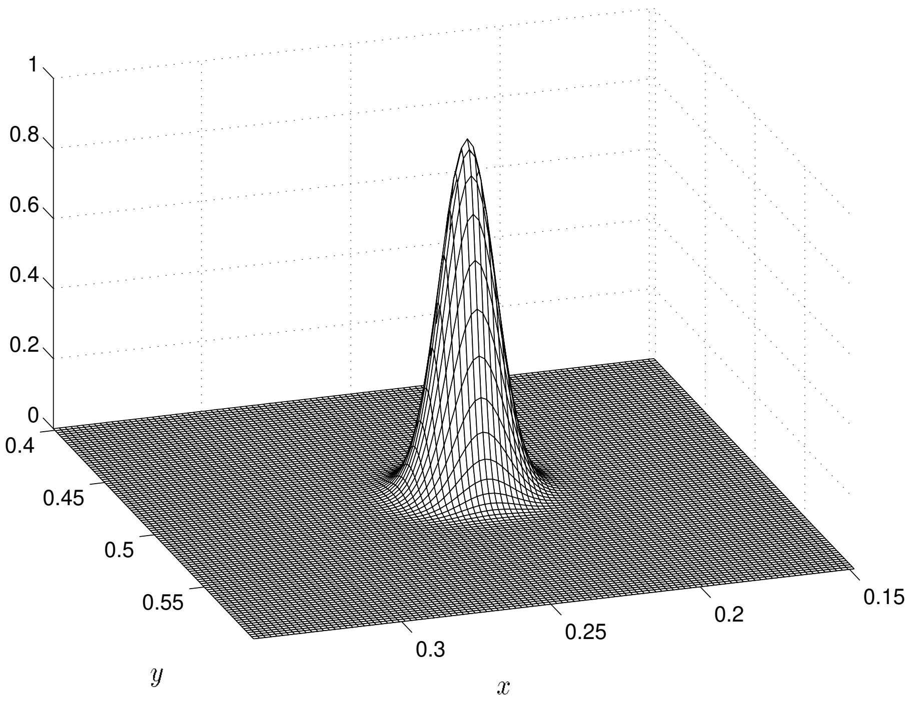


Figure 42: *Approximated solution at $t = 1$ when solving $u_t = 2\pi(y-0.5)u_x - 2\pi(x-0.5)u_y$, with δ_3 filter bank.*

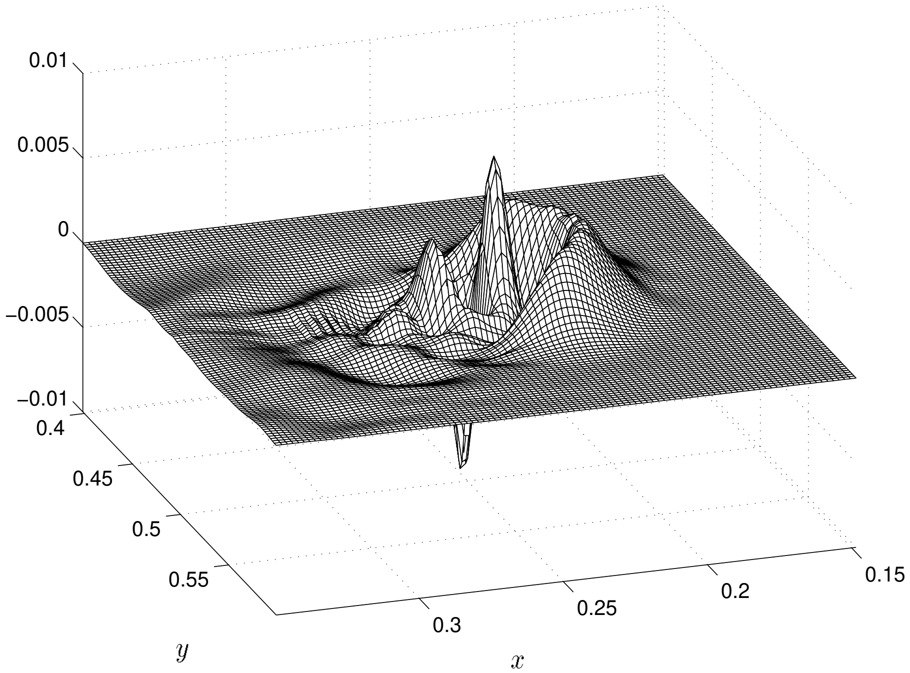


Figure 43: *Error at $t = 1$ when solving $u_t = 2\pi(y - 0.5)u_x - 2\pi(x - 0.5)u_y$, with δ_3 filter bank.*

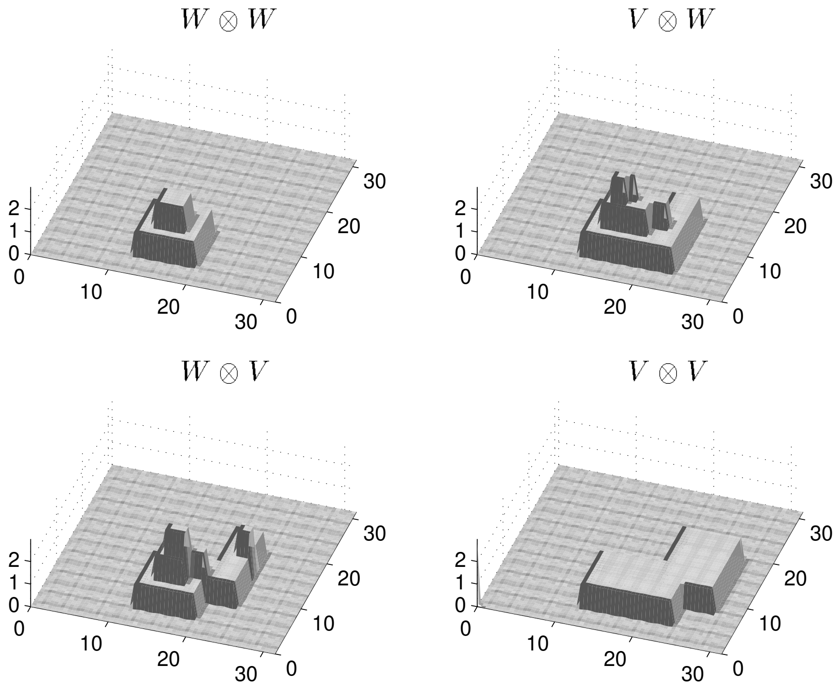


Figure 44: *Sparse representation at $t = 1$ when solving $u_t = 2\pi(y - 0.5)u_x - 2\pi(x - 0.5)u_y$, with δ_3 filter bank.*

5.3 The bi-dimensional Burgers equation

Our last example is the bi-dimensional Burgers equation

$$\begin{cases} u_t = -(\nabla u)u + \mu\triangle u, \\ u(x,0,t) = u(x,1,t), \\ u(0,y,t) = u(1,y,t), \\ u(x,y,0) = (\sin(2\pi(x+y)), \sin(2\pi(x+y)))^T, \\ 0 \le x \le 1,\ 0 \le y \le 1,\ t \ge 0, \end{cases} \tag{98}$$

where $u = (u_1, u_2)^T$. This equation was studied in [27]. With these initial conditions, the solution is a 45° rotation of the solution to the one-dimensional equation,

$$u(x,y,t) = (u_{1\mathrm{d}}(x+y, 2t), u_{1\mathrm{d}}(x+y, 2t))^T, \tag{99}$$

but, because of the rotation, the equation is fully bi-dimensional. The initial condition is shown in Figure 45, and the solution at $t = 0.1$ for $\mu = 10^{-2}$ is shown in Figure 46.

We choose $\mu = 5 \times 10^{-4}$, $n = 11$, $k = 5$, $\epsilon = 8 \times 10^{-5}$, the δ_3 filter bank, the 4th order finite difference schemes, and the time step $\Delta t = 10^{-4}$. The approximated solution near $(x,y) = (0.25, 0.25)$, at $t = 0.2$ is shown in Figure 47, and the representation is shown in Figure 48. In Figure 49 we see the approximated solution around (0.25,0.25) in the diagonal direction, $(x,y) = \gamma(\alpha) = \alpha(1,1)$, compared with the exact solution. The error along this path is $\left\| e|_\gamma \right\|_\infty = 6.07 \times 10^{-4}$. This improvement compared with the one-dimensional case comes from the use of the δ-filter bank. The number of blocks are 2296 out of a total of 65536 ($n_{\mathrm{b}} = 8$) giving a compression factor of 28.5. Thus, the filter bank method, as in the one-dimensional case, works very well for this non-linear problem.

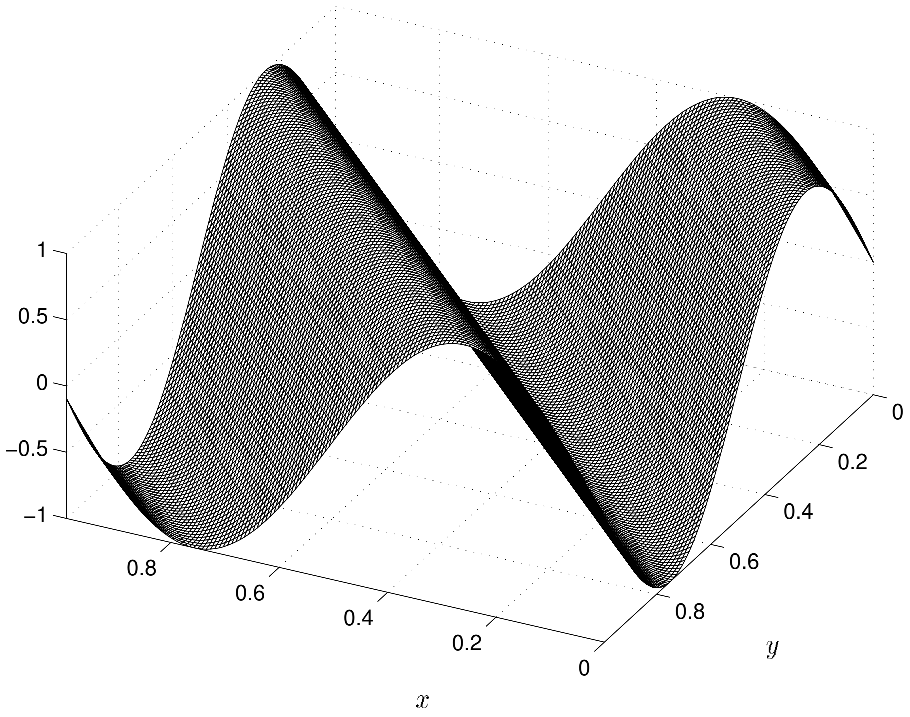


Figure 45: *Initial condition for bi-dimensional Burgers equation.*

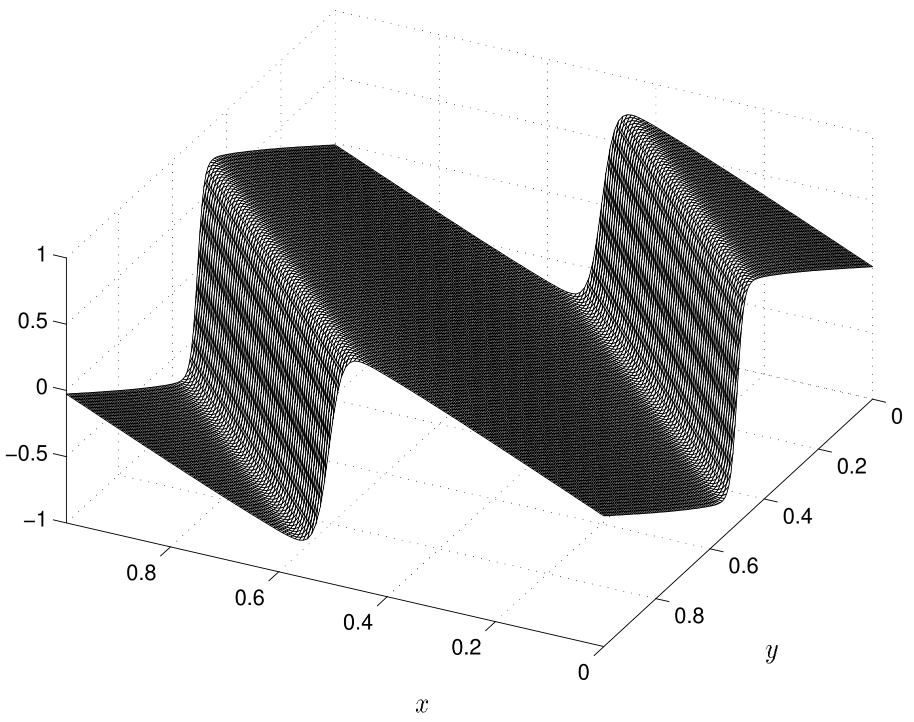


Figure 46: *Solution at $t = 0.1$ for bi-dimensional Burgers equation, with $\mu = 10^{-2}$.*

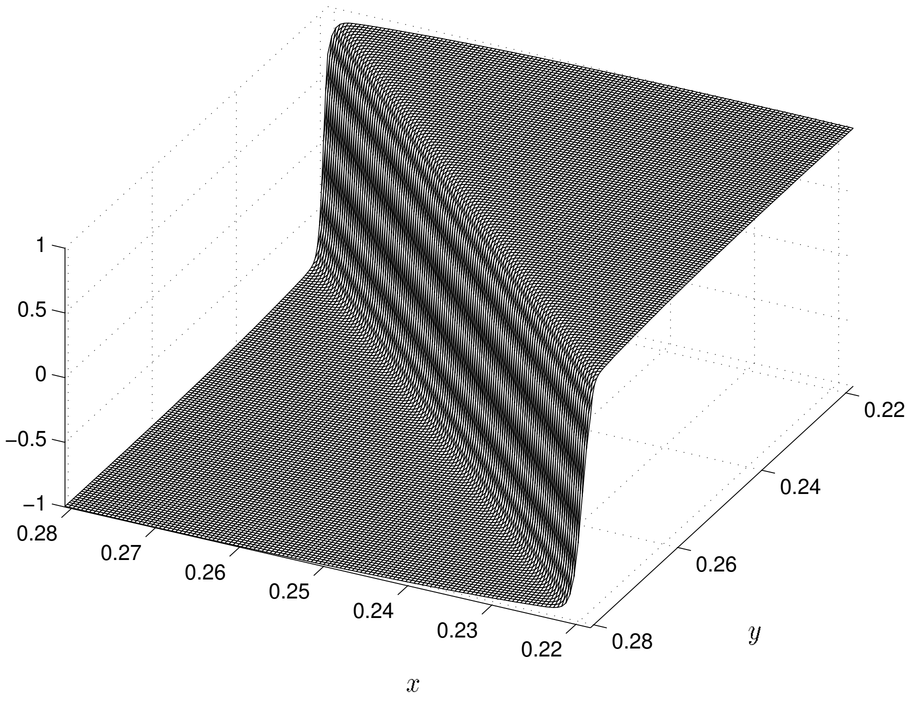


Figure 47: *Approximated solution at $t = 0.1$ for bi-dimensional Burgers equation around (0.25,0.25), with $\mu = 5 \times 10^{-4}$, $\epsilon = 8 \times 10^{-5}$ when solved with δ_3 filter bank, $\tilde{\Lambda}$-cycle.*

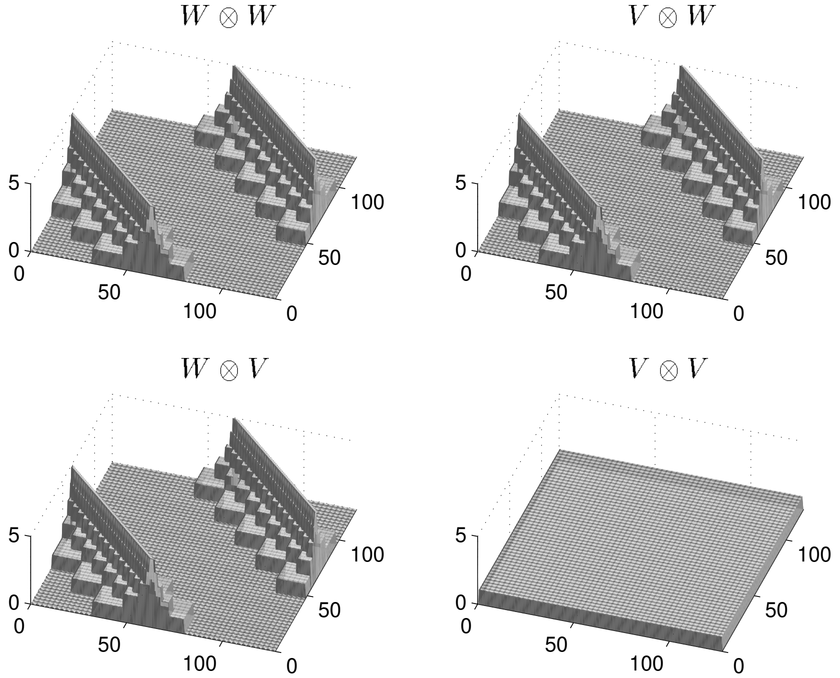


Figure 48: *Sparse representation at $t = 0.1$ when solving the bi-dimensional Burgers equation, with δ_3 filter bank, $\tilde{\Lambda}$-cycle, viscosity $\mu = 5 \times 10^{-4}$, $\epsilon = 8 \times 10^{-5}$.*

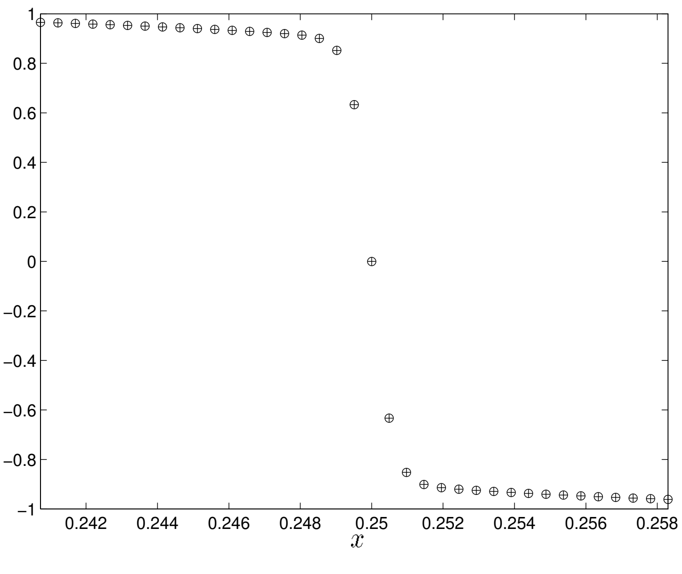


Figure 49: *Approximated solution 'o', and exact solution '+', near $(0.25, 0.25)$ along path $(x, y) = \alpha(1, 1)$, at $t = 0.1$, when solving the bi-dimensional Burgers equation with viscosity $\mu = 5 \times 10^{-4}$ using the $\tilde{\Lambda}$-cycle, filter bank δ_3, and $\epsilon = 8 \times 10^{-5}$.*

6 Conclusions

We have developed a method for solving hyperbolic initial-value problems in an adaptive multi-level manner. By working with perfect reconstruction filter banks, and sampled values of the function, we reduced the length of the filters involved substantially, and decreased the error in the method. The filters used in this paper, the β and δ filter banks, do not correspond to any wavelets in L^2, and the modest price paid is extra growth factors in the error, and number of coefficients.

The filter bank method provides simple rules of how to use locally refined grids in an arbitrary number of dimensions. The only parameter needed is the truncation, ϵ. Furthermore, changing order of the method is easy by switching filter banks.

The Λ, and $\tilde{\Lambda}$-cycles could be used for other local operators than multiplication and differentiation, although, the filter banks used might have to be tailored to satisfy some extra conditions (as was the case when switching from β- to δ-filter banks for multiplication.

The cycles could also be modified to allow adaptive time steps as in [1], although, for non-linear problems, it is an open question whether the increased complexity in the algorithm is out-weighted by the larger time steps.

The compression factors of $13 - 110$ in the test problems make the method highly interesting for large scale problems where the $\tilde{\Lambda}$-cycle seems robust enough to handle a large number of different situations. Adding boundary conditions should impose no major difficulties, due to the attractive combination of using short filters, and working partly in the physical domain.

References

[1] 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.

[2] T.P. Barnwell and M. J. T. Smith. Exact reconstruction techniques for tree-structured subband coders. *IEEE Trans. ASSP*, 34:434–441, 1986.

[3] 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.

[4] A. Cohen. Ondelettes, analyses multirésolutions et filtres miroir en quadrature. *Ann. Inst. H. Poincaré, Anal. non linéeare*, 7:439–459, 1990.

[5] A. Cohen. Biorthogonal wavelets. In *Wavelet analysis and its applications- Wavelets- A tutorial in theory and applications*, volume volume 2, pages 123–152. Academic Press, 1992.

[6] A. Cohen, I. Daubechies, and J. C. Feaveu. Biorthogonal bases of compactly supported wavelets. *Comm. Pure Appl. Math.*, 45:485–560, 1992.

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

[8] I. Daubechies. Orthonormal bases of compactly supported wavelets. *Comm. Pure Appl. Math.*, 41:909–996, 1988.

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

[10] I. Daubechies. Orthonormal bases of compactly supported wavelets II. Variations on a theme. *SIAM J. Math. Anal.*, 24(2):499–519, 1993.

[11] G. Deslauriers and S. Dubuc. Symmetric iterative interpolation processes. *Constr. Approx.*, 5:49–68, 1989.

[12] M. Dorobantu. Potential integral equations of the 2d Laplace operator in wavelet bases. Technical Report TRITA-NA-9401, NADA, KTH, S-10044 Stockholm, Sweden, 1994.

[13] M. Dorobantu. *Wavelet-Based Algorithms for Fast PDE Solvers.* PhD thesis, Royal Institute of Techonology, NADA, KTH, S-10044 Stockholm, Sweden, 1995.

[14] T. Eirola. Sobolev characterization of solutions of dilation equations. *SIAM J. Math. Anal.*, 23(4):1015–1030, 1992.

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

[16] J. Fröhlich and K. Shneider. *An adaptive wavelet-vaguelette algorithm for the solution of nonlinear PDEs.* Konrad–Zuse–Zentrum Berlin, Heilbronner Str. 10, D-10711 Berlin Wilmersdorf, Germany. preprint.

[17] M. Holmström and J. Waldén. Adaptive wavelet methods for hyperbolic PDEs. Technical Report 171, Dept. of Scientific Computing, Uppsala University, Box 120, Uppsala, Sweden, 1995.

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

[19] L. Jameson. On the wavelet optimized finite difference method. Technical Report 94-9, NASA, Langley Research Center, Hampton, VA 23681-0001, USA, 1994.

[20] L. Jameson. A wavelet-optimized, very high order adaptive grid and order numerical method. Technical Report ICASE, 96-30, NASA, Langley Research Center, Hampton, VA 23681-0001, USA, 1996.

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

[22] W. M. Lawton. Tight frames of compactly supported affine wavelets. *J. Math. Phys.*, 31(8):1899–1901, August 1990.

[23] W. M. Lawton. Necessary and sufficient conditions for constructing orthonormal wavelet bases. *J. Math. Phys.*, 32(1):57–61, January 1991.

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

[25] Y. Meyer. *Wavelets and operators.* Cambridge University Press, 1992.

[26] F. Mintzer. Filters for distortion-free two-band multirate filter banks. *IEEE Trans. Acoust. Speech Signal Process.*, 33:626–630, 1985.

[27] Pj. Ponenti and J. Liandrat. Numerical algorithms based on biorthogonal wavelets. Technical Report ICASE 96-13, NASA, Langley Research Center, Hampton, VA 23681-0001, USA, 1996.

[28] W. Sweldens and R. Piessens. Asymptotic error expansion of wavelet approximations of smooth functions II. *Numer. math.*, 68:377–401, 1994.

[29] 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.

[30] O. V. Vasilyev and Samuel Paolucci. A dynamically adaptive multilevel wavelet collocation method for solving partial differential equations in a finite domain. *J. Comput. Phys.*, 125:498–512, 1996.

[31] O. V. Vasilyev, Samuel Paolucci, and Mihir Sen. A multilevel wavelet collocation method for solving partial differential equations in a finite domain. *J. Comput. Phys.*, 120:33–47, 1995.