Comparison of Wavelets for Adaptive Mesh Refinement

J Knipping[1], J du Toit[2], and C Vuik[1]

[1]Department of Applied Mathematics, Delft University of Technology, The Netherlands
[2]Numerical Algorithms Group Ltd, England

February 5, 2020

Abstract

In the field of Scientific Computing there is a big focus on solving time dependent Partial Differential Equations (PDEs) as efficiently and fast as possible. In order to do so, the PDE is discretized and solved on a mesh at every time step. Adaptive mesh refinement is used to develop a mesh at every time step which is sparse and which results in an accurate approximation to the solution.

Interpolating wavelets are successfully used in adaptive mesh refinement (AMR). A detailed comparison of two wavelets for AMR is done on different data sets: Donoho's interpolating wavelet and a lifted version (also called second generation wavelets) of Donoho's interpolating wavelet. Moreover, various ways of handling the boundaries are considered. An algorithm to construct the meshes using wavelets is tested and optimized.

Donoho's interpolating wavelet with lower order boundary stencil implementation appears to be the most accurate, whilst resulting in very high compression compared to the original mesh. Furthermore, adapting the algorithm which constructs the meshes such that it adds more points for very irregular shapes, turns out to be valuable for solutions with fast changing features. For one such PDE, Donoho's interpolating wavelet keeps less than 5% of the points whilst having an error smaller than 10^{-4}, in other words a sparsification of 20 times. Lastly, an improvement on the inverse transform during the adaptive mesh refinement leads to promising results.

1 Introduction

Over the last three decades there have been many developments in wavelet theory. Wavelets can be used for compression of data and allow for fast interpolation. Therefore, wavelets have a lot of applications in different fields, the most famous one being image compression. In the field of scientific computing wavelets also have multiple uses. For the numerical solution of PDEs, the two main ones are *adaptive wavelet Galerkin methods* and *adaptive wavelet collocation methods*. In the Galerkin methods wavelets are used as basis functions to directly discretize partial differential equations (PDEs). In the latter method the properties of the wavelets are used to find a sparse mesh on which the PDEs will be solved.

In this paper the focus will be on adaptive wavelet collocation methods. Because wavelets are very good in compressing data it makes sense to use this property for adaptive mesh refinement. Many wavelets have been developed. Typically, the wavelets are designed in such a manner that the transformation of a data set or function can be calculated quickly. This is important for adaptive mesh refinement so that the mesh can be calculated without a large overhead. Moreover, because wavelets are good at compressing data they can be used to find a highly sparse mesh.

Cohen, Daubechies and Feauveau laid the foundation of the first generation orthogonal and biorthogonal wavelets, [1, 2, 3]. Donoho introduced interpolating wavelets in [4]. Interpolating wavelets can be used for adaptive mesh refinement. Sweldens introduced a faster and more intuitive manner to transform functions in their wavelet representation, the lifting scheme, [5, 6, 7, 8]. This new transform resulted in second generation wavelets, [9]. Due to the lifting scheme a new version of

Donoho's interpolating wavelet was introduced, the lifted Donoho's interpolating wavelet, [10]. This version of Donoho's interpolating wavelet tends to improve the ability of the wavelet to represent functions. This is done by taking a closer look at vanishing moments, [11, 12]. Because meshes are defined on a finite domain, one should take special care at the boundaries of the domain. Therefore, two different boundary stencil implementations will be tested, the interpolating boundary stencil and the lower order boundary stencil.

Algorithms for wavelets based on adaptive mesh refinement were introduced in [13] and [14]. Vasilyev and Bowman combined these methods to introduce an adaptive mesh refinement (AMR) for non-equidistant meshes and second generation wavelets, [15]. The algorithm can be changed in the way it thresholds, [16], or in the way it adapts, i.e., how adjacent points are added. Applying AMR on non-equidistant meshes can lead to stability issues, which are related to the update step, boundary stencil implementation, and lack of orthogonality, [9, 17]. In this paper equidistant meshes are tested, however considering the stability issues of non-equidistant meshes also turned out to be valuable for the equidistant case.

Comparisons between the performance of different wavelets for AMR are scarce. Much research has been done in collocation methods and applying wavelet based AMR to various kind of PDEs. Wirasaet researched Donoho's interpolating wavelet with the interpolating boundary stencil for AMR, [18]. He considered two types of adjacent points and examined both equidistant and non-equidistant meshes for 1D and 2D problems. The mesh refinement algorithm is similar to [15] and the PDEs are solved using finite differences. This collocation method has been tested on different problems, [19, 20, 21].

A similar collocation method which focuses on PDEs arising in mathematical finance has been tested in [22, 23]. This method does not use the lifting scheme for wavelet transformation. It uses Donoho's interpolating wavelet, the interpolating boundary stencil, and the mesh refinement of [15] together with finite differences to approximate the derivatives. The method is tested on equidistant meshes.

Vasilyev and others have designed a similar collocation method, [15, 24, 25, 26, 27, 28]. The only difference is that the lifted Donoho's interpolating wavelet is used. Tests have been done on various PDEs, mainly 1D problems but also in later papers 2D and 3D problems on equidistant and non-equidistant meshes. In [29] the possibility to use multigrid in conjunction with wavelet AMR is investigated.

In the literature both Donoho's interpolating wavelet and the lifted Donoho's interpolating wavelet have been used for AMR. However, the two wavelets have not been directly compared. Because it is not clear which wavelet is favoured, this paper compares both wavelets on various data sets in 1, 2 and 3 dimensions. Furthermore, in the literature the interpolating boundary stencil is always used. In this paper the lower order boundary stencil is compared with the interpolating boundary stencil, since [9] and [17] suggest that the behaviour of this boundary stencil implementation is more stable. Indeed, we observe significant differences in performance between the two boundary stencil implementations, and there is no clear winner, it very much depends on which wavelet is being used.

When comparing the different wavelets and boundary stencils it is interesting to investigate the performance of the wavelet: can a sparse mesh be generated for every function, does the sparse mesh generator have stable behaviour, how local are perturbations in the sparse mesh, what is the sparsity rate of the generated meshes, and finally, how does the sparse solution extrapolate to the whole space.

This paper tests wavelets on data sets generated by time dependent PDEs. The performance of the wavelets and the adaptive mesh refinement (AMR) algorithm are checked by generating sparse meshes on various data sets.

Each data set consists of the numerical solution of a PDE on a regular dense grid at a sequence of time steps. Therefore AMR can be applied every time step and the performance of the adapted meshes can be compared against a (dense) reference solution. Two different classes of problems are considered. The first is from finance and contains 1D and 2D data. The second is from a compressible Navier–Stokes problem where energy, density, velocity and pressure are modelled through a coupled set of equations. We look at 2D and 3D versions of this problem.

We emphasise that in the present work, we *do not* solve PDEs on adaptive sparse meshes. Our goal in this paper is to study only the abilities of different wavelets to generate adaptive

sparse meshes which can accurately reconstruct surfaces which are changing in time. Building a PDE solver on an (irregular) sparse mesh is in itself a challenging topic which is not addressed here. However, we believe there is value in decoupling the wavelet based mesh refinement from the PDE solution and studying this in isolation. A wavelet AMR based PDE solver must work with the mesh the wavelet gives it. It is important therefore to see whether the different wavelets are equally good at creating sparse meshes which can accurately track a time-varying surface. Our study therefore *begins* with sets of dense reference solutions at sequences of time points and we then consider how well the different wavelet AMRs can track these surfaces. In some sense this represents a "best case" scenario for a wavelet AMR based PDE solver: it assumes that at each time step, the PDE solver on the adaptive sparse mesh can produce a solution which is as accurate as a reference solution on the dense mesh. This is unlikely to be the case in practice. For a given wavelet, our results therefore represent an upper bound on accuracy that can be achieved for a given level of sparsity. Accuracy here refers to what happens when the solution on the sparse mesh is extrapolated to the dense mesh and compared with the reference solution there.

In Section 2, the theory behind wavelets will be discussed. In this section both Donoho's interpolating wavelet and the lifted Donoho's wavelet will be introduced. Subsequently, in Section 3 the adaptive mesh refinement algorithm is stated. In Section 4, the tests will be explained and the results will be analysed and evaluated. Finally, Section 5 contains the conclusions.

2 Wavelet Theory

Wavelets can be used to represent data sets or functions. In this sense they could be compared to the Fourier transform. However, the Fourier transform represents functions with finite support as functions with infinite support, whereas wavelets preserve the finite support. This is due to the localisation both in space and frequency of wavelets. Therefore, they have a big advantage compared to the well known Fourier transform, especially in the sense of compression. A wavelet transform means that a function will be written as a sum of localised wavelets. This is typically done in a setting of a family of nested subsets, which are related to each other by dyadic scaling:

$$\{0\} \subset \ldots \subset V_{j-1} \subset V_j \subset V_{j+1} \subset \ldots \subset L^2(\mathbb{R}) \qquad j \in \mathbb{Z}.$$

Each of the spaces V_j can be generated by a Riesz basis of scaling functions, $\varphi_{j,k}(x)$ with $k \in \mathbb{Z}$. The complement of V_j in V_{j+1} is called the detail space W_j ($V_j \oplus W_j = V_{j+1}$). The detail spaces W_j are generated by a Riesz basis of wavelet functions, $\psi_{j,m}$ with $m \in \mathbb{Z}$. This family of subsets is called multiresolution analysis (MRA). More on MRA can be found in [3, 30].

The sums used in the MRA are infinite. However, in order to implement the wavelets finite sums are needed. Typically densest and coarsest levels are fixed and respectively called J_2 and J_1 with J_2, J_1 integers satisfying $J_2 > J_1$. V_{J_2} is considered to be a fine approximation space, it is assumed no finer space exists. In order to approximate a function $f \in L^2$ it should be projected to the densest space V_{J_2}. Due to the properties of the detail space, this space can be written as $V_{J_2} = V_{J_1} \oplus \sum_{j=J_1}^{J_2-1} W_j$, which in turn leads to:

$$f(x) \approx P_{V_{J_2}} f(x) = P_{V_{J_1}} f(x) + \sum_{j=J_1}^{J_2-1} P_{W_j} f(x),$$

where $P_U g(x)$ is the projection of function g onto the space U. Because $\varphi_{j,k}(x)$ form a Riesz basis of V_j, the projection on V_j is defined as:

$$P_{V_j} f(x) = \sum_{k\in\mathbb{Z}} \langle f, \varphi_{j,k}\rangle \varphi_{j,k} = \sum_{k\in\mathbb{Z}} s_{j,k}\varphi_{j,k},$$

where $s_{j,k}$ are called the smoothing coefficients. Similarly,

$$P_{W_j} f(x) = \sum_{m\in\mathbb{Z}} \langle f, \psi_{j,m}\rangle \psi_{j,m} = \sum_{m\in\mathbb{Z}} d_{j,m}\psi_{j,m},$$

where $d_{j,m}$ are called the detail coefficients. This leads to an approximation of $f \in L^2$ on the finest space V_{J_2}:

$$f(x) \approx P_{V_{J_2}} f(x) = \sum_{k \in \mathbb{Z}} s_{J_1,k} \varphi_{J_1,k}(x) + \sum_{j=J_1}^{J_2-1} \sum_{m \in \mathbb{Z}} d_{j,m} \psi_{j,m}(x).$$

In order to determine any wavelet representation, the coefficients $s_{J_1,k}$ and $d_{j,m}$, for $J_1 \leq j < J_2$ and $k, m \in \mathbb{Z}$, are needed. They can be determined by the forward transform. Given the smoothing coefficients at the densest level, $s_{J_2,k}$, the smoothing coefficients and detail coefficients of the lower levels can be determined using the forward (wavelet) transform. If the smoothing coefficients at the coarsest level and the detail coefficients of all levels are known, the inverse (wavelet) transform can be used to determine the smoothing coefficients at the densest level.

The lifting scheme The forward transform used is based on the lifting scheme, see [7]. Firstly, a set of points is divided into a set of smoothing points and detail points. This step is called the split step. Next, the values at the detail points are predicted using the smoothing coefficients: the predict step. Then the values of the smoothing points are updated using the neighbouring detail points. This is called the update step. Usually the split step consists of dividing the set of points into two equally sized sets, where every smoothing point has neighbouring detail points and every detail point has smoothing points as neighbours. The predict step predicts the detail coefficients by using a set of smoothing coefficients. The predicted value of the detail coefficient is then subtracted from the original value. As such, the resulting detail coefficient directly gives feedback on how well the original value can be predicted from surrounding values. Therefore, in the threshold phase the detail coefficients with a value close to zero will be removed. The predict step looks as follows:

$$d_{j,m} = d_{j,m} - \sum_{k \in \mathbb{Z}} p_k s_{j,m+k},$$

where p_m are the predict coefficients, contained in the predict filter $P = \{p_k\}_{k \in \mathbb{Z}}$. The update step usually tends to orthogonalize the wavelets, or to increase the dual vanishing moments. This is done by adding the predicted detail coefficients to the smoothing coefficients:

$$s_{j,k} = s_{j,k} + \sum_{m \in \mathbb{Z}} u_m d_{j,k+m},$$

where u_m are the update coefficients, contained in the update filter $U = \{u_m\}_{m \in \mathbb{Z}}$. One can retrieve the smoothing coefficients at level J_1 and the detail coefficients at every level with the forward transform. The split step will be repeated on the smoothing coefficients which resulted from the update step. On the new sets of smoothing and detail coefficients, the predict and update steps are performed. This process is repeated until level J_1 is reached and is called the forward transform.

To get all the values of the smoothing coefficients at level J_2, the process described above can be performed in the inverse order. Furthermore, the update step should be subtracted and the predict step added. This results in the inverse transform.

Donoho's interpolating wavelet Donoho's interpolating wavelet will be presented in the setting of the lifting scheme, for more details see [4]. The smoothing coefficients at level J_2 are the function values at the locations of the coefficients. Predicting the detail coefficients will be done through polynomial interpolation. A polynomial is fitted on the N surrounding smoothing coefficients, the value of this polynomial at the location of the detail coefficient is the predicted value. Because the prediction filter has N coefficients, Donoho's wavelets are always able to perfectly represent polynomials up to order $N-1$. This means that after the forward transform the detail coefficients will all be 0. The predict coefficients p_k^N are given in Table 1. Because the coefficient sum up to $\frac{1}{2}$, the detail coefficient should be multiplied by a half before the predict step.

Donoho's interpolating wavelet does not have an update step, hence the smoothing coefficients will be scaled original function values. The scaling comes from multiplying all the coefficients with $\sqrt{2}$ at every level, for more details see [3].

Table 1: The predict filter P^N.

$N=2$		$N=4$		$N=6$	
k	p_k^2	k	p_k^4	k	p_k^6
0	$1{\cdot}2^{-2}$	-2	$-1{\cdot}2^{-5}$	-4	$3{\cdot}2^{-9}$
1	0	-1	0	-3	0
2	$1{\cdot}2^{-2}$	0	$9{\cdot}2^{-5}$	-2	$-25{\cdot}2^{-9}$
		1	0	-1	0
		2	$9{\cdot}2^{-5}$	0	$150{\cdot}2^{-9}$
		3	0	1	0
		4	$-1{\cdot}2^{-5}$	2	$150{\cdot}2^{-9}$
				3	0
				4	$-25{\cdot}2^{-9}$
				5	0
				6	$3{\cdot}2^{-9}$

In short, to determine the smoothing and detail coefficients of Donoho's interpolating wavelet, the smoothing coefficients at level J_2 are set equal to the function values. Thereafter, the detail coefficients are calculated by performing the forward transform as described in Algorithm 1.

Algorithm 1 Forward transform Donoho's interpolating wavelet

for j=J_2 downto J_1+1 **do**
  $[s_{j-1}, d_{j-1}] = \text{split}(s_j)$
  for $m \in \mathbb{Z}$ **do**
    $d_{j-1,m} = \frac{1}{2} d_{j-1,m} - \sum_{k=-N}^{N} p_k^N s_{j-1,m+k}$
  end for
end for

Vanishing moments The order of the wavelet (N) and the dual order ($\tilde{N}$) are also called the primal and dual vanishing moments. The orders can be determined by considering the scaling functions (φ) and wavelet functions (ψ) as defined at the beginning of this section. A wavelet system has N primal vanishing moments if

$$\int_{-\infty}^{\infty} x^n \varphi(x) dx = 0 \qquad \text{for } 0 < n < N.$$

Similarly, a wavelet system has $\tilde{N}$ dual vanishing moments if

$$\int_{-\infty}^{\infty} x^n \psi(x) dx = 0 \qquad \text{for } 0 < n < \tilde{N}.$$

Donoho's interpolating wavelet has N primal vanishing moments, however it has 0 dual vanishing moments. This leads to aliasing: local behaviour on a fine level can be carried through all other layers and lead to significantly big coefficients on the coarsest level. In other words, the wavelet cannot isolate a signal at a particular frequency, but passes some of this signal through to lower frequency filters. This is discussed in more detail in [15]. In order to reduce aliasing one can increase the dual vanishing moments $\tilde{N}$. This can be done by adding an update step.

Lifted Donoho's interpolating wavelet In [10] Sweldens introduced an updated version of Donoho's interpolating wavelet, where the dual vanishing moments $\tilde{N}$ can be steered in the initial phase by adding an update step of order $\tilde{N}$. The update step works as a smoother. In [12], it is suggested that for optimal results in accuracy the primal and dual vanishing moments should be equal. The values of the update filter is actually a shifted version of the predict filter, see [30]. Hence the Table 1 can be consulted for the filter values. The forward transform of the lifted Donoho's interpolating wavelet is given in Algorithm 2. For a detailed explanation of this algorithm see [30].

Algorithm 2 Forward transform lifted Donoho's interpolating wavelet

for j=J_2 downto J_1+1 **do**
  $[s_{j-1}, d_{j-1}] = \text{split}(s_j)$
  for $m \in \mathbb{Z}$ **do**
    $d_{j-1,m} = \frac{1}{2} d_{j-1,m} - \sum_{k=-N}^{N} p_k^N s_{j-1,m+k}$
  end for
  for $k \in \mathbb{Z}$ **do**
    $s_{j-1,k} = s_{j-1,k} + 2 \sum_{m=-\tilde{N}}^{\tilde{N}} p_{-m}^{\tilde{N}} d_{j-1,k+m}$
  end for
end for

Table 2: The left interpolating boundary stencil predict filter $P^{L,N}$.

$N=4$		$N=6$		
k	d_0	k	d_0	d_1
0	$5{\cdot}2^{-5}$	0	$63{\cdot}2^{-9}$	$-7{\cdot}2^{-9}$
1	$15{\cdot}2^{-5}$	1	$315{\cdot}2^{-9}$	$105{\cdot}2^{-9}$
2	$-5{\cdot}2^{-5}$	2	$-105{\cdot}2^{-8}$	$105{\cdot}2^{-8}$
3	$1{\cdot}2^{-5}$	3	$63{\cdot}2^{-8}$	$-35{\cdot}2^{-8}$
		4	$-45{\cdot}2^{-9}$	$21{\cdot}2^{-9}$
		5	$7{\cdot}2^{-9}$	$-3{\cdot}2^{-9}$

Boundary treatment If the wavelets are implemented, one typically uses a finite domain. If a predict or update step is performed near the boundary of the domain, it will need points outside the domain which are not defined. There are several ways to deal with the boundary. Two methods will be discussed here. Note that for the lifted Donoho's interpolating wavelet both the smoothing coefficients and detail coefficients have boundary issues, whereas Donoho's interpolating wavelet only needs a boundary stencil implementation for the detail coefficients. Because the predict and update filters are similar, the cases can be handled similarly.

Interpolating boundary stencil Both the predict and update step are based on fitting a polynomial on N points. One manner to treat the boundary is to still use N smoothing coefficients closest to the boundary and fit a polynomial on these points. However, because there are not enough points on one side of the detail coefficient, the filter will not be symmetric. This lead to special boundary filter coefficients. The boundary predict step can be written as a filter, similar to the predict filter. This is done by using the Lagrange polynomial, for more details see [30].

The predict filters at the boundary for $N = 2, 4, 6$ are given in the Tables 2 and 3. The filter coefficients are large compared to the coefficients before (see Table 1). This means that small values near the boundary can have significant effects. At the boundary, polynomials of order $N-1$ are fitted, hence the wavelet is able to reconstruct polynomials up to order $N-1$ near the boundary. However, for data which is not locally a polynomial of order $N-1$, it is quite likely that the boundary polynomial will be a poor approximation to the data since the polynomial is likely to oscillate, especially for large N. Hence these polynomials may not be good predictors.

Table 3: The right interpolating boundary stencil predict filter $P^{R,N}$.

$N=2$		$N=4$			$N=6$			
k	d_m	k	d_{m-1}	d_m	k	d_{m-2}	d_{m-1}	d_m
$m-1$	$-\frac{1}{4}$	$m-3$	$1{\cdot}2^{-5}$	$-5{\cdot}2^{-5}$	$m-5$	$-3{\cdot}2^{-9}$	$7{\cdot}2^{-9}$	$-63{\cdot}2^{-9}$
m	$\frac{3}{4}$	$m-2$	$-5{\cdot}2^{-5}$	$21{\cdot}2^{-5}$	$m-4$	$21{\cdot}2^{-9}$	$-45{\cdot}2^{-9}$	$385{\cdot}2^{-9}$
		$m-1$	$15{\cdot}2^{-5}$	$-35{\cdot}2^{-5}$	$m-3$	$-35{\cdot}2^{-8}$	$63{\cdot}2^{-8}$	$-495{\cdot}2^{-8}$
		m	$5{\cdot}2^{-5}$	$35{\cdot}2^{-5}$	$m-2$	$105{\cdot}2^{-9}$	$-105{\cdot}2^{-8}$	$693{\cdot}2^{-8}$
					$m-1$	$105{\cdot}2^{-9}$	$315{\cdot}2^{-9}$	$-1155{\cdot}2^{-9}$
					m	$-7{\cdot}2^{-9}$	$63{\cdot}2^{-9}$	$693{\cdot}2^{-9}$

Lower order boundary stencil The interpolating boundary stencil keeps the filter size the same and shifts the filter. One other manner to deal with the boundary is by keeping a symmetric filter, this is done by using a lower order polynomial. The closer the wavelet or scaling function is to the boundary, the lower the order of this function. Implementing the lower order boundary stencil is easy. It is not needed to construct different filters, the wavelet and scaling functions closer to the boundaries simply use the internal predict filters of lower order.

Comparison of the boundaries The downside of the lower order boundary stencil is the loss of precision near the boundary. In the case of the interpolating boundary stencil the filter coefficients can be very large compared to the internal filter coefficients. Hence any perturbations from local polynomial behaviour near the boundary can lead to large errors. Furthermore, at lower levels the boundary stencil extends far into the interior of the domain, and hence smoothing coefficients relatively far from the boundary will influence the predicted values, which can lead to higher detail coefficients (poor predictions) near the boundary.

Multiple dimensions Extension to multiple dimension is done by a tensor product. Firstly, a step in the forward transform is done in dimension 1, followed by a forward step in dimension 2. This is repeated until dimension n is reached. For the inverse transform the dimensions should be traversed in the other direction, starting by dimension n ending at dimension 1.

3 Adaptive Mesh Refinement

A mesh adaptation algorithm for wavelets was introduced in [15] and we essentially use the same approach. However, as will be discussed later on, we examine different ways of thresholding, adding neighbours, and calculating the inverse transform. These changes were made after testing adaptive mesh refinement on the different data sets.

Wavelets are usually defined in a dyadic setting, hence a dyadic mesh is needed. In the case of second generation wavelets the mesh is not restricted to be dyadic, however in this paper a dyadic mesh is always chosen. This implies we can use the same predict and update filters on all the internal points of the meshes, which can lead to more stable behaviour (see [9]).

Mesh setup The wavelet system has a set of nested spaces $V_{J_1} \subseteq V_{J_1+1} \subseteq \cdots \subseteq V_{J_2}$, where J_1 corresponds to the coarsest level and J_2 to the densest level. If we relate this to meshes, V_{J_1} corresponds to the coarsest mesh and V_{J_2} to the finest mesh considered. The different spaces V_j are defined to be related through dyadic scaling. Therefore, the different meshes, each corresponding to a V_j, considered will be set up to be dyadic.

There are $J_2 - J_1$ evenly spaced meshes G^j, where G^j exists out of 2^j points for $J_1 \leq j \leq J_2$, i.e.

$$G^j = \{x_{j,k} : 0 \leq k \leq 2^j - 1\},$$

where j indicates the level and k the spatial location. The mesh is dyadic, hence a 1D grid point $x_{j,k} \in G^j$ corresponds to $x_{j+1,2k}$ in G^{j+1} for $0 \leq k \leq 2^j - 1$. So the meshes are nested, i.e. $G^j \subset G^{j+1}$, and G^{j+1} restricted to the points with even k value gives G^j.

Extending this definition to multiple dimensions is simply done by the tensor product, so in the 2D setting the squared mesh G^j looks like

$$G^j = \{x_{j,k,l} : 0 \leq k \leq 2^j - 1, 0 \leq l \leq 2^j - 1\}.$$

Note this definition can be extended to enable non-square meshes.

Thresholding After the forward transform, there remain relatively few smoothing coefficients and a lot of detail coefficients. The idea is to keep all points corresponding to smoothing coefficients in the mesh, and add the points corresponding to detail coefficients with a high absolute value.

The absolute value of a detail coefficient indicates how well that mesh point can be estimated using the wavelet transform. If the absolute value is low then the wavelet can easily approximate the function value as some polynomial passing through the neighbouring points. In this case the

point need not be kept, as long as some neighbouring points are kept. However, if the absolute value is high then the wavelet cannot accurately approximate the function value and thus the mesh point should be kept. The approximation of a function using the wavelet transform can be written as

$$\begin{aligned} f(x) \approx P_{V_{J_2}} f(x) &= \sum_k s_{J_2,k} \varphi_{J_2,k}(x) \\ &= P_{V_{J_1}} f(x) + \sum_{j=J_1}^{J_2-1} P_{W_j} f(x) = \sum_k s_{J_1,k} \varphi_{J_1,k}(x) + \sum_{j=J_1}^{J_2-1} \sum_m d_{j,m} \psi_{j,m}(x). \end{aligned}$$

Every point in the dense mesh corresponds to a smoothing coefficient $s_{J2,k}$ in the densest level. In another representation, all points in the mesh correspond to either a smoothing coefficient $s_{J_1,k}$ in the coarsest level or a detail coefficient $d_{j,m}$ in any other level. Note that if the absolute value of the detail coefficient is small, the term does not influence the approximation of f significantly and therefore the point can be dropped from the mesh. If the value is high, though, the original smoothing coefficient is difficult to determine and thus the corresponding detail coefficient will influence the approximation of f significantly.

Adjacent points If a point $x_{j,k}$ is added to the mesh then it means the absolute value of the detail coefficient at that point is large. The mesh will be used to evolve a PDE for one or more time steps. Mesh adaptation incurs some overheads, so it is common to not adapt the mesh at every time step. It therefore makes sense that the mesh should consist of those points associated with detail coefficients which are currently large or *could possibly become large during the period of time when the mesh remains unchanged.* Therefore, adding points in the *adjacent zone* of the grid points with high detail coefficients is needed, as over time these detail coefficients could become significant. There are two types of neighbouring points considered. Type 1 refers to adding neighbours only in the direction in which the coefficient operates as a detail coefficient. Commonly in the case of two or more dimensions, points which have been thresholded are detail coefficients in all dimensions but in different levels j. In the 1D case, type 1 points in the adjacent zone of $x_{j,k}$ are the points $x_{j',k'}$ such that:

$$|j - j'| \leq L^{\text{level}} \qquad |2^{j-j'}k - k'| \leq L^{\text{neighbour}},$$

where L^{level} are the number of levels in which adjacent points are added and $L^{\text{neighbour}}$ are the number of adjacent points added per level per direction. So if $L^{\text{level}} = 1$ and $L^{\text{neighbour}} = 1$ then 1 neighbour to the right and 1 to the left of the detail coefficient will be added in the level below, the same level and the level above.

A second way of adding neighbours is called type 2. In this case not 1 but in all directions in that level neighbours are added. In other words, the M closest points in level j are added. In both types of neighbouring points, one can also add points in the neighbouring levels.

Perfect reconstruction The points which have been added up until now are considered as important. Because the points are significant, one wants to be able to fully construct the values of these points using the inverse wavelet transform. However, some of the required values might be lost during the thresholding phase. Hence, the points to perfectly reconstruct the mesh points currently in the mesh need to be added. These points are called *perfect reconstruction points.* The particular points which need to be added in this phase depend on the inverse transform, i.e., they are dependent on the used wavelet transform. Because the added points should also be reconstructed perfectly, one should note that the perfect reconstruction check should be repeated until no new points are added.

For all detail points in the mesh the N neighbouring smoothing coefficients need to be added. These neighbouring smoothing coefficients correspond to detail coefficients in the lower levels. The added smoothing coefficients themselves are constructed by neighbouring detail coefficients, however we assume that if one of these detail coefficients is not already in the mesh, the value is small so it does not affect the value of the smoothing coefficient significantly. If these detail coefficients are also added the mesh will end up full. Because both Donoho and lifted Donoho have the same predict filter, the perfect reconstruction of both wavelets is the same. For significant $d_{j,k}$

the points $s_{j,k+n}$ need to be added, where $-N/2+1 \leq n \leq N/2$ with N the number of primal vanishing moments.

Adaptive forward transform Normally, the forward transform is applied to all points in the mesh. However, not all the values of the smoothing coefficients are known, because the mesh is sparse. Due to the perfect reconstruction, all smoothing coefficients needed to calculate the significant detail coefficients of the mesh points are in the mesh. Firstly, it is checked if a point which is handled as a detail coefficient is in the mesh, if not it will be set to zero. If a smoothing coefficient is not in the mesh no update needs to be done. If it is in the mesh the update step can proceed normally, if a detail coefficient is used which is not in the mesh this detail coefficient will be treated as a zero. This is because in the thresholding phase, detail coefficients close to zero were removed.

Adaptive inverse transform In order to reconstruct the full function after the forward transform the inverse transform is performed. All the thresholded detail coefficients are very close to zero, therefore, it is standard to use the normal inverse transform with setting the thresholded values equal to zero.

However, during testing, it was noticed that the lifted Donoho wavelet had a larger error than the Donoho wavelet for some 2D data sets. A detailed examination revealed that in the last update step of the inverse transform, the update was different from the corresponding update step in the adapted forward transform.

In the inverse transform one wants to reform the original function f, and this is done by undoing the operations of the forward transform. Hence, it makes sense to create the same environment as in the last performed forward transform. This adapted forward transform uses zeros for the update step if detail coefficients are used which are not in the mesh. In the inverse transform all coefficients will receive a value, hence the update step is different. Therefore, in the adapted setting one can choose to change the inverse transform. During the update step it is checked if a detail coefficient is in the mesh. If not, this detail coefficient contributes the value of zero to the update step instead of its own value. The mesh used to restrict the inverse transform is not the most recently generated, but the mesh of the time step before the one which was used in the forward transform. Hence, for the adaptive inverse transform one needs access to the previous adapted mesh, for example by storing it in memory. During testing the adaptive inverse transform is compared to the normal inverse transform.

Adaptive thresholding The thresholding of points is connected to the adjacent points, namely, only for thresholded points are adjacent points added (these can be either type 1 or type 2 points). This in a sense broadens the way in which thresholding can be done. For example, if a detail coefficient is high it can be considered as difficult to predict, so it might make sense to add more neighbours for this kind of points. Below three types of thresholding are described.

version 1 This version is the standard setting. There is one threshold value ε and for all thresholded points, adjacent points of type 1 will be added. The adjacent points in the current level, level higher and level lower will be added. Furthermore, in these levels only one neighbour on each side is added.

version 2 In version 2 two values for thresholding $\varepsilon_1 < \varepsilon_2$ are used and again type 1 adjacent points are considered. The motivation behind two levels of thresholding, is that if a certain detail coefficient has a very high absolute value then more neighbours might need to be added. For the very high detail coefficients the two nearest points on both sides can be considered in two levels above and below the current level. For the standard thresholded points, just one adjacent point on either side is added in the current level, level above and below. During testing the results of version 1 and version 2 were similar. Therefore, version 2 will not be mentioned in the results.

version 3 Some of the financial PDEs which were tested produce surfaces with a high peak which spreads out quickly. In this case a more extreme adding of neighbours might be considered.

Setting the threshold value dependent on the data gives an indication of the relative difficulty in predicting the value. A first threshold ε_1 is set as in version 1 above, but then a second threshold $\varepsilon_1^{\text{threshold}} = \frac{|\max(f)-\min(f)|}{4} > \varepsilon_1$ is set at which type 2 adjacent points are added (the value 4 was chosen after testing the result of different values). So in 2D a square of points will be added around the thresholded detail coefficient if it is larger than ε_2. In the 3D setting this will be a cube of neighbours.

In algorithm 3 the steps of the adaptive mesh refinement are given.

Algorithm 3 AMR for time dependent PDE

t=0
$M_t = \{\}$ and denotes the adapted sparse mesh at each time point
Perform the forward transform on smoothing coefficients at level J_2.
    Add the points corresponding to s_{J_1} to M_0.
    Add thresholded points to M_0.
    Add adjacent points to M_0.
    Add perfect reconstruction points to M_0.
    Solve time step(s) on refined mesh M_0.
for t=1 upto T **do**
    Perform the adapted forward transform on the restricted grid $s_{J_2} = f(G^{J_2}|_{M_{t-1}})$.
    Add the points corresponding to s_{J_1} to M_t.
    Add thresholded points to M_t.
    Add adjacent points to M_t.
    Add perfect reconstruction points to M_t.
    Solve time step(s) on refined mesh M_t.
end for
Perform the (adapted) inverse transform on refined mesh to get the solution interpolated onto the full mesh.

4 Results

The wavelet adaptive mesh refinement was tested on two different classes of PDEs[1]. The first class relates to common problems in finance: pricing and calibrating for 1D local volatility and 2D stochastic local volatility (SLV) type models. Depending on input parameters and boundary conditions a range of different shapes are obtained. The second class of PDEs relates to a 2D and 3D fluid transport problem in CFD. The dependent variables pressure, energy and density are characterised by shocks and sharp gradients which travel through time.

The PDEs have been approximated numerically on dense meshes to produce accurate reference results at each time step. The wavelet AMR starts at the initial solution and then restricts the subsequent solution surfaces to the newest mesh. At every time step the values on the sparse mesh are interpolated with the (adaptive) inverse transform onto the dense mesh, after which the result is checked against the reference solution. This gives the ability to evaluate the performance of the wavelets at every time step. Only part of the test data is presented here. For more results see [30].

Two types of wavelets are tested, Donoho's interpolating wavelet and lifted Donoho's interpolating wavelet. Both wavelets are tested with two different boundary stencil implementations, lower order and interpolating boundaries. Furthermore, two wavelet orders will be compared, $N = 4$ and $N = 6$. In the 2D tests, $J_2 = 8$ was assumed to be the finest level of refinement and $J_1 = 4$ the coarsest level. For the 3D test sets $J_2 = 7$ was taken as the finest level of refinement and $J_1 = 3$ the coarsest level.

The previously described version 1 and 3 of thresholding and adding adjacent points will be used. For version 3, the high threshold adds five direct neighbours in all directions in the current level, the level above and below.

In order to indicate which wavelet is tested the following notation is used: Wavelet(primal order N, dual order $\tilde{N}$, boundary). *Don* indicates Donoho's interpolating wavelet, *Swel* the lifted

[1]Provided by the Numerical Algorithms Group.

Donoho's interpolating wavelet, *int* the interpolating boundary stencil and *low* the lower order boundary stencil, for example Don(6,0,int) or Swel(4,4,low).

The sparse representation is extended onto the fine mesh using the (adaptive) inverse transform, i.e. wavelet interpolation. Since the sparse representation must be formed from thresholding on an adaptive mesh and the wavelet interpolation is not exact, we measure the error that results when the sparse data is interpolated back onto the fine mesh.

Accuracy measure During testing the relative Frobenius error has been used to measure accuracy. This relative error is calculated as follows. Denote by f the reference solution computed on the dense mesh and let f^ε be the approximation generated by the wavelets (inverse or adaptive inverse wavelet transform). The relative Frobenius error is then defined as

$$\frac{||f - f^\varepsilon||_F}{||f||_F}$$

where we set $||f||_F = \sqrt{\sum_{i,j,k} f(i,j,k)^2}$ for a 3D discrete function f and $||f||_F = \sqrt{\sum_{i,j} f(i,j)^2}$ for a 2D discrete function.

Finance data sets It turned out that all the wavelets performed equally well on the 1D finance problems. For brevity the results will not be presented here.

Five different 2D data sets were studied all closely related to the stochastic local volatility model (SLV) with Heston volatility dynamics. The SLV model is widely used in finance, especially in foreign exchange markets. The model has to be calibrated to market data, and a key stage in this calibration is solving for the probability density function (PDF) of the process [31]. The five datasets are:

- SLV5: Calibrated PDF for parameter set 5 in [32] table 603, the Feller constant is 0.78.
- SLV17: Calibrated PDF for parameter set 17 in [32] table 603, the Feller constant is 0.35.
- SLV53: Calibrated PDF for parameter set 53 in [32] table 603, the Feller constant is 1.13, but the correlation is 76%.
- Digital: Price of a digital call option driven by SLV.
- Call: Price of a call option driven by SLV.

PDFs with the Feller constant less than one have very high peaks near the zero boundary. The SLV53 is strongly skewed due to the high correlation. The "digital" problem has a discontinuous initial condition which is smoothed out (highly non-linearly) in time. The "call" problem is very smooth - compression rates around 1% were achieved. For the data set "digital", it was very difficult to get high precision. This was solved when the version 3 AMR was used. It was notable that the interpolating boundary stencil version of lifted Donoho wavelet was very precise, but added a lot of points. Furthermore, for both data sets Donoho's interpolating wavelet with the lower order boundary stencil had both a good compression rate and a small relative error.

Initially all the wavelets performed similarly for the three different SLV sets. Notably, the lifted version of Donoho's wavelet had poor performance. This changed when the adaptive inverse transform was used. Donoho's wavelet outperformed the lifted version of Donoho's wavelet mostly in compression rates.

We will examine SLV17 more closely since the SLV data sets presented more difficulties for the wavelets than the "call" and "digital" sets, due to the fast changing surface.

In Figure 1 SLV 17 is shown at time step 0 (the initial solution), time step 26 and the final time step 52 (note the change in scale of the Z axis). The approximation starts off with a high peak on one point, the rest of the approximation is zero. This peak is located near the boundary, which makes the data set interesting as nonsmoothness near the boundary is more difficult for wavelets as there are no data points available beyond the boundary. The high peak spreads out through time.

Figure 1: SLV 17 through time.

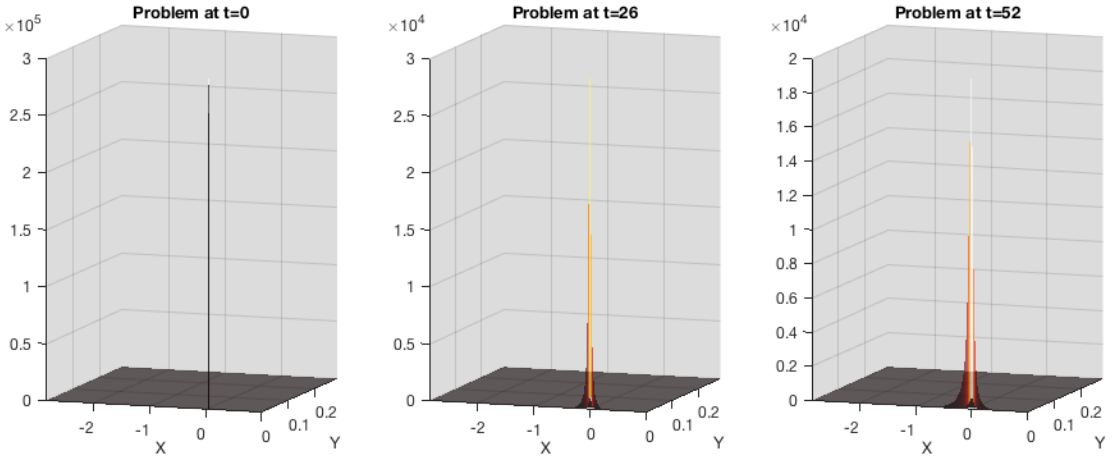
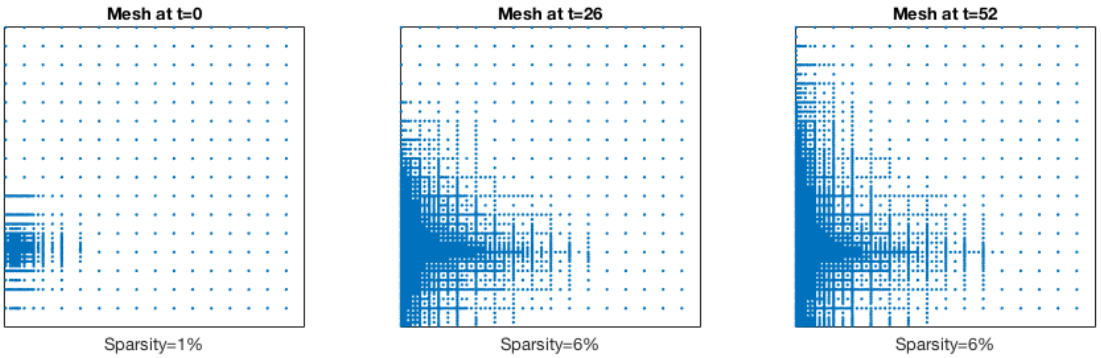


Figure 2: Meshes of SLV 17 using Don(4,0,low).

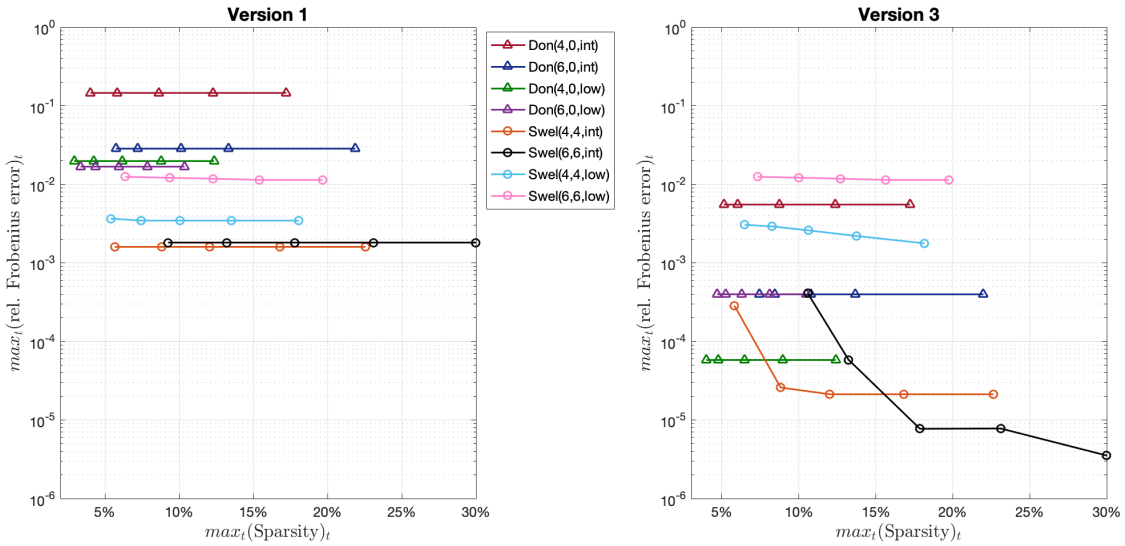


In Figure 2 three different meshes corresponding to time steps 0, 26 and 52 are shown. The wavelet used in this AMR is Donoho's interpolating wavelet with the lower order boundary stencil, primal order $N = 4$ and dual order $\tilde{N} = 0$. Version 1 of AMR is used, this means that there is one threshold value, namely 0.001. The figure captions shows that AMR is capable of refining the mesh in case the nonsmooth area increases in time. In the figure percentages are shown. This is the amount of points kept compared to the dense mesh on which the reference solutions were computed. For J_2 being the densest level and d the number of dimensions, the sparsity is determined by:

$$\frac{\text{number of points kept}}{(2^{J_2})^d} \times 100.$$

In order to visualise the performance of all the different wavelets, 5 tests have been compared. Every wavelet is tested on the data for 2 different versions of the AMR, version 1 and version 3, varying the threshold value 5 times, $\varepsilon^{\text{threshold}} = [10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}]$. After each test the maximum number of points and the maximum error over all time steps are kept. This results in the plots of Figure 3. Version 1 and version 3 of AMR are evaluated. The figure shows that in the first version of AMR all wavelets have difficulties in reducing the error, whilst this error is significantly reduced in version 3 of AMR. Moreover, reducing the threshold results in more points but not a higher accuracy for most wavelets, which is surprising. One would expect the wavelet approximation to improve as the threshold is lowered and the number of points is increased. To investigate this further it is convenient to observe the error at every time step, see Figure 4. There are some remarkable features to be observed in this figure. Recall that we are tracking the maximal error reached, not the average error. The maximal error occurs in the first time steps. Thereafter the error of Donoho's interpolating wavelet appears constant at 10^{-15}. In reality it is not constant,

Figure 3: Comparing all the wavelets on 2D SLV 17.



however the error is so small that computing the Forbenius norm drives the calculations below machine precision.

Secondly, it appears that the maximum error is occurring in the second time step. SLV 17 starts with a sharp peak in one point. This leads to adding points around this point. However, not enough neighbours are added. Because the shape is so irregular at that location, the surface around it will change very fast, not only the closest neighbours will be effected. Version 3 of the AMR successfully overcomes this. In this setting two different threshold values are used. The first is static and is chosen beforehand, while the second is dynamic and depends on the current approximation. The dynamic threshold is relative to the approximation surface. If a point is thresholded by that value, a big square of points is added around the detail point. The bigger the square the more accurate the following time step can be approximated, however this leads to the addition of more points. The addition of points is not a problem as this occurs very infrequently, only taking place if there is a very high detail coefficient. This can be observed in Figure 5: during the first time steps, Donoho and lifted Donoho add more points than version 1 above (see Figure 4), but after only a few more time steps the sparsity of the two approaches is more or less equal.

Furthermore, lifted Donoho structurally uses more points than Donoho's interpolating wavelet. This is caused by the interpolating boundary stencil. In all test cases more points are added near the boundaries for the interpolating boundary stencil. The polynomials used near the boundaries are not good predictors for the function values there. In addition, when going to the lower levels, e.g. level J_1, points spatially far away from the boundary influence the boundary coefficients, because the filter uses N points on one side of the boundary coefficient and at level J_1 this can be a large distance. In the case of SLV17, lifted Donoho also adds more points because the peak is smoothed out so it will cover a larger area. This smoothing is caused by the update step, a procedure which smooths out the irregular peaks and causes a larger area to have high detail coefficients, even though the highest detail coefficient has a smaller absolute value. One beneficial aspect of this smoothing is that lifted Donoho wavelet has less problems to adapt in the second time step, mainly because the mesh is already less sparse.

However, although the interpolating boundary stencil is adding more points it works better than the lower order boundary stencil for lifted Donoho wavelet. The lower order implementation in lifted Donoho case is quite erratic, sometimes the accuracy is good and other times results are very inaccurate. This is a prolem with the lifted Donoho wavelet and not for Donoho's interpolating wavelet.

The lifted Donoho wavelet has to use the boundary stencil implementation both for the smoothing coefficients (update step) and for the detail coefficients, whereas Donoho's wavelet only needs

Figure 4: Version 1, error and sparsity for 2D SLV 17.

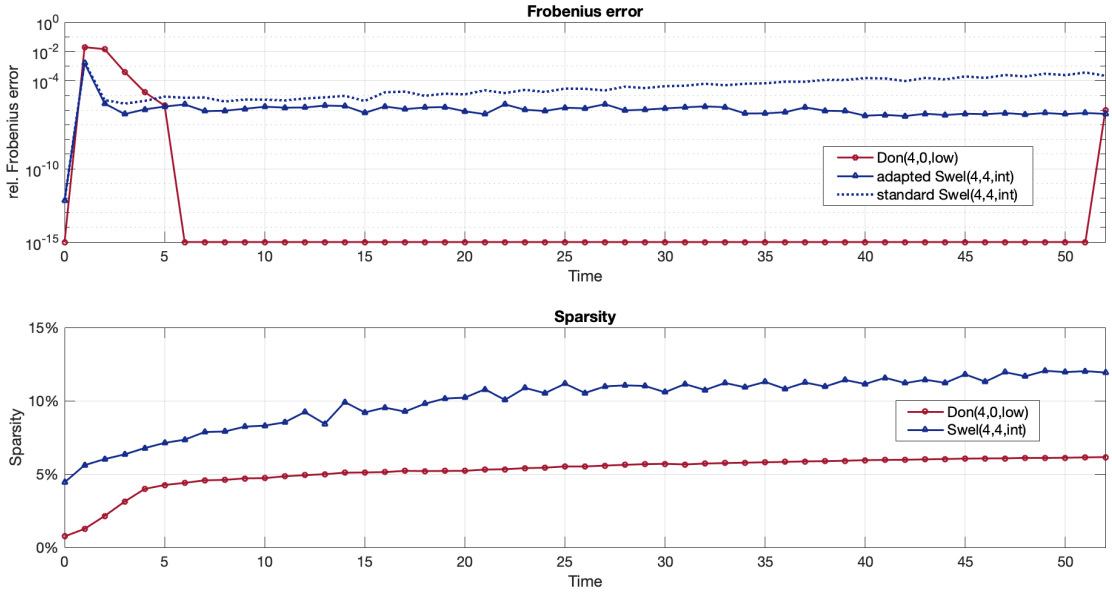
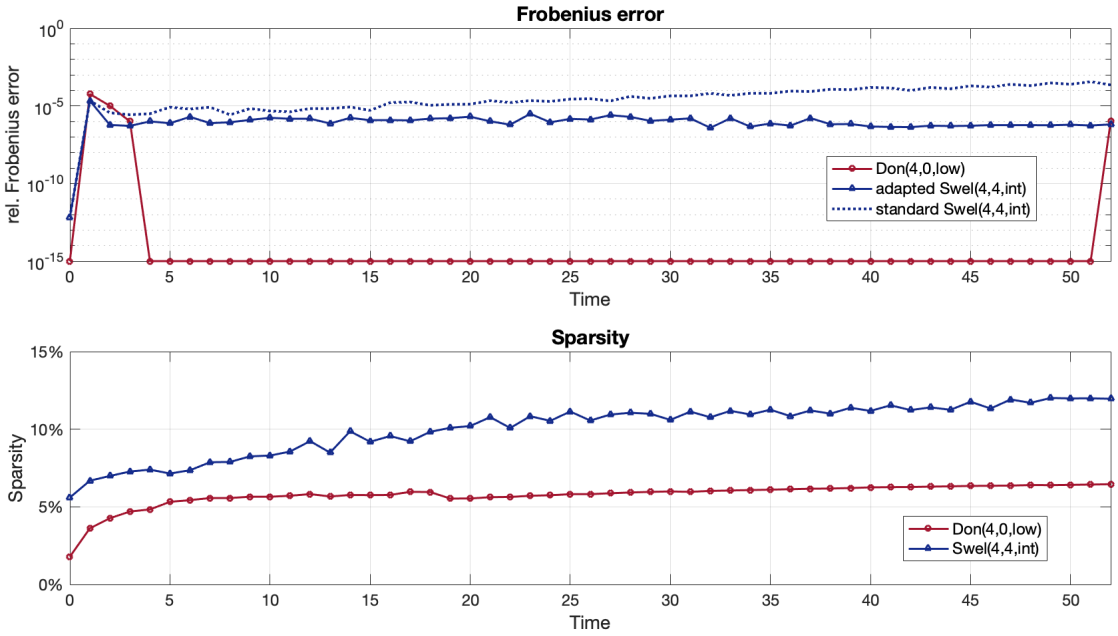


Figure 5: Version 3, error and sparsity for 2D SLV 17.



to deal with boundary points for the detail coefficients. The Donoho's wavelet only has a predict step:

$$d_{j-1,m} = \frac{1}{2} d_{j-1,m} - \sum_{k=-N}^{N} p_k^N s_{j-1,m+k}$$

whilst the lifted version also has an update step, for which boundary points need special care:

$$\begin{cases} d_{j-1,m} = \frac{1}{2} d_{j-1,m} - \sum_{k=-N}^{N} p_k^N s_{j-1,m+k} \\ s_{j-1,k} = s_{j-1,k} + 2 \sum_{m=\tilde{N}}^{\tilde{N}} p_{-m}^{\tilde{N}} d_{j-1,k+m} \end{cases}$$

Moreover, the update step works as a smoother, i.e., it averages the smoothing coefficients. Reducing the order near the boundary can cause an error. This error results in a high detail coefficient, the high detail coefficient is then affecting the surrounding smoothing coefficients in the update step. Hence, due to this smoothing this error will be transported through the grid.

Finally, in the error plots two different graphs for the lifted Donoho wavelet are depicted. In testing we discovered that high errors for the lifted Donoho wavelet are caused by the update step in the inverse transform. In this update step some detail coefficients gave contributions, whilst they where not involved in the adaptive forward transform. They have contributions in the normal inverse transform, because all the coefficients are filled even if they are not in the mesh. But it is exactly the coefficients which did not have a contribution in the forward transform, but now do in the inverse transform, that are causing the problems. Therefore, we made an adapted version of the inverse transform, where detail coefficients only have a contribution in the update step if they were included in the previous mesh. This leads to only changing the decompression/interpolation, so both versions have the same meshes.

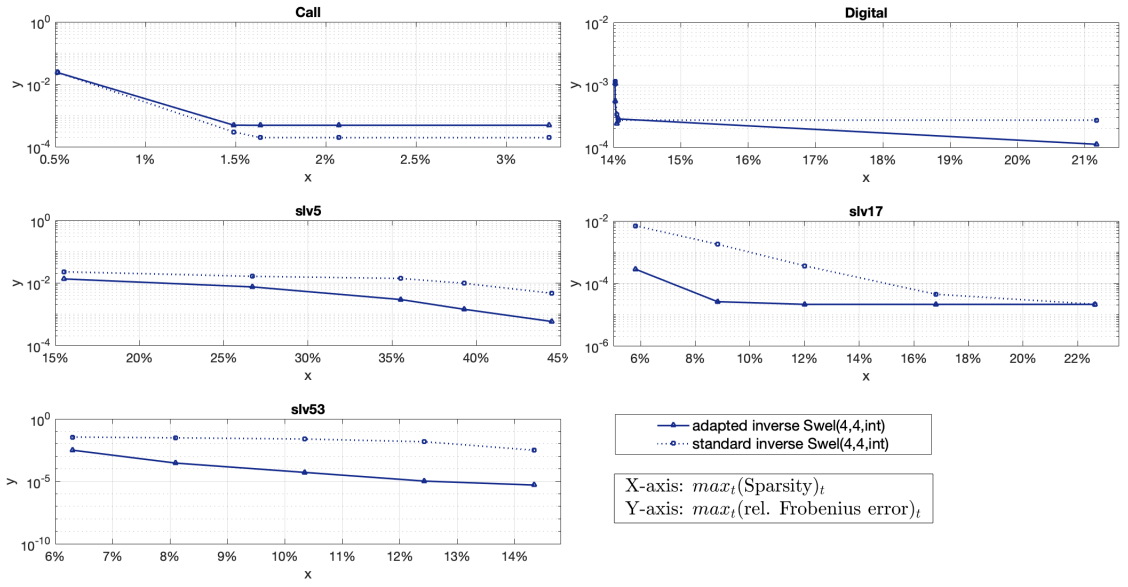
The adapted inverse transform is not specifically designed for the lifted version of Donoho's wavelet, it is for any second generation wavelet with an update step. In Figure 6 the results for Swel(4,4,int) on all 2D finance data sets are given. Both the standard and adapted inverse transform are calculated for each threshold. Major improvements are observed in the SLV sets. Especially for SLV 53 the reduction in error is between 100x and 1000x. In the case of SLV17 it is interesting to note that the adapted inverse has an accuracy better than 10^{-4} with a sparsity of 7%. Meanwhile, the standard inverse transform needs around 17% of the grid points to reach that accuracy. There is almost no difference in the "call" and "digital" data sets. For "call" the error is slightly worse for the adapted inverse transform. Both versions of the inverse transform are capable of keeping the error less than 10^{-3} whilst maintaining a high sparsity. For the other versions of the lifted Donoho wavelets, we encountered similar results, for brevity they are not included in this paper, for more information see [30].

Compressible CFD problem The simulation solves the equations of compressible Navier–Stokes assuming an ideal gas equation of state using a finite volume discretisation. The method uses the MUSCL technique for limiting the slopes of the primitive variables and computing the left and right states at the interfaces between cells. An approximate Riemann solver is used to compute the numerical fluxes at the interfaces between cells. For more information, see [33]. The PDE computes pressure, which depends on the density, energy, x velocity, y velocity and z velocity (in 3D). For the most part we will focus our discussion on the pressure. For a detailed treatment of the other terms, please see [30].

In Figure 7 the pressure surface and heat plots are given at time steps 0, 257 and 514. The solution starts of with a local blob, which results in a wave that interacts with the boundaries. In Figure 8 the meshes at these time steps are shown. Donoho's lower order wavelet is used to generate the meshes. It is interesting to compare the heat plots with the meshes. Wavelets are well known for their edge detection skills in image compression. That ability is observed in the mesh, where points are added around the shock waves. Furthermore, although the shocks are spread over the whole domain the wavelet is able to generate a sparse grid with only 19% of the points kept.

All wavelets are tested on pressure for version 1 and 3 of AMR. The results are very similar. In Figure 9 the results of version 1 are shown. AMR is often only considered worthwhile in 2D if

Figure 6: Comparing adapted and standard lifted Donoho on the 2D data sets price, digital, SLV5, SLV17 and SLV53.



fewer than 40% of the points are kept[2]. Only Donoho's interpolating wavelet with the lower order boundary stencil implementation is capable of having this sparsity whilst having an error around 10^{-3}.

In Figure 10 the Frobenius error and sparsity of pressure through time are given. The threshold value is 10^{-3}. The error plot shows that the error of Donoho's wavelet is lower. Furthermore, the standard inverse transform for lifted Donoho leads to a larger error than using the adapted inverse, although the maximal relative Frobenius errors are similar.

The performance of the adapted inverse transform on all data sets for the lifted Donoho wavelet is given in Figure 11. Although the differences are not big, the adapted version of the inverse transform consistently results in a lower Frobenius error. For the other versions of the lifted Donoho wavelets, we encountered similar results, for brevity they are not included in this paper, for more information see [30].

The 3D CFD solution slices look very similar to the 2D data. In Figure 12 the overall performance of the wavelets is shown. Only the results of version 1 are shown as they are very similar to the results of version 3. When comparing the wavelets it is important to keep in mind that in 3D, AMR is often only considered beneficial if a sparsity of 30% or less is achieved[3]. Like before, only Donoho's interpolating wavelet with the lower order boundary stencil is able to reach this sparsity with a small Frobenius error.

In Figure 13, Don(4,0,low) and Swel(4,4,int) are plotted for every time step. The errors of Donoho and lifted Donoho are very similar, except the normal inverse of lifted Donoho is giving a higher error. Although the error behaves similar for Donoho and lifted Donoho, the number of points needed to achieve this error is different. Lifted Donoho is adding many more points.

5 Conclusion and Discussion

Donoho's interpolating wavelet and the lifted version of Donoho's interpolating wavelet are both used in the literature for adaptive mesh refinement. However, it is not clear which wavelet is the best choice. Therefore, we tested various versions of these wavelets to investigate which wavelet has the most stable results and performance in adaptive mesh refinement.

[2]This was a rule of thumb that emerged from work on the Paramesh package at NASA. Private communication from Dr Kevin Olson, Paramesh co-author.

[3]This was a rule of thumb that emerged from work on the Paramesh package at NASA. Private communication from Dr Kevin Olson, Paramesh co-author.

Figure 7: 2D pressure through time.

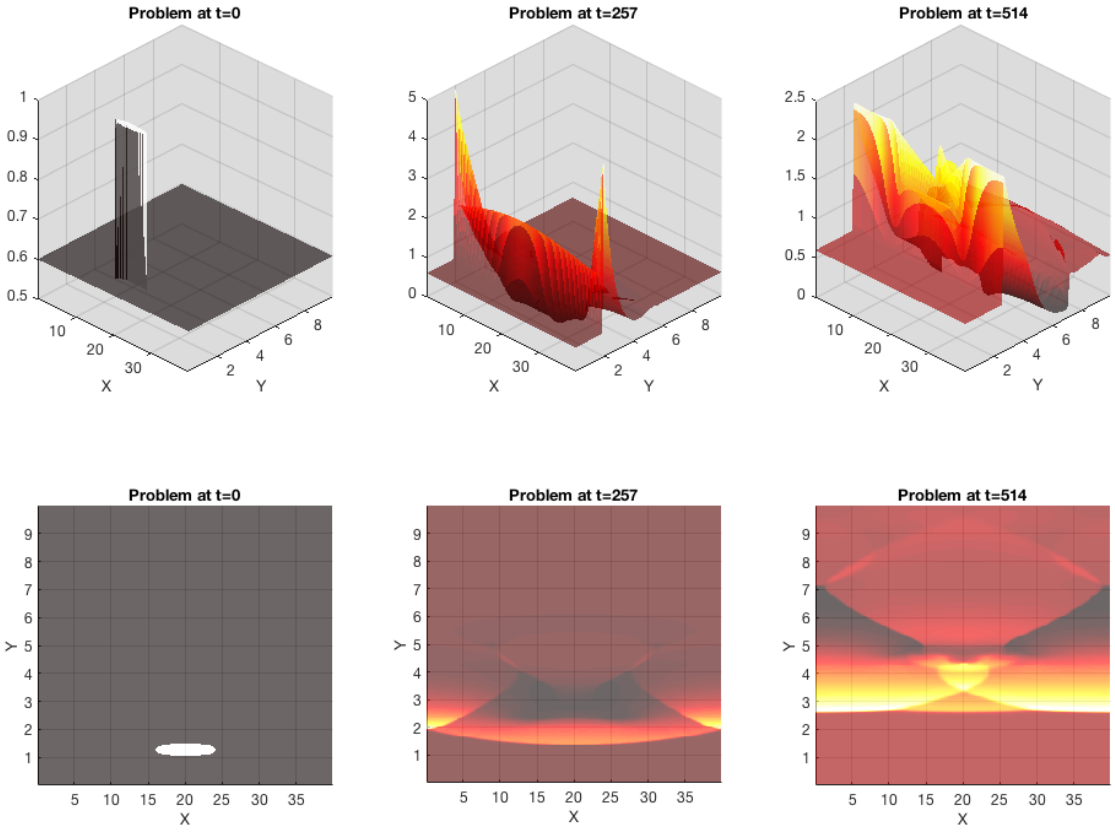


Figure 8: Meshes of 2D pressure using Don(4,0,low).

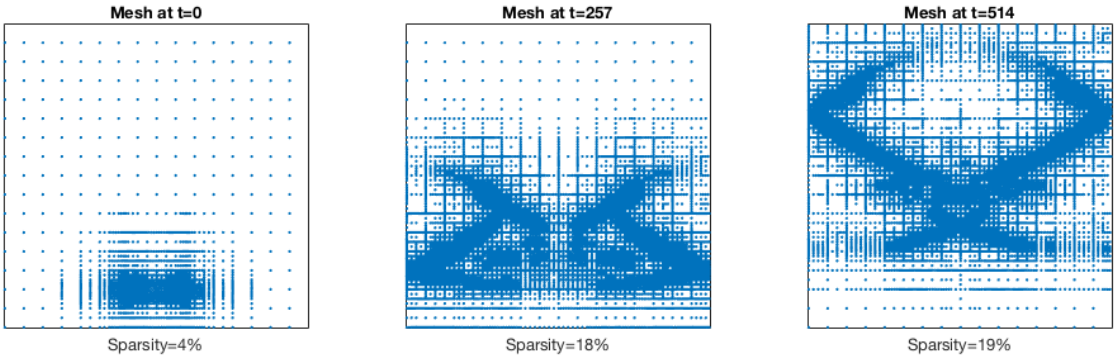


Figure 9: Comparing all the wavelets on 2D pressure.

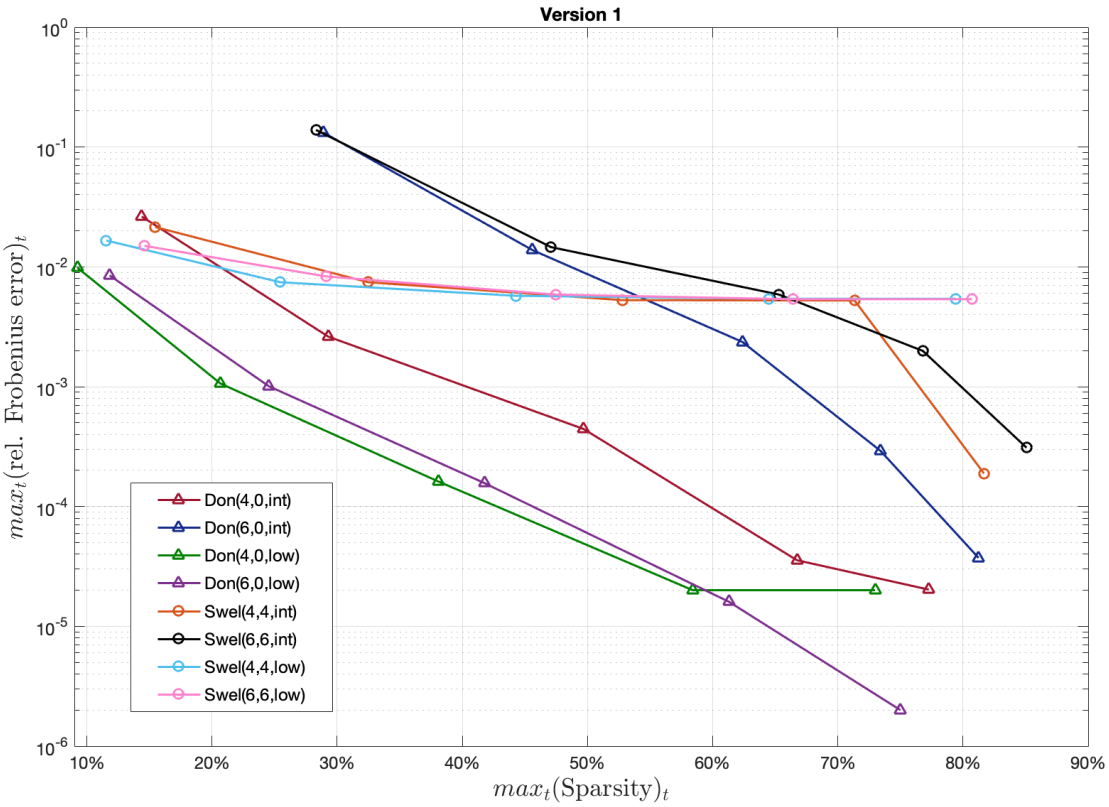


Figure 10: Error and sparsity for 2D pressure with threshold 10^{-3}.

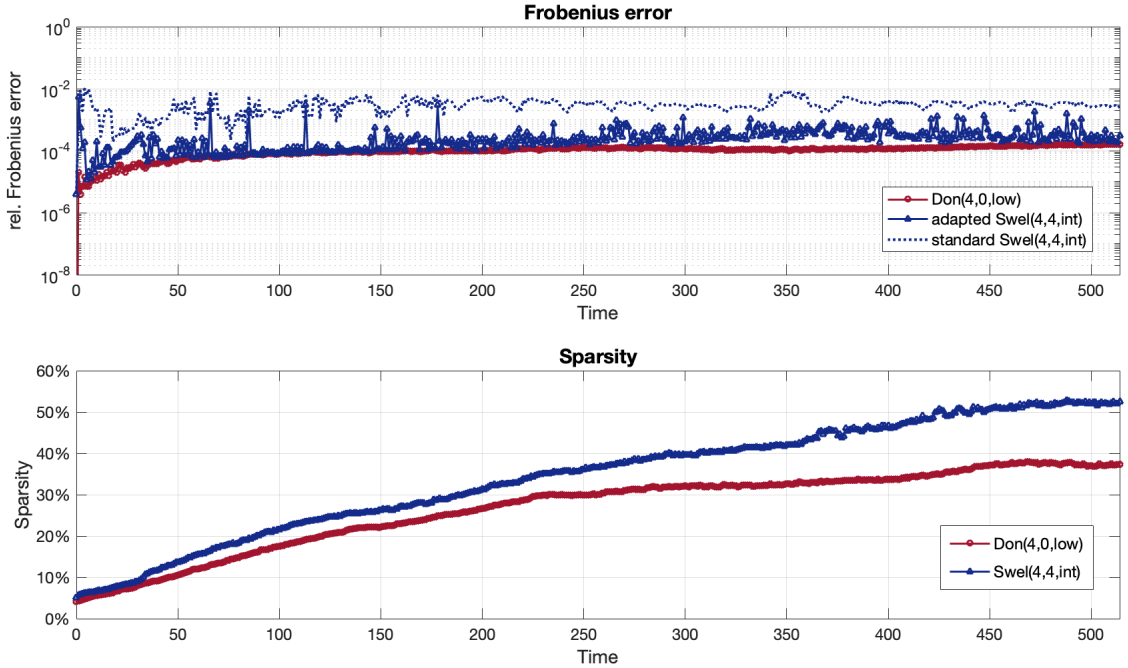


Figure 11: Comparing adapted and standard lifted Donoho on 2D CFD data.

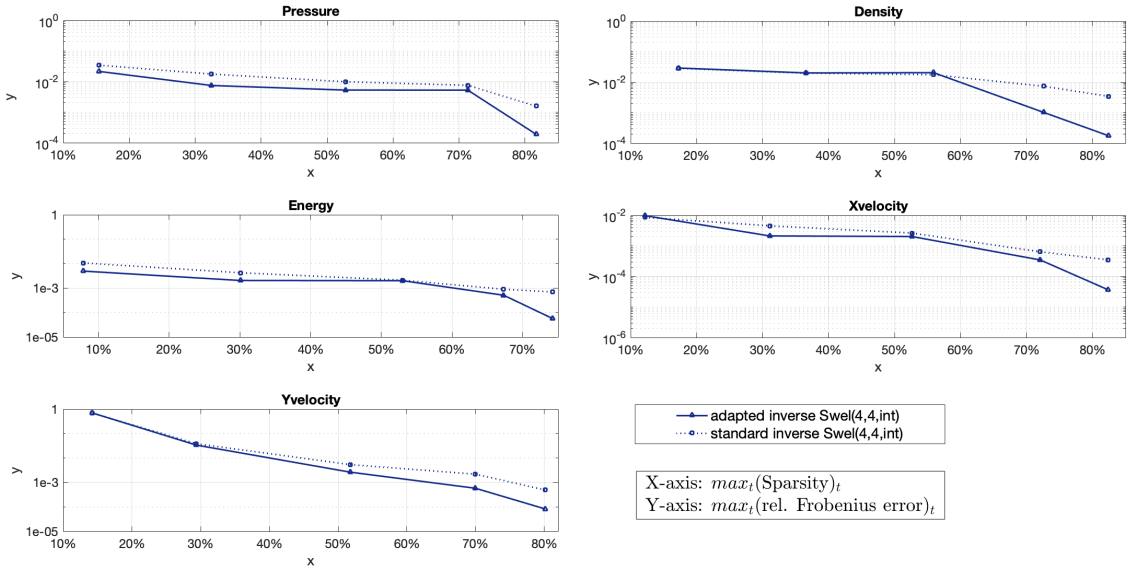


Figure 12: Comparing all the wavelets on 3D pressure.

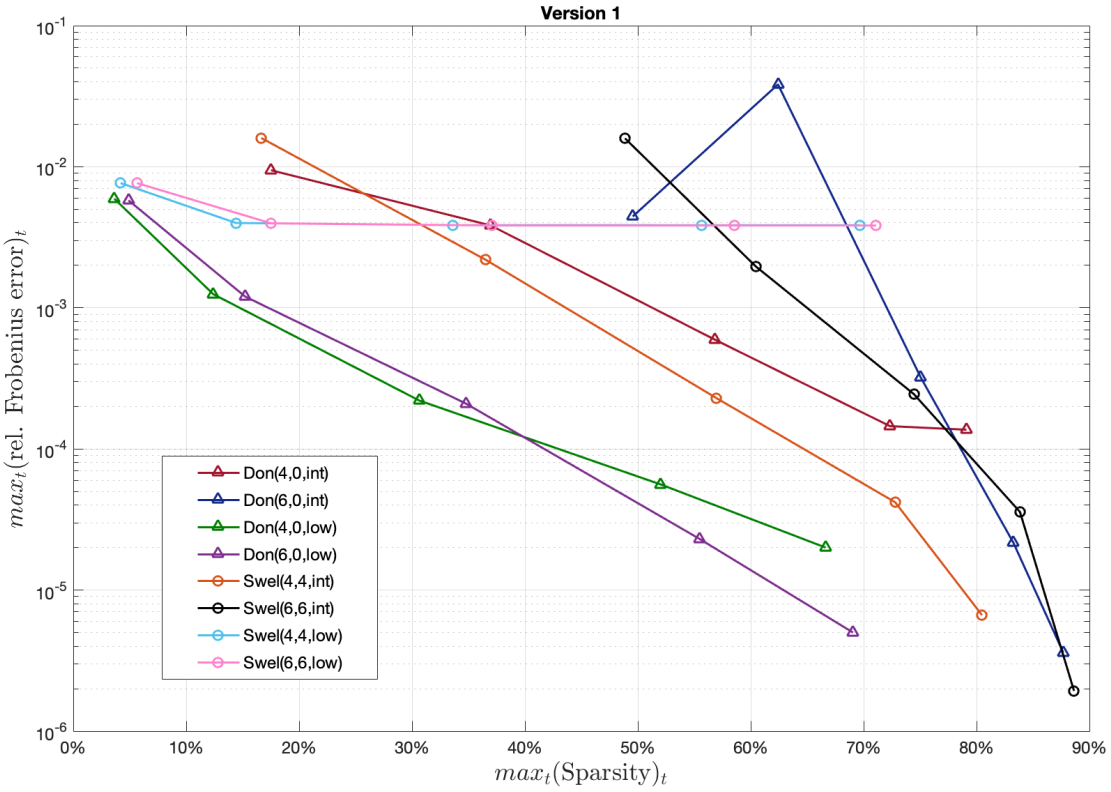
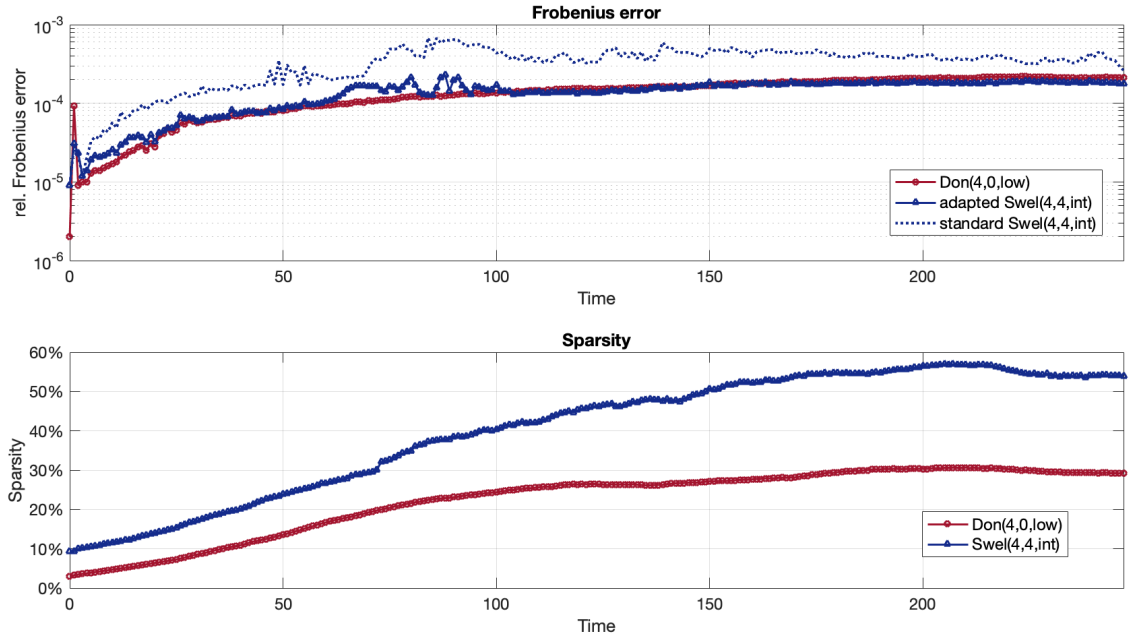


Figure 13: Performance per time step on 3D pressure with threshold 10^{-3}.



We tested different boundary stencil implementations, different versions of the AMR algorithm and different orders (N). The tested orders are 4 and 6. Although $N = 6$ outperforms $N = 4$ sometimes, for all wavelets $N = 4$ ended up being the most stable choice.

Furthermore, for the lifted version of Donoho's interpolating wavelet the interpolating boundary stencil is the best boundary stencil implementation. It adds many points near the boundary due to the long tails of the boundary wavelets, but is accurate for all test problems.

The lower order boundary stencil turned out to be the best performing boundary stencil implementation for Donoho's interpolating wavelet. It is always accurate, whilst having very good sparsity. For example, for SLV 17 Donoho's interpolating wavelet kept less than 5% of the points with an error smaller than 10^{-4}, a 20 fold sparsification.

Comparing Donoho's wavelet and the lifted wavelet, we find that Donoho's interpolating wavelet with the lower order boundary stencil implementation is the best. The lifted Donoho's wavelet can compete with Donoho's interpolating wavelet on PDEs with very local behaviour, i.e. when irregular behaviour is very localised. However, when nonsmooth behaviour is occurring over the whole domain, the lifted Donoho's wavelet adds too many points. For example, for the 2D CFD problem, in order to achieve an error smaller than 10^{-3} Donoho's interpolating wavelet kept fewer than 20% of the points, while the lifted version of Donoho's interpolating wavelet needed 80% of the points. This is caused partly by the update step, which smooths out the function, hence local behaviour becomes less local as the irregular behaviour is covering a larger area. Another cause is the interpolating boundary stencil. Due to the long tail, on lower levels, behaviour further away from the boundary influences the boundary wavelets.
The lifted Donoho's wavelet is a second generation wavelet with an update step. During testing it is noted that using an adapted inverse transform reduces the error significantly. This change in the inverse transform is only for wavelets with an update step, hence it does not affect Donoho's interpolating wavelet.

Although the lifted version of Donoho's interpolating wavelet is presented in the literature as an improvement over Donoho's interpolating wavelet since it increases the dual vanishing moments, it does not appear to be an improvement for AMR. The update step causes the wavelet to predict the function values less accurately, which results in worse performance for AMR.

Concluding, throughout the different tests, Donoho's interpolating wavelet with the lower order boundary stencil is the most stable and best performing wavelet for equidistant meshes. This is remarkable as the lower order boundary stencil is not commonly used in the literature.

References

[1] A. Cohen, I. Daubechies, J. Feauveau, Biorthogonal bases of compactly supported wavelets, Communications on Pure and Applied Mathematics 45 (1992) 485–560.

[2] I. Daubechies, Orthonormal bases of compactly supported wavelets, Communication on Pure and Applied Mathematics 41 (1988) 909–996.

[3] A. Cohen, Wavelet Methods in Numerical Analysis, Elsevier Science B.V., 2000.

[4] D. L. Donoho, Interpolating wavelet transforms, Tech. Rep. 408, Department of Statistics, Stanford University (1992).

[5] I. Daubechies, W. Sweldens, Factoring wavelet transforms into lifting steps, Journal of Fourier Analysis and Applications 4 (1996) 247–269.

[6] W. Sweldens, The lifting scheme: A new philosophy in biorthogonal wavelet constructions, in: Wavelet Applications in Signal and Image Processing III, Vol. 2569, 1995.

[7] W. Sweldens, P. Schröder, Wavelets in the Geosciences, chapter: Building your own wavelets at home, Springer, Berlin, Heidelberg, 2000.

[8] W. Sweldens, The lifting scheme: A construction of second generation wavelets, Society for Industrial and Applied Mathematics 11.

[9] M. Jansen, P. Oonincx, Second Generation Wavelets and Applications, Springer London, 2005.

[10] W. Sweldens, The lifting scheme: A custom-design construction of biorthogonal wavelets, Applied and Computational Harmonic Analysis 3 (1996) 186–200.

[11] W. Sweldens, R. Piessens, Quadrature formulae and asymptotic error expansions for wavelet approximations of smooth functions., SIAM Journal on Numerical Analysis 31 (1994) 1240–1264.

[12] J. Tian, R. O. Wells, A remark on vanishing moments, Conference Record of The Thirtieth Asilomar Conference on Signals, Systems and Computers 2 (1996) 983–987.

[13] A. Harten, Adaptive multiresolution schemes for shock computations, Journal of Computational Physics 115 (1994) 319–338.

[14] J. Liandrat, P. Tchamitchian, Resolution of the 1D regularized burgers equation using a spatial wavelet approximation, Tech. Rep. 90-83, Nasa and ICASE (1990).

[15] O. V. Vasilyev, C. Bowman, Second generation wavelet collocation method for the solution of partial differential equations, Journal of Computational Physics 165 (2000) 660–693.

[16] M. Jansen, M. Malfait, A. Bultheel, Generalized cross validation for wavelet thresholding, Signal Processing 56 (1997) 33–44.

[17] J. Simoens, S. Vandewalle, On the stability of wavelet bases in the lifting scheme, Tech. Rep. TW 306, Department of Computer Science, K.U. Leuven (2000).

[18] D. Wirasaet, Numerical solutions of multi-dimensional partial differential equations using an adaptive wavelet method, Ph.D. thesis, University of Notre Dame (2007).

[19] S. Paolucci, Z. J. Zikoski, T. Grenga, WAMR: An adaptive wavelet method for simulation of compressible reacting flow. part 2. the parallel algorithm, Journal of Computational Physics 272 (2014) 842–864.

[20] S. Paolucci, Z. J. Zikoski, D. Wirasaet, WAMR: An adaptive wavelet method for simulation of compressible reacting flow. part 1. accuracy and efficiency of algorithm, Journal of Computational Physics 272 (2014) 814–841.

[21] Y. Rastigejev, S. Paolucci, Wavelet-based adaptive multiresolution computation of viscous reactive flows, International Journal for Numerical Methods in Fluids 52 (2006) 749–784.

[22] B. C. de Wiart, Wavelet optimised pde methods for financial derivatives, Ph.D. thesis, Centre for Financial Research, Judge Institute of Management and St John's College, University of Cambridge (2005).

[23] B. C. de Wiart, M. A. H. Dempster, Wavelet optimized valuation of financial derivatives, International Journal of Theoretical and Applied Finance 14 (2011) 1113–1137.

[24] E. B.-D. A. Nejadmalayeri, A. Vezolainen, O. V. Vasilyev, Parallel adaptive wavelet collocation method for PDEs, Journal of Computational Physics 298 (2015) 237–253.

[25] E. Brown-Dymkoski, O. V. Vasilyev, Adaptive-anisotropic wavelet collocation method on general curvilinear coordinate systems, Journal of Computational Physics 333 (2017) 414–426.

[26] D. Livescu, O. V. Vasilyev, Comprehensive numerical methodology for direct numerical simulations of compressible rayleigh-taylor instability, Journal of Computational Physics 313 (2016) 181–208.

[27] K. Schneider, O. V. Vasilyev, Wavelet methods in computational fluid dynamics, Annual Review of Fluid Mechanics 42 (2010) 473–503.

[28] O. V. Vasilyev, Solving multi-dimensional evolution problems with localized structures using second generation wavelets, International Journal of Computational Fluid Dynamics 17 (2003) 151–168.

[29] O. V. Vasilyev, N. K. R. Kevlahan, An adaptive multilevel wavelet collocation method for elliptic problems, Journal of Computational Physics 206 (2005) 412–431.

[30] J. Knipping, Wavelet-based adaptive mesh refinement, Master's thesis, Technical University of Berlin, Delft University of Technology (2018).

[31] M. Wyns, J. Du Toit, A finite volume – alternating direction implicit approach for the calibration of stochastic local volatility models, Int. J. Comput. Math. 94 (11) (2017) 2239–2267.

[32] I. J. Clark, Wavelet Methods in Numerical Analysis, Wiley Finance, 2011.

[33] B. Fryxell, K. Olson, P. Ricker, F. X. Timmes, M. Zingale, D. Q. Lamb, P. MacNeice, R. Rosner, J. W. Truran, H. Tufo, FLASH: An adaptive mesh hydrodynamics code for modeling astrophysical thermonuclear flashes, The Astrophysical Journal Supplement Series 131 (2000) 273–334.

6 Acknowledgment

We would like to thank Dr Kevin Olson for providing the reference solutions for the CFD problems and for helpful discussions.